



VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Rene Mikhael

Modifying UNIC2 HSR Communication to Use an External Switch

Technology and Communication
2019

TIIVISTELMÄ

Tekijä	Rene Mikhael
Opinnäytetyön nimi	UNIC2 HSR kommunikaation muuttaminen ulkoisen kytkimen soveltamiseksi
Vuosi	2019
Kieli	englanti
Sivumäärä	52
Ohjaaja	Jani Ahvonen

Tämä opinnäytetyö perustui Wärtsilä Oyj Abp:n Marine Solutions-sektorin Automaation ja Ohjauksen -osaston vaatimukseen UNIC2-kommunikaation muutosten soveltamiseksi ja analysoimiseksi. Nykyinen järjestelmä soveltaa sisäistä FPGA kytkintä, joka huolehtii HSR-toiminnallisuudesta. Tämän kytkimen sijaan käytettäisiin ulkoista kytkintä, jolla minimoitaisiin COM-10 moduulin FPGA-sirun järjestelmävaatimuksia.

2000-luvulla tarve kehittyneemmille tietoliikenneverkoille moottoreissa on lisääntynyt. Jokaisella merellä, maalla ja ilmassa olevalla ajoneuvolla on jonkinlainen ajoneuvoväylä, joka yhdistää ajoneuvon sisällä olevat osat. UNIC2 on Wärtsilän vastaus kehittyneemmästä automaatiojärjestelmästä. UNIC2-kommunikaatio käyttää HSR -protokollaa nopeaan tiedonsiirtoon ja redundanssiin, jotta saumaton vikasieto voidaan estää minkä tahansa verkkokomponentin vikaantumisen varalta.

Tutkimuksessa pyrittiin tutkimaan ja analysoimaan käytössä olevaa järjestelmää ja tekniikoita, sekä ratkaisun suunnittelua, toteutusta ja testausta. Muutosten soveltamiseksi COM-10 moduulin lähdekoodi vaati mittavan asiantuntemuksen, jotta pystyttiin selvittämään, mitä muutoksia lähdekoodi vaatisi, sekä ulkoinen kytkin HSR-konfiguraation.

Projektin lopputuloksena oli toimiva UNIC2-kommunikaatiojärjestelmä, jossa modifioitu COM-10 moduuli on kytkettynä ulkoiseen kytkimeen SAN-väylän kautta ja loput moduulit toimivat HSR-rengastopologiassa kytkimen kanssa. Tämän johdosta COM-10 1 moduulin FPGA-sirun käytössä olevat resurssit laskivat yli 40%, mikä mahdollisti tulevien FPGA-applikaatioiden kehittämisen.

ABSTRACT

Author	Rene Mikhael
Title	Modifying UNIC2 HSR Communication to Use an External Switch
Year	2019
Language	English
Pages	52
Name of Supervisor	Jani Ahvonen

This thesis was done based on the requirements provided by Wärtsilä Oyj Abp's Automation & Controls department of the Marine Solutions sector to modify and analyse the changes for UNIC2 communication. The current system uses an internal FPGA switch which takes care of the HSR functionality. Instead of using that, an external switch will be used to minimise the COM-10 1s FPGA chips system requirements.

During the 21st, century the need for more sophisticated communication networks in motors has increased. Every vehicle on the sea, land, and air has some sort of vehicle bus communication network that interconnects components inside a vehicle. UNIC2 is Wärtsilä's answer for a more sophisticated automation system. It uses HSR to provide high-speed communication as well as redundancy for seamless fail-over against failure of any network component.

The research was conducted with the aim of investigating and analysing the current system and technologies used as well as planning, implementing, and testing of the solution. To apply the changes, the COM-10's source code required extensive knowledge to find what needed to be changed, as well as HSR configuration to the external switch.

The result of the project was a functional UNIC2 communication system, where the modified COM-10 module is connected to an external switch via the SAN bus and the rest of the modules operate in a HSR ring topology with the switch. After disabling the internal switch on the COM-10 1 module, the resources on the FPGA chip dropped by more than 40%, giving a possibility for future FPGA application development.

CONTENTS

TIIVISTELMÄ

ABSTRACT

1	INTRODUCTION	11
1.1	Wärtsilä Oyj Abp	12
1.2	Automation & Controls.....	14
1.3	Wärtsilä Unified Controls (UNIC).....	16
1.4	Objectives and topic of choice	18
1.5	Implementation plan	19
2	THEORY AND BACKGROUND INFORMATION	20
2.1	High-availability Seamless Redundancy	21
2.2	Cisco IE-4000 Series Switch	22
2.3	FPGA Functional Description.....	23
2.4	Wärtsilä HSR	24
2.5	Wärtsilä Modular Application Platform	24
2.6	UNIC2 Communication	25
2.6.1	Hardware environment.....	25
2.6.2	Layer 2 framing.....	26
3	PRE-SOLUTION TESTING.....	27
3.1	Wireshark packet test when connected straight to PC.....	27
3.2	Implementing Cisco switch to capture WHSR messages	29
3.3	Testing UNIC-level communication between COM-10 modules	33
4	SOLUTION IMPLEMENTATION	37
4.1	Disabling the internal HSR switch from FPGA.....	38
4.2	WMAP software modifications	40
4.3	UNITool configuration and setup	41
5	TESTING ENVIRONMENT AND RESULTS	43
5.1	FPGA compilation results.....	43
5.2	Setting up the MiniRig testing environment	44
5.3	MiniRig test results	49
6	CONCLUSIONS AND FURTHER IMPROVEMENTS.....	54
6.1	Further improvements.....	54

REFERENCES..... 56

LIST OF ABBREVIATIONS

A&C	Automation & Controls
BMC	Best Master Clock
CAN	Controller Area Network
CCM-30	Cylinder Control Module
Cisco IOS	Internetwork Operating System
COM-10	Communication Module
CPS	Configuration Parameter Storage
CPU	Central Processing Unit
DANH	Doubly Attached Node
DF	Dual Fuel
EDL	Enhanced Diagnostics Log
ESM-20	Engine Safety Module
FIFO	First In First Out
FPGA	Field Programmable Gate Array
HAL	Hardware Abstraction Layer
HSR	High-availability Seamless Redundancy
IACS	Industrial Automation and Control Systems
IEEE 802.1Q	A networking standard that supports VLANs
IOM-20	Input/Output Module

IoT	Internet of Things
LDU-30	Local Display Unit
MCM	Main Control Module
MCU	Microcontroller Unit
MII	Media-independent interface
PTP	Precision Time Protocol
RedBox	Redundancy Box
rootfs	Root filesystem
RT	Real-Time
SAN	Singly Attached Node
SME	Single Main Engine
TCP/IP	Transmission Control Protocol / Internet Protocol
UNIC	Wärtsilä Unified Controls
UNITool	Wärtsilä engine software configuration tool
VDAN	Virtually Doubly Attached Node
VLAN	Virtual LAN
WE-CAN+	Wärtsilä Extended CAN plus
WHSR	Wärtsilä High-availability Seamless Redundancy
WMAP	Wärtsilä Modular Application Platform
WSDE	Wärtsilä Software Development Environment

LIST OF FIGURES AND TABLES

Figure 1 2019 Board of Management organisation chart	p. 14
Figure 2 2019 Automation & Controls organisation chart	p. 15
Figure 3 UNIC general layout /8/	p. 17
Figure 4 2 nd generation UNIC modules /8/	p. 18
Figure 5 Engineering design plan	p. 19
Figure 6 Objective of the thesis and target setup	p. 20
Figure 7 Example of HSR ring carrying Unicast traffic /11/	p. 21
Figure 8 Cisco IE-4000-8T4G-E switch used in the thesis /14/	p. 22
Figure 9 COM-10 and CCM-30 inputs and outputs	p. 23
Figure 10 UNIC2 Ethernet interfaces	p. 26
Figure 11 Layer 2 frame format	p. 26
Figure 12 COM-10 Layout. Left is Ethernet HSR, right is Ethernet EXT	p. 27
Figure 13 First test setup	p. 28
Figure 14 Modified input and output	p. 29
Figure 15 WHSR debugging messages	p. 29
Figure 16 Wireshark capture	p. 29
Figure 17 VLAN tagging illustrated /15/	p. 30
Figure 18 Second test setup	p. 31
Figure 19 g1/1 and f1/5 side by side comparison	p. 31

Figure 20 Cisco switch configuration commands	p. 32
Figure 21 g1/1 and f1/5 comparison after configuration	p. 32
Figure 22 Wireshark capture after configuration	p. 32
Figure 23 Third test setup	p. 33
Figure 24 Cisco switch HSR configuration commands	p. 34
Figure 25 Ethernet LAN tapper used in the project	p. 34
Figure 26 Wireshark capture on the third test	p. 35
Figure 27 Enhanced Diagnostics Log in this test setup	p. 35
Figure 28 Measured values and status of the engine	p. 36
Figure 29 UNIC2 communication interfaces	p. 37
Figure 30 WMAP stream modification snippet	p. 40
Figure 31 HAL stream interface enumeration	p. 40
Figure 32 UNITool logged on window	p. 41
Figure 33 Enabling WHSR from System Parameters	p. 42
Figure 34 Adding CAN routing over HSR	p. 42
Figure 35 Setting up Virtual Modules	p. 45
Figure 36 UNIC 2.0 Test Rig 1	p. 46
Figure 37 Final test setup MiniRig wiring diagram	p. 47
Figure 38 MiniRig Mapping	p. 47
Figure 39 Module ID configuration	p. 48

Figure 40 Application and Configuration download window	p. 48
Figure 41 W20V31SG system status after the download process	p. 49
Figure 42 EDL capture during final testing	p. 50
Figure 43 MiniRig Engine Control Panel window	p. 50
Figure 44 Ring failure detection on COM-10 2 module	p. 51
Figure 45 WHSR general information	p. 52
Figure 46 Module detection from COM-10 1	p. 52
Figure 47 Different values read from COM-10 module	p. 53
Figure 48 Different values read from COM-10 1 and COM-10 2	p. 53
Table 1 FPGA resource difference after the changes	p. 44

1 INTRODUCTION

During the 21st century, the need for more sophisticated communication networks in motors has increased. Every vehicle on the sea, land, and air has some sort of vehicle bus communication network that interconnects components inside a vehicle.

The main reasons for the development of vehicle network technology have been the advances made in the electronics industry in general and government regulations imposed, especially in the United States in order to make the automobiles more environmentally friendly.

With tight emission standards for automobiles, it became impossible to reach the required degree of control without the help of onboard computing devices. Onboard electronic devices have also contributed to vehicle performance, occupant comfort, ease of manufacture and cost-effectiveness.

An electronic control module gets its input from sensors e.g. speed, temperature, or pressure that it uses in its computation. Various actuators are used to enforce the actions determined by the module e.g. turn the cooling fan on, change gear, etc. The modules need to exchange data among themselves during normal operation of the vehicle. As an example, the engine needs to tell the transmission what the engine speed is, and transmission needs to tell other modules when a gear shift occurs. The need to exchange data quickly and reliably led to the development of the vehicle network, as a medium of data exchange.

The industry's answer to this was to create a central network in the vehicle. Modules could be plugged into the network and would be able to communicate with any other module that was installed on the network. This design was easier to manufacture, easier to maintain and provide the flexibility to add and remove options without affecting the entire vehicle's wiring architecture. Each module, a node on the vehicle network controls specific components related to its function and communicates with the other modules as necessary, using a standard protocol, over the vehicle network.

As a leading force in marine industry, Wärtsilä has been keeping pace with the development of automation technologies. Wärtsilä Unified Controls (UNIC) provides a reliable electronic control system for rugged industrial automation needs. /1/

1.1 Wärtsilä Oyj Abp

“Wärtsilä is a global leader in smart technologies and complete lifecycle solutions for the marine and energy markets. By emphasising sustainable innovation, total efficiency and data analytics, Wärtsilä maximises the environmental and economic performance of the vessels and power plants of its customers.” /2/

As of January 2019, Wärtsilä consists of two businesses; Marine Business and Energy Business. Services have been incorporated into Marine and Energy Businesses. /2/

“Wärtsilä Marine Solutions is a leading provider of ship machinery, propulsion and manoeuvring solutions. Wärtsilä supplies engines and generating sets, reduction gears, propulsion equipment, control systems and sealing solutions for all types of vessels and offshore applications. Wärtsilä commands a strong position in all main marine segments as a supplier of highly rated ship machinery and systems.” /2/.

The Marine Solutions offers /3/:

- Efficiency, reliability, flexibility, and environmental sustainability.
- Solutions are based on the customers’ needs and include innovative products and services.
- Experience, know-how, and dedication of the personnel, Wärtsilä is able to customise optimised solutions for the benefit of their clients around the world.
- Propulsion and generating sets for all vessels such as: cruise ships, yachts, navy, and offshore.
- Ship design for offshore, merchant, fishing, special vessels, and ferries.

Wärtsilä Energy Solutions supplies power plants for baseload, peaking and industrial self-generation purposes as well as for the oil and gas industry. The Energy Business sector is transitioning towards 100% renewable energy future. Wärtsilä offers engine-based flexible power plants – including liquid gas systems – hybrid solar power plants, and energy storage and integration solutions. /2/

The Energy Solutions offers: /4/:

- Energy efficiency – Multiple generating units are far more fuel efficient than traditional single-unit power stations.
- Fuel flexibility – Multi-fuel plants enable the continuous choice of most feasible fuel, including solutions for liquid and gaseous fuels as well as renewables.
- Dependability – Multi-unit configuration allows availability and reliability over 99%.
- Development and financial services unit supports clients with advice and assistance in deal structuring and financing.
- Project Management organisation plans, leads, manages and executes projects for the customer.

As Wärtsilä Services sector has been incorporated into Marine and Energy Solutions all the services, maintenance, and reconditioning solutions are handled through there.

The whole service sector consists of over 4,500 field service professionals in more than 160 locations. The focus lies on optimising operations and lifecycle performance of land-based power plants and ship installations. /5/

Wärtsilä Services offers: /6/

- Lifecycle solutions for marine range from matching maintenance for customers operation, optimising ship energy efficiency to guaranteeing asset performance.

- Lifecycle solutions for oil and gas with measurable and guaranteed benefits in a safe, reliable and environmentally friendly way. These include availability and performance, ensured efficiency and financial predictability, as well as a maximised lifetime for customers installation.
- Lifecycle solutions for energy installation promises customised long-term service agreements, including performance guarantees. Operations and management are available for power plant customers.

