Western Norway University of Applied Sciences

BACHELOR'S THESIS

Handling Real-Time Data from NMEA 2000

Mathias Madsen Aaøyen
Simon Skarbø
Andréas Øihaugen

Bachelor's programme in Computing
Department of Computer science, Electrical engineering and Mathematical sciences/Faculty of Technology, Environmental and Social Sciences/Computing
Atle Geitung
Submission Date: 13.05.2024

I confirm that the work is self-prepared and that references/source references to all sources used in the work are provided, cf. Regulation relating to academic studies and examinations at the Western Norway University of Applied Sciences (HVL), § 10.

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel: Handling Real-Time Data from NMEA 2000	*Dato:* *13.05.2024*
Forfatter(e): Mathias Madsen Aaøyen, Simon Skarbø og Andreas Øihaugen	*Antall sider u/vedlegg: 65* *Antall sider vedlegg: 56*
Studieretning: Dataingeniør	*Antall disketter/CD-er:* 0
Kontaktperson ved studieretning: Atle Geitung	*Gradering:* Ingen
Merknader:	

Oppdragsgiver:	*Oppdragsgivers referanse:*
Oppdragsgivers kontaktperson:	*Telefon:*

Sammendrag:

Denne bacheloroppgaven utforsker utviklingen av en løsning for å muliggjøre fjernovervåking av systemer på marine fartøy ved bruk av NMEA 2000-nettverket. Det primære målet er å tilby tilgang til sanntids- og historiske data fra fartøyets systemer, og overvinne begrensningene ved at disse dataene kun er tilgjengelige lokalt. Prosjektet går dypt inn i design, implementering, og testing av et system som integrerer med eksisterende NMEA 2000-infrastruktur for å tilby omfattende overvåkningskapasiteter for data. Gjennom utviklingen av dette systemet, adresserer oppgaven de tekniske utfordringene med å trekke ut og håndtere NMEA 2000-data og vurderer bruken og mulig innvirkning av systemet på fjernovervåkning. Dette oppsummerer reisen fra den innledende fasen til gjennomføring og refleksjonen over prosjektresultatene.

Stikkord:

NMEA 2000	Remote Monitoring	Data Integration

Høgskulen på Vestlandet, Fakultet for teknologi, miljø- og samfunnsvitenskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

Preface

This report documents the work done for our project “Handling Real-Time Data from NMEA 2000”, which concludes our three-year journey at Western Norway University of Applied Sciences (HVL) and marks the completion of our bachelor’s degree in computing.

We would like to extend our gratitude to the folks at Sensar Marine for providing us with an office and access to their equipment. Their expertise has been invaluable to us as we navigated the unfamiliar world of microcontrollers and marine technology. We are also extremely grateful for their coffee maker, which kept us going during countless hours of project work. Without it, our thesis journey would have been far less caffeinated and cheerful.

We would also like to express our heartfelt thanks to Atle Geitung for his invaluable advice and guidance throughout the project.

Finally, we express our heartfelt gratitude to our friends and family who took the time to read our work and provided us with valuable feedback. Your encouragement and support were greatly appreciated.

Summary

This bachelor thesis explores the development of a solution to enable remote monitoring of marine vessel systems using the NMEA 2000 network. The primary goal is to provide accessibility to real-time and historical data from vessel systems, overcoming the limitations of this data only being accessible locally. The project delves into the design, implementation, and testing of a system that integrates with existing NMEA 2000 infrastructure to provide comprehensive data monitoring capabilities. Through the development of this system, the thesis addresses the technical challenges of extracting and managing NMEA 2000 data and evaluates the application and potential impact of the system on remote monitoring. This summarises the journey from the initial starting phase to the execution and reflection on the project outcomes.

Table of contents

1 INTRODUCTION

This chapter is designed to provide readers with a thorough understanding of the project's background. It will begin by giving an overview of the context in which the project was developed. The chapter will then introduce the project's motivation and stakeholders. The narrative will then delve into a detailed explanation of the problem, clearly outlining its goals. Finally, the chapter will provide a roadmap of the report's layout, explaining the purpose and contents of each subsequent chapter.

1.1 Context

The modern world is full of microcontrollers that gather and transmit data to create exceptional technological solutions. They are utilised in various devices, such as medical equipment, home appliances, robots, and IoT devices [1], [2].

Figure 1.1: TMS 1000 by Texas Instruments [3].

Single-chip computers started in calculators with the introduction of TMS1000 by Texas Instruments in 1974. Before the TMS1000, the norm was utilising several chips to, for example, make a handheld calculator. A few years later, in 1976, Intel launched the first microcontroller. A microcontroller is a single-chip computer that contains all the necessary components for storage, IO, e.g., in one chip. In 1977, Motorola launched the MC6801 chip, which introduced a serial communication interface on a microcontroller unit (MCU). In the following years, Motorola, Texas Instruments, Intel, and other manufacturers like Phillips, Siemens, Hitachi, and NEC all had their line of microcontrollers and were competing to integrate more memory and functionality into their chips. These microcontrollers entered different everyday products like cameras and cars [4].

Even though a microcontroller is interesting, connecting multiple microcontrollers allows for even more advanced operations. One way of connecting microcontrollers is using serial communication. One protocol for using serial communication was invented by Robert Bosch GmbH in the early 90s and was made to be used with automobiles. This protocol was first seen in automobiles in the early to mid-90s. The International Organization for

Standardization standardised this protocol in the document ISO 11898 in 1993. The ISO 11898 standard has been revised and amended multiple times after 1993 and split into different parts among other ISO 11898-1 “Data link layer and physical signalling”, which was not included in the original specification published by Robert Bosch GmbH in 1991 [5], [6], [7].

The CAN bus standard and the ISO 11898 were conceived for automotive use, where different microcontrollers in an automobile can exchange information quickly and reliably through a serial interface. Other standards aimed at varying applications than directly automotive have incorporated the ISO standard into their own; one example is the J1939 standard from the Society of Automobile Engineers (SAE). This standard is for industrial equipment like tractors, trucks, and buses. The J1939 standard has, however, been incorporated by yet another standard called NMEA 2000, created by the National Marine Electronics Association (NMEA). NMEA 2000 enables multiple components from different manufacturers to communicate over a serial bus using the J1939 standard with some adaptations. This communication allows, e.g. an engine to report its RPM[1] to the network and a Multi-Function Display (MFD) to display the RPM without the MFD being the same manufacturer or having to have a translation box in between. [8], [9], [10]

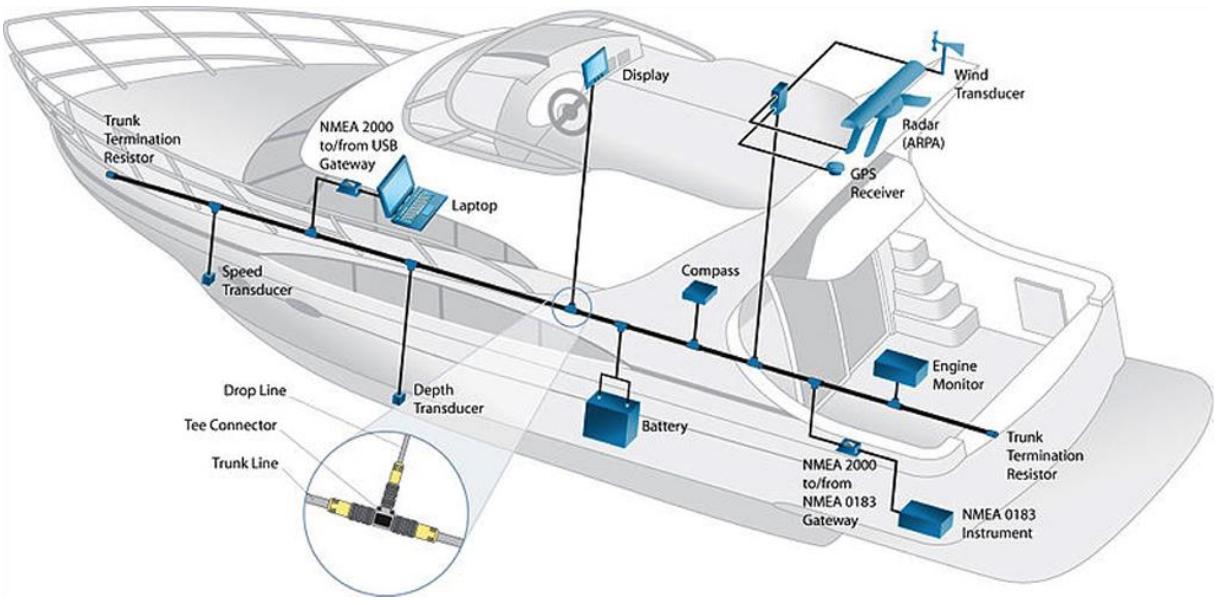


Figure 1.2: An illustration of a boat with an NMEA 2000 network. [11]

[1] Revolutions Per Minute, a measure of how fast the engine is spinning.

On the automotive side, multiple scanners use the CAN standard to retrieve information from automobiles, search for fault codes, and, in some cases, show live information, such as oil pressure from the automobile while in operation. Depending on the equipment installed on a vessel, multiple parameters like wind speed, engine RPM, position, and fuel level, among others, could be transmitted on the NMEA 2000 network. This information is usually only displayed on a Multi-Function Display. Although some products on the market can display information without using an MFD, these typically require the user to connect a computer to the network through a dongle like the NGW-1 from Actisense [12]. This requirement for physical connection to the NMEA 2000 network with a computer to retrieve data present in most of the solutions today makes access to the data impractical for the average user. There are various products available today that aim to provide users with more information about the status of their vessel even when they are not on board. One such product is the "Boat Monitor with Bilge Sentry" from Sensar Marine. This product doesn't connect directly to any vessel components or network. Instead, it uses sensors built into the product and connects directly to the batteries to convey important information to the user through a mobile application. This approach ensures that users receive critical information regardless of the systems installed on their vessel. However, there is still potential to connect the Sensar Marine product to other vessel components, such as the fuel tank or any other NMEA 2000 capable system, which has yet to be explored.

Figure 1.3: Faults read from CAN bus on vehicle.

1.2 Motivation

A large amount of information is being transmitted on the NMEA 2000 network, and that information has significant untapped potential[13].

Vessel owners can monitor their vessels using the data available from the NMEA 2000 network. This data can be viewed live on a multi-function display, and the different components can use data from each other. For example, GPS can provide time and date information for echolocation displays. While the data is available on the vessel, it is not easily accessible to the user remotely or with historical data. Remote access to historical data can provide numerous benefits. For instance, users can analyse fuel consumption, RPM, and

voyage details to identify fuel-saving opportunities.[14]. Additionally, it can provide insights into whether a vessel requires bottom paint to enhance efficiency[15]. Furthermore, if fuel level data is transmitted through the NMEA 2000 network, one can quickly obtain information on the remaining fuel in the tank. Having this information available could result in environmental and economic benefits.

As an owner of a vessel or a manager of a fleet, you would want to keep an eye on the status of your vessel or vessels at any given time. This could imply knowing the weather conditions, the vessel’s position, and how the engines run. For example, if you are a company that rents out boats, you would likely want to know whether your customer is using your boats appropriately and safely or not.

The ease of access to the vessel's information could also optimise customer service. If a marine vessel technician had access to the data recorded on the NMEA 2000 network, he could, in some situations, discover flaws with the vessel or engine remotely. This could significantly impact customer support efficiency and economic benefits for the vessel owner and those offering customer service[16].

While there are many alternatives for monitoring your marine vessel, they usually require the owner to install a new electronic system. However, if you already have a system like NMEA 2000 installed, connecting one more device to the system would be simple, and some vessels even have NMEA 2000 capable devices without the NMEA 2000 network connected and would only require a few components to create a functioning network[10].

1.3 Project Stakeholders

The bachelor group defined the bachelor problem; therefore, no external project owners exist. However, an interested party, Sensar Marine, has a product that would benefit from incorporating NMEA 2000 data. This interest has become a cooperation between the bachelor group and Sensar Marine. The cooperation will be elaborated on in later chapters.

1.4 Problem Description and Goals

A successful solution would enable owners of marine vessels to remotely view real-time and historical data about their vessel's status through an efficient information transfer system from the vessel's onboard systems.

This bachelor thesis's primary objective/goal is to develop a solution that enables vessel owners to remotely access real-time and historical data from their vessel's onboard systems, overcoming the current limitation of data being available only locally.

