Bachelor's thesis

Emilie Skalstad Skarbø
Linda Helen Sperre
Maria Osa Furmyr
Nicklas Høines Mellum

Mcon Thruster Simulator

IE303612 Bachelor thesis

May 2021

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

NTNU
Norwegian University of
Science and Technology

Bachelor's thesis

2021

NTNU
Norwegian University of
Science and Technology

Emilie Skalstad Skarbø
Linda Helen Sperre
Maria Osa Furmyr
Nicklas Høines Mellum

Mcon Thruster Simulator

IE303612 Bachelor thesis

Bachelor's thesis
May 2021

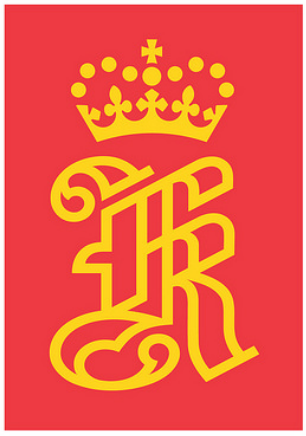
NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

NTNU

Kunnskap for en bedre verden

Mcon Thruster Simulator

IE303612 Bachelor thesis



KONGSBERG

Report 2021

Emilie Skalstad Skarbø
Linda Sperre
Maria Furmyr
Nicklas Mellum

Total number of pages including the front page: 311
Ålesund, 20th May 2021

Title:

Mcon Thruster Simulator

Emilie Skalstad Skarbø

Linda Sperre

Maria Furmyr

Nicklas Mellum

Date:	Subject code:	Subject:	Document access:
20th May 2021	IE303612	Bachelor thesis	

Study:	Number of pages/ attachments:	Bibl. nr:
Automation Engineering	311 / 154	

Adviser:

Anete Vagale

Co-supervisors:

Ottar L. Osen & Robin T. Bye

Preface

What intrigued us about this project was the relevance to the industry and the coverage of several fields in automation. Creating semi-automatic FAT and thruster simulations of the Mcon Thruster Control includes several elements learned during this degree, and opportunities for innovation and new learning. The project touches on different areas within automation, such as computer technology, engineering, coupling, and cybernetics, which benefited our diverse choices of courses and interests.

This work is the result of our bachelor thesis written for Kongsberg Maritime at the Norwegian University of Science and Technology (NTNU) in Ålesund during spring 2021. The thesis is the final part for getting the degree of bachelor in Automation engineering. We want to inform the reader that a general understanding of engineering, computer technology, and automation is required to understand the content of this report fully.

Acknowledgment

We want to express our gratitude towards Kongsberg Maritime for allowing us to complete our degree with a thesis connected to the industry. Also, we want to thank all contributors who have helped us during this project, and we would especially like to thank:

- Our supervisor Anete Vagale at NTNU for guidance.
- Our co-supervisors Ottar L. Osen and Robin T. Bye at NTNU for guidance throughout the project.
- Håkon Lunheim from Kongsberg Maritime, for guidance through the project.
- Chaney Wang Sætre from Kongsberg Maritime, for guidance with physical equipment.
- Thor I. Fossen from NTNU, for guidance with thruster model
- Henrik Daniel Christensen, support from B&R Automation for communications guidance.
- Support from Kongsberg Maritime.
- Family and friends who have supported us throughout this period.

Summary

The project concerns the development of a system for easy assembly of Mcon Thruster Control at Automation Longva facility. The main goal is to create a semi-automatic test system for the thrusters and create realistic thruster feedback to the control system. To be able to achieve this, the physical test equipment will need to be replaced by software. The software should consist of logic, modeling, simulation, testing, printing of test results, and a GUI to aid the inspection missions.

The results prove that the system is successful in its tasks. All the parts are communicating with each other, minimizing time used on testing and verification. The testing has also become more accurate by automating the existing test system. The solution is made expandable for further task addition and improvements.

Sammendrag

Formålet med prosjektet er å utvikle et system for å enklere kunne teste montering av Mcon motorkontroll hos Automation Longva. Hovedmålet er å utvikle et semi-automatisk testsystem for motorene, og simulere realistiske motor-tilbakemeldinger til kontrollsystemet. For å kunne løse disse delene, må den fysiske oppkoblingen byttes ut med digital programvare. Programvaren skal støtte opp inspeksjoner ved å kunne håndtere logikk, modellering, simulering, testing, utskrift av testresultat og et brukergrensesnitt.

Resultatene viser at systemet er vellykket. Alle delene kommuniserer med hverandre, og vil redusere tidsbruk ved testing og verifisering. Automatiseringen av det eksisterende testsystemet har i tillegg gjort testingen mer nøyaktig. Løsningen er utvidbar for videre implementering av funksjonalitet og forbedringer.

Contents

List of Figures

List of Tables

Terminology

Binary A number expressed in the base 2 numeral system, using only two numbers, normally consisting of 0 and 1.

Branch Diverge from main line of development, to continue work without messing with the main line

C++ Early developed general purpose programming language

Commit Operation sending the latest changes of the source code to the repository, making these changes part of the repository

Corrupt data Errors in computer data, occurring during writing, reading, storage, transmission or processing, which introduces unintended changes to the original data

Covid-19 Worldwide ongoing pandemic

Deserialization Process of converting data back into it's original form

File tree One level of information, referring to a hierarchy of files and directories

GUI Graphical User Interface, makes it possible to interact with a computer

Human-readable-text Text that is readable for humans, in contrast to machine-readable-text

Interface An abstract type implemented by classes for specifying behaviour

LaTex Document preparation system

MATLAB® Multi-paradigm programming language and numeric computing environment in a unique programming language

Modbus Communication protocol used for PLC

Objects Abstract data type, which integrates code and data as behavior and state

Object Oriented Programming paradigm based on "objects", which can contain data and code. Data in form of fields or properties, and code in form of procedures or methods

OPC UA Open Platform Communication Unified Architecture, used for communication from machine to machine

Open Source A license which grants everyone the right to use, study, change and share a software

Programming Language Allows giving instructions to a computer in a language the computers can understand

Packet Small amount of data sent over a network

Parameter Values to be passed into a function

Pull request Lets the developer tell others about changes pushed to a branch in a repository

Repository Storage location for software packages

Serialization Process of converting data into a format that can be transmitted or stored

Simulink® MATLAB-based graphical programming environment

Software packages Software collection of individual files or resources

TCP/IP Transmission Control Protocol / Internet Protocol, used for communication between a server and a client

Thread The smallest sequence of programmed instructions managed by a scheduler, scheduling what threads to be run. Normally used in real time programming

Variable Used to store information to be referenced and manipulated

Notation

ACK Acknowledgement message from receiver

Abbreviations

AI Analog Input

AO Analog Output

BC Back-up Control

B&R Company for simulating and creating automation and process control solutions

CIP Common Industrial Protocol

CP Controllable Pitch

CPP Controllable Pitch Propeller

CPU Central Processing Unit

DI Digital Input

DO Digital Output

FAT Factory Acceptance Test

FBD Function block diagram

FGL Functioning Group Leader

FP Fixed Pitch

FPP Fixed Pitch Propeller

FHI Norwegian Institute of Public Health

GUI Graphical user interface

HIL Hardware In-The-Loop

HW Hardware

IP Internet Protocol

I/O Input / output

IMO International Maritime Organization

JVM Java Virtual Machine

LAN Local Area Network

LTI Linear Time-Invariant

Mcon Maneuvering Control

MSS Marine Systems Simulator

NC Normal Control

NTNU Norwegian University of Science and Technology

OID Operator Input Device

OPC UA Open Platform Communications Unified Architecture

PID Proportional-integral-derivative

PLC Programmable Logic Controller

POM Project object model

RPM Rotations per Minute

RPS Rotations per Second

SIL Software In-The-Loop

ST Structured text

TCP Transmission Control Protocol

Nomenclature

A_E/A_O Blade area ratio $[m^2]$

B_m Viscous friction $[m^2/s]$

$C(s)$ Output of transfer function

D Diameter of the propeller $[m]$

e Error signal

$G(s)$ Transfer function based on output and input

h_p Pitch ratio

J Advance coefficient

J_m Rotational inertia $[kg-m^2]$

J_p The moment of inertia of propeller $[kg-m^2]$

K_m Motor constant

K_t Thrust coefficient $[N]$

K_q Torque coefficient $[N]$

N Number of teeth on the gear dial

n Shaft speed $[RPS]$

P/D Pitch-diameter ratio

Q_p Propeller torque $[Nm]$

Q_m Motor torque $[Nm]$

r Radius of drive gear $[m]$

R_e Reynolds number

r_p Propeller radius [m]

r_{RPM} Reference signal on RPM order [%]

$r_{Referense}$ Reference signal

$R(s)$ Input/reference

T Gear torque [Nm]

T_m Motor torque [Nm]

T_p Propeller thrust [N]

u n|n|, where n = RPS [RPS^2]

V_m Voltage value in percent -100 to 100 [%]

y_{Output} Output signal

y_{RPM} Output signal for RPM feedback [%]

z Number of blades

Greek symbols

ω Angular velocity of the propeller [rad/s]

$\frac{d\omega}{dt}$ Angular acceleration of the propeller [rad/s^2]

ρ Water density [kg/m^3]

ω_m Angular velocity of motor [rad/s]

θ Angular displacement of drive gear [$Radian$]

θ_p Pitch angle [$Radian$]

Chapter 1

Introduction

1.1 Background

Testing of Mcon Thruster Control at Longva Automation Facility is time consuming using the present solution. The present solution is a physical setup system containing wiring for each signal, where all the signal handling has to be performed manually. The thruster response is therefore incorrect due to no automatic simulated feedback of the thruster system.

The topic of this thesis is to investigate better solutions for testing with the Mcon Thruster Control. Automation of prior manual tasks is essential for staying competitive.

1.1.1 Lean Manufacturing

The Lean principles appraise how the value for the customer and the society may increase by optimizing the product and streamline the production process and how the streamlining of the process can help reduce waste.

The five Lean Fundamentals:

1. Customer value
2. Value stream

3. Flow
4. Push and pull
5. Perfection

Developing software that can test and verify the Mcon Thruster Control will avail Kongsberg Maritimes production and positively influence the value stream. A value stream is about gaining control of the production process. Examining each link can increase the quality of a given product and streamline the individual links in the value stream. It is essential to deliver quality products effectively.

In this case, it is appropriate to look at the production time, and more specifically the time used building the test setup and performing the factory acceptance test at Automation Longva Facility. By using a simulator instead of a physical setup for testing, time used and cost would decrease. A diagram showing the process with today's solution, and our solution, can be found in Appendix A.

1.2 Project Introduction

This thesis aims to create a better solution for the testing of thrusters by minimizing time use and creating more realistic thruster behavior. Creating a software solution for the test setup and thruster behavior provides a better and faster solution. A graphical user interface with logic handling all the signals will give a better overview and control during testing. Semi-automatic handling of the test will reduce the time spent on testing. Thruster simulation will create a more realistic behavior when performing the test.

1.3 Aim and Objectives

This bachelor thesis functions as a proof of concept in terms of creating a functionally automatic software solution that covers all the requirements. The main task will be to create communication between all the software while keeping the functional quality required for the testing. Whereas the core objectives to fulfill Kongsberg Maritimes requirements are listed below.

- Simulate a set of chosen I/O signals from Mcon
- Simulate thruster
- Create semi-automatic factory acceptance test, with printing of test results

1.4 Limitations

In this project certain limitations has been considered. These limitations are chosen to restrict the project from being too large or too vague. These limitations make the project solvable in the limited time available, and some could be removed or adjusted if the project were to be continued over an extended time period. The chosen limitations for this project are:

- A one quadrant model for the thruster model is used, instead of a four-quadrant model.
- Some information on parameters that could have made the thruster model more realistic were not available.
- Not all tests in FAT can be digitized, and some must therefore be handled manually.
- Signals used outside of the FAT were not implemented.

Some limitations have an impact on the solution presented in this report and would have lead to other results.

1.5 Structure of the Report

The report is structured as follows:

Chapter 1 - Introduction - This chapter is about the background for the project, as well as what the goals and limitations were with this thesis.

Chapter 2 - Theory - This chapter presents the theoretical basis behind all decisions and solutions used during this thesis.

Chapter 3 - Materials - This chapter contains the materials used in the project.

Chapter 4 - Method - This chapter describes the development of the software and contains all the information needed to recreate this project.

Chapter 4 - Results - This chapter contains the presentation of all the results and solutions.

Chapter 6 - Discussion - This chapter presents a discussion regarding the different solutions, test results and the thesis as a whole.

Chapter 7 - Conclusion - This chapter concludes the thesis work.

Chapter 2

Theoretical basis

The following sections will explain the theoretical basis behind all the decisions and solutions for this project. First, an explanation of Mcon will be given, then the theory of the logic and modelling is presented, followed by an explanation of the FAT. The chapter also includes a description of the communication theory and necessary programming concepts and languages.

2.1 Mcon Thruster Control

Mcon is the latest generation of Kongsberg Maritimes remote control systems designed for control for a wide range of propulsion and thruster products. The key benefits of Mcon is [65]:

- Better control of propulsion and thruster units.
- Touch-screen GUI for user-friendly and intuitive operations.
- Integrated force feedback notifying the operator when the bridge is not in command.
- No change in thrust when transferring command between bridge stations, all input devices being in the same position at all times.
- Remote start/stop of thruster motor/servo pumps.

2.2 Normally Open vs Normally Closed

Normally open (NO) and Normally closed (NC) are terms used to define the states of switches, sensors or relay contacts under when its coil is not connected [30], illustrated in Figure 2.1.

An NO contact remains open until a certain condition is satisfied, for example a light switch. A light switch will stay off until the button is pressed, closing the circuit allowing power to go through.
An NC contact remains closed until a certain condition is satisfied. For example an emergency stop. The switch in the circuit is closed when it is not pressed, and opening it will cause the circuit to break and power to be cut [30].

Figure 2.1: Fixed Pitch Propeller [29]

2.3 Thruster

2.3.1 Tunnel Thruster

Tunnel thrusters are designed to provide side force to the ship to enhance maneuvering capability in port or additional station keeping power during dynamic positioning, and tailored to match the vessel application. They are available with Controllable Pitch (CP) or Fixed Pitch (FP) propeller [61].

2.3.1.1 Fixed Pitch Propeller (FPP)

A FPP can only be controlled by controlling the RPM, (see Figure 2.2). The fixed pitch type propellers are casted and the position of the blades and hence the position of the pitch is permanently fixed and cannot be changed during the operation [101].

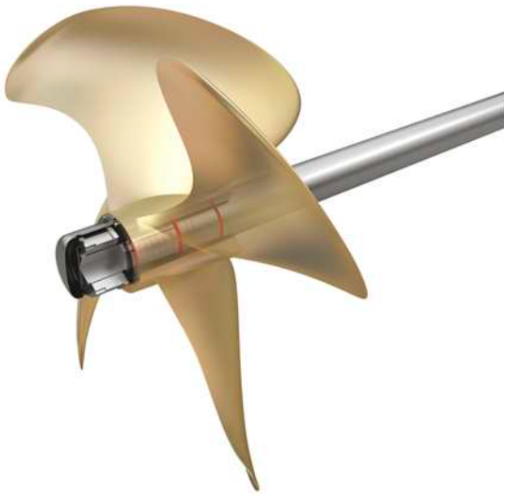


Figure 2.2: Fixed Pitch Propeller [60]

2.3.1.2 Controllable Pitch Propeller (CPP)

A CPP can be controlled by both the propeller speed and the angle of the propeller blades (pitch) (see Figure 2.3). Control of the pitch gives a significant increase in efficiency and a blade foot with decreased exposure to cavitation [59].

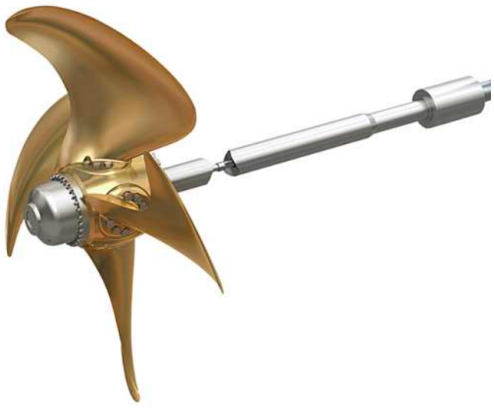


Figure 2.3: Controllable Pitch Propeller [59]

2.4 Modelling

2.4.1 Control system

"A control system consists of subsystems and processes (or plants) assembled to obtain the desired output with desired performance, given a specific input" [72].

A control system can simulate the behavior of real-world objects such as engines and gears. Control systems can also be used to describe complex systems such as aeroplanes, cars and vessels. A mathematical model must be created for the system desired to model or simulate. The mathematical model of control systems often take form in either of the three following shapes [24]:

1. An nth order differential equation representing the system.
2. A transfer function, which can be described as a Laplace transform of the differential equation, is put in the form of output divided by input.
3. A state space model which can be described as an nth order differential equation as n simultaneous first-order differential equations.

The process of designing a control system can be broken down into the following steps [72]:

1. Determine physical system and specifications from requirements.
2. Draw function block diagram.
3. Represent the physical system as schematic.
4. Use the schematic to obtain the mathematical model of the given system.
5. Reduce the block diagram.
6. Analyze and design system to meet specified performance for stability and transient response, steady-state performance.

2.4.2 Transfer Function

A *transfer function* is defined as a linear, time-invariant (LTI) differential equation system. The ratio between the Laplace transform of the output and the Laplace transform of the input describes the system [25]. Initial conditions must always be assumed to be zero in a *transfer function.* A *transfer function* $G(s)$ is given by the input $R(s)$ and the output $C(s)$ of the system (See equation 2.4.1).

$$G(s) = \frac{C(s)}{R(s)} \tag{2.4.1}$$

One of the great benefits of utilizing the transform function is its simplicity. All needs for integration and derivation are removed. Making it dependent on basic algebraic mathematics to solve the mathematical equation by taking the Laplace transform of the output and input. In order to find the complete system of multiple transfer functions connected in cascade, the transfer functions are multiplied to obtain the overall transfer function of the complete system [25]. If the transfer function is connected in parallel, they can be summed to find the overall system. These simple algebraic operations make the transfer functions very simple and efficient to work with if there are zero initial conditions for the system.

2.4.3 PID Controller

PID controller is short for proportional-integral-derivative controller, and can be described as a controller used to stabilize a system with the help of proportional, integral, and derivative gains [22]. PID controllers are the most popular option for regulating a closed-loop system. It is assumed that approximately 95 % of all controllers used are PID controllers [81]. The structure of a PID controller can be seen in 2.4, where *C(s)* represents the PID controller and *P(s)* represents the system being controlled. A feedback signal is coupled back to the controller, where the actual value y_{Output} of the system is subtracted from the desired value $r_{Referense}$. This creates the signal *e*, which is fed through the PID controller again. This is an iterative process that

will eventually drive the error signal to zero and stabilize the system if the PID controllers are correctly tuned.

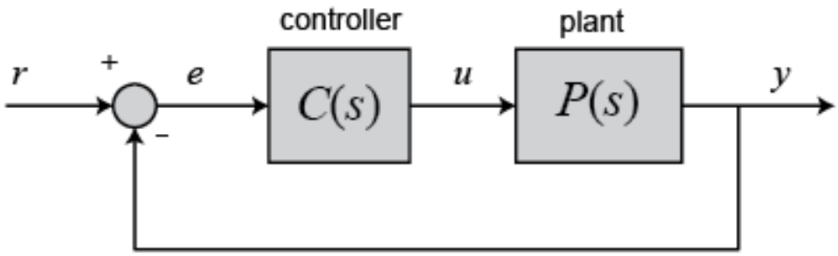


Figure 2.4: Structure of a PID controller [21]

The Output of the PID controller can be calculated using equation 2.4.2[22].

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de}{dt} \tag{2.4.2}$$

K_p represents the proportional part, K_i represents the integral part, and K_d represents the derivative part of the PID controller. e is the difference between the desired and actual output, and can be seen in Figure 2.4 as the input of the controller.

The proportional, integral, and derivative gains that comprise the PID controller affect the characteristics of the model in different ways. Figure 2.5 provides an overview of the different ways each part can affect the performance and characteristics of a system [22].

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Decrease
Kd	Small Change	Decrease	Decrease	No Change

Figure 2.5: Characteristics of the proportional, integral, and derivative parts that make up the PID controller [21]

2.5 Propeller and motor dynamics

When creating a propeller and motor model, the relation between the two is taken into the equation. In addition, some simplifications were necessary since a propeller and motor can be complex.

In order to simplify the dynamics of the model, some of the vessel's dynamics are neglected. It was assumed that the propeller is in water, but the foundation of the propeller does not move forward as the vessel normally would. This assumption provides a basis for a simplified task, namely a physical model, comprised by the dynamics of the propeller and the electrical motor. The dynamic equation for the model can be written as [105]:

$$J_p \frac{d\omega}{dt} = Q_m - Q_p \tag{2.5.1}$$

J_p represents the moment of inertia of the propeller, $\frac{d\omega}{dt}$ is the angular acceleration of the propeller, Q_m represent the torque of the electric motor and Q_p represent the propeller torque. The calculations of the Q_p can be found in the sections 2.5.1.

2.5.1 Propeller

Two of the most common propellers for vessels are fixed-pitch propellers (FPP) and controllable-pitch propellers (CPP). These two variations of a propeller are widely used as prime mover thrust device [39]. When designing a mathematical model of a marine propeller, two equations stand as a central aspect of propeller modeling.

$$\begin{aligned} T_p &= \rho D^4 n|n| K_t \\ Q_p &= \rho D^5 n|n| K_q \end{aligned} \tag{2.5.2}$$

These equations represent propeller thrust (T_p) and propeller torque (Q_p). ρ represents the water density, D represent the diameter of the propeller, n describes shaft speed, while K_t and K_q represents thrust- and torque coefficients, respectively. The thrust is neglected based on assumptions of the propeller being in the water with no motion to the foundation. Calculations that support these assumptions can be found in section 2.5.

2.5.1.1 Finding K_q and K_t coefficient

In order to find the torque and thrust produced by a propeller, the torque and thrust coefficient must be found first. The Wageningen B-series propeller is used as a basis for finding the Q_p coefficients. The B-series' open-water characteristics of 120 propeller models were tested and analyzed with multiple polynomial regression analysis. The derived polynomials with multiple regression analysis express T_p and Q_p in terms of the number of blades, the blade area ratio, the pitch-diameter ratio, and the advanced coefficient. The polynomial coefficients provided in this work are valid for $R_e = 2 * 106$.
The series consists of propellers ranging from 2 to 7 blades, with blade area ratios from 0.30 to 1.05 and pitch to diameter ratios from 0.5 to 1.4 [71]. The coefficients C_s^T, t, u, v and C_s^Q, t, u, v and terms s, t, u, v are given in Appendix B.

$$K_q = \sum_{s,t,u,v} C^Q_{s,t,u,v} (J)^s (P/D)^t (A_E/A_O)^u (z)^v \tag{2.5.3}$$

$$K_t = \sum_{s,t,u,v} C^T_{s,t,u,v} (J)^s (P/D)^t (A_E/A_O)^u (z)^v \tag{2.5.4}$$

2.5.1.2 Four-quadrant model

Forward and reverse shaft revolution speeds and inflow speeds produce four operation quadrants for a propeller. Figure 2.6 shows the propeller definition of the 4-quadrant operating con-

ditions of a propeller-rotor [57].

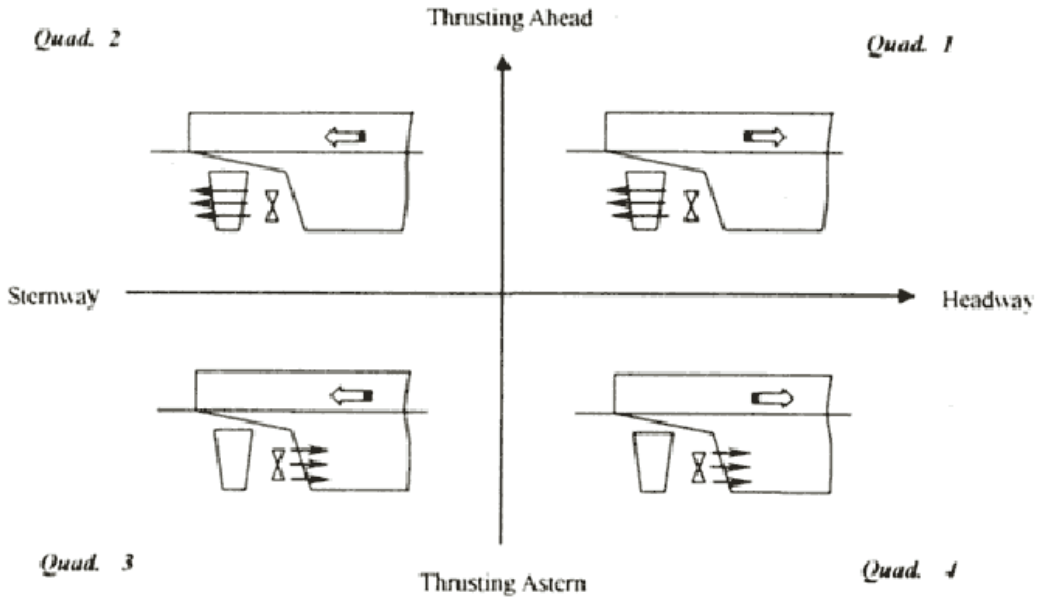


Figure 2.6: Four quadrants of ship speed and propeller operations [87]

The four quadrants defined can also be divided into two modes, propulsion mode, and turbine mode. Below is a summary of the four-quadrant operations:

1st quadrant Normal forward propulsion mode

2nd quadrant Normal turbine power generation mode

3rd quadrant Reserve/astern propulsion mode

4th quadrant Reverse inflow turbine power generation mode

2.5.1.3 Pitch dynamics

CPP uses the variable pitch rate of the propellers, which makes this an essential characteristic for this kind of propeller. A variable-pitch propeller is a propeller that can rotate its blade orientation around its longitudinal axis (see Figure 2.3) [104].

By changing the pitch of the blades, the propeller will be able to regulate the thrust and speed

of the boat without changing the rotational speed of the propeller. The equation for propeller pitch can be seen in equation 2.5.5 [45].

$$\theta_p = \arctan\left(\frac{h_p}{\pi r_p}\right) \tag{2.5.5}$$

Where θ_p represents the angle of the pitch, h_p is the pitch ratio, and r_p is the radius of the propeller.

2.5.2 DC motor

A direct Current motor (DC motor) is defined as an electrical motor, which converts direct current electrical energy into mechanical energy. The most common type of DC motor relies on electromagnetic fields to convert electrical energy into mechanical translational/rotational energy [103].

A simple model of a DC motor consist of the following main components [32]:

Armature Also called a rotor. The armature is made of electrical windings around the rotor arm. These windings produce a magnetic field when subjected to a current. The magnetic poles generated are then attracted to the opposite poles generated by a permanent magnet called the stator.

Stator A magnet or electromagnetic windings that generate a magnetic field around the armature/rotor.

Commutator A DC motor does not use any external current switching device; instead, it uses a mechanical connector. this connector is called a commutator.

Brushes As the motor turns, the brushes slide over the commutator/connector to create a magnetic field. When voltage is applied across the brushes, a dynamic magnetic field is generated.

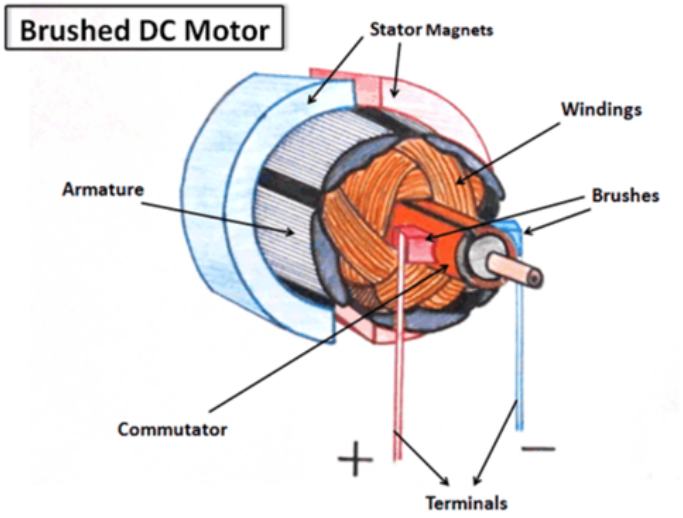


Figure 2.7: Components of a DC Motor [32]

2.5.2.1 Gear Ratio

Gears are leverage to either increase rotational speed or increase the amount of torque produced. The gear ratio describes the relationship between the gear-input and -output in terms of speed or torque produced. The effect a pair of gears has on the rotational speed or torque can be found by obtaining this ratio. Figure 2.8 describe the gear ratio of a system of gears [26].

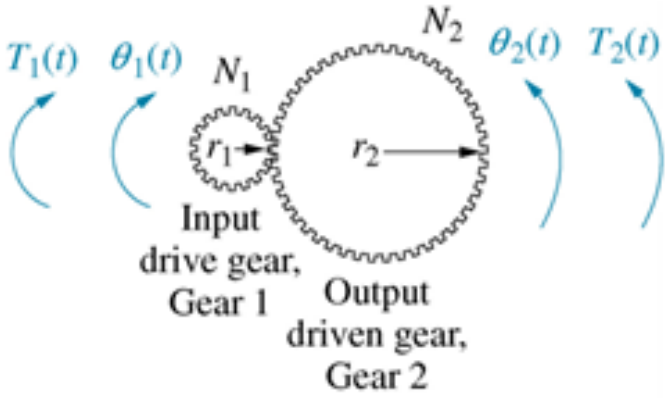


Figure 2.8: Gear ratio [72]

The distance traveled along the circumference of the gears is the same and is represented in equation 2.5.6. Subscript 1 represents the input gear, and subscript 2 represents the output gear

(see Figure 2.8).

$$r_1\theta_1 = r_2\theta_2 \tag{2.5.6}$$

The number of teeth on a geared dial is proportional to the radius of the gear. Rearranging equation 2.5.6, with this knowledge, gives the relationship shown in equation 2.5.7.

$$\frac{\theta_2}{\theta_1} = \frac{r_1}{r_2} = \frac{N_1}{N_2} \tag{2.5.7}$$

θ is the angular displacement, r represents the radius, and N is equal to the number of teeth on the gear dial. The same logic is followed for torque and obtain equation 2.5.8.

$$T_1\theta_1 = T_2\theta_2 \tag{2.5.8}$$

Which also can be rewritten as:

$$\frac{T_2}{T_1} = \frac{\theta_1}{\theta_2} = \frac{N_2}{N_1} \tag{2.5.9}$$

By taking advantage of these formulas, both angular displacement- and torque-relationships can be obtained by dividing the number of teeth on the gears. This relationship obtains a ratio for torque or angular displacement. Multiplying this ratio with a system will obtain either desired speed or torque in the system. For multiple gears, simply multiply the ratios to obtain the total gear ratio of the system.

2.6 FAT

International Maritime Organization (IMO) is a specialized agency responsible for the safety and security of shipping and the prevention of marine and atmospheric pollution by ships [53]. IMO standard is required before delivery and is certificated through a Factory Acceptance Test (FAT). The FAT checks and verify systems and equipment and will discover faults before delivery. Quality assurance and risk management companies, such as DNV, RINA, and others, are present when Kongsberg Maritime performs the FAT and verifies that the result follows given class requirements.

A FAT is unique for each system and must be sequenced thoroughly without a rush to prevent a faulty system and consists of a wide variety of inspection points and tests. The process evaluates the equipment after the assembly process by verifying that it is built and operating by design specifications [27] [70].

Having done a FAT on the equipment is also a great reassurance for both the producer and the customer. The producer can fix problems while the system is still at the factory and make sure that the system they are selling is functioning as expected. It also prevents faults from occurring while equipment is mounted on the vessel, saving the cost and time of returning the equipment or sending a maintenance specialist and preserving security [27] [70] [66].

The FAT used for the Mcon Thruster system consists of "Functional tests" and "Faults and Consequences" for the tunnel thrusters. These tests include tests for general functions, Rpm Control, Start/stop/reset functions, interfaces to external systems, and alarms. The complete FAT is attached as Appendix E [64].

2.7 Communication

2.7.1 Communication Architecture

A server is an always-on host in a server-client architecture which clients request services or functionality from [63]. A server is a machine with large memory and a lot of storage. It has a

fixed, accessible IP address for clients to connect. The server is responsible for sharing data, process calculations, or regulating internet traffic between clients since the clients do not directly communicate with each other [84].

A client is a data program located at a user, such as a computer, mobile, or a browser. All clients connected to a server will get the same data, and the clients will therefore showcase the same information to all the users [85]. The calculations will therefore only need to be handled in the server. This solution makes the program faster, and the probability of data errors, such as missing or changed data, is minimized [3].

One server can connect to multiple clients, and a client can use multiple servers [88]. If a server has multiple clients, the server needs to regulate the internet traffic, so as all the clients can access the content at the server [3]. The server and client can be located on the same device or connect over a network from different devices. The server and client usually are connected using the request and response model, where a client requests a service from the server, which then performs the service and sends back a response in the form of an acknowledgment before cutting the connection [95]. Communication protocol get used when such connections are performed [2].

2.7.2 Communication Protocol

Communication protocols describe digital message formats and rules and are used to exchange data in or between computing systems. They ensure that data are transmitted and received properly and provide consistency and universality for sending and receiving messages between different end parts [91] [94].

2.7.2.1 Ethernet/IP

Ethernet/IP stands for Ethernet/Industrial Protocol and is a protocol that adapts the Common Industrial Protocol (CIP) to standard Ethernet. Ethernet/IP uses the Ethernet standards, internet protocol and IEEE 802.3, to define the features and function for its transport, network, datalink, and physical layers [106].

2.7.2.2 Automation Studio Target for Simulink

Automation Studio Target for Simulink is a toolbox for extending Simulink® and serves as an interface between MATLAB®/Simulink® and B&R Automation Studio. The code in MATLAB®/Simulink® is automatically generated to the language C and integrated into the B&R Automation Studio project, using Simulink® coder together with Automation Studio Target for Simulink [13].

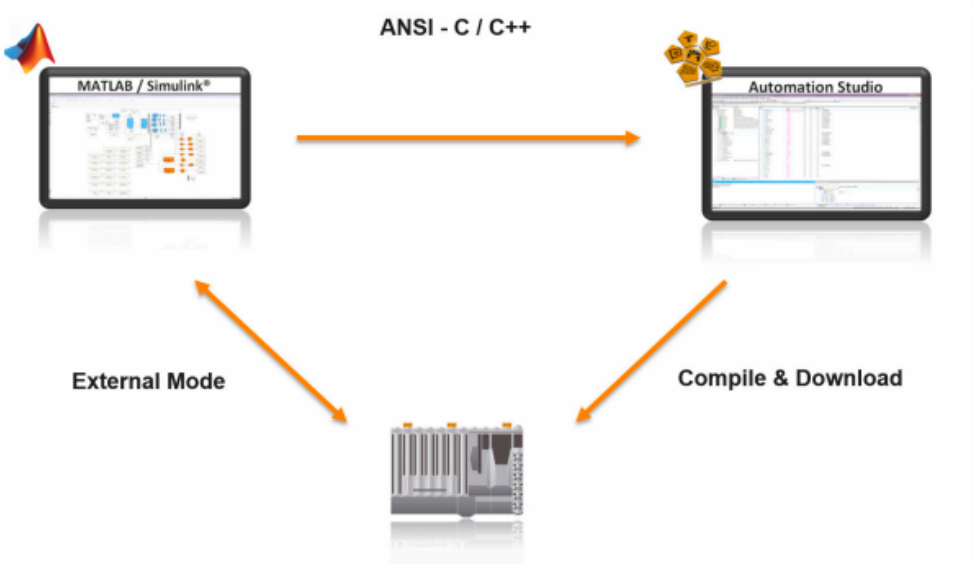


Figure 2.9: Concept of Automation Studio Target for Simulink [13]

The toolbox adds a B&R library into the Simulink® library. It provides blocks that allow a connection to B&R Automation Studio and seamless integration of the generated program code. In addition to B&R-specific blocks, Automation Studio Target for Simulink also supports all standard input/output blocks from Simulink® [13].

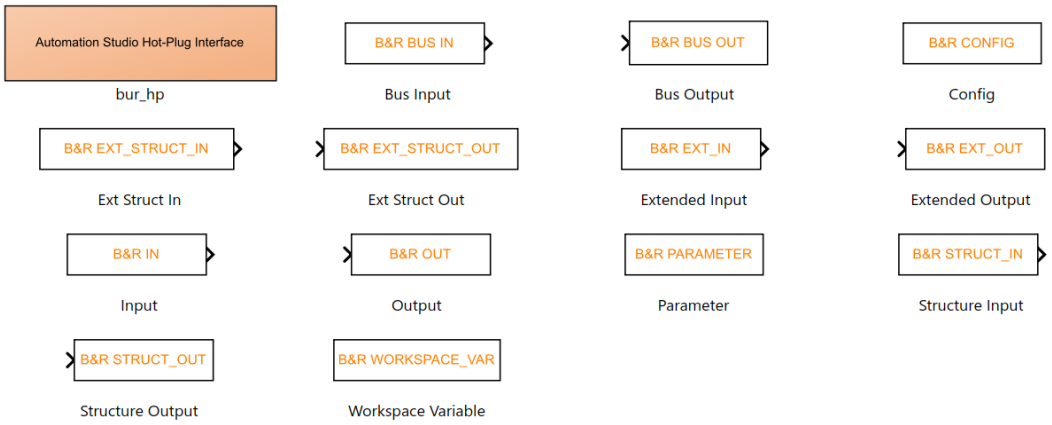


Figure 2.10: Automation Studio Target for Simulink library [46].

B&R Automation Studio Target for Simulink blocks [13]:

B&R Config block Required to configure the Automation Studio connection and to select the coder and programming language.

The B&R IN block / B&R OUT block creates a process variable in Automation Studio that can be linked to inputs or outputs.

B&R PARAMETER block creates a process variable in Automation Studio to make an internal parameter visible externally.

B&R WORKSPACE_VAR block enables variables created in MATLAB®/Simulink® to be used in the Simulink® model and generated as process variables in the Automation Studio project.

B&R EXT_IN block / B&R EXT_OUT block creates process variables in Automation Studio to adapt Simulink® variables to the hardware inputs and outputs.

B&R STRUCT_IN block / B&R STRUCT_OUT block allows existing Automation Studio structures to be used in the Simulink® model. The structures are read from a specified .typ file.

B&R BUS IN / B&R BUS OUT block generates a corresponding structure in the Automation Studio .typ file from a Simulink® bus variable as well as a corresponding process variable of type "structure".

2.7.2.3 TCP/IP

TCP/IP is the most commonly used protocol for clients to communication with servers, and is consisting of two individual protocols, Transmission Control Protocol (TCP) and Internet Protocol (IP). TCP is a connection-oriented protocol, and ensures a secure transportation of data between the connected users [38]. TCP establishes and maintains a connection until all messages is exchanged [1]. The exchange is performed using a three-way handshake, illustrated in Figure 2.11. The sender sends a request to the receiver, which answers by sending back data, before the sender confirms that the data is received [84].

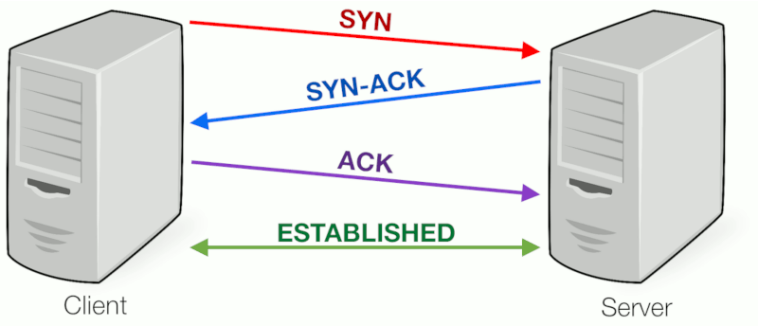


Figure 2.11: Illustration of a three-way handshake between a server and a client [96]

If the data becomes lost and therefore not received, the data will be retransmitted until the sender receives a confirmation from the receiver, as illustrated in Figure 2.12 [84].

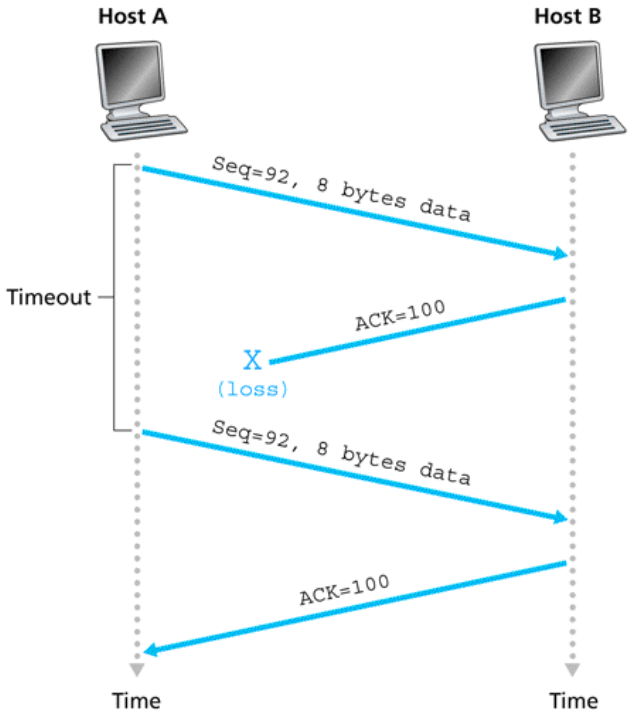


Figure 2.12: Illustration of a retransmission due to lost data [74]

IP is a connectionless protocol, and there are no continuous connection between the users communicating with each other. By using IP, the data is broken into small packets, before sending over the internet. Each packet is travelling as a independent unit of data, before being put together at the destination. The packets needs to be put together in the correct order to extract the same data as before sending. This process is accomplished by TCP, by giving all the packets different sequence- and acknowledgement-numbers, as in Figure 2.12 [1]. IP is responsible for directing each packet to the correct destination [6] [84].

2.7.2.4 OPC UA

The most common communication protocol in automation is Open Platform Communications (OPC), which has evolved from the classical standards to OPC Unified Architecture (UA). This new version of OPC is platform independent, more secure, makes communication over network easier and integrates all the functionality of the earlier individual OPC classic versions into one

extensible framework [41] [73].

The OPC protocol uses the server-client architecture, described in section 2.7.1, for communicating. OPC standardizes the access to machines, devices and other systems in industrial environment and enables the data exchange to be similar and manufacturer-independent. Because of the standardization, each OPC client can communicate with any OPC server. The OPC UA standard was built based on basic web technologies, TCP/IP and http/SOAP, and has therefore TCP/IP integrated, which is necessary for communicating over the network [86].

2.8 Programming Concept

2.8.1 Cyclic

In a CPU, there are four operations, illustrated in Figure 2.13, being repeated continuously. Each operation's execution is called a scan, and the time it takes for the system to perform a scan is called a cycle time/cyclic. The cycle time can vary from one cycle to another [49]. B&R Automation Studio is based on a cyclic time, which is based on a fixed interval.

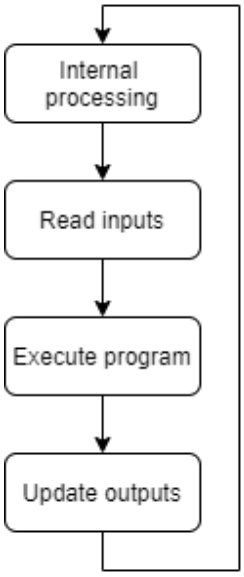


Figure 2.13: The four operations [49]

2.8.2 Object Oriented Programming Paradigm

The object-oriented programming paradigm relies on concepts such as classes and objects. Classes are simple reusable pieces of code blueprints used to create individual instances of objects. The classes contain what attributes an instance of this type will have, but not the value

of those attributes for a specific object type. A class called "Car" will, for example, contain attributes such as *color*, *brand*, *year model* and so on. The classes normally contain methods or functions as well, which are used for performing helpful actions, like reading or changing the values of an attribute for a specific object [35].

2.8.3 Interface

An interface is an entirely abstract class consisting of static constants and a collection of abstract methods without a method body. It is used to group related methods, archive abstraction, and multiple inheritance [54]. For accessing the interface methods, a class needs to implement or inherit the interface, using the *implements* keyword [99].

An interface is similar to a class. However, instead of describing the attributes and behaviors of an object, the interface contains behaviors that a class implements. If the class implementing the interface is not abstract, all the methods in the interface need to be defined in the class [97].

2.8.4 Serialization

Serialization is a process to store data in files on disk, save program's states on disk or send data over a network as objects. Serialization is where an object serializes with sending and deserializes with reception. The object is converted into a sequence of bytes containing data, object type, and type data written into one file. The receiver then reconverts this file into the original content [5]. Usually, the Java Virtual Machine (JVM) is responsible for reading and writing serialized objects [48]. JVM is included in all machines, which enable serialization of objects at one platform, and deserialized at another [4] [56]. Another advantage with serialization is that multiple data can be sent at once, like multiple parameters [9].

2.8.5 Maven

Maven is a software project management and comprehension tool provided by Apache Software Foundation. This tool is based on the project object model (POM) concept and can be used for building and managing any Java-based project [83] [11]. It is a standard for building projects, provides a clear definition of what the project consists of, and enables sharing of JAR

across several projects. The primary goal of this tool is to allow a developer to comprehend the complete state of a development effort in the shortest amount of time. Once the developer gets familiarized with one Maven project, the building of all Maven projects becomes known [10].

2.8.6 Version Control

Version control is a system for managing changes in files over time. The system enables the recalling of specific versions. This function can be used for almost any type of file but is commonly used for code files. Version control allows to revert files or the whole project to a previous version, compare changes over time, see who last modified the files, and makes it easy to recover from lost or faulty files [42].

When working in teams, version control is essential. The project consists typically of multiple developers working simultaneously, and the project, therefore, needs to be organized as a "file tree." It also allows developers to work on different parts of the file tree simultaneously without overwriting each other's code [12].

2.9 Programming Language

Programming languages are languages that computers can understand. These languages vary between many different functions, but the standard for all is to be translated into binary through compilation. In binary code, the computer reads the language as only zeroes and ones, enabling information quickly and efficiently. Different languages perform different tasks, and this project consists of several different programming languages [89]. All the programming languages used in this thesis are listed below.

2.9.1 Java

Java is a free and open-source object-oriented programming language developed for use in the distributed environment of the internet. One of the main advantages of using Java is that it can run on almost any platform [93] [100].

2.9.2 Structured text

Structured text (ST) is one of the five languages of the IEC-61131-3 standard used for PLC programming. This standard deals with PLC communication and allows PLCs to exchange data by a communication network [8] [92].

2.9.3 C

C is a general-purpose, procedural computer programming language supporting structured programming, lexical variable scope, and recursion, with a static type system [102].

2.9.4 LaTeX

LaTeX is a free typesetting system and is developed for creating high-quality technical and scientific documents. LaTeX enables total control over the look of the document, where the writer can change every aspect if desired [80] [77].

2.9.5 MATLAB®

MATLAB® is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features [28].

Chapter 3

Materials

This section includes a review of all the materials used in the completion of this project.

3.1 Screens

The GUI for the Mcon Control system is visualized on two screens. Mcon GUI is designed to enable intuitive operations through its touch-enabled display providing direct access to all underlying functions [65]. Two screens are used in this project, shown in Figure 3.1

Figure 3.1: Mcon GUI [65]

Mcon thruster RPM display is an additional screen to the GUI, but it is only a visualiser. Two screens are used in this project, shown in Figure 3.2

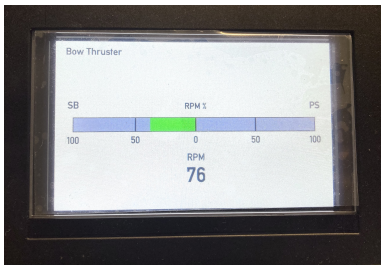


Figure 3.2: Mcon thruster RPM display [46].

3.2 Levers

The levers for the Mcon Control System are high-precision levers with integrated push buttons and dual electronics for normal and back functions. The levers are also motorized [65]. Two levers (see Figure 3.3) are used in this project.

Figure 3.3: Mcon Thruster Lever[65]

3.3 Testjigg

The testjigg (see Figure 3.4) is the present test equipment used during FAT. It contains buttons and dials in order to handle the signals.

Figure 3.4: Physical testjigg [46].

3.4 CPU

The CPU used is the model X20CP3583 (see Figure 3.5) and is delivered by B&R Automation. The X20CP3583 is the entry-level Atom-based X20 CPU. USB, Ethernet, POWERLINK, and removable CompactFlash are all included as standard features. The standard Ethernet interface is capable of handling communication in the gigabit range [14].

Figure 3.5: CPU X20CP3583 [14]

3.5 I/O Modules

The I/O modules needed for this project were two digital inputs, two digital outputs, one analog input, and one analog output. The I/O modules used in this project can be seen in Figure 3.6 and are as follows:

- X20AI4322 [15]
- X20AO4622 [16]
- X20DI6371 [17]
- X20DO6639 [18]

Figure 3.6: I/O modules [46].

3.6 Ethernet cable

The Ethernet cable is used to connect the CPU to the computer, shown in Figure 3.7.

Figure 3.7: Ethernet Cable [7]

3.7 I/O signal cables

The cables used for connecting I/O signals, is shown in Figure 3.8. Approximately 36 wires are needed between the CPUs I/O modules and Mcons I/O modules.

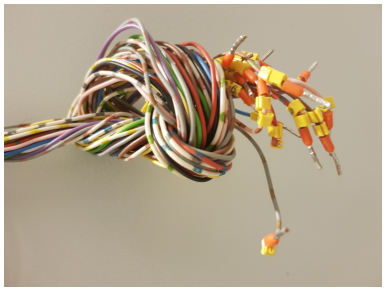


Figure 3.8: I/O wires [46].

3.8 Multimeter

A multimeter is a measuring instrument that can measure multiple electrical properties [107]. The multimeter, shown in Figure 3.9, was used to control and verify the wiring during assembly.

Figure 3.9: Multimeter [107]

3.9 Software

3.9.1 B&R Automation Studio

B&R Automation Studio is an integrated software development environment and allows the user to configure the controller, drive, communication, and visualization in one environment. The software has integrated IEC 61131-3 languages, consisting of the three graphical programming languages ladder diagram, function block diagram, sequential function chart, and the two textual programming languages structured text and instruction list [19].
B&R Automation Studio is used for writing the logic code for the signals in this project and making the GUI.

3.9.2 MATLAB®

MATLAB® is a programming and numeric computing platform used to analyze data, develop algorithms, and create models. MATLAB® combines a desktop environment tuned for iterative analysis and design processes with a programming language that directly expresses matrix and array mathematics [68].
In this project, MATLAB® is used as a init containing the constants used in the Simulink® model 3.9.3.

3.9.3 Simulink®

Simulink® is a block diagram environment for multidomain simulation and Model-Based Design. It supports system-level design, simulation, automatic code generation, and continuous test and verification of embedded systems. Simulink® also provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. Simulink® is integrated with MATLAB®, enabling MATLAB® algorithms into models and exporting simulation results back into MATLAB® for further analysis [69].
Simulink® is, in this project, used to simulate the thruster and propeller, making the RPM feedback more realistic.

MSS

The Marine Systems Simulator (MSS) is a MATLAB® and Simulink® library for marine systems. It includes hydrodynamic models for ships, underwater vehicles, and floating structures [40]. For this project, the Wageningen.m file is used for the propeller calculations.

B&R Automation Studio Toolbox

B&R Automation Studio Toolbox is automatically installed during the setup of Automation Studio Target for Simulink [13]. It is used to create a connection between Simulink® and B&R Automation Studio.

Simulink Coder

Simulink Coder generates and executes C and C++ code from Simulink® models [67]. The coder is used during the transfer of software from Simulink® to B&R Automation Studio.

3.9.4 IntelliJ IDEA CE

IntelliJ IDEA is a program for writing code. It makes coding efficient and enjoyable by using an ergonomic design and intelligent coding assistance. It has integrated version control systems, supported languages and frameworks. IntelliJ is initially developed for the programming language Java and is also the most used program for Java developers [55].
IntelliJ has been used in this project for writing the Java code, which handles the semi-automatic FAT and printing of test results.

3.9.5 Github

Github is a collaboration platform using the Git system, made for sharing code projects between developers. It enables collaborators to work on the same project from anywhere and offers version control. The Git system can handle both small and large projects, and at the same time, provide speed and efficiency. The projects consist of repositories, branches, commits, and pull requests, making it clear who has made which changes in the code. Github also ensures that all the developers has access to both the newest version, as well as all the earlier versions of the code, which reduces the risk of losing any data. [43] [44] [47].
Github has been used for version control as well as for continuous sharing of the developed Java

code.

3.9.5.1 Github Desktop

Github Desktop is a program designed to simplify Github workflow and simplifies working with git. The program is an extension of the Github platform. It makes it easy and fast to contribute projects between the different OS X and Windows platforms. It also makes it easy to up- and download the code to the computer for applying changes, and it highlights the committed changes so that other developers easy can see what is new [33] [23].
In this project Github Desktop has been used as an extension of Github, for a simplified workflow and fast contribution of the Java code.

3.9.6 VNC Viewer

VNC stands for Virtual Network Computing and is a visual desktop-sharing system. VNC Viewer allows users to control another computer remotely. It is connected through IP address, and does not need internet to work [82].
For this project, it was crucial to choose a VNC without a server, as that may interfere with the B&R Automation Studio server. The VNC in this project will be a Graphical User Interface (GUI) where buttons can be pressed, and values can be read and sent from.

3.9.7 UaExpert

UaExpert is a cross-platform OPC UA test client programmed in C++. It is designed as a client for testing the connection to servers. The program is free and available for both Windows and Linux [20].
UaExpert is used in this project for testing if the B&R Automation Studio OPC UA server has been set up correctly and if it is possible to connect to it. UaExpert also provides the NodeId for the variables, which is necessary in the Java client for reading the values in B&R Automation Studio.

3.9.8 Overleaf

Overleaf is an online text editor using Latex. The program requires no installations to use and allows collaboration in real-time. This function means that changes can be viewed immediately by others, and multiple people can edit simultaneously. It also provides version control and templates to be used [76].
Overleaf has been used for writing the bachelor thesis.

3.9.9 Draw.io

Draw.io is a free application for easy creation of charts and diagrams. It is secure to use and provides an opportunity to create visual communication. The application also provides options for sharing and collaborating in a single diagram and enables the diagram to be interactive. Draw.io contains templates, and it allows the movement of boxes, arrows, and other functions, to the exact desired position. It is also easy to convert the diagrams to PDF files [36] [34].
In this project, Draw.io is used to draw diagrams to provide a better description and overview i the report.

3.10 Libraries

Libraries in programming are premade classes made for reusable executing of a task. Libraries enables programmers to expand their code, with a code file that is already tested for their desired need [31].

3.10.1 Library in B&R Automation Studio

3.10.1.1 Standard

The standard library in B&R Automation Studio contains the standard function blocks and functions for IEC 61131-3 [50].

TON

Within the library *Standard* the function TON is used. A description of this function is written

below.

If IN is FALSE, then the Q output is FALSE and the ET output is 0. As soon as IN is TRUE, time in ET begins counting in milliseconds until the value is equal to the value in PT. It then remains at that value. Q is TRUE if IN is TRUE and ET equals PT. Otherwise, it is FALSE. As a result, Q has a rising edge if the time (specified in milliseconds) in PT has elapsed [51].

- IN = Input signal, BOOL type
- PT = Delay Time, , TIME type
- Q = output signal
- ET = Elapsed time, TIME type

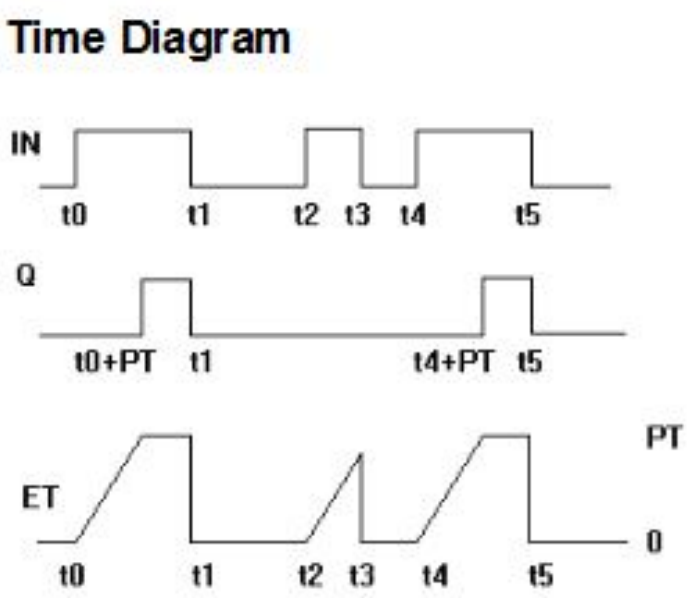


Figure 3.10: Setting the Datapoint of the button. [51]

3.10.2 Libraries in Java

Libraries are easy to implement, using JAR files in Maven. Many thousands of libraries are available, because of open source software, allowing to use, study and change others source code [90] [75].

3.10.2.1 Eclipse Milo

Milo is an open source implementation of OPC UA provided by Eclipse. It provides all the tools needed to implement OPC UA client and server functionality in a JVM-based project. Milo provides a high-performance stack used for serialization and security, among other things. It

also provides an SDK built on top of the stack for developing compliant UA client and server applications [58] [37].

- Eclipse Milo

3.10.2.2 Apache PDFBox

Apache PDFBox is an open-source Java library tool for working with PDF documents provided by Apache Software Foundation. This tool allows for the creation of new PDF documents, manipulation of existing documents, and the ability to extract content from documents [62] [78].

- pdfbox-2.0.1.jar
- fontbox-2.0.1.jar
- preflight-2.0.1.jar
- xmpbox-2.0.1.jar
- pdfbox-tools-2.0.1.jar
- pdfbox-debbugger-2.0.22.jar

Chapter 4

Method

This section contains a closer description of how to fulfill this project and contains all the information needed to recreate this project. First an executive description is provided of the project and the physical assembly of the Mcon rack. Followed by B&R Automation Studio, modeling, FAT, and lastly, the communication between the different parts.

4.1 Approach

4.1.1 Project Approach

During this project, the group has been rotating the roles of leader and secretary. The reasoning is to distribute the work equally and let everyone try the different positions. The leader's tasks have been to organize a meeting for the steering group and update the Gantt diagram. The secretary's tasks have been to write minutes of meetings and write the progress report for every week.

Meetings with the steering group were held every even week, depending on the availability of the steering group. During these meetings, the group has discussed the progress of the project and received guidance and tips from the steering group on different issues the group has faced. Analysis of risks are performed in order to be prepared for worst case scenarios, and can be found in Appendix K. To ensure a good result and efficient approach a detailed project plan

was made, containing detail information on tasks and strategy. The full plan can be found in Appendix F, while a short summary of the plan is listed below.

- Gather information and knowledge about the specifications and equipment used in the project. Moreover, create an overview for the project group. The overview would prepare the whole group on what tasks to perform and how to execute them.
- Early start on software development. All the different tasks within this area will be started at once, divided amongst the group.
- Testing will be performed in two ways. Firstly testing during development will be executed, both separate and with test programs made for communication. When all the software components are finished, tests will be performed on the whole system.

4.1.2 Approach due to Covid-19

The lockdown rules stated that the schools (NTNU and Fagskolen) had to close, except for anyone needing lab equipment. Since the project entails testing with the setup equipment, the group could meet. Regarding the group member's health concerns and the severity of Covid-19, some chose to stay home during this lockdown period.

As Norway is on high alert since March 2020, meetings and guidance with supervisors, clients from Kongsberg Maritime, and support were conducted online.
The project was minorly affected by the pandemic. A few problems arose due to not meeting every day and discussing the project with group members in quarantine, which resulted in online meetings in this time-period.

4.2 Physical Assembly

Most of the equipment provided by Kongsberg Maritime came pre-assembled. However, some assembly was required to finish the Mcon rack. Found on the backside of the rack is the electrical panel that came semi-assembled. See Figure 4.1 for a visual representation of the electrical

panel when arriving. Standard I/O modules, power supplies, fuses, and marine controllers were mounted.

Mcon Rack

To complete the assembly (see Figure 4.2) it was required to connect the wires from the testjigg to the correct inputs and outputs at the Mcon rack. The coupling was done according to the I/O list. The testjigg was first connected to be able to run the FAT and knowing what had to be done in order for the thesis to be done.

When attaching the CPU and I/O modules, the testjigg wiring had to be disconnected for the new wiring from the CPU to be connected. When wiring the I/O modules from the external system, an I/O list (see Appendix H) was created to keep the connection points in order. The I/O modules were also wired in accordance with their data sheet as cited in section 3.5.

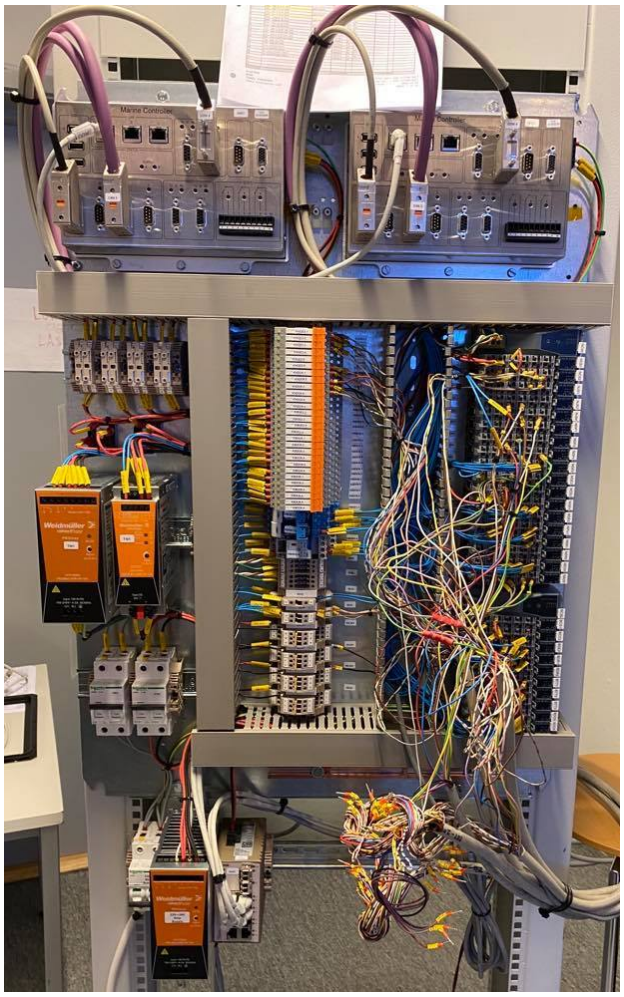


Figure 4.1: Wiring unfinished [46].