The Board of Management is chaired by the President & CEO, Mr. Jaakko Eskola. It considers strategic issues related to the Group and businesses, as well as investments, product policy, and the Group's structure and corporate steering systems. It also supervises the company's operations. He is assisted in his work by the Board of Management, as seen in **Figure 1**. The President & CEO and all members in the Board of Management are situated in Wärtsilä's Headquarters Helsinki Campus, Salmisaari. /7/

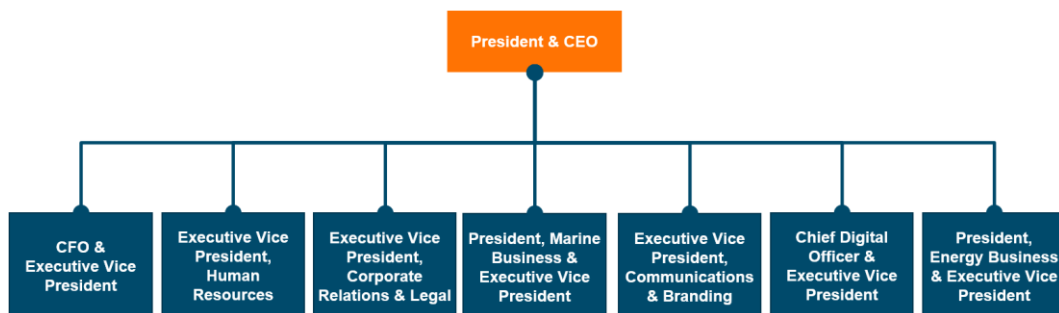


Figure 1 2019 Board of Management organisation chart

1.2 Automation & Controls

The Automation & Controls (A&C) department is situated in Vaasa, City factory. A&C's departments' main role is to convert customers' requirements and future needs to optimised performance solutions. A&C provides engine process and controls expertise to ensure the most competitive level of quality, performance, and cost throughout the lifetime of the product. The vision simply put is, connecting

engine processes and controls to provide engine performance beyond customer expectations.

As seen from **Figure 2** the A&C is divided into five different teams where each team has a specific area of expertise.

Automation & Controls team roles:

- Platform Hardware's main responsibility is the development and maintenance of all automation components for Wärtsilä 4-stroke engines.
- Platform Software team focuses on maintaining and further development for WMAP, WSDE, and UNITool.
- Control System Development team combines both the Hardware and Software products and incorporates them into specific projects. The main attribution of Control System Development team is to develop and implement the architecture of the engine control system and software.
- The Propulsion Control Systems team develops systems for controlling the vessels with levers, touch screen interfaces, displays, indicators, and modules to suit all the possible propulsion configurations of a modern ship.
- Customer Delivery team handles engine software configurations, external communication lists, engine wiring diagrams, documentations, and test instructions and test software.

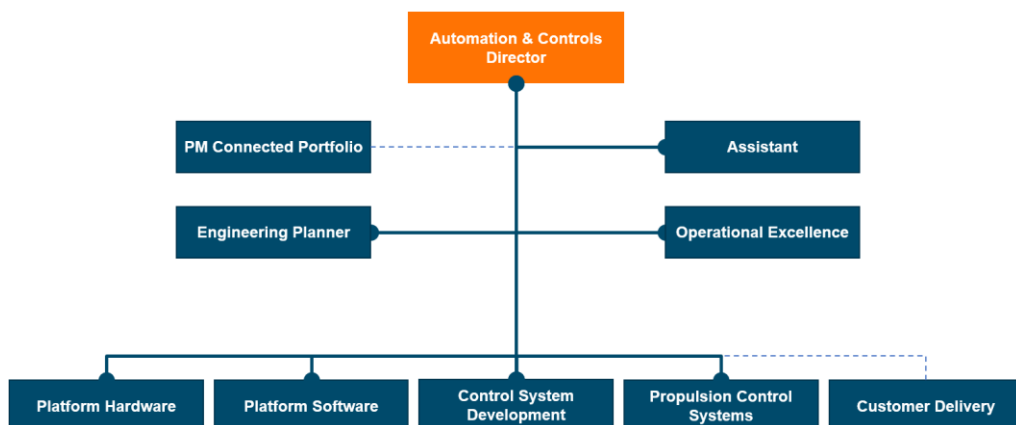


Figure 2 2019 Automation & Controls organisation chart

1.3 Wärtsilä Unified Controls (UNIC)

“The Wärtsilä UNIC is an embedded engine control system for Wärtsilä 4-stroke engines incorporating decades of experience, knowledge, and expertise. It makes engines safe, environmentally energy efficient, reliable and flexible. The system is designed to meet the highest levels of reliability, including special measures for redundancy, fault tolerance, and first class mechanical and electrical design.” /8/

The development of the first generation UNIC version began during the year 2002 and was released for customers in 2006. The product was actively maintained and further developed until 2016.

The second generation UNIC for 6-series engines was first initiated in 2011 with the objective to create a common platform for all Wärtsilä machinery. This 2nd generation UNIC is an evolution of the first generation with a number of added benefits.

The key benefits between 1st and 2nd generation UNIC /9/:

- Communication redundancy – 2nd gen UNIC uses an HSR ring topology for faster and more reliable communication between modules, whereas UNIC1 uses double CAN setup.
- Performance improvement – 2nd gen UNIC uses more up to date modules which have more versatile high-power channels, increased computing power, and memory.
- Communication separated from the display – 1st gen UNIC combined the LDU and MCM wherein 2nd gen UNIC there is a separate communication module.
- New user interface – LDU-30 has a new design with extra information display for the operator.
- Redundancy and fault tolerance – 2nd gen UNIC design meets SME demands, meaning that a single automation failure shall not result in a shutdown of the engine.
- 2nd gen UNIC introduces new features such as enhanced emission control, DF engines and load control when gas quality gets worse.

UNIC's modular and user-centric design allows easy access to the components of the system for installation and serviceability. Some parts and functions within the UNIC configuration are optional, depending on the engine and the installation requirements.

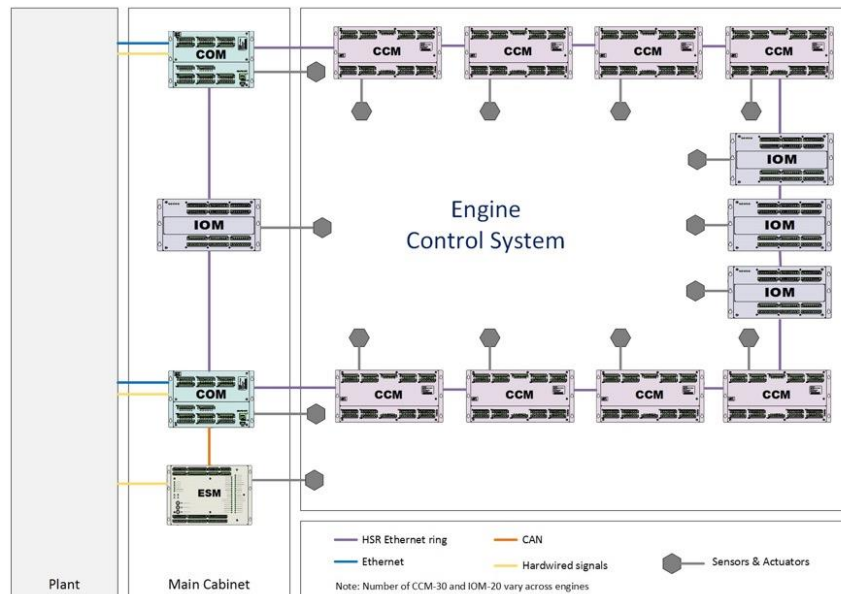


Figure 3 UNIC general layout /8/

The main components of the UNIC system are as follows:

- LDU-30 is an intelligent operation panel. It is designed for use in harsh environments and supports different mounting options, such as control cabinet and rail mounting. The key functions are local and remote-control selection, local start-stop, trip, shutdown reset, and status information. /10/
- COM-10 is designed to primarily act as the interface of UNIC as well as to measure speed and position of an engine. External control systems can be connected to UNIC via COM-10.
- CCM-30 is mainly responsible for combustion control. It monitors and controls all the injection and combustion functions, and inlet valve timing for the cylinders. The number of modules varies according to the number of cylinders. /8/
- IOM-20 handles all measurements in specific areas of the engine. Modules are placed close to sensors and measurable devices. The number of modules

varies according to the number of cylinders, the engine type, and the application. /8/

- ESM-20 handles functions related to the safety of the crew in case of failures with the engine. It provides safety functions such as shut down due to over speed or low lubricating oil pressure, etc. /8/



Figure 4 2nd generation UNIC modules /8/

1.4 Objectives and topic of choice

The main goal of the project is to analyse the current implementation, research, and test the improvements done. The current problem and reason for this topic are that the currently used COM-10 module's FPGA chip resources are nearing the maximum limit. This will be a big problem in the future as that would impact further development of applications used by the modules. The two solutions found are:

- The First solution would be to use a dual FPGA design, where one FPGA chip's sole function is communication with other modules and second chip for the applications.
- The Second solution is to disable HSR functionality from the module and use an external HSR compatible networking switch for communication.

The first solution was not chosen, as it would require extensive changes to the hardware and software including much more time for development. The topic chosen for this document is the second solution, as it incorporates elements of embedded systems software analysis and configuration, electrical measurements and control as well as network troubleshooting and analysis encompassing major subjects studied in the degree Information Technology at Vaasa University of Applied Sciences.

1.5 Implementation plan

The scope of this thesis is to research and test one solution that meets the following requirements:

- Free up the FPGA resources used by the chip
- Communication works with other modules
- Low latency between modules

These requirements required knowledge on how the system currently works, what are the major problems when incorporating this solution, and how HSR protocol works. It then continues with the study of the concepts needed to fully understand the topic, as well as the gathering of relevant information that clarifies the technical, specific details of the issues to be solved, a process covered in the theory and background information chapter. Steps for implementing the required prerequisites are explained in the solution implementation chapter. The testing environment is presented in the testing environment and results chapter. The results obtained after testing the solution are presented in the testing environment and results chapter, as well. Finally, conclusions and further possible improvements are proposed in the conclusions and further improvements chapter.

The thesis work is following the engineering design plan that is presented in **Figure 5**

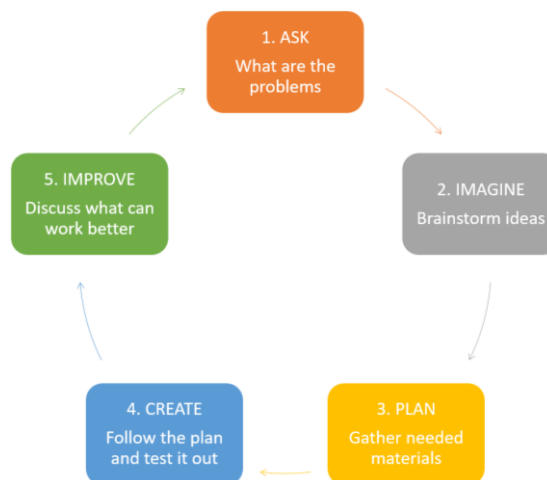


Figure 5 Engineering design plan

2 THEORY AND BACKGROUND INFORMATION

This chapter describes all the fundamental theories needed in the thesis and will be split into five sections. This is required to get an understanding of how the system works. The final setup for the objective of the thesis is presented in **Figure 6**.

- The first section explains background information about High-availability Seamless Redundancy protocol or HSR in short.
- The second section explains why Cisco's IE-4000 industrial switch was used for this project.
- The third section explains information about Wärtsilä's HSR protocol.
- The fourth section explains what Wärtsilä Modular Application Platform is.
- The final section explains how the communication works in UNIC2 system.

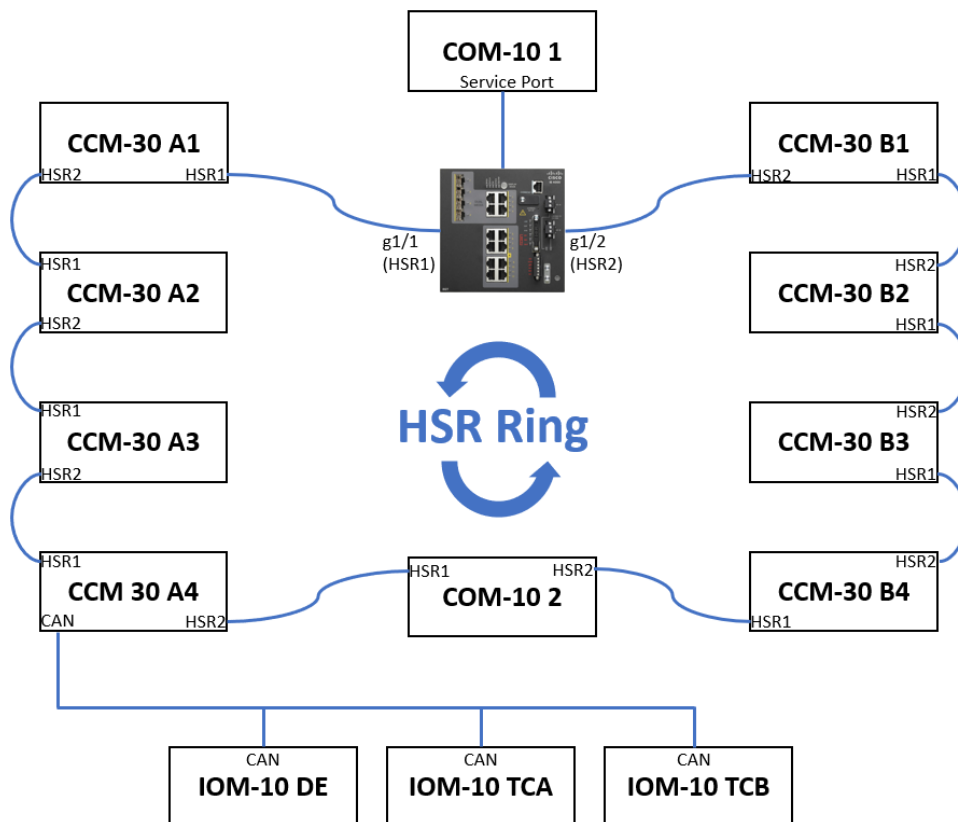


Figure 6 Objective of the thesis and target setup

2.1 High-availability Seamless Redundancy

High-availability Seamless Redundancy is a network protocol for Ethernet that provides seamless failover against any failure of network component. HSR protocol is defined in IEC 62439-3 and is designed to achieve zero recovery time in ring topologies, making it suitable for automation networks where downtime of Industrial Automation and Control Systems (IACS) applications should be kept to a minimum. /11/