To achieve this, the thesis is structured around the following subgoals:

- *Develop a method to read information from an NMEA 2000 network, ensuring compatibility and efficient data extraction from a marine vessel´s onboard systems.*
- *Implement a system to manage the information collected from the NMEA 2000 network, including processing, storing and transmitting the data.*
- *Provide a solution for displaying the data received.*

The research question for our bachelor thesis has been narrowed down to:

- *How can data from the NMEA 2000 network be efficiently read and extracted beyond local access to enable remote monitoring of a marine vessel´s systems?*

1.5 Report Structure

The 1st chapter of this report introduces the context of the thesis and the thesis problem. Chapter 2 gives an in-depth description of the project with available resources. To understand the standard upon which this project was built, the 3rd chapter is introduced. Following this, Chapter 4 provides a thorough run-through of possible solutions, with the selected solution described in greater detail in Chapter 4.2. The explorative nature of the project demands a chapter to discuss tool selection and evaluation, as this is a major part of the project and is elaborated upon in Chapter 5. The 6th chapter describes the results of the product produced during the thesis project. Chapters 7, 8 and 9 are dedicated to discussing the thesis project's results and concluding on the project findings: the last two chapters, 10 and 11, list, respectively, references and attachments.

2 PROJECT DESCRIPTION

This chapter delves into the comprehensive background of the project. The chapter starts by examining the practical background, the initial requirements, and the initial idea for the solution. Additionally, the chapter discusses the project's scope limitations, which are essential for maintaining a clear and focused approach to the project's development. To ensure the project progresses in an optimal environment, the chapter will cover all the crucial resources, including workplace, equipment, documents and software, and advising. Finally, the chapter concludes by discussing relevant literature about the project description.

2.1 Practical Background

2.1.1 Previous Work

No direct previous work has been discovered through the research phase of this thesis project, as will be further explored in Chapter 2.4. Some individuals and communities have created libraries and texts, making the individual parts of this project possible. An example is the individual Timo Lappalainen, who created a C++ library to handle conversion from NMEA 2000 messages to human-readable information; this library and similar projects will be crucial in building the solution for the thesis problem [17].

Some commercial and non-commercial actors touch on the thesis problem, like retrieving data from the NMEA 2000 network. An example is the company Actisense, which offers solutions for physically reading data to a computer connected to the network or logging the data and retrieving it later [12], [18], [19]. Another example is the open-source project Open Skipper, which provides software for displaying collected NMEA 2000 data on a computer [20]. None of the solutions found during research completely answers the thesis problem. Most are proprietary, require specific software, or do not provide live data export from the network to a remote device.

2.1.2 Initial Requirements Specification

The solution should be able to read and interpret information exchanged between devices connected to an NMEA 2000 network. Once the information is read, it must be decoded and transmitted to an external storage. This storage will serve as a central repository, allowing a client to access and view the data received from the NMEA 2000 network.

2.1.3 Initial Idea for Solution

The initial idea for the solution is to create a device that will connect to an NMEA 2000 network on a marine vessel using wires. When connected, the device should be able to intercept the messages transmitted across the network. Once the messages are read, the device must understand them by decoding and storing them.

After the device has stored the messages, it should be able to send the information via a mobile network, allowing the user to view the messages remotely.

2.2 Scope Limitations

The bachelor's thesis problem is broad enough that the group can select the most interesting aspects and focus our attention therein. Still, it also forces the group to look for libraries and commercial solutions to the parts of the project where suitable solutions exist. This helps the group end up with a functional prototype by the end, which the project has been limited to.

Due to the constraints, this project will focus on the more necessary parts despite the availability of numerous existing functions that could be implemented into the final solution. The focus is shifted away from these additional features, described in a later chapter, with thoughts on implementing these. Instead of exploring all these features, the project will first focus on creating a working solution.

2.3 Resources

This section of the report delves into all the relevant resources that enable the group to advance efficiently with the project. These include the workplace, documents and software, equipment, and advice. The group will ensure that the project proceeds seamlessly by utilising these resources.

2.3.1 Workplace

Throughout the project, the bachelor group can utilise an office Sensar Marine and GC Rieber provided. This dedicated workspace has two desks, a whiteboard and access to a coffee maker.

Figure 2.1: The provided office.

Additionally, a significant advantage of the office is its proximity to the engineers, allowing immediate consultation.

2.3.2 Equipment

Sensar Marine plays a pivotal role during the project's runtime. They supply the microcontrollers on which the solution is developed and provide the necessary equipment and parts to assemble an NMEA 2000 network.

Additionally, the project group can order spare parts necessary for the solution's development and maintenance.

2.3.3 Documents and Software

To create a solution, detailed specifications and documentation about the NMEA 2000 network, specifying data formats and communication protocols will be needed.

Predefined libraries from third parties will be used to read and filter the CAN messages and associate various codes with their meanings.

A simulator will be used to emulate an NMEA 2000 network. This will be very useful during the solution's testing as it allows testing inside the office rather than going down by the water and connecting to a vessel.

2.3.4 Advising

During the project's research and development, the bachelor group will access technical support from Sensar Marine employees. They can offer consultation regarding the technical aspects of the project's scope.

In addition to technical support, the bachelor group has access to an internal advisor provided by Western Norway University of Applied Sciences. This advisor will be available to answer any questions about writing a bachelor's thesis, including academic standards and report structure.

This dual support system will ensure that the bachelor group is well-equipped to navigate the project's technical and academic aspects.

2.4 Literature Survey

This chapter aims to find relevant literature about the thesis question and present relevant articles and books.

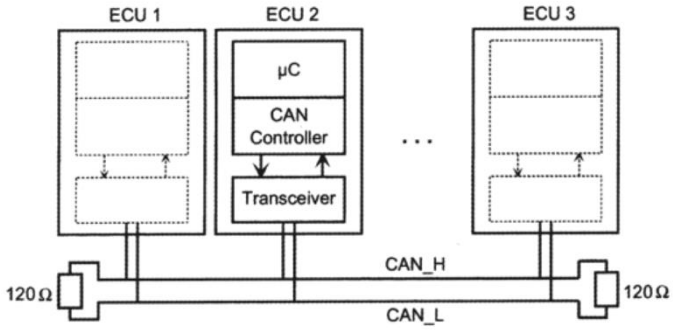


Figure 2.2: CAN main based network.[13]

Our first aim was to find as much literature as possible about the NMEA 2000 standard and its application. The available literature on this is scarce, but interesting articles are to be found. Most of the available literature is written by commercial actors explaining NMEA 2000 to sell or demonstrate their product. Interesting academic articles exist like "On the adaptation of CAN bus network for use in the ship electronic systems" written by Andrzej Piętak and Maciej Mikulski, published in the Polish Maritime Research Journal and the "NMEA Communication Standard for Shipboard Data Architecture" written by Srećko Krile, Danko Kezić and Franc Dimc published in the Naše more, International Journal of Maritime Science

& Technology [13], [21]. The articles explain the history of the NMEA standards, how the NMEA 2000 standard is utilised in ships, and how it works on a technical level.

The articles give a foundation of noteworthy topics like the J1939 standard by the Society of Automobile Engineers (SAE) and the CAN bus standard by Robert Bosch GmbH, and the ISO11783-3 by the International Organization for Standardization (ISO). Searching for articles and books relating to the CAN bus standard yields several thousand results compared to the NMEA 2000 standard. One noteworthy book that thoroughly explains the CAN bus standard is “Understanding and Using the Controller Area Network Communication Protocol” by Marco Di Natale, Haibo Zeng, Paolo Giusto and Arkadeb Ghosal[22]. This book, along with the articles on NMEA 2000, gives a solid understanding of the NMEA 2000 side of the thesis problem. Understanding NMEA 2000 and the requirements to extract the information from the network allowed the group to shift focus to the available hardware.

Initial searches for articles on choosing microcontrollers gave some immediate results, giving insight into some popular choices for boards like Arduino, Raspberry Pi, or ESP32-equipped alternatives [14], [15]. This prompted an interest in finding more information regarding the most exciting alternatives for microcontrollers, yielding some interesting articles [23], [24], [25], [26]. These articles and books gave the group a foundation to undertake the research question.

3 UNDERSTANDING NMEA 2000

This chapter will provide a detailed and thorough explanation of the NMEA 2000 protocol, a fundamental standard upon which this bachelor project is built. It will be divided into three parts, covering the protocol's background, practical applications and usage, and technical intricacies, offering the reader a comprehensive understanding.

3.1 Background

The evolution of the NMEA 2000 protocol is deeply rooted in the efforts of the US-based National Marine Electronics Association (NMEA), an organisation dedicated to advancing the marine electronics industry through developing standards and protocols. Since its inception in 1957, when a group of electronic dealers collaborated to improve the relationship with electronic manufacturers, the NMEA organisation has profoundly impacted the industry, playing a pivotal role in ensuring interoperability and compatibility among marine electronic devices. [27]

Figure 3.1: National Marine Electronics Association (NMEA) logo.[28]

Before the advent of the NMEA 2000 protocol, its predecessors laid the groundwork for marine electronics communication. The initial protocols to be defined and introduced were NMEA 0180 and NMEA 0182, which paved the way for the widely adopted NMEA 0183 in 1983. NMEA 0183 utilises a simple communication protocol (ASCII) with slow data transmission. The data transmission is based on one “talker” and one-to-many “listeners”, which limits the potential to create networks as there can only be one unit to transmit data per setup. The NMEA 0183 standard remained the industry norm for two decades. Although

NMEA 0183 is still a valid standard today, the advancement of technology and the need for faster and more robust communication protocols meant a newer and improved standard would make its way to the market. [21]

In response to the challenge of the industry needing a more advanced protocol, the National Marine Electronics Association initiated the development of a new standard that would address the shortcomings of its predecessor and meet the evolving needs of the marine industry. Unlike its predecessor, NMEA 2000 was designed with a focus on higher data speeds, increased capacity, and units within the network to transmit and receive data. By incorporating standard connectors and cables for use among all manufacturers, NMEA has established a *plug-and-play*[2] system with a real-time data stream that also enables backwards compatibility with the predecessor. This official release of the NMEA 2000 standard clarified matters for electronic manufacturers and led to its adoption as an international standard by the International Electrotechnical Commission (IEC) as IEC61162-3. [29]

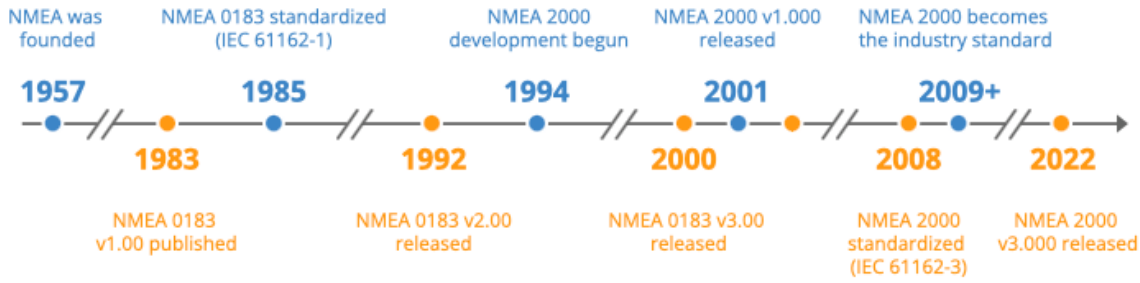


Figure 3.2: NMEA history. [30]

To ensure NMEA 2000 products are interoperable with other such products, NMEA facilitates a certification process. This involves a self-certification process that utilises mandated test equipment, which can be purchased from NMEA. The tests consist of two parts: the first is primarily focused on the physical hardware of the product, and the second focuses on functionality. The manufacturer may add the NMEA 2000 logo to the product if it becomes a certified NMEA 2000 product. The results are to be submitted to and validated by NMEA upon completion. However, some products may not be verified but are eligible for the *approved* classification. Products with this classification can implement NMEA 2000 and

[2] A feature that allows an electronic device to be used as soon as it is connected.

have the physical hardware tests from the certification process passed, resulting in compatibility with NMEA 2000-certified products. [28]

Major marine electronics manufacturers eventually began opting into this NMEA 2000 certification process, incorporating compatibility into their products, leading to widespread adoption of the standard. This acceptance by the industry helped establish NMEA 2000 as the most common standard for marine electronic systems, where a wide variety of platforms are utilising the protocol today. [31]

3.2 Practical Applications and Usage

Since NMEA 2000 was introduced to the market, it has become the backbone of leisure marine applications, providing a standardised and seamless integration between various devices and manufacturers. This has allowed the manufacturers to make products and devices that are easy to use, provide additional features, and be more reliable.
Today, various platforms use NMEA 2000, and some leading manufacturers consist of Garmin, Simrad, Lowrance, Raymarine, B&G, and Navico. [32]

An example use case will describe an NMEA network to provide further insight into how it can be and is being utilised today.