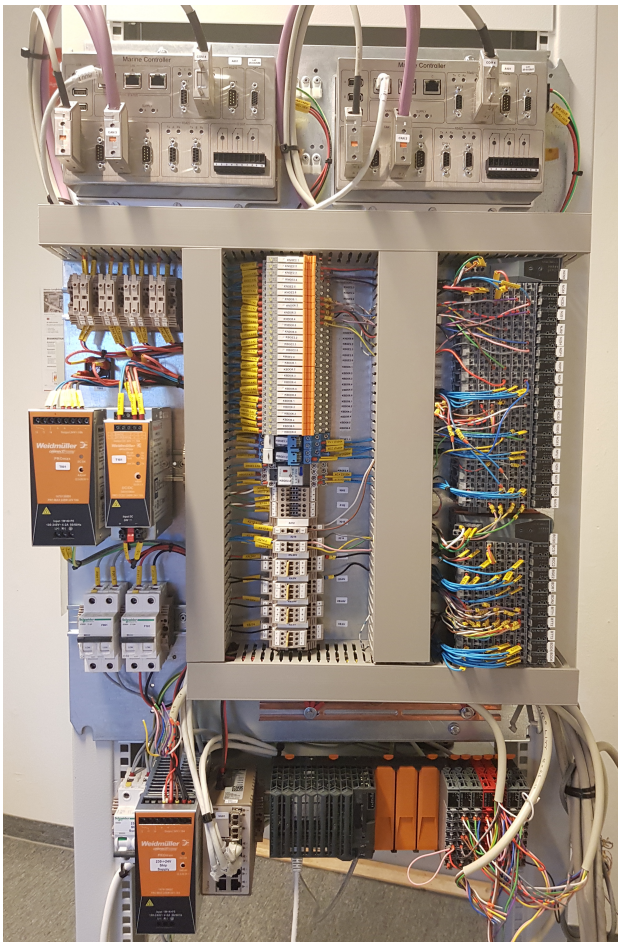


Figure 4.2: Wiring complete, with CPU and I/O modules attached and coupled [46].

Selection of I/O Modules

Based on limited experience with I/O module selection, and the various types of I/O modules B&R Automation offers, the selection of I/O modules are based on what seemed to fit the best for the project.

The signals required from Kongsberg Maritime set the base for how many channels was needed. The signals required from Kongsberg Maritime was as followed: Pump running (digital output (DO)), stop/start pump (digital input (DI)), thruster running (DO), stop/start thruster(DI), RPM order (analog input (AI)) and RPM feedback (analog output (AO)). It was originally chosen one DI and one DO for each function, i.e. one module for stop/start pump and one for stop/start thruster. This would also give enough channels per I/O module to add more functions as the

project went further along. Therefore, each I/O module had to have more than two channels, and the DI module had to have the most. The AI and AO modules also had to be configurable to be able to receive 4-20 mA.

Testjigg PU102

A testjigg PU102 unit was provided and came fully assembled from Kongsberg Maritime (see Figure 4.3 and Figure 4.4). This test unit would act as a simulation of the propeller until the final product was done. The testjigg unit consisted of numerous switches and dials that simulate certain actions performed by the propeller and its systems. The testjigg switches were used to simulate discrete signals such as *pump running, thruster running, DP enabled* etc. Dials were used to simulate continuous signals such as *RPM, DP RPM,* and *Joystick RPM*. Numbers and stickers marked the various signals to make the assembly of the electrical panel easier. In addition to the markings on the testjigg unit, an I/O list was provided to assist in the coupling of the electrical panel.

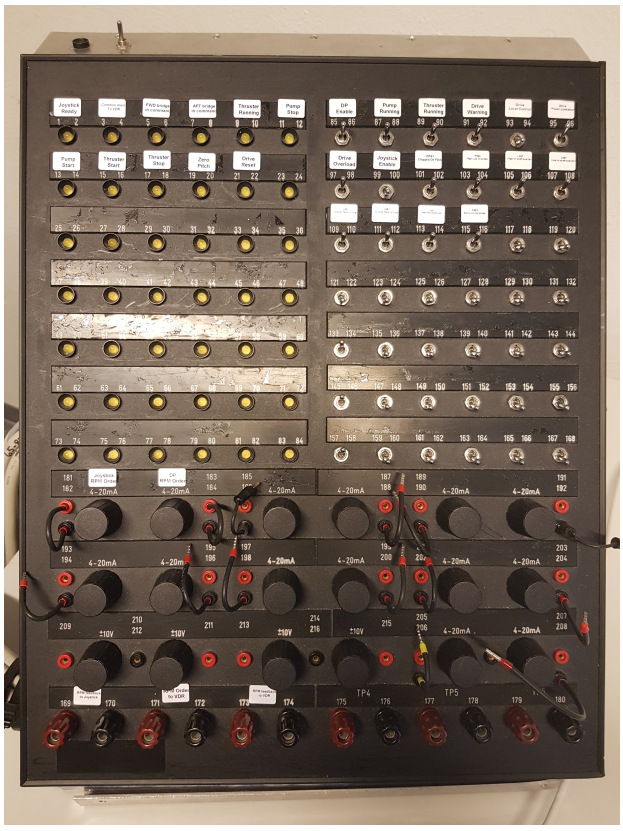


Figure 4.3: Panel testjigg [46].

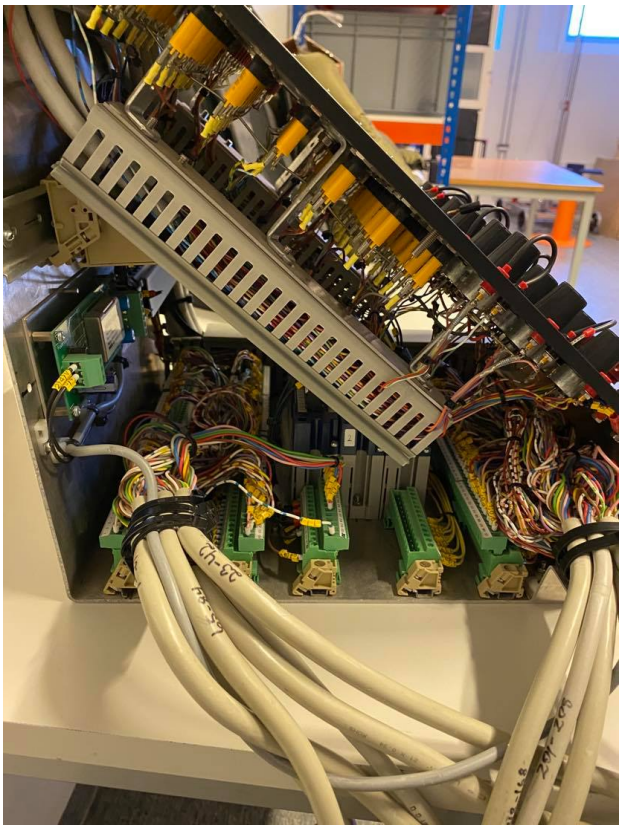


Figure 4.4: Wiring on the side of the testjigg [46].

Screens

After the coupling of the electrical panel and the testjigg were complete, the screens were mounted. The screens were two touch capacitive screens mounted on two separate brackets. The screens were connected to the rest of the equipment by an Ethernet port found under each screen (see Figure 4.5).

Figure 4.5: Underside of the screen [46].

Levers

The levers were inserted on the tabletop of the testjigg and screwed in place by four screws, as shown in Figure 4.6. Usually, the levers should be turned 90 degrees (horizontal) to be in the correct position because each lever is typically placed on each side of the person operating the levers. However, because of the nature of the project setup, a decision was made to mount the levers in a vertical position.

For connecting the levers to the rest of the system, some connectors marked with *NC* and *BC* were used. Each lever (see Figure 4.6) was equipped with one *NC* and one *BC* connector at the underside of the levers. These have been coupled according to the labeling on both the levers and the connectors and can be seen in Figure 4.7.

Figure 4.6: Levers [46].

Figure 4.7: Setup for levers [46].

Complete setup

The complete Mcon system at the beginning of the project can be seen in Image 4.8. The testjigg was replaced by a computer with digital buttons during the project, as seen in Image 4.9. Two more screens have as well been provided for use during testing of the system.

Figure 4.8: Complete setup at start of project [46].

Figure 4.9: Complete setup at end of project [46].

4.3 Programming the Logic in B&R Automation Studio

B&R Automation Studio was chosen for this project as it efficiently connects and reads/sends signals from/to the B&R Automation I/O modules, and it has everything needed for handling the logic and the GUI. B&R Automation Studio is also used by Kongsberg Maritime, which meant assistance could be provided from this resource.

B&R Automation Studio supports the IEC 61131-3 language. As the group members have mostly been programming software developments in Java/Python, the most appropriate language was Structured Text.

The logic programming in B&R Automation Studio was made for handling inputs and outputs from Mcon, and the signals were based upon what was required to run the semi-automatic FAT. The program was based on the order of the FAT, which would be the order of execution.

4.3.1 RPM Logic

The RPM signal handling had to be scaled correctly for the MATLAB®/Simulink® program. The value was scaled from INT to percent, since the MATLAB®/Simulink® model is based on values between -100 and 100. In order to get both negative and positive values a step scaling the value to 4-20 mA was added.

The RPM signal coming back into the logic had to be scaled back from percent to INT, as the

analog I/O modules are based on INT values. Figure 4.10 showcase the handling of RPM signals. 4.10

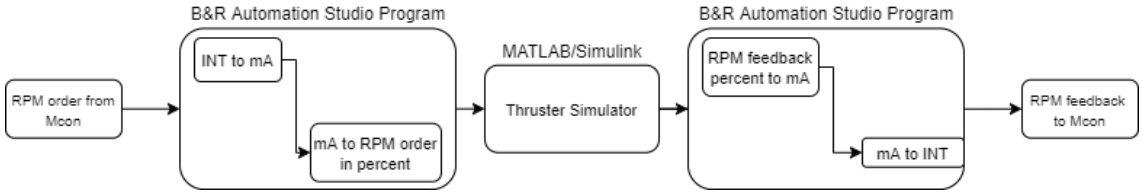


Figure 4.10: Logic for RPM. [46].

Firstly, the signal is scaled from INT (16 bit signed) to REAL (32-bit floating-point number); this was done by using a converter on the A_IN signal divided by 32767 called INT_TO_REAL. As seen in Eq. 4.3.1, the analog in signal, A_IN, is divided by the max INT value. The sum of the division will give a number between 0 and 1, which will then be multiplied by 16; 16 is the steps between 4-20 mA. The multiplication will give a number between 0-16; since the mA is 4-20, 4 is added to the equation to get the correct value.

$$STA\_IN = \frac{A\_IN}{32767} * 16 + 4 \tag{4.3.1}$$

The second step was to calculate the RPM order from mA to the percentage. As seen in Eq. 4.3.2, 12 is subtracted from the STA_IN value, giving a value between -8 and 8. This value was then divided by 8, giving a new value between -1 and 1, then multiplied by 100 to get the percent. This value was then fed into the MATLAB®/Simulink® model.

$$RPM\_Order = \frac{STA\_IN - 12}{8} * 100 \tag{4.3.2}$$

The third step was to calculate the value, RPM_Feedback, back to mA. RPM_Feedback is given in percent, and dividing this by 100 will give a value between -1 and 1. Multiplying by 8, then adding 8+4 will give a value between 4-20, as shown in Equation 4.3.3

$$STA\_OUT = \frac{RPM\_Feedback}{100} * 8 + 8 + 4 \tag{4.3.3}$$

The fourth step was to calculate the value sent out to the analog output module and then to Mcon. Using the STA_OUT value from the last equation and subtracting 4, then dividing by 16, giving a value between 0 and 1. This value was then multiplied by 32767 to get the INT. In equation 4.3.4, REAL_TO_INT has to be used on the whole Eq. 4.3.4 to convert it back to the original value type.

$$A\_OUT = \frac{STA\_OUT - 4}{16} * 32767 \tag{4.3.4}$$

Adding and subtracting *4* in Eq. 4.3.3 and 4.3.4 was done to make the values more apparent in regards to watching them in B&R Automation Studio.

4.3.2 Pump Logic

The logic program for pump running is based on two input signals, start and stop, and one output signal, pump running. The input signals were connected to TON functions to get a realistic delay when starting and stopping the pump.
When the start signal was received, the TON function was triggered, and the output from the TON triggered the pump running function, as illustrated in Figure 4.11. The same function applies to stopping the pump.
The timer has been set to 5 seconds, as this gives it enough time to see when it is triggered and turning on/off the pump, making it easier to troubleshoot if errors occur.

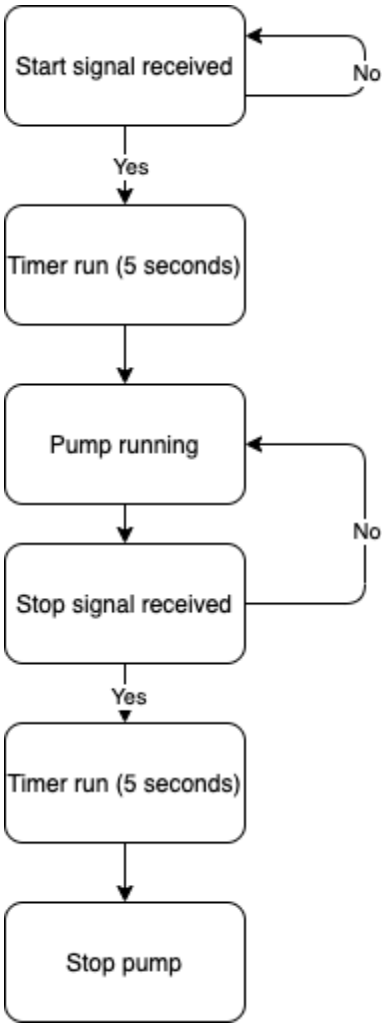


Figure 4.11: Logic for pump [46].

4.3.3 Thruster Logic

For the thruster to start, the pump has to be running. It is important to note that this is only a requirement for starting, not stopping the thruster.

As illustrated in Figure 4.12, the thruster logic consists of 4 cases (0-3). Case 0 will check if the pump is running and move to case 1 if it is. In case 1, there are two if-statements. The first if-statement states that if a start signal is received and the timer is elapsed, it will move onto case 2. The second if-statement will move the case back to case 0 if the pump has been turned off.

Case 2 will keep the thruster running until the elapsed time of a stop signal is received. When the thruster is off, the case will go to case 3 before going back to case 0, if the pump is off, or 1, if the pump is still on.

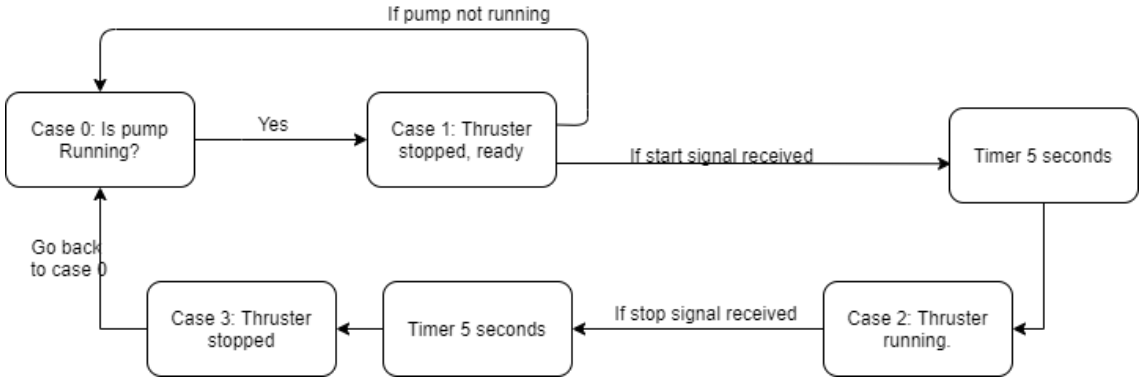


Figure 4.12: Setting the Datapoint of the button [46].

4.3.4 Drive Reset Logic

The Drive Reset signal is a pulse sent as an output from Mcon to an input to the CPU/B&R Automation Studio. In the FAT it is stated that it should only be verified that the digital contact is sent to Drive, as specified in Appendix E.

4.3.5 DP and Joystick Logic

The DP and Joystick signals are both sent from an external source, in this case, B&R Automation Studio, as illustrated in Figure 4.13.
As the signal is sent from B&R Automation Studio as an output, it should register in Mcon and trigger the DP Ready. This will set Mcon into DP mode, where the external source will control the RPM.

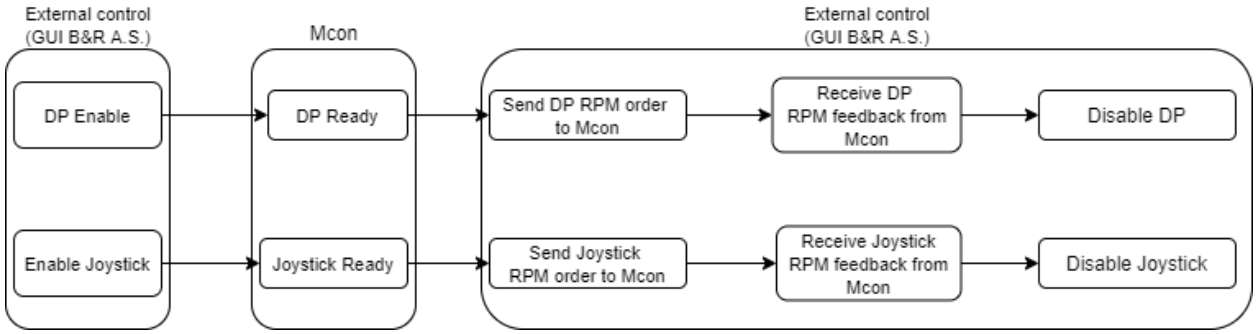


Figure 4.13: Logic for DP and Joystick. [46].

DP enable, DP ready and DP RPM order and feedback is displayed in the GUI. The feedback has no function in the GUI, but it is added as an extra control function.

4.3.6 I/O Mapping

The mapping of I/O modules had to be done in *Physical View* before opening the I/O modules and entering the variables, seen in Figure 4.15. The I/O list and an overview of which variable is being mapped to which I/O module can be found in Appendix H.

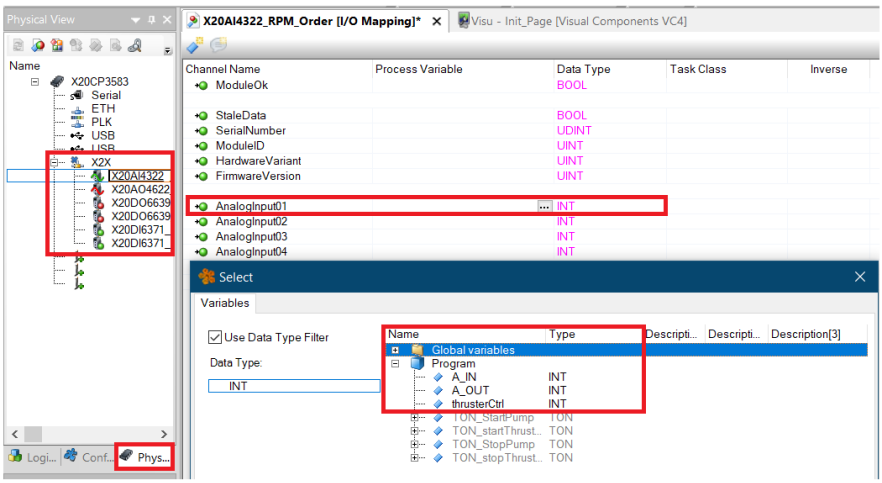


Figure 4.14: Mapping I/O [46].

The I/O modules also needed to be configured to the correct input values according to the I/O list provided by Kongsberg Maritime, see Figure 4.15. This needed to be done for both the AI and AO modules.

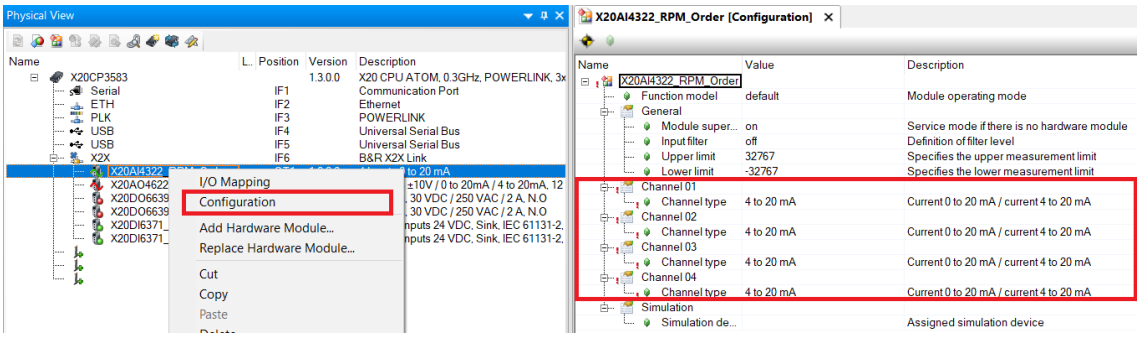


Figure 4.15: Configuring AI module [46].

Since stop pump is a NC loop, the signal had to be inverted for the logic to work, see Figure 4.16, this was done under *Physical View* , and entering *I/O Mapping* for the digital input I/O module.

Figure 4.16: Inverting a signal [46].

4.3.7 Simulation Mode

Simulation mode was activated by pressing the traffic light button or going to *Online* and *Activate Simulation*, as seen in Figure 4.17

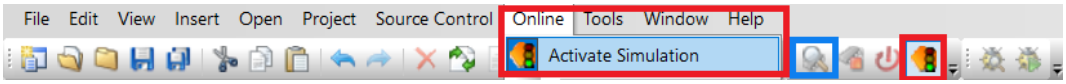


Figure 4.17: Activate Simulation [46].

When the simulation is running, it is possible to open the watch window, marked in blue in Figure 4.17. This window will show all the values in real-time.
The simulation mode has been actively used to test logic, GUI and the MATLAB®/Simulink® program.

4.4 GUI in B&R Automation Studio

For execution of the following steps, the VNC Viewer has to be downloaded beforehand, and B&R Automation Studio needs to be connected to the CPU or the simulation. The external

device will in testing be the computer/software program.

The visualization also needed to be made accessible. This was done under *ETH*, *Configuration* and then adding the *Visu* under *VC Mapping*, as illustrated in Figure 4.18

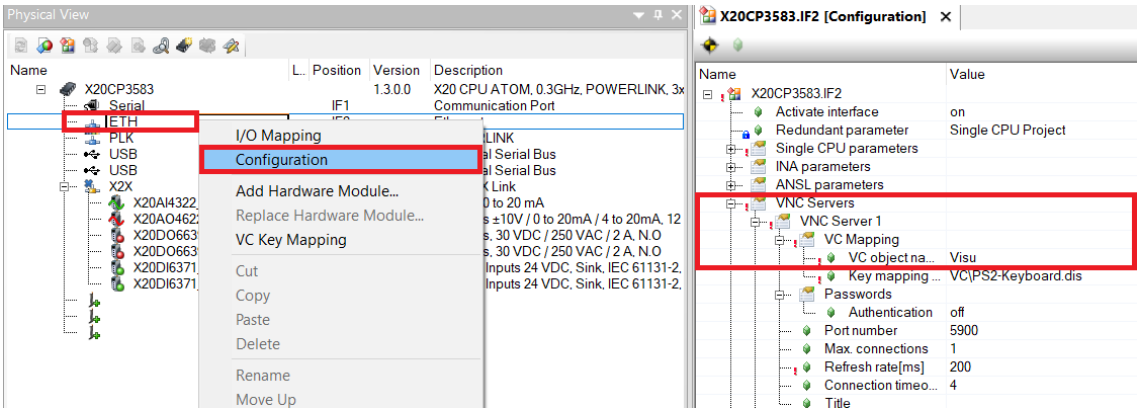


Figure 4.18: Making the GUI accessible [46].

4.4.1 Approach

The first step for making a visualization in B&R Automation Studio was adding a Visual Component while in Logical View. It was chosen VC4, then proceeded to choose the wanted resolution, 640x480 (VGA), and lastly, the *Basic* format was chosen.

The next step was to add buttons. After adding the buttons, they had to be connected to the variables in the program. For doing so it was necessary to go to *Keys*, then *Actions* and change the type to *ToggleDatapoint*, and lastly, adding the *Datapoint* to the wanted variable, as illustrated in Figure 4.19 and Figure 4.20.

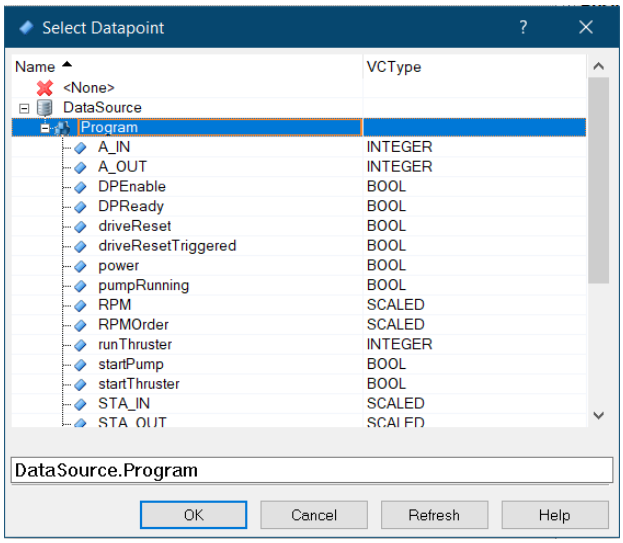


Figure 4.19: *Select Datapoint* entered from *Datapoint* [46].

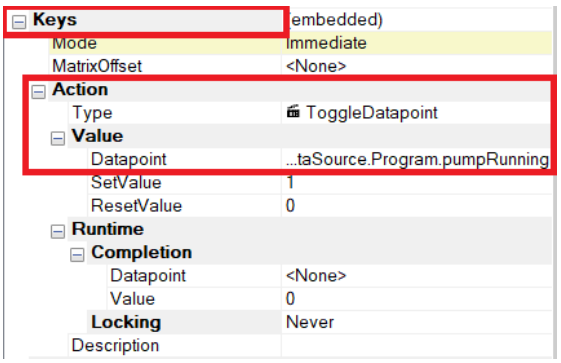


Figure 4.20: Datapoint added [46].

After the buttons had been connected to the variables in the program, a description was added to make them more apparent when the variables change. This was done under *Format*, changing the *TextSource* to *Single Text* and then adding the wanted text for the button under *Text*. The same was done under the tab *Pressed*, which would show what state it is when pressed (see Figure 4.21).

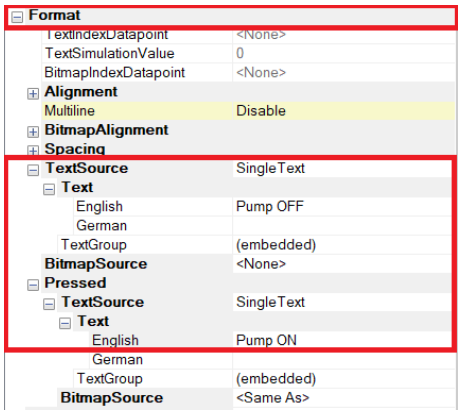


Figure 4.21: Setting the Datapoint of the button [46].

For adding the numerics, entering the datapoint was done under *Value*. The *StyleClass* also had to be changed, depending on if it was an output or input. Output numerics were made for DP and Joystick mode since these are controlled through an external device sending orders to the

Mcon system.

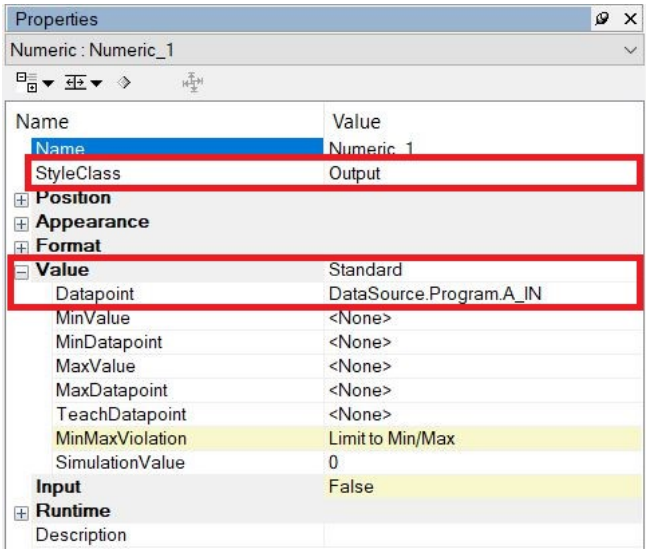


Figure 4.22: Adding numerics [46].

To be able to connect the VNC Viewer to the B&R Automation Studio GUI, a connection between the two had to be made. This was done in the VNC Viewer app, under *File* and *New Connection*, then adding the IP Address (see section 4.7.1.1) and name in the *General* tab (see Figure 4.23.) In the *Options* tab, the picture quality had to be set to High, as seen in Figure 4.24.

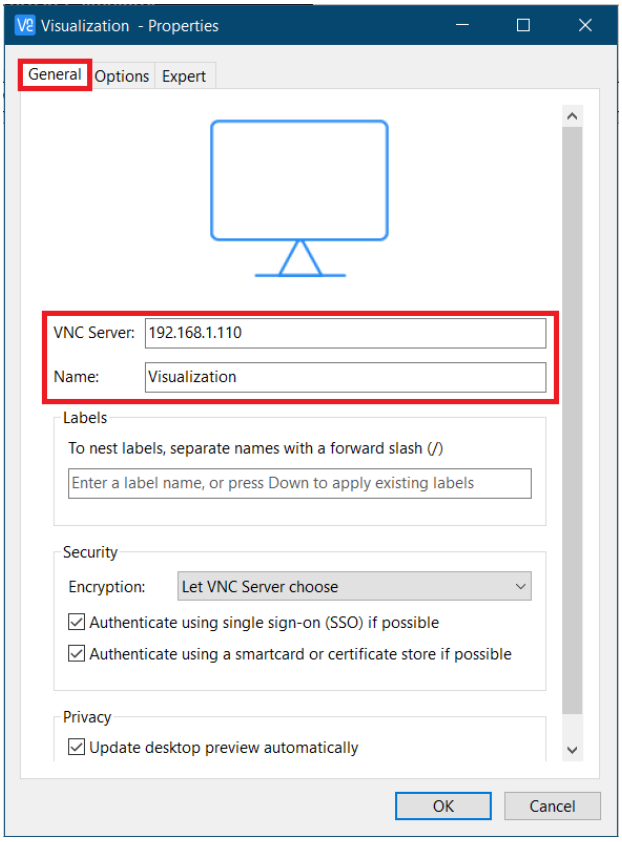


Figure 4.23: Set the VNC Server to the CPU's IP address [46].

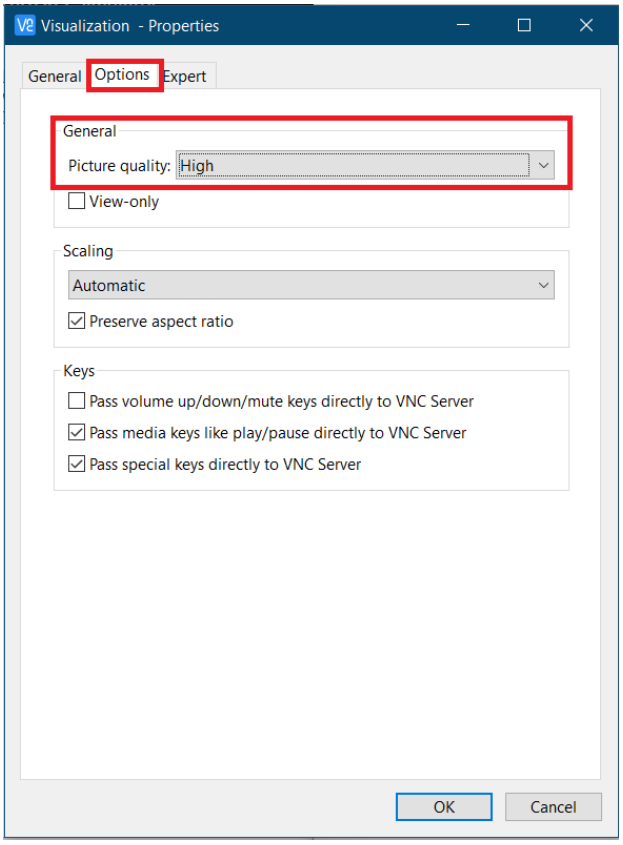


Figure 4.24: Set the *Picture Quality* to *High* [46].

4.5 Modelling FPP

Digital models of a DC motor and propellers were created in Simulink®, using MATLAB® as an init holding all the variables. The model created a natural response of the RPM build-up when operating the levers of the test station. Using the theoretical knowledge provided in the theory section, the following model (see Figure 4.25) demonstrates the block diagram of the FFP with a DC motor, gears and a PID regulator.

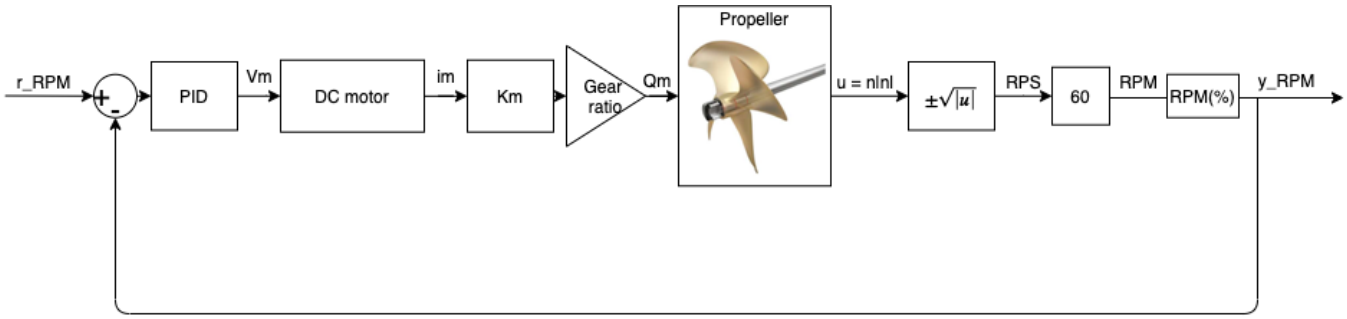


Figure 4.25: Model of FPP [46].

The reference signal is a value between -100 and 100, which represents the RPM order in percentage. The process of the model gives the actual RPM value, considering the moments of inertia. The DC motor converts the signal into a current, which is amplified by the motor constant. The output of the DC motor, with the constant, changes the value to motor torque. Then, the gear ratio is added. By using the output of the motor and gear ratio, the propeller calculates the $n|n|$. Taking the square root of the value gives RPS (n) in both negative and positive direction, multiplying with 60 to get the RPM. The final box changes the value into a percentage between -100 to 100. The PID regulator regulates the process and will eventually drive the error signal to zero and stabilize the system. Further information about the blocks and the plant is elaborated in the following sub-chapters.

4.5.1 DC Motor

A simplified model of the DC motor was created in MATLAB®, consisting of a torque constant, K_m, and a time constant, τ_m. Due to the speed of the electric DC motor, assumptions were made to simplify the modeling of the DC motor. The transfer function for the DC motor can be seen in Figure 4.26.

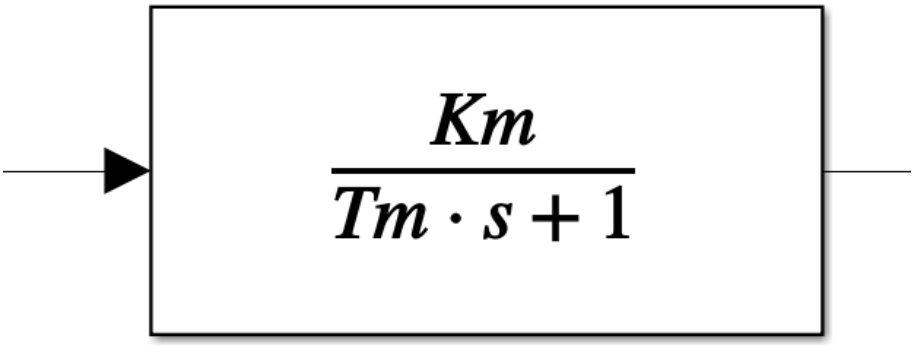


Figure 4.26: Transfer function for DC motor [46].

A DC motor was chosen to simulate the motor; even though, in reality, an AC motor is used. Since the principal purpose of this model was to simulate RPM feedback realistically, it was agreed between the group and Kongsberg Maritime that the focus should be on creating a simple DC motor. This motor is much less complex to model than an AC motor.

Regarding the choice of the complexity of the motor, a 1. order transfer function was chosen. Firstly, the constant option was eliminated, as it would not create the desired result of the motor. Both 1. order and 2. order transfer function options could have been chosen since both give similar results in the working area of the engine, which is within the bandwidth. Also, the two options give relatively similar results in low frequencies but different results on high frequencies. The 1. order was therefore chosen as it would give the desired results with minimum complexity.

4.5.1.1 Finding K_M and τ_m

Figure 4.27 shows the code used in MATLAB® to inspect the transfer function.

```
%DC MOTOR

%1. order system
     Km = 1;         % Torque constant
     Tm = 1;         % Time constant

    TF_motor = tf(Km,[Tm 1]);
    stepplot(TF_motor) %test to see if system behaves as intended
```

Figure 4.27: Code used to find the optimal step response of simplified DC motor model [46].

A step signal was used to perturb the system to check if the transfer function behaved as a simplified version of a DC motor.

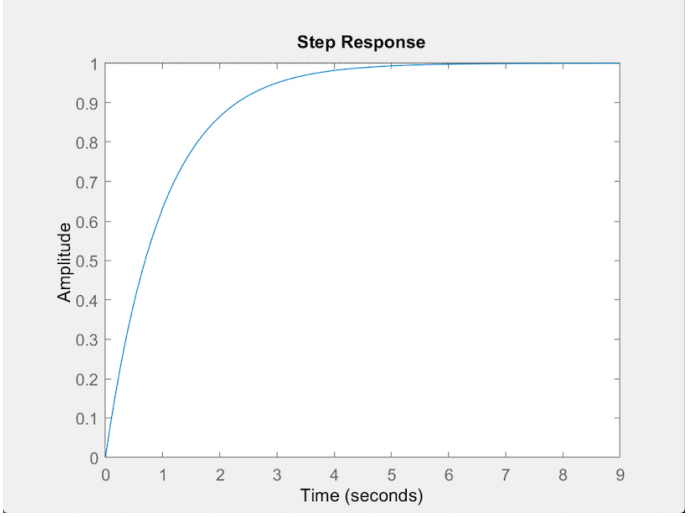


Figure 4.28: Step response for simplified DC motor [46].

After the DC motor was tested, the proper magnitude of the transfer function was set. The magnitude of the system was decided based on a data sheet from Kongsberg Maritime (see TTC 80 CP in Appendix C). The only deciding parameters were thruster force measured at 243 kilonewton meter or 243 000 newton meter, and settling time of the system. Given an input range of -100 % to 100 %, the following calculations were made to decide the magnitude of the DC motor.

$$\frac{Max\ engine\ torque}{Max\ input} = \frac{243000Nm}{100\%} = 2430 \tag{4.5.1}$$

The calculations in equation 4.5.1 shows that the transfer function needed to produce 2430 Nm for each percent of input from the levers. A Bode plot of the transfer function was used to control that the right gain was obtained. The magnitude in a Bode plot is represented in decibel (dB), so the magnitude found in equation 4.5.1 would therefore have to be transformed into dB first.

$$dB = 20 * log(2430) = 67.71dB \tag{4.5.2}$$

After the gain was converted into dB, the *Bode* function was used in MATLAB® to inspect the

magnitude of the transfer function.

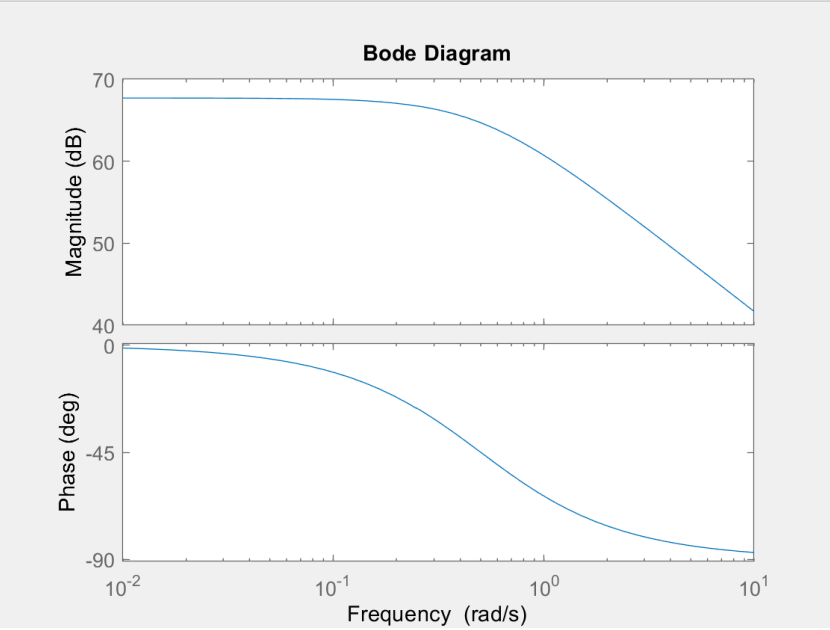


Figure 4.29: Bode plot of the transfer function displaying the magnitude and phase of the DC motor [46].

After the magnitude of the Bode diagram was confirmed to be at the correct level, the time constant, τ_m, needed to be calculated. For first order transfer functions, one time constant equals 63% of the final value. Settling time is equal to 4 time constants. The following calculations was therefore used to decide the time constant for the transfer function.

$$\tau_m = \frac{Settling\ time}{4} = \frac{8\ seconds}{4} = 2\ seconds \tag{4.5.3}$$

The calculated magnitude and time constant resulted in the following transfer function for the DC motor.

$$\frac{magnitude}{time\ constant + 1} = \frac{K_m}{\tau_m s + 1} = \frac{2430}{2s + 1} \tag{4.5.4}$$

The tests to verify if the DC motor values were correct can be found in section 5.1.4.1 of results.

4.5.2 Gear Ratio

In addition to the DC motor, a gearbox was needed as the max RPM rating of the DC motor and engine did not match. By introducing a gearbox with a gear ratio, both the Torque and the RPM of the propeller can be controlled relative to the DC motors torque- and RPM output. For the sake of simplicity, factors such as frictional loss and backlash are excluded from the gear ratio.

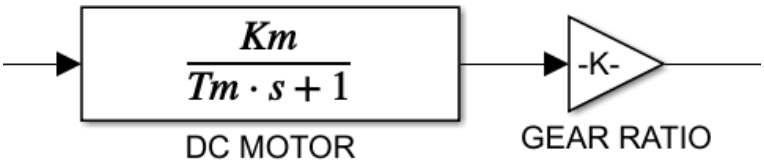


Figure 4.30: overview of DC motor and gears in Simulink® model [46].

Formula 4.5.5 is used calculate the gear ratio.

$$Gear\ ratio = \frac{Max\ propeller\ RPM}{Max\ motor\ RPM} \tag{4.5.5}$$

4.5.3 Propeller

A simplified model of a propeller was made by using the Wageningen B-series propeller. The calculated thrust- and torque coefficients were retrieved from a code made by Thor I. Fossen. The table for the Wageningen B-series propeller can be found in Appendix B.

For simplification and assuming that the propeller is in water but is not moving forward, only the first quadrant is modeled in both positive and negative direction. Factors such as ventilation, cavitation and slippage were not considered when designing this model.

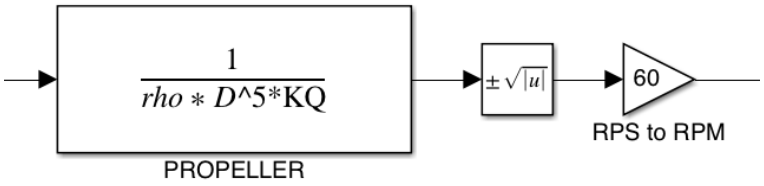


Figure 4.31: Simulink® blocks representing the propeller dynamics with output in the form of RPS [46].

For the propeller transfer function *rho* represents the water density of salt water, D represents the propeller diameter, and KQ represents the torque coefficient. Theory about the KQ is presented in section 2.5.1. D was decided to be 2.4m as it is obtained from a data sheet of tunnel thrusters from Kongsberg Maritime (see Appendix C). The transfer function for the propeller in Figure 4.31 can be described by solving equation 2.5.3 for torque produced with respect to n, which represents the angular velocity of the propeller in RPS. By solving the equation with respect to n, the following equation was obtained.

$$n = \sqrt{\frac{Q_p}{\rho D^5 K_q}} \tag{4.5.6}$$

Where n represents the angular velocity in RPS. By multiplying n with 60, the RPM will be the output. The equation for the propeller RPM can therefore be written as

$$Propeller\ RPM = (\sqrt{\frac{Q_p}{\rho D^5\ K_q}}) * 60 \tag{4.5.7}$$

4.5.4 PID

The block *PERCENTAGE* in Figure 4.32 convert RPM into percentage in order for the B&R Automation Studio software to process the signal and the PID regulation.

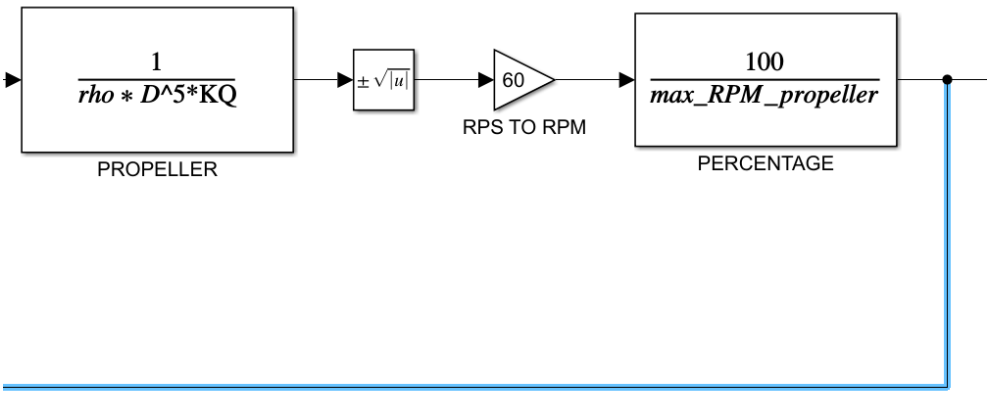


Figure 4.32: Conversion from propeller RPS to RPM and percentage of RPM [46].

The blue highlighted line in Figure 4.32 represents the signal that was fed back into the PID controller. The feedback signal was then subtracted from the original signal in order to regulate the system. Figure 4.33 shows how the feedback signal is connected to form a feedback-loop.

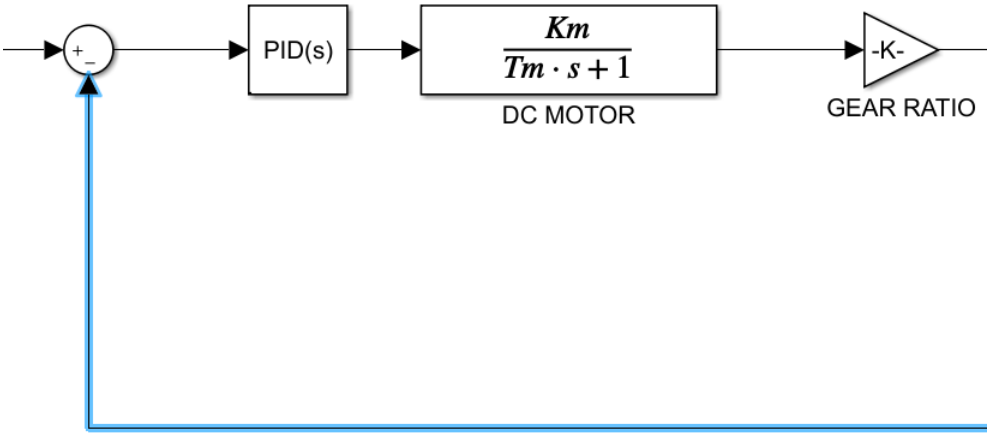


Figure 4.33: Feedback signal is connected to a block that subtracts the feedback from the original signal [46].

4.5.4.1 Finding PID parameters

Different approaches for tuning of the PID controller was tested out, such as Ziegler-Nichols and the auto-tune function implemented in Simulink®. None of these approaches gave a desirable

outcome. Therefore, so the PID controller had to be tuned manually. The disadvantage with this technique is that it takes a lot of time. No saturation of the PID or DC motor transfer functions were used when tuning the system. The testing and tuning of the PID controller and the system can be seen in section 5.1.4.

After the PID parameters were tuned to the desired values, tests were performed. The tests were performed both in simulation mode on the B&R software, and real life testing with the physical equipment. Results of the simulation testing can be seen in section 5.1.5. Results from the real life tests can be seen in section 5.2.4.

4.5.5 Modelling CPP

In addition to the FPP model, a model for controllable pitch was created. This is a simplified model of a controllable pitch propeller. It is not mathematically accurate, however the model simulates the behavior of a controllable pitch propeller. The purpose of this model was to simulate how RPM build-up, resistance, and thrust produced would vary based on the pitch angle of the propeller. The model was created in Simulink®, and is ready to be implemented in the B&R software. This model was only tested in simulation with Simulink®, and was not connected to the B&R software or the Mcon controller.

The CPP is a continuation of the FPP model, and therefore the system responsible for the RPM output was largely unchanged, except for some small additions. The CPP model is a more complex model with more components involved, therefore the structure of the model was a bit more complicated. In Figure 4.34, an overview of the CPP model can be seen. Figure 4.35 represents the subsystem *PROPELLER DYNAMICS* in the CPP model.

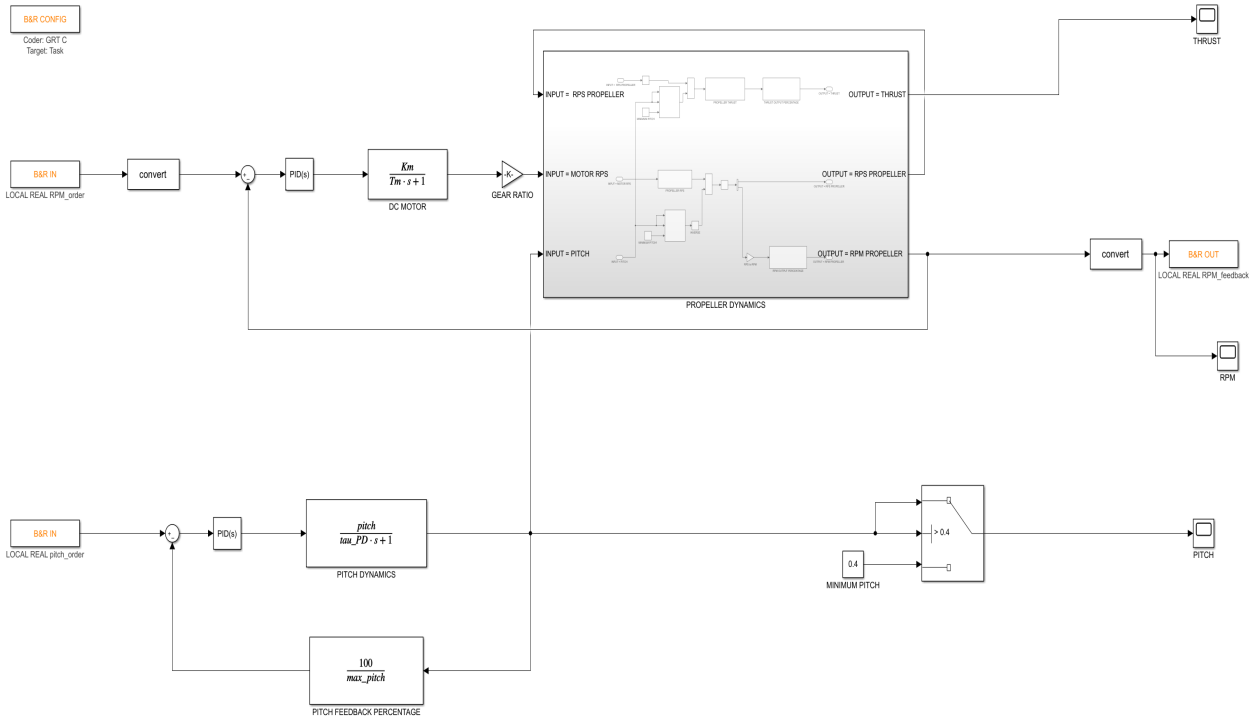


Figure 4.34: Overview of CPP model created in Simulink® [46].

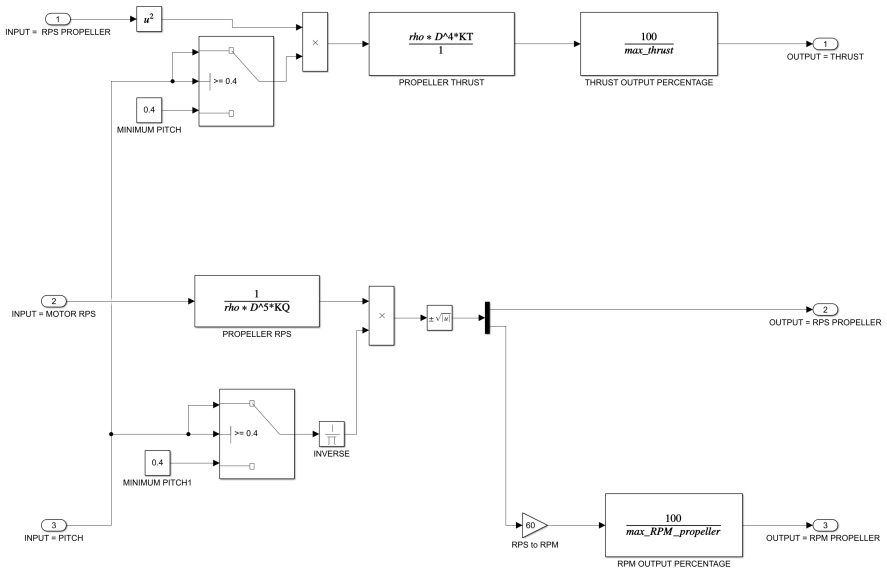


Figure 4.35: Overview of subsystem *PROPELLER DYNAMICS* in CPP model [46].

As seen from Figure 4.35, some minor changes was applied to the RPM loop. The only difference made between the FPP and CPP model in regards to the RPM was the implementation of the

inverse value of the pitch. The inverse value of the pitch was merged with the *propeller RPS* block in order to simulate more resistance with a greater pitch value. The inverse value was connected to a switch, as shown in Figure 4.36. The switch was introduced in order to regulate the minimum and maximum values allowed for the pitch.

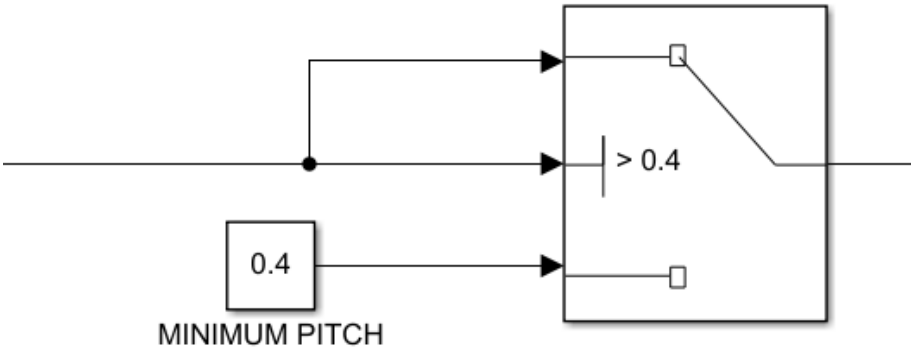


Figure 4.36: Switch that limits the working area of the pitch value to 0.4-1.4 [46].

Output 2 of the subsystem represents the propeller RPS(n), and was fed back into input 1 of the subsystem. The reason for the feedback of the RPS was because the values were needed in order to calculate the value of the thrust produced, as can be seen in equation 4.5.8 for thrust produced.

$$T_p = \rho D^4 n|n|K_t \tag{4.5.8}$$

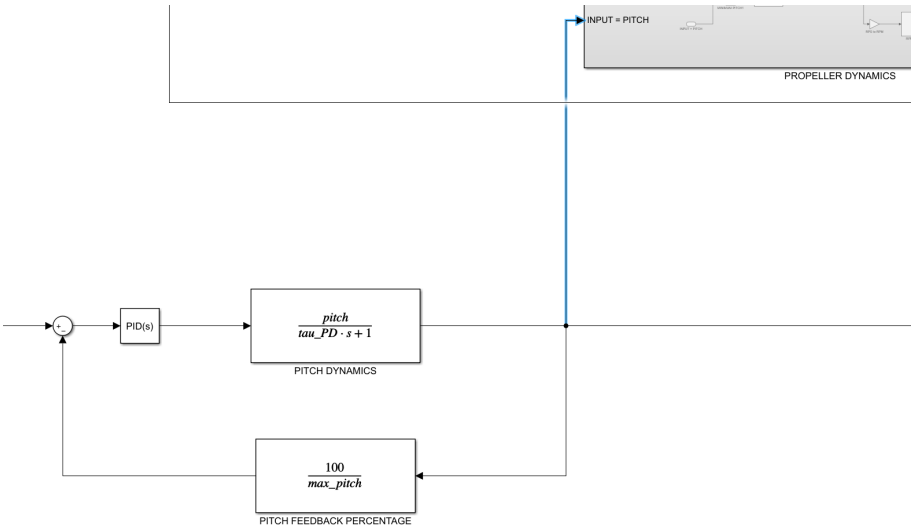


Figure 4.37: Pitch control system [46].

The pitch control system consisted of a simplified transfer function with a time constant *tau_PD*, in order to simulate the pitch actuator. A feedback loop with a PID controller was also implemented. The output of the pitch was fed into the *PROPELLER DYNAMICS* subsystem, as can be seen from the blue line in 4.37.

Figure 4.38 (blue highlighted line) illustrates how the pitch input was used as an inverse value in the *propeller RPS* block to simulate resistance for the propeller. The original pitch value was used to simultaneously increase thrust produced by the *propeller thrust* block, as the propeller resistance increased.

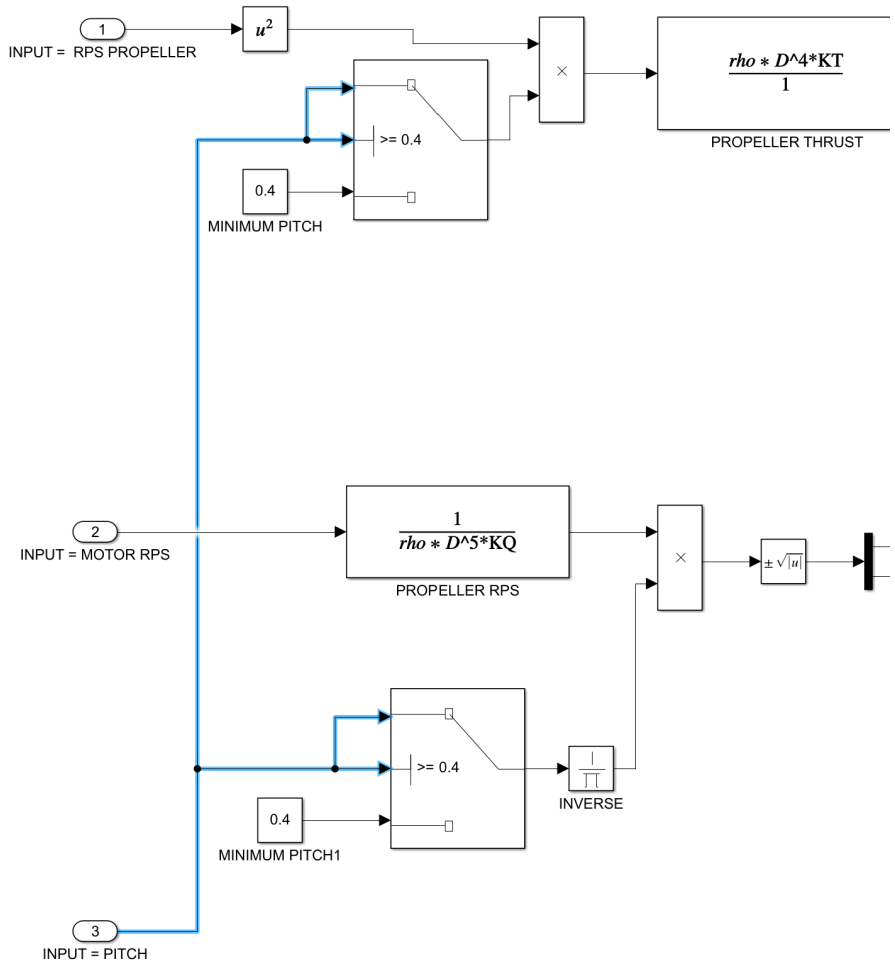


Figure 4.38: Pitch input of the *PROPELLER DYNAMICS* subsystem [46].

After the CPP model was built, the PID controllers for the DC motor and pitch actuator had to be tuned in order for the system to behave as intended.

4.6 Software Development in Java

Multiple programming languages are suitable for creating the FAT. When choosing a language, it was desired to choose a well-known language that the user was experienced in.