The nodes connecting to the HSR ring are referred to as Doubly Attached Nodes or DANHs. Singly Attached Nodes (SANs) are attached to the HSR ring through a device called Redundancy Box (RedBox). The RedBox acts as a DANH for all traffic for which it is the source or the destination. /11/

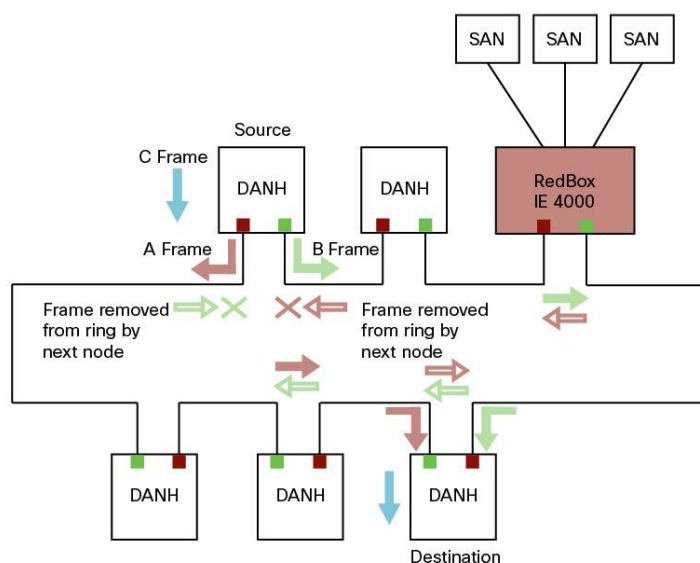


Figure 7 Example of HSR ring carrying Unicast traffic /11/

Devices that do not support HSR out of the box (computers, printers, etc) cannot be attached to the HSR ring directly because an HSR device needs two ports. These nodes are attached to the HSR ring through a RedBox, as seen from **Figure 7**. Non-HSR SAN devices are attached to the upstream switch ports. The RedBox generates the supervision frames on behalf of these devices so that they are seen as DANH devices on the ring. Because the RedBox emulates these as DANH, they are called Virtual Doubly Attached Nodes (VDAN). /11/

2.2 Cisco IE-4000 Series Switch

Cisco Systems, Inc is an American multinational corporation that develops, manufactures and sells networking hardware, telecommunications equipment, and other high-technology services and products. Cisco is headquartered in San Jose, California. /12/

The model of Cisco networking hardware used in this project was IE-4000-8T4G-E which was released for the public during autumn 2015. There are plenty of solutions from other manufacturers available to achieve the desired results, but this switch was used because of its ease of use and my background knowledge in Cisco products, IOS (Internetwork Operating System), and also that Cisco was kind enough to let me loan the switch for the period of this project.

IE-4000 Series switches are built to withstand extreme environments while adhering to overall IT network design, compliance, and performance requirements. The series is ideal for industrial Ethernet vertical industries applications where hardened products are required, including factory automation, energy and process control, intelligent transportation systems, oil and gas field sites, city surveillance programs, and mining. With improved overall performance, greater bandwidth, a richer feature set, and enhanced hardware, the Cisco IE-4000 Series complements needed requirements for this test setup. /13/



Figure 8 Cisco IE-4000-8T4G-E switch used in the thesis /14/

2.3 FPGA Functional Description

All the FPGA development work is done using FPGA vendor Xilinx specific tool-set: ISE Design Suite, System Edition. The design suite includes all the necessary tools to generate a binary file that will be programmed to the FPGA chip.

The FPGA design files are organised into three main directories:

- COM-10 specific files
- CCM-30 specific files
- Common files

Where common part includes the following main functions:

- PCI slave interface
- External DDR2 memory controller
- Positioning engine
- Interrupt controller and failure event FIFO
- HSR Ethernet switch
- Diagnostics MCU interface
- Accelerometer interface

COM-10 and CCM-30 designs are differentiated with custom input and output subsystem blocks that contain a different set of input and output channels and peripherals.

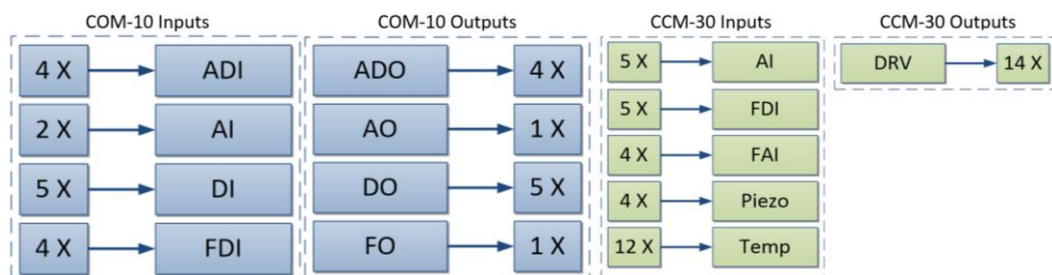


Figure 9 COM-10 and CCM-30 inputs and outputs

2.4 Wärtsilä HSR

The Wärtsilä High-availability Seamless Redundancy protocol or WHSR, in short, is a Wärtsilä in-house developed protocol for communication functionality on top of HSR extended Ethernet frames. This protocol is to replace WE-CAN+ protocol used in 1st generation UNIC systems.

Since it is an Ethernet-based protocol, it can offer a better payload to header ratio compared to CAN based protocol. That is because the largest Ethernet frame is much bigger than the largest CAN frame.

In addition, the physical layers of the existing Ethernet interfaces are based on 100Base-TX which supports 100 Mbps bit rate compared with maximum 500 kbps which CAN bus supports.

For these two reasons, WHSR is able to move vastly more data between the modules compared to the WE-CAN+ counterpart, enabling use cases which would be impossible to achieve before.

2.5 Wärtsilä Modular Application Platform

Wärtsilä Modular Application Platform (WMAP) engine control software is divided into three major parts: System Platform, Application Modules, and Configuration.

System Platform provides the means to use the hardware I/O, HSR communication, data storage, and so on. It is a common platform for all engines, but it can be configured in various ways – even at the level where functionalities are left out as example CAN communication.

Application Modules implement the actual process control. The combination of Application Modules varies greatly between different engine types. The simplest engine control software is for Diesel and the most complex being for Dual Fuel.

Configuration presents the third major part of the software architecture. Via configuration it is possible to utilise the system software and applications in a completely different manor making some software work in completely different engine application. All sensors, safety settings, etc. are stored to the configuration data. System software simply operates for example from Hardware I/O point of view exactly how I/O points are configured.

2.6 UNIC2 Communication

This chapter describes the communication protocol for UNIC2 real-time Ethernet communication. This will go through Hardware environment where the protocol is used, Layer 2 protocol framing that is used in UNIC2 real-time communication and the software blocks and their functionality.

2.6.1 Hardware environment

UNIC2 COM-10 module contains five Ethernet interfaces: one Service Port connected directly to CPU and four interfaces connected via FPGA to CPU that can operate in HSR mode (2 x ring) or as normal Ethernet interfaces. Both HSR rings have a separate interface to CPU, named HSR1 and HSR2.

Communication between HSR rings and from HSR ring to Service Port is enabled via CPU. CPU Software is required to be able to switch traffic with best possible performance.

The main advantages of an FPGA chip compared to a CPU are:

- Low latency – Meaning the time between an input and its response. This can be 50 times faster when using an FPGA chip.
- Connectivity – On an FPGA, you can implement any data source, such as networking interface or a sensor, directly to the pins of the chip.

UNIC2 Ethernet interfaces are described in **Figure 10**.

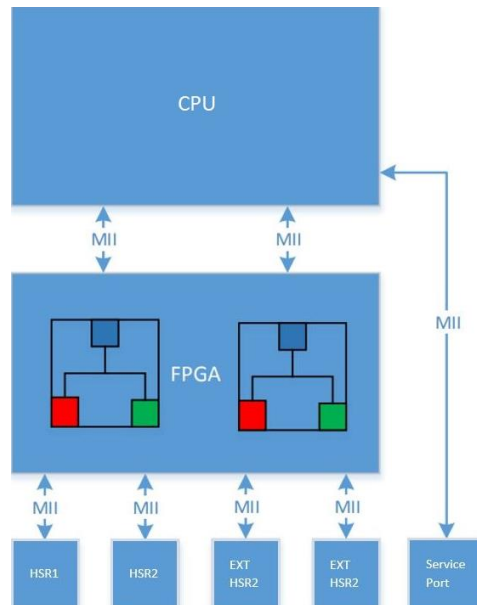


Figure 10 UNIC2 Ethernet interfaces

2.6.2 Layer 2 framing

UNIC2 system uses different communication protocols for real-time communication. The used Ethernet Layer 2 framing utilises VLAN tagging. The Ethernet frame format for real-time data is depicted in **Figure 11**.

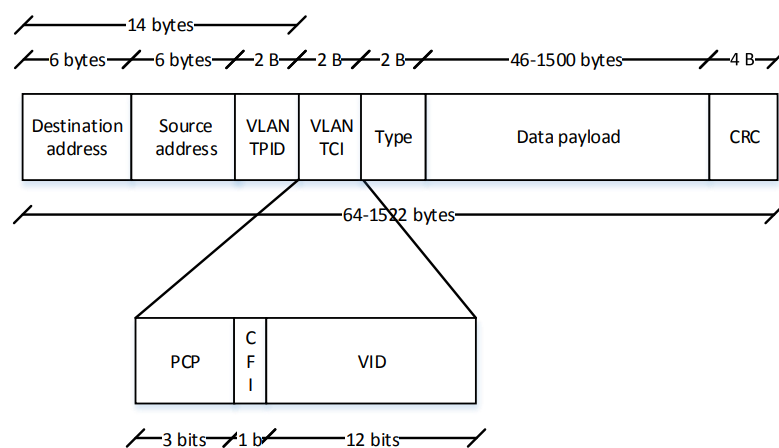


Figure 11 Layer 2 frame format

3 PRE-SOLUTION TESTING

Before implementing a solution for this project there needs to be tests done to figure out what needs to be modified for the external switch to work seamlessly with the rest of the modules. Looking back to **Figure 6**, the modifications needed to be done are only subjected to COM-10 1 and the external Cisco switch. What changes are done to COM-10 1 are explained later in this chapter.

3.1 Wireshark packet test when connected straight to PC

First conducted test was to figure out in what form WHSR packets are sent from COM-10 module to PC using Wireshark, which is an open-source packet analyser used for network troubleshooting, analysis, and communication protocol development. As WHSR is a Wärtsilä in-built communications protocol built on top of HSR, Wireshark is unable to interpret without using a translator plugin provided by Wärtsilä for development purposes.

For the first test there only needs to be one connection from Ethernet (EXT), as seen from **Figure 12** to PC's network adapter.

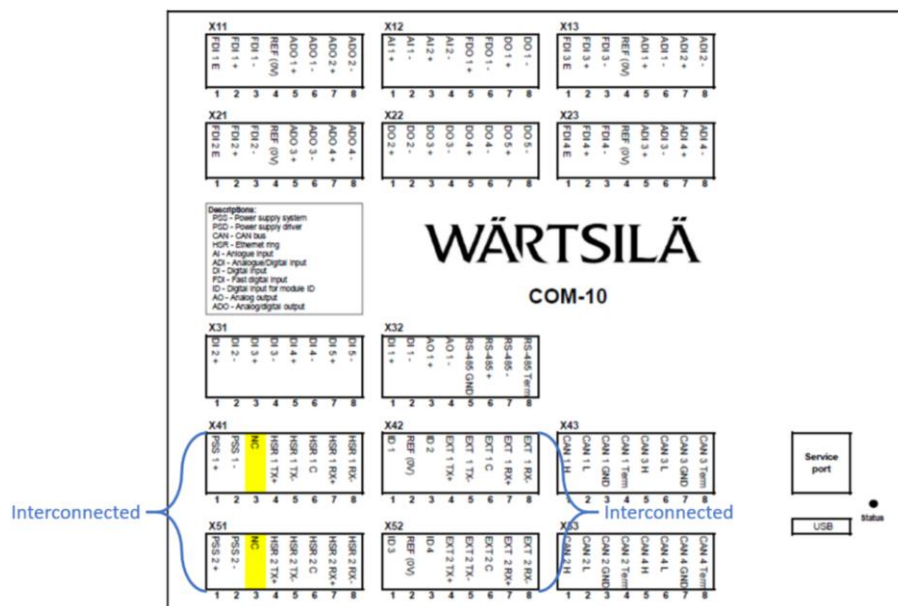


Figure 12 COM-10 Layout. Left is Ethernet HSR, right is Ethernet EXT

The first test setup is depicted in **Figure 13**. COM-10 requires an external power supply that is providing 24 VDC and 360 mA.



Figure 13 First test setup

In this setup COM-10 will be sending HSR supervision packets from Ethernet (HSR), meaning that there is an internal HSR switch enabled and functioning as intended, as well as WHSR broadcast messages for all modules to receive from Ethernet (EXT).

Without modifications to the source code, this would not work as intended because it would be sending HSR supervision packets and WHSR broadcast messages from Ethernet (HSR). When HSR has been enabled, it adds an extra trail to the frame. Getting around this issue requires some modification to the source code.

Hardcoding it to use the correct interface required only two extra lines of code. The source code is linked to HAL or Hardware Abstraction Layer which is a software subsystem that emulates platform-specific details, giving programs direct access to the hardware resources. HAL points to a hexadecimal value of 0x21, which means that the input and output are using the Ethernet (HSR) interface. Modifying the value from 0x21 to 0x22 changes the input and output interface to Ethernet (EXT) (**Figure 14**).