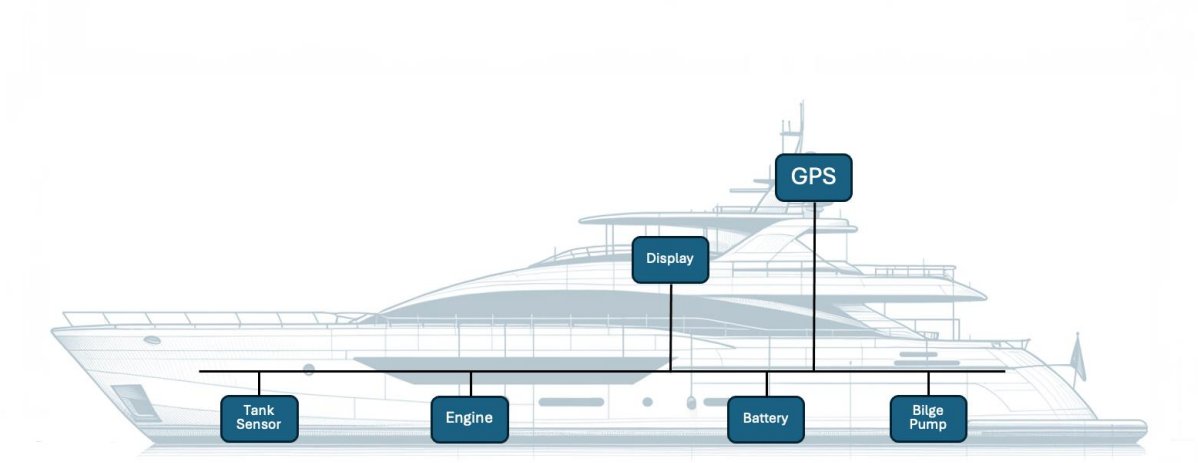


Figure 3.3: A figure showcasing how an NMEA 2000 network is connected throughout a vessel.

The figure above shows an example of a connected NMEA network with different components throughout the vessel. The network comprises products from various manufacturers, and the components do not necessarily communicate directly with each other; they are all connected to the same network. This connection facilitates information exchange

between the equipment, for example, allowing the GPS to communicate current time to other equipment connected to the network. One use case that is easy to imagine is connecting a multi-function display to the network and utilising it to show information from all connected components on one screen, like fuel status, battery status or wind direction.

3.3 Technical Explanation

To delve deeper into how NMEA 2000 functions, it is essential to understand how the network is constructed. When considering the NMEA 2000 standard within the broader context of network communication standards, it aligns with several layers of the OSI model[3]:

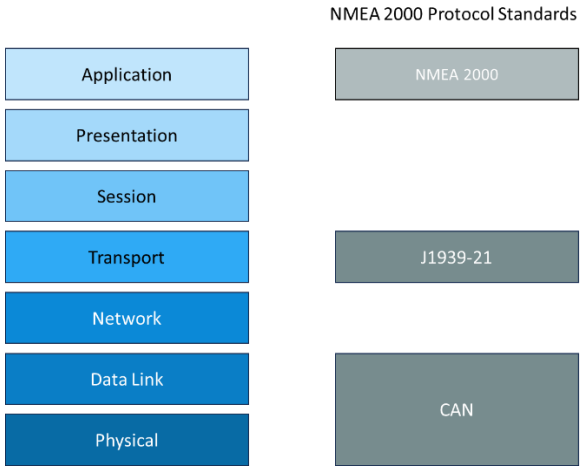


Figure 3.4: NMEA 2000 Protocol OSI Layer Mapping [34]

3.3.1 Physical Layer

The lowest layer of the OSI model is the physical layer, which is entirely defined by the NMEA 2000 standard and encompasses signal voltages, cables, and connectors. It incorporates 11898-2 (CAN) as the basis for NMEA 2000, with a standard baud rate of 250K, which is a vast improvement compared to NMEA 0183, which had a baud rate of 4800.

[3] The Open Systems Interconnection (OSI) model is a conceptual framework that splits network communication into seven abstraction layers. [33]

Additionally, it specifies details on the physical components of the network:

- T-connector
    - A T-connector is a three-way connector with one male and two female connectors. The T-connector is used to construct the NMEA 2000 CAN bus backbone linearly and connect the nodes to the network. The connectors are standardised DeviceNet 5-pin A-coded M12 connectors and are defined in two sizes by the NMEA 2000 standard. The smaller of the two is called “Micro” and is primarily used for networks of smaller vessels, while the larger is called “Mini” and is used for larger vessels.

Figure 3.5: A 5-pin A-coded M12 connector utilised in the NMEA 2000 network.

- Node
    - A node, also known as a device or equipment, is an electronic hardware capable of connecting to the NMEA 2000 network.
- Backbone
    - The backbone serves as the primary communication pathway connecting all devices on the network. The backbone must be constructed linearly to ensure proper network operation.
- Backbone Cable
    - The primary purpose of the backbone cable is to extend the backbone to nodes placed in various locations on the vessel, and a single backbone cable may have a maximum length of 100 meters. Like the backbone, the backbone cable needs to be constructed linearly and connected to the sides of two T-connectors.
- Drop Cable
    - A drop cable connects a node to the backbone of the NMEA 2000 network. The maximum length of a drop cable is 6 meters, and the maximum cumulative length is 78 meters. If the cable is under 6 meters, it may also be a backbone cable.

- Terminator
    - To ensure signal integrity, proper termination is required. A terminator is a 120-ohm resistor located at each end of the NMEA 2000 network, with one terminator being male and the other female.

Figure 3.6: A terminator utilised to set up the NMEA 2000 network.

- In-line Terminator
    - A special terminator that may be used instead of a male terminator. The in-line terminator allows for direct connection to the node at the end of the NMEA 2000 network and simplifies installation by eliminating the need for a T-connector, a male terminator, and a drop cable.
- Power Supply
    - The power supply provides power to the NMEA 200 network at 12 volts DC. Some of the nodes on an NMEA 2000 network are always turned on when power is present, and the power is therefore connected through a switch to avoid draining the battery. Nodes on an NMEA 2000 network are required to operate from 9 to 16 volts DC, with a nominal voltage of 9 volts DC.

Despite the user-friendly plug-and-play nature of the standard, setting up a functioning NMEA 2000 network requires a basic understanding of its components and proper installation techniques. [21]

3.3.2 Data Link and Transport Layer

The data link layer in NMEA 2000 is responsible for transmitting data across the communication channel and ensuring its reliable delivery. It utilises ISO1178-3 (ISOBUS), SAE J1939-21, and 11898-1 (CAN), incorporating the Controller Area Network (CAN) as the basis for the MAC (Media Access Control). [21]

Communication in NMEA 2000 networks operates on a decentralised architecture, eliminating the need for a central network controller. Each unit functions as a controller, following the protocols established by the standard. The units within an NMEA 2000 network also support self-configurable addressing, which dynamically determines their address through internal calculations, enhancing system efficiency.

Cyclic Redundancy Check (CRC) calculates a remainder by dividing frame contents by a prime binary divisor. This technique is proficient at detecting errors, allowing for the recognition of up to 6 errors per frame.

The Fast Packet is another crucial part of NMEA 2000 that falls under the data link layer. The NMEA 2000 standard allows multi-packet communication of up to 1785 bytes via the ISO 15765-2 (ISO TP) standard. However, Fast Packets were introduced as a more efficient transport protocol as the need for payloads that proceed 8 bytes but far from 1785 bytes was reoccurring. The medium-sized payloads can hold up to 223 bytes without transfer protocol delays.

In NMEA 2000, the transport layer establishes and maintains connections between devices and fragments and reassembles messages as needed. Like the Data Link layer, this can be achieved by the transport protocol used in ISOBUS/J1939 or via NMEA Fast Packets, as detailed above.

3.3.3 Application Layer

The standard fully defines the application layer in NMEA 2000 and encompasses approved parameter group numbers (PGNs), including provision for manufacturer-specific proprietary messages.

At the heart of the NMEA 2000 communication protocol is a structured message format that enables devices to share data in a standardised manner, as defined in Appendix B of the standard. NMEA 2000 utilises the concept of PGNs, based on the same concept from the J1939 standard, which is an 18-bit subset of the 29-bit extended CAN ID. These messages are transmitted over the network as a sequence of data frames, each composed of a 0 to 8-byte data field and a 29-bit identification field. The PGNs utilise unique identifiers which appear in the CAN identifier field. In addition to the identifiers, one may be able to find data fields containing:

- Frame Count
    - A CAN data frame only contains 8 bytes and may require more CAN frames. This parameter indicates whether the data can fit within a single 8-byte data frame or requires multiple frames.
- Priority
    - NMEA 2000 employs a priority-based message transmission system to deliver critical data promptly. Messages are assigned priority levels based on their importance, with higher-priority messages taking precedence over lower-priority ones. This prioritisation scheme ensures that critical data, such as navigational information and engine diagnostics, is transmitted promptly, minimising the risk of delays and data loss.
- Periodic Rate
    - This field contains information on whether the parameter group will be transmitted periodically.
- Destination
    - A PNG may be sent to one specific address, or it may be sent to all devices within the NMEA 2000 network. In the scenarios where the PNG is sent to all devices, each device chooses to accept or ignore this information.
- Query Support
    - Indicates which fields within the parameter group can be queried with optional query support.
- Field Definitions
    - Fields within a parameter group are all defined by a name.

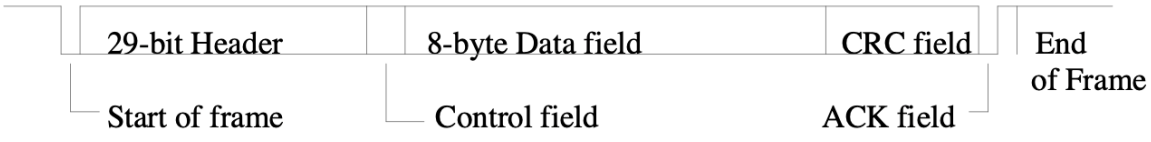


Figure 3.7: A simplified CAN frame.[28]

The numerical codes categorise data into specified groups, making it easier for devices to interpret incoming messages. Using PGNs, NMEA 2000 ensures data is organised and classified efficiently, enabling effective communication between devices. [21]

4 DESIGN AND SOLUTION

Based on the findings from previous chapters, this chapter will aim to provide an overview of the chosen solution. Additionally, it will discuss the alternative solutions and why they came up short. The project will also need the correct set of tools for development to be successful, and the criteria these tools should meet will be detailed in this chapter. Finally, the evaluation plan will be introduced and explained towards the end of the chapter.

4.1 Proposed Solution

This section of the report will systematically introduce and discuss each alternative solution proposed for the project's design.

4.1.1 Solution Requirements

As described in Chapter 1.4, Problem Description and Goals, it will be a complete solution. The group will start by outlining the essential components to understand what a solution should contain.

- Retrieval of data from the NMEA 2000 network.
    - Translation from the physical layer to the digital layer.
    - Handling of relevant data locally on an NMEA 2000 capable platform.
- Transfer data from the NMEA 2000 capable unit to data storage.
    - Use appropriate carrier from mobile unit to data storage.
- Receive and store data.
    - Endpoint for transferring data from remote unit to central data storage.
- Data presentation.
    - User interface where data is presented.

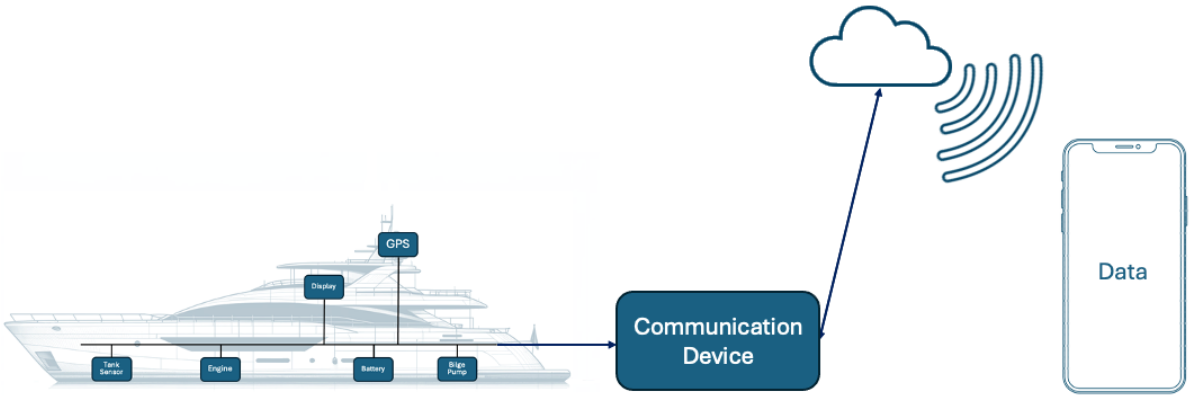


Figure 4.1: An illustration of how the desired solution is envisioned.

Building upon the components outlined above as a template, the subsequent subchapters will each delve into a specific solution to address the challenges presented in the problem description in Chapter 1.4.

4.1.2 Alternative Solution 1

Alternative solution: one will utilise as many *off-the-shelf* [4] components and solutions as possible.