The Java code consists of one coding project and contains the client-side for communication with B&R Automation Studio. The code also contains read-functions, the FAT, as well as a function for printing the test results. The project has been created in GitHub for version control and

sharing the code between the group members.

4.6.1 Files

The Java project consists of multiple files:

- Interface ClientExample
  Interface for establishing a connection between the OPC UA server and OPC UA client. The port number of the server and the security policy are defined here.

- Interface Connection
  This file is an interface, and the purpose of this file is to make sure that the file "TestRunner" can run, likewise for both the files "RunTestsMilo" and "RunTestsVirtual". The "Connection" file must contain the get-methods for all the tests in the code project.

- Class CreatePDF
  This class creates and updates a PDF file. It receives the test result parameters from "TestRunner" and creates a PDF file containing these results. The results will be shown as text in different colors based on the result of the test.

- KeyStoreLoader
  Class responsible for enabling security and certificates by default, used for establishing an OPC UA connection.

- Class MiloClient
  "MiloClient" is a class accompanying the class "RunTestsMilo". It is responsible for reading data between Java and B&R Automation Studio. Therefore, this class contains get-methods for each of the tests.

- Class RunTestsMilo
  This class is connecting the Java code with B&R Automation Studio. It connects using an endpoint, which consists a port number for connection. After the connection, the class "TestRunner" is called.

- Class RunTestsVirtual

  This class is used for running the code virtually. This means that the code can be run without connection to B&R Automation Studio. This is necessary for testing the code. After the virtual connection is established, the class "TestRunner" is called.

- Class TestRunner

  "TestRunner" is the class for running all the FAT and checking if they are passed.

- Class VirtualConnection

  This class is accompanying the class "RunTestsVirtual" and contains get-methods for each of the tests.

- pom.xml

  File created by turning the project into a Maven project, used for implementing libraries.

- PDF file Fat-test.pdf

  PDF file used as a template for printing out the test results.

- PDF file Fat-test-result.pdf

  PDF file updated with the test results, using the template "Fat-test.pdf".

4.6.2 Tests

The tests are created in the class "TestRunner", where the requirements to pass a test are located. In order to read values as they change in B&R Automation Studio, a while loop is created in this class, running through all the tests continually. When the requirements for a test are fulfilled, the test is completed, and the while loop will continue to check for the remaining tests. The while loop is shown in Image 4.39.

```
while (keepRunning) {
    try {
        Thread.sleep( millis: 100);
        System.out.println("Press enter to quit");
    } catch (InterruptedException ex) {
        System.out.println("InterruptedException...");
    }
    try {
        if (reader.ready()) {
            int readKey = reader.read();
            if (readKey == 10) {
                keepRunning = false;
                System.out.println("Quitting..");
            }
        } else if (numberOfTestsFinished == 6) {
            keepRunning = false;
            System.out.println("Quitting..");
        }
        if (!rpmControlTestIsFinished) {
            rpmControlTestResult = rpmControlTest();
        }
        if (!remoteStartStopThrusterMotorTestIsFinished) {
            remoteStartStopThrusterMotorTestResult = remoteStartStopThrusterMotorTest();
        }
        if (!remoteStartStopThrusterServoPumpsTestIsFinished) {
            remoteStartStopThrusterServoPumpsTestResult = remoteStartStopThrusterServoPumpsTest();
        }
        if (!resetDriveTestIsFinished) {
            resetDriveTestResult = resetDriveTest();
        }
        if (!dpInterfaceTestIsFinished) {
            dpInterfaceTestResult = dpInterfaceTest();
        }
        if (!joystickTestIsFinished) {
            joystickTestResult = joyStickTest();
        }
        testsFinished();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Figure 4.39: While loop in class "TestRunner" for reading variables continually [46].

For implementing the tests, one interface and four classes needed to be updated. Below is the implementation of *rpmControl* test explained.

Interface "Connection" needs to create get-methods for each test as shown in Image 4.40. This interface is needed for both the actual connection and the virtual connection to run the code equally.

```
/**
 * getting the value of rpmControl
 * @return the value of rpmControl
 */
int getRpmControlValue();
```

Figure 4.40: Code needed in interface "Connection" for implementing tests [46].

Class "MiloClient" also needs to create get-methods for each test, as illustrated in 4.41. The *NodeId*, retrieved from UaExpert, needed to be specified before being sent to function *getVariable* illustrated in Image 4.7.3.2.1. After that, the variable's value is turned into an *int* or *boolean* and sent back to class "TestRunner".

```
/**
 * getting value of rpmControl
 * @return the value of rpmControl
 */
public int getRpmControlValue() {
    NodeId objectId = NodeId.parse("ns=6;s=::Program:A_IN");
    Object value = getVariable(objectId);
    int calculatedValue = ((Short) value).intValue();
    System.out.println("RPM: " + calculatedValue);
    return calculatedValue;
}
```

Figure 4.41: Code needed in class "MiloClient" for implementing tests [46].

Class "VirtualConnection" needs to initialize a variable representing the value of the representative variable to be tested. This is where the value of a variable is saved when B&R Automation Studio is not connected. It also needs get-methods for all of the individual tests.

```java
/**
 * class for creating a virtual connection
 */
public class VirtualConnection implements Connection {
    public int rpmControlValue;
    public boolean thrusterMotorOn;
    public boolean thrusterMotorStartOn;
    public boolean thrusterMotorStopOn;
    public boolean thrusterPumpsOn;
    public boolean thrusterPumpsStartOn;
    public boolean thrusterPumpsStopOn;
    public boolean resetDriveOn;
    public int dpInterfaceValue;
    public boolean dpInterfaceOn;
    public int joystickValue;
    public boolean joystickOn;

    /**
     * getting value of rpmControl
     * @return the value of rpmControl
     */
    public int getRpmControlValue() {
        return rpmControlValue;
    }
```

Figure 4.42: Code needed in class "VirtualConnection" for implementing tests [46].

For each test to be run, the class "TestRunner" needs to initialize the variables to be tested. Variable *rpmControlTestResult* represents the result of the test and *rpmControlTestIsFinished* advises if a test has been completed.

```java
/**
 * class for running tests
 */
public class TestRunner {
    public boolean thrusterMotorStartIsFinished;
    public boolean thrusterMotorStopIsFinished;
    public boolean thrusterPumpsStartIsFinished;
    public boolean thrusterPumpsStopIsFinished;

    public boolean rpmControlTestIsFinished;
    public boolean remoteStartStopThrusterMotorTestIsFinished;
    public boolean remoteStartStopThrusterServoPumpsTestIsFinished;
    public boolean resetDriveTestIsFinished;
    public boolean dpInterfaceTestIsFinished;
    public boolean joystickTestIsFinished;

    public boolean rpmControlTestResult;
    public boolean remoteStartStopThrusterMotorTestResult;
    public boolean remoteStartStopThrusterServoPumpsTestResult;
    public boolean resetDriveTestResult;
    public boolean dpInterfaceTestResult;
    public boolean joystickTestResult;

    public int numberOfTestsFinished = 0;
    public Connection client;

    public TestRunner(Connection client) {
        this.client = client;
    }
```

Figure 4.43: Code needed in class "TestRunner" for implementing tests [46].

4.6.2.1 Explanation of the Individual Tests

Before printing the results, the class "TestRunner" needs to run the tests itself. These tests are based on the FAT Appendix E. Each test is retrieving values of variables from either class "Milo-Client" or class "VirtualConnection."

4.6.2.1.1 RPM Control, DP Interface and Joystick Interface

For the *rpmControl* (shown in Image 4.44), *dpInterface* and the *joystickInterface* tests, the structure of the code is the same. All the tests are using int RPM values, ranging from 0 to 32767, and the last two tests also use buttons. The tests start by reading the value of all the relevant variables. When pushing the relevant button at B&R Automation Studio GUI, a short pause gives the RPM values time to change. If the RPM value has changed more than a range of 100, the test is passed and finished. The small number range ensures that the test does not pass by

any disposition of the levers.

```
/**
 * test five
 * @return returns the result of the test
 */
public boolean rpmControlTest() {
    int rpmControlValue = client.getRpmControlValue();
    boolean testOK = false;

    int highestValue = rpmControlValue + 100;
    int lowestValue = rpmControlValue - 100;

    try {
        Thread.sleep( millis: 100);
    } catch (InterruptedException ex) {
        System.out.println("InterruptedException...");
    }

    if (client.getRpmControlValue() < lowestValue || client.getRpmControlValue() > highestValue) {
        testOK = true;
        numberOfTestsFinished++;
        rpmControlTestIsFinished = true;
    }
    return testOK;
}
```

Figure 4.44: Code needed in class "TestRunner" for creating *rpmControl* test [46].

4.6.2.1.2 Remote Start/Stop of Thruster Motor and Servo Pumps

The test *thrusterMotor* and test *thrusterPumps* are also similar to each other. Both tests require three variables from B&R Automation Studio and consist of two parts. Part one, seen in Image 4.45, is to check if *thrusterMotor*/*thrusterPumps* can be turned *on* successfully. This is done by setting a variable *start* to *true* if pushing the corresponding start button at Mcon. If *thrusterMotor*/*thrusterPumps* after an intermission is *on*, the part test is completed.

The second part, seen in Image 4.46, is to check if *thrusterMotor*/*thrusterPumps* can be turned *off*. Here the same principle occurs, setting the variable *stop* to *true* if pushing the corresponding stop button at Mcon. This part is completed if *thrusterMotor*/*thrusterPumps* is turned *off* after an intermission. When both the part tests have been completed, the test is passed and finished.

```java
/**
 * test seven
 * @return the result of the test
 */
public boolean remoteStartStopThrusterMotorTest() {
    boolean thrusterMotorIsOn = client.getThrusterMotorValue();
    boolean thrusterMotorStartIsOn = client.getThrusterMotorStartValue();
    boolean thrusterMotorStopIsOn = client.getThrusterMotorStopValue();
    boolean testOK = false;
    boolean start = false;
    boolean stop = false;

    if (thrusterMotorStartIsOn) {
        start = true;
    }
    try {
        Thread.sleep( millis: 100);
    } catch (InterruptedException ex) {
        System.out.println("InterruptedException...");
    }
    if (start && !thrusterMotorStartIsFinished) {
        if (thrusterMotorIsOn) {
            thrusterMotorStartIsFinished = true;
        }
    }
```

Figure 4.45: Code needed in class "TestRunner" for creating *thrusterMotor* test part 1 [46].

```
    if (thrusterMotorStopIsOn) {
        stop = true;
    }
    try {
        Thread.sleep( millis: 100);
    } catch (InterruptedException ex) {
        System.out.println("InterruptedException...");
    }
    if (stop && !thrusterMotorStopIsFinished) {
        if (!thrusterMotorIsOn) {
            thrusterMotorStopIsFinished = true;
        }
    }

    if (thrusterMotorStartIsFinished && thrusterMotorStopIsFinished) {
        testOK = true;
        numberOfTestsFinished++;
        remoteStartStopThrusterMotorTestIsFinished = true;

    }
    return testOK;
}
```

Figure 4.46: Code needed in class "TestRunner" for creating *thrusterMotor* test part 2 [46].

4.6.2.1.3 Drive Reset For *resetDrive* test, the value of the *resetDrive* variable is being gathered and checked. If this value is *on*, the test is passed and finished, as seen in Image 4.47.

```
public boolean resetDriveTest() {
    boolean resetDriveIsOn = client.getResetDriveValue();
    boolean testOK = false;

    if (resetDriveIsOn) {
        testOK = true;
        numberOfTestsFinished++;
        resetDriveTestIsFinished = true;
    }

    return testOK;
}
```

Figure 4.47: Code needed in class "TestRunner" for creating ninth test [46].

4.6.3 Print PDF

To present the FAT results, a PDF file is created and printed for showcasing all of the individual test results. The automatic printing of the PDF ensures that the results of the FAT are both clear and detailed. By writing the results onto a PDF file, no coding experience is required to perform this test or read the results. Saving and storing is also enabled of each test result for easy logging of the test result history.

The creation of the PDF file is inspired by the Apache PDFBox library [79]. This open-source Java tool is used to create PDF files and manipulate existing PDF files. Six files from the library collection were required and downloaded. For keeping the code project clean, a lib folder was created for storing the libraries.

4.6.3.1 Setting up the Initial PDF

It was necessary to implement some JAR files for creating the PDF file (see Figure 4.48).

- Open *File* from the toolbar in IntelliJ
- Click on *Project Structure*
- Select *Modules* at the left panel
- Click on *Dependencies* tab
- Click on the "+", and select *JARs or directories*
- Mark the six libraries and click *Open*

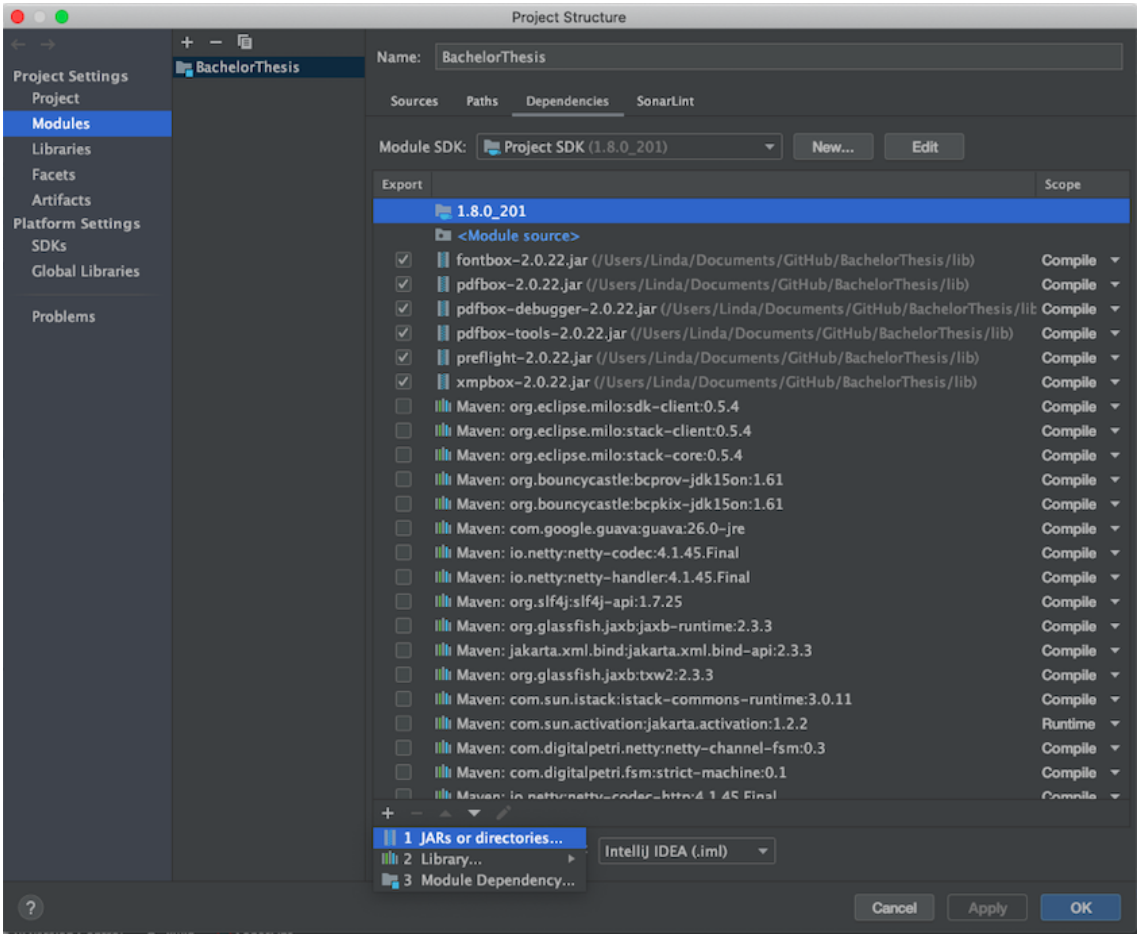


Figure 4.48: Steps for implementing JAR files [46].

After this, the project needed to be converted into Maven.

- Right click the main folder for the Java project
- Click on *Add Framework Support* (see Figure 4.49)
- Click on Maven
- Add required code in "pom.xml" from tutorialspoint [98], and specify *groupId* in file "pom.xml" (see Figure 4.50)
- Add dependencies for the six libraries in "pom.xml" (see Figure 4.51)
- Click the Maven-tab to the right in IntelliJ

- Click on *lifecycle*
- Double click *install* to generate the JAR files (see Figure 4.52)

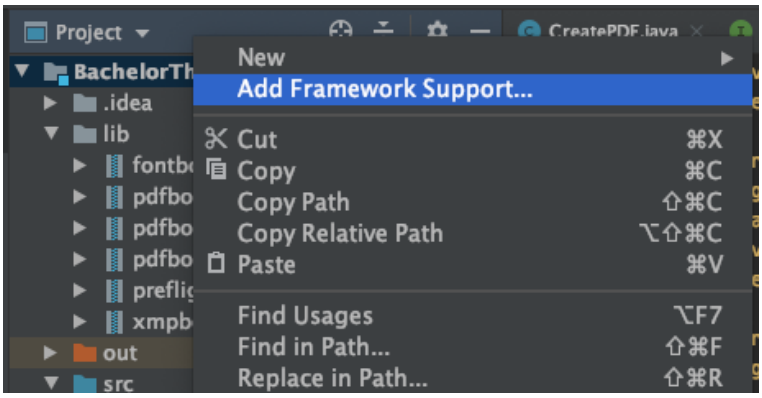


Figure 4.49: First steps for turning the Java project into Maven [46].

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>FAT-test-report</groupId>
    <artifactId>BachelorThesis</artifactId>
    <version>1.0-SNAPSHOT</version>

    <build>
        <sourceDirectory>src</sourceDirectory>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.3</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Figure 4.50: Required code in file "pom.xml" [46].

```
<dependency>
    <groupId>org.apache.pdfbox</groupId>
    <artifactId>pdfbox</artifactId>
    <version>2.0.1</version>
</dependency>

<dependency>
    <groupId>org.apache.pdfbox</groupId>
    <artifactId>fontbox</artifactId>
    <version>2.0.0</version>
</dependency>

<dependency>
    <groupId>org.apache.pdfbox</groupId>
    <artifactId>jempbox</artifactId>
    <version>1.8.11</version>
</dependency>

<dependency>
    <groupId>org.apache.pdfbox</groupId>
    <artifactId>xmpbox</artifactId>
    <version>2.0.0</version>
</dependency>

<dependency>
    <groupId>org.apache.pdfbox</groupId>
    <artifactId>preflight</artifactId>
    <version>2.0.0</version>
</dependency>

<dependency>
    <groupId>org.apache.pdfbox</groupId>
    <artifactId>pdfbox-tools</artifactId>
    <version>2.0.0</version>
</dependency>
```

Figure 4.51: Dependencies for adding libraries from Apache PDFBox [46].

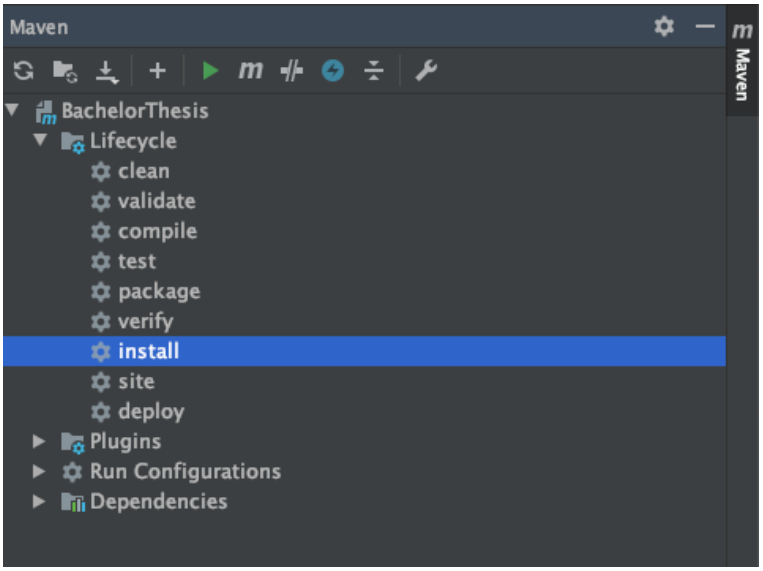


Figure 4.52: Install the JAR files [46].

The class "CreatePDF" is called from the class "TestRunner", containing the FAT, when all tests are finished. The results are then sent as parameters to the "CreatePDF" class.

For creating the PDF, some inspiration was taken from a tutorial made by Tutorialspoint [98]. The first main part of the code is to create an instance of a *PDDocument*, save the document with a name like "FAT.pdf" as in Image 4.53, and close the document. When running this code, a PDF file with the specified filename will be created in the same folder as the code project is located.

```
public class CreatePDF {

    public static void createPDF(boolean powerUpResult) throws IOException {

        //Creating PDF document object
        PDDocument document = new PDDocument();

        //Saving the document
        document.save( fileName: "Fat-test.pdf");

        System.out.println("PDF created");

        //Closing the document
        document.close();
    }
}
```

Figure 4.53: Create a PDF-file [46].

This file is currently empty and, therefore, not possible to open yet. For doing so, some blank pages will have to be created. This is done by creating an instance of a *PDPage* and adding this blank page to the document. As shown in Image 4.54, 10 blank pages are created to the existing document, by using a for-loop. Now the PDF file is possible to open and consists of 10 blank pages.

```java
public class CreatePDF {

    public static void createPDF(boolean powerUpResult) throws IOException {

        //Creating PDF document object
        PDDocument document = new PDDocument();

        for (int i=0; i<10; i++) {
            //Creating a blank page
            PDPage blankPage = new PDPage();

            //Adding the blank page to the document
            document.addPage( blankPage );
        }

        //Saving the document
        document.save( fileName: "Fat-test.pdf");
        System.out.println("PDF created");

        //Closing the document
        document.close();
    }
}
```

Figure 4.54: Add pages to the PDF-file [46].

4.6.3.2 Updating the PDF

Content from the FAT can be loaded into the PDF file and be used as a template. The PDF file can be manipulated as required by choosing the font, text size, content, and more. As shown in the two following Images 4.55 and 4.56, an instance of *PDPageContentStream* is created to add content, as seen in Image 4.57. The content consists of text strings containing letters and values sent as parameters from the "TestRunner" class. Lastly, the instance of the *PDPageContentStream* has to be closed, and the duplicated version of the document has to be saved as a new file.

```java
public class CreatePDF {

    public static void createPDF(boolean powerUpResult) throws IOException {

        //Loading an existing template document
        File file = new File( pathname: "Fat-test.pdf");
        PDDocument doc = PDDocument.load(file);

        //Creating a PDF Document
        PDPage page = doc.getPage( pageIndex: 1);

        PDPageContentStream contentStream = new PDPageContentStream(doc, page);

        //Begin the Content stream
        contentStream.beginText();

        //Setting the font to the Content stream
        contentStream.setFont(PDType1Font.TIMES_ROMAN,  fontSize: 16);

        //Setting the leading
        contentStream.setLeading(14.5f);

        //Setting the position for the line
        contentStream.newLineAtOffset( tx: 25,  ty: 725);

        String text1 = "1. test: 2.1.1.1 Power-up test: " + powerUpResult;
        String text2 = "2. test: 2.1.1.2 Dimmer: ";
```

Figure 4.55: Add content to PDF-file part 1 [46].

```
        //Adding text in the form of string
        contentStream.showText(text1);
        contentStream.newLine();
        contentStream.showText(text2);
        //Ending the content stream
        contentStream.endText();

        System.out.println("Content added");

        //Closing the content stream
        contentStream.close();

        //Saving the document as a new file
        doc.save(new File( pathname: "Fat-test-result.pdf"));

        //Closing the document
        doc.close();
    }
}
```

Figure 4.56: Add content to PDF-file part 2 [46].

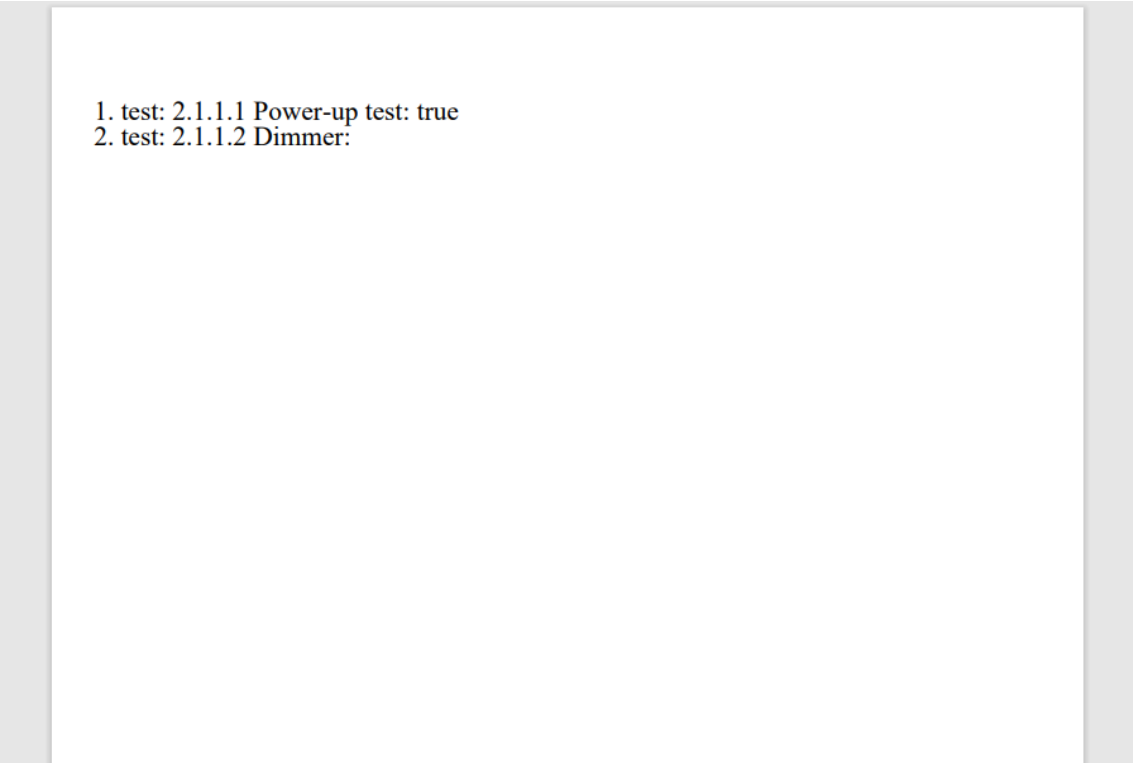


Figure 4.57: The PDF-file after adding some content [46].

Specific page numbers can be removed by loading the PDF template, specifying which pages to remove, and save the document as a new PDF file. Image 4.58 showcase removal of excess pages.

```java
public class CreatePDF {

    public static void createPDF(boolean powerUpResult) throws IOException {

        //Loading an existing document
        File file = new File( pathname: "Fat-test.pdf");
        PDDocument document = PDDocument.load(file);

        //Listing the number of existing pages
        int noOfPages= document.getNumberOfPages();
        System.out.print(noOfPages);

        //Removing the pages
        document.removePage( pageNumber: 2);

        System.out.println("page removed");

        //Saving the document
        document.save( fileName: "Fat-test.pdf");

        //Closing the document
        document.close();
    }
}
```

Figure 4.58: Removing pages from PDF-file [46].

4.6.3.3 Improving Readability of PDF

For better representation of results, color is displayed beside the test name. The color change between each test name and test result. The name of the test and the result of the test is divided into two separate strings, as displayed in Image 4.59, for setting different characteristics to each of them.

```java
String text5 = "  5. test: 2.1.2.1 Rpm Control: ";
String result5 = "" + rpmControlTestResult;
```

Figure 4.59: Name and value of test is divided into two strings [46].

For deciding on the color of the test result, the result is checked. If the test is passed, the text color is set to *green*, if the test is not passed, the color is set to *red*. The function is shown in

Image 4.60.

Figure 4.60: Changing the color of the result value [46].

For setting the text color back to default, the color is set to *black*, as shown in Image 4.61. The black text is used for displaying the test names.

Figure 4.61: Change text color back to original [46].

See Image4.62 for an example of the use of coloring for indicating if a test does pass or not. The first test has passed, and the second one has failed.

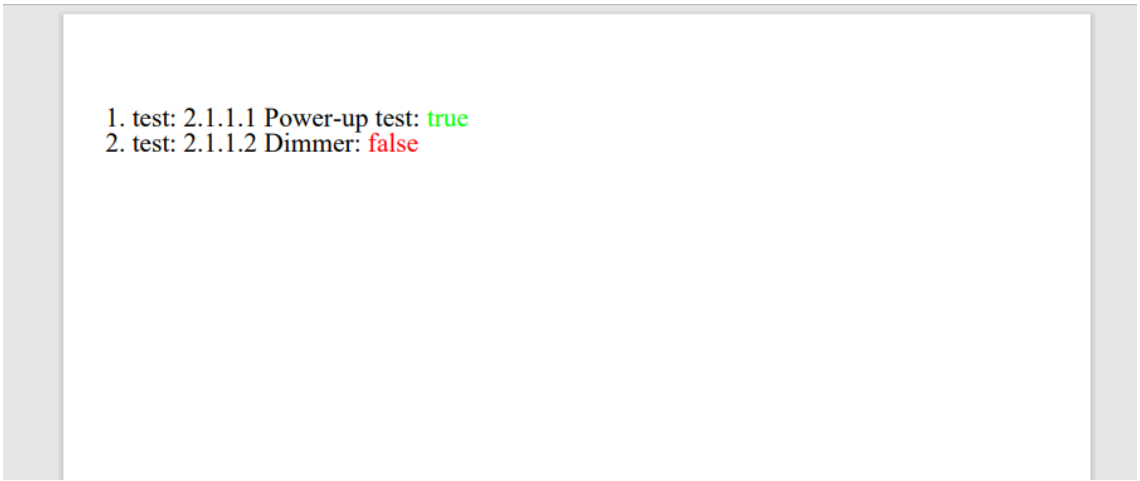


Figure 4.62: Color indication of test results [46].

4.6.3.4 Set Up for Using Multiple Computers

The first time the Java code runs on a new computer, some steps illustrated earlier in this section are required. First, the JAR files needed to be installed into Maven by following these steps:

- Click the Maven-tab to the right in IntelliJ
- Click on *lifecycle*
- Double click *install* to generate the JAR files

Secondly, a PDF template has to be created with the file name used earlier in this section, "Fat.pdf", and with some blank added pages. After this, the Java code can be run at the corresponding computer. However, no extra setup is required if all the files, including the PDF files, are transferred over to the other computer.

4.7 Communication

For the following sections, the IP address remains the same throughout, except when transferring the Simulink® project to B&R Automation Studio.

4.7.1 Ethernet/IP

For the B&R Automation Studio software to connect to the CPU, an Ethernet/IP protocol has to be used. This protocol requires the IP address of the CPU and computer to be in the same range, such as 192.168.1.xxx.

4.7.1.1 Approach

The first step to making a new connection is to open *Physical view -> ETH -> Configuration* and enter the wanted CPU IP address and subnet mask, see Figure 4.63 and Figure 4.64.

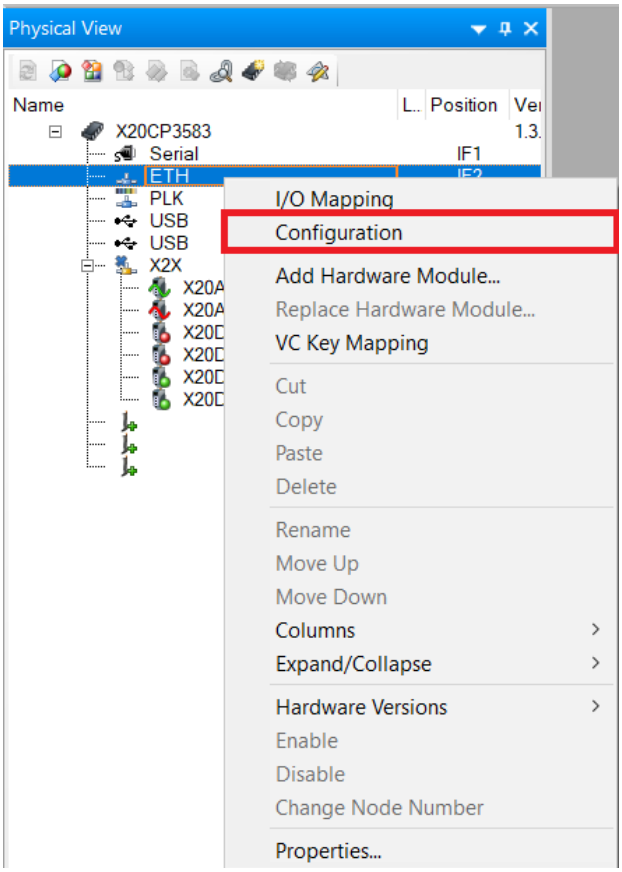


Figure 4.63: ETH Configuration [46].

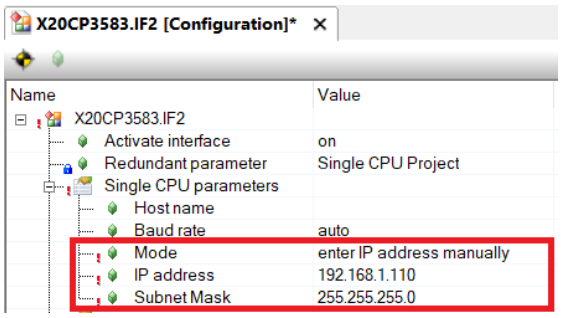


Figure 4.64: Configuring IP address [46].

For the second step of making a new connection, go to *Online -> Settings...*, as illustrated in Figure 4.65.

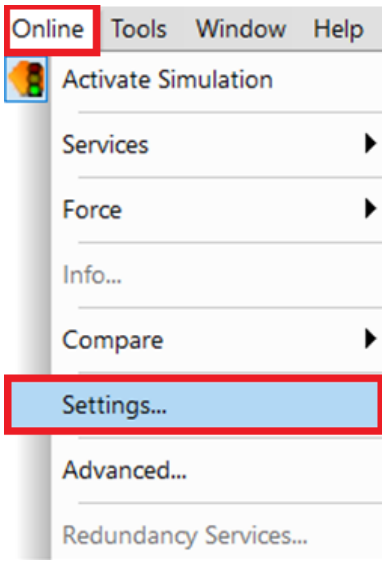


Figure 4.65: *Online -> Settings...* [46].

When the *Settings...* tab had been opened, either *New Connection* to enter a connection manually or *Refresh* to find an existing connection was chosen (see Figure 4.66). Opening the CPU's IP Parameter tab, the IP parameters had to be changed to the wanted address (see Figure 4.67).

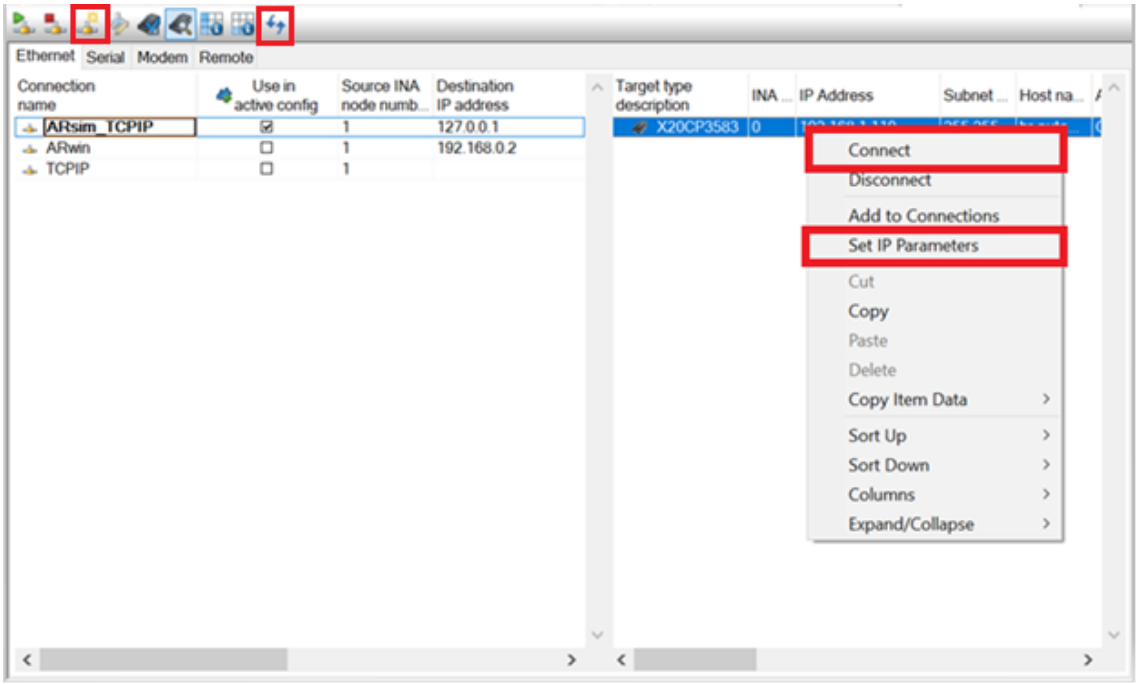


Figure 4.66: *New Connection* or *Refresh*, then *IP Parameter* and later *Connect* [46].

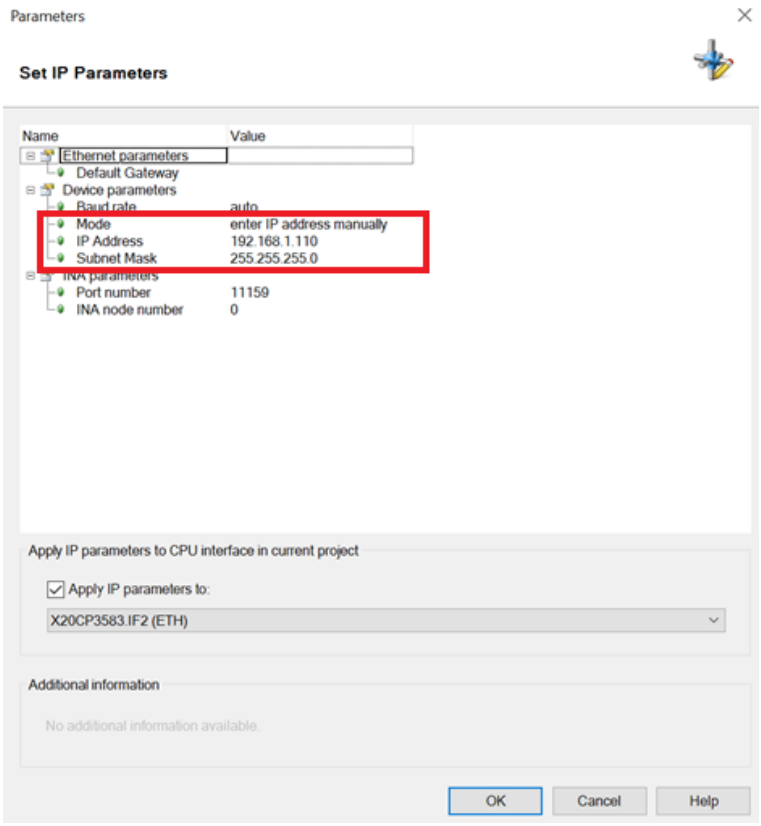


Figure 4.67: The IP Address and Subnet Mask must match [46].

After the previous steps, the CPU could now be connected to the B&R Automation Studio, previously shown in Figure 4.66, and a successful connection is established, shown in Figure 4.68.

Figure 4.68: Successful connection [46].

The connection is now complete, and the project can now be built and transferred onto the CPU.

4.7.2 Automation Studio Target for Simulink

For the communication between B&R Automation Studio and MATLAB®/Simulink®, the toolbox Automation Studio Target for Simulink was used together with a Simulink® coder.

B&R Automation Studio sends values in datatype *REAL* while the thruster model use *DOUBLE*.

Therefore, a conversion block was added at both ends of the thruster model. Figure 4.69 shows how the B&R input block was connected to the system.

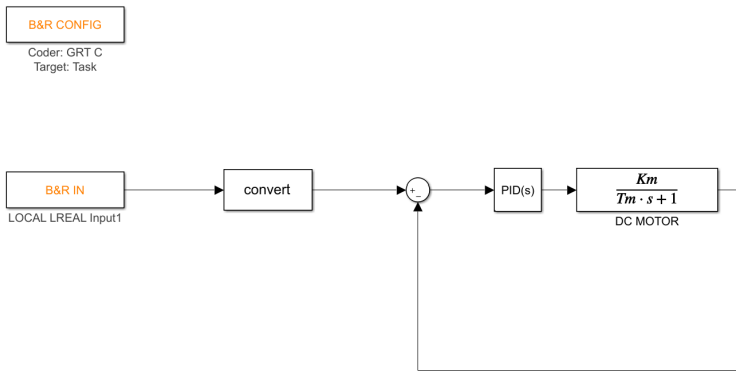


Figure 4.69: Illustrates how the B&R input block and conversion block used to convert datatype was connected to the system [46].

The B&R config block seen in the upper left corner of Figure 4.69 was used to establish a connection between the B&R software and Simulink®.

4.7.2.1 Approach for transfer

B&R Config block

The B&R Config block is required to configure the B&R Automation Studio connection. Only one of these blocks can be inserted in the Simulink® project and should not be connected to the plant.

1. Insert B&R Config Block, from the Simulink Library Browser, in Simulink®.
2. Set the right configurations in the B&R Config Block

   The configurations in the B&R Config Block have to be specified to connect correctly with the project target. In the section *model configurations*, the desired language and the coder need to be chosen. The Fixed- step size also needs to match the cycle time in the target [13]. Figure 4.70 displays the configurations used.

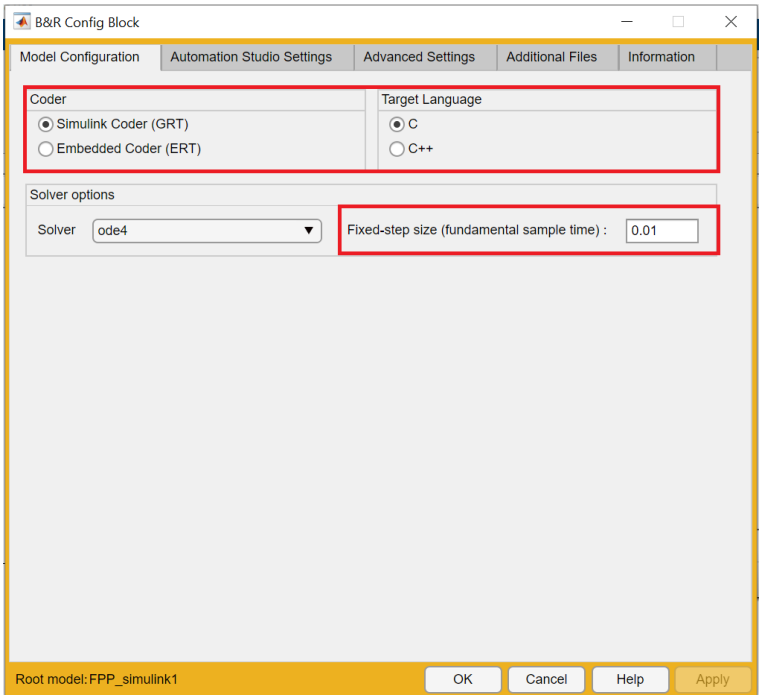


Figure 4.70: B&R Config block Model Configuration [46].

In section *Automation Studio Settings* the path to the Automation Studio project needs to be specified. A task name is created and added to hardware.

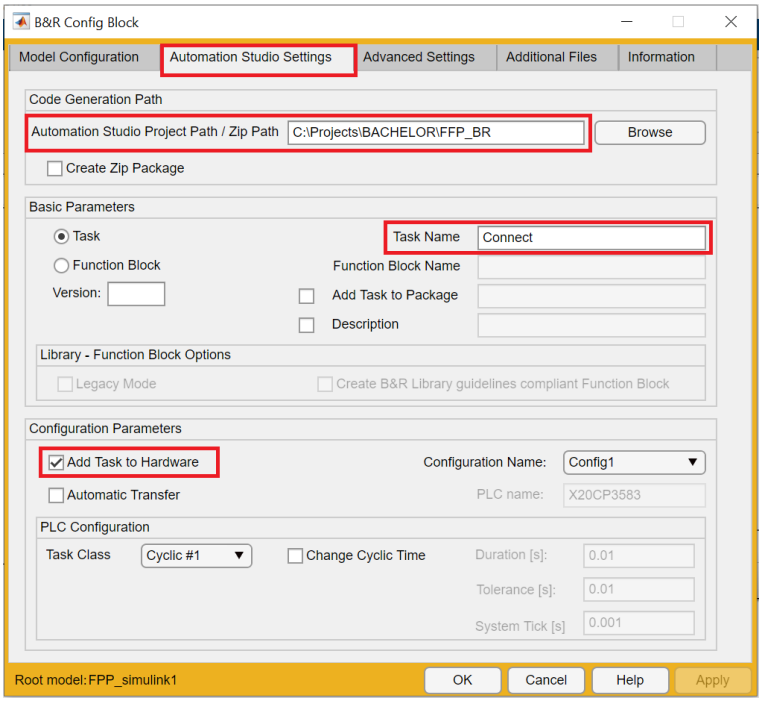


Figure 4.71: B&R Config block Automation Studio Setting [46].

In *Advanced Settings*, the external mode can be chosen in order to connect with the target project. The Simulink® model contains blocks that are not a part of the B&R- specific blocks. When enabling "create Simulink I/O as variable," the code generation will complete successfully with the Simulink® blocks.

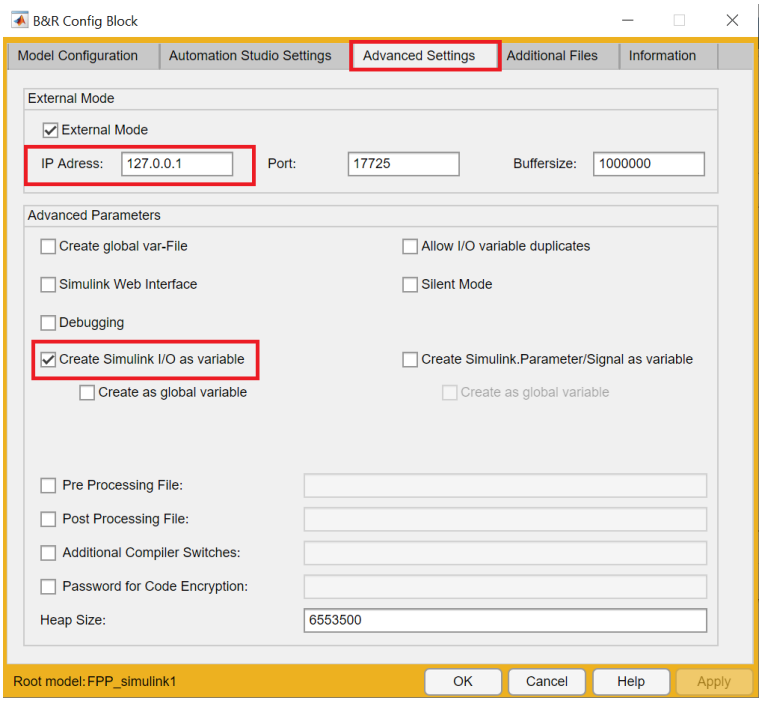


Figure 4.72: B&R Config block Advance Settings [46].

The B&R IN block / B&R OUT block

The B&R IN block / B&R OUT block is used to create input and output variables that can be controlled from B&R Automation Studio.

The settings in the block can be configured. The variables are set to GLOBAL in order for the B&R Automation Studio project to use the model when connected to the Mcon Control System. The data type is set to REAL to match the data type in the target system (See Figure 4.73 and 4.74).

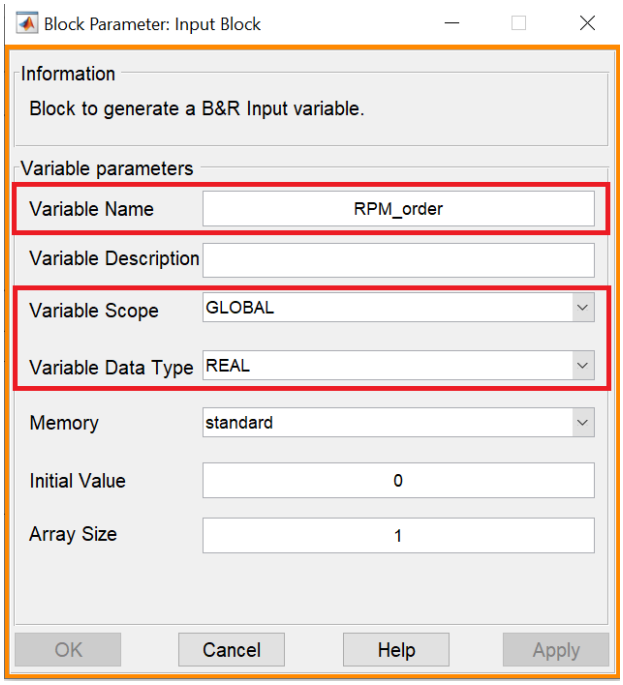


Figure 4.73: Automation Studio Target for Simulink: The B&R IN block [46].

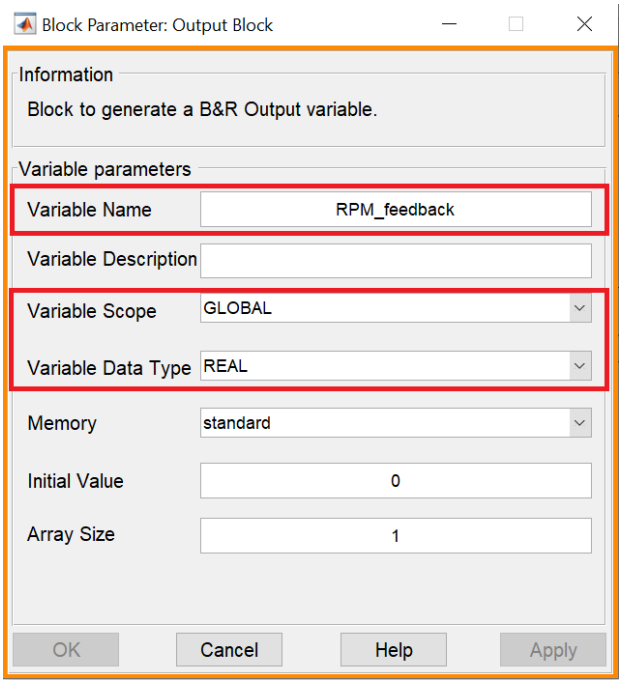


Figure 4.74: Automation Studio Target for Simulink: The B&R OUT block [46].

Transfer to target

The transfer to target is done using the Simulink® coder, seen in Figure 4.75. When building the project, using the coder, the Simulink® project is automatically generated to C code and transferred into the B&R Automation Studio. The input value can be set, and the output value can now be viewed from the B&R Automation Studio project.

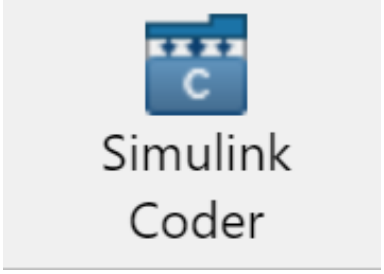


Figure 4.75: Simulink® coder [46].

4.7.3 OPC UA

When starting with this project, Kongsberg Maritime had only been using Modbus as a communication protocol in their systems. This protocol ensures a safe and fast transition of data, but it uses physical wires for transmission. Kongsberg Maritime wanted the communication protocol for the system to be upgraded and suggested Canbus as a suitable communication protocol between B&R Automation Studio and Java. Multiple communication protocols were

considered, such as Canbus and TCP/IP.

Some research discovered that B&R Automation Studio is commonly using OPC UA as communication protocol. Using Canbus could have made the communication easier to implement from the Java side. OPC UA is, however, more optimal for B&R Automation Studio. Since the group had more experience with Java than with B&R Automation Studio, it was decided to choose the protocol that was easiest for implementing into B&R Automation Studio. It was as well discovered that OPC UA has TCP/IP implemented as the default communication method.

B&R Automation Studio would be responsible for keeping the variables needed for the system to work. Java would then get the variables in B&R before running the tests and print out the results. Java is an OPC UA client, and B&R Automation Studio is an OPC UA server.

4.7.3.1 B&R Automation Studio Server

Any controller that supports the minimum Automation Runtime supports the OPC UA protocol, meaning that OPC UA only has to be activated before use.

In *Physical View*, *Configuration* had to be opened to access the tab where the *OPC UA System* could be set to *on*, as illustrated in Figure 4.76. Activating the OPC UA created a port for the connection.

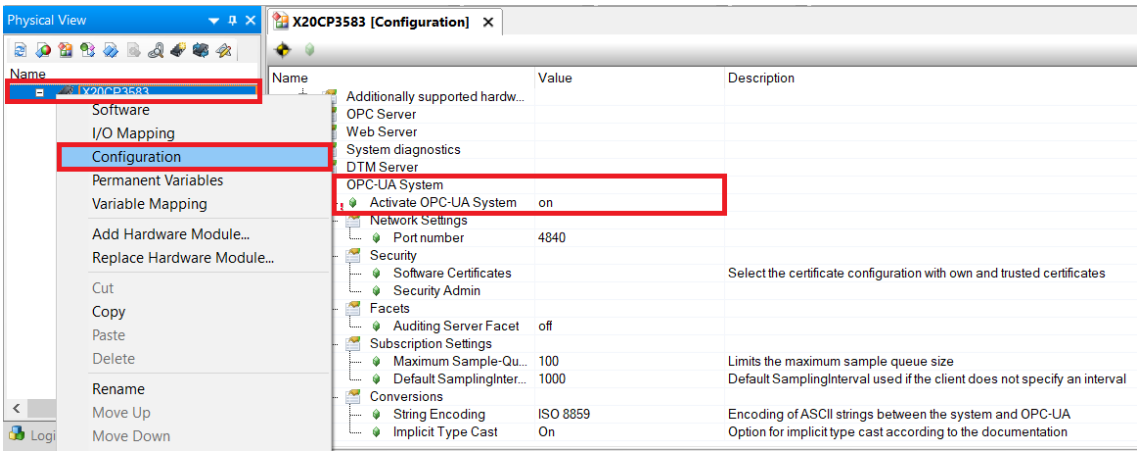


Figure 4.76: Open the configuration menu [46].

In the next step, the *Connectivity* and then *OpcUA* folders had to be opened in the *Configuration*

View, to be able to add the *OPC UA Default Viewer* under the map *OpcUA*, shown in Figure 4.77.

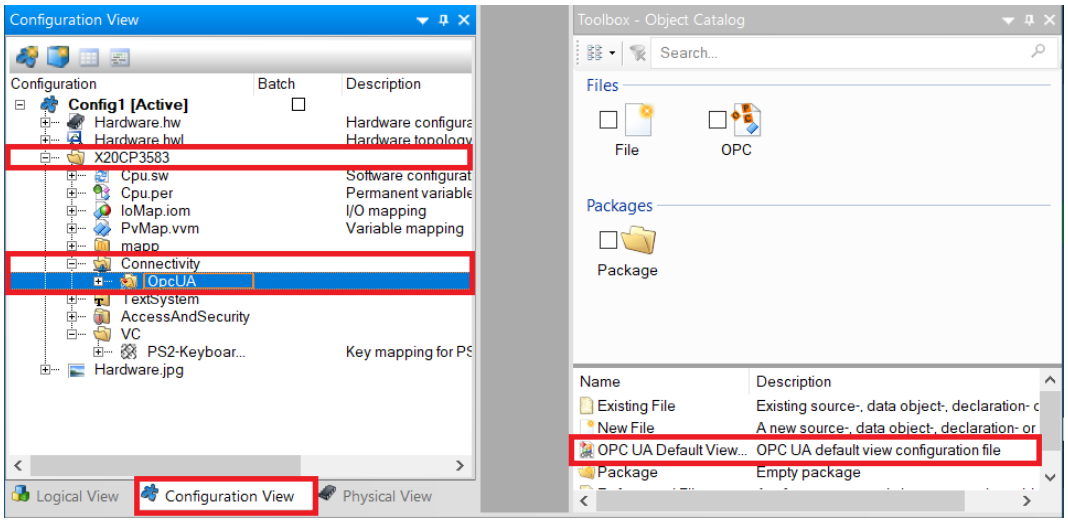


Figure 4.77: Add the *OPC UA Default View File* to the *OpcUA* folder [46].

The *OPC UA Default View File* contains all the project variables. These variables needed to be enabled, as seen in Figure 4.78 to be able to expose the data from the the controller to the OPC UA server.

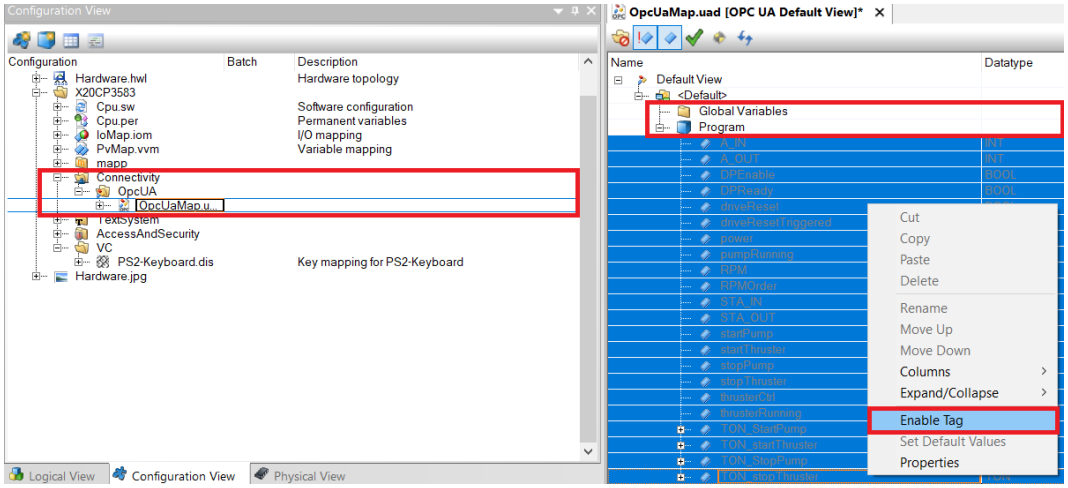


Figure 4.78: Set the *enable* to *true* for all variables [46].

When opening *Tools* -> *Offline Install* the popup, showed in Figure 4.79, was loaded. Here the *Path in local file system* had be chosen, and a path needed to be assigned. Here a new folder created in the B&R project folder has to be referenced. The folder was then installed.

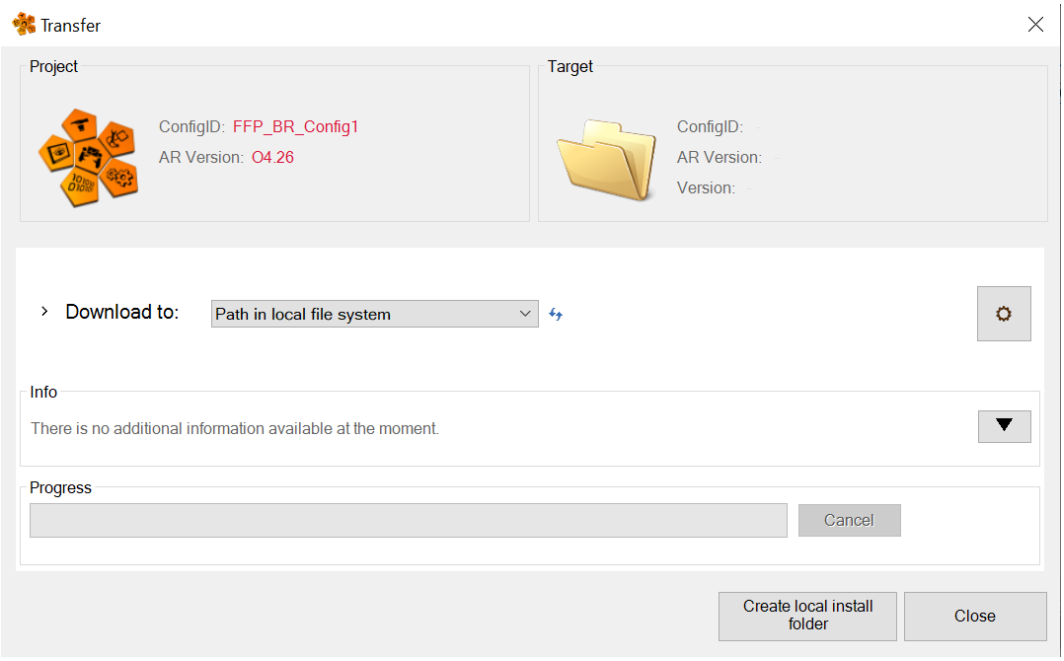


Figure 4.79: Transfer the program to CPU [46].

After these configurations, the B&R server part of the program is complete. An OPC UA server will be opened at the CPU, ready for connection requests from clients.

4.7.3.2 Java Client

The OPC UA Java client is inspired by Eclipse Milo [58]. The inspiration is used towards establishing a connection, found in Eclipse Milo interface "ClientExample" and classes "ClientExampleRunner" and "KeyStoreLoader". Classes "ReadExample" and "ReadWriteCustomDataTypeNodeExample" are used for making read functions, for sending parameters.

For taking advantage of the open source implementation of OPC UA from Eclipse Milo, a library was needed. To import the library, a dependency was added in the file "pom.xml", illustrated in Image4.80. For making the rest of the code accept the library in IntelliJ, it was necessary to go to *file -> settings -> build.execution, deployment -> build tools -> maven -> importing -> import maven projects automatically*, as well as *file -> settings -> build.execution, deployment -> build tools -> maven -> always update snapshots.*

```xml
<dependency>
    <groupId>org.eclipse.milo</groupId>
    <artifactId>sdk-client</artifactId>
    <version>0.5.4</version>
</dependency>
```

Figure 4.80: A dependency for adding Eclipse Milo as a library [46].

For creating the Java client, four classes are made. The class "RunTestsMilo" in Image 4.81 controls the start of all other necessary classes for connecting the server and client, running the tests, and disconnecting the server and client.