This can also be seen from the console view of the module when activating debugging messages, examine **Figure 15**.

```
pStreamData->streamConfig.src = 0x22;
pStreamData->streamConfig.dst = 0x22;
```

Figure 14 Modified input and output

```
1041: [WHSR_COMM INFO] Created stream:
1042: [WHSR_COMM INFO] - streamId = 2052
1043: [WHSR_COMM INFO] - streamCfg.vlanId = 2052
1044: [WHSR_COMM INFO] - streamCfg.vlanPcp = 7
1045: [WHSR_COMM INFO] - streamCfg.srcItf = 0x21
1046: [WHSR_COMM INFO] - streamCfg.dstItf = 0x21
1047: [WHSR_COMM INFO] - streamCfg.dstAddr = 1F:11:E1:01:FF:FD
1048: [WHSR_COMM INFO] - bufConfig.maxFrame = 1500
1049: [WHSR_COMM INFO] - bufConfig.rxFifoSize = 416
1050: [WHSR_COMM INFO] - bufConfig.txFifoSize = 416
```

Figure 15 WHSR debugging messages

The modified software is uploaded to the COM-10 module using UNITools' download tool. For PC to be able to communicate with the module, both devices need to be in the same subnet mask.

Source	Destination	Protocol	Length	Info
COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCN30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCN30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCN30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
COM10_1	BCAST	WHSR DISTDC	952	COM10_1 -> BCAST, Template 5
COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCN30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCN30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCN30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCN30, 0 COM10, 0 IOM10, 0 IOM10 busy bits

Figure 16 Wireshark capture

As **Figure 16** shows, the modifications were successful. The PC can now receive WHSR broadcast messages from COM-10 1 Ethernet (EXT) port. These messages carry important information, for example, the status of the module and which VLAN ID was used.

3.2 Implementing Cisco switch to capture WHSR messages

The reason for this second test was to make sure the switch used in this project was able to forward WHSR broadcast messages to the PC. The module did not require

any further modifications, other than changing the input and output hexadecimal values from 0x21 to 0x22. Like it was done on the first test.

As stated before, COM-10 uses VLAN Tagging, also known as Frame Tagging, is a method developed by Cisco to help identify packets travelling through trunk links. When an Ethernet frame traverses through a trunk link, a special VLAN tag is added to the frame and sent across the trunk link. As it arrives to the end of the trunk link, the tag is removed, and the frame is sent to the correct access link port according to the switch's table so that the receiving end is unaware of any VLAN information.

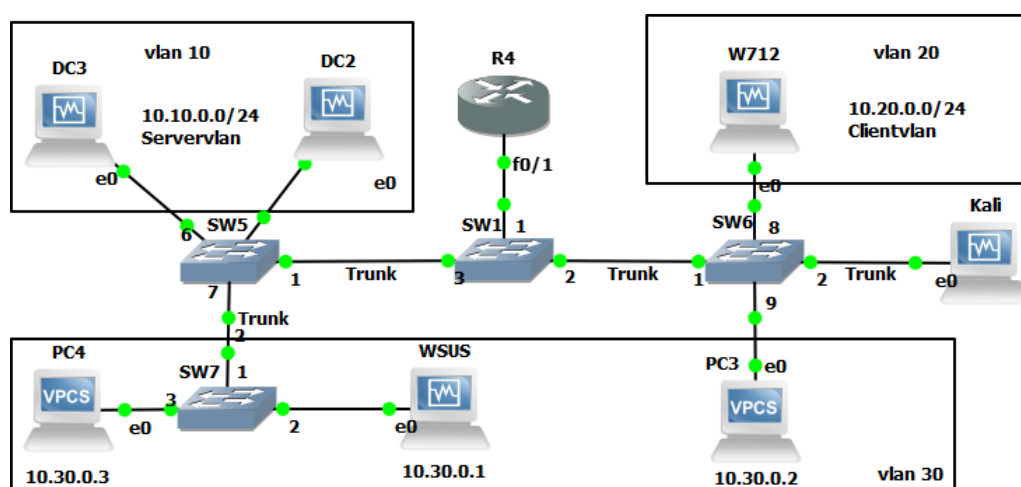


Figure 17 VLAN tagging illustrated /15/

The idea of VLAN Tagging is illustrated in **Figure 17**. Without VLAN Tagging and assigning trunked ports VLAN 30 can't talk to VLAN 10 or 20. However, when an IEEE 802.1Q trunked port is presented, it will be able to communicate with all the VLANs.

COM-10 uses VLAN IDs 2052, 2053, 2054, and 2055 in WHSR broadcast messages, so for the Cisco switch to correctly forward messages with those IDs, the switch needs to be configured. The switch is running with an external power supply that is supplying 20 VDC and 1,55 A.



Figure 18 Second test setup

Before configuring the Cisco switch, it is possible to look-up how many different frames each interface has transmitted and received from the console.

```

gigabitEthernet 1/1                               Switch#show controllers ethernet-controller fastEthernet 1/5
Receive                                             Transmit FastEthernet1/5                               Receive
4539828 Bytes                                     16125 Bytes                                           15864 Bytes
  0 Unicast frames                                15 Unicast frames                                     0 Unicast
  22234 Multicast frames                          100 Multicast frames                                 115 Multica
  0 Broadcast frames                              3 Broadcast frames                                   40 Broadca
  0 Unicast bytes                                 0 Too old frames                                     0 Unicast
4450892 Multicast bytes                           0 Deferred frames                                    11954 Multica
  0 Broadcast bytes                               0 MTU exceeded frames                               3910 Broadca
  0 Alignment errors                             0 1 collision frames                                0 Alignme
  0 FCS errors                                   0 2 collision frames                                0 FCS err
  0 Oversize frames                              0 3 collision frames                                0 Oversiz
  0 Undersize frames                             0 4 collision frames                                0 Undersi
  0 Collision fragments                          0 5 collision frames                                0 Collisi
  0 Minimum size frames                         0 6 collision frames                                0 Minimum
10538 65 to 127 byte frames                       0 7 collision frames                                19 Minimum
10537 128 to 255 byte frames                      0 8 collision frames                                113 65 to 1
  0 256 to 511 byte frames                      0 9 collision frames                                23 128 to
1159 512 to 1023 byte frames                     0 10 collision frames                               0 256 to
  0 1024 to 1518 byte frames                    0 11 collision frames                               0 512 to
  0 Overrun frames                              0 12 collision frames                               0 1024 to
  0 Pause frames                                0 13 collision frames                               0 Overrun
  0 Pause frames                                0 14 collision frames                               0 Pause f
  
```

Figure 19 g1/1 and f1/5 side by side comparison

Figure 19 shows that the gigabitEthernet 1/1 port has received more than 22000 multicast (broadcast) frames from COM-10 and fastEthernet 1/5 port, which is connected to the PC has received only 100 multicast (broadcast) frames. This means that the WHSR broadcast messages are not being forwarded by the switch.

Using the configuration commands shown in **Figure 20**, the switch is now able to forward VLAN Tagged frames through the assigned trunk ports.


```

Switch> en
Switch# configure terminal
Switch(config)# spanning-tree extended system-id
Switch(config)# vtp mode transparent
Switch(config)# vlan 2052
Switch(config-vlan)# vlan 2053
Switch(config-vlan)# vlan 2054
Switch(config-vlan)# vlan 2055
Switch(config-vlan)# exit
Switch(config)# interface range gigabitEthernet 1/1, fastEthernet 1/5
Switch(config-if-range)# switchport mode trunk
Switch(config-if-range)# switchport trunk allowed vlan 2052, 2053, 2054, 2055
Switch(config-if-range)# switchport trunk native vlan 1

```

Figure 20 Cisco switch configuration commands

```

gigabitEthernet 1/1          Switch#show controllers ethernet-controller fastEthernet 1/5

```

Receive	Transmit FastEthernet1/5	Receive
33926232 Bytes	28503917 Bytes	22729 Bytes
0 Unicast frames	79 Unicast frames	0 Unicast frames
166150 Multicast frames	140487 Multicast frames	143 Multicast frames
0 Broadcast frames	0 Broadcast frames	51 Broadcast frames
0 Unicast bytes	0 Too old frames	0 Unicast bytes
33261632 Multicast bytes	0 Deferred frames	17829 Multicast bytes
0 Broadcast bytes	0 MTU exceeded frames	4900 Broadcast bytes
0 Alignment errors	0 1 collision frames	0 Alignment errors
0 FCS errors	0 2 collision frames	0 FCS errors
0 Oversize frames	0 3 collision frames	0 Oversize frames
0 Undersize frames	0 4 collision frames	0 Undersize frames
0 Collision fragments	0 5 collision frames	0 Collision fragments
0 Minimum size frames	0 6 collision frames	0 Minimum size frames
78744 65 to 127 byte frames	0 7 collision frames	18 Minimum size frames
78744 128 to 255 byte frames	0 8 collision frames	124 65 to 127 byte frames
0 256 to 511 byte frames	0 9 collision frames	52 128 to 255 byte frames
8662 512 to 1023 byte frames	0 10 collision frames	0 256 to 511 byte frames
0 1024 to 1518 byte frames	0 11 collision frames	0 512 to 1023 byte frames
0 Overrun frames	0 12 collision frames	0 1024 to 1518 byte frames
0 Pause frames	0 13 collision frames	0 Overrun frames
	0 14 collision frames	0 Pause frames

Figure 21 g1/1 and f1/5 comparison after configuration

No.	Time	Source	Destination	Protocol	Length	Info
2083..	1068.424149	COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
2083..	1068.433919	COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCH30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
2083..	1068.433920	COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
2083..	1068.445125	COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCH30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
2083..	1068.445125	COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
2083..	1068.456436	COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCH30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
2083..	1068.456436	COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
2083..	1068.461903	COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCH30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
2083..	1068.461903	COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
2083..	1068.472061	COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCH30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
2083..	1068.472061	COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
2083..	1068.481802	COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCH30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
2083..	1068.481803	COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
2083..	1068.491899	COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCH30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
2083..	1068.491900	COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
2083..	1068.494097	COM10_1	BCAST	WHSR DISTDC	952	COM10_1 -> BCAST, Template 5
2083..	1068.495444	COM10_1	BCAST	WHSR DISTDC	920	COM10_1 -> BCAST, Template 10
2083..	1068.501908	COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCH30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
2083..	1068.502541	COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
2083..	1068.511862	COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCH30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
2083..	1068.511862	COM10_1	BCAST	WHSR DISTDC	200	COM10_1 -> BCAST, Template 1
2083..	1068.523297	COM10_1	BCAST	WHSR HB	118	COM10_1 Operational, 0 CCH30, 0 COM10, 0 IOM10, 0 IOM10 busy bits

Figure 22 Wireshark capture after configuration

Figure 21 and **Figure 22** show that the configuration on the switch was successful and the PC is now able to receive WHSR broadcast messages from COM-10. Configuring the switch using Cisco's IOS command-line interface (**Figure 20**) is very straightforward and most if not any command and feature can be found using Google's search engine.

3.3 Testing UNIC-level communication between COM-10 modules

The significance of the third test is to find out what are the necessary steps for achieving a working UNIC2 HSR communication system without any major modifications done to the modules source code. This test will incorporate the same steps done in subchapters 3.1 and 3.2.

One COM-10 module will act as a SAN device through Ethernet (EXT) port to the switch's port fastEthernet 1/5 and the other as DANH device through HSR1 and HSR2 ports to the switch's gigabitEthernet 1/1 and 1/2 ports. See **Figure 23** for a more detailed representation of the test setup layout.

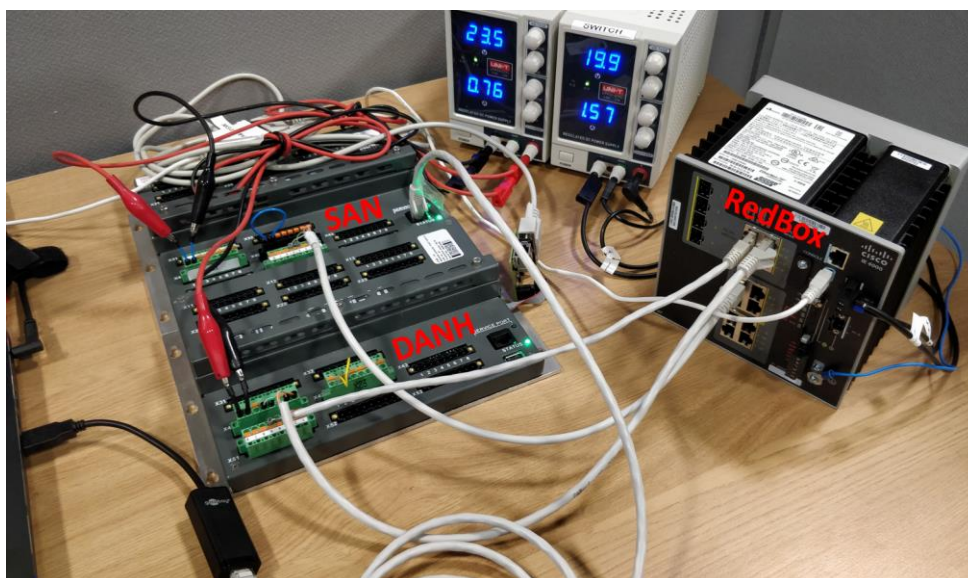


Figure 23 Third test setup

As the SAN device is connected to the RedBox from Ethernet (EXT) port, the same modification to the source code is needed as done in section 3.1. Modifying the hexadecimal value from 0x21 to 0x22 changes the input and output interface to Ethernet (EXT). DANH device does not require any modification, as stated before.

The RedBox (Cisco's switch) will be using almost the same configuration commands as seen from **Figure 20**, but with an addition of enabling HSR-ring 1, see **Figure 24** for the full command set.

```

Switch> en
Switch# configure terminal
Switch(config)# spanning-tree extend system-id
Switch(config)# vlan 2052
Switch(config-vlan)# vlan 2053
Switch(config-vlan)# vlan 2054
Switch(config-vlan)# vlan 2055
Switch(config-vlan)# exit
Switch(config)# interface range gigabitEthernet 1/1-2, fastEthernet 1/5
Switch(config-if-range)# switchport mode trunk
Switch(config-if-range)# switchport trunk allowed vlan 2052, 2053, 2054, 2055
Switch(config-if-range)# switchport trunk native vlan 1
Switch(config-vlan)# exit
Switch(config)# interface range gigabitEthernet 1/1-2
Switch(config-if-range)# shutdown
Switch(config-if-range)# no ptp enable
Switch(config-if-range)# hsr-ring 1
Switch(config-if-range)# no shutdown

```

Figure 24 Cisco switch HSR configuration commands

Using an Ethernet LAN tapper (**Figure 25**) gives a possibility to monitor what Ethernet devices are sending to each other, in this case, what DANH device is sending to the RedBox and SAN device.