- Retrieval of data from the NMEA 2000 network.
  - Actisense NGW-1[5] or other commercial NMEA 2000 data retrieval unit.
- Transfer data from the NMEA 2000 capable unit to data storage.
  - Of the shelf internet solution, e.g. satellite or cellular connection.
- Receive and store data.
  - InfluxDB[6] or other Time Series Database.
- Data presentation.
  - Grafana[7] or other solution for presenting collected data.
  - Possible custom solution for presentation.

[4] Products, solutions, or materials that are readily available for purchase and use without the need for customization.
[5] Devices that can be connected to a computer and an NMEA 2000 network. With compatible software, the NMEA 2000 messages can be displayed on the computer.
[6] A time series database solution.
[7] A tool for visualisation of data.

The data retrieval process will include an adapter connected to the NMEA 2000 network and a computer. The computer will then parse the data locally by selecting different NMEA 2000 parser software like the *Actisense NMEA reader*[8] [19]. The computer would then be connected to the internet through an off-the-shelf solution like a mobile hotspot [35]. The data collected on the computer is then transferred to a service like *Amazon Web Services Time Stream*[9] [36]. The data is presented to the user through Grafana or another interface to present real-time data.

This solution has a few key advantages and disadvantages.

Advantages:

- Fast development. (if a custom solution is not chosen for data presentation)
- Not directly dependent on hardware and physical network layer knowledge.
- Gives the project the possibility to focus on different options for commercial products or further development within time and resource constraints.

Disadvantages:

- Solution is not viable as a commercial product or for further development. (strictly a proof of concept)
- No in-depth code development.

4.1.3 Alternative Solution 2

Sensar Marine offers a product called Boat Monitor, which connects to a Bilge Sentry to provide users with real-time data on water level and temperature in the bilge [37]. Users may monitor this data through the corresponding app. Sensar Marine aims to integrate their Boat Monitor product with the NMEA 2000 network. This integration is the focus of the second alternative solution, which involves adding NMEA 2000 functionality to the existing product.

- Retrieval of data from the NMEA 2000 network.
    - Off-the-shelf microcontroller with onboard CAN bus transceiver.

[8] Software for use with e.g. Actisense NGW-1 do display or save NMEA 2000 transmitted data to a computer.
[9] Cloud based time series database solution.

- Transfer data from the NMEA 2000 capable unit to data storage.
    - Make the data available to the Sensar Marine Boat Monitor through an API and available communication protocol.
    - Utilize Sensar Marine Boat Monitor, which has a built-in modem and transfer protocol for transmission to data storage.
- Receive and store data.
    - Utilize Sensar Marine infrastructure for data reception and handling.
- Data presentation.
    - Utilize existing Sensar Marine infrastructure for data presentation.

Data retrieval from the NMEA 2000 network is the focus of this solution. Data is retrieved from the NMEA 2000 network utilising an off-the-shelf microcontroller solution with an onboard CAN bus transceiver. With the help of existing libraries, custom software will process the data and make the data available through an API. The existing Sensar Marine product has a set specification; therefore, the proposed solution has requirements that must be incorporated. The specification of the Sensar Marine product entails that after the NMEA 2000 data is processed, it must be made available to the Boat Monitor through I^2C[10] in a format specified by the Sensar Marine development team [38].

The Sensar Marine Boat Monitor will poll the I^2C API for data and transfer the data to a storage location utilising a built-in IoT LTE network. The existing Sensar Marine infrastructure for handling information created by the Boat monitor is utilised to store and present the data from the NMEA 2000 network.

The solution has the following key advantages and disadvantages:

Advantages:

- Provides a unique implementation that is valuable in a real-world environment.
- Opportunity to utilise senior programmers as mentors in the project.
- Working with a company to develop a functional prototype.

[10] Inter-Integrated Circuit

Disadvantages:

- It requires in-depth knowledge of equipment and standards, which could be difficult to obtain within the given timeframe and resources.
- Gives imposed restrictions from outside parties.
- Complete project goal completion is dependent on an external partner.

4.1.4 Alternative Solution 3

Alternative solution three tries to create a solution where all components and software are customised. This solution is intended to be viable for further development and implementation as a stand-alone product in a future iteration.

- Retrieval of data from the NMEA 2000 network.
    - Off-the-shelf microcontroller with onboard CAN bus transceiver.
- Transfer data from the NMEA 2000 capable unit to data storage.
    - Off-the-shelf LTE module for microcontroller expansion.
- Receive and store data.
    - InfluxDB or other Time Series Database.
- Data presentation.
    - Custom user interface that presents available data to the user.

The data is retrieved from the NMEA 2000 network utilising an off-the-shelf microcontroller with a built-in CAN transceiver. The microcontroller will then utilise custom code and existing libraries to process data and transfer the data to central data storage. The data accumulated is then processed and presented in a custom user interface.

Advantages:

- Gives the project the possibility to focus on different options for a potential commercial product or further development.
- Complete control of the whole value chain.

Disadvantages:

- The solution would require large amounts of custom code for the vessel and infrastructure.
- Large project scope.

4.2 Discussion of Alternatives and Presentation of the Chosen Solution

The solutions described above try to incorporate the initial idea for the solution described in 2.1.3 in different ways.

Solution one offers an approach that could give an advantage when writing the report because the solution has the potential to be a quicker route to a functioning prototype than the other two alternatives. This could be advantageous as the original problem description invites the possibility of the group trying to cover too many topics in one report. In this regard, alternatives two and three not only have more potential as a basis for further research and product development but also require more of the group regarding focus and limitations within the finished report and product. If either solution two or three is chosen, the group must manage a higher risk of not completing the project goals within the given timeframe [39]. Resource management will be especially difficult if any custom development work within the domains unknown to the group proves more difficult than anticipated.

Solutions two and three use similar technology, while solution one provides a different approach. Solution one would, by all accounts, have a more comprehensive knowledge base available as it largely uses commonly used software and premade solutions where possible. Solutions two and three would allow the group to learn and use technology new to all group members, providing a greater arena for learning.

This project has the possibility of having economic and environmental implications if the project culminates in a product. The different solutions do not directly give any advantage over each other, considering the economic or environmental aspects. Due to the nature of the different projects, the second alternative holds the highest possibility of becoming an actual

product. As a result, the group considers it to have a slightly elevated potential for real-world economic or environmental impact.

After deliberation, the group concluded that although alternative solution one has the potential to be a safer alternative and solution three would offer some of the same technological challenges as alternative solution two, alternative solution two has the most potential for the group and is, therefore, the chosen alternative for this report. The chosen solution will be named N2K Insight, encompassing the microcontroller and the complete hardware.

4.3 Choice of Tools

Selecting the optimal set of development tools is highly important for the project's progression. Many tools are available and offer different functions, with their various advantages and disadvantages. Frequently, tools that provide a wide range of functionalities are accompanied by increased complexities [39]. Some tools can take too much of the available time to learn, considering the limited previous knowledge of programming microcontrollers within the development team.

To maximise efficiency, it is crucial to prioritise options that balance ease of understanding for the development team and the ability to facilitate the creation of satisfactory solutions. Additionally, it is advantageous to choose tools that provide readily available resources to address issues that might occur. Active help forums are particularly useful, meaning the more popular tools have an advantage.

In Chapter 5, the alternatives will be discussed, and a selection of development tools will be picked.

4.4 Methodology

In Chapter 4.2, a solution is discussed that has a clear goal but lacks a clear path for completion. It is common for projects to require the creation of a development path while they are underway. Without best practices to guide the creation of this path, a project may encounter difficulties along the way. [40]. To ensure the project stays smooth and efficient, it is crucial to adopt a structured approach to the project’s methodology[41]. This section details the strategic framework, which will guide the group through the project period, focusing on

effectiveness and risk mitigation. This includes development methodology, development plan and risk assessment.

4.4.1 Project Methodology

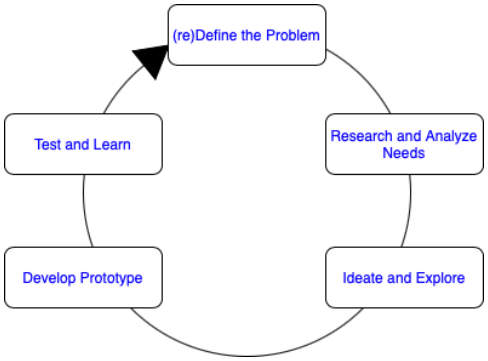


Figure 4.2: Design Thinking loop.

The methodology chosen to help the team stay on the path to completion is design thinking. Design thinking is a team-centric methodology that encourages the team to find problems and not only solutions. This approach of a constant loop of trying to understand the problem before creating a solution and evaluating the result before restarting the process seems to be a good fit for our project (fig. 4.1). Many tools in design thinking can be utilised to help further the project when necessary, such as visualising the problem and model solutions [42], [43]. Design thinking, along with other methodologies like agile development, has been utilised successfully, and this aligns well with the suggested development methodology in the next chapter [44], [45].

The group intends to use tools related to design thinking, such as brainstorming and flowcharts, which will be valuable assets in identifying and improving parts of the process. For the applied part of design thinking, quick prototyping and testing are the intended plans for integrating design thinking into all aspects of the project.

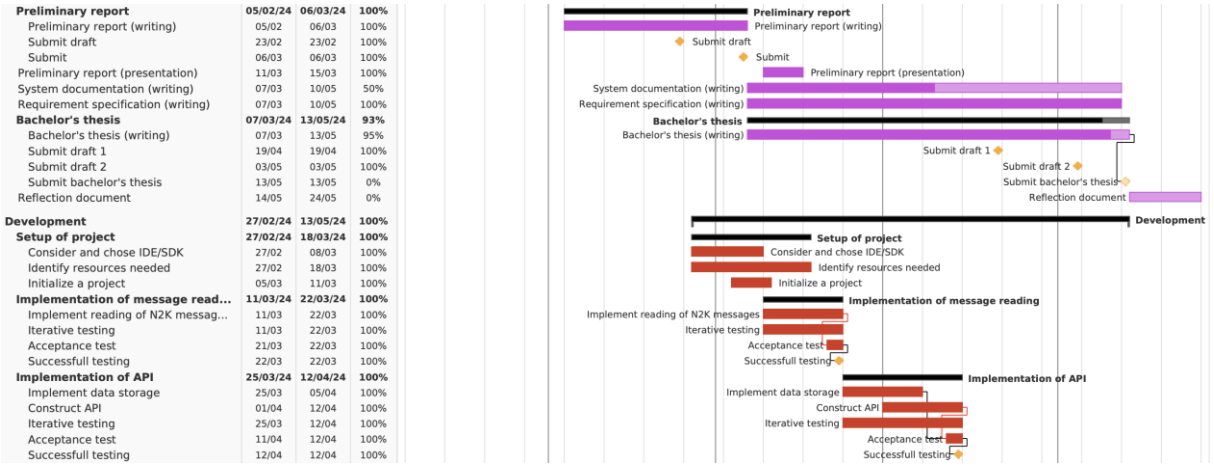
4.4.2 Development Methodology

Many different development methods and methodologies could be used in this project. The group members have tried multiple methodologies in different classes while attending Western Norway University of Applied Sciences. A requirement established early was test-driven development, as this has been effective for the group members in previous projects; this made agile development the natural choice. There are different philosophies and methodologies within agile programming, and here again, the previous experience was vital in letting the group create a mixture of XP, Scrum, and others to make a setup that works for the group. The intention is to start the project by creating unit tests and delve into making the unit tests pass by creating the corresponding methods. This is an iterative process where the group will try to get the least viable product functionality working by developing minor features and continuously pushing the code through the unit tests. Other components from agile

methodologies implemented are sprints from Scrum, backlog from Scrum, Scrum board from Scrum, pair programming from XP, test first programming from XP and small frequent releases from XP [46], [47]. This collection of agile development strategies is the basis for the development that will be done in this project.

4.4.3 Development Plan

A GANTT diagram is created to plan the process. The diagram is divided into three sections, which are then divided into sub-sections with their included tasks.



Preliminary report	**05/02/24**	**06/03/24**	**100%**
Preliminary report (writing)	05/02	06/03	100%
Submit draft	23/02	23/02	100%
Submit	06/03	06/03	100%
Preliminary report (presentation)	11/03	15/03	100%
System documentation (writing)	07/03	10/05	50%
Requirement specification (writing)	07/03	10/05	100%
Bachelor's thesis	**07/03/24**	**13/05/24**	**93%**
Bachelor's thesis (writing)	07/03	13/05	95%
Submit draft 1	19/04	19/04	100%
Submit draft 2	03/05	03/05	100%
Submit bachelor's thesis	13/05	13/05	0%
Reflection document	14/05	24/05	0%
Development	**27/02/24**	**13/05/24**	**100%**
Setup of project	**27/02/24**	**18/03/24**	**100%**
Consider and chose IDE/SDK	27/02	08/03	100%
Identify resources needed	27/02	18/03	100%
Initialize a project	05/03	11/03	100%
Implementation of message read...	**11/03/24**	**22/03/24**	**100%**
Implement reading of N2K messag...	11/03	22/03	100%
Iterative testing	11/03	22/03	100%
Acceptance test	21/03	22/03	100%
Successfull testing	22/03	22/03	100%
Implementation of API	**25/03/24**	**12/04/24**	**100%**
Implement data storage	25/03	05/04	100%
Construct API	01/04	12/04	100%
Iterative testing	25/03	12/04	100%
Acceptance test	11/04	12/04	100%
Successfull testing	12/04	12/04	100%

Figure 4.3: Excerpt from GANTT diagram.