```java
/**
 * running tests using milo
 */
public static void runTestsMilo() {
    MiloClient miloClient = new MiloClient();

    TestRunner testRunner = new TestRunner(miloClient);
    testRunner.runTests();

    miloClient.stop();
}
```

Figure 4.81: Class "RunTestsMilo" controls all other classes [46].

Class "MiloClient" shown in Image 4.82 connects and disconnects to the server, using port number and security policy from interface "ClientExample" in Image 4.83, and certificates from class "KeyStoreLoader". In "MiloClient" a client is created before connecting with the B&R Automation Studio server using the IP address for the CPU, as explained in section 4.7.1.1, as well as the port number. In class "ClientExample" the *securityPolicy* has to be set to *None*.

```java
/**
 * creating a client of milo
 */
public MiloClient() {

    try {
        Path securityTempDir = Paths.get(System.getProperty("java.io.tmpdir"),  ...more: "client", "security");
        Files.createDirectories(securityTempDir);
        if (!Files.exists(securityTempDir)) {
            throw new Exception("unable to create security dir: " + securityTempDir);
        }

        File pkiDir = securityTempDir.resolve("pki").toFile();
        trustListManager = new DefaultTrustListManager(pkiDir);
        DefaultClientCertificateValidator certificateValidator = new DefaultClientCertificateValidator(trustListManager);
        KeyStoreLoader loader = new KeyStoreLoader().load(securityTempDir);

        client = OpcUaClient.create(                                          //creates a client
                ClientExample.getEndpointUrl(),
                endpoints ->
                        endpoints.stream()
                                .filter(ClientExample.endpointFilter())
                                .findFirst(),
                configBuilder ->
                        configBuilder
                                .setApplicationName(LocalizedText.english("eclipse milo opc-ua client"))
                                .setApplicationUri("urn:eclipse:milo:examples:client")
                                .setKeyPair(loader.getClientKeyPair())
                                .setCertificate(loader.getClientCertificate())
                                .setCertificateChain(loader.getClientCertificateChain())
                                .setCertificateValidator(certificateValidator)
                                .setIdentityProvider(ClientExample.getIdentityProvider())
                                .setRequestTimeout(uint( value: 5000))
                                .build());

        client.connect().get();                                               //connects the client and server
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figure 4.82: Function for creating a client, and connecting to a server [46].

```java
/**
 * interface for establishing a connection
 */
public interface ClientExample {

    /**
     * getting the endpoint
     * @return the connection endpoint containing IP-address and port number
     */
    static String getEndpointUrl() {
        return "opc.tcp://192.168.1.110:4840";
    }

    /**
     * filter for endpoint
     * @return the filtered endpoint
     */
    static Predicate<EndpointDescription> endpointFilter() {
        return e -> getSecurityPolicy().getUri().equals(e.getSecurityPolicyUri());
    }

    /**
     * getting the security policy
     * @return the security policy
     */
    static SecurityPolicy getSecurityPolicy() {
        return SecurityPolicy.None;
    }

    /**
     * getting the identity provider
     * @return the identity provider
     */
    static IdentityProvider getIdentityProvider() {
        return new AnonymousProvider();
    }
}
```

Figure 4.83: Interface for specifying port number and security policy [46].

For disconnecting with the server, the function *stop* is called. The *stop* function disconnects the client from the connected server, as illustrated in Image 4.84.

```java
/**
 * stopping the connection
 */
public void stop() {
    try {
        client.disconnect().get();
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (ExecutionException e) {
        e.printStackTrace();
    }
}
```

Figure 4.84: Function for disconnecting the connection with the server [46].

4.7.3.2.1 Sending Parameters For enabling Java to read the value of a variable in B&R, a get-function is created in class "MiloClient". Image 4.85 shows how a boolean variable is collected. The *NodeId* of the variable needs to be specified. The value of the *NodeId* was copied from the UaExpert program since this program connects in the same way as in the Java code. This *NodeId* is then sent as a parameter to the standardized read function *getVariable*, as in Image 4.86. The requested variable will then be fetched, allowing Java to read the value of this variable, using serialization before getting decoded into human-readable text. After that, the variable's value is sent back to the get-function as an Object and converted into an int or a boolean.

```java
/**
 * getting state of thrusterMotor
 * @return the state of thrusterMotor
 */
public boolean getThrusterMotorValue() {
    NodeId objectId = NodeId.parse("ns=6;s=::Program:thrusterRunning");
    Object value = getVariable(objectId);
    boolean calculatedValue = ((Boolean) value).booleanValue();
    System.out.println("ThrusterRunning: " + calculatedValue);
    return calculatedValue;
}
```

Figure 4.85: Function for converting value of variable to boolean [46].

```java
/**
 * getting the value of a variable
 * @param nodeId the variable to collect value of
 * @return the value of the requested variable
 */
public Object getVariable(NodeId nodeId) {
    try {
        DataValue value = null;
        UaVariableNode node = client.getAddressSpace().getVariableNode(nodeId);
        value = node.readValue();
        variant = value.getValue();
        decoded = (Object) variant.getValue();
    } catch (UaException e) {
        e.printStackTrace();
    }
    return decoded;
}
```

Figure 4.86: Function for reading value of variables [46].

Chapter 5

Results

The following sub-chapters show the results for the entire project. First included are the results of the reviews before showcasing the final results.

5.1 Reviews

This chapter elaborates the tests and reviews performed during the project, and how they affected the final result.

5.1.1 Electrical testing

Before powering up the CPU, all the wiring of the I/O modules was controlled. This ensured the wiring had been done correctly, minimizing the risk of short circuits happening, which could potentially damage the electrical components. All connection points and wires, both on the CPU and the Mcon rack, were measure with a multimeter to further control if the I/O modules were working after powering up.

5.1.2 Logic testing

The logic has been tested in two ways, in simulation mode and connected to the CPU. When running the logic in simulation mode, the troubleshooting would be easier as the hardware were

eliminated from the equation, meaning more focus on the software.

When testing the software in simulation mode, the inputs had to be manually set to *TRUE* to check if the logic would behave the way it was intended to behave.
As seen in Figure 5.1, it was tested that the *TON* function worked as intended and the CASE-statement, named *thrusterCtrl*, followed the instructions.
Illustrated in Figure 5.2, the test shows the logic is given a INT value of 0, making RPM order and *STA_IN* the correct values at the same time, and then seeing if RPM feedback, *STA_OUT* and *A_OUT* would go to the same values as the inputs with a delay.

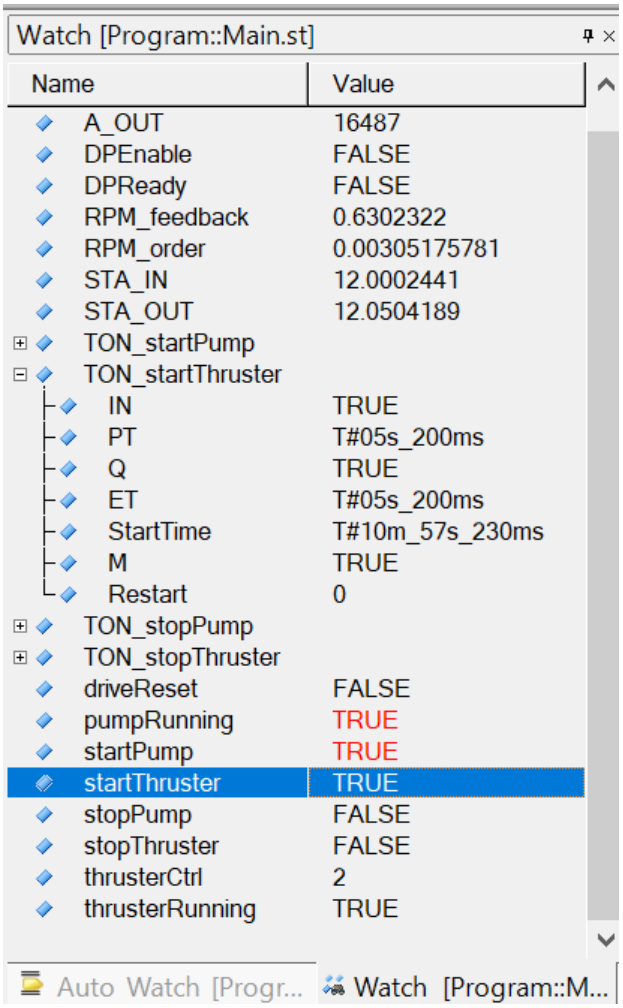


Figure 5.1: Testing the logic's correspondence to starting pump and thruster [46].

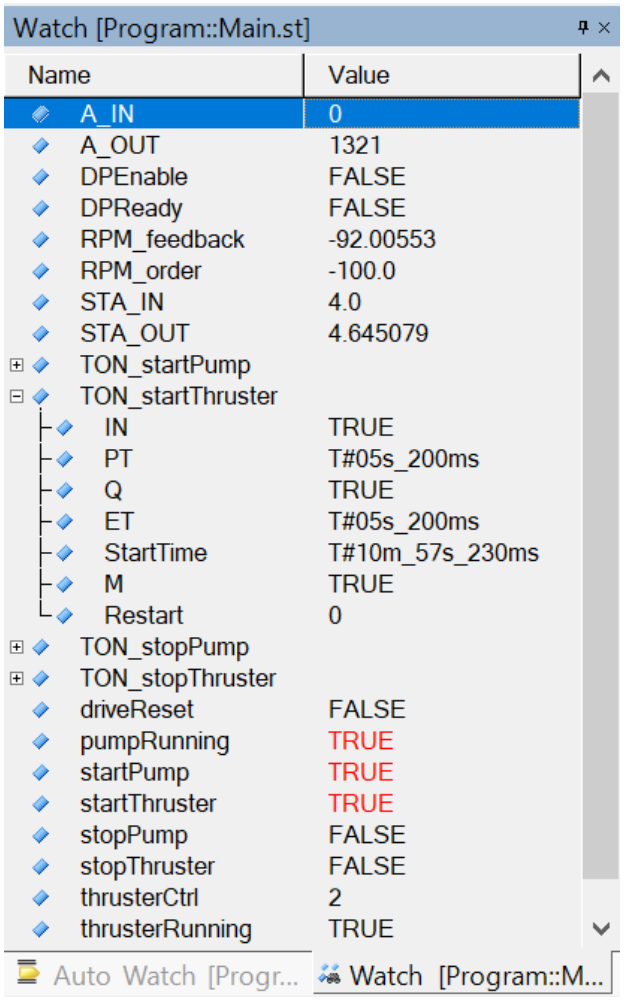


Figure 5.2: Testing the logic's correspondence to an A_IN order [46].

If the logic did not behave as intended, a check was done to see if the software had been success-

fully transferred. This was done under *Online -> Compare -> Software*. If the software had been uploaded to the target, the compare window did not show any red lines, as illustrated in Figure 5.3. This comparison was made both in simulation mode and when connected to the CPU.

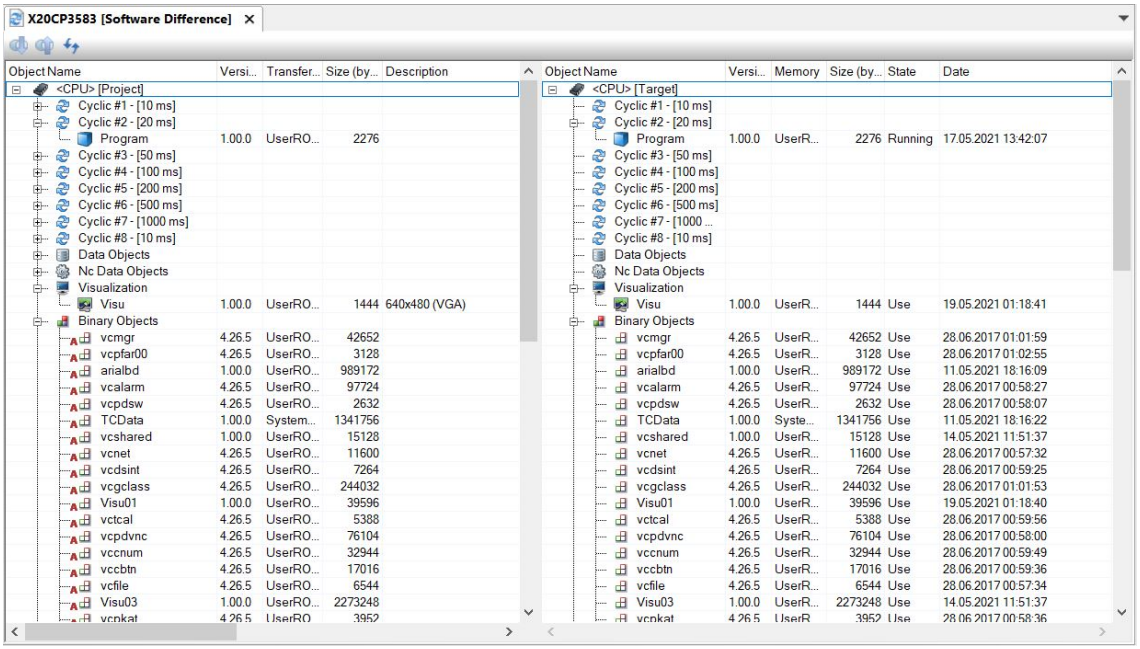


Figure 5.3: Checking similarities [46].

The GUI was also tested in simulation mode before connecting to the Mcon system.

When everything has been working in simulation mode, hardware is added to the equation. Adding the hardware, had to provide the same results as in simulation mode, except for the *stop pump* signal, which had to be inverted. The *stop pump* signal was then tested by pulling out either the *stop pump* wire or the 24V wire from the I/O module.

5.1.3 Semi-automatic FAT

5.1.3.1 Virtual Connection

For testing the Java code without being connected to the B&R Automation Studio, a virtual connection is created. The interface "Connection" is required in order to run the Java code likewise for both the virtual connection in class "VirtualConnection" and the connection with B&R Automation Studio in class "MiloClient".

For making the virtual connection, the "RunTestsVirtual" and "VirtualConnection" classes are created. The first one, illustrated in Image 5.4, starts the code and runs the tests with a virtual connection. The second one handles the reading of variables requested by the class "TestRunner".

```
/**
 * running tests using a virtual connection
 * @param args contains the supplied command-line arguments as an array of String objects
 */
public static void main(String[] args) {
    TestRunner testRunner = new TestRunner(new VirtualConnection());
    testRunner.runTests();
}
```

Figure 5.4: Function for starting the virtual connection [46].

For reading the value of a variable, a get-function is created. When the "TestRunner" asks for the value of a variable, "VirtualConnection" sends back the current value of this variable. The tests in "TestRunner" only have to specify which variable to read, like the variable *rpmControl*, in the function *getRpmControlValue*, in Image 5.5. Each variable to be read has its own get-function.

```
/**
 * getting value of rpmControl
 * @return the value of rpmControl
 */
public int getRpmControlValue() {
    return rpmControlValue;
}
```

Figure 5.5: Function for reading the *RpmControl* variable with a virtual connection [46].

5.1.3.2 Results during testing

To keep an eye on the progress of the FAT, an update on the test results is printed continually during the testing. This update can be seen in the system window in Java, as illustrated in Image 5.6.

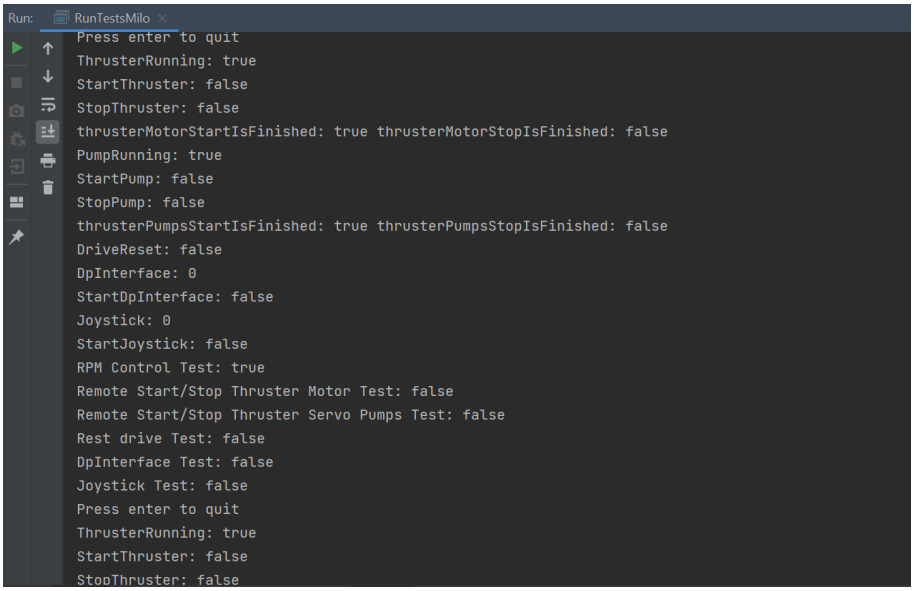


Figure 5.6: Overview of the progress of the FAT [46].

5.1.4 Model testing

5.1.4.1 DC motor

Figures 5.7 and 5.8 shows the magnitude, phase response, and step response of the DC motor without load. The magnitude of the Bode plot was used to ensure that the transfer function of the DC motor had the correct gain.

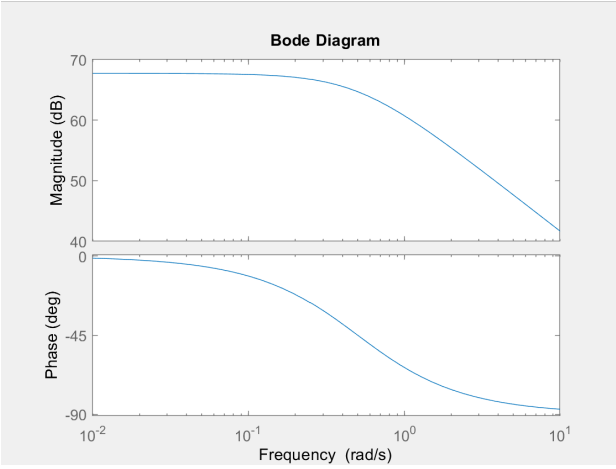


Figure 5.7: System response to a step input of 100% without any PID controller [46].

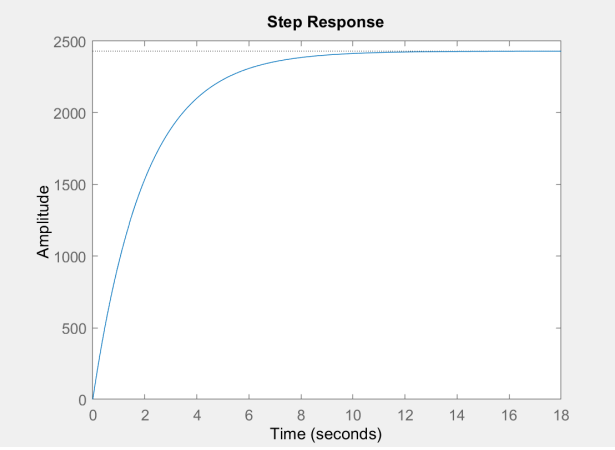


Figure 5.8: System response to a step input of 100% with a tuned PID controller [46].

The results from the Bode plot in Figure 5.7 and the step plot in Figure 5.8 indicate a magnitude that is right below 70 and the amplitude right below 2500. The plotted magnitude agrees with the previously calculated magnitude of 67.7 dB and the amplitude of 2430 (see section 4.5.1 for the calculations). The settling time can be seen to be about 8 seconds, which is also correct for our system, with the time constant of 2. These results indicate that the values for the gain and time constant of the DC motor transfer function was chosen correctly.

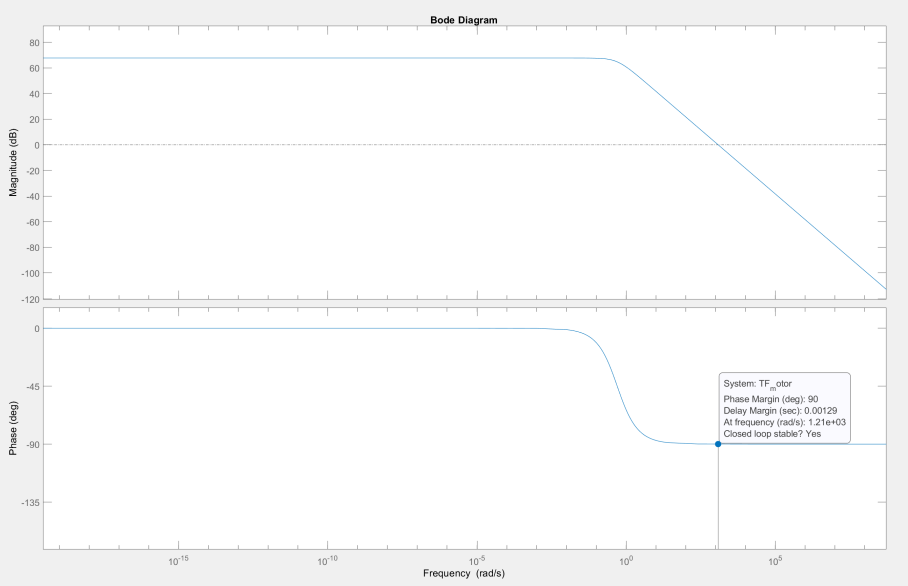


Figure 5.9: open-loop System response to a step input of 100% [46].

The phase margin is found by using the magnitude curve. At the point where the magnitude curve crosses $0dB$, a line can be drawn down until the line crosses the phase curve [72]. In Figure 5.9 the point on the phase curve is marked, containing information about the plot results. The phase margin is at 90 degrees, which means it is stable.

5.1.4.2 Tuning of FPP model

The FPP model illustrated in Figure 4.25 represents the system used for simulation of a FPP system. For testing purposes, the B&R input- and output-blocks were disconnected and replaced with a step input and a scope.

The system response without a PID controller with a step input of 100 can be seen in Figure 5.10. The results of the same system after the PID is tuned to the desired values, with the step input of 100, is shown in Figure 5.11

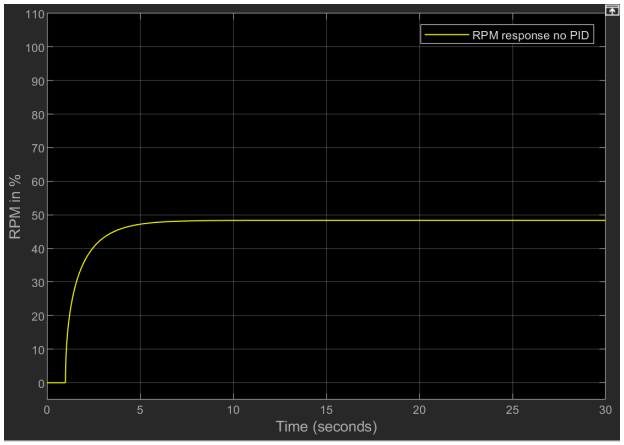


Figure 5.10: System response to a step input of 100% without any PID controller [46].

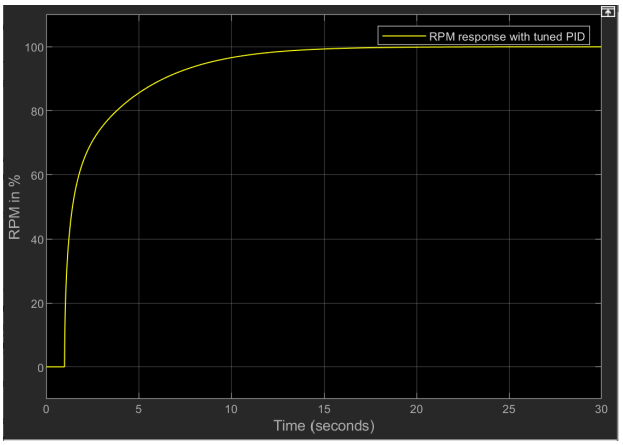


Figure 5.11: System response to a step input of 100% with a tuned PID controller [46].

Table 5.1 displays the values used for the PID controller when testing the FPP model in Simulink®. These values are also used for further testing in simulation mode and real-time testing of the FPP model.

P	I	D
2	0.89	1.1

Table 5.1: List of PID values

5.1.4.3 Tuning of CPP model

For the CPP model, a step input test of a 100 for the RPM was done with the minimum and maximum pitch of 0.4 and 1.4 respectively. These tests were performed both with and without a

tuned PID controller. Figure 5.12 shows the response without a tuned PID, and a pitch value of 0.4. Figure 5.13 shows the system response without a tuned PID and a pitch of 1.4.

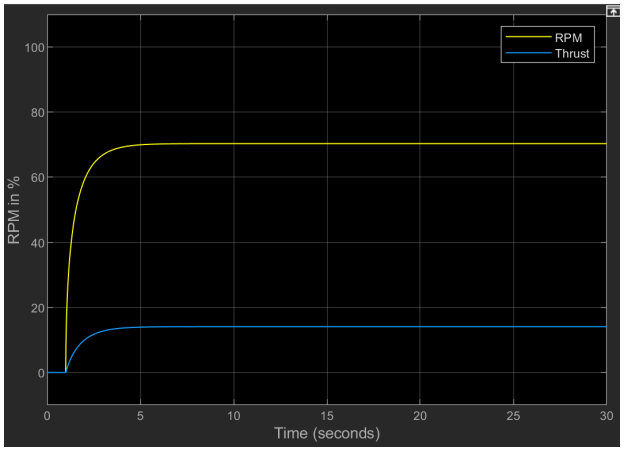


Figure 5.12: System response: No PID tuning and minimum pitch [46].

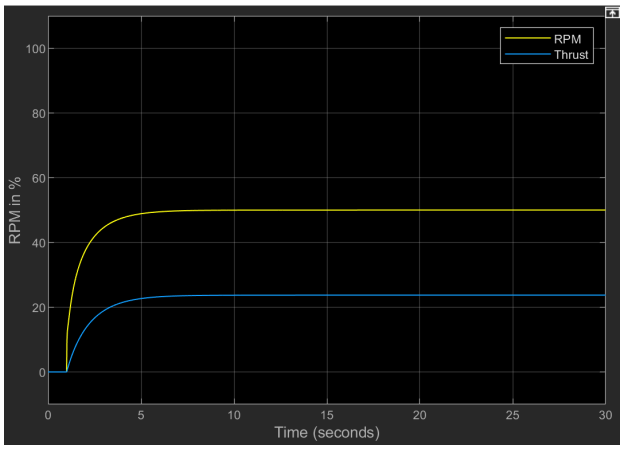


Figure 5.13: System response: No PID tuning and maximum pitch [46].

As seen in Figures 5.12 and 5.13, the system could not reach maximum RPM, as requested by the input. The test also shows that the RPM with minimum pitch is lower than the RPM with maximum pitch. In contrast, the thrust is higher than the test with maximum pitch. The settling time also increased when pitch was set to maximum.

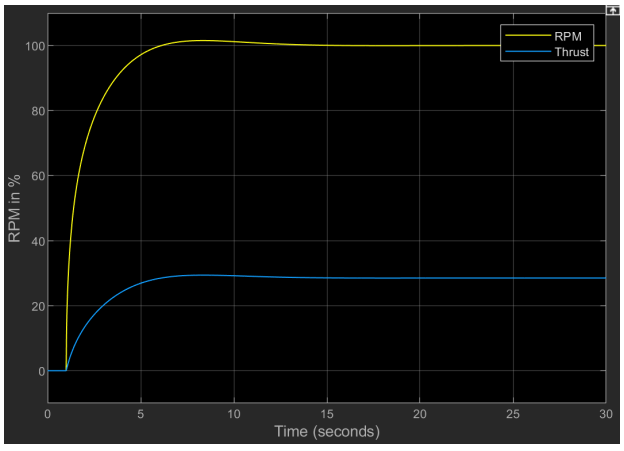


Figure 5.14: System response: PID tuned and minimum pitch [46].

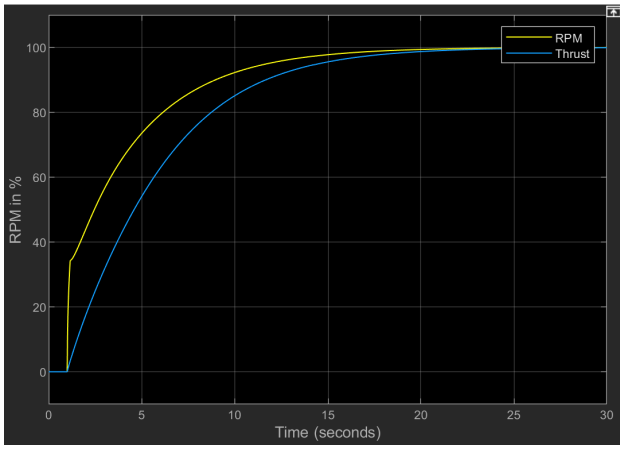


Figure 5.15: System response: PID tuned and maximum pitch [46].

After the PID controller was tuned, shown in Figure 15.14 and Figure 5.15, the RPM reached the desired values. With the minimum input of pitch, the rise time is low. Also, the system

produces an overshoot due to the low resistance given by the pitch value. In comparison, the thrust produced with maximum pitch is higher. "The rise time" and "settling time" of the system increases, resulting from the added resistance from the pitch.

Table 5.2 displays the values used for the PID controllers when testing the CPP model in Simulink®.

Controller	P	I	D
RPM	0.9	0.727	0.3
Pitch	0.1222	0.1244	0.0026

Table 5.2: List of PID values for CPP model

5.1.5 FPP simulation mode testing

The graphical representation of the models, with specific input orders, shows similarities in the characteristics of the curves. The curves with the same behavior are concave and convex, producing opposite directions and fluctuation. Since only the first quadrant is modeled in both positive and negative directions, the curves for similar inputs were expected to be equal in the opposite direction and curvature.

Figures 5.16, 5.17, 5.18 and 5.19, 5.20 and 5.21 represent tests performed connected to B&R Automation Studio.

The RPM response shown in Figures 5.16 and 5.17 showcase the system response when going from maximum thrust to idle in both directions, with a settling time of approximately 5 seconds and no overshoot.

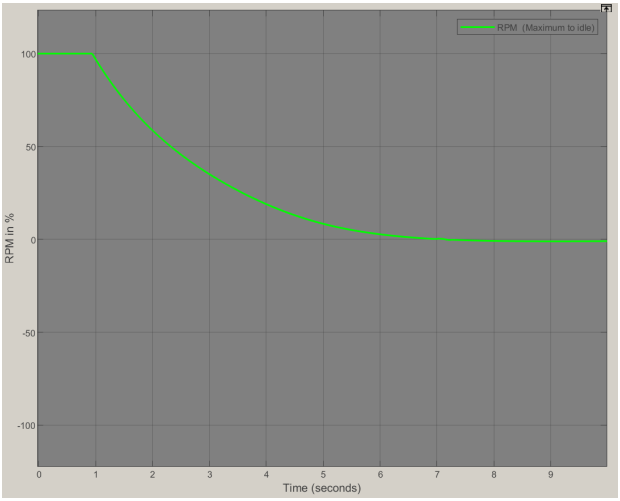


Figure 5.16: RPM response: Maximum value to idle [46].

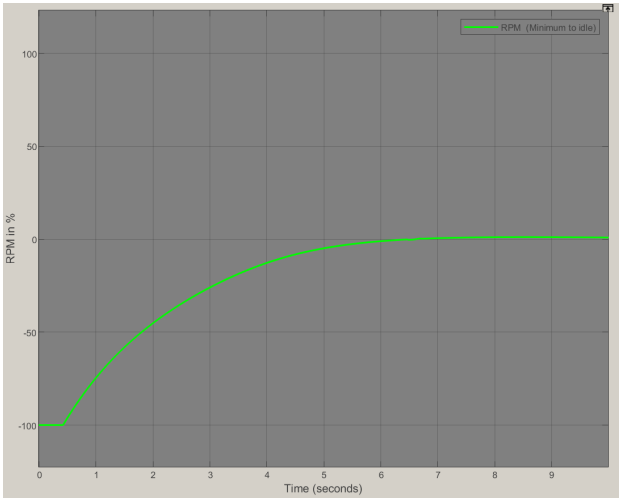


Figure 5.17: RPM response: Mainimum value to idle [46].

The RPM response shown in Figures 5.18 and 5.19 showcase. the system response from idle to maximum RPM in both directions with a settling time of 10 seconds and no overshoot.

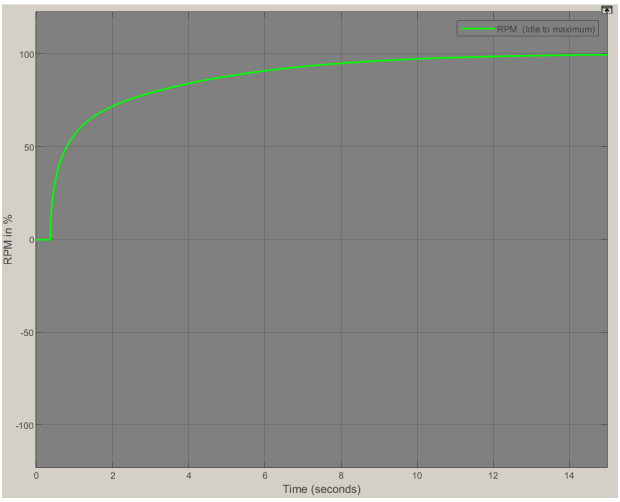


Figure 5.18: RPM response: Idle to maximum value [46].

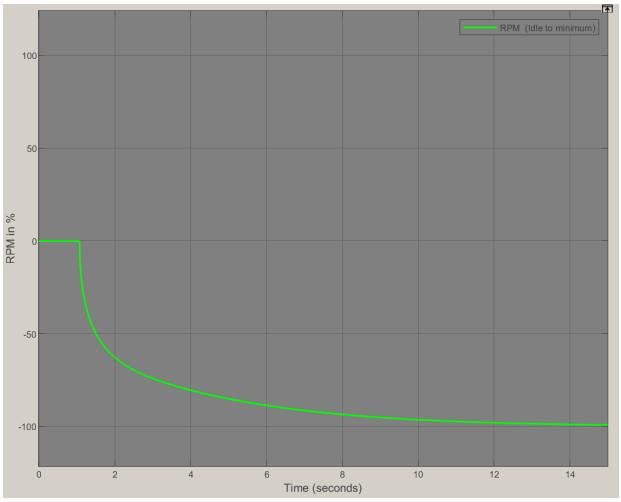


Figure 5.19: RPM response: Idle to minimum value [46].

The curves in Figures 5.20 and 5.21 shows an uneven fluctuation, as the curve is vertically crossing idle. This was expected since the solution used for the model does not operate in all four quadrants, and does not account for advance speed and the mass of the vessel as the propeller changes direction.

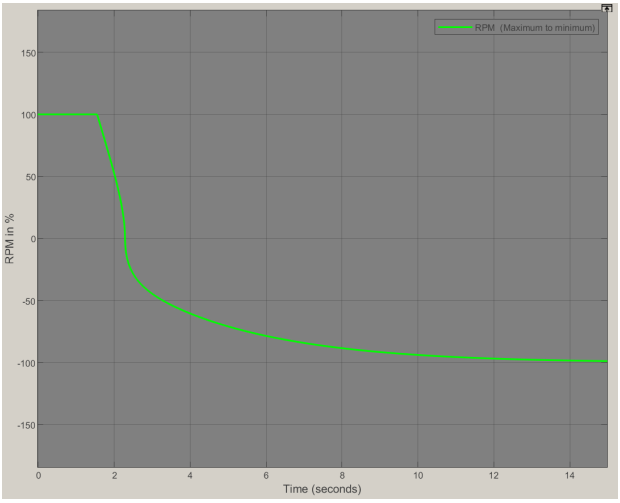


Figure 5.20: RPM response: Maximum value to minimum value [46].

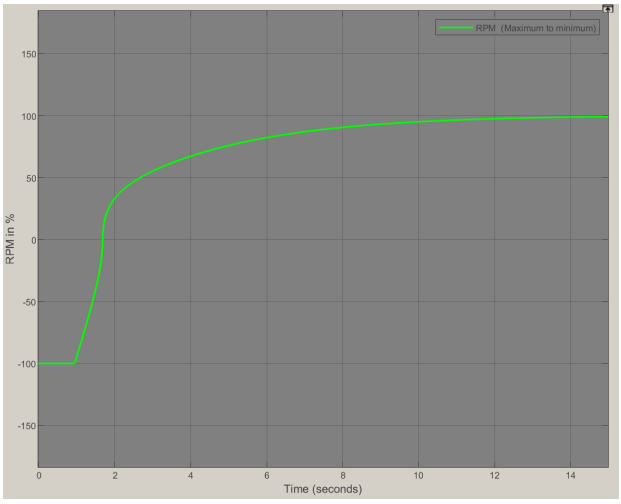


Figure 5.21: RPM response: minimum value to maximum value [46].

The graphs in Figures 5.22 and 5.23 shows input orders at 50% and 25% with settling times of approximately 6 and 5 seconds respectively. These two tests were performed to evaluate the feedback with smaller input values.

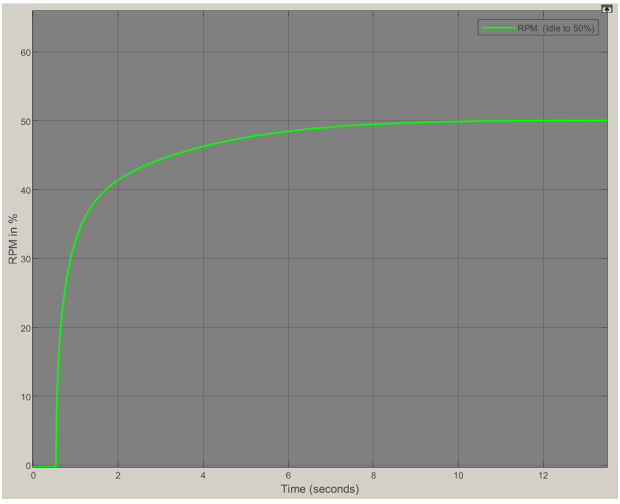


Figure 5.22: RPM response: Idle to 50% [46].

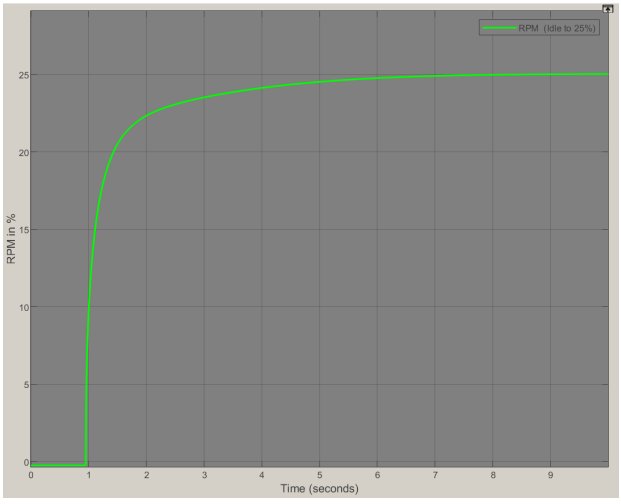


Figure 5.23: RPM response: Idle to 25% [46].

In addition to the pure input tests, tests containing several input orders to compare and evaluate the results with expected behavior were performed. Figure 5.24 showcases the result of the input orders in table 5.3. The change of input order was effected when the prior reached its order. The table also shows at what time the new input was ordered and the time used to reach the value.

No.	Input order	Time to target
1	100%	11.84
2	0%	7.72
3	50%	5.64
4	-100%	6.32
5	100%	16.72

Table 5.3: Input orders

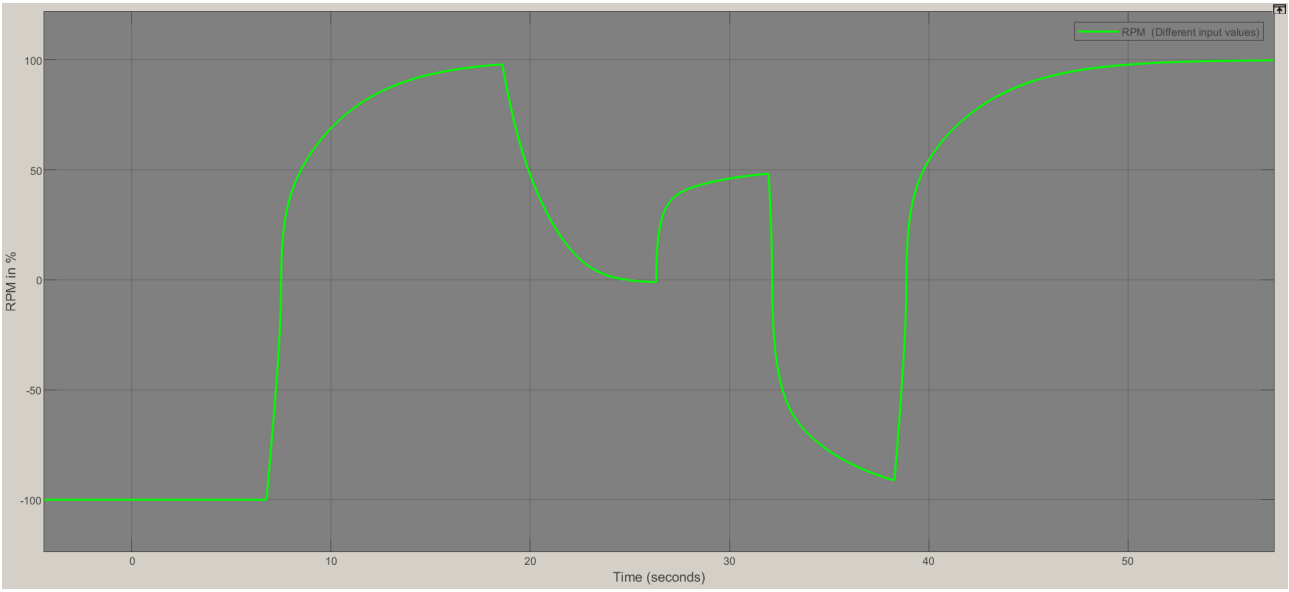


Figure 5.24: RPM response: Different input values [46].

In the following Figure 5.25, a step function is used to simulate the response in Simulink®. Compared to Figure 5.18, which has an identical input type, it results in the same curve. The settling time is close, which is expected.

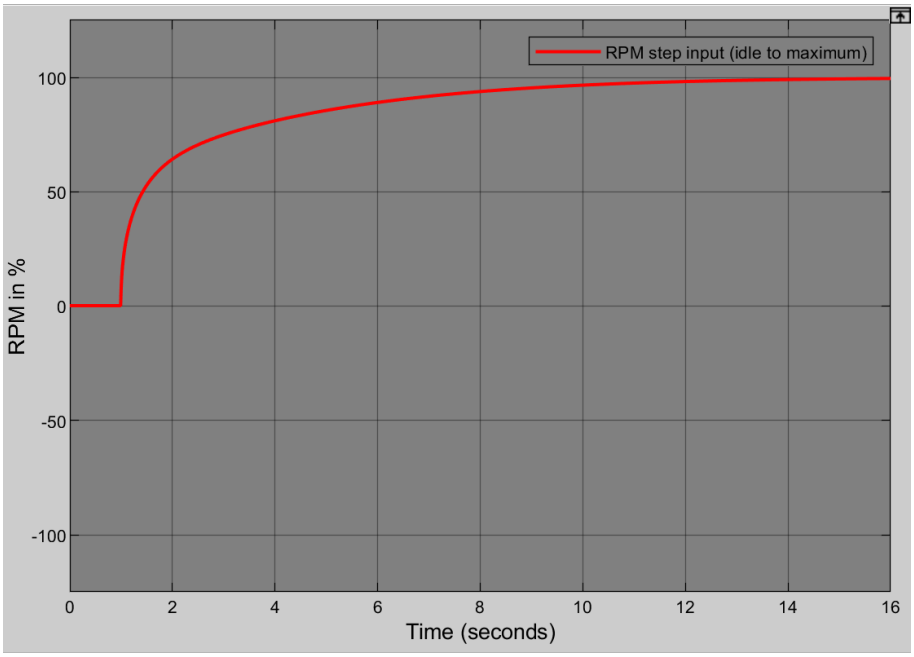


Figure 5.25: RPM response: Idle to maximum, using STEP function in Simulink® [46].

5.1.6 Logic of the FAT

For understanding the logic of the Java code, some diagrams are made to give an overview of the tests, shown in Figures 5.26 - 5.31. These diagrams are supplemented by the diagrams in 5.1.7.1, which shows an overview of the complete communication between B&R Automation Studio and Java. The yellow boxes symbolize the tests, the read ones are for communication, and the purple ones symbolize the logic of the FAT.

The yellow boxes from these diagrams can replace the yellow ones in the diagrams at 5.1.7.1, to give a complete overview of how the FAT and OPC UA communication is cooperating.

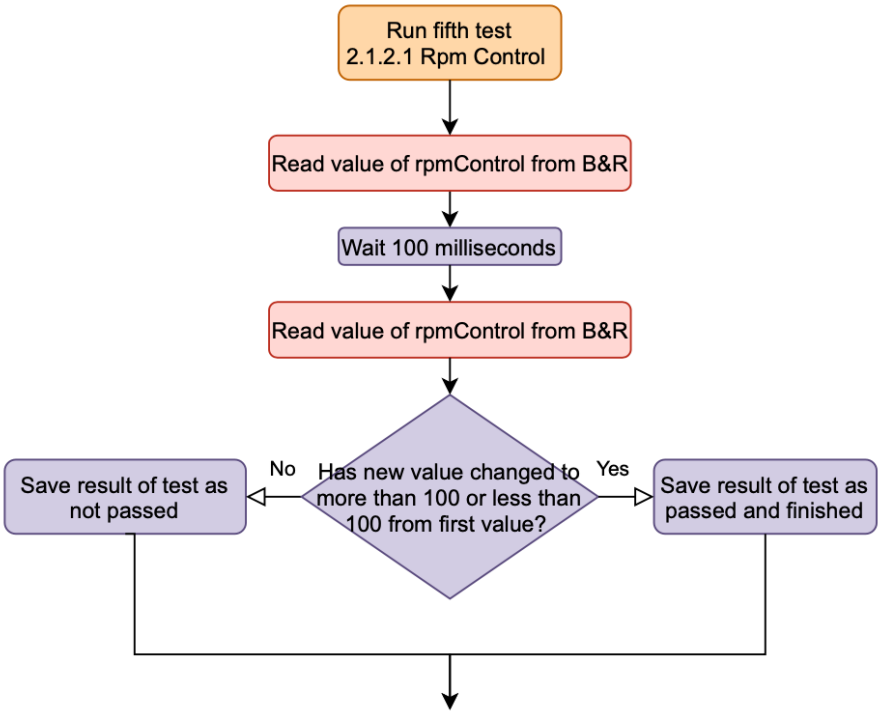


Figure 5.26: Overview of Rpm Control test logic [46].

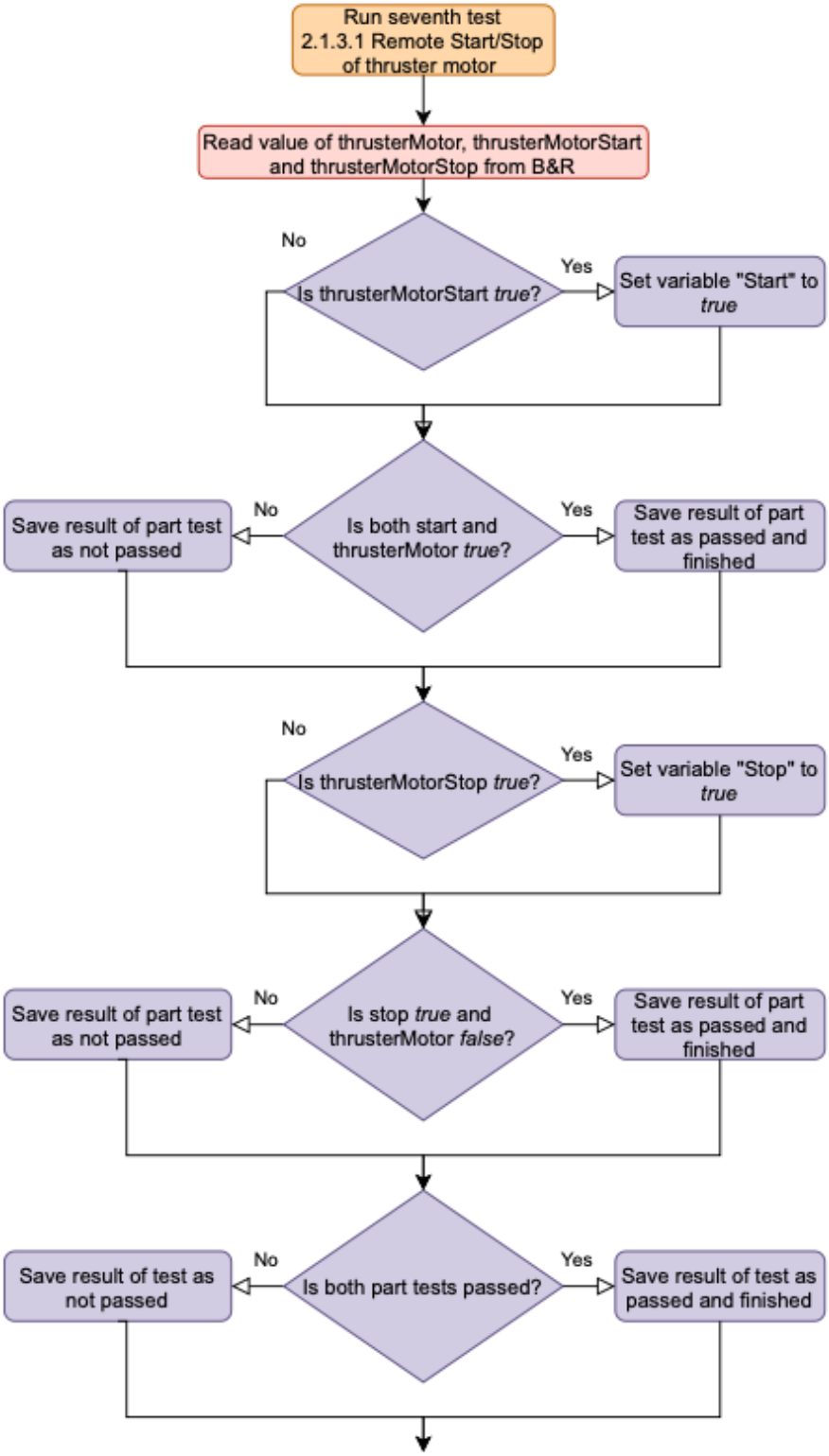


Figure 5.27: Overview of Remote Start/Stop of thruster motor test logic [46].

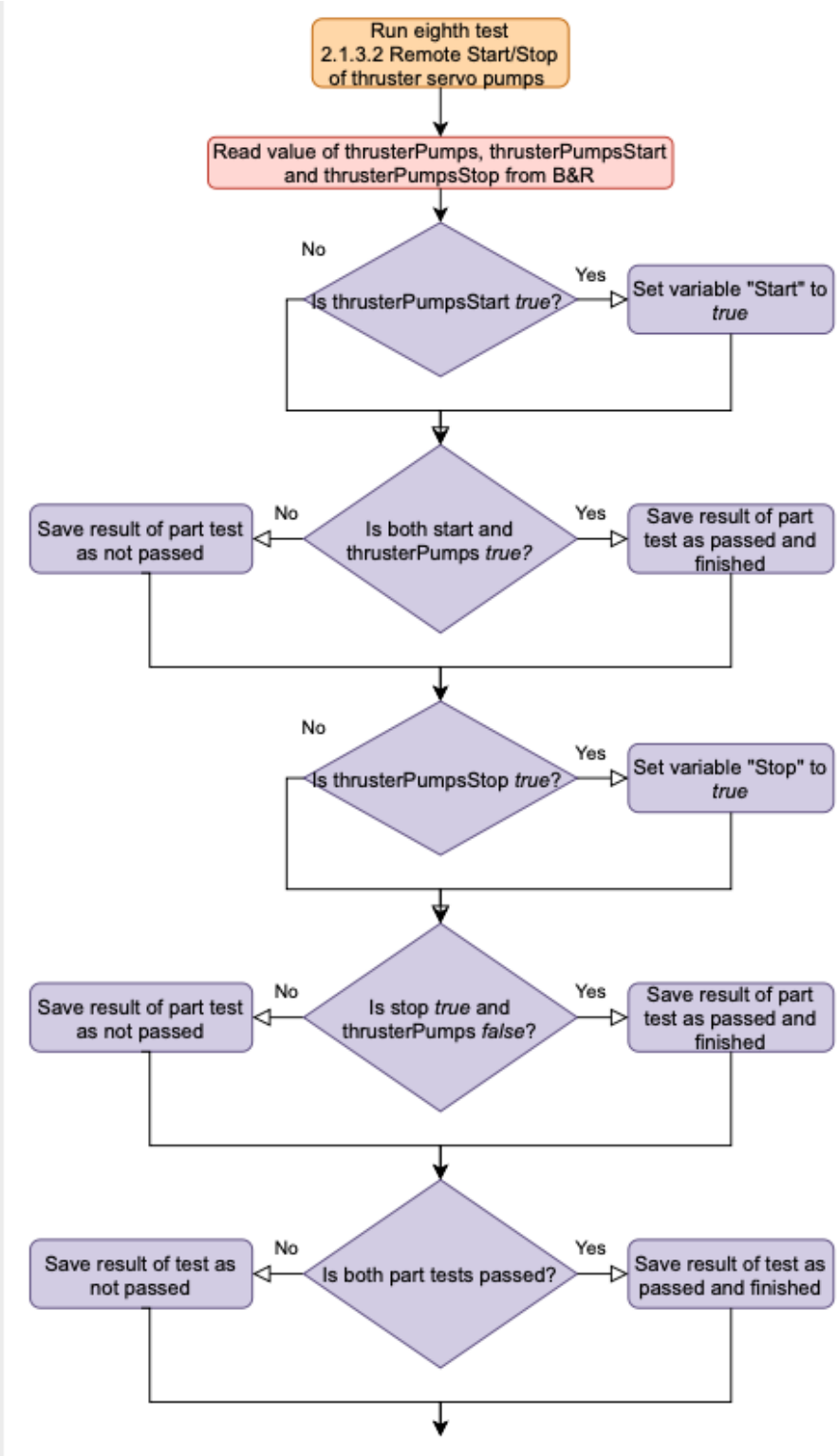


Figure 5.28: Overview of Remote Start/Stop of thruster servo pumps test logic [46].

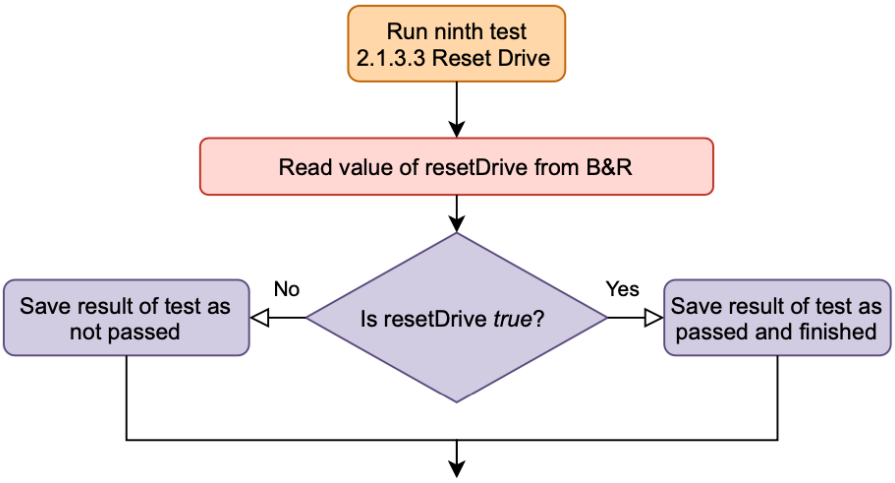


Figure 5.29: Overview of Reset Drive test logic [46].

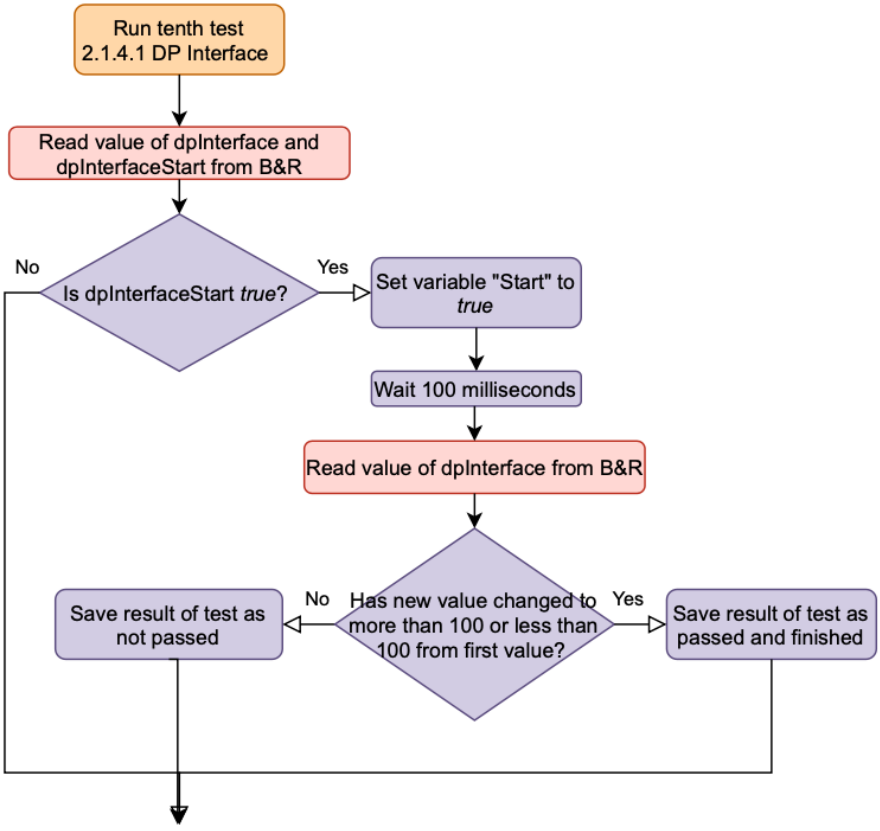


Figure 5.30: Overview of DP Interface test logic [46].

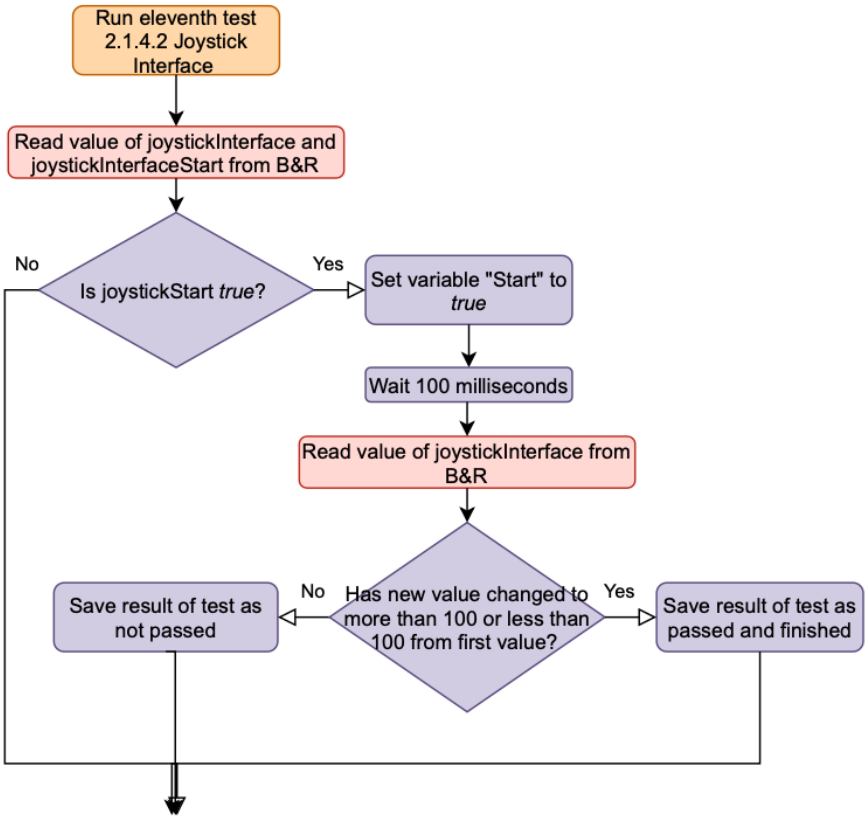


Figure 5.31: Overview of Joystick Interface test logic [46].

5.1.7 Communication protocol

Figure 5.32 showcases the system's setup when received, and Figure 5.33 shows the system's setup at the end of this project. As seen in the figures, the setup changed a lot. A more complex and improved version has replaced the old one.

The digitized buttons, simulations and the FAT are connected to the CPU using Ethernet/IP, Automation Studio Target for Simulink, and OPC UA. The MATLAB®/Simulink® program is uploaded into B&R Automation Studio using Automation Studio Target for Simulink, which connects to the CPU with Ethernet. FAT in Java is connected using OPC UA.