Figure 25 Ethernet LAN tapper used in the project

As expected, DANH device is sending the same WHSR broadcast messages stating in what status the module is, in this case OPERATIONAL. In addition to that, it is also sending information directed specifically to the other module (SAN/COM-10 2) connected to the RedBox. UNIC2 modules can automatically see if there are other modules connected on the same network. See **Figure 26** for a Wireshark capture.

COM10_1	BCAST	WHSR HB	124 COM10_1 Operational, 0 CCM30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	206 COM10_1 -> BCAST, Template 1
COM10_1	BCAST	WHSR HB	124 COM10_1 Operational, 0 CCM30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	206 COM10_1 -> BCAST, Template 1
COM10_1	BCAST	WHSR HB	124 COM10_1 Operational, 0 CCM30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	206 COM10_1 -> BCAST, Template 1
COM10_1	BCAST	WHSR HB	124 COM10_1 Operational, 0 CCM30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	206 COM10_1 -> BCAST, Template 1
COM10_1	BCAST	WHSR HB	124 COM10_1 Operational, 0 CCM30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	206 COM10_1 -> BCAST, Template 1
COM10_1	BCAST	WHSR HB	124 COM10_1 Operational, 0 CCM30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	206 COM10_1 -> BCAST, Template 1
COM10_1	BCAST	WHSR HB	124 COM10_1 Operational, 0 CCM30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	206 COM10_1 -> BCAST, Template 1
COM10_1	BCAST	WHSR HB	124 COM10_1 Operational, 0 CCM30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	206 COM10_1 -> BCAST, Template 1
COM10_1	COM10_2	WHSR EDL Req.	958 COM10_1 -> BCAST, Template 5
COM10_1	COM10_2	WHSR EDL Req.	70 COM10_1 -> COM10_2, ID:03224
COM10_1	COM10_2	WHSR EDL Req.	70 COM10_1 -> COM10_2, ID:03225
COM10_1	COM10_2	WHSR EDL Req.	70 COM10_1 -> COM10_2, ID:03226
COM10_1	COM10_2	WHSR EDL Req.	70 COM10_1 -> COM10_2, ID:03227
COM10_1	BCAST	WHSR WE-CAN+ frame	70 [51, 21, DC, 2D][Malformed Packet]
COM10_1	BCAST	WHSR HB	124 COM10_1 Operational, 0 CCM30, 0 COM10, 0 IOM10, 0 IOM10 busy bits
COM10_1	BCAST	WHSR DISTDC	206 COM10_1 -> BCAST, Template 1
COM10_1	BCAST	WHSR WE-CAN+ frame	112 [51, FF, CUS, 0E]

Figure 26 Wireshark capture on the third test

The title of the section means that the communication established now should be visible and measurement values from both devices are readable and writable from UNITool. When connecting to a system, EDL or Enhanced Diagnostics Log gives useful information about the connected system. EDL can inform the user for example in what state the connected modules are, what communication protocol has been established, what system modules are missing, what sensor failures are found, and if measurement values are above or below the pre-set limits.

PC Time	Module	Type	Message	Parameter Data
2019-03-11 08:13:35...	COM10_1	CPS	CPS Record Retrieved	WE-CAN+ CFG (16777279)
2019-03-11 08:13:35...	COM10_1	CPS	CPS Record Created	WE-CAN+ RT (16777280)
2019-03-11 08:13:35...	COM10_1	CPS	CPS Record Created	CANOpen RT (16777220)
2019-03-11 08:13:35...	COM10_1	USM	USM initialization s...	
2019-03-11 08:13:35...	COM10_1	USM	USM performed sta...	Initialization, Pre-operational, Normal state transition
2019-03-11 08:13:37...	COM10_1	WHSR	HSR communicatio...	HSR 1
2019-03-11 08:13:37...	COM10_1	WHSR	HSR communicatio...	HSR 2
2019-03-11 08:13:42...	COM10_1	USM	Module is taking m...	Communication gateway module group
2019-03-11 08:13:42...	COM10_1	HWIO	Sensor Failure	HWIO reported channel's value is out of range, HWIO_COM10_1_PSS2 -
2019-03-11 08:13:42...	COM10_1	Communication	CANOpen: SDO op...	2, 2, 0x1000, 0, 0x05040000, Mentioned SDO parameter initialization failed for the given node, initialization will not be aborted
2019-03-11 08:13:42...	COM10_1	Communication	CANOpen: SDO op...	2, 2, SDO initialization of an optional node has failed
2019-03-11 08:13:52...	COM10_1	USM	System module mi...	IOM20_1
2019-03-11 08:13:52...	COM10_1	USM	System module mi...	COM10_2
2019-03-11 08:13:52...	COM10_1	USM	System module mi...	CCM30_1
2019-03-11 08:13:52...	COM10_1	USM	System module mi...	CCM30_2
2019-03-11 08:13:52...	COM10_1	USM	System module mi...	COM10_3
2019-03-11 08:13:52...	COM10_1	USM	System module mi...	CCM30_3
2019-03-11 08:13:52...	COM10_1	USM	System module mi...	IOM20_2
2019-03-11 08:13:52...	COM10_1	USM	USM performed sta...	Pre-operational, Operational, Normal state transition
2019-03-11 08:19:31...	COM10_1	WHSR	Module appeared o...	1, 5, 2
2019-03-11 08:19:31...	COM10_1	USM	Module back to no...	COM10_2
2019-03-11 08:19:31...	COM10_1	CAN	Module CAN com...	COM10_2
2019-03-11 09:06:41...	COM10_1	Communication	User has connected...	rm066
2019-03-11 09:08:22...	COM10_1	DC	DC value written ex...	32768, DC_UNIT_DEFAULT - code name for DC unit testing

Figure 27 Enhanced Diagnostics Log in this test setup

Given that there are no external sensors, actuators or solenoids connected to the modules, the only possible things to measure are for example voltages of the modules, engine status, and speed of the engine. See **Figure 28** for more detailed representation.

Update	Symbol Name	Value	Unit	Limits
<input checked="" type="checkbox"/>	HWIO_COM10_1_PSS1.DC Value -	23.56	V	0.00 / 40.00
<input checked="" type="checkbox"/>	HWIO_COM10_2_PSS1.DC Value -	23.51	V	0.00 / 40.00
<input checked="" type="checkbox"/>	IS872.DC Value - Ready for start	0		0 / 1
<input checked="" type="checkbox"/>	IS883.DC Value - Engine status run mode	1		0 / 1
<input checked="" type="checkbox"/>	IS882.DC Value - Engine status standby mode	0		0 / 1
<input checked="" type="checkbox"/>	IS879.DC Value - Engine status shutdown mode	0		0 / 1
<input checked="" type="checkbox"/>	IS871.DC Value - Engine status start mode	0		0 / 1
<input checked="" type="checkbox"/>	STY196.DC Value - Engine speed	720	rpm	0 / 1000
<input checked="" type="checkbox"/>	IS889.DC Value - Engine status emergency stop mode	0		0 / 1

Figure 28 Measured values and status of the engine

The main reason for this third test was to check that the modules operate as intended regardless of the Cisco's external switch and looking from the outputs **Figure 26**, **Figure 27**, and **Figure 28**, it does. Wireshark can capture the messages sent from DANH to SAN devices through the RedBox and UNITool is able to connect to the system without any issues.

4 SOLUTION IMPLEMENTATION

The results gathered from the three tests in chapter 3 **Pre-Solution Testing** give a good starting point on what works and what still needs to be changed before the final solution is achieved.

The chapter 3 **Pre-Solution Testing** tests were successful, as UNIC-level communication was established but the required specifications were not yet implemented. For this project to be useful in any way, routing between the CPU and FPGA needed to be disabled to see, if the FPGA chips' resources were minimised. When disabling routing between the CPU and FPGA, HSR1, HSR2 and External HSR ports 1 and 2 are also disabled, as those are connected to the FPGA. The only possible WSHR communication interface available is the Service Port. **Figure 29** represents how the interfaces are connected inside COM-10 modules.

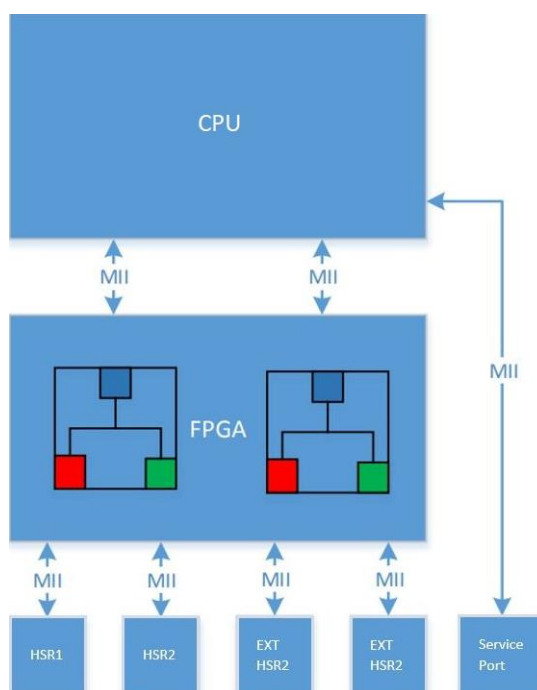


Figure 29 UNIC2 communication interfaces

Cisco's switch does not require any extra configurations, as it has already been configured correctly in the section 3.3 **Testing UNIC-level communication between COM-10 modules**, so the configuration steps are not going to be repeated again in this chapter.

This chapter will explain the changes done to the existing setup to achieve the desired requirement results.

4.1 Disabling the internal HSR switch from FPGA

As stated in section **2.3 FPGA Functional Description**, the internal HSR switch is configured and handled by the FPGA, and as such, a modified binary file for the FPGA chip is needed.

There were two possible ways to incorporate the solution: the hard way or the easy way.

The hard way would be to use the latest HAL (Hardware Abstraction Layer). Using the most recent HAL version would require modifications to both HAL and WMAP, whereas the easy way would be to use an older HAL version that requires changes only in WMAP. For the sake of keeping this project as simple as possible, this was the approach used in the project as the results would be the same with using either method.

The process for disabling the internal HSR switch is simple, as it is only a Boolean true/false statement in the top-level source file, which combines all the other source files together.

This Boolean true/false statement checks whether to generate the needed HSR switch wrapper, and it also assigns the Ethernet interfaces to the correct pins on the FPGA chip.

When Implementing Top Module, or compiling the binary file for the FPGA chip, the compiler first runs synthesis where the ISE Design Suite synthesises VHDL designs to create Xilinx-specific netlist files, which contain both logical design data and constraints. /16/

If the synthesiser ran without errors, it then proceeds to implement the design with four different steps: /17/

- Translate – Merges the incoming netlists and constraints into a Xilinx design file.
- Map – Fits the design into available resources on the target device, and optionally, places the design.
- Place and Route – Places and routes the design to the timing constraints.
- Generate Programming File – Creates a bitstream (binary) file that can be downloaded to the FPGA chip.

After the compilation has finished, ISE Design Suite will display the Design Summary where errors, warnings, and all the different reports can be seen. It will also report all used and available resources on the FPGA chip.

The generated FPGA bitstream is “baked” into a new firmware image using Yocto Project.

The Yocto Project is an open source collaboration project that provides templates, tools and methods to help create custom Linux-based systems for embedded and IoT products, regardless of the hardware architecture. /18/

“As an open source project, the Yocto Project operates with a hierarchal governance structure based on meritocracy and managed by its chief architect Richard Purdie. This enables the project to remain independent of any one of its member organizations, who participate in various ways and provide resources to the project.” /18/

The results of the FPGA binary compilation with and without the internal FPGA switch are presented in chapter **5 Testing Environment and Results**.

4.2 WMAP software modifications

WMAPI requires the same modifications as done in the previous testing (**Figure 14**). Instead of using the External HSR ports on COM-10 1 module, it would use the Service Port, as that is the only possible WHSR stream interface available now that the internal HSR switch has been disabled on the COM-10 1 module.

To avoid special software in the COM-10 1 module, the source code needed an IF-statement (**Figure 30**), where it would check the System ID and only execute the code on modules with the System ID 81, which correspond to only COM-10 1 module. Each module has its own Systems ID to distinguish themselves from each other.

```
/**
 * Author: Rene Mikhael
 * Reason for modification: Thesis work
 * What does this do: Checks whether to execute the IF statement: IF ModuleID is COM10_1.
 * What are the outputs: Changes the source and destination of the stream to Service Port
 */

if (SysGetModuleID() == 81)
{
    pStreamData->streamConfig.src = HAL_STREAM_IF_RT_COMM_API + HAL_STREAM_IF_SERVICE_PORT;
    pStreamData->streamConfig.dst = HAL_STREAM_IF_RT_COMM_API + HAL_STREAM_IF_SERVICE_PORT;
}

printf(" \n\n\n SYSTEM MODULE ID: %d \n\n\n", SysGetModuleID());
```

Figure 30 WMAPI stream modification snippet

The code in **Figure 30** fetches the hexadecimal values from HAL stream interface enumeration (**Figure 31**) and configures the source and destination interface to Service Port. $\text{HAL_STREAM_IF_RT_COMM_API (0x20) + HAL_STREAM_IF_SERVICE_PORT (0x10) = 0x30}$.

```
enum HAL_StreamInterface {
    HAL_STREAM_IF_HSR1 = 0x01, ///< First HSR ring interface
    HAL_STREAM_IF_HSR2 = 0x02, ///< Second HSR ring interface
    HAL_STREAM_IF_SERVICE_PORT = 0x10, ///< Service port
    HAL_STREAM_IF_RT_COMM_API = 0x20, ///< interface API
    HAL_STREAM_IF_LINUX = 0x80, ///< Linux
};
```

Figure 31 HAL stream interface enumeration

This was the only modification required on the WMAPI platform side, and the only thing left to do is to compile the target module files, which will be downloaded to the corresponding modules.

4.3 UNITool configuration and setup

Wärtsilä UNITool is an in-house developed maintenance tool used for downloading software to modules, and tuning, monitoring, testing, and troubleshooting systems equipped with the UNIC automation system.