Start-up section:

This short section aims to initialise the process of writing this thesis. A few questions need to be answered.

- What tools would be used in cooperating and communication?
- Was it possible to obtain a fixed workspace?

In addition to these questions, the group needed to understand the task that had to be answered.

Academic work:

In this section, the documentation and all their submission dates are included. Some documents have multiple submission dates because multiple iterations and drafts of these documents were to be submitted. There are milestones on all these dates that can be checked to mark the progress.

Development:

The development section divides the process into iterations.

The first subsection or iteration will not include many programming tasks. It is mostly meant to prepare for subsequent iterations. This includes determining the most suitable frameworks for developing the solution and the required resources. The available resources are researched and considered during the process leading up to the development phase, and the developers should have a good idea of what resources are needed.

The following three subsections are for tracking the programming tasks. These sections have been divided into implementation and testing tasks, with the testing task spanning the same period as the programming. An acceptance test task accompanies each programming section's last two days. When the acceptance test is successful, a milestone is passed, and the focus will be shifted to the next programming section.

Lastly, a subsection is dedicated to testing the entire system, possible optimisation and bug fixing, and evaluating the solution according to the evaluation plan. The section ends with a self-explanatory milestone labelled “Ready for publishing.”

4.4.4 Risk Assessment

A well-crafted risk assessment ensures smooth progress for the bachelor’s thesis. The intention is to highlight meaningful and realistic risks that may disrupt the project, their consequences, and ways to mitigate them.

The risks are categorised by probability (P), consequence (C), and risk (R), with P and C based on a scale of 1 to 5. The overall risk is then calculated by multiplying the values of P and C, providing a quantifiable measure of each risk's severity. The risks are colour-graded with colours ranging from green (low severity) to red (high severity) to aid visualisation of the severity of each risk.

Probability						
	Very High (5)	5	10	15	20	25
	High (4)	4	8	12	16	20
	Medium (3)	3	6	9	12	15
	Low (2)	2	4	6	8	10
	Very Low (1)	1	2	3	4	5
		Very Low (1)	Low (2)	Medium (3)	High (4)	Very High (5)
	Consequence					

Table 4.1: The risk assessment categorisations.

The risk assessment table lists potential events, their causes, assessed risks, and proposed mitigation measures. This dynamic table will be updated throughout the project to reflect any changes in risk assessment or mitigation strategies.

4.5 Evaluation Plan

The project involves developing a system to let vessel owners access real-time and historical data from their onboard systems. Therefore, the project group must deploy proper test methods. Below, there will be introduced three test methodologies the group will utilise to achieve the expected results, and finally, a subchapter involving consideration regarding the test methodologies.

4.5.1 Unit Testing

Unit testing will be implemented to ensure the reliability and functionality of a developed system. This technique allows the group to verify individual components or units of code to ensure they function as expected and minimise bugs when modifying existing functionality.

Unit testing works by writing tests to check that classes and methods behave in the intended way. The tests are intended to check simple functionality and testing integration within different classes. Unit testing is a key part of test-driven development and is a key part of XP, as mentioned in Chapter 4.4.3 [48], [49]. Mock objects are essential to simulate external

systems and devices and ensure proper project functioning as a functional NMEA 2000 network could be difficult to control [50].

In the project context, the group intends to utilise PlatformIO Unit Testing, a comprehensive framework isolating and testing individual parts of the firmware/program. PlatformIO enables the execution of tests on the local host machine, multiple embedded devices/boards, or both, which provides flexibility in testing environments. [51]

4.5.2 Integration Testing

Due to the nature of the project, it has been concluded that integration testing will be utilised to validate the interaction between different system components or modules.

Integration testing is testing the components as a group. In this project, an example could be connecting a device and checking that the messages are both read and saved as intended. If successfully executed, this would show that multiple classes and libraries work together as intended [52], [53], [54].

The group will be able to verify the integration of various functionalities and ensure the system operates as a cohesive unit.

4.5.3 System Testing

System testing will be an essential part of the testing process that evaluates the system’s end-to-end functionality in a real-world environment.

System or higher-order testing is testing that the expected result is produced from a given input. In this scenario, system testing would test the co-functionality of the required software and physical connections that make the N2K Insight work [55].

By conducting tests on actual NMEA 2000 devices, the group can assess the system’s performance, compatibility, and integration with external components, which will be Sensar Marine’s Boat Monitor in this scenario.

Given the access to Sensar Marine’s NMEA 2000 network, system testing will play a crucial role in validating the system's requirements and expectations. By simulating real-world scenarios and interactions, the group may be able to ensure that the system operates

seamlessly within the intended environment, providing potential users with a reliable and satisfactory experience.

4.5.4 Considerations

However, in addition to the project's scope of developing a system for vessel owners to access real-time and historical data from their onboard systems, it is important to acknowledge a potential challenge that might occur: the unfamiliarity with utilising these tests in the context of this project. This challenge may arise due to the complex nature of the tests required or the limited experience of the team members in this specific area. As a result, there may be a risk that the time spent learning to implement these tests may not justify the trade-off, considering the project's scope and time constraints.

If this happens to be the case, it may also affect the test-driven approach discussed in 4.4.2, more specifically, the unit test process and the test first programming from XP. Considering this, alternative testing approaches, such as simplifying the testing process, may be considered. This may involve the group prioritising the most critical tests for verifying the system's core functionality. Potential deviations from the evaluation plan, along with testing results and evaluation findings, will be discussed further in Chapter 7, along with testing results and evaluation findings.

5 TOOL SELECTION AND EVALUATION

This chapter will explain the available development tools and their weaknesses and strengths. Some tools are specific to microcontroller programming, and some are general development tools.

5.1 Microcontroller

It is necessary to have a device that can read and process the data signals sent across the NMEA 2000 network to develop this solution. Some requirements need to be considered when choosing the development device. The requirements are:

- The controller must have sufficient memory and storage to host the software.
- The controller must be able to connect to the 12-volt CAN network to read the NMEA 2000 messages.
- The controller must be able to communicate with other devices.
- The controller must be fast enough to handle all the data and communication without falling behind.

Microcontrollers usually operate at 3,3 or 5 volts, requiring a step-down module between the 12-volt CAN wires and the pins on the microcontroller [56]. This is easily achieved with some resistors, and multiple guides online explain how to do this. Therefore, this issue is not particularly difficult to overcome. However, some microcontrollers exist with a built-in module; some even come with interfaces designed to communicate on a CAN network [57].

Figure 5.1: The Arduino Logo.[58]

This project does not have a dedicated budget for expenses to cover the purchase of the most appropriate microcontrollers for development. However, Sensar Marine is an interested party, and they might be interested in buying the hardware and lending it to us during the development period.

At the beginning of the project, the group had a couple of Arduinos available. An Arduino is an inexpensive and easy-to-use microcontroller used in thousands of projects. They can be found with various microprocessors, offering different speeds and capabilities. Arduino also has a programming language, an integrated development

environment, and a framework specialised in programming microcontrollers, which will be discussed in the next chapter. Because of Arduino's popularity, a great deal of previous work and solutions can be used to guide the development. [59], [60], [61]

As mentioned, microcontrollers usually have only 3,3 and 5-volt pins; this is the case with the Arduino. Despite this downside, the original plan was to use the Arduinos the group already had and make the module that would convert the 12-volt signals from the CAN network to 5 volts. Since there was no actual NMEA 2000 network available, the group would execute testing with another Arduino that would simulate an NMEA 2000 network, using 5 volts instead of 12. Therefore, having the 12-volt to 5-volt converter was unnecessary until the group had access to an actual network up and running. This decision was reconsidered after the cooperation with Sensar Marine had started.

Sensar Marine suggested using a microcontroller board with a built-in CAN interface from SK Pang Electronics (figure 5.1) [57]. Furthermore, no additional electronics would be needed to create the solution; everything needed is included. The wires of an NMEA 2000 connector could be directly connected to this microcontroller board. ESP32 has most of the features of an Arduino; if needed, it can be programmed using the Arduino framework [62].

Figure 5.2: ESP32 microcontroller with CAN interface.

These microcontroller boards are a bit more expensive, but Sensar Marine would purchase them and let the group use them during development. With the practicality of the equipped CAN interface and larger flash and random-access memory, in addition to meeting the set requirements, the group decided to use the microcontroller board from SK Pang Electronics.

For a commercial solution, one might opt for another setup with custom controller boards, excluding some unnecessary features.

5.2 IDE and Framework

Many IDEs are available for programming microcontrollers, such as the ESP32, that will be used in this project. Two of the most popular IDEs that would work in developing the solution

are the *Arduino IDE* and the *PlatformIO IDE* [63]. These options support programming in C++, a language the developers are familiar with [64], [65]. Additionally, they come with many useful libraries and frameworks [64], [66].

Arduino IDE is particularly easy to use and not very complex compared to the alternatives. It is usually the recommended IDE for beginners and developing less advanced projects and has an extensive support community [65]. However, because of its low complexity, it has more limitations and might not be suitable for this project [67]. While getting familiar with this IDE, it was discovered that including libraries from other sources and creating custom libraries can be tricky in the Arduino IDE and can sometimes lead to conflicts. This project relies heavily on external libraries, and these kinds of limitations can significantly slow down development.

Figure 5.3: PlatformIO logo.[68]

PlatformIO IDE is an extension of Visual Studio Code. It is more advanced than Arduino IDE, but the project structure in PlatformIO is simple and like familiar structures. Adding libraries to a project in this IDE is easy; issues rarely occur when adding or creating libraries. It is a flexible tool with more features, such as advanced debugging and a Command Line Interface. The IDE has many other frameworks, including Arduino and ESP-IDF, detailed below. [64]

ESP-IDF is an official IoT Development Framework (IDF) from Espressif. This IDF can be installed as an extension in Visual Studio Code or as a plugin in Eclipse [69]. It can also be installed manually and run directly from the command line. Furthermore, selecting the ESP-IDF as the chosen framework within PlatformIO is possible. The benefit of using this framework is that it will give direct access to all the features of the ESP32 microcontroller [70]. Other frameworks, like Arduino, would lack some microcontroller features because they may not yet have been implemented. It is advised to utilise the official frameworks for the microcontroller. This is generally considered as the best practice. However, the IoT Development Framework is complex and has a steep learning curve, which makes it less viable for the less experienced and would use a lot of the available time.

Considering the pros and cons of the options and the requirements of this project, the decision falls on the PlatformIO IDE extension for VSCode, using the Arduino framework. This option offers all the functionality needed to create a working solution without being unnecessarily

complex. The IDE has many useful features and a graphic user interface, making it easier to learn and get comfortable with. Furthermore, because of its popularity, many online support articles can be found that help resolve issues that might occur.

5.3 Programming Language

Figure 5.4: C++ logo. [71]

Historically, the domain of microcontrollers has been dominated by Assembly language. However, the landscape has evolved, and today, C and C++ are usually the most popular and recommended choices for embedded programming [72]. There is also a strong argument for using Python because of its simpler syntax and the great extent of libraries [73]. In particular, a version of Python called MicroPython is optimised for programming microcontrollers. Other viable alternatives such as Java, Rust and TinyGo are considered. However, seeing as the libraries used in the solution are written in C++ and compatible with microcontroller development, the group decided to use C++.

PlatformIO, the selected IDE, offers built-in code completion and a formatting tool for C/C++, greatly simplifying the coding process [64].

5.4 Version Control

During development, the group will use Git as the version control system. This is by far the most popular choice of version control systems [74]. The team is experienced with this system and has used it daily for other projects. A lot of resources required for development can be found in GitHub repositories. Most of these repositories utilise Git as their version control system. While the platform supports other version control systems, most use Git [75]. Therefore, selecting Git would allow for easier integration of the resources required.

Another advantage of using Git, which makes it superior to other alternatives, is its ability to create and merge branches [67]. On the created branches, developers can work on different features, add new features, or solve problems. Using branches ensures that the main codebase remains undisturbed. Once the features or the bugfix are ready to be implemented, they can seamlessly merge into the main branch [76]. This way, Git allows for developing different features without interfering with the rest of the project.

5.5 Debugging

When working with a microcontroller, a few subtle changes can make a difference when debugging software.