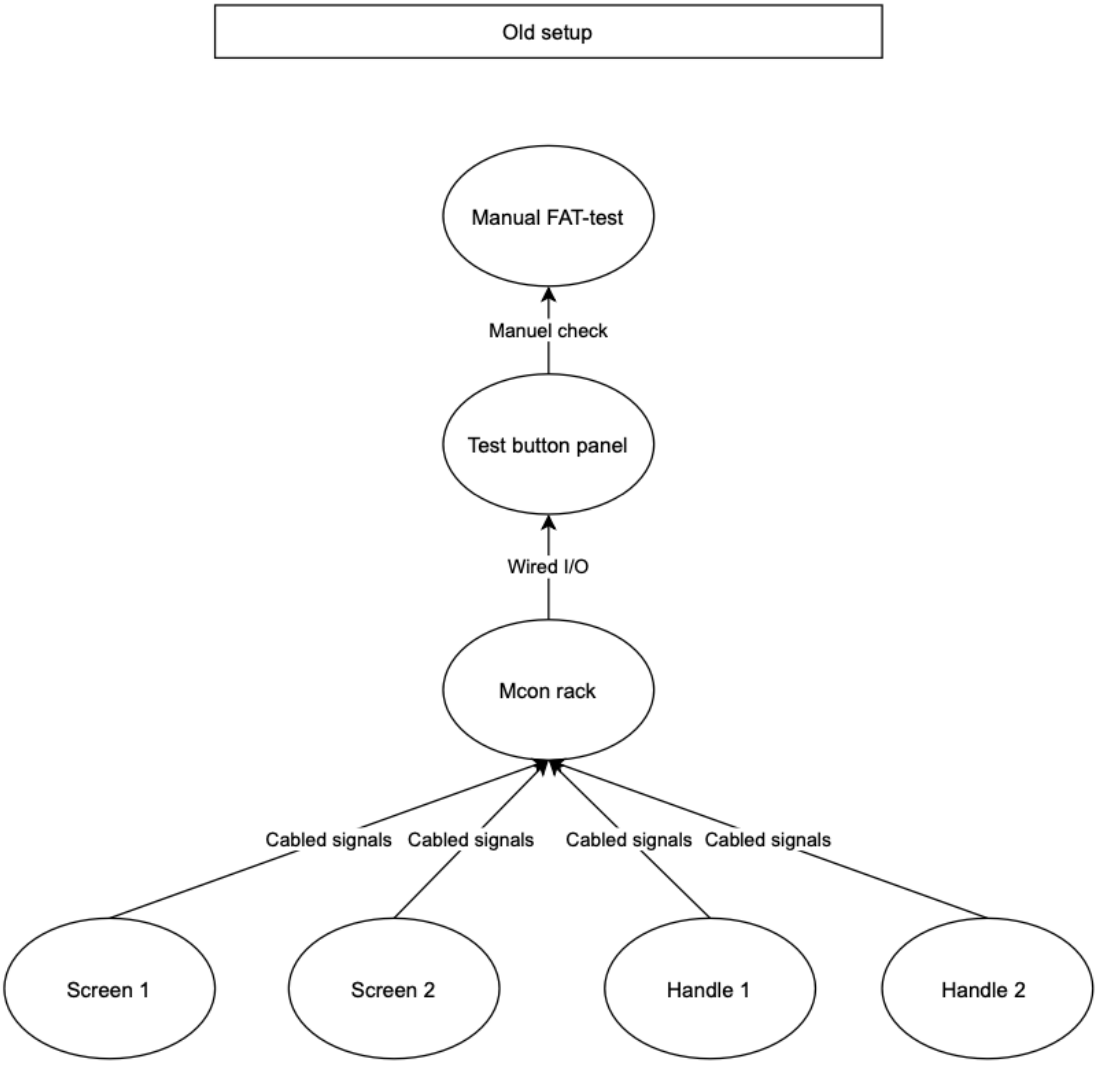


Figure 5.32: Overview of the old setup [46].

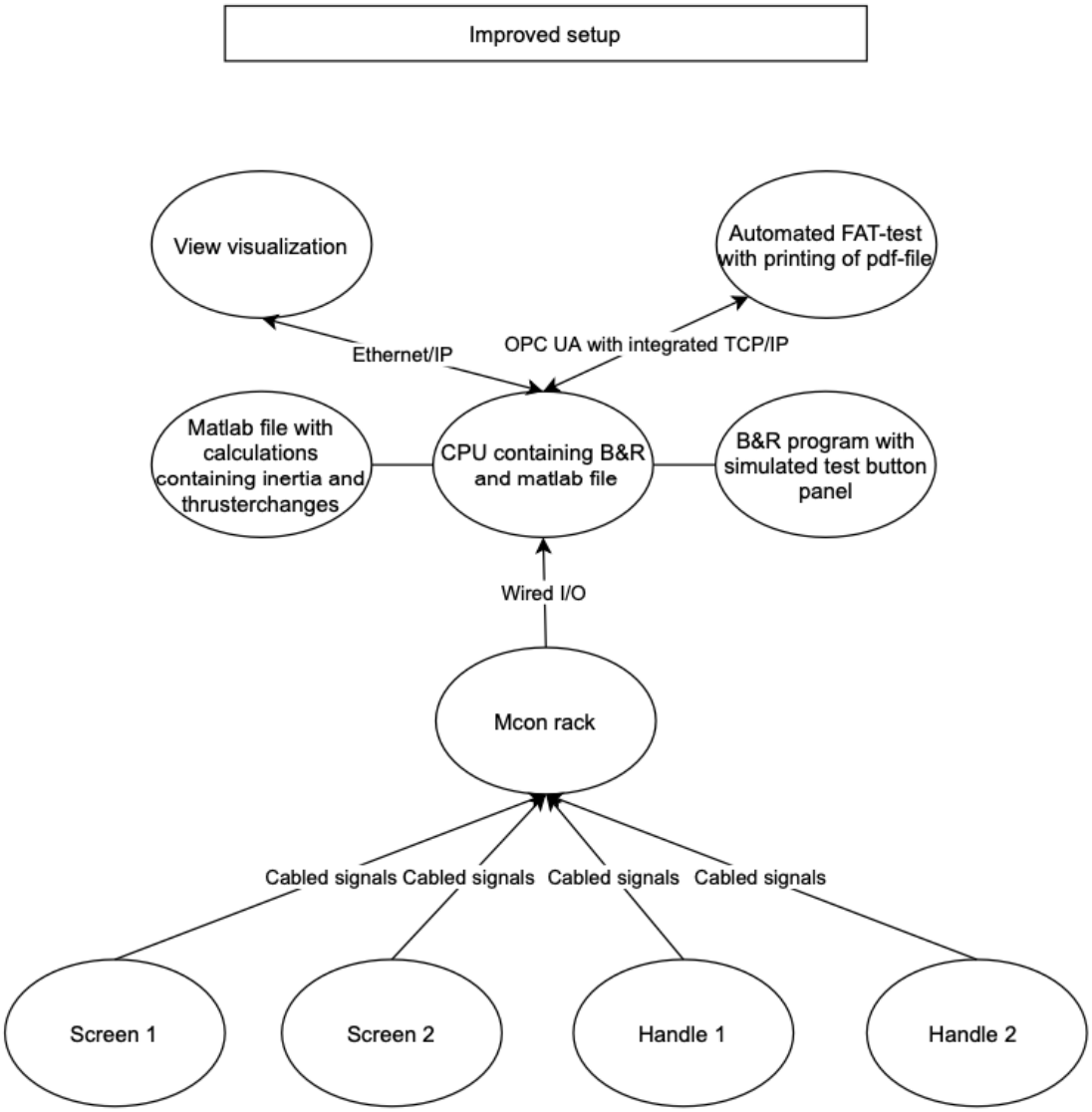


Figure 5.33: Overview of the improved setup [46].

5.1.7.1 OPC UA

In Figure 5.34 and 5.35 is an overview of the complete communication between B&R Automation Studio and Java. The figures are continuous and should be read from top to bottom. First, the OPC UA connection is established between the Java client and B&R server before Java executes the tests from the FAT, by reading the values of variables stores at B&R Automation Studio. The result of these tests are stored in Java and will be printed into a PDF at the end of the program. This is done by using the wireless connection, OPC UA. Because of this, the tests can

be performed easily and fast from anywhere close to the system.

Here the same principles occur as in 5.1.6, where the boxes in yellow symbolize the tests, which are closer described in section 5.1.6. The red boxes correlate to the communication, and the purple ones control the logic of which tests to be checked.

All the tests are reading the value of variables located at B&R Automation Studio for checking the tests. Some are as well reading multiple times to check if a value has changed. When all the tests have been checked once, the ones not yet passed are checked again by reading updated values from B&R Automation Studio until passed.

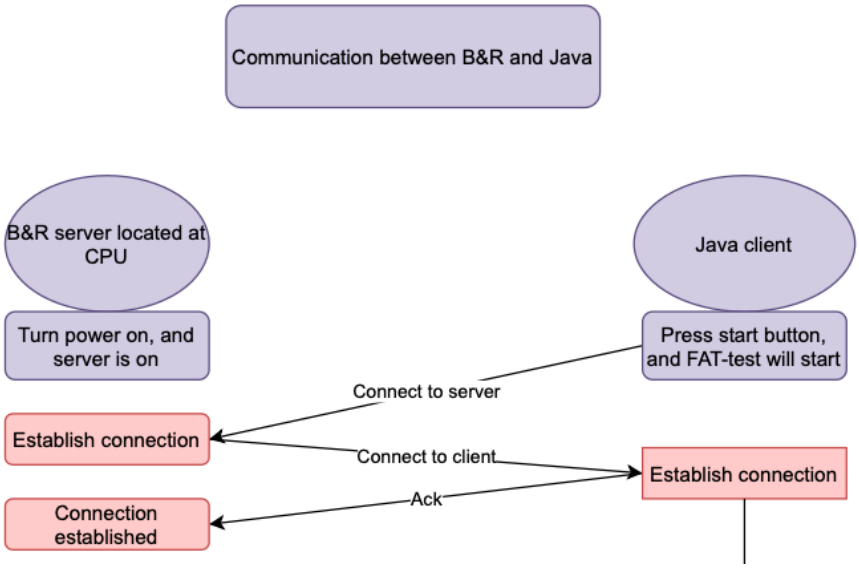


Figure 5.34: Overview of the OPC UA communication for establishing a connection [46].

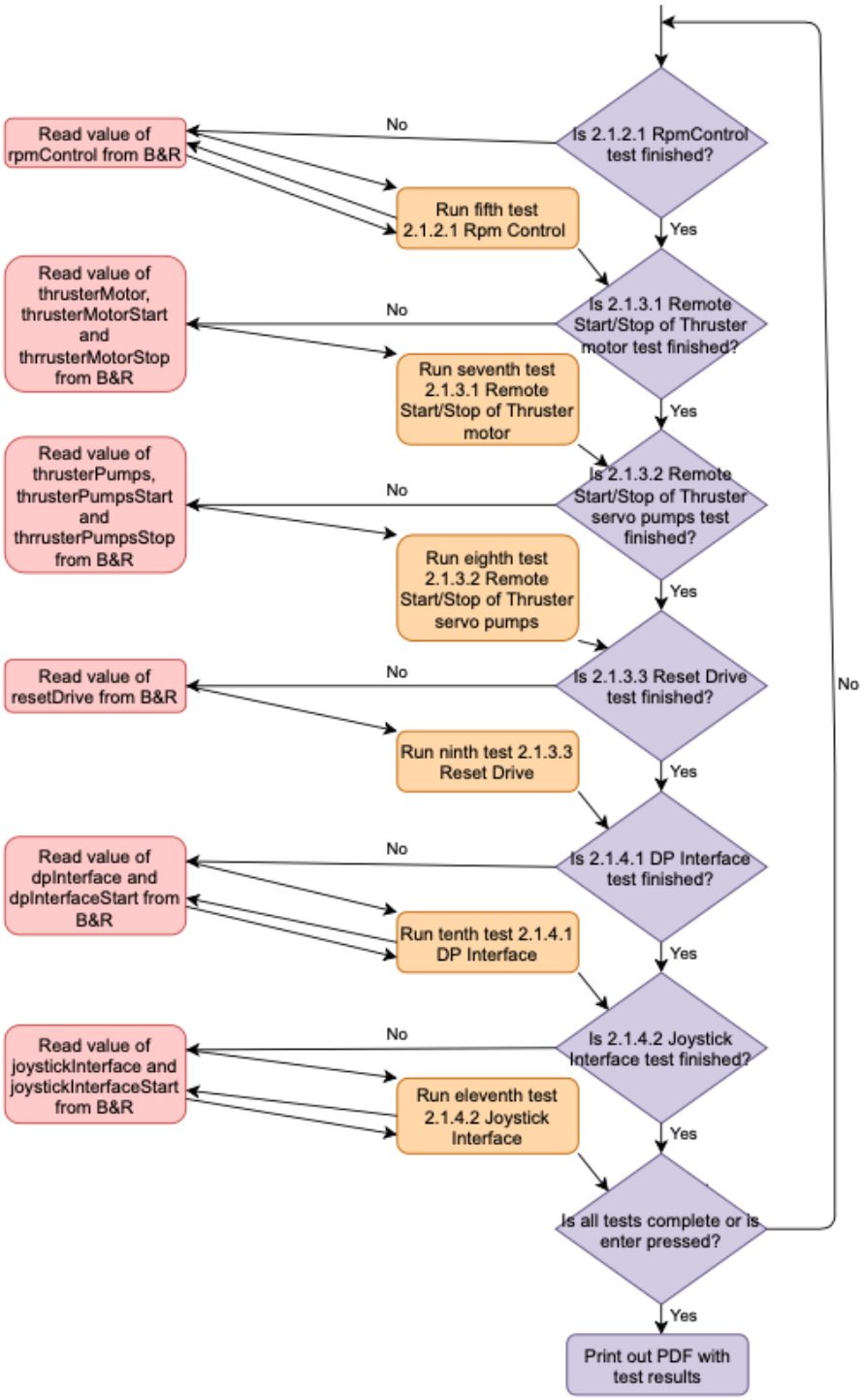


Figure 5.35: Overview of the OPC UA communication [46].

5.2 Final results

This section elaborates on results of the project as a whole containing all the different components.

5.2.1 Project

All the I/O signals and logic are handled in B&R Automation Studio, utilizing MATLAB®/Simulink® to model thruster systems and Java to create the semi-automatic FAT. The new version connects to the Mcon system using an Ethernet/IP cable, which significantly reduces time consumption in coupling. The old push-buttons are now automatically handled because of the logic in B&R Automation Studio, which reduces workload during the FAT. This solution also makes the test system more portable and reliable.

Figure 5.36 showcases the result of the whole project. The Mcon rack System is displayed in the left box, containing the essential parts for this project. The CPU box, in the middle, showcases all I/O signals needed to perform the FAT. The software used for performing the FAT and simulating the thrusters is shown in the box to the right.

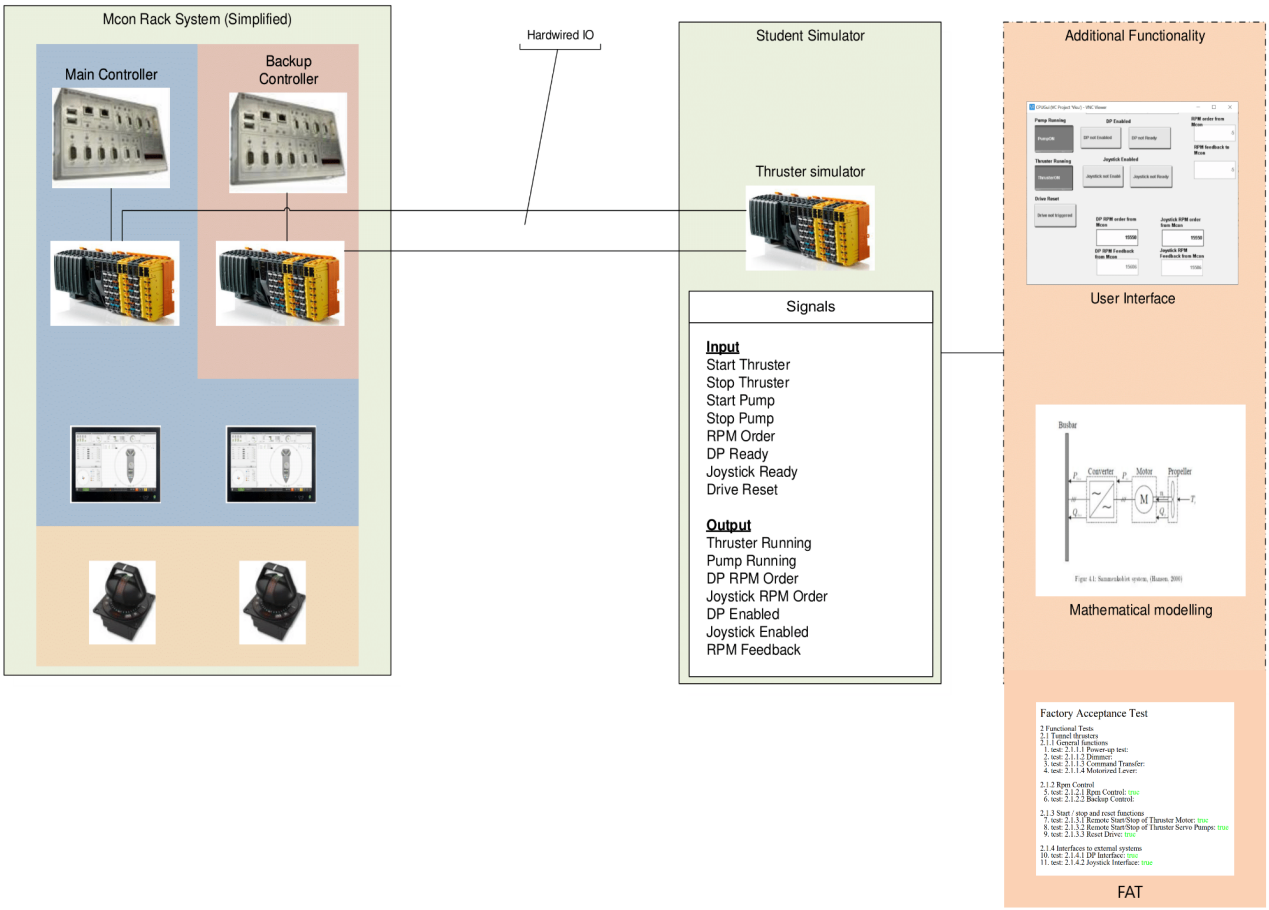


Figure 5.36: Desired setup [72]

5.2.2 Logic

The program developed in B&R Automation Studio contains a digitized system for saving the current values of buttons based on user input from the Mcon system and the B&R Automation Studio GUI. This system has replaced the testjigg delivered at the start of the project for simplifying testing. The signals chosen to digitize are based on the importance of the specific signal. Therefore not all the buttons on the testjigg have been digitized. The program is created based on the easy implementation of new buttons for a later possibility to expand this solution.

The GUI created in this program, as seen in Figure 5.37, enables the changing of the output values. The GUI also provides a clear overview of the current values of the different signals.

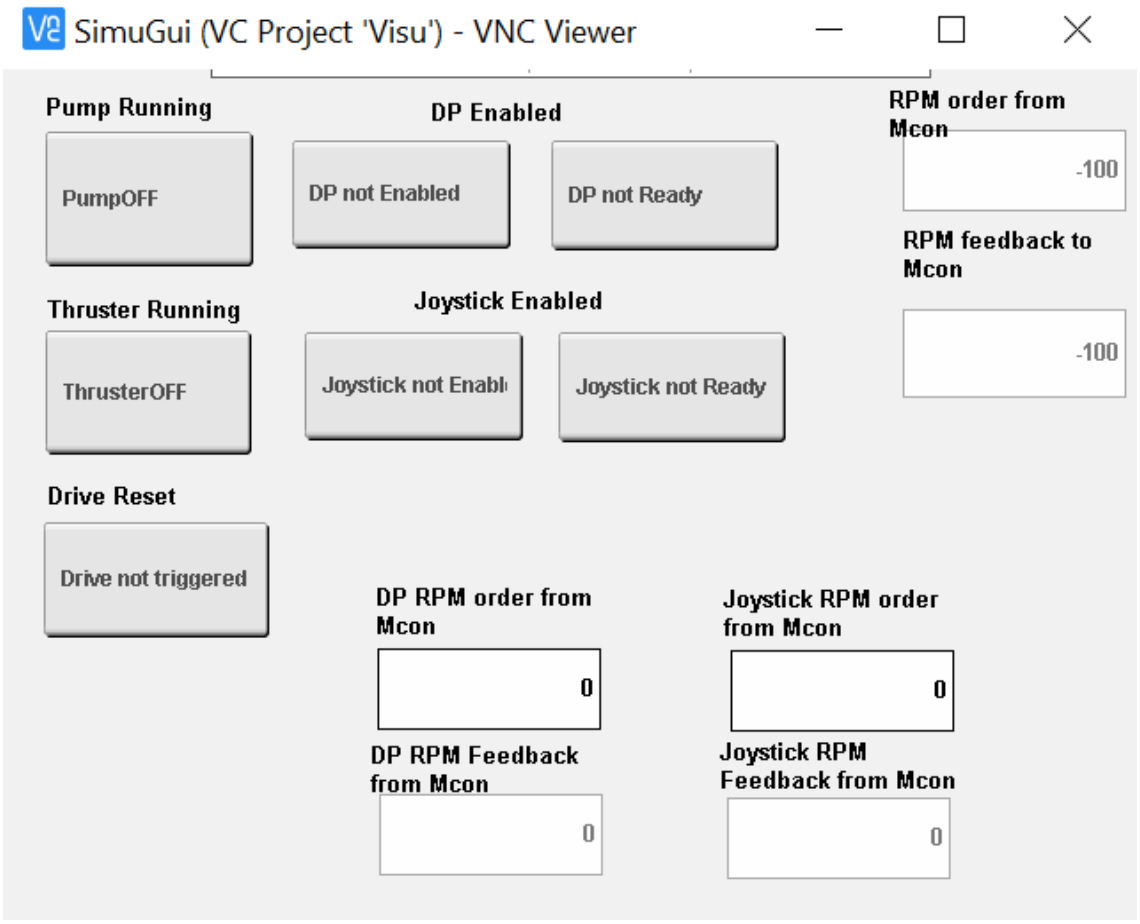


Figure 5.37: VNCViewerSimu [46].

5.2.3 FFP Model

The complete setup for testing of the FFP model can be seen in Figure 5.38, containing the communications blocks from Automation Studio Target for Simulink and the propeller model.

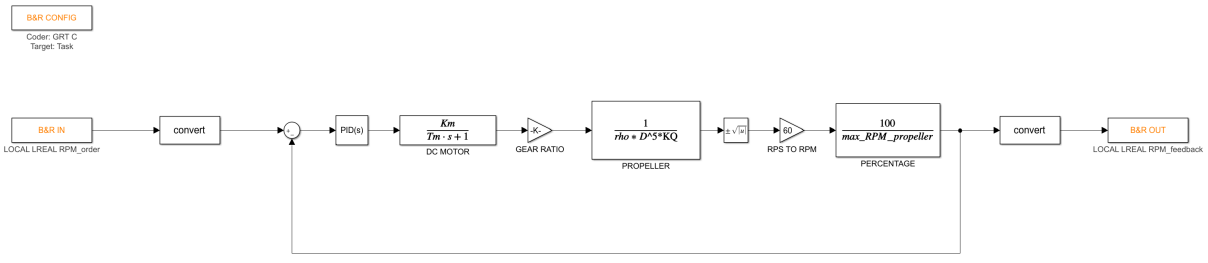


Figure 5.38: Overview of FPP model in Simulink® [46].

5.2.4 Mcon Thruster Control with Thruster Simulator for FPP

The graphical representation of the thruster response with Mcon as reference input is represented below. Plotting the data through Simulink® was not possible due to the program being uploaded to the CPU. A video monitoring the values in the B&R software was recorded during the tests. The results were plotted manually from the video, with a 1-second interval between each value recorded.

The input signal is an INT value, which represents values between 0 - 32767. A difference error of 685 is detected, which affects the results. When the input order is supposed to be 0, the actual value coming in is -685. This error value corresponds approximately to 4%.

The blue curve represent the reference signal and the orange curve represents the actual thruster feedback. Because the levers are manually operated, the results are affected and contain some uneven curves. The data sheet used for the graph plotting can be found in Appendix D.

In Figures 5.39 and 5.40, the settling time is low as expected, using little time to reach idle. Compared to the simulated results in Figures 5.16 and 5.17, the form of the curves and time to reach reference is approximately identical.

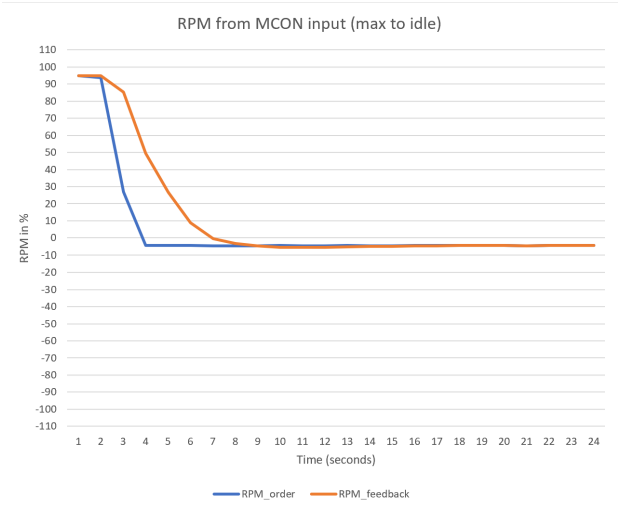


Figure 5.39: RPM from Mcon: Maximum value to idle [46].

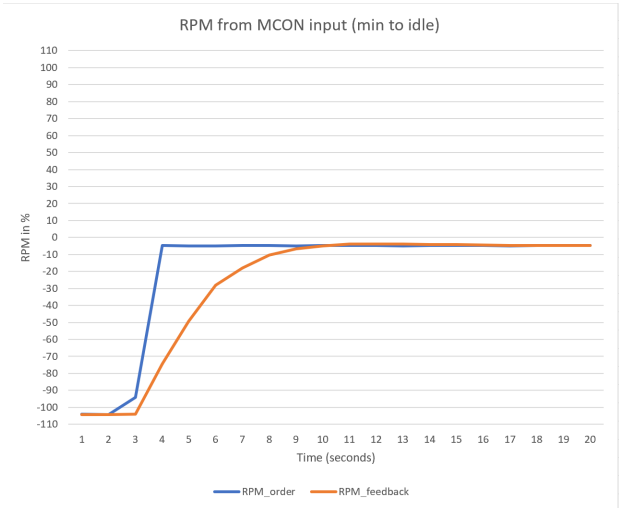


Figure 5.40: RPM from Mcon: Minimum value to idle [46].

As expected, the RPM build-up from idle takes longer than the RPM build down to idle. Figures

5.41 and 5.42 visualizes the build-up from idle to 100% and -100%. Compared to Figures 5.18 and 5.19, the results are here as well approximately identical.

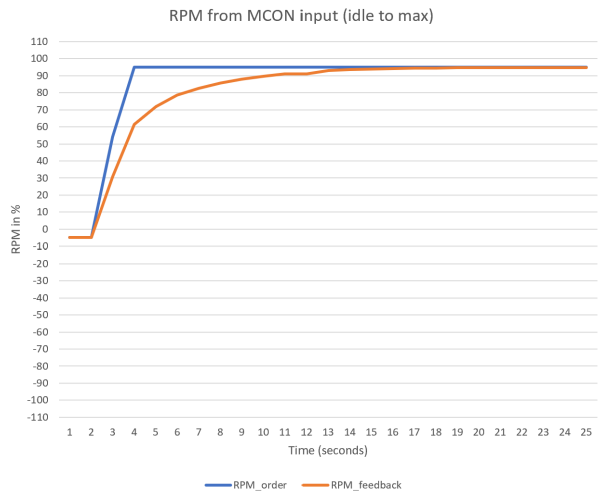


Figure 5.41: RPM from Mcon: Idle to maximum value [46].

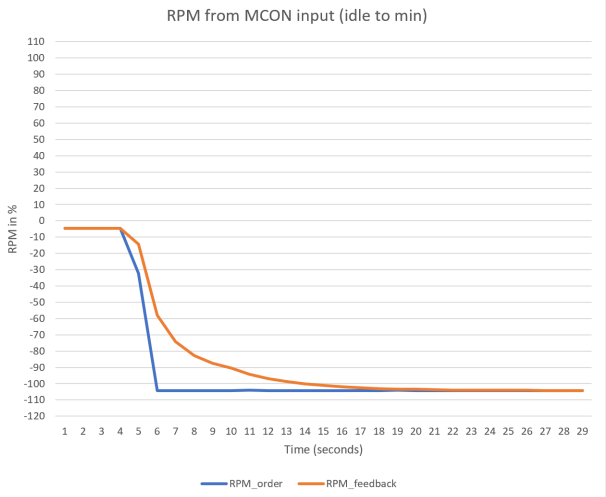


Figure 5.42: RPM from Mcon: Idle to minimum value [46].

Figures 5.43 and 5.44 visualize the RPM going from minimum to maximum and the opposite. Compared to Figures 5.20 and 5.21, the curve is not affected by the vertical linearization over idle. Other than that, the curve shape and settling time are similar.

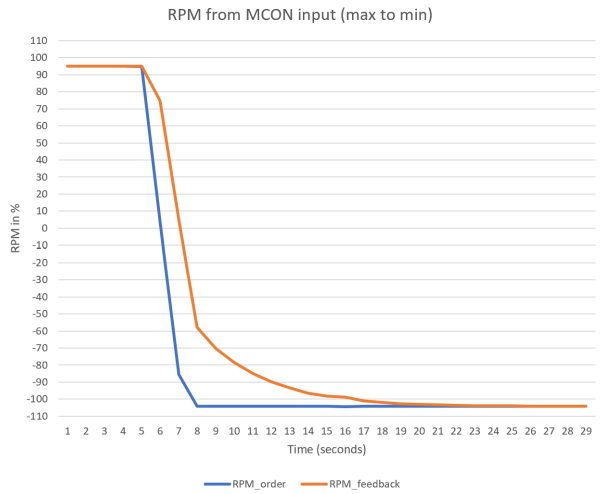


Figure 5.43: RPM from Mcon: maximum value to minimum value [46].

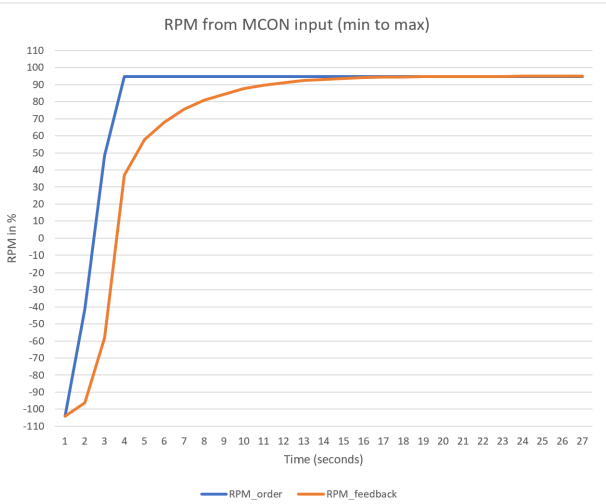


Figure 5.44: RPM from Mcon: minimum value to maximum value [46].

In Figure 5.45 and 5.46 the feedback of a reference signal of 50% and 25% is showcased. The

settling time and curve shape is as expected.

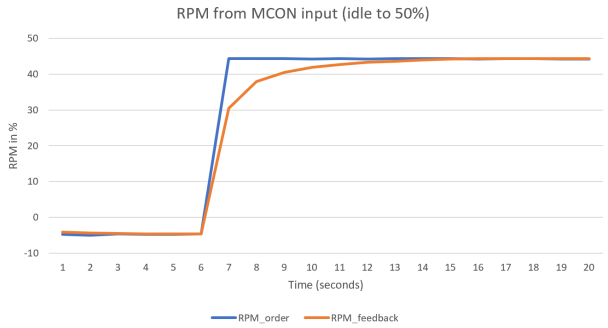


Figure 5.45: RPM from Mcon: Idle to 50%[46].

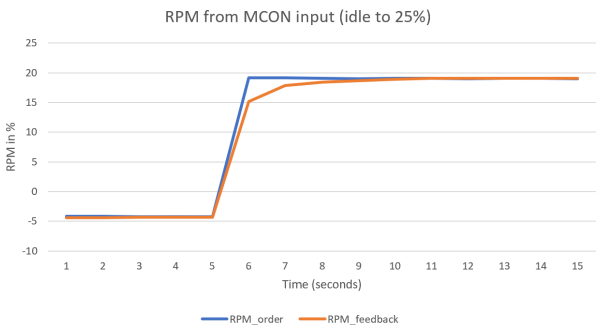
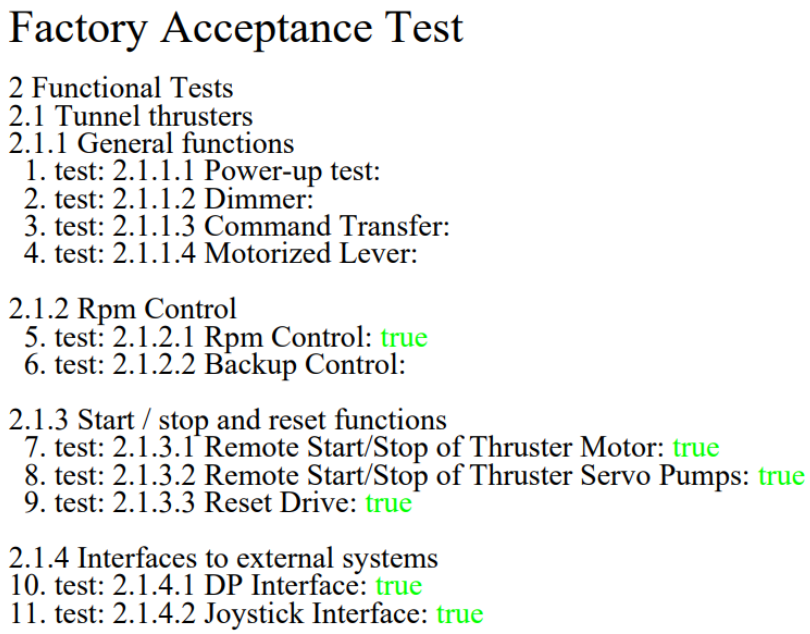


Figure 5.46: RPM from Mcon: Idle to 25% [46].

5.2.5 FAT

In Figure 5.47 the complete PDF-file of the FAT is showcased. Most of the tests were implemented, except for tests that require manual driving. The test results are colored red for faults and green for approval, making the results clear to see. Only one button must be clicked for running the FAT, and the rest will happen automatically, as user input is provided at the Mcon rack. The tests not implemented will still need to be checked manually, but the code provides a straightforward implementation of more tests if desired.

The Fat is automatically executed when all the tests have been completed. This can also be done by pressing enter on the keyboard along the way if desired. When the FAT execution is complete, a PDF file with the test results is printed, making the testing of the system easy and consistent, saving time, and avoiding possible critical errors.



Factory Acceptance Test

2 Functional Tests
2.1 Tunnel thrusters
2.1.1 General functions
1. test: 2.1.1.1 Power-up test:
2. test: 2.1.1.2 Dimmer:
3. test: 2.1.1.3 Command Transfer:
4. test: 2.1.1.4 Motorized Lever:

2.1.2 Rpm Control
5. test: 2.1.2.1 Rpm Control: true
6. test: 2.1.2.2 Backup Control:

2.1.3 Start / stop and reset functions
7. test: 2.1.3.1 Remote Start/Stop of Thruster Motor: true
8. test: 2.1.3.2 Remote Start/Stop of Thruster Servo Pumps: true
9. test: 2.1.3.3 Reset Drive: true

2.1.4 Interfaces to external systems
10. test: 2.1.4.1 DP Interface: true
11. test: 2.1.4.2 Joystick Interface: true

Figure 5.47: The printed PDF-file containing the FAT results [46].

5.2.5.1 Functional tests

Table 5.4 shows which tests has been implemented from section 2.1 *Functional tests* in the FAT. The buttons for the tests not included have not been automatized in B&R Automation Studio, and the values of these are therefore not possible to extract. However, the number of tests implemented is more than anticipated at the start of the project and is considered a success.

FAT	Implemented
2.1.1.1 Power-up test	x
2.1.1.2 Dimmer	x
2.1.1.3 Command Transfer	x
2.1.1.4 Motorized Lever	x
2.1.2.1 Rpm Control	implemented
2.1.2.2 Backup Control	x
2.1.3.1 Remote Start/Stop of thruster motor	implemented
2.1.3.2 Remote Start/Stop of thruster servo pumps	implemented
2.1.3.3 Reset Drive	implemented
2.1.4.1 DP Interface	implemented
2.1.4.2 Joystick Interface	implemented

Table 5.4: Tests implemented in the semi-automatic FAT

5.2.5.2 Faults and consequences

Section 3.1 *Faults and consequences* in the FAT contains several tests that need manual execution. Running these tests, B&R CPU has to stay connected at all times in order to stay in control even when faults occur. During all the tests, the project program worked as desired and is therefore considered a success.

5.3 Class Diagram

Figure 5.48 is a class diagram of the code project of the FAT in Java. This diagram is showing how the classes communicate with each other and what methods each of them contains. It is possible to run either the class "RunTestsVirtual" or the class "RunTestsMilo" for executing the program. Either way, the class "TestRunner" will be called, as well as "VirtualConnection" or "MiloClient". The last-mentioned option will as well run the classes "KeyStoreLoader" and "ClientExample". At last, "CreatePDF" and "Connection" will be used.

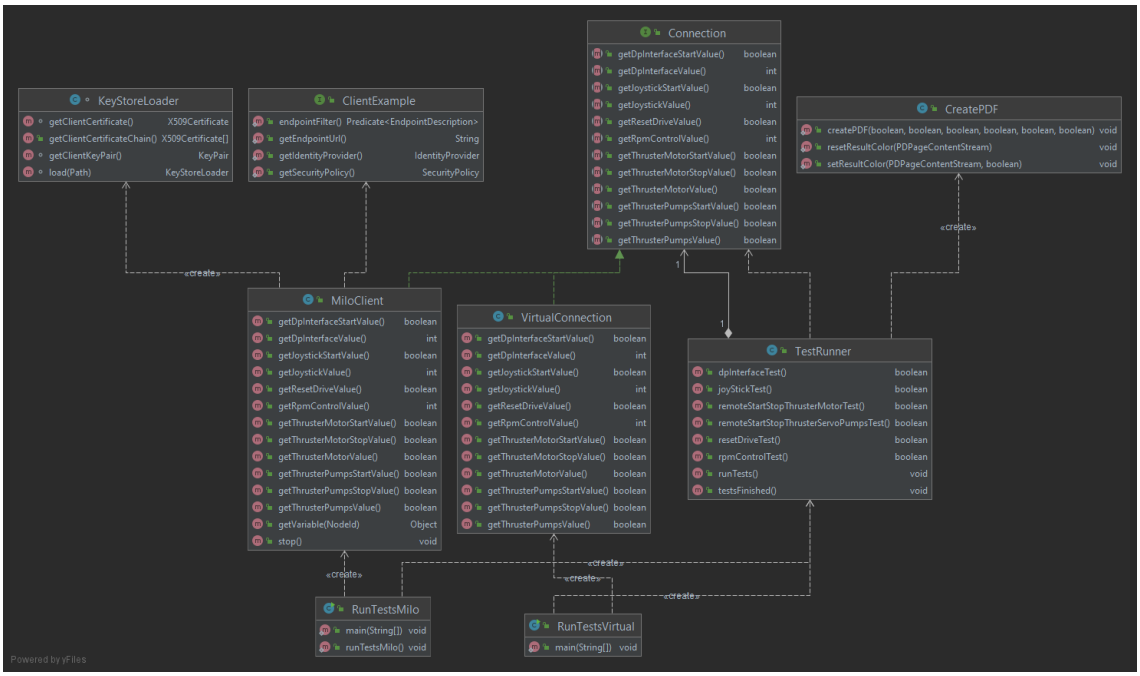


Figure 5.48: Overview of the classes and methods of the FAT in Java [46].

The code project is executed with low coupling and high cohesion by dividing the tasks and methods into different classes. Classes know the least amount of information about other classes, and each class is responsible for solving one task. Only the needed code must be run by following these terms, making the project fast and well structured, and easy to reuse.

Chapter 6

Discussion

This section consists of a discussion of the project, the results and possible ideas for further work.

6.1 Project

Technology and automation are playing an increasingly bigger part in the industry. Automation and digitization is replacing heavier and slower physical solutions. Companies are taking advantage of these technologies to become more efficient and improve their productivity. This is highly relevant in an industry where automation, cost-saving, and efficiency are important factors.

This way of using automation can be tied back to Lean manufacturing, which we have discussed in section 1.1.1 of our introduction. The solution provided in this thesis can help Kongsberg Maritime cut down on waste, as fewer materials are needed to test the system. Fewer materials will also reduce material cost during testing. The solution of this project has created a system that use less time, thus making the testing more efficient. The testing will also be made more consequent by reducing the change of human error.

6.2 B&R Automation Studio

As a result of a lack of knowledge about B&R Automation Studio, a lot of time was spent learning the basics of the application. After having done that, however, the design process improved considerably and rapidly.

As the variables start/stop in B&R Automation Studio are inputs that had to change manually in simulation mode, some confusion occurred as to how to set them back to *FALSE* automatically. This confusion resulted in unnecessary time spent on figuring out a way to do so. After asking for clarification, it was cleared that this was not necessary to think about, as it will be a pulse that is high for a few seconds.
After this delay, the rest of the variables were added, and logic was made based on this knowledge.

All the necessary functions holding the variables needed in the semi-automatic FAT and modeling of the propellers were created regarding the project, as well as a GUI for overview and external control.

6.3 FFP Model

A model of a motor and propeller is, in reality, complicated and needs many parameters that were not accessible. In addition, the main goal for the project was to create a simple model that will simulate more realistic feedback and the possibility to expand later on. Therefore, some simplifications were made such that the modeling would match the competence level, time limit, and requirements of this project. In order to create a more realistic model, these parameters should be included.

Gear ratio

The gear ratio can be found as a constant in the data sheet for a specific vessel. This constant could replace the solution used in this project, using max propeller RPM and the max motor RPM to calculate the gear ratio. A constant is more appropriate to use if a specific gear is desired. The model characteristics would become more accurate.

6.4 FFP testing

The tests performed in simulation mode and the tests performed with the Mcon system as a reference displays similar curves and approximately the same settling time for the same input order and origin. One of the desired model results was to get approximately 8 - 10 seconds from idle to maximum speed for the DC motor. The system uses approximately 12 seconds, which is slower than the settling time of the DC motor. This decrease in settling time is caused by the increased resistance from the propellers. This is also something that could be tuned faster with the PID controller or choosing different values for the time constant of the DC motor.

Mcon Thruster Control with Thruster Simulator

All the graphical representations of the thruster response gave the expected and desired results, especially regarding the curve shape and settling time, which was the main focus. In compassion to the simulated tests, the response was approximately equal. The success of getting similar results means that changes in the model can be developed using only the simulation in both Simulink® and connected to B&R Automation Studio in Simulation mode.

The graphs can contain inaccuracies because of imprecise movement levers and human error in logging and plotting.

6.5 Design for FAT

6.5.1 Result of FAT

All the test results in the FAT has been passed. If all the tests are passed during testing of an actual Mcon thruster Control, it would mean that no problems has been uncovered for the thruster, making it ready for installation on a vessel. In our case it means that there are no problems with the testing or with the Mcon rack.

6.5.2 Continuous Reading of Values

At first, the FAT would be running each test once, reading and writing values of corresponding variables in B&R Automation Studio. It was, however, discovered that B&R Automation Studio could not send variables to the Mcon system. The Java code, therefore, had to be changed for responding to changes of values for the variables at B&R Automation Studio, based on user input from the Mcon system. A while loop created around the tests would allow all the tests to read values of their corresponding variables. When a test becomes complete, it is not included in the while loop anymore. When all tests are complete, the while loop becomes terminated, and the rest of the code can continue, printing out the result of the tests into the PDF.

6.5.3 Threads

We considered using threads in this project, but since the FAT does not need to be run in real-time or perform multiple tasks simultaneously, there was no need for this implementation. The code should not be more complex than necessary if there are no real advantages. Threads also provide secure sending of data and ensure that the data does not become corrupt, which TCP/IP also can ensure.

6.5.4 Implementation of tests

The FAT consists of more tests than the ones implemented in this project. Some of the tests are hard to automate and are therefore not created. It has as well not been enough time to implement them all. Since we expected this at the start of the project, it has not caused any unseen problems or delays. The ones implemented have been prioritized based on importance, the toughness of implementation, and the necessity for the complete solution.

6.6 Communication

One of the main obstacles was the communication between the Java and B&R Automation Studio due to a lack of knowledge and experience. Communication between the other components was easier to implement as there are adequate communication tools for these parts.

6.6.1 Automation Studio Target

For communications between MATLAB®/Simulink® and B&R Automation Studio, the choice was relatively easy and worked well. B&R Automation Studio Target for Simulink supports both B&R Automation Studio and Simulink® and is integrated to make communication better and more straightforward.

There were some issues in the beginning which resulted in rejected connection. After some troubleshooting, it was discovered that the B&R Automation Studio needed upgrading. After upgrading *Automation Runtime* and *B&R Automation Studio*, the connection worked perfectly.

6.6.2 OPC UA

6.6.2.1 Choosing Method for Sending Multiple Parameters

TCP/IP usually only allows the sending of one parameter at once. The encountering of possibilities for sending multiple parameters uncovered three possible options:

- Serialization
- XML
- JSON

At first sight, the group believed that serialization would only be possible to use if the server and the client would be running on the same computer. Using serialization was, therefore, undesirable since the B&R server would be running at a CPU. JSON was then the second choice since some group members had some experience with JavaScript's programming language, used for JSON. Before testing this, the discovery of serialization inside Eclipse Milo uncovered itself. Eclipse Milo, used for creating an OPC UA client, has serialization implemented as the default method for sending and receiving data, even though the server and client are running on different computers. Serialization is, however, implemented in a part of the Eclipse Milo code, not being used for this project.

This project never needs to read more than one variable at once, and the implementation of serialization needs a lot more complicated code. This would only make the code slower and more complex than necessary, leading to the decision to exclude the possibility of sending multiple parameters.

6.6.2.2 Problems Connecting B&R Automation Studio and Java

There were problems when trying to connect the Java client with the B&R server. The displaying of the error message "connection refused" resulted in the trial of multiple solution options.

From the client-side, the following methods were attempted for connecting to the B&R server:

- Connect with Eclipse Milo client created in Java
  Eclipse Milo is the client created in Java to communicate with B&R Automation Studio.
- Connect to both IP address and port number.
  Since it was unsure if the connection required connection to IP address or a port number, both connections were tested, with IP-address 127.0.0.1 for the simulation, 192.168.1.110 for the CPU with port number 4840.
- Connect with Telnet
  Used the command window and wrote "telnet 127.0.0.1 4840."
- Connect with localhost in browser
  Opened a browser and searched for "localhost:4840".
- Turn off the firewall
  It was thought that the firewall was causing some problems for the communication and was, therefore, turned off.

From the server-side, the following was attempted:

- Activate OPC UA System in B&R Automation Studio
  The OPC UA System would have to be activated in B&R Automation Studio to create an OPC UA server.

- Use both alternates for Automation Runtime type: AR Simulation and AR Windows
  It was unsure which of the Automation Runtime types to use and tested both.
- Turn simulation on and off
  It was thought that the server would open when the simulation got turned on.

All the options mentioned above were combined until every combination had been tested. None of them worked, and it was considered to make an OPC UA server in Java to test, but this was much more complicated than anticipated.

After some research, it was discovered that maybe some other steps were necessary. The considered ones were:

- Add library "AsOPCUac" for OPC UA in B&R Automation Studio
  This library was found to be pre-installed in all B&R programs.
- Add an "X20BC008U" bus controller to B&R Automation Studio
  This bus controller is for remote I/Os. Since the I/Os are physically connected, with Ethernet, to the PLC, containing the B&R program, this was not needed.
- Run the B&R program from a CPU
  The connection was still not available after connecting the program to a CPU.

When none of the previous options solved the problem, the B&R support in Denmark was contacted. They suggested trying many of the options above, as well as using the program UaExpert. We downloaded and used this program as an OPC UA client for connecting to the OPC UA server at B&R Automation Studio. The connection was still refused, creating error messages in UaExpert. After that, the whole B&R program was sent to support. There the connection between UaExpert and B&R Automation Studio worked at once, using the same settings and IP addresses as we did. They had four possible reasons for why the connection worked with them and not with us:

- They used B&R Automation Studio version 4.9, and we used 4.2

We, therefore, asked to get a time-limited trial for this version to check if this would solve the problem.

- Could be an anti-virus issue
  The anti-virus was turned off, without any change of result.

- Could be a firewall issue
  The whole firewall was turned off, without any change of result.

- Could happen if windows were running on a virtual machine
  This was not the case since the program was running on a windows machine.

After that, a lecturer at NTNU was contacted and gave the following advice:

- Run the connection from a different computer
  This was already tested out, using a computer at the NTNU network and one at the home network.

- Run as administrator
  The connection could be refused if the administrator user were not the one in use. This user was, however, the one in use already.

- Could be firewall or anti-virus issues
  These issues were again tested but without any change of result.

- Connect with "netstat -a"
  Opened the command window and wrote "netstat -a".This command shows a list of all open ports, where the connection 127.0.0.1:4840 was listed, with the state "TIME_WAIT".

B&R support was again contacted, and a meeting was arranged. There the last piece of the puzzle was provided, clicking on *Tools -> offline install*, as explained in the last part of section 4.7.3.1. Now the server-side of the connection could be accessed and used.

6.7 Further Work

Closing up to the end of this project, the lack of time limits how much prevalence could be achieved. Some additional improvements are proposed for further work, whereas these improvements are not essential for the project.

6.7.1 Variables in B&R Automation Studio

As the logic in B&R Automation Studio has been based on the testjigg functions and the FAT, some considerations were not taken into account. Such a consideration is that the RPM has not been limited as optimized. For creating the system more realistic, the RPM should not be able to steer without starting the pumps and thruster. This limitation is already implemented for the thruster but remains to be implemented for the RPM and reset drive.

More buttons can be added for further work, broadening the possibilities for the Mcon Control System testing. Test simulations could also benefit from this, as the system can be tested as it would be when installed on a vessel. Developing and updating the system can be easily tested, providing a better understanding for the developer.

The logic is created for easy expansion and implementation of new buttons or other logic. The GUI is also possible to expand, implementing more functionality regarding more buttons and an improved layout.

6.7.2 Propeller types

Beginning this project, one of the desired functions was the ability to choose the propeller type to simulate. In Simulink®, there are therefore made two different propellers, FP and CP propellers, which are the most common propellers. For further work, other propeller types could be modeled such that the system could be tested on the propeller that it is going to control when implemented on a vessel.

6.7.3 Modelling

Simplifications and assumptions done when making the models in Simulink® mean that the result will not be accurate. The model will need to be modified in order to be more accurate in relation to reality. Some ideas on how to do this are described in the following sub-sections.

6.7.4 CPP model

The CPP model used in this thesis was used to mimic the behavior of an actual CPP system. The CPP model is, however, not mathematically accurate. In our model of the CPP system, the pitch value was multiplied into the equation for thrust produced, and the inverse value of the pitch was merged with the propeller block responsible for the RPM calculations. While this solution may give us a system that responds similarly to a CPP system, the values will not be correct. In order to get a mathematically correct value for this system, K_T and K_q (thrust- and torque coefficient) has to be calculated for each value of the pitch, as the thrust- and torque coefficient is the function of the advance speed, pitch ratio, RPS, and water density. Changing the pitch would alter the pitch ratio, which in turn would alter the K_t and K_q. In our model, the torque coefficient is a constant and does not change when the pitch changes. This results in an inaccurate thrust, and RPM response since our model multiplies the pitch value with the thrust/RPM equation on top of the already existing thrust- and torque coefficients.

Due to time constraints and the requirements of this thesis, the group determined that the solution provided in this thesis would be a good enough estimation of a CPP, as our primary concern with the thruster simulation was to prove that MATLAB®/Simulink® could be used in conjunction with B&R Automation Studio and Mcon to simulate a model of a thruster. A better and more realistic model should be implemented for use in development or testing aimed at propeller/-thruster performance.

The model of the pitch actuator is also something that could be improved upon, as it is a simplified version of a realistic pitch actuator. In Figure 5.15 of the result section, a sharp break can be seen in the RPM graph. The reason for this sharp change of direction is because of the nature of the pitch actuator model.

The pitch model is a 1. order system and produces a steep, almost linear graph in its starting phase, much like the simplified DC motor seen in Figure 4.28. This response kicks in slightly after the RPM and causes a big increase in the pitch value over a short period. In Figure 4.38 we can see that the pitch is merged with the RPM transfer function to increase the resistance as the pitch increases. When the large increase in pitch occurs over a very short period, the resistance of the propeller increases significantly in a short period. This sudden change in resistance will make the RPM dip slightly before being corrected by the PID controller. A more realistic model with a more natural starting phase would probably resolve this issue. The same applies to the DC motor and propeller model, as this system also has a rather aggressive and fast starting phase.

6.7.4.1 Modification of FPP and CPP model

The models of the FFP and CPP were simplified and are therefore missing some parameters that would make the models work even more realistic. Such parameters could be the size and weight of the vessel or current in the water.

6.7.4.2 Modification of motor model

A simple DC motor was modeled when in reality, an AC motor is used. An AC motor is more complicated than a DC motor but will give a more accurate result. In addition, implementing a drive would also increase the quality of the model.

For this thesis, a first order transfer function was chosen for our DC motor. While this solution may have given a good estimation of the behaviour of a real DC motor, some benefits could be seen from switching to a system of higher order. One of the main benefits of using a DC motor with higher order is a more realistic response in the starting phase of the system due to the inertia of the system. This would make the FPP model behave more accurately in the starting phases of the system response. A higher order system would take into account the mass of the motor, shaft and propeller. This would likely also resolve the issue with the sharp break in the graph of the CPP model as explained in subsection 6.7.4.

6.7.5 Components

By simulating more components, such as different motors, the components could be put together for a complete simulation for a vessel. This allows putting together a specific pre-made motor and propeller to simulate a good and realistic behavior.

6.7.6 Testing of Mcon Thruster Control with Thruster Simulator

All the tests performed on the thruster model controlled by the Mcon Thruster Control was plotted manually. This solution for a graphical representation is not ideal. For better and more accurate presentation of the curves, a digital system should handle the plotting.

6.7.7 Tests in Java

Many of the tests from the FAT has been made semi-automatic in Java. It is, however, possible to implement more by adding variables for these tests in B&R Automation Studio. The implementation of more tests is made easy for this purpose.

6.7.7.1 Weakness in FAT levers

The tests regarding the input from levers, such as *Rpm Control* will pass if the values change some. However, these tests do not check if the value corresponds to the correct tuning. Checking the tuning could therefore be a possible improvement in further work.

6.7.8 Communication

In continued work, the OPC UA communication protocol can be replaced by another protocol if desired. The advantage of this change is unknown but possibly not significant since OPC UA already is fulfilling the communication between B&R Automation Studio and Java as desired.

6.8 Experiences

6.8.1 Group Dynamics

Our diverse backgrounds in electives have facilitated good dynamics and a broad knowledge area, which has come in handy during the project. All the participants in the group have worked steadily and well with the project along the way. Some lack of communication within the group and across the different tasks resulted in some minor misunderstandings, but they were relatively quickly set straight. Overall there has been good cooperation between everyone, which has resulted in a neat and nice progress.

6.8.2 Progress based on Gantt-diagram

Appendix F contains a Gantt-diagram created at the start of the project, and Appendix G shows how the Gantt-diagram was followed during the project. The versions vary, showing that the planned activities took a lot longer than anticipated. Most of the activities were, however, almost complete, long before the completion date. This is because some activities were put on hold while waiting for other parts to be completed.

Because of this waiting, the group worked with the activities more in parallel than sequential. A disadvantage of executing the activities in parallel is that the first results are created later in the project. The group tackled this well and was given a better understanding of the project earlier in the project.

6.8.3 Progress based on Analysis of Risk

Appendix K reviews different scenarios facing a thread to the progress for the project. Scenario 1, "shutdown duo to COVID 19", has affected the project the most since Ålesund was shut down in three weeks in March/April. This was because of a high number of infected during this period. The group was skeptical to meeting during this period, and some, therefore, worked from home.

The different workspaces made it harder to collaborate and test the solutions, making the work less effective than for the rest of the project. This scenario was, however, planned for, and the

experienced difficulty was handled efficiently and well.

6.8.4 Learning Outcome

This project provided the group members with new and valuable knowledge about testing and production and how to use our knowledge, acquired from the bachelor degree, in a project. A lot of the knowledge that was necessary to complete each task was acquired underway. While being a complex and large assignment, the team felt that breaking the problems into pieces, assigning dedicated personnel to each task, and being open to help each other was a success.

Chapter 7

Conclusions

To conclude the project, the task was given by Kongsberg Maritime for creating digitized buttons and an automated testing system used for testing Mcon systems before being installed on vessels. The result of this task would make the testing of Kongsberg Maritimes systems faster and easier. Because of the size of the group, we agreed to expand this task by adding a simulation of the thruster system to make the testing easier to understand by seeing how the thrusters operate.

The system consists of levers and screens used for simulating the actual controls on a vessel. By controlling these systems, the corresponding values for the Mcon system should change.

These values are saved in digitized buttons created in B&R Automation Studio, as requested by Kongsberg Maritime. The digitized solution will replace the existing system, consisting of actual buttons and much wiring. The digitized buttons will be uploaded into a CPU at the Mcon system and will only need to be connected by an Ethernet cable, making the testing faster.

The simulation of the thrusters is made in MATLAB®/Simulink® and shows a visualization of the thrusters based on the user input from the Mcon system. This solution is uploaded into B&R Automation Studio, and the simulation is based on the value of variables collected, using the communication protocol "Automation Studio Target for Simulink". This solution will make the testing easier to understand and follow for the testers, reducing the possibility of making critical

errors during testing.

The Mcon systems have to go through the *Factory Acceptance Test* to be approved for delivering. This testing is semi-automated in Java, using inputs from the Mcon system stored at B&R automation Studio. The values are made accessible at Java by using an OPC UA communication protocol, with B&R Automation Studio as a server and Java as a client. When performing the tests at the Mcon system, the semi-automatic test will automatically check if the requirements to pass a test are fulfilled. When all the tests have been complete, a PDF file containing the result of the tests is printed. This makes the testing more accurate since the chance of human error is reduced, and the testing is performed more effortless and faster.

Every main part of the project is complete, and all the different parts are communicating with each other as desired. Every group member has contributed the best they can, and the collaboration has been good. The complete project is therefore considered a success.

Bibliography

[1] *Client-server model (client-server architecture).* Techtarget. URL: https://searchnetworking.techtarget.com/definition/client-server.

[2] *client/server protocol.* PCmag. URL: https://www.pcmag.com/encyclopedia/term/clientserver-protocol.

[3] *Hva er en server.* Datamaskin. URL: http://www.datamaskin.biz/Hardware/servers/62368.html.

[4] *Java - serialization.* TutorialsPoint. URL: https://www.tutorialspoint.com/java/java_serialization.htm.

[5] *Serializable Objects.* Oracle. URL: https://docs.oracle.com/javase/tutorial/jndi/objects/serial.html.

[6] *Web TCP/IP.* w3schools. URL: http://www-db.deis.unibo.it/courses/TW/DOCS/w3schools/website/web_tcpip.asp.html.

[7] 6ft cat 5e rj45 lan cable - grey, 2021.

[8] PLC Academy. Structured text tutorial to expand your plc programming skills. URL: https://www.plcacademy.com/structured-text-tutorial/.

[9] Jalai Ali. *Java - Sockets and Serialization.* Code Project, 2015. URL: https://www.codeproject.com/Tips/991180/Java-Sockets-and-Serialization.

[10] Maven Apache. Introduction. URL: https://maven.apache.org/what-is-maven.html.

[11] Maven Apache. Welcome to apache maven. URL: https://maven.apache.org/.

[12] Atlassian. What is version control? URL: https://www.atlassian.com/git/tutorials/what-is-version-control.

[13] B&R Automation. Tm140 b&r automation studio target for simulink, 2020.

[14] B&R Automation. X20cp3583, 2020.

[15] B&R Automation. Data sheet x20ai4322, April 2019.

[16] B&R Automation. Data sheet x20(c)ao4622, April 2020.

[17] B&R Automation. Datasheet x20(c)di6371, March 2020.

[18] B&R Automation. Data sheet x20(c)do6639, September 2019.

[19] Real Time Automation. Control iec 61131-3. URL: https://www.rtautomation.com/technologies/control-iec-61131-3/.

[20] Unified Automation. Uaexpert—a full-featured opc ua client. URL: https://www.unified-automation.com/products/development-tools/uaexpert.html.

[21] Mathworks B. messner, D . Tilbury. introduction: Pid controller design. URL: https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID.

[22] Mathworks B. messner, D . Tilbury. introduction: Pid controller design. URL: https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID.

[23] The Github Blog. Github desktop is now available. URL: https://github.blog/2015-08-12-github-desktop-is-now-available/.

[24] Robin T. Bye. *Lecture 1 - Basic control system concepts Transfer functions.* NTNU, 2020.

[25] Robin T. Bye. *Lecture 1 - Basic control system concepts Transfer functions.* NTNU, 2020.

[26] Robin T. Bye. *Lecture 3 - Modelling in the frequency domain Nonlinearities and linearization.* NTNU, 2020.

[27] CareLabz. What is factory acceptance testing, and how is fat done. URL: https://carelabz.com/what-factory-acceptance-testing-how-fat-done/.

[28] CIMSS. What is matlab? URL: https://cimss.ssec.wisc.edu/wxwise/class/aos340/spr00/whatismatlab.htm.

[29] Electrical Classrooom. Difference between no and nc, 2021. URL: https://www.electricalclassroom.com/difference-between-no-and-nc/.

[30] Electrical Classrooom. Difference between no and nc, 2021. URL: https://www.herga.com/pressrelease/detail.php?aid=164&did=What-is-a-Normally-Open-Switch?

[31] Happy Coding. Libraries. URL: https://happycoding.io/tutorials/java/libraries.

[32] CVEL. Brushed dc motors. URL: https://cecas.clemson.edu/cvel/auto/actuators/motors-dc-brushed.html.

[33] Github Desktop. Github desktop. URL: https://desktop.github.com.

[34] Diagrams. The offical blog of the diagrams.net project. URL: https://www.diagrams.net/blog.

[35] Erin Doherty. What is object oriented programming? oop explained in depth. URL: https://www.educative.io/blog/object-oriented-programming.

[36] Draw.io. The easiest way for confluence teams to collaborate using diagrams. URL: https://drawio-app.com.

[37] Eclipse. Eclipse milo. URL: https://projects.eclipse.org/projects/iot.milo/governance.

[38] Estudie. Tcp/ip. URL: https://estudie.no/tcp-ip/.

[39] Thor I. Fossen. *Guidance and Control of Ocean Vehicles*. Wiley, 1994.

[40] Thor I. Fossen. Marine systems simulator (mss), 2021.