The version of UNITool used in this project is UNITool 4.6, but the latest one available is UNITool 5.1 with many new features and improvements over the previous versions. The main reason for using an older version of UNITool is related to the HAL version used in this project, as that dictates which WMAP/UNITool version it is compatible with.

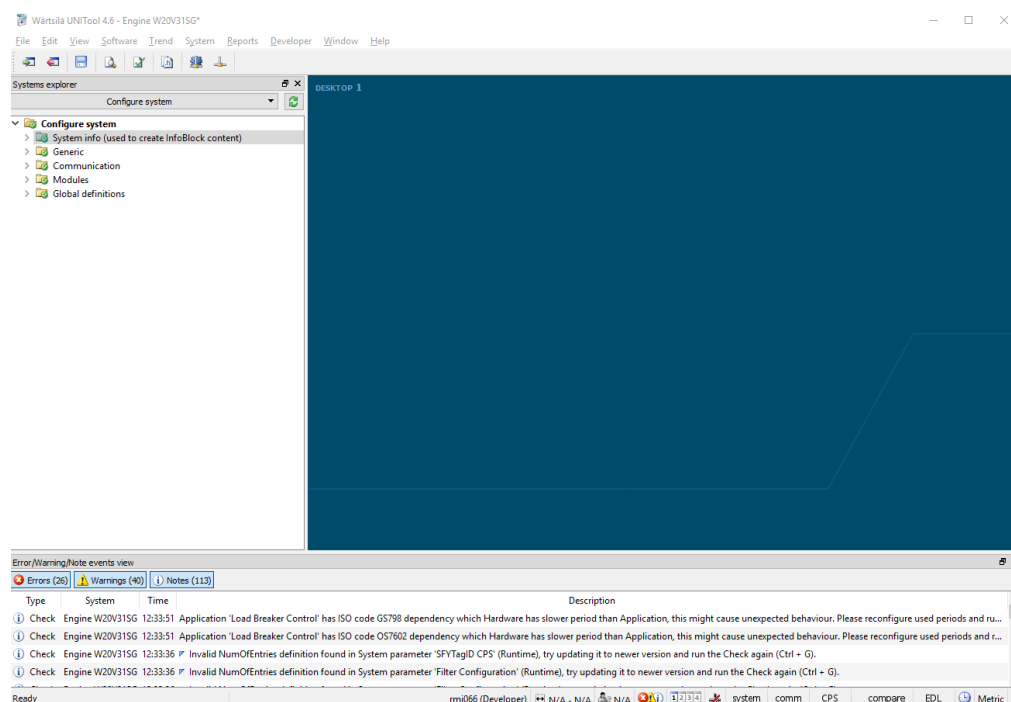


Figure 32 UNITool logged on window

When starting UNITool, you are given an option to what system configuration the user wants to log in. In **Figure 32** the user has logged on to an engine configuration package W20V31SG used in energy solutions, which will be the one used in this project. The reason for choosing this specific engine configuration is because this projects' testing environment has as a set number of modules available. Another major reason for choosing this engine package is that it is a 5-series engine because it supports both HSR and CAN protocols.

By default, this engine package is configured with using only CAN as its communication protocol. For enabling HSR, it is first required to change the option in System Parameter from False to True (**Figure 33**) and then add CAN routing over HSR protocol (**Figure 34**) in Communication buses for IOM-10 to work in the test setup.

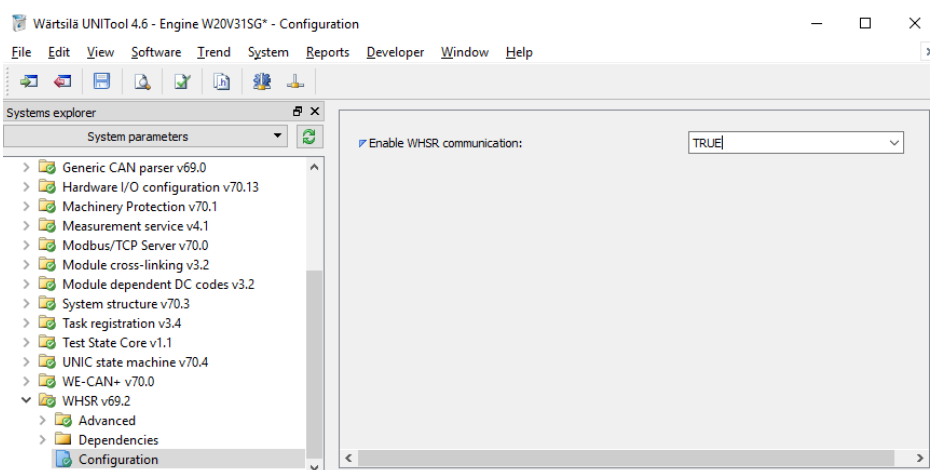


Figure 33 Enabling WHSR from System Parameters

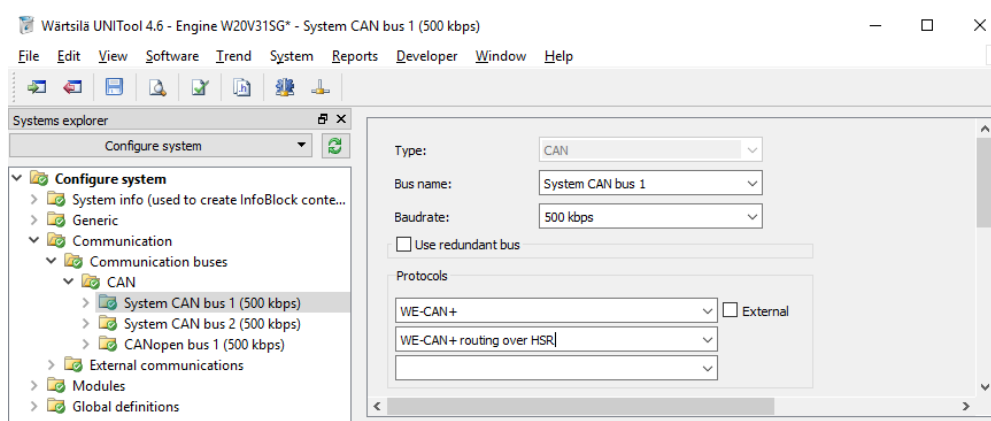


Figure 34 Adding CAN routing over HSR

The last step is to reload all the previously compiled target module files to UNITool, and also the modified Firmware, which will only be downloaded to the COM-10 1 module.

The testing environment and results gathered from the tests will be presented and explained in chapter 5 **Testing Environment and Results**.

5 TESTING ENVIRONMENT AND RESULTS

This chapter explains how the final solution was tested and what were the results. As this project has been split into smaller pieces, this chapter will also follow the same principle. The first section **5.1 FPGA compilation results** explain what differences were found in the Xilinx Design Summary report with and without the internal FPGA switch. The second section **5.2 Setting up the MiniRig testing environment** explains how the testing environment was set up, and what were the pre-requisites for doing that. When the MiniRig test setup has been established, the third section **5.3 MiniRig test results** present the testing results that were obtained during the testing process

5.1 FPGA compilation results

For there to be something to compare to, there was a requirement to do two separate compilations one after another. The first compilation was made without the internal FPGA switch and all the related dependencies removed and another compilation without any modifications done to it.

The time required for both compilations to complete vastly varied as the first compilation without the internal switch finished in under one hour when the second compilation took almost 3.5 hours. This was a time difference of over 2.5 hours or 75%, which is a considerable decrease in time taken for the compilation to finish.

The main objective for this project was to research and test whether disabling the internal switch would free up resources on the FPGA chip. This can be observed in the Xilinx Design Suite Summary report after the compilation of the bitgen binary file has finished. **Table 1** below explains how many Slices or bits were ripped off from the FPGA bus net with and without the internal HSR switch.

Slice Logic Utilisation	With the internal FPGA switch	Without the internal FPGA switch	Percentage difference
Number of Slice Registers	37%	11%	-26%
Number of Slice LUTs	65%	23%	-42%
Number used as logic	60%	22%	-38%
Number used as Memory	8%	1%	-7%
Number of occupied Slices	94%	37%	-57%
Number of MUXCY's used	31%	15%	-16%

Table 1 FPGA resource difference after the changes

When looking at the Xilinx Design Summary report, the most relevant information regarding resource utilisation is the number of Slice LUTs used in the system. In FPGA systems, the resource utilisation should never reach the maximum mark i.e. 100%. Looking at **Table 1**, the FPGA Slice resources used is 65% when the internal FPGA switch is utilised and all the required constraints generated for the correct pins on the FPGA chip. In turn, when the internal FPGA switch is not utilised and all the related constraint generation files have been skipped, the used FPGA Slice resources dropped to 23%. The percentage difference when comparing with and without the internal FPGA switch is over 40%, which is a considerable difference, as that gives new possibilities for FPGA application development.

The next section **5.2 Setting up the MiniRig testing environment** will explain more in detail how the testing environment was set up and what were the pre-requisites.

5.2 Setting up the MiniRig testing environment

There are two kinds of MiniRig testing environments available in Wärtsilä for testing engine configuration packages, applications, or modules.

Virtual MiniRig is what the name says. A virtualised testing environment located in servers that are mainly used for engine configurations and application developments before testing on an actual physical MiniRig. Virtualisation provides a possibility to quickly test how the engine configuration package works and give an

indication if there are any problems regarding the configuration. The same can be done when developing UNIC applications.

Setting up the virtual environment is simple, as it only requires setting up the server IP address and creation of the Virtual Modules (**Figure 35**). After that, the configuration is ready to be downloaded and tested on the modules.

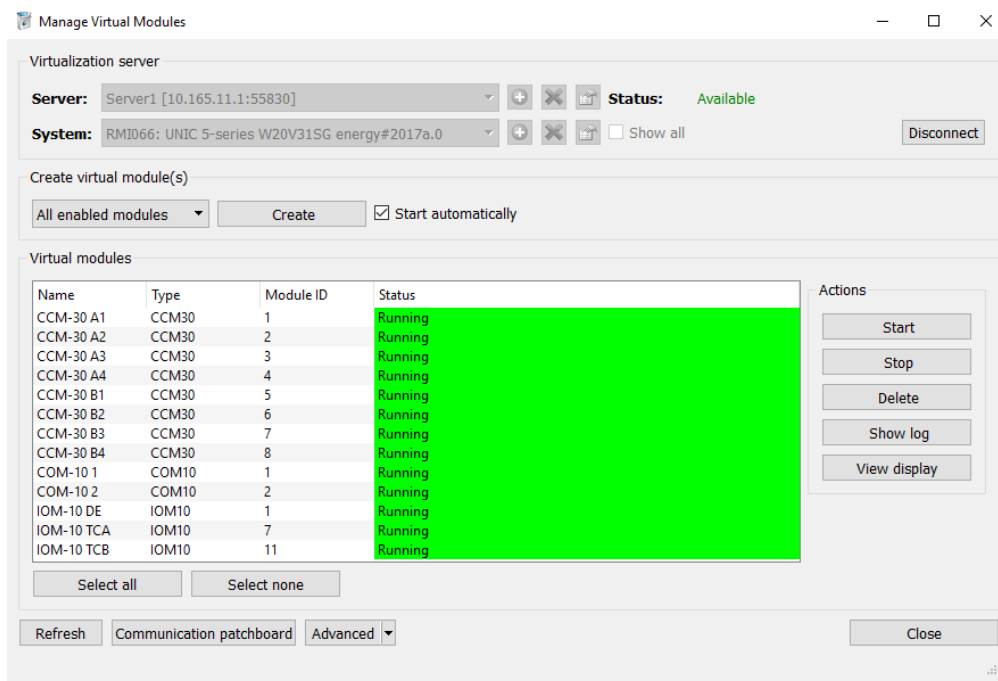


Figure 35 Setting up Virtual Modules

The project testing was carried out only with the physical MiniRig because the test setup was using Cisco’s external switch and that required non-standard communication wiring to the MiniRig.

Wärtsilä has three different MiniRig test racks available. The one used for this project is called “UNIC 2.0 Test Rig 1” (**Figure 36**) and it is mainly used for testing 5-series engines that utilise older and newer modules combined. CCM-30s, COM-10s, and IOM-10s. The 5-series engines use CAN protocol by default as the communication standard but have HSR support if it is needed. The latest 6-series engines use CCM-30, COM-10, and IOM-20 modules, and do not use CAN as the default protocol. Instead, HSR is used as the communication protocol between the modules.

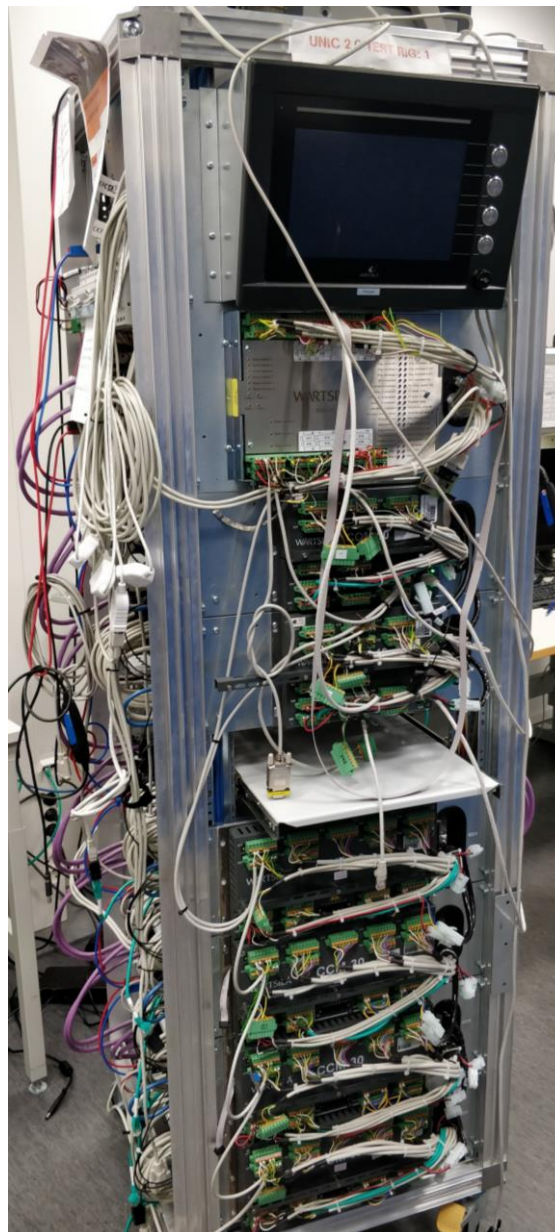


Figure 36 UNIC 2.0 Test Rig 1

As previously stated, the project was tested using a real engine configuration package, W20V31SG which is configured for energy solutions markets. The engine contains 13 modules in total. Eight CCM-30s, two COM-10s, and three IOM-10s. **Figure 37** on the next page represents the final test setup wiring diagram that was built with the UNIC 2.0 Test Rig 1.