1. The hardware running the software can be different from the one used for development.
2. The software might expect input from other devices to the microcontroller that is hard or impossible to mock, either from sensors, other microcontrollers, or anything that can change the microcontroller's input.

The fact that the hardware differs requires extra steps to debug the software efficiently. There are two approaches to this problem: interactive debugging and post-process debugging. Interactive debugging works so that you can set breakpoints in your code, and the hardware will pause execution when a breakpoint is hit. The state of the hardware can then be read, and the debugging experience will be like debugging *application software*[11]. Post-process debugging is based on simulating the running of the software and analysing the result after execution. [77]

The methodology chosen largely depends on the type of hardware being utilised, as not all hardware can be easily simulated.

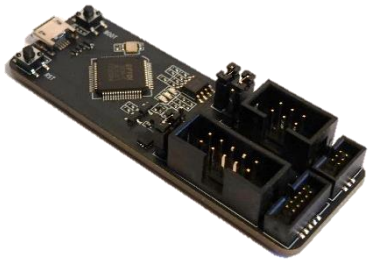


Figure 5.5: ESP-Prog.

For this project, the ESP 32 was chosen. It implements a physical standard for interactive debugging called JTAG. JTAG, alongside a debugging application, allows the user to read registers and stop execution on a specific line of code in the program. As stated above, this allows for an experience similar to debugging application software. A separate device has to be utilised to utilise JTAG with the ESP 32. The creator of the ESP 32 Espressif has created a board specifically for debugging called the ESP-Prog (Figure 5.3) that, when connected to the development environment through debugging software like OpenOCD, enables the use of JTAG to communicate with the ESP 32 and

[11] Application software is general-purpose software designed to run on general computing platforms like PCs, smartphones, or tablets.

5.6 Device Communication

In the chosen solution, two devices (microcontrollers) must be able to connect and communicate. More specifically, one device, referred to as the controller, must be able to ask another device, the peripheral device, for some data, which the peripheral device should then return.

Microcontroller devices have many options for communicating with other devices; some even have built-in wireless connectivity. For example, the ESP32 microcontroller by Espressif offers Wi-Fi and Bluetooth built into the chip [78]. These technologies can transfer data at multiple megabits every second, which is very fast for this application [79]. When it comes to non-wireless communication peripherals, most microcontrollers are equipped with the communication protocols UART (Universal Asynchronous Receiver and Transmitter), SPI (Standard Peripheral Interface) and I^2C (Inter-Integrated Circuit) [80]. UART is very simple but can only connect two devices. At the same time, SPI and I^2C can be more complex but can connect many *peripheral* devices to one *controller* device (or multiple controllers in the case of I^2C) [81].

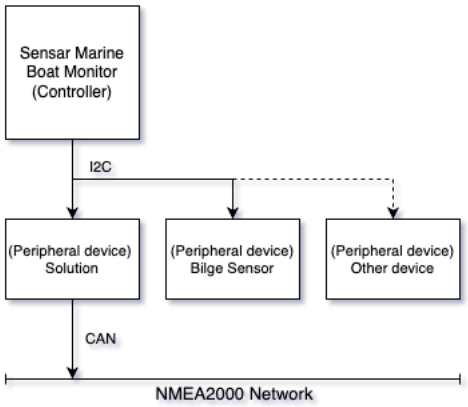


Figure 5.6: The I^2C bus.

Since the chosen solution describes connecting one device (the peripheral) to the Sensar Marine Boat Monitor, Sensar Marine's needs must be considered. Their Boat Monitor does not have wireless connectivity, and they would likely want to connect multiple devices simultaneously. The engineers at Sensar Marine suggested using the I^2C protocol because this allows them to connect other peripheral devices, like a bilge sensor, to monitor a vessel's water level.

I^2C comes in different modes that can transmit data at different speeds. These modes are detailed in Table 5.1. Even though I^2C has modes that allow it to transfer data at up to 5 Mbit/s, the ultra-fast mode is not common on most microcontrollers. Either way, ultra fast-mode only allows for one-way transmission and would not work in this solution [82]. However, the speed does not need to be that high considering the limited amount of data sent; fast mode and even standard mode should be sufficient for the solution.

Mode	*Transmission speed*
Standard mode	100 kbit/s
Fast-mode	400 kbit/s
Fast-mode plus	1 Mbit/s
High speed	3.4 Mbit/s
Ultra fast-mode	5 Mbit/s

Table 5.1: The speeds offered by I²C. [82]

6 DETAILED SOLUTION

This chapter will explain the prototyping, milestones, and specifics of the solution developed during this project. It aims to provide an understanding of the choices made while conducting our research and substantial knowledge of the final solution and setup.

6.1 Project Progression

This sub-chapter aims to provide a chronological insight into the choices and direction changes during the project and why.

6.1.1 Proof of Concept Setup

At the start of the project, it was deemed crucial to confirm the feasibility of extracting NMEA 2000 data from an NMEA 2000 network. The reasoning is that the information regarding how to do this is scarce. If the extraction were deemed too difficult, a change of direction would be necessary.

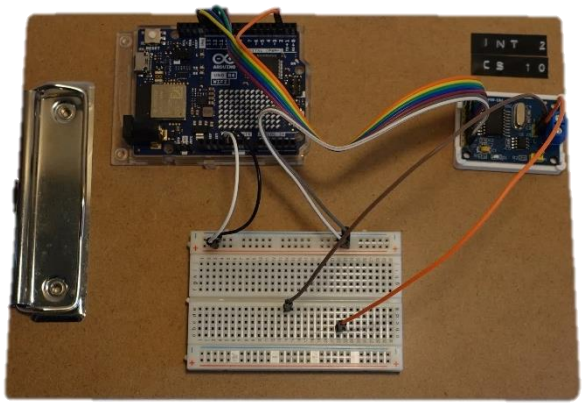


Figure 6.1: Initial set-up of proof of concept.

The setup for this proof of concept involved two Arduinos running a customised version of the example software found in the NMEA 2000 library for Arduino made by Timo Lappalainen. The library will be further discussed in Chapter 6.3.1. The NMEA 2000 network is a CAN bus network, and thus, it requires both a CAN bus transceiver and controller. The Arduino selected has onboard CAN bus controller capabilities but lacks the transceiver hardware; therefore, an external CAN bus transceiver was needed (For details of CAN bus and hardware requirements, see Chapter 4.2**Feil! Fant ikke referansekilden.**). A separate board called HW-154 was connected to both Arduinos to add the missing functionality. One of the Arduinos was running the *ActisenseListenerSender* example while connected to a computer running NMEA simulator software made by the Finnish company Kave Oy [17], [83]. The other Arduino was running the DataDisplay2 example software that converts the NMEA 2000 message format to human-readable text and presents it through a serial interface to a connected computer [17].

The setup proved that the library and examples with available hardware and software could be utilised. The group could send data from one Arduino to another and read and translate the data, but a test on a real NMEA 2000 network was also required.

6.1.2 Initial setup of test environments

For the group to be able to facilitate testing, a functioning NMEA 2000 network was necessary. The creation of such a network was possible due to Sensar Marine having access to the physical components needed. Any components not available were promptly ordered. The components used in this NMEA 2000 network include a GPS unit with NMEA connectivity, a Victron device with multiple sensors connected and reports different types of NMEA 2000 messages, and NMEA T-connectors.

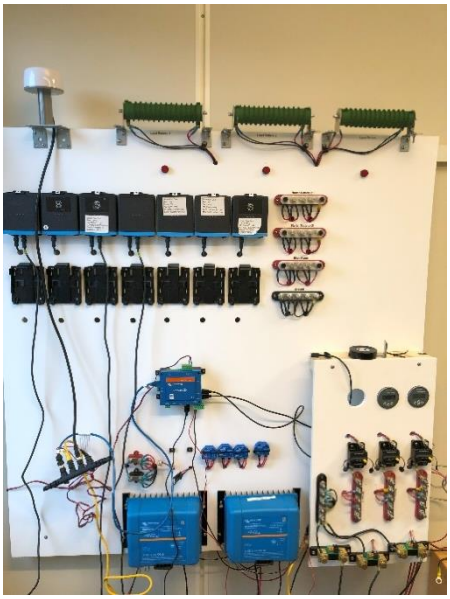


Figure 6.2: Set-up of a functioning NMEA network at Sensar Marine's office.

Additionally, the ESP32 microcontroller the group utilised for this project had integrated Bluetooth serial capabilities, which enabled the group to view the serial monitor on a Bluetooth-connected device. Connecting a computer via USB to view the serial monitor could disturb the electrical signal, so viewing the serial monitor wirelessly was very useful.

6.1.3 Finding Resources

After succeeding in setting up the physical NMEA 2000 network and ensuring it worked as intended, it was necessary to delve into the extensive libraries made by Timo Lappalainen, mentioned in 2.1.1. This library was intended for Arduino-based boards, and since this project is built on an ESP32 chip, the library had to be adapted accordingly with compatible CAN “drivers” and necessary classes. Fortunately, Timo had already developed an inherited NMEA 2000 library specifically tailored for ESP32 boards, saving the group valuable time.

An IDE and framework had to be chosen to develop and use these libraries. Initially, the ESP-IDF was considered the framework. The group installed the framework as an extension in VSCode but soon encountered the steep learning curve associated with it, introduced in Chapter 5.2. After too many hours were spent trying to make a simple program to print

Bluetooth serial data, the group discarded this framework and went for an alternative option: PlatformIO as a VSCode extension with the Arduino framework. Within a short time, the group ran the simple test program smoothly, which marked a swift transition to the project's development phase.

6.1.4 First test of NMEA 2000 message retrieval

After the initial programming phase, an integration test was needed to confirm that NMEA 2000 messages were read off a functional NMEA 2000 network and that the messages could be presented through other means.

The NMEA 2000 network was the same as described in Chapter 6.1.2. The microcontroller used in this setup was the SK Pang ESP32 CAN bus board, as described in Chapter 6.2.1.

The board was set up to send the received NMEA 2000 network message in a human-readable format to a Bluetooth serial interface.

This setup proved to be successful. Both information regarding COG/SOG and data regarding battery health were sent through the NMEA 2000 network, proving that the code was functional and that the next step could be coded.

6.1.5 Inter-Integrated Circuit Communication

One requirement for communication between the N2K Insight and Sensar Marine Boat Monitor was that the communication protocol be I2C. Chapter 5.6 provides a detailed explanation of I2C.

An integration test was set up to verify the group's understanding of the library and how to use it was satisfactory. This test involved two connected SK Pang ESP32 CAN bus boards, with one board running controller[12] software and the other running target[13] software.

This setup proved that the group understood the I2C library and the use necessary to implement I^2C functionality in the main codebase.

[12] Master as a term for controlling unit is deprecated, the replacing term as stated in the I^2C standard is controller [84].
[13] Slave as a term for unit being controlled by another unit is deprecated, the replacing term as stated in the I^2C standard is target [84].

6.1.6 API

The final unimplemented module of the code is the API for communication between the N2K Insight and Sensar Marine Boat Monitor. This was a team effort between the Sensar Marine development team and the group, as this must be implemented on both devices. The hardware consists of the SK Pang ESP32 CAN bus and the Boat Monitor, where the SK Pang ESP32 CAN bus is connected to the NMEA 2000 network and the Boat Monitor.

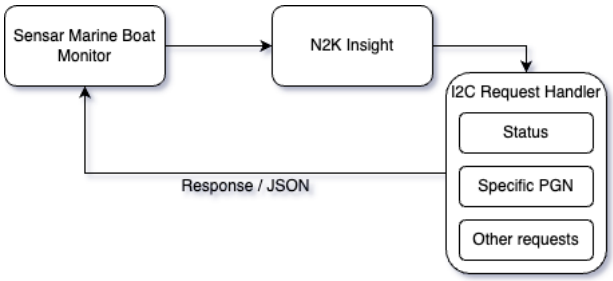


Figure 6.3: Simplified diagram showing how the boat monitor selects which method to call.

6.2 Hardware

This report section will provide an overview of all the hardware components used to construct the solution. The microcontroller belonging to the Boat Monitor, the NMEA 2000 network, and the N2K Insight microcontroller that ties the final solution together will all be elaborated upon.

6.2.1 Microcontroller (N2K insight)

The microcontroller used for the N2K Insight final solution is based on the ESP32 CAN bus board manufactured by SK Pang. This board features an ESP32-WROOM-32D chip, which Espressif, a global high-tech leader in microcontroller solutions, develops.

The SKU for this board is ESP32-CAN, which indicates compatibility with ESP32 microcontrollers and its integration with a CAN transceiver. The transceiver enables communication via CAN bus. In addition to the ESP32's WiFi functionality, it has Bluetooth Classic and a BLE Module. An RGB LED is also present, which is helpful for visual indication. The microchip has 520 KB of RAM and a 4 MB flash memory.