[41] OPC Foundation. Unified architecture. URL: https://opcfoundation.org/about/opc-technologies/opc-ua/.

[42] Git. Getting started - about version control. URL: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control.

[43] Git. Git –local-branching-on-the-cheap. URL: https://git-scm.com.

[44] Github. Diversity, inclusion, and belonging at github. URL: https://github.com/about/diversity/report.

[45] "DNV GL".

[46] Bachelor Group. Image taken by group, 2021.

[47] Github Guides. Hello world. URL: https://guides.github.com/activities/hello-world/.

[48] Nam ha Minh. *Why Do We Need Serialization in Java?* Code Java, 2019. URL: https://www.codejava.net/java-se/file-io/why-do-we-need-serialization-in-java.

[49] Dag Håkon Hanssen. *Programmerbare Logiske Styringer*. Fagbokforlaget, 4. edition, 2015, page 19-21.

[50] B&R Help Explorer Automation Help. Standard, 2021.

[51] B&R Help Explorer Automation Help. Standard - ton(), 2021.

[52] Hilite.me. Hilite.me. URL: http://hilite.me.

[53] IMO. Introduction to imo. URL: https://www.imo.org/en/About/Pages/Default.aspx.

[54] JavaTPoint. Interface in java. URL: https://www.javatpoint.com/interface-in-java.

[55] Jetbrains. Why intellij idea. URL: https://www.jetbrains.com/idea/.

[56] Naresh Joshi. *Everything You Need to Know About Java Serialization Explained*. DZone, 2019. URL: https://dzone.com/articles/what-is-serialization-everything-about-java-serial.

[57] Soteris Kalogirou. *Renewable Energy*. ScienceDirect, 2021. URL: https://www.sciencedirect.com/science/article/pii/S0960148117312454#fig1.

[58] kevinherron. Eclipse milo. URL: https://github.com/eclipse/milo.

[59] KONGSBERG. Controllable pitch propeller. URL: https://www.kongsberg.com/maritime/products/propulsors-and-propulsion-systems/propellers/controllable-pitch-propeller/.

[60] KONGSBERG. Fixed pitch propeller. URL: https://www.kongsberg.com/no/maritime/products/propulsors-and-propulsion-systems/propellers/fixed-pitch-propeller/.

[61] Kongsberg. Tunnel thrusters. URL: https://www.kongsberg.com/no/maritime/products/propulsors-and-propulsion-systems/thrusters/tunnel-thrusters/.

[62] Lehmi. Apache pdfbox. URL: https://github.com/apache/pdfbox.

[63] Ivar M. Liseter. *Server*. Store norske leksikon, 2020. URL: https://snl.no/server.

[64] Kongsberg Maritime. *Factory Acceptance Test (Appendix E)*. Kongsberg Maritime.

[65] Kongsberg Maritime. Mcon –one maneuvering control.

[66] DXP Marketing. Factory acceptance testing – what is fat, and how does it work? URL: https://www.dxpe.com/what-is-factory-acceptance-test-protocol-purpose/.

[67] MathWorks. Simulink coder, 2021.

[68] MATLAB. Math. graphics. programming. URL: https://se.mathworks.com/products/matlab.html.

[69] MATLAB. Simulink. URL: https://se.mathworks.com/help/simulink/.

[70] Jennifer Mayo. Factory acceptance tests: What they are and why they're important. URL: https://www.ddpsinc.com/blog-0/factory-acceptance-tests-what-they-are-and-why-theyre-important.

[71] P. Kinley M.M. Bernitsas, D. Ray. Kt, kq and effeniency curves for the wageningen b-series propellers. URL: http://kashti.ir/files/ENBOOKS/B-series%20propeller.pdf.

[72] Norman S. Nise. *Control Systems Engineering*. Wiley, 2008.

[73] Novotek. Opc og opc ua. URL: https://www.novotek.com/no/l-sninger/kepware-opc-kommunikasjonsplattform/opc-og-opc-ua/.

[74] Birkbech University of London. 10. transport layer. URL: https://www.dcs.bbk.ac.uk/~ptw/teaching/IWT/transport-layer/notes.html.

[75] Oracle. Best practices for designing and implementing a library in java. URL: https://www.oracle.com/corporate/features/library-in-java-best-practices.html.

[76] Overleaf. About us. URL: https://www.overleaf.com/about#who-we-are.

[77] Overleaf. Learn latex in 30 minutes. URL: https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes.

[78] PDFBox. Apache pdfbox - a java pdf library. URL: https://pdfbox.apache.org/.

[79] PDFBox. Download. URL: https://pdfbox.apache.org/download.html.

[80] The Latex Project. Latex – a document preparation system. URL: https://www.latex-project.org.

[81] Rajani K RamaKoteswara Rao. A, Lekyasri N. Pid control designs for second order systems. URL: http://www.mecs-press.org/ijem/ijem-v9-n4/IJEM-V9-N4-4.pdf.

[82] REALVNC. Secure remore access and support, 2021.

[83] rfscholte. Apache maven. URL: https://github.com/apache/maven.

[84] James F. Kurose & Keith W. Ross. *Computer Networking*. Pearson, 6. edition, 2012, page 112.

[85] Eirik Rossen. *klient/tjener-teknologi - IT*. Store norske leksikon, 2019. URL: https://snl.no/klient/tjener-teknologi_-_IT.

[86] OPC Router. What is opc ua? a practical introduction. URL: https://www.opc-router.com/what-is-opc-ua/#OPC-Server.

[87] Sunarsih S. A chebyshev polynomial on torque and thrust coefficients of mathematical propeller properties for a lng manoeuvring simulation. URL: https://www.researchgate.net/publication/277708818_A_Chebyshev_Polynomial_on_Torque_and_Thrust_Coefficients_of_Mathematical_Propeller_Properties_for_a_LNG_Manoeuvring_Simulation.

[88] Kjetil Sander. *Nettverksserver (Server, også kalt tjener)*. eStudie, 2019. URL: https://estudie.no/nettverksserver/.

[89] Computer Science. Computer programming languages. URL: https://www.computerscience.org/resources/computer-programming-languages/.

[90] Towards Data Science. Top 10 libraries every java developer should know. URL: https://towardsdatascience.com/top-10-libraries-every-java-developer-should-know-37dd136dff54.

[91] ScienceDirect. Communication protocol. URL: https://www.sciencedirect.com/topics/engineering/communication-protocol.

[92] ScienceDirect. Structured text. URL: https://www.sciencedirect.com/topics/computer-science/structured-text.

[93] The Server Side. Java. URL: https://www.theserverside.com/definition/Java.

[94] Techopedia. Communication protocol. URL: https://www.techopedia.com/definition/25705/communication-protocol.

[95] TechTarget. client-server model (client-server architecture). URL: https://searchnetworking.techtarget.com/definition/client-server.

[96] Catalin Tudose. *Building Java Client/Server Applications with TCP*. Luxoft-training. URL: https://www.luxoft-training.com/news/building-java-client-server-applications-with-tcp/.

[97] Tutorialspoint. Java - interfaces. URL: https://www.tutorialspoint.com/java/java_interfaces.htm.

[98] Tutorialspoint. Pdfbox - environment. URL: https://www.tutorialspoint.com/pdfbox/pdfbox_environment.htm.

[99] W3Schools. Java interface. URL: https://www.w3schools.com/java/java_interface.asp.

[100] w3schools. Java introduction. URL: https://www.w3schools.com/java/java_intro.asp.

[101] Anish Wankhede. Propeller, types of propellers and construction of propellers, 2021.

[102] Wikipedia. C (programming language). URL: https://en.wikipedia.org/wiki/C_(programming_language).

[103] Wikipedia. Dc motor. URL: https://en.wikipedia.org/wiki/DC_motor.

[104] Wikipedia. Variable-pitch propeller. URL: https://en.wikipedia.org/wiki/Variable-pitch_propeller.

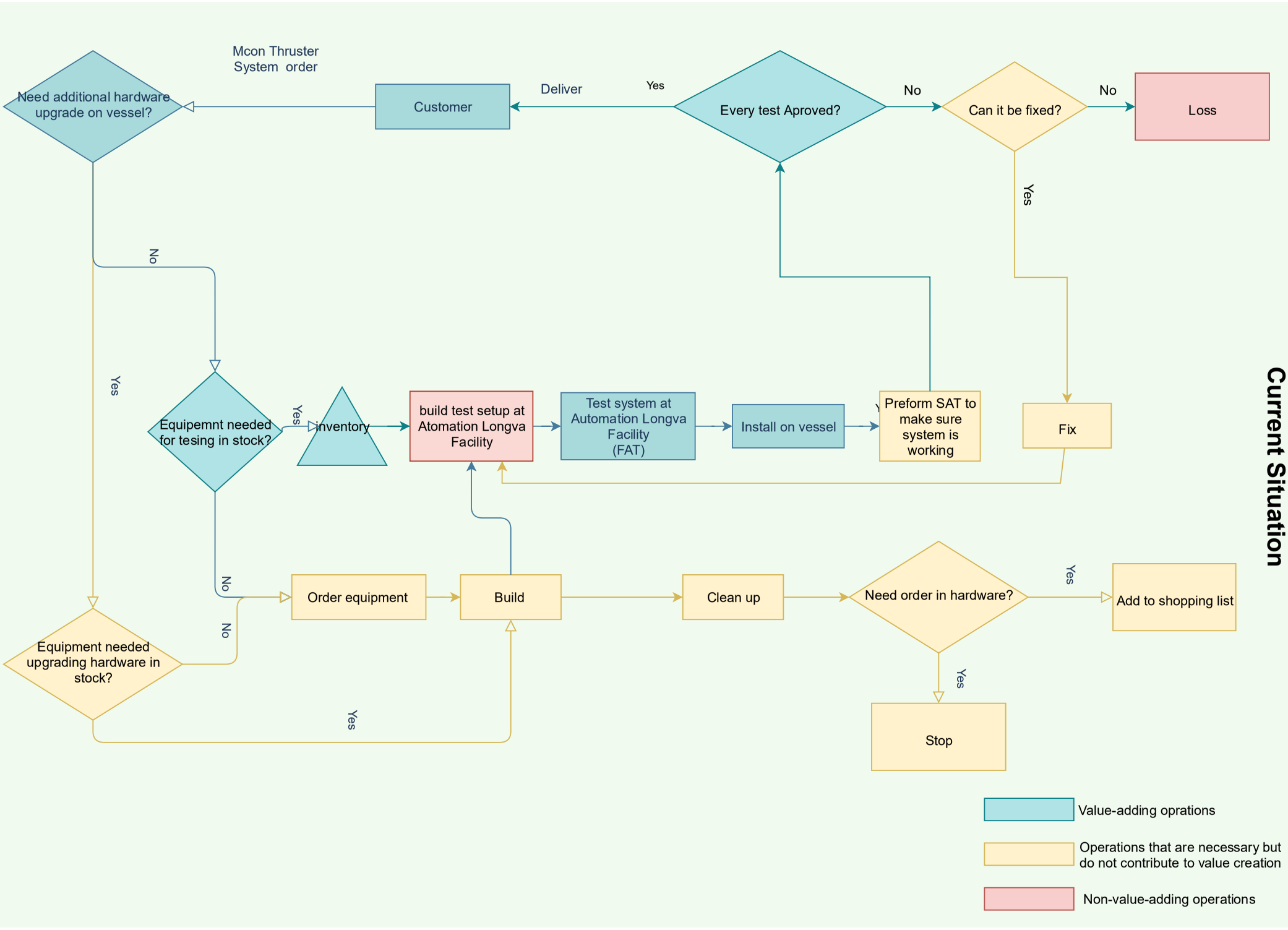
[105] Wikipedia. Modeling of basic propeller thrust test system and thrust control using pid method, 2016.

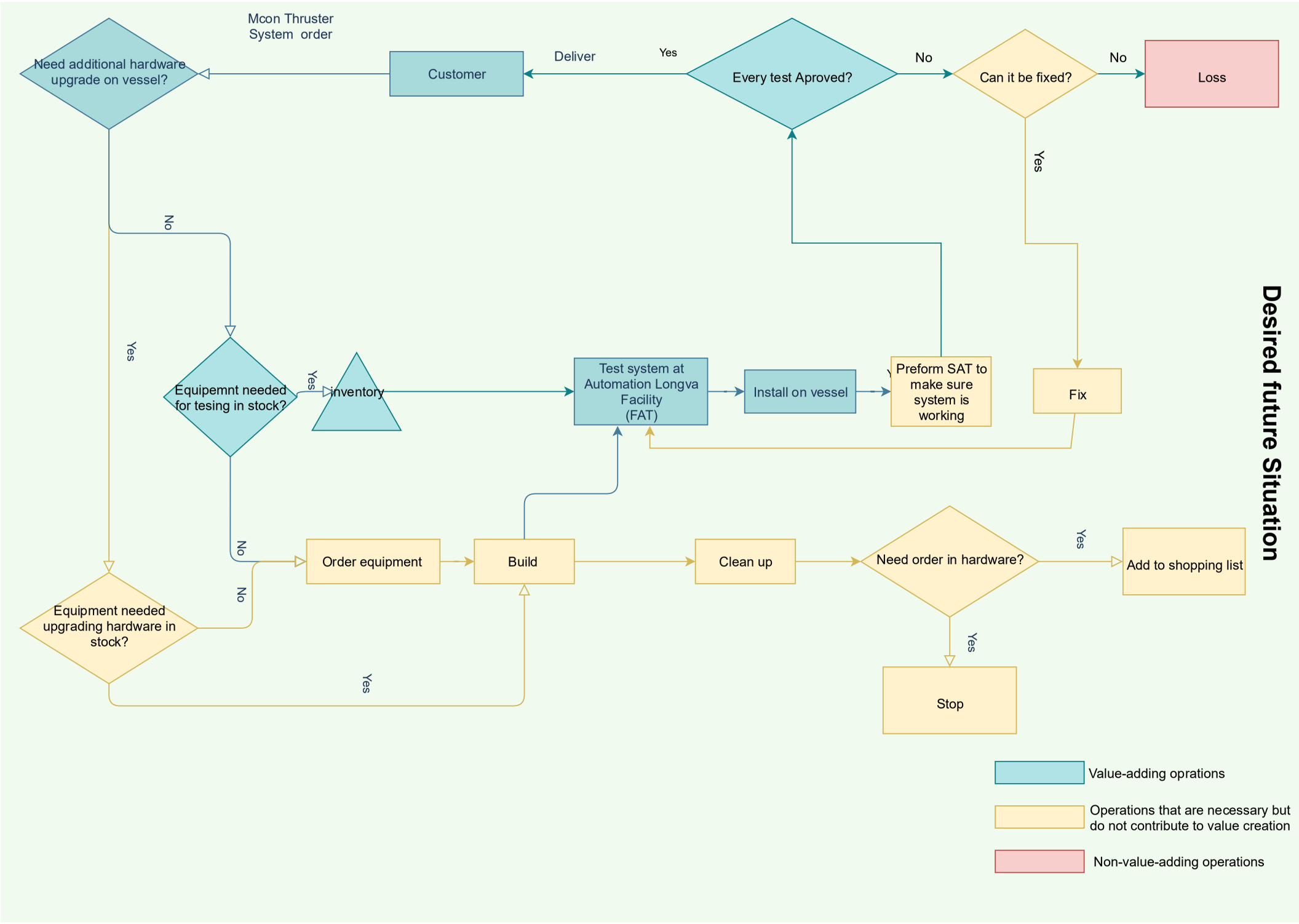
[106] Wikipedia. Ethernet/ip, 2021.

[107] Wikipedia. Multimeter, 2021.

Appendixes

A Lean Manufacturing Diagram


Mcon Thruster System order
Need additional hardware upgrade on vessel?
Customer
Deliver
Yes
Every test Aproved?
No
Can it be fixed?
No
Loss
Yes
No
Yes
Equipemnt needed for tesing in stock?
Yes
inventory
build test setup at Atomation Longva Facility
Test system at Automation Longva Facility (FAT)
Install on vessel
Preform SAT to make sure system is working
Fix
No
Order equipment
Build
Clean up
Need order in hardware?
Yes
Add to shopping list
No
Equipment needed upgrading hardware in stock?
Yes
Yes
Stop
Current Situation
Value-adding oprations
Operations that are necessary but do not contribute to value creation
Non-value-adding operations


Mcon Thruster System order
Need additional hardware upgrade on vessel?
Customer
Deliver
Yes
Every test Aproved?
No
Can it be fixed?
No
Loss
Yes
No
Yes
Equipemnt needed for tesing in stock?
Yes
inventory
Test system at Automation Longva Facility (FAT)
Install on vessel
Preform SAT to make sure system is working
Fix
No
Order equipment
Build
Clean up
Need order in hardware?
Yes
Add to shopping list
Equipment needed upgrading hardware in stock?
No
Yes
Yes
Stop
Desired future Situation
Value-adding oprations
Operations that are necessary but do not contribute to value creation
Non-value-adding operations

B Wageningen B-series

TABLE 1

Coefficients and terms of the K_T and K_Q polynomials for the Wageningen B-screw Series for $R_n=2\times10^6$. Reproduced from [1]

$$K_T = \sum_{s,t,u,v} C^T_{s,t,u,v} \cdot (J)^s \cdot (P/D)^t \cdot (A_E/A_O)^u \cdot (z^v)$$

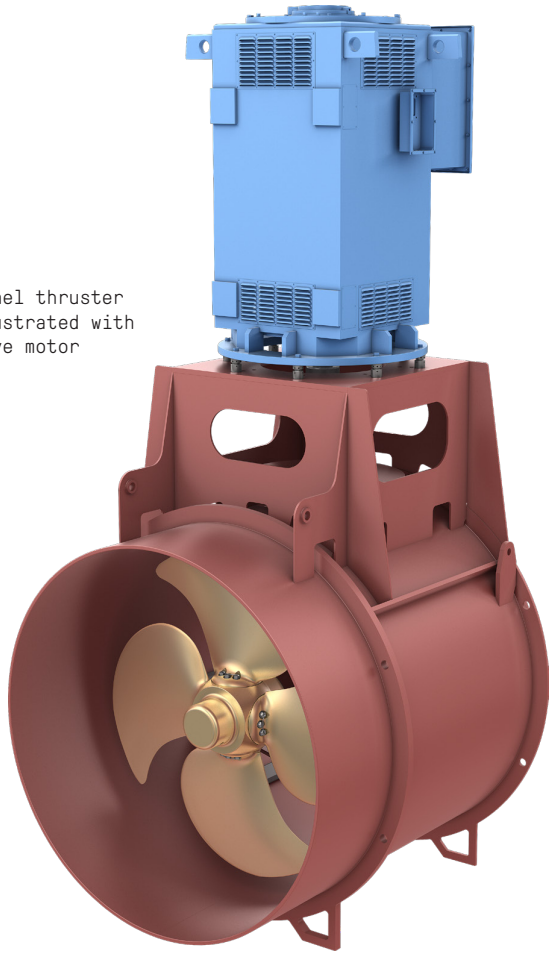
$$K_Q = \sum_{s,t,u,v} C^Q_{s,t,u,v} \cdot (J)^s \cdot (P/D)^t \cdot (A_E/A_O)^u \cdot (z^v)$$

K_T $C^T_{s,t,u,v}$	s (J)	t (P/D)	u (A_E/A_O)	v (Z)	$C^Q_{s,t,u,v}$	s (J)	t (P/D)	u (A_E/A_O)	v (Z)
+0.00880496	0	0	0	0	+0.00379368	0	0	0	0
-0.204554	1	0	0	0	+0.00886523	2	0	0	0
+0.166351	0	1	0	0	-0.032241	1	1	0	0
+0.158114	0	2	0	0	+0.00344778	0	2	0	0
-0.147581	2	0	1	0	-0.0408811	0	1	1	0
-0.481497	1	1	1	0	-0.108009	1	1	1	0
+0.415437	0	2	1	0	-0.0885381	2	1	1	0
+0.0144043	0	0	0	1	+0.188561	0	2	1	0
-0.0530054	2	0	0	1	-0.00370871	1	0	0	1
+0.0143481	0	1	0	1	+0.00513696	0	1	0	1
+0.0606826	1	1	0	1	+0.0209449	1	1	0	1
-0.0125894	0	0	1	1	+0.00474319	2	1	0	1
+0.0109689	1	0	1	1	-0.00723408	2	0	1	1
-0.133698	0	3	0	0	+0.00438388	1	1	1	1
+0.00638407	0	6	0	0	-0.0269403	0	2	1	1
-0.00132718	2	6	0	0	+0.0558082	3	0	1	0
+0.168496	3	0	1	0	+0.0161886	0	3	1	0
-0.0507214	0	0	2	0	+0.00318086	1	3	1	0
+0.0854559	2	0	2	0	+0.015896	0	0	2	0
-0.0504475	3	0	2	0	+0.0471729	1	0	2	0
+0.010465	1	6	2	0	+0.0196283	3	0	2	0
-0.00648272	2	6	2	0	-0.0502782	0	1	2	0
-0.00841728	0	3	0	1	-0.030055	3	1	2	0
+0.0168424	1	3	0	1	+0.0417122	2	2	2	0
-0.00102296	3	3	0	1	-0.0397722	0	3	2	0
-0.0317791	0	3	1	1	-0.00350024	0	6	2	0
+0.018604	1	0	2	1	-0.0106854	3	0	0	1
-0.00410798	0	2	2	1	+0.00110903	3	3	0	1
-0.000606848	0	0	0	2	-0.000313912	0	6	0	1
-0.0049819	1	0	0	2	+0.0035985	3	0	1	1
+0.0025983	2	0	0	2	-0.00142121	0	6	1	1
-0.000560528	3	0	0	2	-0.00383637	1	0	2	1
-0.00163652	1	2	0	2	+0.0126803	0	2	2	1
-0.000328787	1	6	0	2	-0.00318278	2	3	2	1
+0.000116502	2	6	0	2	+0.00334268	0	6	2	1
+0.000690904	0	0	1	2	-0.00183491	1	1	0	2
+0.00421749	0	3	1	2	+0.000112451	3	2	0	2
+0.0000565229	3	6	1	2	-0.0000297228	3	6	0	2
-0.00146564	0	3	2	2	+0.000269551	1	0	1	2
					+0.00083265	2	0	1	2
					+0.00155334	0	2	1	2
					+0.000302683	0	6	1	2
					-0.0001843	0	0	2	2
$R_n = 2 \times 10^6$					-0.000425399	0	3	2	2
					+0.0000869243	3	3	2	2
					-0.0004659	0	6	2	2
					+0.0000554194	1	6	2	2

C Data sheet thruster

TUNNEL THRUSTER
TYPE TTC CP

KONGSBERG



Tunnel thruster illustrated with drive motor

FUNCTIONAL FEATURES

- Propeller hub design based upon well proven TT-standard propeller hub.
- Hydraulically balanced propeller hub design.
- Propeller blade bolted from outside for easy blade seal change.
- Well proven blade seal solution.
- No need to remove intermediate shaft and drive motor for overhauling of thruster. All drive modules can be changed with access from tunnel side.
- Thruster available in “Thruster Support Pool” (TSP).
- Worldwide service network.
- Approved for use of EAL-oil.

KONGSBERG MARITIME TUNNEL THRUSTERS

Type TTC CP

Key facts

The tunnel thruster is designed for giving max. sideforce to the ship in manoeuvering condition.

Application
Auxiliary use – limited hours/year lifetime

The system normally contains of thruster unit with tunnel, hydraulic equipment, remote control and electrical drivemotor with starter.

- Skew blades.
- Easy installation
- Less adjustment during installation (compared to standard TT-thruster).
- Interchange with TT-thruster
- Increased clearance between propeller blade tip and tunnel plating.
- Reduced noise and vibration.
- TT-tunnel can be delivered by yard according to our drawing
- Interchange with TT-thruster
- Simplified hydraulic.
- Less external piping for the yard during installation.
- Less complexity during commissioning and maintenance.

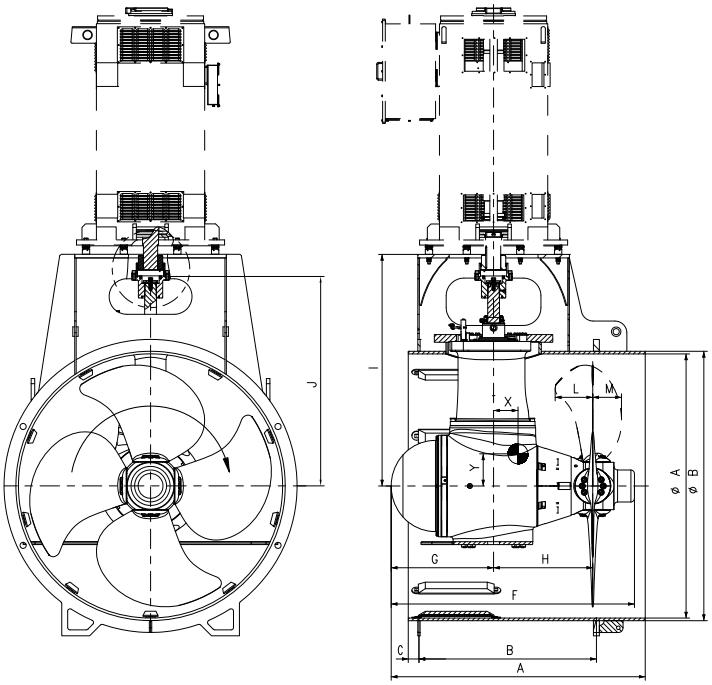
AVAILABLE GEAR-RATIO ON TTC

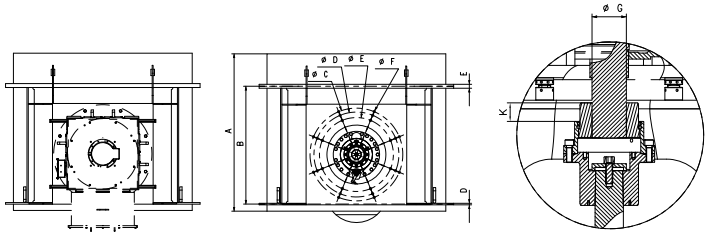
Design data

	TIP SPEED	MOTOR	PROPELLER	POWER	PRIME MOVER		TUNNEL DIA		THRUSTER/ SIDE FORCE
TTC-SIZE	m/s	RPM	RPM	(KW)	TYPE	HZ	NOMINAL	MAX	KN(*)
TTC 65 CP	34.6	1200	318	1500	El. motor	60	2200	2230	196
TTC 65 CP	35.2	1500	324	1500	El. motor	50	2200	2230	196
TTC 80 CP	35.0	1200	295	1900	El. motor	60	2400	2430	243
TTC 80 CP	32.2	1000	271	1900	El. motor	50	2400	2430	243
TTC 83 CP	31.9	900	244	2160	El. motor	60	2650	2680	285
TTC 83 CP	32.2	1000	246	2160	El. motor	50	2650	2680	285
TTC 85 CP	33.8	900	244	2500	El. motor	60	2800	2836	324
TTC 85 CP	34.1	1000	246	2500	El. motor	50	2800	2836	324

NOTES

*1) A steady force generated without operation of a nearby TT and ideally integrated with the hull with optimum inlet geometry, no grid and without any degradation effects from current and waves (ventilation).





TTC RANGE

VARIANTS	TTC 65	TTC 80	TTC 83	TTC 85
Power (kW)	1500	1900	2200	2500
øA	2230	2453	2680	2896
øB	2274	2480	2730	2836
A	2145	2482	2492	2631
B	1498	1665.5	1811	1879
C	90	75	75	105
D	20	20	20	20
E	50	50	50	50
F	1702	2620	2620	2620
G	865	1082	1082	1082
H	837	1075	1075	1075
I	1955	2090	2283	2425
J	1708	1888	2032	2110
K	65	94	94	94
L	317	341	375	398
M	242	246	276	293
X	206	232	0	265
Y	270	341	0	382

Weight of standard tunnel and propeller unit, excl. oil.

NOTES

All data is subject to change without prior notice.

Kongsberg Maritime
P.O.Box 483, NO-3601
Kongsberg, Norway

Switchboard: +47 815 73 700
Global support 24/7: +47 33 03 24 07
E-mail sales: km.sales@km.kongsberg.com
E-mail support: km.support@kongsberg.com

D FPP test data Mcon simulation

seconds	RPM_order	RPM_feedback
MIN-MAX		
1	-104.25428	-104.199448
2	-41.1175842	-96.23384
3	48.3871	-58.17028
4	94.87898	36.7642136
5	94.87898	57.73763
6	94.87898	68.03717
7	94.87898	75.43226
8	94.87898	80.80474
9	94.87898	84.3414841
10	94.87898	87.6693039
11	94.87898	89.76986
12	94.87898	91.14681
13	94.87898	92.35294
14	94.87898	93.14081
15	94.87898	93.7021942
16	94.87898	94.15052
17	94.87898	94.3391342
18	94.87898	94.55014
19	94.87898	94.6795654
20	94.87898	94.74855
21	94.87898	94.80109
22	94.87898	94.84115
23	94.87898	94.8718262
24	94.87898	94.87896
25	94.87898	94.8844147
26	94.87898	94.89125
27	94.87898	94.89516
MAX-MIN		
1	94.87898	94.88012
2	94.94004	94.8838
3	94.87898	94.8848648
4	94.87898	94.88414
5	94.81795	94.88292
6	4.01318073	74.74541
7	-85.43046	5.05802965
8	-104.181038	-57.9937363
9	-104.119995	-70.30955
10	-104.119995	-78.51479
11	-104.181038	-85.04067
12	-104.25428	-89.85645
13	-104.18403	-93.3032
14	-104.181038	-96.39902
15	-104.181038	-98.10898
16	-104.25995	-98.7083359
17	-104.181028	-101.1148
18	-104.119995	-101.962875
19	-104.25428	-102.654961

20	-104.119995	-103.096169
21	-104.119995	-103.372665
22	-104.25428	-103.612305
23	-104.119995	-103.784378
24	-104.181038	-103.920723
25	-104.25428	-103.998282
26	-104.181038	-104.066238
27	-104.25428	-104.106552
28	-104.25428	-104.141693
29	-104.181038	-104.159073
MIN-IDLE		
1	-104.119995	-104.196053
2	-104.181038	-104.196831
3	-94.09162	-103.94136
4	-4.776144	-74.42611
5	-4.89822626	-49.2403564
6	-4.89822626	-28.0964851
7	-4.776144	-17.9538231
8	-4.837179	-10.3745394
9	-4.89822626	-6.818443
10	-4.837179	-4.972011
11	-4.776144	-3.966474
12	-4.837179	-3.75827479
13	-4.89822626	-3.92338777
14	-4.837179	-4.08486557
15	-4.776144	-4.277877
16	-4.776144	-4.46177
17	-4.89822626	-4.67163563
18	-4.776144	-4.68631554
19	-4.776144	-4.71429729
20	-4.776144	-4.791469
IDLE-MAX		
1	-4.776144	-4.80007124
2	-4.776144	-4.799315
3	54.05743	30.838974
4	94.87898	61.43797
5	94.87898	71.95902
6	94.94004	78.53007
7	94.94004	82.583725
8	94.87898	85.6563
9	94.87898	87.87058
10	94.87898	89.72004
11	94.87898	91.1753159
12	94.94004	91.13227
13	94.87898	92.9044952
14	94.87898	93.50289
15	94.87898	93.88788
16	94.94004	94.20287
17	94.87898	94.440094
18	94.94004	94.5676346

19	94.87898	94.66889
20	94.87898	94.7234
21	94.87898	94.7809753
22	94.87898	94.87898
23	94.94004	94.84756
24	94.87898	94.86211
25	94.87898	94.87121
MAX-IDLE		
1	94.87898	94.88321
2	93.70098	94.87342
3	26.7555714	85.1609
4	-4.5259	49.34366
5	-4.5259	26.73314
6	-4.5259	8.716376
7	-4.59303856	-0.3394618
8	-4.59303856	-3.359396
9	-4.59303856	-4.769736
10	-4.5259	-5.453189
11	-4.59303856	-5.5985455
12	-4.59303856	-5.419911
13	-4.464853	-5.16858
14	-4.59303856	-5.08442354
15	-4.59303856	-4.870197
16	-4.5259	-4.755985
17	-4.5259	-4.63035
18	-4.5259	-4.50429
19	-4.5259	-4.53675365
20	-4.5259	-4.509827
21	-4.59303856	-4.568879
22	-4.464853	-4.47395849
23	-4.5259	-4.561048
24	-4.464853	-4.53628826
IDLE_MIN		
1	-4.59303856	-4.51286173
2	-4.59303856	-4.53603
3	-4.59303856	-4.56467772
4	-4.464853	-4.53552961
5	-32.2000732	-14.4651
6	-104.25428	-57.9530373
7	-104.25428	-74.0755
8	-104.181038	-82.87079
9	-104.181038	-87.54089
10	-104.181038	-90.52071
11	-104.119095	-94.15338
12	-104.181038	-96.9795456
13	-104.181038	-98.70324
14	-104.19995	-100.098091
15	-104.25428	-101.15802
16	-104.181038	-101.892616
17	-104.25438	-102.551849

18	-104.25428	-103.003227
19	-104.119995	-103.299286
20	-104.181038	-103.513222
21	-104.181038	-103.674454
22	-104.25428	-103.86739
23	-104.181038	-103.964462
24	-104.181038	-104.031517
25	-104.181038	-104.083328
26	-104.181038	-104.113762
27	-104.181038	-104.145378
28	-104.181038	-104.165985
29	-104.25428	-104.171822
IDLE-50		
1	-5	-4.160493
2	-4.98622626	-4.351392
3	-4.715109	-4.53316975
4	-4.837179	-4.6801815
5	-4.776144	-4.706339
6	-4.69303856	-4.63601637
7	44.39528	30.48127
8	44.3342323	37.932372
9	44.3342323	40.5216675
10	44.2731972	41.89232
11	44.3342323	42.7657623
12	44.2371972	43.40334
13	44.3342323	43.662056
14	44.3342323	44.0636
15	44.3342323	44.2395668
16	44.2731972	44.33226
17	44.3342323	44.3682365
18	44.3342323	44.38282
19	44.2731972	44.37939
20	44.2731972	44.37564
Idle_25		
1	-4.15357351	-4.379888
2	-4.15357351	-4.41226673
3	-4.220712	-4.36220263
4	-4.28174734	-4.3375597
5	-4.220712	-4.30992365
6	19.1564674	15.1981106
7	19.1564674	17.83905
8	19.0954323	18.4301662
9	19.0282936	18.6812534
10	19.0954323	18.9502
11	19.0954323	19.078783
12	18.9672585	19.1089325
13	19.0954323	19.1166762
14	19.0954323	19.1148376
15	19.0282936	19.1009426

E Factory Acceptance Test

PUBLIC

Factory Acceptance Test

Doc No:	EAC-117765-01TF
Revision:	B
Project:	20S000408 ELEKTRON
Status:	This document is under configuration control at Kongsberg Maritime.

Document history

Revision	*Description of Change*
A	First Issue
B (10.11.2020)	3 – Faults and consequences: Changed test for +CU01-F001 and F101

Disclaimer

Kongsberg Maritime AS endeavours to ensure that all information in this document is correct and fairly stated, but does not accept liability for any errors or omissions.

Table of contents

PROJECT

Project/ installation no.	20S000408
Vessel	ELEKTRON
System	Mcon

1 Foreword

This document describes two tests: A “functional” and a “consequences of a fault” test for the Mcon remote control system.

This test procedure will be performed after the production of the system is completed and initially tested. The test report covers the requirements stated in the Det Norske Veritas Rules “Instrumentation and Automation, Computer based Systems”.

1.1 General

Prior to the two tests all units and components have been checked for proper workmanship and that correct procedures have been used during the manufacturing operations.

All units supplied by Kongsberg Maritime CM AS Automation & Control are connected and tested according to the cable diagram, and during the tests external equipment are simulated.

2 Functional tests.

2.1 Tunnel thrusters

2.1.1 General functions

Functional test:	Verification:	Checked:
2.1.1.1 Power-up test With the power turned off, engage: +CU01-F001, +CU01-F101	The control systems should boot and come up with command request.	Checked: ☐ Comment: ☐
2.1.1.2 Dimmer Turn the dimming up and down on each control panel	The illumination in all panels, levers and indicators at the control station should be illuminated according to the dimmer position.	Checked: ☐ Comment: ☐
2.1.1.3 Command transfer. Activate backup control for all other systems than the one on test. Activate the “IN COMMAND” button on a bridge control panel which is not “IN command”	The command should be transferred to the selected panel without any restrictions	Checked: ☐ Comment: ☐
2.1.1.4 Motorized Lever On the station in command, move the pitch/rpm –lever to various positions.	The levers on all the other command stations should move to the same position as the one in command.	Checked: ☐ Comment: ☐

2.1.2 Rpm Control

Functional test:	Verification:	Checked:
2.1.2.1 Rpm Control Move the Pitch/rpm lever in stbd. /port direction.	The rpm order output(s) should fluctuate between the idle and maximum setting according to the lever pos.	Checked: ☐ Comment: ☐
2.1.2.2 Backup Control From the control lever in command, activate the backup control system with the “EMERG. ON” switch, and increase/decrease RPM with the lever.	The RPM feedback shown on the indicator should move smoothly towards the position corresponding to the lever	Checked: ☐ Comment: ☐

2.1.3 Start / stop and reset functions

Functional test:	Verification:	Checked:
2.1.3.1 Remote Start/Stop of thruster motor. Activate the “Start/stop thruster” button in the control panel.	The start/stop thruster motor indication will flash on the corresponding activated button, until the requested run/stop feedback is obtained, then the indicator remains steady ON.	Checked: ☐ Comment: ☐
2.1.3.2 Remote Start/Stop of thruster servo pumps. Activate the “Start/stop servo pump” button in the control panel.	The start/stop thruster servo pump indicator will flash on the corresponding activated button, until the requested servo pump run/stop feedback is obtained, then the indicator remains steady ON.	Checked: ☐ Comment: ☐
2.1.3.3 Reset Drive Activate the “Reset Drive” button in the control panel.	Verify that digital contact is sent to Drive.	Checked: ☐ Comment: ☐

2.1.4 Interfaces to external systems

Functional test:	Verification:	Checked:
2.1.4.1 DP Interface Activate the "DP request signal" and simulate various pitch/rpm order signals from the DP-system into the Mcon system.	The "acknowledge DP" signal should be activated after receiving the "AM request" signal. The Mcon rpm feedback should move corresponding to the given orders from the DP-system.	Checked: ☐ Comment: ☐
2.1.4.2 Joystick Interface Activate the "JS request signal" and simulate various pitch/rpm order signals from the JS-system into the Mcon system.	The "acknowledge JS" signal should be activated after receiving the "AM request" signal. The Mcon rpm feedback should move corresponding to the given orders from the JS-system.	Checked: ☐ Comment: ☐

3 Faults and Consequences.

This section describes which consequences a failure in the power supply circuits, external signals or computer cards will have on the remote control system. One failure at the time is described.

When a failure occurs this is indicated with means of:

- A common alarm contact to the alarm plant.
- Visible and audible failure indication in control panels alarm manager.
- System failure indication in the control levers, panels and automatically change over to backup control

  (note, only when serious faults).

Note: Watchdog detected failure, A/D failure and checksum failure can not be simulated, a failure on these will result in backup control.

The following tables will describe which of the above fault actions/indications that appear at different faults in the system.

3.1 Faults and consequences table.

3.1.1 Tunnel thrusters

Failure	System affected/Consequences:	Failure indication:	Checked:
Broken main fuse: +CU01 –F001 Loss of 230V Main supply	Automatic change-over to 24V DC backup supply. The system will continue to work normally.	Visible and audible failure indication in control panels Warning signal to alarm plant.	Checked: ☐ Comment: ☐
Broken fuses: +CU01-F002 to Controller, Main I/O and fan Loss of 24VDC internal normal supplies Loss of 24VDC supply to cabinet cooling fan	Not possible to operate normal control. Activation of backup buzzer. Fan will stop and temperature inside cabinet will increase.	Visible and audible failure indication in control panels Warning signal to alarm plant.	Checked: ☐ Comment: ☐
Broken fuse: +CU01-F003 to NC levers. Loss of 24VDC NC supply to levers on Bridge	Not possible to operate normal control from levers on bridge. If operating in Normal control on manual levers backup buzzer activates.	Visible and audible failure indication in control panels Warning signal to alarm plant.	Checked: ☐ Comment: ☐

Failure	System affected/Consequences:	Failure indication:	Checked:
Broken main fuse: +CU01-F101 Loss of 24VDC backup supply	The system will continue to work normally, using normal supply.	Visible and audible failure indication in control panels Warning signal to alarm plant.	Checked: ☐ Comment: ☐
Broken fuses: +CU01-F102 to Controller and I/O. Loss of 24VDC internal backup supplies	Not possible to operate backup control. The system will continue working normally.	Visible and audible failure indication in control panels Warning signal to alarm plant.	Checked: ☐ Comment: ☐
Broken fuses: +CU01-F103 to operator devices Loss of 24VDC NC supply to levers and indicators on Bridge	Not possible to operate backup control from levers on bridge. Feedback indicators not working. The system will continue working normally.	Visible and audible failure indication in control panels Warning signal to alarm plant.	Checked: ☐ Comment: ☐
Broken CAN line to lever unit (Normal control)	If this operator position is In Command, Backup buzzer activates.	Visible and audible failure indication in control panels Warning signal to alarm plant.	Checked: ☐ Comment: ☐

Failure	System affected/Consequences:	Failure indication:	Checked:
Broken CAN line to lever unit (Backup control)	Warning signal to alarm plant	Visible and audible failure indication in the control panels and levers. Alarm in the alarm manager and on the control lever in command	Checked: ☐ Comment: ☐
Broken wire in rpm feedback signal from Drive (Normal control)	Rpm Indication not working. Rpm feeback signal to external systems (DP/JS) not working. The system will continue working normally without a visual feedback	Visible and audible failure indication in control panels Warning signal to alarm plant.	Checked: ☐ Comment: ☐
Broken Ethernet switch in ECR (disconnect power to switch)	Communication to thrusters interface units connected to switch in question not working. (only used for commissioning/service purpose) System operate as before fail.	Local indication on switch.	Checked: ☐ Comment: ☐

F Preliminary Report

Preliminary Project
MCON Thruster Simulator

Technical Report
NTNU students
Ålesund, Spring 2021

NTNU

Candidates (Surname, Name):

Skarbø, Emilie Skalstad

Sperre, Linda Helen

Furmyr, Maria

Mellum, Nicklas

DATE:	SUBJECT CODE:	GROUP (name/nr):	Pages	BIBL. NR:
17/05/2021	IE303612	Mcon Thruster Simulator	24	N/A

CLIENT / SUPERVISOR(S):

Kongsberg Maritime AS

Anete Vagale

Ottar L. Osen

Robin T. Bye

TITLE:

MCON Thruster Simulator

SUMMARY:

Kongsberg Maritime wants to develop a simulator that can be used to test and verify Mcon thruster control system during assembly and test at Automation Longva facility. This task is given as a bachelor thesis to students at the department of engineering and science, automation technology. This pre-project report is a project description of the bachelor thesis.

The bachelor thesis will study a concept of a Software-In-the-Loop (SIL) setup. The complete setup consists only of software, and can be viewed as a digital twin component, containing both the physics models of the equipment and the control system running it. Important design criteria will be to simulate the measurements, as the actuators, the dynamics and the sensors correctly.

In addition, a semi-automatic test system, interface thrust-simulator and a visualisation of the thrust-simulator is developed.

Contents

List of Tables

1 Introduction

Our group came together through a common interest in the task Mcon Thruster Simulator, which is about developing a SIL simulator that will be used for test and verification of the Mcon Thruster control system during assembly and test at Automation Longva facility.

Kongsberg Maritime is currently using a testing rig to test thrusters. This testing rig consists of psychically connected components, which makes testing unreliable and time consuming. By automating the testing rig, and making it consist of only software, not hardware, this process can be improved.

The physical equipment (hardware) becomes replaced with a simulator that models the real actuators, dynamics and sensors of the physical system. The control system sends control signals to the simulated equipment and receives simulated measurements. MATLAB will be used to create a realistic model of the thruster. Readings such as pitch and RPM will be measured from the MATLAB/Simulink model created. The simulation should be able to simulate different thruster types based on the different config files that are being read. B & R Automation studio will be used to build and import the components for the thrusters and all other necessary parts to simulate the thruster system. The IO will be simulated through CAN bus, and the IO should be able to be changed or adjusted through a GUI.

The software/simulator should work the same way as the hardware would have, so that the control system does not notice whether it is connected to Hardware or Software. Finally, the operating system hardware must be visualized. In addition a semi-automatic test system, following the Factory Acceptance Test requirements is to be developed. This FAT test is semi-automatic, and will go through spesific checkpoints to test the simulated system.

The task is given by Kongsberg Maritime.

2 Terminology

- Definition of key concepts in the project
NTNU - Norwegian University of Science and Technology
KM - Kongsberg Maritime
B&R - Company for simulating and creating automation and process control solutions
Mcon thruster - Thruster system for a boat
FAT - Factory Acceptence Test
I/O - Input / output
HW - Hardware
SW - Software
SIL - Software In-The-Loop
HIL - Hardware In-The-Loop
CANbus - Controller Area Network bus
MATLAB - Multi-paradigm programming language and numeric computing environment in a unique programming language
Simulink - MATLAB-based graphical programming environment
MapleSim - Advanced system-level modeling tool
LaTeX - Document preparation system
OneDrive - File hosting service and synchronization service
Google Docs - Word processor
FHI - Norwegian Institute of Public Health
Covid-19 - Worldwide ongoing pandemic
FGL - Functioning Group Leader

3 Project organization

3.1 Project group

Name	Student number
Emilie Skalstad Skarbø	509369
Linda Helen Sperre	509365
Maria Furmyr	494663
Nicklas Mellum	498740

Table 1: Student number

3.1.1 Tasks for the project group - organization

We have decided to rotate the positions of leader and secretary in this project, to divide the tasks and responsibility likewise. Here is an overview of the current assigned leader and secretary.

Date	Leader	Secretary
20/01 - 20/02	Emilie	Maria
20/02 - 20/03	Maria	Nicklas
20/03 - 20/04	Nicklas	Linda
20/04 - 20/05	Linda	Emilie

Table 2: Leader and secretary

3.1.2 Tasks for project leader

The project leader will be responsible for

- setting up meetings
- make sure the desired progress is followed
- update the Gantt-scheme

3.1.3 Tasks for secretary

The secretary will have the responsibility for

- writing notes during meetings.
- sending weekly reports to group leader

3.1.4 Tasks for other members

The rest of the members will be

- assisting the leader and secretary

3.2 Steering group (supervisor, contact person and client)

Our client is Kongsberg Maritime, with Håkon Lunheim as our contact person, and Bjørnar Vik, Erlend Rangnes and Thomas Haraldsen will be assisting. Chaney W. Sætre will be assisting when needed.
As from NTNU Anete Vagale will be our main supervisor, Ottar L. Osen will be co-supervisor and Robin T. Bye will be a resource person we can rely on.

4 Agreements

4.1 Agreement with the client

The task is to develop a simulator that can be used to test and verify Mcon thruster control system during assembly and testing at Longva. The focus is to develop a scalable program that later can be expanded upon. Therefore, the communication and software that will be used have to be in line with what Kongsberg Maritime utilize. Since the goal of this project is to minimize the time use, it also has to implement the FAT test faster then the current method.

4.2 Workplace and resources

The project is to be carried out at NTNU in Ålesund, at the lab setup.

All of the supervisors work at NTNU in Ålesund. The thesis is given by Geir Olav Otterlei in Kongsberg Maritime, but our contact person during the project will be Håkon Lunheim, who work within walking distance from the school. Under normal circumstances this would mean that the supervisors will be available in a relatively short time, on request. Due to the COVID-19 situation our supervisors may not be as easy to meet up with at the workplace, and in that case communication will be carried out digitally.

Due to COVID-19, the group will be following the recommendations from FHI and the government. Guidelines may include shutdown of schools and workplaces. If this is the case, the group will be working from home. Working from home will also be done if members of the group are feeling ill or are in quarantine.

Our contact person for the lab setup, works av Automation Longva facility. Requests regarding the lab setup will there therefore have to be scheduled.

4.3 Group norms - cooperation rules - attitudes

The group have agreed to meet every weekday where the testing rig is, from 08.00 - 15.00/16.00 during this bachelor project. This is to ensure good and continuous work throughout this project period, and ensure good collaboration within the group.

The group is to be helpful to each other, encourage each other and be patient. Everyone in the group should be able to voice their opinions and treat each vision with respect. Everyone should be honest and precise with their work, and be on time for meetings and events.

5 Project description

5.1 Problem statement - objective - purpose

The problem statement can be divided into two parts. The first is to develop a SIL setup and system that allows for further expansion, regarding implementation of more thruster models. The other part of the problem statement is to make the semi-automatic test system, following the FAT-test requirements.

5.1.1 Problem formulation

Working through the initial phases of the project, it was scoped out a set of unknowns that had to be solved prior to starting, or during the progress of the project. It was known that CANbus and B&R simulation for the IO, were to be used for the communication. It was also known that mathematical calculation, such as thrust calculations where to be executed in MATLAB®/Simulink® and MapleSim. The remainder of the variables were unknowns that had to be solved. Therefore a range of cases were noted and is listed below

- How to develop a SIL setup that allow for further expansion?
- Which program is best suited for making the semi-automatic FAT-test?

5.1.2 Effect goals

- Create a project that can be further expanded and implemented with ease.

5.1.3 Performance target

- Simplify testing and create less connection time. (Reduce time spent on switching which further reduces costs?)

5.2 Requirements for solution or project result - specification

Run a successful FAT test without using physical components and successfully model the thruster control systems, so that the software cannot distinguish between simulation and physical testing.

5.3 Planned procedure(s) for the development work - method(s)

The group will work towards the deadlines in the Gantt chart. The tasks with the shortest timeline will be prioritized. The group will work on several activities at the same time, where the activities at any given time will have different priorities according to which must be completed first.

The group leader has the overall responsibility, considering working relation, time use and schedule meeting if necessary. The group member who is responsible for an activity has full responsibility for completing this activity on time. If possible, each activity will also have a co-manager. The starting point is that these two should collaborate on the activity so that two and two work together with each activity. This is to prevent any delays caused by problems such as being stuck on a problem, or the workload being to large for a single person.

If there are problems with completing an activity, the functioning group leader must be informed. The problem will first be discussed within the project group. In cases where important changes has to be made, the functional leader must schedule a meeting with the steering group to discuss the next step. The agenda for this meeting should be how to solve the issue, and what resources are needed so that the problem can be solved.

It is difficult to estimate how long and how much resources each activity will take. Activities can take more or less time resources than planned. Therefore, a dynamic approach is used, that facilitates the opportunity to update the plan continuously. A time buffer should therefore be included to accommodate for any potential adjustments or deviations from the original timeline/schedule.

5.4 Information gathering - performed and planned

During the preliminary project the group has acquired relevant literature on the topic of Mcon and thrusters. There is a lot of different thruster types which will give a different degree of difficulty. An important part of the project will be to find the best way to simulate and model the thrusters, as well as choosing the most fitting thruster. As regard to thrusters the group had little or zero prior knowledge.

5.5 Assessment - analysis of risk

Covid-19 could pose a large threat to the completion of this project, as we are relying on equipment which is located in a facility which could possibly be closed down due to a national (or regional) lockdown. Due to the risk of lockdown caused by Covid-19, the group should prioritize the modelling of the control system as a preventative measure. Completing the modelling of the thruster systems would dramatically decrease the risk of

a shutdown halting the progress of our project, since the group would not need the physical components to confirm test results. By completing the modelling aspect of the project, the group would be able to work remotely without depending on physical equipment or facilities to complete the project.

Other aspects in the event of the Covid-19 situation getting better is listed below.

Success factors:

- Realistic goals
- Updated plan
- Good communication and cooperation within the group
- Good communication and cooperation between the project group and the steering group

failure factors:

- Unrealistic goals
- Poor task distribution
- Lack of communication and bad cooperation within the group
- Lack communication and guidance from the steering group

5.6 Main activities in further work

- Literature study, maritime thruster systems.
- Simulate IO via CANbus
- Communication HIL
- Tunnel thrust calculations
- Simulation HIL - B&R Sim. Studio
- Visualization SIL
- Semi-automatic FAT
- Azimuth thrust calculations
- Test of systems

- Finish bachelor thesis report

For a detailed activity plan with a time frame and responsible person, please refer to the attached Gantt chart.

5.7 Progress plan - management of the project

5.7.1 Master plan

The project's main activities will be carried out, based on the Gantt chart. The diagram shows a full overview on estimated start date and end date for each activity, as well as who is responsible for the activity to be completed. The activities are divided into a main levels and a sub-levels since activities often with advantage can be divided into several sub-activities. Completed main activity is a milestone in the project.

5.7.2 Steering aids

The aid used for steering the project is Excel. This software is logical and clear, which makes it easy to keep a good overview of the project regarding activities to be preformed, person in charge, time management and milestones.

The project report is written in LaTeX, which is a professional text editor running in the browser. Everyone with access can modify and compile the report simultaneous. It also divide the chapters, which make it easier to work at the same time.

5.7.3 Development aids

A lab setup for Mcon thruster control is needed, for testing and verification for the simulated setup. Automation Studio simulation tool from B&R will be used to create models using tools such as MATLAB® /Simulink® and MapleSim for dynamic modeling of machines and machine components. For the thruster calculations MATLAB will be needed. For the semi-automatic Factory Acceptance Test, Java will be used.

5.7.4 Internal control - evaluation

The functional leader has the superior responsibility in regard to updating the Gantt chart, at least once a week. This will give a better work structure and overview.

The group will be working together from 08:00 to 15:00/16:00 in the weekdays. Doing this the communication and decisions during the project can be made fast and effectively.

A sub-goals is reached or completed when the results are good enough to be used in the later sub-goals.

5.8 Decisions - decision process

Important decisions on delimitation and clarification of the task and other key decisions made during the preliminary project, has and will be determined in our meetings with the steering group or with Kongsberg Maritime. These meeting was held 12.01.2021 and 21.01.2021. Minutes of meetings can be found the appendix.

Significant changes that has an impact on the project will be presented for the steering group, where a decision are to be made with everyone present.

6 Documentation

6.1 Reports and technical documents

The project leader will send out meeting notices to the steering group. This notice should include an agenda for the meeting, progress report (written by secretary) and planned events for the next period.
The minutes of meeting are to be written the same day as the meeting took place. It should include the key points discussed in the meeting. The minutes of meeting will be written by the secretary.

Every document will be stored in Google drive. This includes data sheets, progress reports, minutes of meetings, component lists and documentation from KM.

7 Planned Meetings and Reports

To validate that the progress is kept at a steady pace, and on the right track, the group will be having continuous meetings during the whole project.

7.1 Meetings

There will be a meeting with the supervisors at NTNU every other week, to get follow-up and feedback for the current provided work. Planned meetings:

- 12/01 - 14.30-14.55 Meeting with Kongsberg and NTNU to discuss the project

During this meeting, it was decided to have a meeting with supervisors at NTNU every other week to give updates on the progress of the project.

7.1.1 Meetings with Steering Group

Meetings with the steering group is primarily for providing relevant information, resources and equipment for executing the project. This is very important in the beginning of the project, to help us get started, but we will as well be having this meetings throughout the project. Planned meetings:

- 12/01 - 14.30-14.55 Meeting with Kongsberg and NTNU to discuss the project
- 21/01 - 10.00-11.00 Meeting with Kongsberg to discuss workplace, distributed files, software and equipment, starting goal and writing of contract

7.1.2 Project Meetings

There will as well be held meetings inside the group during the project to discuss any changes or uncertainty any of the members would carry, to help each other find the best solutions. Planned meetings:

- 14/01 - 09.00-10.30 Meeting about the formation and contents of the preliminary project report.

7.2 Periodic Reports

By writing periodic reports the work will be continuous throughout the project, and we will be provided with a bigger overview of the whole project.

7.2.1 Progress Reports (incl. milestone)

We will be writing a progress report each week, to see the progress of the current week. This will be sent to our supervisors at NTNU, which makes it easy for them to give feedback at the project meetings. The gantt-scheme will as well be updated each week to follow the progress of the planned work.

8 Planned deviation treatment

In the events of a deviation from the process of the project, the group member responsible for the task is to give notice to the group leader. The group leader will convene a meeting with the group, and together the group will consider the following questions:

- Can the deviation be solved with more resources?
- Can the deviation be avoided?
- Can the deviation be solved with external help?
- Can the task be replaced so the deviation can be avoided?

The steering group is to be informed after the group has evaluated.

9 Equipment requirements / Requirements for implementation

This project is a collaboration between students at NTNU and Kongsberg Maritime. Kongsberg Maritime will provide for everything the students needs in terms of equipment during this project period.