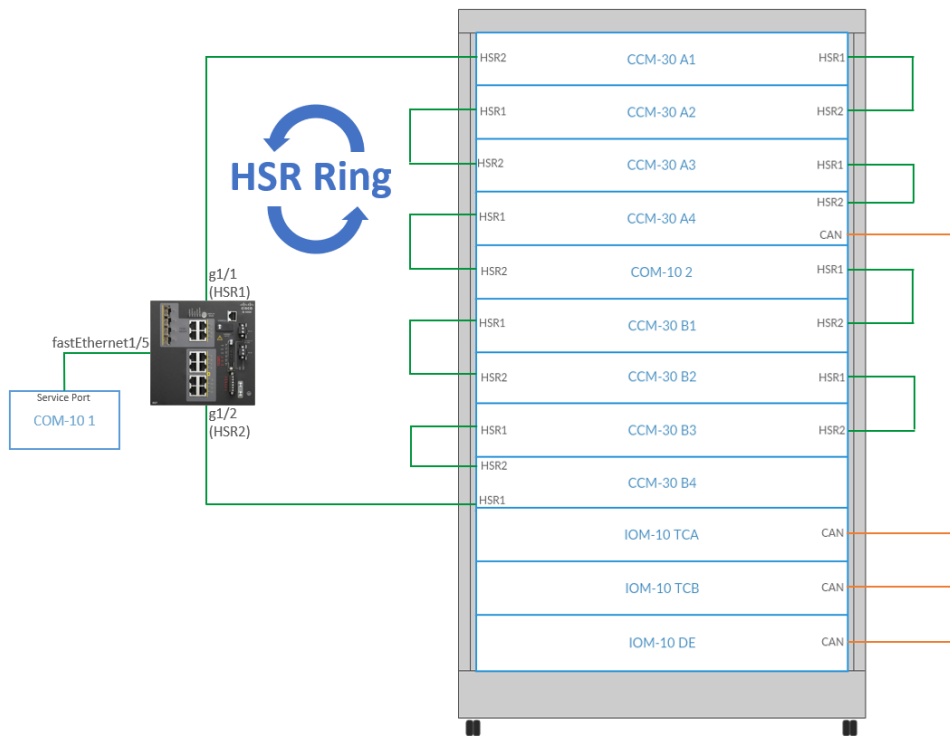


Figure 37 Final test setup MiniRig wiring diagram

Setting up the physical MiniRig differs from Virtual MiniRig. Each module in the physical MiniRig has its own corresponding module IDs, and that is changed using wiring modifications. When configuring the MiniRig for testing purposes, the developer first needs to set up all the modules on the MiniRig for UNITool to find them. This can be done using MiniRig Mapping function in UNITool (**Figure 38**).

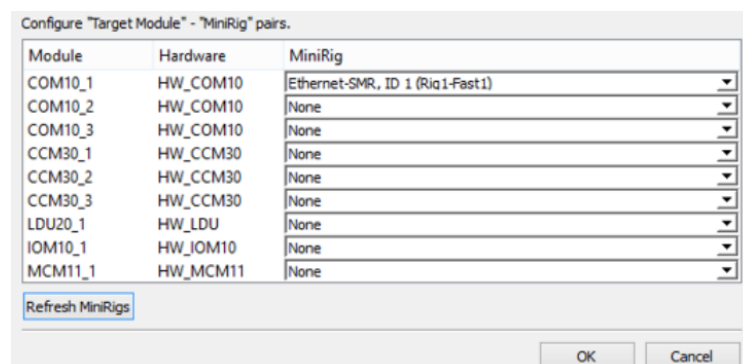


Figure 38 MiniRig Mapping

The MiniRig has a function to change the module IDs without doing the hardware changes manually. This can be done in the view, CAN ID Configuration. **Figure**

39 has only one module enabled, and that is only for reference purposes. After clicking OK, the user is required to reboot all the modules for the changes to take effect, as the module IDs are checked only during the boot up process.

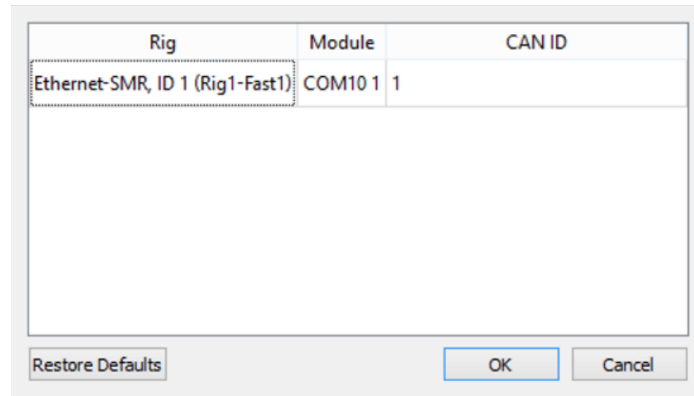


Figure 39 Module ID configuration

After the modules have booted up, then the download process can be started. As the testing setup uses an older software version, WMAP 4.6, the first step is to downgrade the Bootloader2 file to the correct version, 4.6.4.1. UNITool does not allow the download of the application and engine configuration file to the modules before the Bootloader2 downgrade process has finished (**Figure 40**).

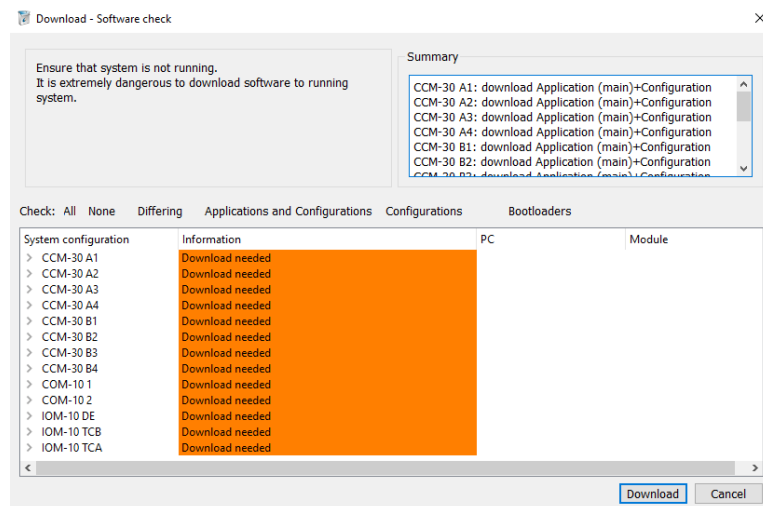
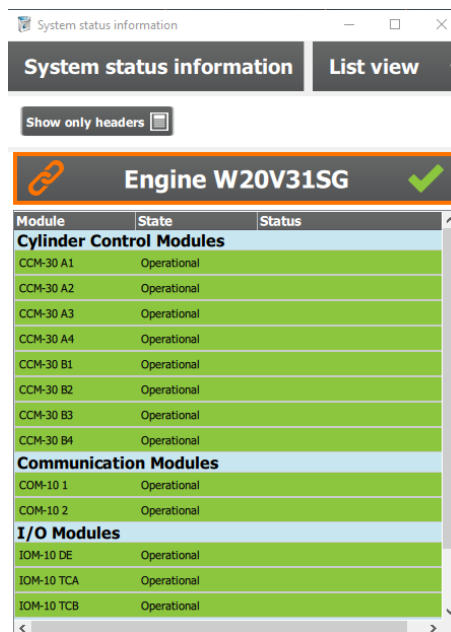


Figure 40 Application and Configuration download window

After the download process has finished, the system status information window indicates in what state each module is. During the download process, some modules

might not reboot automatically and require manually rebooting them up. If every module is in the Operational state, then the testing process can be started. **Figure 41** represents in what state each module was after the download process had finished.



Module	State	Status
Cylinder Control Modules		
CCM-30 A1	Operational	
CCM-30 A2	Operational	
CCM-30 A3	Operational	
CCM-30 A4	Operational	
CCM-30 B1	Operational	
CCM-30 B2	Operational	
CCM-30 B3	Operational	
CCM-30 B4	Operational	
Communication Modules		
COM-10 1	Operational	
COM-10 2	Operational	
I/O Modules		
IOM-10 DE	Operational	
IOM-10 TCA	Operational	
IOM-10 TCB	Operational	

Figure 41 W20V31SG system status after the download process

Now that the testing environment has been established and all the modules rebooted and that they are in the Operational state, the testing process can begin. Next section **5.3 MiniRig test results** explain what was tested and what were the results.

5.3 MiniRig test results

The testing mainly focused on how the COM-10 1 module cooperated with the other modules that were on the MiniRig. Once the connection has been established to the MiniRig from UNITool, the EDL starts printing every machinery protection alarm that was triggered from simulated sensors, actuators, and solenoids. On top of that, EDL informs if there are any modules missing from the WHSR ring. A snippet of the EDL log in **Figure 42** displays the WHSR communication status of the test environment.

PC Time	Module	Type	Message	Parameter Data
2019-04-03 15:13:07.585	COM-10 1	WHSR	Module appeared on the HSR ring	1, 2, 4
2019-04-03 15:13:07.585	COM-10 1	USM	Module back to normal	CCM-30 A4
2019-04-03 15:13:07.585	COM-10 1	WHSR	Module appeared on the HSR ring	1, 2, 6
2019-04-03 15:13:07.585	COM-10 1	USM	Module back to normal	CCM-30 B2
2019-04-03 15:13:07.585	COM-10 1	WHSR	Module appeared on the HSR ring	1, 2, 5
2019-04-03 15:13:07.585	COM-10 1	USM	Module back to normal	CCM-30 B1
2019-04-03 15:13:07.585	COM-10 1	WHSR	Module appeared on the HSR ring	1, 2, 8
2019-04-03 15:13:07.585	COM-10 1	USM	Module back to normal	CCM-30 B4
2019-04-03 15:13:07.585	COM-10 1	WHSR	Module appeared on the HSR ring	1, 2, 7
2019-04-03 15:13:07.585	COM-10 1	USM	Module back to normal	CCM-30 B3
2019-04-03 15:13:07.585	COM-10 1	WHSR	Module appeared on the HSR ring	1, 2, 3
2019-04-03 15:13:07.585	COM-10 1	USM	Module back to normal	CCM-30 A3
2019-04-03 15:13:07.585	COM-10 1	WHSR	Module appeared on the HSR ring	1, 2, 6
2019-04-03 15:13:07.585	COM-10 1	USM	Module back to normal	CCM-30 B2
2019-04-03 15:13:07.585	COM-10 1	WHSR	Module appeared on the HSR ring	1, 2, 8
2019-04-03 15:13:07.585	COM-10 1	WHSR	Module appeared on the HSR ring	1, 2, 5
2019-04-03 15:13:07.585	COM-10 1	USM	Module back to normal	CCM-30 B1
2019-04-03 15:13:07.591	COM-10 1	USM	Module back to normal	IOM-10 TCA
2019-04-03 15:13:07.591	COM-10 1	USM	Module back to normal	CCM-30 B3
2019-04-03 15:13:07.594	COM-10 1	WHSR	Module appeared on the HSR ring	1, 2, 1
2019-04-03 15:13:07.594	COM-10 1	USM	Module back to normal	CCM-30 A1
2019-04-03 15:13:07.594	COM-10 1	WHSR	Module appeared on the HSR ring	1, 2, 7
2019-04-03 15:13:07.594	COM-10 1	WHSR	Module appeared on the HSR ring	1, 2, 4
2019-04-03 15:13:07.594	COM-10 1	USM	Module back to normal	CCM-30 A4
2019-04-03 15:13:07.600	COM-10 1	USM	Module back to normal	IOM-10 TCA

Figure 42 EDL capture during final testing

UNITool provides a similar layout (**Figure 43**) as seen when operating with the Local Display Unit that is mounted next to the engine and has almost the same functionality. With this Engine Control Panel window, the developer can see what machinery protections where triggered, start or stop the engine, and see the status and different values measured from the engine. In **Figure 43**, the engine is in the running state with a steady speed of 750 *rpm*.

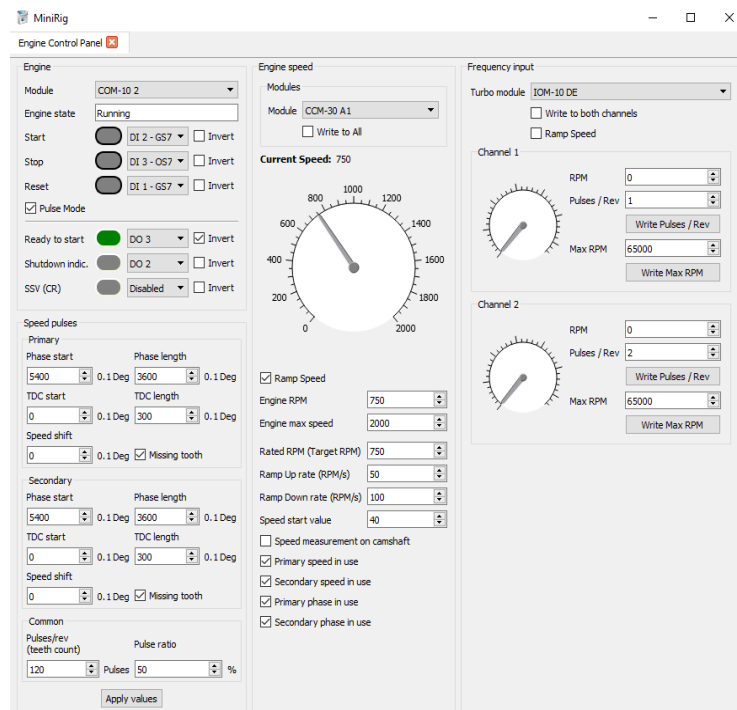


Figure 43 MiniRig Engine Control Panel window

First thing tested was what UNITool displays when accessing COM-10 2s HSR diagnostics on ring failure detection. **Figure 44** lists all the frame latencies found from all the HSR compatible modules, except for IOM-10 modules, as they do not support HSR but only CAN.