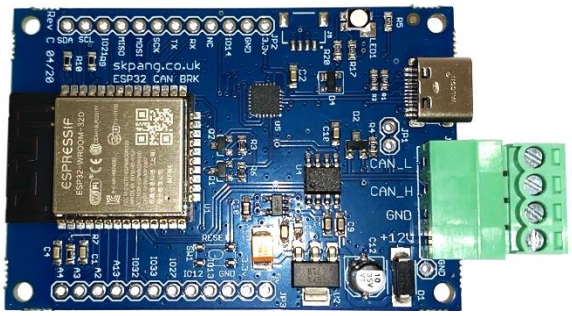


Figure 6.4: Board used in the N2K Insight solution, manufactured by SK Pang.

Programming on this board is done through a USB-to-Serial converter with a USB-C connector, automatic bootloader and reset functionalities. This simplifies loading new firmware or programs onto the device, as the automatic bootloader is triggered once you connect the board to the computer without needing manual bootloading.

6.2.2 Microcontroller (Sensar Marine Boat Monitor)

To get an understanding of the broader hardware solution, an overview of the hardware of the Boat Monitor is necessary.

The Boat Monitor has multiple custom PCBs[14], each equipped with a TI CC1312R7 MCU developed by Texas Instruments. This MCU is a versatile wireless microcontroller designed for low-power communication and advanced sensing in several applications. [85]

Additionally, to utilise the abilities of the PCBs, an integration of an LTE[15] modem is also present in the Boat Monitor. This modem essentially serves as a means for cellular connectivity, which allows the Boat Monitor to transmit and receive data over LTE networks.

6.3 Software

The complete solution's software ecosystem contains several components to meet specific functionalities and needs. The subchapters below will delve into these components.

6.3.1 Library

The libraries made by Timo Lappalainen are made for developing NMEA 2000 bus devices, and since its inception, they have been used in several commercially certified NMEA 2000 devices. They have all the functionalities one may need to communicate with the NMEA 2000 network. However, considering the library author has reversed-engineered the standard, there

[14] Printed Circuit Board (PCB) is a medium used to connect components to another in a circuit.
[15] Long-term Evolution (LTE) is a standard for wireless broadband communication for mobile devices and data terminals.

are some exceptions where a few functionalities may be missing or not working as intended. This was, however, a non-issue in the context of this project.

Understanding how these libraries worked would be essential for the project's success without access to the original NMEA 2000 standard.

6.3.2 N2K Insight Custom firmware

The solution developed is based on the DataDisplay2 example in Timo Lappalainen's NMEA 2000 library. This example uses his library to parse the NMEA 2000 messages sent to an Arduino and write them to the serial monitor[16]. This solution needed to be adjusted to parse the messages and store them for transmittance to the Sensar Marine boat monitor.

In the DataDisplay2 example, the code parses the messages using the associated functions and prints them to the serial monitor. In the context of this project, there is no point printing out any values but rather storing them for transmittance to the third-party device. Therefore, instead of printing out the values, they will be stored as objects and give them a function for turning the information into a JSON string, as that is the format requested by Sensar Marine. A simplified class diagram for defining and creating objects with the PGN 129026 can be seen in Figure 6.4. The JSON formatted string is the message to be sent to the Sensar Marines device.

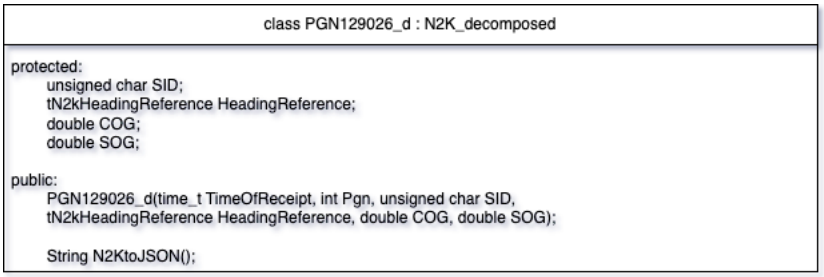


Figure 6.5: Example from the decomposed class.

I²C was selected as the solution for communicating with the Sensar Marines boat monitor; this meant that the group needed to include the "Wire" Arduino library, which is a library that has all the necessary functions for achieving I²C

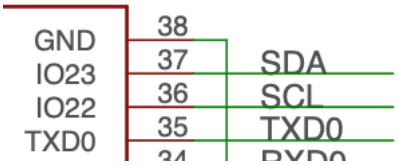


Figure 6.6: Excerpt from schematic showing the pins for I²C communication.

[16] The serial monitor allows for two-way communication with a serial port device, useful for monitoring and configuring the device [86].

communication. The library is used to set the pins for I²C communication, which in this case are pins 23 and 22, seen as SDA and SCL in the schematic. SDA (Serial Data) is the data signal, and SCL (Serial Clock) is the clock signal. These pins could be set to any two I/O pins on the board, but these are the standard pins for I²C communication on an ESP32. They must still be set because the Arduino framework defaults them to pin 21 for SDA and pin 22 for SCL.

```
Wire.setPins(23,22);
Wire.onReceive(receiveData);
Wire.onRequest(requestData);
Wire.begin(I2C_ADDRESS);
```

Figure 6.7: Part of code where I²C values are set.

After the pins are set, the methods to be called when the ESP32 receives a signal need to be set. There are two types of calls the I²C controller can send to the peripheral device. The controller can send some data to the peripheral, or it can send a request for data. These methods, "receiveData" and "requestData", are defined in an API that handles the requests and responses. The controller sends a list of four bytes, where the first byte defines which type of request to do, and the next three bytes are the parameters for the function associated with the request if a parameter is needed. If the associated function does not take a parameter, one may only look at the first byte and send the accurate response.

When the peripheral receives a message from the controller, it forms a byte array response. The length of the rest of the array is defined in the first two bytes. This is so that the controller can read for two bytes first and then know how many more bytes to read. If the response is to give some information about a device on the NMEA 2000 network, the message after the first two bytes will be in the JSON format.

At the time of delivery of this project, only one type of message is implemented for transmittance by I²C, specifically the PGN 129026. Depending on which version of the code you look at, the rest of the messages captured from the NMEA 2000 network are either printed to the serial monitor or ignored.

After a message has been sent from the peripheral, it is further handled by the Sensar Marines controller device. This process will be summarised in the next chapter.

6.3.3 Sensar Marine Boat Monitor Custom firmware

This chapter will briefly explain Sensar Marine's system and how they handle the data.

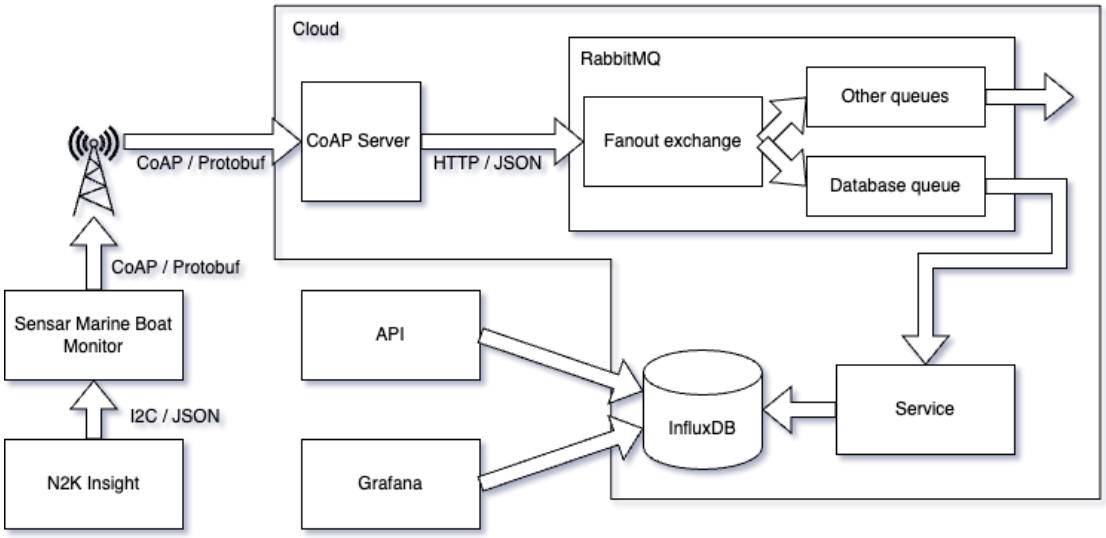


Figure 6.8: Diagram of the movement of data through Sensar Marine's system.

Controller unit:

Sensar Marine uses an operating system called Contiki-NG. They have also used a software development kit called SimpleLink by Texas Instruments, which gives them access to libraries like the one needed for I^2C communication.

Because of the low data rate and storage, it is beneficial to compress data as much as possible. JSON objects are stored in memory as at least one byte per character; this is not very efficient, and there is potential to make the data smaller and more efficient to transmit. They store data using Protocol Buffers (Protobuf), which is a "language-neutral, platform-neutral, extensible mechanism for serialising structured data" developed by Google [87]. This is a very compact and fast data format, which serialises data in binary format instead of the text-based format JSON uses [88].

The next step is to send the data to the cloud. Data is sent using the Constrained Application Protocol (CoAP), a transport protocol for constrained IoT devices based on UDP. It is similar to HTTP, where a client sends a request, and the server sends a response. It is also based on the REST model [54]. The CoAP packet is then sent to an LTE modem, which transmits the data by cellular to the cloud.

Recorded data is stored locally on the device continuously, but it is sent to the cloud at different frequencies depending on the vessel's status. While using the vessel that the Boat Monitor is connected to, data is transmitted every four minutes, every hour when not in use, and every fifteen minutes when not in use but connected to a charger.

Cloud:

When packets arrive at the CoAP server in the cloud, they are parsed, and the Protobuf-serialized data is converted back into JSON format for increased readability. The server then sends the JSON-serialized data as HTTP packets to a RabbitMQ message broker that uses a fanout exchange to copy the messages and put them into multiple queues. A service then gets items from one of the queues and puts them into a database, which a web client or an API can poll when they want to get data. One of the queues streams live data directly to the Sensar Marine app.

6.3.4 Time Series Database

Time series data, in simple terms, is essentially measurements or events that are tracked and analysed over time. This may include a wide range of things like server metrics, network data, stock prices, and many other types of analytical data. The main feature of time series data is that it’s always looked at in relation to time, which is interesting from a standpoint where a user would look at how the data changes over time. A quick check to identify time series data is to see if any graph axes are time. [89]

There are two forms of time series data: regular and irregular. Regular time series data is collected at fixed intervals (this could be every 5 seconds, for example). In contrast, irregular data doesn’t follow any set pattern and may be driven by user input or other external events.

InfluxDB is a database designed to handle time series data and is utilised by Sensar Marine to collect and handle their regular time series data effectively.

6.3.5 Visual Representation of Data

An appropriate tool is necessary for the end user to visualise the data. While there are many options for retrieving and presenting the data from the database, Sensar Marine has specifically chosen Grafana for their solution.

Figure 6.9: Status of N2K Insight in Grafana.

Grafana is an interactive data visualisation tool which works seamlessly with InfluxDB (and other databases). This tool allows the user to see data via charts and graphs unified into one dashboard to gain insights from their data.

7 RESULTS

This chapter will present the project's factual results, how the group evaluated it, and the evaluation's findings.

7.1 Evaluation Methodology

The intended evaluation for this project is described in Chapter 4.5, evaluation plan. While the project was underway, the group discovered some issues with the initial plan to evaluate the project. One of the main evaluation methods was intended to be unit testing as the possibility of easy testing of new implementations and changes in code would be valuable in any situation, especially when the project timeframe is one of the more limiting factors, as described in Chapter 4.4.4. After an initial phase of research and decision-making, the complexity of implementing the mock objects needed for unit testing in a microcontroller environment while using the libraries needed for our project was deemed too comprehensive for this task due to the lack of experience in the group and time constraints.

Therefore, the intended evaluation plan was changed underway to reflect a pivot to continuous integration testing and system integration testing.

Continuous integration testing has been conducted by connecting the N2K Insight module to the NMEA 2000 network and checking for expected behaviour by either printing to Bluetooth or COM port serial monitor. When implementing new features, an effort was made to try the features already implemented and verify that the continued functionality has been completed.

System integration has been the project's goal from the start as a simple but effective way of confirming or rejecting the finished project as successful in implementing and answering the project questions and initial project idea.

7.2 Evaluation Results

Continuous integration testing has been conducted for every project progression sub-goal, as outlined in Chapter 6.1.

7.2.1 NMEA 2000 Network Integration

To verify that the N2K Insight module was indeed capable of reading messages from the NMEA 2000 network, an implementation test where messages sent on a functioning NMEA 2000 network were relayed in a human-readable form to Bluetooth Serial and thus showing that the NMEA 2000 messages were successfully read from the network by the N2K Insight module.