- B&R Automation Studio, KM/NTNU license
- Testing equipment delivered by KM.
- MATLAB®/Simulink®, NTNU licence.
- Windows computers

10 Attachments

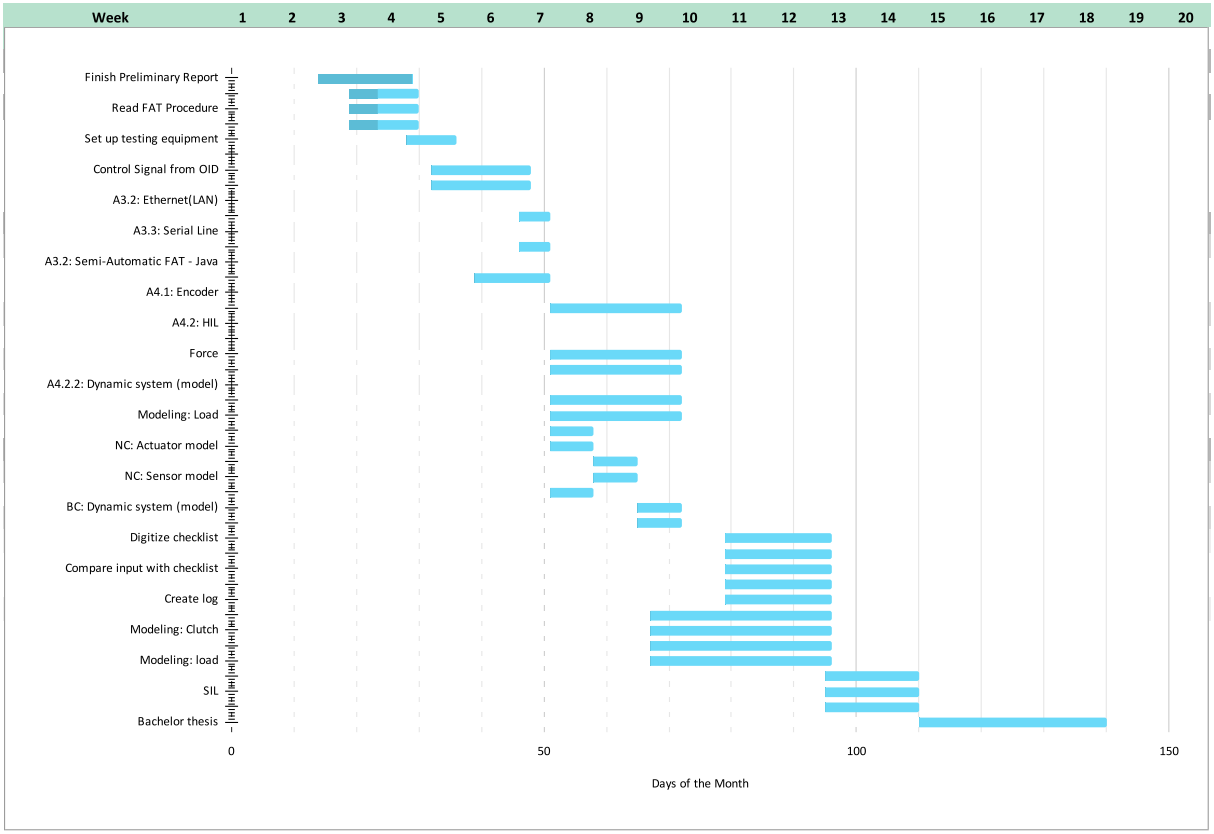
1. Gantt-scheme

10.1 Gantt-scheme

* = an automatically calculated cell

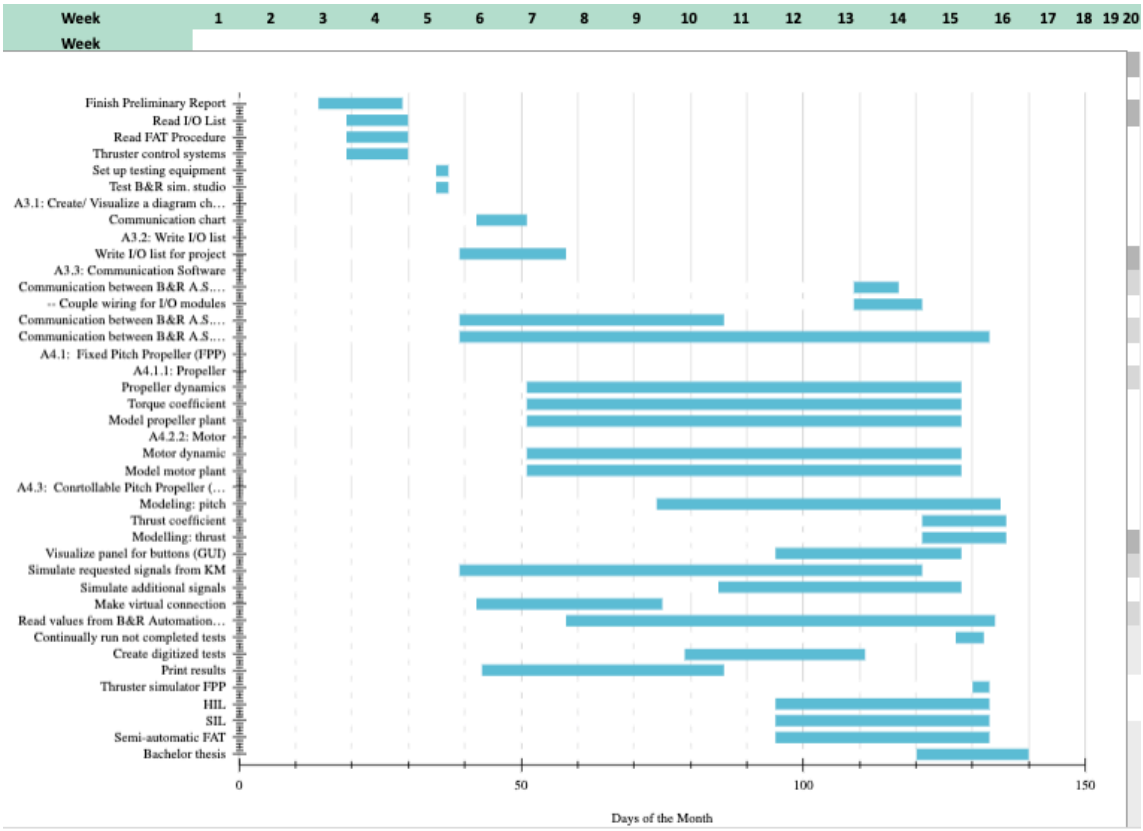
Mcon Thruster Simulator	START DATE	DAY OF YEAR*	END DATE	DURATION* (WORK DAYS)	DAYS COMPLETE*	DAYS REMAINING*	TEAM MEMBER	PERCENT COMPLETE
A1: Prelimany project							FGL	
Finish Preliminary Report	14.01	14	28.01	15	15	0	Everyone	100%
A2: Literatur Study							FGL	
Read I/O List	19.01	19	29.01	11	4,4	7	Everyone	40%
Read FAT Procedure	19.01	19	29.01	11	4,4	7	Everyone	40%
Thruster control systems	19.01	19	29.01	11	4,4	7	Everyone	40%
Set up testing equipment	28.01	28	04.02	8	0	8	Everyone	0%
A3: Communication HIL							FGL	
A3.1: CAN-bus							**Linda H.S.**	
Control Signal from OID	04.02	32	19.02	16	0	16	Linda H.S.	0%
Simulated measurements	04.02	32	19.02	16	0	16	Linda H.S.	0%
A3.2: Ethernet(LAN)							**Maria O.F.**	
Simulation on/off	15.02	46	19.02	5	0	5	Maria O.F.	0%
A3.3: Serial Line							**Nicklas M.**	
Simulation on/off	15.02	46	19.02	5	0	5	Nicklas M.	0%
A3.2: Semi-Automatic FAT - Java							**Emilie S.S.**	
TCP/IP -Java	08.02	39	19.02	12	0	12	Emilie S.S.	0%
A4: Tunnel thrust calculations							FGL	
A4.1: Encoder							**Nicklas M.**	
Program encoder	20.02	51	12.03	21	0	21	Nicklas M.	0%
A4.2: HIL							**Maria O.F.**	
A4.2.1: Actuator model							**Maria O.F.**	
Force	20.02	51	12.03	21	0	21	Maria O.F.	0%
Max voltage to power ratio	20.02	51	12.03	21	0	21	Maria O.F.	0%
A4.2.2: Dynamic system (model)							**Nicklas M.**	
Modeling: RPM	20.02	51	12.03	21	0	21	Nicklas M.	0%
Modeling: Load	20.02	51	12.03	21	0	21	Nicklas M.	0%
A5: Simulation HIL - B&R Sim. Studio							**FGL**	
Simulate panel for buttons	20.02	51	26.02	7	0	7	Linda H.S.	0%
NC: Actuator model	20.02	51	26.02	7	0	7	Emilie S.S.	0%
NC: Dynamic system (model)	27.02	58	05.03	7	0	7	Linda H.S.	0%
NC: Sensor model	27.02	58	05.03	7	0	7	Emilie S.S.	0%
BC: Actuator model	20.02	51	26.02	7	0	7	Emilie S.S	0%
BC: Dynamic system (model)	06.03	65	12.03	7	0	7	Linda H.S.	0%
BC: Sensor model	06.03	65	12.03	7	0	7	Emilie S.S.	0%

A6: Semi-automatic FAT							**FGL**	
Digitize checklist	20.03	79	05.04	17	0	17	Emilie S.S.	0%
Input from control system	20.03	79	05.04	17	0	17	Emilie S.S.	0%
Compare input with checklist	20.03	79	05.04	17	0	17	Emilie S.S.	0%
Print result	20.03	79	05.04	17	0	17	Linda H.S.	0%
Create log	20.03	79	05.04	17	0	17	Linda H.S.	0%
A7: Azimuth thrust calculations							**FGL**	
Modeling: Azimuth	08.03	67	05.04	29	0	29	Maria O.F	0%
Modeling: Clutch	08.03	67	05.04	29	0	29	Maria O.F	0%
Modeling: Pitch	08.03	67	05.04	29	0	29	Nicklas M.	0%
Modeling: load	08.03	67	05.04	29	0	29	Nicklas M.	0%
A8: Test of systems							**FGL**	
HIL	05.04	95	19.04	15	0	15	Nicklas M.	0%
SIL	05.04	95	19.04	15	0	15	Maria O.F.	0%
Semi-automatic FAT	05.04	95	19.04	15	0	15	Linda H.S. / Emilie S.S.	0%
A9: Finish bachelor thesis report							**FGL**	
Bachelor thesis	20.04	110	19.05	30	0	30	Everyone	0%


Week
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Finish Preliminary Report
Read FAT Procedure
Set up testing equipment
Control Signal from OID
A3.2: Ethernet(LAN)
A3.3: Serial Line
A3.2: Semi-Automatic FAT - Java
A4.1: Encoder
A4.2: HIL
Force
A4.2.2: Dynamic system (model)
Modeling: Load
NC: Actuator model
NC: Sensor model
BC: Dynamic system (model)
Digitize checklist
Compare input with checklist
Create log
Modeling: Clutch
Modeling: load
SIL
Bachelor thesis
0
50
100
150
Days of the Month

G Gantt-scheme

Mcon Thruster Simulator	START DATE	DAY OF YEAR*	END DATE	DURATION* (WORK DAYS)	DAYS COMPLETE*	DAYS REMAINING*	TEAM MEMBER	PERCENT COMPLETE
A1: Prelimany project							FGL	
Finish Preliminary Report	14.01	14	28.01	15	15	0	Everyone	100%
A2: Literatur Study and equipment							FGL	
Read I/O List	19.01	19	29.01	11	11	0	Everyone	100%
Read FAT Procedure	19.01	19	29.01	11	11	0	Everyone	100%
Thruster control systems	19.01	19	29.01	11	11	0	Everyone	100%
Set up testing equipment	04.02	35	05.02	2	2	0	Everyone	100%
Test B&R sim. studio	04.02	35	12.02	2	2	0	Everyone	100%
A3: Communication							FGL	
A3.1: Create/ Visualize a diagram chart of							**Maria O.F**	
Communication chart	11.02	42	19.02	9	9	0	Maria O.F.	100%
A3.2: Write I/O list							**Emilie S.S**	
Write I/O list for project	08.02	39	26.02	19	19	0	Emilie S.S	100%
A3.3: Communication Software							**Maria O.F**	
Communication between B&R A.S. and Mcon system	19.04	109	26.04	8	8	0	Emilie S.S	100%
-- Couple wiring for I/O modules	19.04	109	30.04	12	12	0	Emilie S.S	100%
Communication between B&R A.S. and Matlab/ Simulink (PLC)	08.02	39	26.03	47	47	0	Maria O.F.	100%
Communication between B&R A.S. and IntelliJ (OPC UA)	08.02	39	12.05	94	94	0	Linda H.S.	100%
A4: Thrust/RPM calculations							FGL	
A4.1: Fixed Pitch Propeller (FPP)							**Maria O.F.**	
A4.1.1: Propeller							**Maria O.F.**	
Propeller dynamics	20.02	51	07.05	77	77	0	Maria O.F.	100%
Torque coefficient	20.02	51	07.05	77	77	0	Maria O.F.	100%
Model propeller plant	20.02	51	07.05	77	77	0	Maria O.F	100%
A4.2.2: Motor							**Nicklas M.**	
Motor dynamic	20.02	51	07.05	77	77	0	Maria O.F	100%
Model motor plant	20.02	51	07.05	77	77	0	Nicklas M.	100%
A4.3: Conrtollable Pitch Propeller (CPP)							**Nicklas M.**	
Modeling: pitch	15.03	74	14.05	61	61	0	Nicklas M.	100%
Thrust coefficient	01.05	121	15.05	15	15	0	Nicklas M.	100%
Modelling: thrust	01.05	121	15.05	15	15	0	Nicklas M.	100%
A5: Simulation HIL							**FGL**	
Visualize panel for buttons (GUI)	05.04	95	07.05	33	33	0	Emilie S.S	100%
Simulate requested signals from KM	08.02	39	30.04	82	82	0	Emilie S.S	100%
Simulate additional signals	26.03	85	07.05	43	43	0	Emilie S.S.	100%
A6: Semi-automatic FAT							**FGL**	
Make virtual connection	11.02	42	15.03	33	33	0	Linda H.S	100%
Read values from B&R Automation Studio	27.02	58	13.05	76	76	0	Linda H.S	100%
Continually run not completed tests	07.05	127	11.05	5	5	0	Linda H.S	100%
Create digitized tests	20.03	79	20.04	32	32	0	Linda H.S.	100%
Print results	12.02	43	26.03	43	43	0	Linda H.S.	100%
A8: Test of systems							**FGL**	
Thruster simulator FPP	10.05	130	12.05	3	3	0	Maria O. F.	100%
HIL	05.04	95	12.05	38	38	0	Emilie S.S.	100%
SIL	05.04	95	12.05	38	38	0	Everyone	100%
Semi-automatic FAT	05.04	95	12.05	38	38	0	Linda H.S.	100%
A9: Finish bachelor thesis report							**FGL**	
Bachelor thesis	30.04	120	19.05	20	20	0	Everyone	100%


Week 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Week
Finish Preliminary Report
Read I/O List
Read FAT Procedure
Thruster control systems
Set up testing equipment
Test B&R sim. studio
A3.1: Create/ Visualize a diagram ch…
Communication chart
A3.2: Write I/O list
Write I/O list for project
A3.3: Communication Software
Communication between B&R A.S.…
-- Couple wiring for I/O modules
Communication between B&R A.S.…
Communication between B&R A.S.…
A4.1: Fixed Pitch Propeller (FPP)
A4.1.1: Propeller
Propeller dynamics
Torque coefficient
Model propeller plant
A4.2.2: Motor
Motor dynamic
Model motor plant
A4.3: Conrtollable Pitch Propeller (…
Modeling: pitch
Thrust coefficient
Modelling: thrust
Visualize panel for buttons (GUI)
Simulate requested signals from KM
Simulate additional signals
Make virtual connection
Read values from B&R Automation…
Continually run not completed tests
Create digitized tests
Print results
Thruster simulator FPP
HIL
SIL
Semi-automatic FAT
Bachelor thesis
0
50
100
150
Days of the Month

H I/O List

Cable/signal		From				To			
Signal		Term	Con		Pot	Term	Con	POT	POT
Pump Running	(Closed = Running)	-NGE3		21	Di	X20DO6639 (1)	11	NO	NO1
Pump Running		-NGE3		24	24V	X20DO6639 (1)	21	C	COM1
Stop System Pump	(Open = stop)	-X214		3	NC	X20DI6371 (2)	11	Di	DI1
Common		-X214		4		X20DI6371 (2)	14	24V	
Common		-X214		2	C	X20DI6371 (2)	24	24V	24V/ 24V
Start System Pump	(Closed = Start)	-X214		1	NO	X20DI6371 (2)	21	Di	DI2
RPM Order	4-20 mA	-X212		1	Ao	X20AI4322	11	Ai	AI +1 I
RPM Order	4-20 mA	-X212		2	ref	X20AI4322	13	Ref	AI -1 I
Drive RPM Feedback	4-20 mA	-NGE5		11	Ai	X20AO4622	11	Ao	AO +1 I
Drive RPM Feedback	4-20 mA	-NGE5		13	0V	X20AO4622	13	Ref	AO -1 U/I
Thruster Start	(Closed = Start)	-KNGE2.1		14	NO	X20DI6371 (1)	11	Di	DI1
Thruster Start		-KNGE2.1		11	C	X20DI6371 (1)	14	24V	24V
Thruster Stop	(Closed = Stop)	-KNGE2.2		14	NO	X20DI6371 (1)	21	Di	DI2
Thruster Stop		-KNGE2.2		11	C	X20DI6371 (1)	24	24V	24V
Thruster Running	(Closed = Running)	-NGE3		22	Di	X20DO6639 (2)	11	NO	NO1
Thruster Running		-NGE3		25	24V	X20DO6639 (2)	21	C	COM1
Drive Reset	(Closed = reset)	-KNGE2.3		14	NO	X20DI6371 (2)	12	Di	DI3
Drive Reset		-KNGE2.3		11	C	X20DI6371 (2)	15	24V	24V
DP enable	(Closed = enable)	-NCS2		12	Di	X20DO6639 (2)	12	NO	NO2
DP enable		-NCS2		15	24V	X20DO6639 (2)	22	C	COM2
DP ready	(Closed = ready)	-NDP1		11	NO	X20DI6371 (1)	12	Di	DI3
DP ready		-NDP1		12	C	X20DI6371 (1)	15	24V	24V
DP rpm order	4-20 mA	-NDP4		21	Ai	X20AO4622	21	Ao	AO +2 I
DP rpm order		-NDP4		23	0V	X20AO4622	23	Ref	AO -2 U/I
Dp rpm feedback	4-20 mA	-NDP2		23	Ao	X20AI4322	21	Ai	AI +2 I
Dp rpm feedback		-NDP2		24	Ref	X20AI4322	23	Ref	AI -2 I
Joystick enable	(Closed = enable)	-NCS2		21	Di	X20DO6639 (1)	12	NO	NO2
Joystick enable		-NCS2		24	24V	X20DO6639 (1)	22	C	COM2
Joystick ready	(Closed = ready)	-NJS1		11	NO	X20DI6371 (1)	22	Di	DI4
Joystick ready		-NJS1		12	C	X20DI6371 (1)	25	24V	24V
Joystick rpm order	4-20 mA	-NJS4		21	Ai	X20AO4622	14	Ao	AO +3 I
Joystick rpm order		-NJS4		23	0V	X20AO4622	16	Ref	AO -3 U/I
RPM feedback to Joystick	4-20 mA	-NJS2		23	Ao	X20AI4322	14	Ai	AI +3 I
RPM feedback to Joystick		-NJS2		24	Ref	X20AI4322	16	Ref	AI -3 I

I Progress Report

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 3	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 12.01.21 - 29.01.21		Project group	Date 29.01.2021

Main purpose/ focus for the work during this period

This period has been the first project period after the start-up meeting on 12.01.2021. Focus for the work during this period has been to work with the pre-project report, as well as get an overview of the task to be performed.

Planned activities during this period

Planned activities for this period were:

1. Getting started/ finish the pre-project report.
2. Draft progress plan (Gantt chart).
3. Refine and specify the task.
4. Create a project report template in LaTex.
5. Obtain relevant literature to be able to solve the problem.
6. Obtain relevant equipment to be able to execute the project.
7. Get the lab setup at NTNU in Ålesund.

Actually completed activities during this period

1. The pre-project report was finished on 28.01.2021.
2. Draft for the Gantt chart is made, with suggestions for activities, as well as duration of activity and person in charge of the activity.
3. During the meetings 21.01.2021 and 28.01.2021 refining and specifying the task has been of discussion. This has helped specify the task more than the original, but it may need more discussion later.
4. Report template is created in Latex and all team members have editing rights in the document.
5. The group has provided some relevant literature of the different thruster systems, from the internet and directly from Kongsberg Maritime. From this an agreement to start with a tunnel thruster. The group this thruster will be a good thruster to start with.
6. The windows computers will be provided by Kongsberg Maritime during week 5. The licenses for B&R Simulation Studio will be provided as fast as possible.
7. The lab setup was delivered to NTNU in Ålesund on 29.01.2021, but not put together.

Description of/ justification for any discrepancies between planned and actual activities

A small deviation in planned activity number 3. The task has been specified, but the group feels that the thesis may be more specified. This will be discussed within the group when the necessary equipment has been looked at and tried out, in order to get a better overview. Any changes will be discussed and determined with the steering group.

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 3	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 12.01.21 - 29.01.21		Project group	Date 29.01.2021

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

-

Main experience from this period

Gained an overview of the thesis and the time required.

Main purpose/ focus next period

The focus for the next period will be to put together the lab setup, as well as get started with the program B&R Simulation Studio. There will also be focus on looking over the Gannt chart after having tested some of the programs and equipment to see if anything should be changed.

Planned activities next period

A2: Literature study and equipment (set up testing equipment) - Everyone
A2: Literature study and equipment (Test B&R sim. studio) - Everyone

Other

-

Desire for/ need for guidance, topic in the teaching - discussion otherwise

-

Approval/ signature functional group leader	Signature of other group participants
Emilie Skalstad Skarbø - Signature	Linda Helen Sperre - Signature Maria Osa Furmyr - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 29.01.21 - 05.02.21		Project group	Date 05.02.2021

Main purpose/ focus for the work during this period

Focus during this period has been to get all the necessary equipment/ software and build the lab setup, as well as looking into how we should execute the communication.

Planned activities during this period

Planned activities for this period were:

1. Setting up the lab setup, and test it/getting to “know” it.
2. Test B&R Automation Studio

Actually completed activities during this period

1. The lab setup is ready for use and is tested.
2. The B&R Simulation Studio is not yet tested.

Description of/ justification for any discrepancies between planned and actual activities

A deviation can be found in section 2. The reason for this deviation is because we were missing the licenses for accessing the software.

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

- Removed NC/BC: Dynamic System (model)/Sensor model (Mcon system handles this internally)
- Added: Visualize panel for buttons (GUI), Simulate requested signals from KM, Simulate additional signals.

Main experience from this period

Came to understand better the lab setup and the FAT requirements.

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 29.01.21 - 05.02.21		Project group	Date 05.02.2021

Main purpose/ focus next period

The focus for the next period will be to work out a good plan for communication between B&R Simulation Studio and the Mcon system/ lab setup, as well as learning the program B&R Simulation Studio. Communication between the other parts will also be prioritized when this is done.

Planned activities next period

A2: Literature study and equipment (Test B&R sim. studio) - Everyone

A3: Communication
A3.1 Can-bus - Linda H.S.
A3.2: Ethernet (LAN) - Maria O.F.
A3.3: Serial Line - Nicklas M.
A3.4 Semi- Automatic FAT (Java) - Emilie S.S.

Other

-

Desire for/ need for guidance, topic in the teaching - discussion otherwise

-

Approval/ signature functional group leader	Signature of other group participants
Emilie Skalstad Skarbø - Signature	Linda Helen Sperre - Signature Maria Osa Furmyr - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 2	Company - Client Kongsberg Maritime	Page 1 of 3
Report for process Progress report	Period/ week(s) 05.02.21 - 12.02.21		Project group	Date 12.02.2021

Main purpose/ focus for the work during this period

Focus during this period has been to get the communication to work.

Planned activities during this period

Planned activities for this period were:

1. Test B&R Simulation Studio
2. Start working on the communication between the softwares in the project.

Actually completed activities during this period

1. Two of the group members with Microsoft computers have started getting to know B&R Simulation Studio.

Description of/ justification for any discrepancies between planned and actual activities

There is a small deviation in number 1, concerning the training in B&R Simulation Studio. We have decided that only two of the group members will use time to learn the program, in order to save valuable time.

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

The communication is to be done a little different than we first thought/ expected. After some discussion with Håkon Lunheim in the meeting 28.01.2021, we found out and gained a better understanding of how the communication should take place. Therefore we have decided, after an internal group meeting, that the communication section in the Gantt chart will be changed accordingly.

The section A3: Communication Software the new topics are listed below:

- Create/ Visualize a diagram chart of communication
- Write I/O list
- Communication between software
  - Communication between B&R and Mcon system (CPU)
  ----- Couple wiring for I/O modules
  - Communication between B&R and Matlab/ Simulink (PLC)
  - Communication between B&R and IntelliJ (TCP/IP)

In addition the group decided to go through the Gannt chart and update several parts, as the group has gained a better understanding on how to complete the task. Since these changes will not change the end result, only how it's accomplished, the group has been given free rein to

make the changes without approval of the steering group. All these changes are listed below.

Deleted tasks:
- Delete all tasks involving programming of encoder, since this is unnecessary.
- Delete all task about NC and BC for actuators

In the section A4: Tunnel Thruster calculations
- Fixed Pitch Propeller (FPP)
- Controllable Pitch Propeller (CPP)

Main experience from this period

Getting to know the test lab better and the software communication.

Main purpose/ focus next period

Keep working on the communication.

Planned activities next period

A3: Communication Software
A3.1: Create/ Visualize a diagram chart of communication
A3.2: Write I/O list for
A3.3: Communication between software

Other

-

Desire for/ need for guidance, topic in the teaching - discussion otherwise

-

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 2	Company - Client Kongsberg Maritime	Page 3 of 3
Report for process Progress report	Period/ week(s) 05.02.21 - 12.02.21		Project group	Date 12.02.2021

Approval/ signature functional group leader	Signature of other group participants
Emilie Skalstad Skarbø - Signature	Linda Helen Sperre - Signature Maria Osa Furmyr - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 12.02.21 - 19.02.21		Project group	Date 19.02.2021

Main purpose/ focus for the work during this period

Focus this period has been communication.

Planned activities during this period

Planned activities for this period were:

1. Create/ Visualize a diagram chart of communication
2. Write I/O list for CPU
3. Communication between software

Actually completed activities during this period

1. The group has created a diagram chart for communication. This chart will be extended with time.
2. I/O list has been started.

Description of/ justification for any discrepancies between planned and actual activities

The I/O list is not fully done. Therefore the group has decided to expand the time limit until 26.02.2021.

The group has a plan for the communication, but have not gotten the software to communicate yet. In addition, we are missing some I/O modules that have to be ordered in regard to the I/O list.

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

The communication section in the Gantt chart will have an extended time limit, and other tasks were started earlier than planned.

The task started earlier was:
A6: Semi- automatic FAT (Digitize checklist)
A4.1: Fixed pitch propeller (FPP)

Main experience from this period

- B&R Mapp technology

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 12.02.21 - 19.02.21		Project group	Date 19.02.2021

Main purpose/ focus next period

The main focus for the next period is to keep working on the B&R Simulation Studio, get the tunnel thruster calculations done and finish the digitized checklist. In addition, finish the I/O list and order the necessary I/O modules.

Planned activities next period

A4: Tunnel thruster calculations
A4.1: Fixed Pitch Propeller (FPP)

A5: Simulation HIL (Simulate panel buttons)

A6: Semi- automatic FAT (Digitize checklist)

Other

-

Desire for/ need for guidance, topic in the teaching - discussion otherwise

-

Approval/ signature functional group leader	Signature of other group participants
Emilie Skalstad Skarbø - Signature	Linda Helen Sperre - Signature Maria Osa Furmyr - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 1	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 22.02.21 - 26.02.21		Project group	Date 26.02.2021

Main purpose/ focus for the work during this period

Focus this period has been on learning B&R automation studio, and modelling of a fixed pitch propeller.

Planned activities during this period

Planned activities for this period were:

1. FPP modelling
2. HIL - Simulate panel buttons
3. Digitize FAT checklist

Actually completed activities during this period

1. The group has started working on creating the mathematical model for the fixed pitch propeller. The group has started programming the mathematical model into MATLAB. Expect to finish up the first model of FPP next week.
2. The IO list is completed, and have had meetings regarding B&R automation studio to learn more about the software. Have not started to simulate panel buttons yet.
3. Worked on thesis as the group is still waiting on getting the communication up and running

Description of/ justification for any discrepancies between planned and actual activities

The group is still waiting on getting the communication up and running, therefore we have not yet started working on the simulation of panel buttons and the digitization of the FAT checklist. The group members focused on writing up the theory section of the bachelor thesis in the meantime.

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

No changes in plan. The tasks that were started earlier than planned will be continued on, as both of these are not yet completed.

The task started earlier last week was:
A6: Semi- automatic FAT (Digitize checklist)
A4.1: Fixed pitch propeller (FPP)

Main experience from this period

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 1	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 22.02.21 - 26.02.21		Project group	Date 26.02.2021

- Physics, mathematics and understanding of propeller dynamics in regards to control theory
- IO mapping on B&R, and communication between B&R and Java

Main purpose/ focus next period

The main focus for the next period is to keep working on the B&R Simulation Studio, get the tunnel thruster calculations done and finish the digitized checklist.

Planned activities next period

- Tunnel thruster calculations
- B&R communication

The group will continue working on the same activities as last week as they are not completed yet, and also because the group will be occupied with Industri 4.0 most of next week.

Other

-

Desire for/ need for guidance, topic in the teaching - discussion otherwise

- How to implement thruster handles in B&R (not mapped on IO list)

Approval/ signature functional group leader	Signature of other group participants
Maria Osa Furmyr - Signature	Linda Helen Sperre - Signature Emilie Skalstad Skarbø - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 01.03.21 - 05.03.21		Project group	Date 05.03.2021

Main purpose/ focus for the work during this period

The main focus for this period is to keep working on the B&R Simulation Studio, get the tunnel thruster calculations done and finish the digitized checklist.

Planned activities during this period

Planned activities for this period were:

1. FPP modelling
2. HIL - Simulate panel buttons
3. Digitize FAT checklist

Actually completed activities during this period

1. The group has finished the mathematical model for the propeller, and programmed this into MATLAB.
2. The group started to work on simulating the buttons of the test jigg on B&R automation.
3. Started working on communication, and finished the digitized checklist for FAT checklist (except communication part)

Description of/ justification for any discrepancies between planned and actual activities

some delays due to issues on how to solve the MATLAB/simulink model of propeller. Half the week was spent on industri 4.0, so the group did not work as much on the project this week

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

continue to work on the FPP model for MATLAB. The deadline for the FPP model might be delayed since all of next will consist of lectures from 9-14 in industri 4.0.
The deadline for A5: simulation HIL will have to be delayed by 2 weeks (to 19.3) because of Industri 4.0.

Main experience from this period

- Physics, mathematics and understanding of propeller dynamics in regards to control theory
- IO mapping on B&R, and communication between B&R and Java

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 01.03.21 - 05.03.21		Project group	Date 05.03.2021

Main purpose/ focus next period

The main focus for the next period is to keep working on the B&R Simulation Studio, connect the tunnel thruster model to a model of a DC motor, and keep working on the communication between java and B&R.

Planned activities next period

- Tunnel thruster calculations/DC motor calculation
- B&R communication
- IO modules for B&R automation studio

The group will continue working on mostly the same activities as last week as they are not completed yet, and also because the group will be occupied with Industri 4.0 all of next week.

Other

Desire for/ need for guidance, topic in the teaching - discussion otherwise

Approval/ signature functional group leader	Signature of other group participants
Maria Osa Furmyr - Signature	Linda Helen Sperre - Signature Emilie Skalstad Skarbø - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 08.03.21 - 12.03.21		Project group	Date 12.03.2021

Main purpose/ focus for the work during this period

The main focus for this period is to keep working on the B&R Simulation Studio, and connect the DC motor model to the transfer function for the propeller.

Planned activities during this period

Planned activities for this period were:

1. Finish FPP model (including DC motor)
2. HIL - Simulate panel buttons
3. communication between B&R and java

Actually completed activities during this period

1. The group has finished the mathematical model for the DC motor.

Description of/ justification for any discrepancies between planned and actual activities

The group did not get a lot of time to work on the project, as every group member had industri 4.0 lectures from 9-14 mon-fri.

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

FPP model activity will be delayed. Because of industri 4.0. because of this delay, the next planned activity for the involved group members will also be pushed back. (A3.3 communication software delayed to 26.3)

It is also decided to remove Azimuth thrust calculations. Since this was an additional task added by the group, this will have no interfering with the result of the project. The tasks removed are:

A7: Azimuth thrust calculations
Modelling: Azimuth
Modelling: Clutch
Modelling: Pitch
Modelling: Load
Visualize load and RPM fpr thruster

Main experience from this period

- Physics, mathematics and understanding of DC motors for modelling.

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 08.03.21 - 12.03.21		Project group	Date 12.03.2021

- IO mapping on B&R
- Communication between B&R and Java

Main purpose/ focus next period

The main focus for the next period is to keep working on the B&R Simulation Studio, connect the tunnel thruster model to a model of a DC motor, and keep working on the communication between java and B&R.

Planned activities next period

- finish programming tunnel thruster calculations/DC motor calculation and implement the model in simulink.
- B&R communication between java and B&R

Other

Desire for/ need for guidance, topic in the teaching - discussion otherwise

Approval/ signature functional group leader	Signature of other group participants
Maria Osa Furmyr - Signature	Linda Helen Sperre - Signature Emilie Skalstad Skarbø - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 1	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 15.03.21 - 19.03.21		Project group	Date 19.03.2021

Main purpose/ focus for the work during this period

Focus during this period has been to get the communication to work.

Planned activities during this period

- finish programming tunnel thruster calculations/DC motor calculation and implement the model in simulink.
- B&R communication between java and B&R

Actually completed activities during this period

1. Have not completed any of the assigned tasks for this week. Waiting on feedback regarding MATLAB/simulink program for thruster calculations. B&R communication between Java and B&R is almost finished, just need to test connections.

Description of/ justification for any discrepancies between planned and actual activities

Have had some issues with the implementation of the thruster calculations in MATLAB/simulink. Sent a zip file with necessary information to Robin for feedback. Currently waiting for feedback. Group members have decided to start focusing on MATLAB-B&R communication and lab report in the meantime.

Group is falling more and more behind schedule for the past few weeks. Need to have an internal meeting where we go over the plan, and evaluate what can be done to solve the issues and get back on track.

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

The virtual connection in Java is now completed, a little earlier than anticipated.

Main experience from this period

Getting familiar with MATLAB-B&R communication and working on the lab report.

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 1	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 15.03.21 - 19.03.21		Project group	Date 19.03.2021

Main purpose/ focus next period

Keep working on the communication. Continue working on the report while waiting for feedback regarding thruster calculations. Have a meeting regarding the progress of the project and make changes necessary to get back on time according to schedule.

Planned activities next period

- Communication Software
- Write report
- Meeting regarding schedule
- Fix matlab/simulink implementation when receiving feedback from Robin.

Other

-

Desire for/ need for guidance, topic in the teaching - discussion otherwise

-

Approval/ signature functional group leader	Signature of other group participants
Maria Osa Furmyr - Signature	Linda Helen Sperre - Signature Emilie Skalstad Skarbø - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 22.03.21 - 26.03.21		Project group	Date 26.03.2021

Main purpose/ focus for the work during this period

Focus during this period has been to get the communication to work and write lab report.

Planned activities during this period

- Communication Software
- Write report
- Meeting regarding schedule
- Fix matlab/simulink implementation when receiving feedback from Robin.

Actually completed activities during this period

1. Have been working on the communication between B&R and matlab/simulink and downloaded necessary resources. The report writing is well underway, with theory and methods. The desired meeting has been held loosely when all the team members have been together physically.

Description of/ justification for any discrepancies between planned and actual activities

The communication was due to this week, but this requires a little bit more time. The matlab/simulink and B&R communication is thought to be finished soon, probably next week, or the week after. Between java and B&R the communication is thought to be complete, but is not tested since the PLC is not ready.

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

The group soon needs to put together the different parts of the project, and check that everything is working. The printing of test results in Java is completed, before the plan.

Main experience from this period

Getting familiar with matlab/simulink and B&R communication and working on the lab report.

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 22.03.21 - 26.03.21		Project group	Date 26.03.2021

Main purpose/ focus next period

Keep working on the communication and the report. Modelling the azimuth thrust calculations.

Planned activities next period

- Communication Software
- Write report
- Modelling azimuth thrust calculations

Other

-

Desire for/ need for guidance, topic in the teaching - discussion otherwise

-

Approval/ signature functional group leader	Signature of other group participants
Linda Helen Sperre - Signature	Maria Osa Furmyr - Signature Emilie Skalstad Skarbø - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 29.03.21 - 02.04.21		Project group	Date 02.04.2021

Main purpose/ focus for the work during this period

Focus during this period has been to get the communication to work and write lab report.

Planned activities during this period

- Communication Software
- Write report
- Modelling azimuth thrust calculations

Actually completed activities during this period

1. Have been working on the communication between B&R and matlab/simulink and have uploaded the matlab file into B&R. A lot of the report has been written, with focus on the theory and method parts. Have been looking at the pitch for the propeller.

Description of/ justification for any discrepancies between planned and actual activities

The matlab file is uploaded into B&R, but has not yet been tested. The modelling of azimuth thrust calculations has not been done, because the group has been working on communication and report writing.

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

The group has started putting together the different parts of the project and needs to be tested. The azimuth thrust calculation needs to be started on, as fast as the communication is tested and is working.

Main experience from this period

Getting familiar with matlab/simulink uploading to B&R and working on the lab report.

Main purpose/ focus next period

Test that the uploaded matlab file into B&R are working, and keep working on the report. Begin working with the azimuth thrust calculations after finished testing of matlab file into B&R.

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 29.03.21 - 02.04.21		Project group	Date 02.04.2021

Planned activities next period

- Test that the uploaded matlab file into B&R is working
- Write report
- Modelling azimuth thrust calculations
- Continue working with pitch for propeller

Other

-

Desire for/ need for guidance, topic in the teaching - discussion otherwise

-

Approval/ signature functional group leader	Signature of other group participants
Linda Helen Sperre - Signature	Maria Osa Furmyr - Signature Emilie Skalstad Skarbø - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 05.04.21 - 09.04.21		Project group	Date 09.04.2021

Main purpose/ focus for the work during this period

Focus during this period has been to get the communication to work, fix the buttons in B&R and write report.

Planned activities during this period

- Test that the uploaded matlab file into B&R is working
- Write report
- Modelling azimuth thrust calculations
- Continue working with pitch for propeller

Actually completed activities during this period

1. Have been working on the communication between B&R and matlab/simulink
2. Tried to fix the buttons in B&R
3. Started working on rpm
4. Written a lot of theory and method in the report.

Description of/ justification for any discrepancies between planned and actual activities

The modelling of azimuth thrust calculations has not yet been started working on, since the communication is not completed. Creation of the buttons in B&R has taken more time than planned, since this still isn't working as required. The communication between Java and B&R will need some more time.

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

The Gantt chart is updated in order to get better names and overview. The section updatet is A4: Tunnel Thrust calculations. This section now contains:

A4: Modelling
A4.1: Fixed Pitch Propeller (FPP)
A4.1.1: Propeller
Proeller dynamics
Torque coefficients
Model propeller plant
A4.2.2: Motor
Motor dynamics
Model motor plant
A4.3: Controllable Pitch Propeller (CPP)
Modeling: pitch

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 05.04.21 - 09.04.21		Project group	Date 09.04.2021

Thrust coefficient
Modelling: thrust

The finish date for this section is also moved to 01.05.

Main experience from this period

Learned a lot more about the tasks for the buttons in B&R, as well as the communication. Have also gotten familiar with the theory and method for the lab report.

Main purpose/ focus next period

Create the buttons in B&R, and get them to work properly. Continue working on rpm, communication and lab report. Begin working with the azimuth thrust calculations after finished communication.

Planned activities next period

- Create and fix buttons in B&R
- Work on rpm
- Fix communication
- Write report
- Modelling azimuth thrust calculations

Other

-

Desire for/ need for guidance, topic in the teaching - discussion otherwise

- Would be nice if the supervisor wants to read the report so far, for feedback.

Approval/ signature functional group leader	Signature of other group participants
Linda Helen Sperre - Signature	Maria Osa Furmyr - Signature Emilie Skalstad Skarbø - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 3 of 2
Report for process Progress report	Period/ week(s) 05.04.21 - 09.04.21		Project group	Date 09.04.2021

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 1	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 12.04.21 - 16.04.21		Project group	Date 16.04.2021

Main purpose/ focus for the work during this period

Main focus during this period has been to write report, try to fix the communication between B&R and simulink, and try to fix the buttons in B&R.

Planned activities during this period

- Create and fix buttons in B&R
- Work on rpm
- Fix communication
- Write report
- Modelling azimuth thrust calculations

Actually completed activities during this period

1. Tried to fix the communication between B&R and matlab/simulink
2. Tried to fix the buttons in B&R
3. Written a lot of theory and method in the report.

Description of/ justification for any discrepancies between planned and actual activities

The communication and buttons in B&R has been prioritised since this is necessary for the complete solution. The report has also been prioritised between the other activities, since this anyway is required at some point, and important to finish before the final deadline.

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

The buttons in B&R have to be fixed as fast as possible, since both the matlab file and the java FAT-test depend on this to work. Have been getting feedback from support regarding the communication between B&R and matlab/simulink, and have asked supervisors for help. The report will be continued on, and have been sent to supervisor as preliminary for feedback.

Main experience from this period

Have received a lot of feedback on the communication between B&R and simulink from support as well as the buttons in B&R from supervisors. Have also gotten familiar with some more of the theory and method for the lab report.

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 1	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 12.04.21 - 16.04.21		Project group	Date 16.04.2021

Main purpose/ focus next period

Since the whole project was scheduled to be finished now, except for the report, all the remaining activities will be delayed some.

The main focus for the next period will be to fix the communication between B&R and simulink, fix the buttons in B&R, decide what variables to be represented in B&R, finish rpm and write report.

Planned activities next period

- Fix communication between B&R and simulink
- Fix buttons in B&R, and create more
- Decide what variables to be in B&R
- Finish work on rpm
- Write some more tests in java
- Write report

Other

-

Desire for/ need for guidance, topic in the teaching - discussion otherwise

Approval/ signature functional group leader	Signature of other group participants
Linda Helen Sperre - Signature	Maria Osa Furmyr - Signature Emilie Skalstad Skarbø - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 19.04.21 - 23.04.21		Project group	Date 23.04.2021

Main purpose/ focus for the work during this period

Main focus during this period has been to write the report, fix the communication between B&R and simulink and try to finish the B&R program.

Planned activities during this period

- Finish the B&R program
- Work on rpm
- Fix communication
- Write report

Actually completed activities during this period

1. Fixed the communication between simulink/matlab & B&R automation studio
2. Tried to fix the program in B&R Automation Studio
3. Written a lot of theory and method in the report.

Description of/ justification for any discrepancies between planned and actual activities

The communication and buttons in B&R have been prioritised since this is necessary for the complete solution. The report has also been prioritised between the other activities, since this anyway is required at some point, and important to finish before the final deadline.

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

The program in B&R has to be fixed as fast as possible, since both the matlab file and the java FAT-test depend on this to work.

Main experience from this period

The communication between simulink/matlab and B&R automation studio have been resolved. The issue believed to cause this problem was an outdated B&R automation studio version, but this was possible to update.
Reviewing the mathematics behind the simulation in matlab.
Made some more FAT tests in Java.

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 19.04.21 - 23.04.21		Project group	Date 23.04.2021

Main purpose/ focus next period

Finish the RPM work.

The main focus for the next period will be testing all the programs (matlab/simulink, java and B&R as a whole) connected to the Mcon rack.

Planned activities next period

- Finish the last details on RPM work.
- Test the system as a whole
- Write report
- Review the mathematics

Other

-

Desire for/ need for guidance, topic in the teaching - discussion otherwise

Approval/ signature functional group leader	Signature of other group participants
Linda Helen Sperre - Signature	Maria Osa Furmyr - Signature Emilie Skalstad Skarbø - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 26.04.21 - 30.04.21		Project group	Date 30.04.2021

Main purpose/ focus for the work during this period

Main focus during this period has been to write the report, fix the communication between B&R and simulink and try to finish the B&R program.

Planned activities during this period

- Finish the B&R program
- Work on rpm
- Review the mathematics
- Test the communication between all the systems together
- Write report

Actually completed activities during this period

1. Finished the necessary signals in B&R Automation Studio, can add more if needed.
2. Fixed RPM
3. Written report
4. 95% up and running on the system

Description of/ justification for any discrepancies between planned and actual activities

The communication and buttons in B&R has been prioritised since this is necessary for the complete solution. The report has also been prioritised between the other activities, since this anyway is required at some point, and important to finish before the final deadline.

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

The communication between Java and B&R are starting to work some.

Main experience from this period

The B&R program is finished (can add more functions if needed).
Reviewing the mathematics behind the simulation in matlab.

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 26.04.21 - 30.04.21		Project group	Date 30.04.2021

Main purpose/ focus next period

Finish the RPM work.

The main focus for the next period will be testing all the programs (matlab/simulink, java and B&R as a whole) connected to the Mcon rack.

Planned activities next period

- Test the system as a whole
- Write report
- (Fix errors (if some occur))
- (Add more functions to the programs)

Other

-

Desire for/ need for guidance, topic in the teaching - discussion otherwise

Approval/ signature functional group leader	Signature of other group participants
Emilie Skalstad Skarbø - Signature	Maria Osa Furmyr - Signature Linda Helen Sperre - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 1	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 03.05.21 - 07.05.21		Project group	Date 07.05.2021

Main purpose/ focus for the work during this period

Main focus during this period has been to write the report, fix the communication between B&R and simulink and Java.

Planned activities during this period

- Test the system as a whole
- Write report
- (Fix errors (if some occur))
- (Add more functions to the programs)

Actually completed activities during this period

1. Tested the system (with MatLab/SimuLink)
2. Have added Drive Reset (not connected to Mcon yet)
3. Fixed errors, as they occurred.

Description of/ justification for any discrepancies between planned and actual activities

Having some problems with the communication between B&R and Java (OPC UA). This is believed is caused by having an outdated version, waiting for B&R.
Some problems with the signals from the I/O and B&R AS.

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

Main experience from this period

B&R and MatLab/SimuLink are working well together.
The signals are working fine in simulation mode, but not when connected to the CPU.

Main purpose/ focus next period

Get to test the whole system (with Java)
Finish the report

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 1	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 03.05.21 - 07.05.21		Project group	Date 07.05.2021

Planned activities next period

- Test the system as a whole
- Write report
- (Fix errors (if some occur))

Other

-

Desire for/ need for guidance, topic in the teaching - discussion otherwise

Could use some help regarding the signals.

Approval/ signature functional group leader	Signature of other group participants
Linda Helen Sperre - Signature	Maria Osa Furmyr - Signature Emilie Skalstad Skarbø - Signature Nicklas Mellum - Signature

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 1 of 2
Report for process Progress report	Period/ week(s) 03.05.21 - 07.05.21		Project group	Date 07.05.2021

Main purpose/ focus for the work during this period

Main focus during this period has been to write the report, fix the communication between B&R and simulink and Java.

Planned activities during this period

- Write report
- Run final tests
- (Fix errors (if some occur))

Actually completed activities during this period

1. Write report
2. Run final tests
3. Fixed errors, as they occurred.

Description of/ justification for any discrepancies between planned and actual activities

Description of/ justification for changes that are now desired in the content of the project itself or in the further procedure - or the progress plan

Main experience from this period

Everything is working as it should.

Main purpose/ focus next period

Finish the report

IE303612 Bachelor Thesis	Project: Mcon Thruster Simulator	Number of meetings this periode 0	Company - Client Kongsberg Maritime	Page 2 of 2
Report for process Progress report	Period/ week(s) 03.05.21 - 07.05.21		Project group	Date 07.05.2021

Planned activities next period

- Write report

Other

-

Desire for/ need for guidance, topic in the teaching - discussion otherwise

Approval/ signature functional group leader	Signature of other group participants
Linda Helen Sperre - Signature	Maria Osa Furmyr - Signature Emilie Skalstad Skarbø - Signature Nicklas Mellum - Signature

J Minutes of Meeting

Minutes of meeting

Date: 12.01.2021
Location: Digitally
Participants: Emilie Skalstad Skarbø, Linda Sperre, Maria Osa Furmyr, Nicklas Mellum, Håkon Lunheim, Geir Olav Otterlei, Robin T. Bye, Anete Vågal, Ottar L. Osen
Purpose of meeting: Start-up meeting main project

Start- up meeting for the main project Mcon Thruster Simulator was held Tuesday 12 of January 2021, digitally. The meeting started with every part presenting their vision of the project, and potential for expansion and reduction.

Since the project difficulty level can be regulated with ease, it was suggested to start with one thruster type, and if time allows, implement more. This was also something the group desired to focus on in the starting face. It was also brought up that this is a task that can be worked on further after our bachelor project is finished.

Maria Osa Furmyr, Secretary

Minutes of meetings

Date:	21.01.2021
Location:	Digitally
Participants:	Emilie Skalstad Skarbø, Linda Sperre, Maria Osa Furmyr, Nicklas Mellum, Håkon Lunheim
Purpose of meeting:	Start-up meeting wIth Kongsberg Maritime

The start- up meeting for the main project and the preliminary project Mcon Truster Simulator was held Thursday 21 of January 2021, digitally. The group had some questions about the parts that were a little defused in the thesis and about necessary equipment.

After some discussion on how to best attack our task we came up with the following conclusion. We will start with the IO together, then split into two and two. One group will work in the semi-automatic test program and one group will work on the simulation and with the cybernetic part.

Håkon Lunheim gave a short introduction on thruster, mainly the difference between solid pitch and variable pitch.

Solid pitch:	Only the speed of the propeller can be adjusted.
Variable pitch:	The angel of the blades on the propeller kan by adjusted, and by that adjust the speed.

We got to an agreement to start with solid pitch, and work on that until nearly finished. Then, if time allows, implement for variable pitch.

Håkon informed us that:

- The lab upset is nearly ready for transport to NTNU.
- The B and R simulation studio software licenses were good to go and were sent after the meeting.
- Loan of microsoft computers is approved, and will be handed out as fast as possible.
- A NDA (non disclosure agreement) must be signed, in order for us to get the information and some of the equipment needed.

In the time leading up to the next status meeting, the group will work on the pre-project report, investigate the possibilities within the software B and R simulation studio, and get to know and use the lab setup. In the future we will have status meetings every second week with Håkon Lunheim.

Maria Osa Furmyr, Secretary

Minutes of meetings

Date:	28.01.2021
Location:	Digitally
Participants:	Emilie Skalstad Skarbø, Linda Sperre, Maria Osa Furmyr, Nicklas Mellum, Håkon Lunheim
Purpose of meeting:	Update on necessary equipment and feedback on plan

Case list:

1. **Go through gantt-chart**

The template that the group has created is good, but a few things could be removed and some changes were added. This gave the group a better insight into how best to solve the discovery and which parts to focus on the most.

- When NC is disconnected, BC is to be turned on
- Which IO is in need to simulate
- Remove focus on encoder
- Remove focus on CANbus

2. **Necessary equipment**

Our test lab setup will be delivered to NTNU in Ålesund 29.01.2021. The group will have to set the lab together, with guidance from Chaney Wang Sætre digitally if needed.
Windows computers will be available within week 5.
B&R simulation Studio is to be downloaded, and licenses will be provided from Kongsberg Maritime.

3. **Other**

Until the next meeting, the group will focus on the different possibilities and what the group envisions.

Maria Osa Furmyr, Secretary

Minutes of meetings

Date:	10.02.2021
Location:	Digitally
Participants:	Emilie Skalstad Skarbø, Linda Sperre, Maria Osa Furmyr, Nicklas Mellum, Håkon Lunheim, Geir Olav Otterlei, Robin T. Bye, Anete Vågal, Ottar L. Osen, Chaney Sætre
Purpose of meeting:	Methods for communication and update on progress

Case list:

1. **Update on progress**
   The group shared info on what has been done since the previous meeting. The lab setup is mounted and the FAT has been completed. B&R simulation Studio is downloaded and has been gradually testet. The group has started working on the communication, but has not yet figured out how to do it.

2. **Communication between B&R simulation Studio and Mcon System/ lab**

   Håkon Lunheim will meet up at the lab at NTNU on 11.02.21 at 13:00 in order to show us and discuss how to best make the communication work. The plan is to prepare a chart for the IO flow, in order to create a better vision. The group will write a summary from the meeting, which will be shared with the steering group.

   We also got informed that a CPU is ordered, and will arrive at 09.03.21. Since this is a little late in the process, Geir Olav Otterlei will try to hurry the delivery. In the meanwhile, we talked about finding another CPU we could use while we wait for the main CPU. The application can be downloaded into the CPU. It's important when choosing the CPU, that there is enough force. CPU is the counterparty to the Mcon system we already possess.

3. **Necessary equipment**
   The computers are ready. The group will receive one computer on 11.02.21, when Håkon Lunheim will come visit the lab at NTNU. The other computer is located at the Automation facility Longva, and will be delivered as soon as possible.

4. **Other**
   <u>FAT:</u>
   There was some difficulty with the implementation of the FAT, in regard to if it was done the right way. Therefore we requested that a representative from

Kongsberg would come and show us. It was determined that Håkon Lunheim will show us on 11.02.21.

Information from Kongsberg:
Geir Olav Otterlei shared that Per Magne Dalseth, an employee at Kongsberg, is working in a similar task as we do. Even though he uses another method, he is working at NMK which is within walking distance. Therefore he can be of good help, and can show up at relatively short time if the group is stuck on something.

Geir Olav Otterlei has shared a Mcon presentation which contains more information on principle and how the system can be used on different vessels.

It was also shared that Kongsberg has great competence on the B&R Automation Studio, that we can use if any trouble comes up.

Information from NTNU:
Ottar Osen shared some thoughts on how to best attack the project in order to get the best report at the end, and the best results. It is important to take a step back and discuss why the programs, equipment etc. is the best to use in our situation. The decisions should be well-considered, and given a proper explanation as to why it is chosen, such as future upgrades to the project may choose another solution than we did.

Maria Osa Furmyr, Secretary

Minutes of meetings

Date: 11.02.2021
Location: 033 Gnisten, NTNU
Participants: Emilie Skalstad Skarbø, Linda Sperre, Maria Osa Furmyr, Nicklas Mellum, Håkon Lunheim
Purpose of meeting: Methods for communication and FAT review.

Case list:

1. **Communication between B&R simulation Studio and Mcon System/ lab**

The CPU is going to be placed with the I/O modules and is to replace the button setup. It will communicate with B&R Simulation Studio. The group will receive the CPU as fast as possible, as it is located at Automation Longva Facility at the moment.

2. **FAT**

The part the group had trouble with during the FAT is not possible to implement.

3. **Calculation of thrusters**

The group wanted to focus on a fixed pitch propeller in the beginning, to then develop a system for more advanced thrusters, such as controllable pitch propeller and Azimuth propeller. The group was unsure on how high difficulty the calculations should have. After some discussion, it was determined to keep it to only the necessary calculations, which at a later time can be implemented if time allows.

4. **Other**

An I/O signal chart has to be created, using information on the I/O in the cabinet. Håkon Lunheim will be providing a signal list for the cabinet.

Until next time the group will focus on getting the I/O signal list finished and learning to use B&R Simulation Studio.

Maria Osa Furmyr, Secretary

Minutes of meetings

Date: 24.02.2021
Location: Digitally
Participants: Emilie Skalstad Skarbø, Linda Sperre, Maria Osa Furmyr, Nicklas Mellum, Håkon Lunheim, Geir Olav Otterlei, Anete Vagale, Ottar L. Osen, Chaney Sætre

Purpose of meeting: Update on progress and figure out goal for the next 2 weeks

Case list:

1. **Update on progress**
   Tunnel Thruster Fixed Pitch Propeller Calculations (Matlab/ Simulink)
   Group has worked on figuring out the mathematical model for propeller

   B&R Automation Studio
   Learned software and done tutorials. Will be having a meeting with Yngve from Kongsberg who will teach us more about B&R automation studio.

   FAT digitized checklist
   Have looked into making a FAT test with java but uncertain about how to connect B&R to java. Usual way of doing it has been by running Java on different PCs with Canbus modules connecting to B&R automation. Ask Yngve about alternative methods of connecting java to B&R (TCP) when we have a meeting with him regarding B&R automation studio.

2. **Update on further plan**
   Make a blueprint of the project to get an overview of how the actual project will be in its final state and present it in the next meeting. This will be reviewed during the next steering meeting.
3. **I/O modules - order**
   Emilie has sent I/O modules that we want ordered. Ottar Osen stated that it is important to justify choices and write it in the report as well. After justification for IO modules, håkon will order/get the parts needed.
4. **Bachelor Thesis structure**
   Keep structure given by supervisor, however subheadings can be divided up into different categories.

   The theory section:
   Should contain all formulas necessary. If the group knows formulas by hand, there is no need to write it in theory. If the group has to look it up, write it down in the theory section. The formulas should be general.

Nicklas Mellum, Secretary

The method section:
There should be no new information with citations. Application of formulas to fit our problem.

The result section:
Using the formulas with numbers in to show the result of the project. Most of the citations should be in theory.

The group was advised not to use much time on dividing into correct parts, rather to focus on content. It is also important to make good figures and illustrations to improve readability of the report.

software for making figures:
- -draw.io
- -inDesign

5. **Other**

   <u>System Identification Toolbox</u>
   The group found some information on system identification Toolbox, which could come in handy if the input/output data was available and Kongsberg can provide it. Håkon Lunheim deliberated that he did not think such data was available, but is going to ask some co-workers, just in case.

   Information on input regarding I/O data:
   input data will always be in time domain. Can do a fft.

Minutes of meetings

Date: 17.03.2021
Location: Digital teams meeting
Participants: Emilie Skalstad Skarbø, Linda Sperre, Maria Osa Furmyr, Nicklas Mellum, Håkon Lunheim, Anete Vagale, Ottar L. Osen, Chaney Sætre, Robin T Bye.

Purpose of meeting: Update on progress

Case list:

1. **Update on progress**

   Tunnel Thruster Fixed Pitch Propeller Calculations (Matlab/ Simulink)

- The group has figured out how to solve the FPP and the motor mathematics, and is working on the model in Simulink.

  B&R Automation Studio

- Problem with definition of variable

- Mail sent to Anete to help with B&R

  FAT digitized checklist

- The test in Java is almost done. only need to establish connection to server

2. **Update on further plan**

   Have been busy with industri 4.0 for the past two weeks, so will provide an update on progress in the next meeting.

Nicklas Mellum, Secretary

3. **Communication using OPC UA**

   The FAT test is almost done. need to establish connection with the server in order to check if the FAT test is successful.

   suggestions for solving the problem:

   - will test the client with another server and also check if the port number is closed on the PLC (closed by default due to security reasons).
   - Check if the group needs to upload the B&R program to hardware in order to get the server to work.
   - Check if the group can establish connection with another OPC UA server first (can be found online).

4. **Propeller modelling/MATLAB/simulink**
   The group has figured out the mathematical equation for the DC motor and propeller, and is working on implementing this in MATLAB/simulink.

   The group is having some issues getting the matlab/Simulink code to work. Will send files to Robin for feedback.

5. **other**

Minutes of meetings

Date: 15.04.2021
Location: Digital teams meeting
Participants: Emilie Skalstad Skarbø, Linda Sperre, Nicklas Mellum, Håkon Lunheim, Anete Vagale, Ottar L. Osen, Robin T Bye.

Purpose of meeting: Update on progress

Case list:

1. **Update on progress**

   Communication

   - Problems with communication between B&R and simulink (have contacted B&R for help)

   B&R Automation Studio

   - Problems with variables in B&R. Can't set variables to false after being true.

   Other

   - General update on progress according to Gantt.

2. **Update on further plan**
   Everything should be finished now, for connecting the systems for testing. We are a little behind schedule, but have premade a buffer for this.

Linda Helen Sperre, Secretary

3. **Communication with B&R and simulink**
   Problems with communication between simulink and B&R. Works with B&R support, but not on our system. Wonder if this is because of an old version of the B&R program. The computers delivered by kongsberg should have no security limits. Can send the project to B&R support, and gets a lot of help. Can also try to uninstall the program and download it at new.

4. **Problem with B&R variables**
   If variables have been set to true, the variables will not go back to being false. Been looking at the B&R code during this meeting. Can try to test the program using buttons. Can't connect the variables to the buttons. Possible the start variable needs to be set to false when stop is set to true. The group and supervisors will continue to look at this after the meeting.

5. **Needs variables from B&R to make the FAT-test**
   Can decide what variables to be used in B&R so that the tests can be created. Should also decide variable names for these variables.

6. **Other**
   Will send the report to Anete for reading the report so far, for feedback.

Linda Helen Sperre, Secretary

Minutes of meetings

Date: 07.05.2021
Location: Digital teams meeting
Participants: Emilie Skalstad Skarbø, Maria O. Furmyr, Linda Sperre, Nicklas Mellum, Håkon Lunheim, Anete Vagale, Ottar L. Osen, Robin T. Bye, Chaney W. Sætre, Erlend Rangnes

Purpose of meeting: Update on progress

Case list:

1. **Update on progress**

- Date for presentation of bachelor
  20th of May

- Update regarding testing
  B&R Automation Studio and Simulink are now working as they should, also when connected to Mcon. Having some problems with connecting Java to OPC UA and B&R Automation Studio.
  - For problems regarding Java and OPC UA different solutions have been discussed and tried. Ex. UaExpert, Modicon, set up server and test with a testclient.

  Some fixing regarding the MATLAB/Simulink model is ongoing-
  - Add a PID regulator, some constants will be difficult to find and the output will be different since the model is not 100% realistic.

Other

- General update on progress according to Gantt.

2. **Update on further plan**
   Everything should be finished now, for connecting the systems for testing. We are a little behind schedule, but have premade a buffer for this.

Other talking points:

1. How to cite PDF sources.
   - It's ok to cite title, author and year.
2. Create a “red thread” that can be followed throughout the report.
3. The beginning and end should everyone be able to read and understand.