The first thing to note in **Figure 44** is that COM-10 1 was not found on the ring. That was an expected result, as it is connected to the external Cisco switch via the uplink port on the switch and acting as a SAN device and not as a DANH node on the HSR ring.

The average frame latency in LAN or local area network between the PC and router is usually less than 1 *ms*. The reason why the frames on **Figure 44** take much longer is related to the dual kernel environment where Xenomai handles the real-time communication and Linux all the non-real-time tasks. The values that are measured here is done by Linux and not Xenomai, and as such, they are not time-critical. The frame delays are measured between each node, e.g. COM-10 2 → CCM-30 A2 with the time delay of 2.31 *ms* when the frame is sent from the HSR1 port.

Ring failure detection [COM-10 2]				
<input type="checkbox"/>	Module ID	Available on links (BitMask)	HSR1 Frame Delay (ms)	HSR2 Frame Delay (ms)
1	COM-10 1	None	0.00	0.00
2	CCM-30 A1	HSR 1+HSR 2	2.31	3.97
3	CCM-30 A2	HSR 1+HSR 2	1.76	5.96
4	CCM-30 A3	HSR 1+HSR 2	1.59	5.50
5	CCM-30 A4	HSR 1+HSR 2	1.23	2.10
6	CCM-30 B1	HSR 1+HSR 2	5.96	6.43
7	CCM-30 B2	HSR 1+HSR 2	2.25	2.98
8	CCM-30 B3	HSR 1+HSR 2	5.35	7.65
9	CCM-30 B4	HSR 1+HSR 2	5.01	4.02

Figure 44 Ring failure detection on COM-10 2 module

Figure 45 below exhibits the general WHSR comparison between COM-10 1 and COM-10 2. The figure displays the same number of modules as before because IOM-10s and COM-10 1 are not on the HSR ring so that totals to 9 modules. The figure also shows which COM-10 is the acting router for CAN modules and in this case, it is COM-10 2 because COM-10 1 is not on the HSR ring.

Update	Symbol Name	Value	Unit	Limits	Module
<input type="checkbox"/>	Acting router for CAN modules	False	Boolean	-	COM-10 1
<input type="checkbox"/>	Enable latency measurement	False	Boolean	-	COM-10 1
<input type="checkbox"/>	Number of modules for latency	32		0 / 255	COM-10 1
<input type="checkbox"/>	HSR ring detected modules	9		0 / 255	COM-10 1
<input type="checkbox"/>	Number of HSR connection points	4		0 / 255	COM-10 1
<input type="checkbox"/>	Acting router for CAN modules	True	Boolean	-	COM-10 2
<input type="checkbox"/>	Enable latency measurement	False	Boolean	-	COM-10 2
<input type="checkbox"/>	Number of modules for latency	32		0 / 255	COM-10 2
<input type="checkbox"/>	HSR ring detected modules	9		0 / 255	COM-10 2
<input type="checkbox"/>	Number of HSR connection points	4		0 / 255	COM-10 2

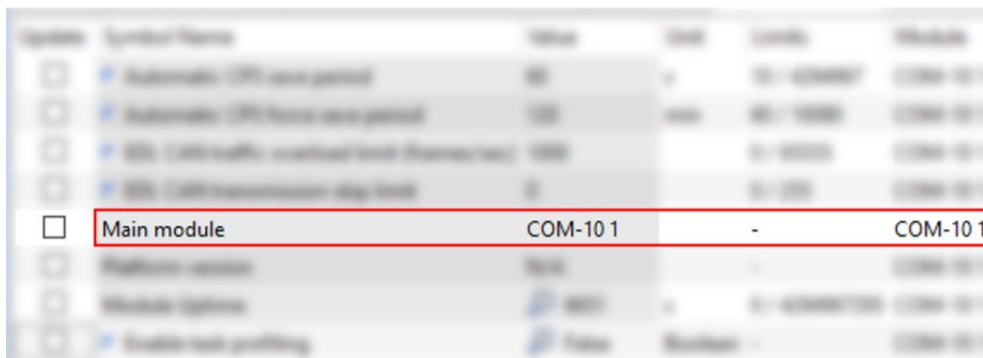
Figure 45 WHSR general information

The engine package W20V31SG where the “V” stands on how the cylinders and pistons are aligned. In a V-engine, each pair of corresponding pistons from each bank of cylinders share one crankpin on the crankshaft. V-engines generally are more compact than straight engines with cylinders of the same dimensions and number. The same bank system configuration can also be seen from **Figure 46**, where all the other modules are in the A-bank, except CCM-30 B1 - B4 modules, which are in the B-bank. The figure also gives a good indication of whether any modules are missing from the system. For this testing purposes, the most significant module to see was COM-10 1.

Module	HardwareID	ModuleGroup	TargetModuleID	Bank	Index
1	1	Communication gateway module group	COM-10 1	0	0
2	2	Communication gateway module group	COM-10 2	0	1
3	1	Cylinder control module group	CCM-30 A1	0	0
4	2	Cylinder control module group	CCM-30 A2	0	1
5	3	Cylinder control module group	CCM-30 A3	0	2
6	4	Cylinder control module group	CCM-30 A4	0	3
7	5	Cylinder control module group	CCM-30 B1	1	0
8	6	Cylinder control module group	CCM-30 B2	1	1
9	7	Cylinder control module group	CCM-30 B3	1	2
10	8	Cylinder control module group	CCM-30 B4	1	3
11	1	I/O module group	IOM-10 DE	0	0
12	7	I/O module group	IOM-10 TCA	0	1
13	11	I/O module group	IOM-10 TCB	0	2

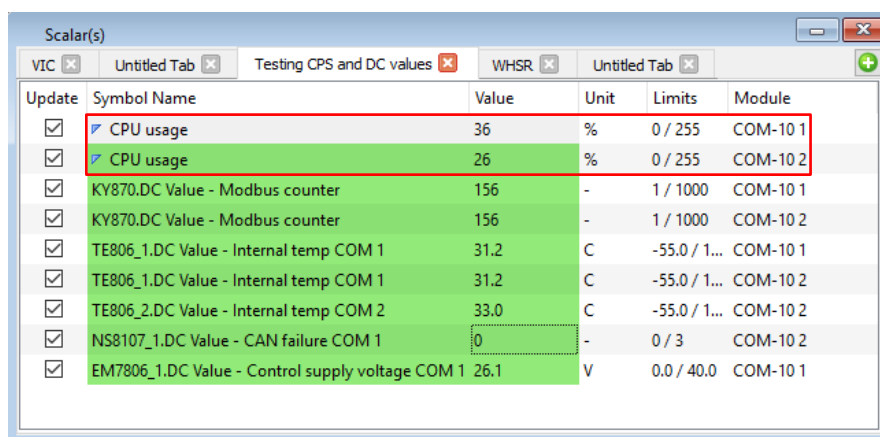
Figure 46 Module detection from COM-10 1

It is also crucial to be able to read internal values from COM-10 1, while connected to the MiniRig from UNITool. **Figure 47** and **Figure 48** demonstrate that it is in fact possible. The first figure points that COM-10 1 module has taken the main role in the communication system, while the second figure compares different internal values captured from COM-10 1 and COM-10 2. The most significant values to compare are the CPU usages on both modules. Both modules share the same internal hardware components but the reason why COM-10 1 has a 10% higher CPU usage that it is single-handedly responsible for 20 different applications while COM-10 2 only monitors and controls the variable exhaust valve closing application.



Update	Symbol Name	Value	Unit	Limits	Module
<input type="checkbox"/>	Addressable CPU usage period	60	s	0 - 600000	COM-10 1
<input type="checkbox"/>	Addressable CPU usage period	15	min	0 - 10000	COM-10 1
<input type="checkbox"/>	20% Addressable control limit threshold	100	%	0 - 10000	COM-10 1
<input type="checkbox"/>	20% Addressable stop limit	0	%	0 - 100	COM-10 1
<input type="checkbox"/>	Main module	COM-10 1	-	-	COM-10 1
<input type="checkbox"/>	Module status	OK	-	-	COM-10 1
<input type="checkbox"/>	Module update	400	s	0 - 600000000	COM-10 1
<input type="checkbox"/>	Module test profiling	True	Boolean	-	COM-10 1

Figure 47 Different values read from COM-10 1 module



Update	Symbol Name	Value	Unit	Limits	Module
<input checked="" type="checkbox"/>	CPU usage	36	%	0 / 255	COM-10 1
<input checked="" type="checkbox"/>	CPU usage	26	%	0 / 255	COM-10 2
<input checked="" type="checkbox"/>	KY870.DC Value - Modbus counter	156	-	1 / 1000	COM-10 1
<input checked="" type="checkbox"/>	KY870.DC Value - Modbus counter	156	-	1 / 1000	COM-10 2
<input checked="" type="checkbox"/>	TE806_1.DC Value - Internal temp COM 1	31.2	C	-55.0 / 1...	COM-10 1
<input checked="" type="checkbox"/>	TE806_1.DC Value - Internal temp COM 1	31.2	C	-55.0 / 1...	COM-10 2
<input checked="" type="checkbox"/>	TE806_2.DC Value - Internal temp COM 2	33.0	C	-55.0 / 1...	COM-10 2
<input checked="" type="checkbox"/>	NS8107_1.DC Value - CAN failure COM 1	0	-	0 / 3	COM-10 2
<input checked="" type="checkbox"/>	EM7806_1.DC Value - Control supply voltage COM 1	26.1	V	0.0 / 40.0	COM-10 1

Figure 48 Different values read from COM-10 1 and COM-10 2

6 CONCLUSIONS AND FURTHER IMPROVEMENTS

The work as a whole was suitable for the thesis as it was both challenging, and at the same time, it gave me an inside look on how Wärtsilä uses the HSR protocol in their UNIC2 communication system. The project also showed me that I should further develop my skills in FPGA development, as the need for more FPGA-based solutions are increasing.

The final result of the project will not be used in commercial purposes but instead only as proof of concept, as the HAL version used in the project was outdated. It is up to the executives and stakeholders in Wärtsilä to decide whether to move forward with this solution and incorporate it in Wärtsilä engines.

The gathered results in chapter **5 Testing Environment and Results** confirm that it is, in fact, possible to use an external HSR compatible switch when the FPGA resources on the COM-10 1 module are crucially limited. The results also confirm that the time latencies for the frames to move between the modules and the external switch are within the UNIC2 communication standard. The second requirement given by Wärtsilä was the reduction of used FPGA resources. That was also met, as the requirements dropped by more than 40% giving a possibility for further FPGA application development.

6.1 Further improvements

Although a functional system was achieved, there are further improvements that could be done.

First and foremost, the most significant improvement for the system would be to use the latest system platform version, as that would support Wärtsilä's newest 6-series engines. For this to work, it would require extensive remodification to HAL and WMAP source files to achieve real-time communication from the Service Port.

The external switch used in the project does not support PTP (Precision Time Protocol) when the switch is configured in HSR mode. For achieving a fully functional system, a switch with PTP support over HSR is required. PTP uses Best Master Clock algorithm (BMC) for selecting the GrandMaster and Master devices. The functionality of PTP is that the whole network (modules) are synchronised to one common clock, which is the GrandMaster.

Finally, the Enhanced Diagnostic Log in UNITool would require more comprehensive testing to confirm that it works without issues. During final testing with the MiniRig, some messages sent from the COM-10 1 module were not detected in EDL. After later investigations using only two COM-10s and the external switch, all the messages were received to the EDL. Given the time limit of the project, I cannot conclude what caused the unreliability issues as it could be related to the older system platform version used in the project.

REFERENCES

- /1/ UNIC - the reliable solution for robust industrial control. https://cdn.wartsila.com/docs/default-source/power-plants-documents/technology/electrical-and-automation/unic---the-reliable-solution-for-robust-industrial-controls.pdf?sfvrsn=584ce345_2
- /2/ This is Wärtsilä. <https://www.wartsila.com/about>
- /3/ Wärtsilä Marine. <https://www.wartsila.com/marine>
- /4/ Wärtsilä Energy. <https://www.wartsila.com/energy/what-we-do>
- /5/ A Strategic Approach to Optimising Power Plant Operations and Performance. https://cdn.wartsila.com/docs/default-source/services-documents/white-papers/wartsila-bwp---a-strategic-approach-to-optimising-power-plant-operations-and-performance.pdf?sfvrsn=dc2dc545_8
- /6/ Wärtsilä Services. <https://www.wartsila.com/services/services>
- /7/ Operations of the Board of Management. <https://www.wartsila.com/investors/governance/board-of-management/operations-of-the-board-of-management>
- /8/ Wärtsilä UNIC. <https://cdn.wartsila.com/docs/default-source/product-files/engines/brochure-o-e-unic-ms.pdf>
- /9/ Tomi, V. 2017. Next Generation UNIC automation system enables Wärtsilä 31 performance. <https://www.wartsila.com/twentyfour7/in-detail/next-generation-unic-automation-system-enables-wartsila-31-performance>
- /10/ LDU-30. <https://red-dot-21.com/p/design-products/industrial-equipment/diagnosis-control-units/ldu-30/>
- /11/ High-Availability Seamless Redundancy in the Factory Network Design Guide. <https://www.cisco.com/c/en/us/td/docs/solutions/Verticals/Machines/HSR/ConnMach-HSR/ConnMach-HSR.html>
- /12/ Cisco Overview. <https://newsroom.cisco.com/overview>
- /13/ Cisco Industrial Ethernet 4000 Series Switches Data Sheet <https://www.cisco.com/c/en/us/products/collateral/switches/industrial-ethernet-4000-series-switches/datasheet-c78-733058.html>
- /14/ Cisco IE-4000-8T4G-E Industrial Managed Layer 3 Ethernet Switch. <https://shop.elkome.com/en/ie-4000-8t4g-e-industrial-managed-layer-3-et.html>
- /15/ VLAN tagging in Kali Linux 2.0. <https://www.adamcouch.co.uk/vlan-tagging-kali-linux-2-0/>

/16/ XST Synthesis Overview. https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/ise_c_using_xst_for_synthesis.htm

/17/ Implementation Overview for FPGAs. https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/ise_c_implement_fpga_design.htm

/18/ ABOUT THE YOCTO PROJECT. <https://www.yoctoproject.org/about/>