Emilie Skalstad Skarbø, Secretary

K Analysis of Risk

Report: Analysis of risk
Risk document:
Risk log

Terminology	
S	Sum, of risk consequence and probability
C	Consequence, for the risk to occur
P	Probability, for the risk to occur
Emilie Skalstad Skarbø	ESS
Linda Helen Sperre	LHS
Maria Osa Furmyr	MOF
Nicklas Mellum	NM
Functional group leader	FGL

Risk and uncertainty log: Mcon Thruster Simulator

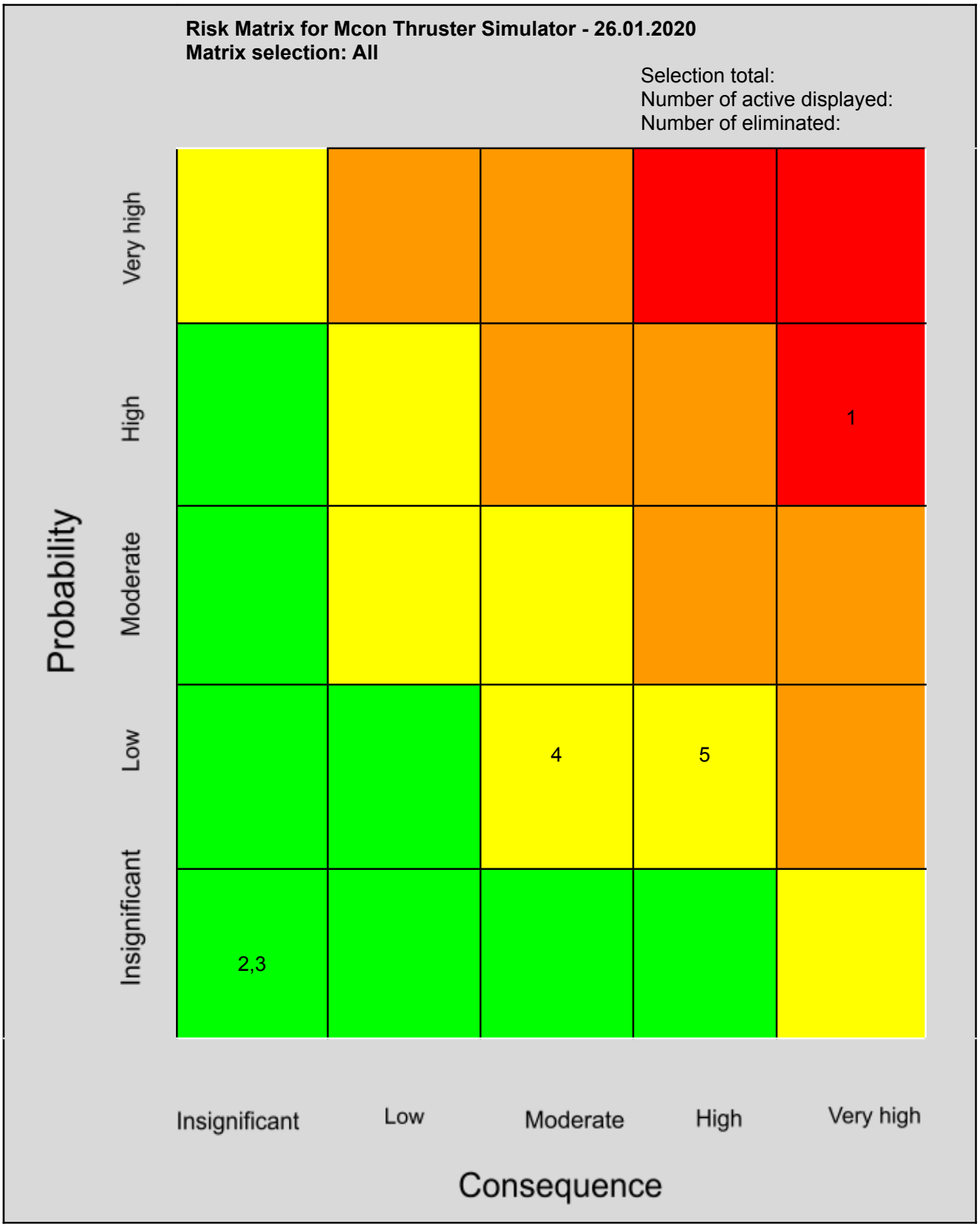
Risk area			Possible consequence					Propalsas for risk reduction measures						
Nr.	date identified: Risk description:	S	C	P	Data analysis: Impact assessment	Risk type	Date may occur	Date action assessed: Action description	Detail plan	Follow-up				
										Part	Deadline date	Responsibility	Date: State: Comment	State
1	26.01.2021 Shutdown due to COVID 19	9	5	4	26.01.2021	Project	26.01. 2021 -	26.01.2021 Work longer days at det lab setup when available	Bigger plan includ ed at the botto m			ESS LHS MOF NM		Not finish ed
2	26.01.2021 Too short time frame for the project	1	1	1	26.01.2021 Project not completed according to agreement	Project		26.01.2021 Work longer days Reduce the scope of activities in consultatio n with the steering group if the problem arises	Bigger plan includ ed at the botto m			FGL		Not finish ed
3	26.01.2021 Unclear project plan	1	1	1	26.01.2021 Activities are	Project		26.01.2021 Prepare a thorough	Bigger plan includ			ESS LHS MOF		Not finish ed

					overlooked, forgotten or not completed			project plan	ed at the botto m			NM		
4	26.01.2021 Project stop or unpredictable events occur	5	3	2	Project may stop due to unpredictab le reasons. These reasons may be lack of information, equipment or resources.	Project and syste m		26.01.2021 Asking for help from supervisor s and persons at KM, or others.	Bigger plan includ ed at the botto m			ESS LHS MOF NM		Not finish ed
5	26.01.2021 Loss of data	6	4	2	Loss of data such as code, or other relevant information could cause small to moderate delays.	Project and syste m	26.01 .2021 -	Prevent data loss by uploading all information to google drive, and all code to github	Bigger plan includ ed at the botto m			ESS LHS MOF NM		Not finish ed
6														

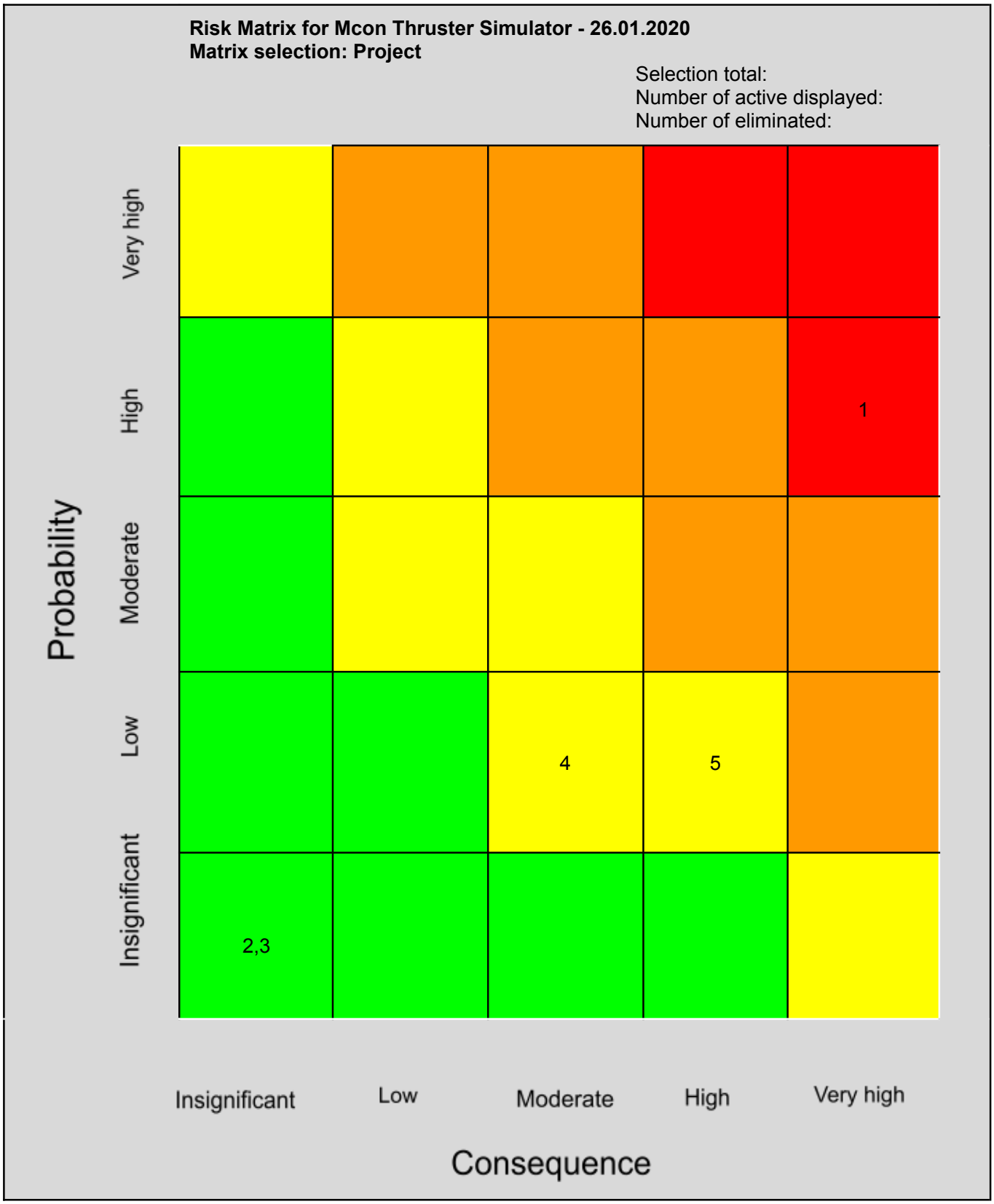
Report: Analysis of risk

Explanation: The meaning of "Consequences"		
	Value	Description
1	Insignificant	- No significant delay - No additional financial costs - Insignificant damage that can be repaired quickly - Traces of dissatisfaction among some people - No disturbance due to COVID 19
2	Low	- Minor delays - Low additional financial costs - Small damage that can be repaired quickly - Dissatisfaction arises in som groups - Low disturbance due to COVID 19
3	Moderate	- Moderate delays - Moderate additional financial costs within budget - Moderate damage that can be repaired within an acceptable time - Moderate dissatisfaction leads to some absence in use / participation - Moderate disturbance due to COVID 19
4	High	- Longer delays - High financial additional cost that goes beyond budget - Long-term damage that requires a long time to repair - Large degree of dissatisfaction which leads to a small degree of application/ participation - High disturbance due to COVID 19
5	Very high	- Very long delay or full stopp - Very high additional financial costs - Large damage that is difficult to repair - Very high dissatisfaction and no application/ participation - Very high disturbance due to COVID 19

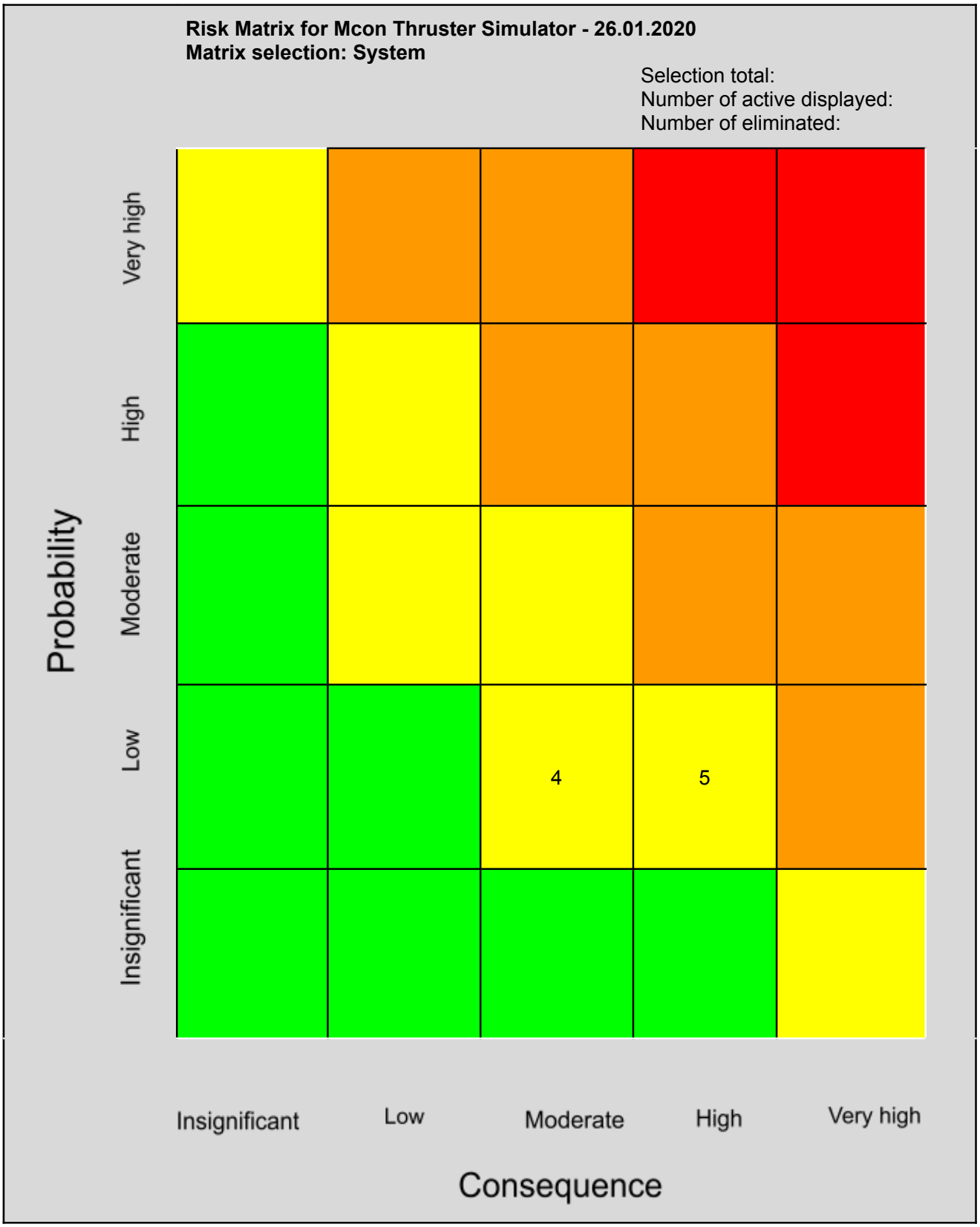
Risk document:
Risk matrix



Risk document:
Risk matrix



Risk document:
Risk matrix



Bigger detail plan for reducing risk

Project Risk	Explanation	Measures	Action
1. Shutdown due to Covid-19	Due to the ongoing pandemic, the schools and companies may have to close down to prevent further contamination of the Covid-19 virus.	Work longer days at det lab setup when available.	Work longer days at det lab setup when available.
2. Too short time frame for the project	The project may turn out to be too big for a few months' work.	Don't over do the tasks. Keep them simple and then add if needed.	Work longer days. Reduce the scope of activities in consultation with the steering group if the problem arises.
3. Unclear project plan	May stray far from the project plan (Gantt) if misinterpreted project or the group went in a different direction.	Set up meetings with the supervisor(s) and make sure every member of the group understands the project goal. Clarify any issues or uncertainties as early as possible.	Prepare a thorough project plan.
4. Project stop or unpredictable events occur	The project may stop due to unpredictable events.	Make a general plan for measures to be taken if any unpredictable events occur. Work location, change priorities/order of tasks.	Ask for help from supervisors and persons at KM, or others.
5. Loss of data	Gathered data/documentation and code can be lost if not saved or uploaded to google drive and github.	If things are written in a word file, save often. Upload information and code so it is saved.	Prevent data loss by uploading all information to google drive and all code to github. Save as well versions locally on harddrive.

L Source Code B&R Automation studio

For converting the source code into PDF-files, the website Hilite.me [52] was used.

L.1 B&R Source Code

```
PROGRAM _INIT
	(* Insert code here *)
END_PROGRAM

PROGRAM _CYCLIC
	(* This code is a part of a bachelor thesis regarding the Mcon
Thruster Control System for Kongsberg Maritime.
		The signal handling is handled here, as well as the GUI.*)
	STA_IN := (INT_TO_REAL(A_IN)/32767) * 16 + 4; // INT to 4-20mA

	RPM_order := ((STA_IN-12)/8)*100; //RPM order in percent

	STA_OUT := ((RPM_feedback)/100)*8+8+4; // 4-20mA

	A_OUT := REAL_TO_INT(((STA_OUT-4)/16) * 32767); // 4-20mA to INT

	// Setting variables for pump inputs
	TON_startPump.IN := startPump;
	TON_stopPump.IN := stopPump;

	// Setting timer for pump
	TON_startPump(PT := T#5s200ms);
	TON_stopPump(PT := T#5s200ms);

	// Logic for pump running (output)(Stop Pump: open = stop (NC))
	pumpRunning := (pumpRunning OR TON_startPump.Q) AND NOT
(TON_stopPump.Q);

	// Setting variables for Thruster inputs
	TON_startThruster.IN := startThruster;
	TON_stopThruster.IN := stopThruster;

	// Setting timer for thruster
	TON_startThruster(PT := T#5s200ms);
	TON_stopThruster(PT := T#5s200ms);

	// Case for Thruster control. Thruster will only run when/if pump is
running.
	CASE thrusterCtrl OF
		0: //Stopped, not ready.
			IF pumpRunning THEN
				thrusterCtrl := 1;
			END_IF

		1: // Stopped, ready
			IF TON_startThruster.Q THEN (* Go to Runnig state when
timer for start signal is elapsed.*)
				thrusterCtrl := 2;
			END_IF

			IF NOT pumpRunning THEN  (* Go back to not ready state
if pump is stopped.*)
				thrusterCtrl := 0;
```

```
                        END_IF


                2: // Running
                        thrusterRunning := TRUE;
                        IF TON_stopThruster.Q THEN (* Go to Stopping state when
timer for stop signal is elapsed.*)
                                thrusterCtrl := 3;
                        END_IF

                3: // Stopping
                        thrusterRunning := FALSE;
                        IF NOT thrusterRunning THEN (* Go to case 0 or 1
depending on if the pump is running or not.*)
                                thrusterCtrl := 0;
                        END_IF

                4: // Tripped
                        stopThruster := TRUE;

        END_CASE

        // Drive Reset
        driveReset;

        // DP enabled, send RPM order, receive RPM Feedback.
        IF DPEnable THEN
                DP_RPM_Order;
                DP_RPM_Feedback;
                DPReady := TRUE;
        ELSE
                DPReady := FALSE;
        END_IF


        // Joystick Enables, send RPM order, receive RPM Feedback.
        IF JoystickEnable THEN
                Joystick_RPM_Order;
                Joystick_RPM_Feedback;
                JoystickReady := TRUE;
        ELSE
                JoystickReady := FALSE;
        END_IF

END_PROGRAM

PROGRAM _EXIT
        (* Insert code here *)
END_PROGRAM
```

M Source Code Matlab

For converting the source code into PDF-files, the website Hilite.me [52] was used.

M.1 Source Code FPP model Matlab

```
%% DC MOTOR-------------------------------------------------------------
-----------------------------

   %1.st order system used to simulate/approximate a DC motor.
   Km = 2430;     % Torque constant, based of torque produced from
datasheet.
   Tm = 2;        % Time constant, based off of settling time of 8 sec.

   TF_motor = tf(Km, [Tm 1]);

   %To check magnitude and time constant.
   stepplot(TF_motor) %test to see if system behaves as intended
   bode(TF_motor);
   %% PROPELLER DYNAMICS-----------------------------------------------
----------------------------------------
   %calculates the torque and thrust coefficients based on the
parameters
   %given.


   %Max rpm of motor and propeller. Used to calculate gear ratio.
   max_RPM_motor = 1200;       % max RPM of motor
   max_RPM_propeller = 199;    % max RPM of propeller
   gear_ratio = max_RPM_propeller/max_RPM_motor;



   %creates a torque and thrust coefficient based on the values
   %from the Wagening B screw series propeller table

   %The propeller characteristics
   D = 2.4;                    % diameter of propeller in meters
   z = 4;                      % number of blades on propeller
   rho = 1025;                 % average density of surface seawater
(1020-2019)
   AEAO = 0.7;                 % blade area ratio. The area of the
propeller blade divided by the circular area of the propeller rotation
   PD = 1;                     % Pitch diameter ratio
   Ja = 0;                     % Avdvanced velocity

   [KT, KQ] = wageningen(Ja,PD,AEAO,z);  % Function call on
wageningen.m made by Thor I. Fossen
```

M.2 Source Code FPP model Matlab

```
%% DC MOTOR-------------------------------------------------------------
------------------------------
   %1.st order system used to simulate/approximate a DC motor.
   Km = 2430;      % Torque constant, based of torque produced from
datasheet.
   Tm = 2;         % Time constant, based off of settling time of 8 sec.

   TF_motor = tf(Km, [Tm 1]);

   %To check magnitude and time constant.
   stepplot(TF_motor) %test to see if system behaves as intended
   bode(TF_motor);
   %% PROPELLER DYNAMICS-----------------------------------------------
----------------------------------------
   %calculates the torque and thrust coefficients based on the
parameters
   %given.


   %Max rpm of motor and propeller. Used to calculate gear ratio.
   max_RPM_motor = 1200;       % max RPM of motor
   max_RPM_propeller = 199;    % max RPM of propeller
   gear_ratio = max_RPM_propeller/max_RPM_motor;

   %Maximum thrust produced. Used to convert thrust into percentage in
   %MATLAB. Aquired experimentally by setting RPM and pitch to max
during
   %Simulink simulation.
   max_thrust = 238156.81785138;



   %creates a torque and thrust coefficient based on the values
   %from the Wagening B screw series propeller table

   %The propeller characteristics
   D = 2.4;                    % diameter of propeller in meters
   z = 4;                      % number of blades on propeller
   rho = 1025;                 % average density of surface seawater
(1020-2019)
   AEAO = 0.7;                 % blade area ratio. The area of the
propeller blade divided by the circular area of the propeller rotation
   PD = 1;                     % Pitch diameter ratio
   Ja = 0;                     % Avdvanced velocity

   [KT, KQ] = wageningen(Ja,PD,AEAO,z);  % Function call on
wageningen.m made by Thor I. Fossen



    %% PITCH DYNAMICS
    %Pitch calculated based on values given below.

    h = 1;                              % Pitch ratio
```

```
    considered_radius = 1;              % considered radius based on
the angel of attack
    R = D/2;                            % Radius [m]
    r = (considered_radius / R);        % relative radius
    max_pitch = 1.4;

    %Variables used to create simplified pich actuator transfer
function
    pitch = atan(h/ pi* r);       % calculate pitch angel
    tau_PD = 1;                   % Pitch dynamics time constant
```

N Source Code Java

For converting the source code into PDF-files, the website Hilite.me [52] was used.

N.1 Interface ClientExample

```java
package main.java;

import java.util.function.Predicate;
import org.eclipse.milo.opcua.sdk.client.api.identity.AnonymousProvider;
import org.eclipse.milo.opcua.sdk.client.api.identity.IdentityProvider;
import org.eclipse.milo.opcua.stack.core.security.SecurityPolicy;
import
org.eclipse.milo.opcua.stack.core.types.structured.EndpointDescription;

/**
 * interface for establishing a connection
 */
public interface ClientExample {

    /**
     * getting the endpoint
     * @return the connection endpoint containing IP-address and port
number
     */
    static String getEndpointUrl() {
        return "opc.tcp://192.168.1.110:4840";
    }

    /**
     * filter for endpoint
     * @return the filtered endpoint
     */
    static Predicate<EndpointDescription> endpointFilter() {
        return e ->
getSecurityPolicy().getUri().equals(e.getSecurityPolicyUri());
    }

    /**
     * getting the security policy
     * @return the security policy
     */
    static SecurityPolicy getSecurityPolicy() {
        return SecurityPolicy.None;
    }

    /**
     * getting the identity provider
     * @return the identity provider
     */
    static IdentityProvider getIdentityProvider() {
        return new AnonymousProvider();
    }
}
```

N.2 Interface Connection

```java
package main.java;

/**
 * interface for making two options for connection
 */
public interface Connection {

    /**
     * getting the value of rpmControl
     * @return the value of rpmControl
     */
    int getRpmControlValue();

    /**
     * getting the state of thrusterMotor
     * @return the state of thrusterMotor
     */
    boolean getThrusterMotorValue();

    /**
     * getting the state of thrusterMotorStart
     * @return the state of thrusterMotorStart
     */
    boolean getThrusterMotorStartValue();

    /**
     * getting the state of thrusterMotorStop
     * @return the state of thrusterMotorStop
     */
    boolean getThrusterMotorStopValue();

    /**
     * getting the state of thrusterPumps
     * @return the state of thrusterPumps
     */
    boolean getThrusterPumpsValue();

    /**
     * getting the state of thrusterPumpsStart
     * @return the state of thrusterPumpsStart
     */
    boolean getThrusterPumpsStartValue();

    /**
     * getting the state of thrusterPumpsStop
     * @return the state of thrusterPumpsStop
     */
    boolean getThrusterPumpsStopValue();

    /**
     * getting the state of reset drive
     * @return the state of reset drive
     */
    boolean getResetDriveValue();

    /**
     * getting the value of dpInterface
```

```
     * @return the value of dpInterface
     */
    int getDpInterfaceValue();

    /**
     * getting the state of dpInterfaceStart
     * @return the state of dpInterfaceStart
     */
    boolean getDpInterfaceStartValue();

    /**
     * getting the value of joystick
     * @return the value of joystick
     */
    int getJoystickValue();

    /**
     * getting the state of joystickStart
     * @return the state of joystickStart
     */
    boolean getJoystickStartValue();
}
```

N.3 Class CreatePDF

```
package main.java;

import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.pdmodel.PDPage;
import org.apache.pdfbox.pdmodel.PDPageContentStream;
import org.apache.pdfbox.pdmodel.font.PDType1Font;

import java.awt.*;
import java.io.File;
import java.io.IOException;

/**
 * class for creating a PDF-file
 */
public class CreatePDF {

    /**
     * creating a PDF based on test results
     * @param rpmControlTestResult rpmControlTestResult result of rpm test
     * @param remoteStartStopThrusterMotorTestResult result of thruster
motor test
     * @param remoteStartStopThrusterServoPumpsTestResult result of
thruster pumps test
     * @param resetDriveTestResult result of reset drive test
     * @param dpInterfaceTestResult result of dp interface test
     * @param joystickTestResult result of joystick interface test
     * @throws IOException throw if something goes wrong
     */
    public static void createPDF(boolean rpmControlTestResult, boolean
remoteStartStopThrusterMotorTestResult, boolean
remoteStartStopThrusterServoPumpsTestResult, boolean resetDriveTestResult,
boolean dpInterfaceTestResult, boolean joystickTestResult) throws
IOException {

        File file = new File("Fat-test.pdf");
//saving the path to an existing template document
        PDDocument doc = PDDocument.load(file);
//loading an existing template document

        PDPage page = doc.getPage(0);
//creating a PDF Document
        PDPageContentStream contentStream = new PDPageContentStream(doc,
page);          //creates an instance of a PDPageContentStream

        contentStream.beginText();
//beginning the content stream
        contentStream.setFont(PDType1Font.TIMES_ROMAN, 24);
//setting the font of the content stream
        contentStream.setLeading(14.5f);
//setting the leading
        contentStream.newLineAtOffset(25, 725);
//setting the position for the line

        String textHeadline = "Factory Acceptance Test";
//headline for document
        String textUnderHeadline = "2 Functional Tests";
//under headline
```

```
        String textUnderUnderHeadline = "2.1 Tunnel thrusters";
//under under headline
        String textUnderUnderUnderHeadline1 = "2.1.1 General functions";
//under under under headline 1
        String textUnderUnderUnderHeadline2 = "2.1.2 Rpm Control";
//under under under headline 2
        String textUnderUnderUnderHeadline3 = "2.1.3 Start / stop and reset
functions";  //under under under headline 3
        String textUnderUnderUnderHeadline4 = "2.1.4 Interfaces to external
systems";    //under under under headline 4

        String text1 = "  1. test: 2.1.1.1 Power-up test: ";
//the name of test one
        //String result1 = "" + powerUpResult;
//the result of test one
        String text2 = "  2. test: 2.1.1.2 Dimmer: ";
//the name of test two
        //String result2 = "" + dimmerTestResult;
//the result of test two
        String text3 = "  3. test: 2.1.1.3 Command Transfer: ";
//the name of test three
        //String result3 = "" + commandTransferTestResult;
//the result of test three
        String text4 = "  4. test: 2.1.1.4 Motorized Lever: ";
//the name of test four
        //String result4 = "" + motorizedLeverTestResult;
//the result of test four
        String text5 = "  5. test: 2.1.2.1 Rpm Control: ";
//the name of test five
        String result5 = "" + rpmControlTestResult;
//the result of test five
        String text6 = "  6. test: 2.1.2.2 Backup Control: ";
//the name of test six
        //String result6 = "" + backupControlTestResult;
//the result of test six
        String text7 = "  7. test: 2.1.3.1 Remote Start/Stop of Thruster
Motor: ";       //the name of test seven
        String result7 = "" + remoteStartStopThrusterMotorTestResult;
//the result of test seven
        String text8 = "  8. test: 2.1.3.2 Remote Start/Stop of Thruster
Servo Pumps: "; //the name of test eight
        String result8 = "" + remoteStartStopThrusterServoPumpsTestResult;
//the result of test eight
        String text9 = "  9. test: 2.1.3.3 Reset Drive: ";
//the name of test nine
        String result9 = "" + resetDriveTestResult;
//the result of test nine
        String text10 = "10. test: 2.1.4.1 DP Interface: ";
//the name of test ten
        String result10 = "" + dpInterfaceTestResult;
//the result of test ten
        String text11 = "11. test: 2.1.4.2 Joystick Interface: ";
//the name of test eleven
        String result11 = "" + joystickTestResult;
//the result of test eleven
```

```
        contentStream.showText(textHeadline);
//add a headline
        contentStream.newLine();
//add a new line
        contentStream.setFont(PDType1Font.TIMES_ROMAN, 16);
//setting the font of the content stream

        contentStream.newLine();
//add a new line
        contentStream.showText(textUnderHeadline);
//add a under headline
        contentStream.newLine();
//add a new line
        contentStream.showText(textUnderUnderHeadline);
//add a under under headline
        contentStream.newLine();
//add a new line

        contentStream.showText(textUnderUnderUnderHeadline1);
//add a under under under headline
        contentStream.newLine();
//add a new line
        contentStream.showText(text1);
//adding the name of test one
        //setResultColor(contentStream, powerUpResult);
//set color of text based on result of test
        //contentStream.showText(result1);
//adding the result of test one
        resetResultColor(contentStream);
//reset color of text to black

        contentStream.showText(text2);
//adding the name of test two
        //setResultColor(contentStream, dimmerTestResult);
//set color of text based on result of test
        //contentStream.showText(result2);
//adding the result of test two
        resetResultColor(contentStream);
//reset color of text to black

        contentStream.showText(text3);
//adding the name of test three
        //setResultColor(contentStream, CommandTransferTestResult);
//set color of text based on result of test
        //contentStream.showText(result3);
//adding the result of test three
        resetResultColor(contentStream);
//reset color of text to black

        contentStream.showText(text4);
//adding the name of test four
        //setResultColor(contentStream, motorizedLeverTestResult);
//set color of text based on result of test
        //contentStream.showText(result4);
//adding the result of test four
        resetResultColor(contentStream);
//reset color of text to black
```

```
        contentStream.newLine();
//add a new line
        contentStream.showText(textUnderUnderUnderHeadline2);
//add a under under under headline
        contentStream.newLine();
//add a new line
        contentStream.showText(text5);
//adding the name of test five
        setResultColor(contentStream, rpmControlTestResult);
//set color of text based on result of test
        contentStream.showText(result5);
//adding the result of test five
        resetResultColor(contentStream);
//reset color of text to black

        contentStream.showText(text6);
//adding the name of test six
        //setResultColor(contentStream, backupControlTestResult);
//set color of text based on result of test
        //contentStream.showText(result6);
//adding the result of test six
        resetResultColor(contentStream);
//reset color of text to black

        contentStream.newLine();
//add a new line
        contentStream.showText(textUnderUnderUnderHeadline3);
//add a under under under headline
        contentStream.newLine();
//add a new line
        contentStream.showText(text7);
//adding the name of test seven
        setResultColor(contentStream,
remoteStartStopThrusterMotorTestResult);           //set color of text
based on result of test
        contentStream.showText(result7);
//adding the result of test seven
        resetResultColor(contentStream);
//reset color of text to black

        contentStream.showText(text8);
//adding the name of test eight
        setResultColor(contentStream,
remoteStartStopThrusterServoPumpsTestResult);      //set color of text
based on result of test
        contentStream.showText(result8);
//adding the result of test eight
        resetResultColor(contentStream);
//reset color of text to black

        contentStream.showText(text9);
//adding the name of test nine
        setResultColor(contentStream, resetDriveTestResult);
//set color of text based on result of test
        contentStream.showText(result9);
//adding the result of test nine
```

```
        resetResultColor(contentStream);
//reset color of text to black

        contentStream.newLine();
//add a new line
        contentStream.showText(textUnderUnderUnderHeadline4);
//add a under under under headline
        contentStream.newLine();
//add a new line
        contentStream.showText(text10);
//adding the name of test seven
        setResultColor(contentStream, dpInterfaceTestResult);
//set color of text based on result of test
        contentStream.showText(result10);
//adding the result of test seven
        resetResultColor(contentStream);
//reset color of text to black

        contentStream.showText(text11);
//adding the name of test seven
        setResultColor(contentStream, joystickTestResult);
//set color of text based on result of test
        contentStream.showText(result11);
//adding the result of test seven
        resetResultColor(contentStream);
//reset color of text to black

        contentStream.endText();
//ending the content stream

        System.out.println("Content added");
//prints out message to terminal

        contentStream.close();
//closing the content stream

        doc.save(new File("Fat-test-result.pdf"));
//saving the document as a new file

        doc.close();
//closing the document
    }

    /**
     * setting the color of a result of a test
     * @param contentStream instance of PDPageContentStream
     * @param value boolean value of true or false
     * @throws IOException throw if something goes wrong
     */
    public static void setResultColor(PDPageContentStream contentStream,
boolean value) throws IOException {
        if (value) {
//checks if the test is OK
            contentStream.setNonStrokingColor(new Color(0, 255, 0));
//if test is OK, color is set to green
        }
        else {
```

```
            contentStream.setNonStrokingColor(new Color(255, 0, 0));
//if test is not OK, color is set to red
        }
    }

    /**
     * resetting the color of text
     * @param contentStream instance of PDPageContentStream
     * @throws IOException throw if something goes wrong
     */
    public static void resetResultColor(PDPageContentStream contentStream)
throws IOException {
        contentStream.setNonStrokingColor(new Color(0, 0, 0));
//resets the color of text to black
        contentStream.newLine();
//add a new line
    }
}
```

N.4 Class KeyStoreLoader

```
package main.java;

import java.io.InputStream;
import java.io.OutputStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.security.Key;
import java.security.KeyPair;
import java.security.KeyStore;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.cert.X509Certificate;
import java.util.Arrays;
import java.util.regex.Pattern;
import org.eclipse.milo.opcua.stack.core.util.SelfSignedCertificateBuilder;
import
org.eclipse.milo.opcua.stack.core.util.SelfSignedCertificateGenerator;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * class for getting security and certificates for connection
 */
class KeyStoreLoader {

    private static final Pattern IP_ADDR_PATTERN = Pattern.compile(
            "^(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.){3}([01]?\\d\\d?|2[0-
4]\\d|25[0-5])$");

    private static final String CLIENT_ALIAS = "client-ai";
    private static final char[] PASSWORD = "password".toCharArray();

    private final Logger logger = LoggerFactory.getLogger(getClass());

    private X509Certificate[] clientCertificateChain;
    private X509Certificate clientCertificate;
    private KeyPair clientKeyPair;

    /**
     * getting the security for the connection
     * @param baseDir security temporary directory
     * @return security for the connection
     * @throws Exception throw if something goes wrong
     */
    KeyStoreLoader load(Path baseDir) throws Exception {
        KeyStore keyStore = KeyStore.getInstance("PKCS12");

        Path serverKeyStore = baseDir.resolve("example-client.pfx");

        logger.info("Loading KeyStore at {}", serverKeyStore);

        if (!Files.exists(serverKeyStore)) {
            keyStore.load(null, PASSWORD);

            KeyPair keyPair =
SelfSignedCertificateGenerator.generateRsaKeyPair(2048);
```

```
            SelfSignedCertificateBuilder builder = new
SelfSignedCertificateBuilder(keyPair)
                    .setCommonName("Eclipse Milo Client")
                    .setOrganization("digitalpetri")
                    .setOrganizationalUnit("dev")
                    .setLocalityName("Folsom")
                    .setStateName("CA")
                    .setCountryCode("US")
                    .setApplicationUri("urn:eclipse:milo:examples:client")
                    .addDnsName("br-automation")
                    .addIpAddress("192.168.1.110");

            X509Certificate certificate = builder.build();

            keyStore.setKeyEntry(CLIENT_ALIAS, keyPair.getPrivate(),
PASSWORD, new X509Certificate[]{certificate});
            try (OutputStream out = Files.newOutputStream(serverKeyStore))
{
                keyStore.store(out, PASSWORD);
            }
        } else {
            try (InputStream in = Files.newInputStream(serverKeyStore)) {
                keyStore.load(in, PASSWORD);
            }
        }

        Key clientPrivateKey = keyStore.getKey(CLIENT_ALIAS, PASSWORD);
        if (clientPrivateKey instanceof PrivateKey) {
            clientCertificate = (X509Certificate)
keyStore.getCertificate(CLIENT_ALIAS);

            clientCertificateChain =
Arrays.stream(keyStore.getCertificateChain(CLIENT_ALIAS))
                    .map(X509Certificate.class::cast)
                    .toArray(X509Certificate[]::new);

            PublicKey serverPublicKey = clientCertificate.getPublicKey();
            clientKeyPair = new KeyPair(serverPublicKey, (PrivateKey)
clientPrivateKey);
        }

        return this;
    }

    /**
     * getting certificate
     * @return certificate
     */
    X509Certificate getClientCertificate() {
        return clientCertificate;
    }

    /**
     * getting certificate chain
     * @return the certificate chain
     */
    public X509Certificate[] getClientCertificateChain() {
```

```
        return clientCertificateChain;
    }

    /**
     * getting key pair
     * @return key pair
     */
    KeyPair getClientKeyPair() {
        return clientKeyPair;
    }
}
```

N.5 Class MiloClient

```
package main.java;

import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.eclipse.milo.opcua.sdk.client.OpcUaClient;
import org.eclipse.milo.opcua.sdk.client.nodes.UaVariableNode;
import
org.eclipse.milo.opcua.stack.client.security.DefaultClientCertificateValida
tor;
import org.eclipse.milo.opcua.stack.core.UaException;
import org.eclipse.milo.opcua.stack.core.security.DefaultTrustListManager;
import org.eclipse.milo.opcua.stack.core.types.builtin.*;
import java.io.File;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.security.Security;
import java.util.concurrent.ExecutionException;

import static
org.eclipse.milo.opcua.stack.core.types.builtin.unsigned.Unsigned.uint;

/**
 * class for creating a client of milo
 */
public class MiloClient implements Connection {

    static {
        Security.addProvider(new BouncyCastleProvider());
//required for SecurityPolicy.Aes256_Sha256_RsaPss
    }

    public String endpoint;
//creates a endpoint for connection to server
    private OpcUaClient client;
//creates a client of OpcUaClient
    Object decoded = null;
    Variant variant = null;
    private DefaultTrustListManager trustListManager;

    /**
     * creating a client of milo
     */
    public MiloClient() {

        try {
            Path securityTempDir =
Paths.get(System.getProperty("java.io.tmpdir"), "client", "security");
            Files.createDirectories(securityTempDir);
            if (!Files.exists(securityTempDir)) {
                throw new Exception("unable to create security dir: " +
securityTempDir);
            }

            File pkiDir = securityTempDir.resolve("pki").toFile();
            trustListManager = new DefaultTrustListManager(pkiDir);
            DefaultClientCertificateValidator certificateValidator = new
DefaultClientCertificateValidator(trustListManager);
```

```
            KeyStoreLoader loader = new
KeyStoreLoader().load(securityTempDir);

            client = OpcUaClient.create(
//creates a client
                    ClientExample.getEndpointUrl(),
                    endpoints ->
                            endpoints.stream()
                                    .filter(ClientExample.endpointFilter())
                                    .findFirst(),
                    configBuilder ->
                            configBuilder

.setApplicationName(LocalizedText.english("eclipse milo opc-ua client"))

.setApplicationUri("urn:eclipse:milo:examples:client")
                                    .setKeyPair(loader.getClientKeyPair())

.setCertificate(loader.getClientCertificate())

.setCertificateChain(loader.getClientCertificateChain())

.setCertificateValidator(certificateValidator)

.setIdentityProvider(ClientExample.getIdentityProvider())
                                    .setRequestTimeout(uint(5000))
                                    .build());

            client.connect().get();
//connects the client and server
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * stopping the connection
     */
    public void stop() {
        try {
            client.disconnect().get();
//disconnects the client from the server
        } catch (InterruptedException e) {
            e.printStackTrace();
        } catch (ExecutionException e) {
            e.printStackTrace();
        }
    }

    /**
     * getting the value of a variable
     * @param nodeId the variable to collect value of
     * @return the value of the requested variable
     */
    public Object getVariable(NodeId nodeId) {
        try {
```

```
            DataValue value = null;
//creating a value of DataValue
            UaVariableNode node =
client.getAddressSpace().getVariableNode(nodeId); //gets the variable to be
read
            value = node.readValue();
//reads the variable
            variant = value.getValue();
//gets the value of the read variable
            decoded = (Object) variant.getValue();
//converts the value to an Object
        } catch (UaException e) {
            e.printStackTrace();
        }
        return decoded;
    }

    /**
     * getting value of rpmControl
     * @return the value of rpmControl
     */
    public int getRpmControlValue() {
        NodeId objectId = NodeId.parse("ns=6;s=::Program:A_IN");
//node ID copied via UaExpert
        Object value = getVariable(objectId);
//read value of variable
        int calculatedValue = ((Short) value).intValue();
//convert value of variable to int
        System.out.println("RPM: " + calculatedValue);
//print out value of variable
        return calculatedValue;
//value of the variable to be read
    }

    /**
     * getting state of thrusterMotor
     * @return the state of thrusterMotor
     */
    public boolean getThrusterMotorValue() {
        NodeId objectId = NodeId.parse("ns=6;s=::Program:thrusterRunning");
//node ID copied via UaExpert
        Object value = getVariable(objectId);
//read value of variable
        boolean calculatedValue = ((Boolean) value).booleanValue();
//convert value of variable to boolean
        System.out.println("ThrusterRunning: " + calculatedValue);
//print out value of variable
        return calculatedValue;
//the variable to be read
    }

    /**
     * getting state of thrusterMotorStart
     * @return the state of thrusterMotorStart
     */
    public boolean getThrusterMotorStartValue() {
```

```
        NodeId objectId = NodeId.parse("ns=6;s=::Program:startThruster");
//node ID copied via UaExpert
        Object value = getVariable(objectId);
//read value of variable
        boolean calculatedValue = ((Boolean) value).booleanValue();
//convert value of variable to boolean
        System.out.println("StartThruster: " + calculatedValue);
//print out value of variable
        return calculatedValue;
//the variable to be read
    }

    /**
     * getting state of thrusterMotorStop
     * @return the state of thrusterMotorStop
     */
    public boolean getThrusterMotorStopValue() {
        NodeId objectId = NodeId.parse("ns=6;s=::Program:stopThruster");
//Node ID copied via UaExpert
        Object value = getVariable(objectId);
//read value of variable
        boolean calculatedValue = ((Boolean) value).booleanValue();
//convert value of variable to boolean
        System.out.println("StopThruster: " + calculatedValue);
//print out value of variable
        return calculatedValue;
//the variable to be read
    }

    /**
     * getting state of thrusterPumps
     * @return the state of thrusterPumps
     */
    public boolean getThrusterPumpsValue() {
        NodeId objectId = NodeId.parse("ns=6;s=::Program:pumpRunning");
//node ID copied via UaExpert
        Object value = getVariable(objectId);
//read value of variable
        boolean calculatedValue = ((Boolean) value).booleanValue();
//convert value of variable to boolean
        System.out.println("PumpRunning: " + calculatedValue);
//print out value of variable
        return calculatedValue;
//the variable to be read
    }

    /**
     * getting state of thrusterPumpsStart
     * @return the state of thrusterPumpsStart
     */
    public boolean getThrusterPumpsStartValue() {
        NodeId objectId = NodeId.parse("ns=6;s=::Program:startPump");
//node ID copied via UaExpert
        Object value = getVariable(objectId);
//read value of variable
        boolean calculatedValue = ((Boolean) value).booleanValue();
//convert value of variable to boolean
```

```
        System.out.println("StartPump: " + calculatedValue);
//print out value of variable
        return calculatedValue;
//the variable to be read
    }

    /**
     * getting state of thrusterPumpsStop
     * @return the state of thrusterPumpsStop
     */
    public boolean getThrusterPumpsStopValue() {
        NodeId objectId = NodeId.parse("ns=6;s=::Program:stopPump");
//node ID copied via UaExpert
        Object value = getVariable(objectId);
//read value of variable
        boolean calculatedValue = ((Boolean) value).booleanValue();
//convert value of variable to boolean
        System.out.println("StopPump: " + calculatedValue);
//print out value of variable
        return calculatedValue;
//the variable to be read
    }

    /**
     * getting state of resetDrive
     * @return the state of resetDrive
     */
    public boolean getResetDriveValue() {
        NodeId objectId = NodeId.parse("ns=6;s=::Program:driveReset");
//node ID copied via UaExpert
        Object value = getVariable(objectId);
//read value of variable
        boolean calculatedValue = ((Boolean) value).booleanValue();
//convert value of variable to boolean
        System.out.println("DriveReset: " + calculatedValue);
//print out value of variable
        return calculatedValue;
//the variable to be read
    }

    /**
     * getting value of dpInterface
     * @return the value of dpInterface
     */
    public int getDpInterfaceValue() {
        NodeId objectId = NodeId.parse("ns=6;s=::Program:DP_RPM_Feedback");
//node ID copied via UaExpert
        Object value = getVariable(objectId);
//read value of variable
        int calculatedValue = ((Short) value).intValue();
//convert value of variable to int
        System.out.println("DpInterface: " + calculatedValue);
//print out value of variable
        return calculatedValue;
//value of the variable to be read
    }
```

```
    /**
     * getting state of dpInterfaceStart
     * @return the state of dpInterfaceStart
     */
    public boolean getDpInterfaceStartValue() {
        NodeId objectId = NodeId.parse("ns=6;s=::Program:DPReady");
//node ID copied via UaExpert
        Object value = getVariable(objectId);
//read value of variable
        boolean calculatedValue = ((Boolean) value).booleanValue();
//convert value of variable to boolean
        System.out.println("StartDpInterface: " + calculatedValue);
//print out value of variable
        return calculatedValue;
//the variable to be read
    }

    /**
     * getting value of joystick
     * @return the value of joystick
     */
    public int getJoystickValue() {
        NodeId objectId =
NodeId.parse("ns=6;s=::Program:Joystick_RPM_Feedback");//node ID copied via
UaExpert
        Object value = getVariable(objectId);
//read value of variable
        int calculatedValue = ((Short) value).intValue();
//convert value of variable to int
        System.out.println("Joystick: " + calculatedValue);
//print out value of variable
        return calculatedValue;
//value of the variable to be read
    }

    /**
     * getting state of joystickStart
     * @return the state of joystickStart
     */
    public boolean getJoystickStartValue() {
        NodeId objectId = NodeId.parse("ns=6;s=::Program:JoystickReady");
//node ID copied via UaExpert
        Object value = getVariable(objectId);
//read value of variable
        boolean calculatedValue = ((Boolean) value).booleanValue();
//convert value of variable to boolean
        System.out.println("StartJoystick: " + calculatedValue);
//print out value of variable
        return calculatedValue;
//the variable to be read
    }
}
```

N.6 Class RunTestsMilo

```java
package main.java;

/**
 * class for creating a connection between client milo and B&R server
 */
public class RunTestsMilo {

    /**
     * starting the code
     * @param args contains the supplied command-line arguments as an array
of String objects
     */
    public static void main(String[] args) {
        runTestsMilo();
    }

    /**
     * running tests using milo
     */
    public static void runTestsMilo() {
        MiloClient miloClient = new MiloClient();           //creates an
instance of MiloClient

        TestRunner testRunner = new TestRunner(miloClient); //creates an
instance of TestRunner
        testRunner.runTests();                              //runs tests
with milo connection

        miloClient.stop();                                  //stops the
milo client
    }
}
```

N.7 Class RunTestsVirtual

```java
package main.java;

/**
 * class for creating a virtual connection
 */
public class RunTestsVirtual {

    /**
     * running tests using a virtual connection
     * @param args contains the supplied command-line arguments as an array
of String objects
     */
    public static void main(String[] args) {
        TestRunner testRunner = new TestRunner(new VirtualConnection());
//creates an instance of TestRunner
        testRunner.runTests();
//runs tests with a virtual connection
    }
}
```

N.8 Class TestRunner

```
package main.java;

import java.io.IOException;

/**
 * class for running tests
 */
public class TestRunner {
    public boolean thrusterMotorStartIsFinished;
//variable for first part of thruster motor test
    public boolean thrusterMotorStopIsFinished;
//variable for second part of thruster motor test
    public boolean thrusterPumpsStartIsFinished;
//variable for first part of thruster pumps test
    public boolean thrusterPumpsStopIsFinished;
//variable for second part of thruster pumps test

    public boolean rpmControlTestIsFinished;
//variable for checking if test five is finished
    public boolean remoteStartStopThrusterMotorTestIsFinished;
//variable for checking if test seven is finished
    public boolean remoteStartStopThrusterServoPumpsTestIsFinished;
//variable for checking if test eight is finished
    public boolean resetDriveTestIsFinished;
//variable for checking if test nine is finished
    public boolean dpInterfaceTestIsFinished;
//variable for checking if test ten is finished
    public boolean joystickTestIsFinished;
//variable for checking if test eleven is finished

    public boolean rpmControlTestResult;                            //saves
variable as result of the fifth test
    public boolean remoteStartStopThrusterMotorTestResult;          //saves
variable as result of the seventh test
    public boolean remoteStartStopThrusterServoPumpsTestResult;     //saves
variable as result of the eighth test
    public boolean resetDriveTestResult;                            //saves
variable as result of the ninth test
    public boolean dpInterfaceTestResult;                           //saves
variable as result of the tenth test
    public boolean joystickTestResult;                              //saves
variable as result of the eleventh test

    public int numberOfTestsFinished = 0;
//variable for number of tests finished
    public Connection client;
//creates an instance of Connection

    public TestRunner(Connection client) {
        this.client = client;
    }

    /**
     * running of the tests
     */
    public void runTests() {
```

```
        CreatePDF createPDF = new CreatePDF();
//creates an instance of class CreatePDF
        java.io.InputStreamReader reader = new
java.io.InputStreamReader(System.in);
//creates an instance of a InputStreamReader
        boolean keepRunning = true;
//variable for keep running the code

        while (keepRunning) {
//loop for running the tests continually
            try {
                Thread.sleep(100);
//wait for 100 milliseconds
                System.out.println("Press enter to quit");
            } catch (InterruptedException ex) {
                System.out.println("InterruptedException...");
            }
            try {
                if (reader.ready()) {
                    int readKey = reader.read();
//read the keyboard input
                    if (readKey == 10) {
//checks if enter is pressed
                        keepRunning = false;
//if enter is pressed, the loop will stop
                        System.out.println("Quitting..");
                    }
                } else if (numberOfTestsFinished == 6) {
//checks if number of finished tests is 3
                    keepRunning = false;
//if 3 tests are finished, the loop will stop
                    System.out.println("Quitting..");
                }
                if (!rpmControlTestIsFinished) {
//check if test five is finished
                    rpmControlTestResult = rpmControlTest();
//saves variable as result of the fifth test
                }
                if (!remoteStartStopThrusterMotorTestIsFinished) {
//check if test seven is finished
                    remoteStartStopThrusterMotorTestResult =
remoteStartStopThrusterMotorTest();                         //saves variable
as result of the seventh test
                }
                if (!remoteStartStopThrusterServoPumpsTestIsFinished) {
//check if test eight is finished
                    remoteStartStopThrusterServoPumpsTestResult =
remoteStartStopThrusterServoPumpsTest();              //saves variable as
result of the eighth test
                }
                if (!resetDriveTestIsFinished) {
//check if test nine is finished
                    resetDriveTestResult = resetDriveTest();
//saves variable as result of the ninth test
                }
                if (!dpInterfaceTestIsFinished) {
//check if test ten is finished
```

```
                    dpInterfaceTestResult = dpInterfaceTest();
//saves variable as result of the tenth test
                }
                if (!joystickTestIsFinished) {
//check if test eleven is finished
                    joystickTestResult = joyStickTest();
//saves variable as result of the eleventh test
                }
                testsFinished();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        try {
            createPDF.createPDF(rpmControlTestResult,
remoteStartStopThrusterMotorTestResult,
remoteStartStopThrusterServoPumpsTestResult, resetDriveTestResult,
dpInterfaceTestResult, joystickTestResult); //calls the method createPDF in
class CreatePDF with results of tests
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * test five
     * @return the result of the test
     */
    public boolean rpmControlTest() {
        int rpmControlValue = client.getRpmControlValue();
//get the current value of rpmControl in B&R
        boolean testOK = false;
//Create a variable for checking test

        int highestValue = rpmControlValue + 100;
//make variable for high rpmControl value
        int lowestValue = rpmControlValue - 100;
//make variable for low rpmControl value

        try {
            Thread.sleep(100);
//wait for 100 milliseconds
        } catch (InterruptedException ex) {
            System.out.println("InterruptedException...");
        }

        if (client.getRpmControlValue() < lowestValue ||
client.getRpmControlValue() > highestValue) { //check if value has changed
to less than the low value or more than the high value
            testOK = true;
//if value has changed, test is correct
            numberOfTestsFinished++;
//if value has changed, number of finished tests is increased
            rpmControlTestIsFinished = true;
//if value has changed, this test is finished
        }
```

```
        return testOK;
    }

    /**
     * test seven
     * @return the result of the test
     */
    public boolean remoteStartStopThrusterMotorTest() {
        boolean thrusterMotorIsOn = client.getThrusterMotorValue();
//get the current state of thrusterMotor in B&R
        boolean thrusterMotorStartIsOn =
client.getThrusterMotorStartValue();  //get the current state of
thrusterMotorStart in B&R
        boolean thrusterMotorStopIsOn = client.getThrusterMotorStopValue();
//get the current state of thrusterMotorStop in B&R
        boolean testOK = false;
//create a variable for checking test
        boolean start = false;
//create a variable for checking thrusterMotorStart
        boolean stop = false;
//create a variable for checking thrusterMotorStop

        if (thrusterMotorStartIsOn) {
//checks if thrusterMotorStart is on
            start = true;
//if thrusterMotorStart is on, start is set to true
        }
        try {
            Thread.sleep(100);
//wait for 100 milliseconds
        } catch (InterruptedException ex) {
            System.out.println("InterruptedException...");
        }
        if (start && !thrusterMotorStartIsFinished) {
//check if start is on and part test is not finished
            if (thrusterMotorIsOn) {
//check if thrusterMotor is on
                thrusterMotorStartIsFinished = true;
//if thrusterMotor is on, part test is finished
            }
        }

        if (thrusterMotorStopIsOn) {
//checks if thrusterMotorStop is on
            stop = true;
//if thrusterMotorStop is on, stop is set to true
        }
        try {
            Thread.sleep(100);
//wait for 100 milliseconds
        } catch (InterruptedException ex) {
            System.out.println("InterruptedException...");
        }
        if (stop && !thrusterMotorStopIsFinished) {
//check if stop is on and part test is not finished
            if (!thrusterMotorIsOn) {
//check if thrusterMotor is on
```

```
                thrusterMotorStopIsFinished = true;
//if thrusterMotor is on, part test is finished
            }
        }

        if (thrusterMotorStartIsFinished && thrusterMotorStopIsFinished) {
//checks if both part tests are finished
            testOK = true;
//if both tests are finished, test is correct
            numberOfTestsFinished++;
//if both tests are finished, number of finished tests is increased
            remoteStartStopThrusterMotorTestIsFinished = true;
//if both tests are finished, this test is finished

        }
        return testOK;
    }

    /**
     * test eight
     * @return the result of the test
     */
    public boolean remoteStartStopThrusterServoPumpsTest() {
        boolean thrusterPumpsIsOn = client.getThrusterPumpsValue();
//get the current state of thrusterPumps in B&R
        boolean thrusterPumpsStartIsOn =
client.getThrusterPumpsStartValue();   //get the current state of
thrusterPumpsStart in B&R
        boolean thrusterPumpsStopIsOn = client.getThrusterPumpsStopValue();
//get the current state of thrusterPumpsStop in B&R
        boolean testOK = false;
//create a variable for checking test
        boolean start = false;
//create a variable for checking thrusterPumpsStart
        boolean stop = false;
//create a variable for checking thrusterPumpsStop

        if (thrusterPumpsStartIsOn) {
//checks if thrusterPumpsStart is on
            start = true;
//if thrusterPumpsStart is on, start is set to true
        }
        try {
            Thread.sleep(100);
//wait for 100 milliseconds
        } catch (InterruptedException ex) {
            System.out.println("InterruptedException...");
        }
        if (start && !thrusterPumpsStartIsFinished) {
//check if start is on and part test is not finished
            if (thrusterPumpsIsOn) {
//check if thrusterPumps is on
                thrusterPumpsStartIsFinished = true;
//if thrusterPumps is on, part test is finished
            }
        }
```

```
        if (thrusterPumpsStopIsOn) {
//checks if pumpsStop is on
            stop = true;
//if pumpsStop is on, stop is set to true
        }
        try {
            Thread.sleep(100);
//wait for 100 milliseconds
        } catch (InterruptedException ex) {
            System.out.println("InterruptedException...");
        }
        if (stop && !thrusterPumpsStopIsFinished) {
//check if stop is on and part test is not finished
            if (!thrusterPumpsIsOn) {
//check if thrusterPumps is on
                thrusterPumpsStopIsFinished = true;
//if thrusterPumps is on, part test is finished
            }
        }

        if (thrusterPumpsStartIsFinished && thrusterPumpsStopIsFinished) {
//checks if both part tests are finished
            testOK = true;
//if both tests are finished, test is correct
            numberOfTestsFinished++;
//if both tests are finished, number of finished tests is increased
            remoteStartStopThrusterServoPumpsTestIsFinished = true;
//if both tests are finished, this test is finished

        }
        return testOK;
    }

    /**
     * test nine
     * @return the result of the test
     */
    public boolean resetDriveTest() {
        boolean resetDriveIsOn = client.getResetDriveValue();
//get the current state of driveReset in B&R
        boolean testOK = false;
//create a variable for checking test

        if (resetDriveIsOn) {
//check if resetDrive is on
            testOK = true;
//if resetDrive is on, test is correct
            numberOfTestsFinished++;
//if resetDrive is on, number of finished tests is increased
            resetDriveTestIsFinished = true;
//if resetDrive is on, this test is finished
        }
        return testOK;
    }

    /**
     * test ten
```

```
     * @return the result of the test
     */
    public boolean dpInterfaceTest() {
        int dpInterfaceValue = client.getDpInterfaceValue();
//get the current value of dpInterface in B&R
        boolean dpInterfaceIsOn = client.getDpInterfaceStartValue();
//get the current state of dpInterfaceStart in B&R
        boolean start = false;
//create a variable for checking dpInterfaceStart
        boolean testOK = false;
//create a variable for checking test

        if (dpInterfaceIsOn) {
//check if dpInterface is on
            start = true;
//if dpInterface is on, start is set to true
        }
        if (start) {
            int highestValue = dpInterfaceValue + 100;
//if dpInterface is on, make variable for high dpInterface value
            int lowestValue = dpInterfaceValue - 100;
//if dpInterface is on, make variable for low dpInterface value

            try {
                Thread.sleep(100);
//wait for 100 milliseconds
            } catch (InterruptedException ex) {
                System.out.println("InterruptedException...");
            }

            if (client.getDpInterfaceValue() < lowestValue ||
client.getDpInterfaceValue() > highestValue) { //check if value has changed
to less than the low value or more than the high value
                testOK = true;
//if value has changed, test is correct
                numberOfTestsFinished++;
//if value has changed, number of finished tests is increased
                dpInterfaceTestIsFinished = true;
//if value has changed, this test is finished
            }
        }
        return testOK;
    }

    /**
     * test eleven
     * @return the result of the test
     */
    public boolean joyStickTest() {
        int joystickValue = client.getJoystickValue();
//get the current value of joystick in B&R
        boolean joystickIsOn = client.getJoystickStartValue();
//get the current state of joystickStart in B&R
        boolean start = false;
//create a variable for checking joystickStart
        boolean testOK = false;
//create a variable for checking test
```

```
        if (joystickIsOn) {
//check if joystick is on
            start = true;
//if joystick is on, start is set to true
            }
        if (start) {
            int highestValue = joystickValue + 100;
//if joystick is on, make variable for high joystick value
            int lowestValue = joystickValue - 100;
//if joystick is on, make variable for low joystick value

            try {
                Thread.sleep(100);
//wait for 100 milliseconds
            } catch (InterruptedException ex) {
                System.out.println("InterruptedException...");
            }

            if (client.getJoystickValue() < lowestValue ||
client.getJoystickValue() > highestValue) {     //check if value has
changed to less than the low value or more than the high value
                testOK = true;
//if value has changed, test is correct
                numberOfTestsFinished++;
//if value has changed, number of finished tests is increased
                joystickTestIsFinished = true;
//if value has changed, this test is finished
            }
        }
        return testOK;
    }

    /**
     * printing result of tests to system panel
     */
    public void testsFinished() {
        System.out.println("RPM Control Test: " +
rpmControlTestIsFinished);
//print out test finished for rpmControl
        System.out.println("Remote Start/Stop Thruster Motor Test: " +
remoteStartStopThrusterMotorTestIsFinished);            //print out test
finished for thruster motor
        System.out.println("Remote Start/Stop Thruster Servo Pumps Test: "
+ remoteStartStopThrusterServoPumpsTestIsFinished); //print out test
finished for thruster pumps
        System.out.println("Reset Drive Test: " +
resetDriveTestIsFinished);
//print out test finished for reset drive
        System.out.println("DpInterface Test: " +
dpInterfaceTestIsFinished);
//print out test finished for dpInterface
        System.out.println("Joystick Test: " + joystickTestIsFinished);
//print out test finished for joystick
    }
}
```

N.9 Class VirtualConnection

```
package main.java;

/**
 * class for creating a virtual connection
 */
public class VirtualConnection implements Connection {
    public int rpmControlValue;          //creates a variable for test five
    public boolean thrusterMotorOn;      //creates a variable for test
seven
    public boolean thrusterMotorStartOn; //creates a variable for test
seven
    public boolean thrusterMotorStopOn;  //creates a variable for test
seven
    public boolean thrusterPumpsOn;      //creates a variable for test
eight
    public boolean thrusterPumpsStartOn; //creates a variable for test
eight
    public boolean thrusterPumpsStopOn;  //creates a variable for test
eight
    public boolean resetDriveOn;         //creates a variable for test nine
    public int dpInterfaceValue;         //creates a variable for test ten
    public boolean dpInterfaceOn;        //creates a variable for test ten
    public int joystickValue;            //creates a variable for test
eleven
    public boolean joystickOn;           //creates a variable for test
eleven

    /**
     * getting value of rpmControl
     * @return the value of rpmControl
     */
    public int getRpmControlValue() {
        return rpmControlValue;
    }

    /**
     * getting state of thrusterMotor
     * @return the state of thrusterMotor
     */
    public boolean getThrusterMotorValue() {
        return thrusterMotorOn;
    }

    /**
     * getting state of thrusterMotorStop
     * @return the state of thrusterMotorStop
     */
    public boolean getThrusterMotorStartValue() {
        return thrusterMotorStartOn;
    }

    /**
     * getting state of thrusterMotorStop
     * @return the state of thrusterMotorStop
     */
    public boolean getThrusterMotorStopValue() {
        return thrusterMotorStopOn;
```

```
}

/**
 * getting state of thrusterPumps
 * @return the state of thrusterPumps
 */
public boolean getThrusterPumpsValue() {
    return thrusterPumpsOn;
}

/**
 * getting state of thrusterPumpsStart
 * @return the state of thrusterPumpsStart
 */
public boolean getThrusterPumpsStartValue() {
    return thrusterPumpsStartOn;
}

/**
 * getting state of thrusterPumpsStop
 * @return the state of thrusterPumpsStop
 */
public boolean getThrusterPumpsStopValue() {
    return thrusterPumpsStopOn;
}

/**
 * getting state of reset drive
 * @return the state of reset drive
 */
public boolean getResetDriveValue() {
    return resetDriveOn;
}

/**
 * getting value of dpInterface
 * @return the value of dpInterface
 */
public int getDpInterfaceValue() {
    return dpInterfaceValue;
}

/**
 * getting state of dpInterfaceStart
 * @return the state of dpInterfaceStart
 */
public boolean getDpInterfaceStartValue() {
    return dpInterfaceOn;
}

/**
 * getting value of joystick
 * @return the value of joystick
 */
public int getJoystickValue() {
    return joystickValue;
}
```

```
    /**
     * getting state of joystickStart
     * @return the state of joystickStart
     */
    public boolean getJoystickStartValue() {
        return joystickOn;
    }
}
```

N.10 Maven-file Pom

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>FAT-test-report</groupId>
    <artifactId>BachelorThesis</artifactId>
    <version>1.0-SNAPSHOT</version>

    <build>
        <sourceDirectory>src</sourceDirectory>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.3</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

    <dependencies>
        <dependency>
            <groupId>org.eclipse.milo</groupId>
            <artifactId>sdk-client</artifactId>
            <version>0.5.4</version>
        </dependency>

        <dependency>
            <groupId>org.apache.pdfbox</groupId>
            <artifactId>pdfbox</artifactId>
            <version>2.0.1</version>
        </dependency>

        <dependency>
            <groupId>org.apache.pdfbox</groupId>
            <artifactId>fontbox</artifactId>
            <version>2.0.0</version>
        </dependency>

        <dependency>
            <groupId>org.apache.pdfbox</groupId>
            <artifactId>jempbox</artifactId>
            <version>1.8.11</version>
        </dependency>

        <dependency>
            <groupId>org.apache.pdfbox</groupId>
            <artifactId>xmpbox</artifactId>
            <version>2.0.0</version>
        </dependency>

        <dependency>
```

```
            <groupId>org.apache.pdfbox</groupId>
            <artifactId>preflight</artifactId>
            <version>2.0.0</version>
        </dependency>

        <dependency>
            <groupId>org.apache.pdfbox</groupId>
            <artifactId>pdfbox-tools</artifactId>
            <version>2.0.0</version>
        </dependency>
    </dependencies>
</project>
```