7.2.2 I^2C Integration

To verify the functionality of the I^2C setup, a separate unit with code to simulate the controller unit was set up to send requests to the target unit. The N2K Insight module was connected to the NMEA 2000 test network and the test controller unit. The N2K Insight module was then polled for the defined messages, and the controller printed the result to Bluetooth Serial, enabling verification of the expected result. The integration testing showed that the N2K Insight module presented the expected JSON string with information from the NMEA 2000 message to the controller unit through I^2C.

7.2.3 System Integration Test

The final and most definite test to check if our code and integrations are working as intended is to test the system. This test was conducted by connecting the N2K Insight module with the NMEA 2000 network through a T-connector and the Sensar Marine Boat Monitor through I^2C. The test itself showed that the group were able to extract information from the NMEA 2000 network and present it to the Boat Monitor through the API. The information was then successfully transmitted to Sensar Marine cloud service and presented through Grafana.

This showed that all libraries, custom code, and hardware worked in unison as intended and that the system integration was successful.

7.3 Project Results

The evaluation results demonstrate that both the individual project sub-parts and the project as a whole are functioning as intended. The N2K Insight module can read the messages from an NMEA 2000 network, store the information, and make the information available to other devices through an I^2C interface and a stated API. The information is then sent to a central storage and made available to the user. The final results of the project, therefore, show successful integration of the complete system.

8 DISCUSSION

This chapter aims to delve into the project's journey from inception to finish. The intention is to highlight key phases of the work done, the challenges encountered, how this affected the progress, the solutions implemented, and the overall learning outcome for the group.

8.1 Pre-Project Planning

Before the project's starting date, the group defined a project based on an idea of one of the members.

During the early discussions before the project phase, the group identified something that very likely could become an issue: limited rooms on campus. Ideally, a dedicated room or some desks where the group could meet and store equipment securely for the semester was wanted. Proactive steps were taken early to seek alternative solutions, and emails were sent to various local businesses and office spaces explaining the situation. One of these companies was Sensar Marine, who expressed great interest in the project and ultimately was able to provide an office space for the group. During the course of this project, it has been immensely helpful to have a dedicated space where the group can conduct discussions and test out our hardware at a moment's notice. This has played a crucial role in the progression of our project, and it has enabled us to achieve our goals successfully. Without this dedicated space, the group would not have been able to conduct our research and testing efficiently, and our progress would have been hindered.

8.2 Project Execution

The group consisted of three students, who all had the same schedule for the most part. One of the students in the group worked part-time; however, due to the nature of the project, this did not affect the progress overall. The student could catch up on missed work with no issue. It was decided early on to track each member's hours to get an overview of the work done and to avoid an uneven spread of hours during the project. There were times during the project run when assignments from a different course had to be prioritised; however, since the group were efficient at planning ahead while keeping these factors in mind, the project never fell too far behind schedule.

During the project run, no members had set roles. One member had a small amount of prior experience with microcontrollers, but the focus was on making sure all members got caught up to ensure the workload was even. Tasks were continuously created in Notion, and the group members chose the tasks they wanted. This way of working ensured that all members' learning outcomes were equal, as everyone was involved in all project areas.

In the first half of the project run, the course managers gave some mandatory supplementary assignments. These included tasks such as analysing bachelor projects from previous years and presenting the findings to an audience. While this had valuable learning benefits, the group felt less time could have been spent on it, as it slightly delayed getting fully immersed in the actual project.

A potential challenge that might have hindered the project's progress was the group's lack of experience with soldering ESP32 chips and other electrical components. Fortunately, Sensar Marine solved this, providing valuable support.

Progress in the development phase was reset several times due to the difficulty of selecting the appropriate development framework, as elaborated in Chapter 5.2. The lack of experience within the group regarding this level of microcontroller programming hindered the group from being able to implement as many features as wanted.

Given the steep learning curve of the project, readjusting constraints for the project scope was essential during the duration of the project. The amount of time needed to develop and implement PGNs (Parameter Group Number) proved to be greater than first expected due to the lack of prior relevant experience within the group and time spent understanding Timo’s library. Considering the project's time limit and the approaching deadline, one of the measures the group settled for was to limit the amount of PGNs to implement in the final solution.

8.3 Reflections

From the beginning, clear expectations for the project were set, and the group largely met them by the deadline. However, in retrospect, certain aspects of the project could have been done differently. One major change the group agreed upon was starting the development earlier, which was more time-consuming than expected. A great amount of time went into research before any code was written. By starting with the development in conjunction with

the exploratory research the group conducted beforehand, the time could have been managed slightly better and resulted in more PGNs implemented.

This could also have been improved by setting more appropriate constraints on the project scope in the early stages of the planning phase, as the project scope was too big relative to the time limit. However, it would have been hard to make such a prediction due to the group's inexperience within this field before the project run.

Another area where the group could have improved during the project is actively choosing time-efficient options. One such example, as mentioned in 6.1.3, is initially choosing ESP-IDF as the framework. Due to prior research of all the alternatives, the group knew this framework had a steep learning curve. After several hours were spent trying to learn it, it was dropped in favour of PlatformIO. Ultimately, these hours could have been spent elsewhere had the group initially selected the less complex framework.

During the semester, the group was concerned with building synergy. This was done through walks together post lunch and other teambuilding activities outside of office hours, which, looking back in retrospect, helped boost team morale throughout the project. Overall, the group is delighted with the project run and what has been achieved.

9 CONCLUSION AND FURTHER WORK

9.1 Conclusion

In Chapter 1.4, a primary goal was defined, which was further divided into subgoals. The general goal was to create a solution that would allow owners of marine vessels to view information about their vessels remotely. This goal is divided into three subgoals, which can be considered the three main problems to solve to create a functioning solution. A research question was formulated based on these goals.

The research question:

- *How can data from the NMEA 2000 network be efficiently read and extracted beyond local access to enable remote monitoring of a marine vessel´s systems?*

After exploring alternative solutions, the process based on the implemented solution looks like the following:

- Data Retrieval: The group utilise an off-the-shelf microcontroller with an onboard CAN bus transceiver to extract data from an NMEA 2000 network. Initially, the group was only able to implement the retrieval of one PGN.
- Data Transfer: The data is then made accessible via an API and transferred to data storage using the built-in modem of the Boat Monitor.
- Data Handling: Sensar Marines infrastructure is then utilised for data reception, storage, and presentation.

Based on the research question and the implemented solution, the project has demonstrated how remote monitoring through successful extraction and transmission of data from an NMEA 2000 network can be done, albeit it is currently limited to just one specific PGN. The current result of the project provides a solid foundation for further exploration and improvements of the system.

9.2 Further work

There is great potential in the product developed in this project, and some features and improvements that could be added to make it even better have been identified.

9.2.1 Finish Implementation

The solution's first step in further development should be implementing more than the single Parameter Group Number (type of message) implemented when writing this. There are, however, very many PGNs that could be implemented. The library the group have used by Timo Lappalainen currently supports around 60 PGNs and makes for a good starting point [17]. If additional PGNs are to be implemented, the library would need to be extended to be able to parse these. This would imply knowing what information a message with that PGN includes, which can be acquired by purchasing the NMEA 2000 standard.

9.2.2 Expansion

As has been mentioned multiple times in this document, NMEA 2000 works on a CAN bus network. Many other technologies use the CAN bus to communicate; some of these are used in marine technology. Some boat motors, for example, communicate using the J1939 standard. Some older boats and ships might use one of the predecessors of NMEA 2000. There is potential to capture all this communication by translating them into the NMEA 2000 standard or by having multiple interfaces that can connect to those networks and the CAN bus of the NMEA 2000 network. This would increase portability, allowing users to monitor more of their equipment.

9.2.3 History

Currently, there is no history of measures from monitored devices implemented in the solution. Reviewing the history of NMEA 2000 connected devices enables troubleshooting, performance monitoring, maintenance planning and data analysis. This could be achieved by creating data storage locally on the microcontrollers and having functions for searching through it. However, given the limited capacity of microcontrollers, one might want to store only a selection of message types, for example, focusing on the engine parameters history.

9.2.4 Direct Connection

While using the ESP32 microcontroller during this solution's development, the group became increasingly familiar with its capabilities and features. This sparked some ideas for functionality that could be included in the product, mainly in transferring data by Bluetooth. One idea the group got is that if a user were near the device, they could connect their phone to the device and continuously save data on their phone. This would allow the user to monitor

very recent data instead of waiting until the Boat Monitor sends data. A solution like this could also be expanded to transferring the data to the cloud directly from your phone, using the mobile network, instead of keeping the information always stored on your phone and taking up space. Information could also temporarily be stored locally on a mobile phone until an internet connection is established.

Instead of using Bluetooth in the solution suggested above, one might be able to do something similar by using the Wi-Fi capabilities that the ESP32 offers. However, this might cause some issues because your phone can usually not be connected to the Wi-Fi on the device and the mobile network simultaneously.

9.2.5 Handling Data

As mentioned in the motivation for this project, many useful things could be done with the data collected from an NMEA 2000 network. In the solution the group have come up with, not much is done to handle or analyse the data. Currently, the data is recorded and displayed on a screen in Grafana or the Sensar Marine app. One suggestion is to use machine learning here to analyse data and determine when the engine or other equipment might need maintenance. This could help optimise fuel efficiency and engine durability in the long run.

9.2.6 Improvement

Without having finished the implementation of all the message types, it is hard to conclude if our device is efficient enough regarding sending the information to the Sensar Marine device. Other communication protocols and variable formats might need to be explored to optimise transfer speed and minimise data size. The group have seen that some messages include information that might not be interesting for most uses and might be alright to exclude when transmitting. This would depend on the users, and some further research into the use cases should be done before choosing what information to keep and what not to keep.

10 REFERENCES

[1] 'What Is A Microcontrollers: Its Types, Applications, And How Does It Work? - Tesca Technologies Pvt. Ltd.' Accessed: Mar. 05, 2024. [Online]. Available: https://www.tescaglobal.com/blog/what-is-a-microcontrollers-and-how-does-it-work/

[2] R. Chéour, S. Khriji, M. abid, and O. Kanoun, 'Microcontrollers for IoT: Optimizations, Computing Paradigms, and Future Directions', in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, Jun. 2020, pp. 1–7. doi: 10.1109/WF-IoT48130.2020.9221219.

[3] C. Bassow, *TI TMS1000NP*. [Online]. Available: https://en.wikipedia.org/wiki/Texas_Instruments_TMS1000#/media/File:TI_TMS1000NP_1.jpg

[4] K. R. Raghunathan, 'History of Microcontrollers: First 50 Years', *IEEE MICRO*, vol. 41, no. 6, pp. 97–104, 2021, doi: 10.1109/MM.2021.3114754.

[5] J. Bartlett, *Electronics for Beginners: A Practical Introduction to Schematics, Circuits, and Microcontrollers*, 1st Edition. Berkeley, CA: Apress L. P, 2020. doi: 10.1007/978-1-4842-5979-5.

[6] 'can-newsletter.org - Applications'. Accessed: Feb. 20, 2024. [Online]. Available: https://can-newsletter.org/engineering/applications/160322_25th-anniversary-mercedes-w140-first-car-with-can/

[7] D. Hristu-Varsakelis and W. S. Levine, Eds., *Handbook of Networked and Embedded Control Systems*. Boston, MA: Birkhäuser, 2005. doi: 10.1007/b137198.

[8] 'CAN in Automation (CiA): History of the CAN technology'. Accessed: Feb. 20, 2024. [Online]. Available: https://www.can-cia.org/can-knowledge/can/can-history/

[9] 'https://www.sae.org/site/about/history'. Accessed: Feb. 22, 2024. [Online]. Available: https://www.sae.org/site/about/history

[10] 'A Quick Guide to NMEA 2000 | KUS Americas, Inc.' Accessed: Jan. 22, 2024. [Online]. Available: https://kus-usa.com/resources/a-quick-guide-to-nmea-2000/

[11] 'NMEA 2000 installations for yachts | Yachting Pages'. Accessed: May 12, 2024. [Online]. Available: https://www.yachting-pages.com/articles/a-guide-to-nmea-2000-installations-for-yachts.html

[12] 'NGW-1', Actisense. Accessed: Feb. 22, 2024. [Online]. Available: https://actisense.com/products/nmea-2000-gateway-ngw-1/

[13] A. Piętak and M. Mikulski, 'On the adaptation of CAN BUS network for use in the ship electronic systems', *Polish maritime research*, vol. 16, no. 4, pp. 62–69, 2009, doi: 10.2478/v10012-008-0058-9.

[14] T. Borkowski, L. Kasyk, and P. Kowalak, ‘Assessment of ship’s engine effective power, fuel consumption and emission using the vessel speed’, *Journal of KONES*, vol. Vol. 18, No. 2, pp. 31–39, 2011.

[15] M. L. Hakim, B. Nugroho, M. N. Nurrohman, I. K. Suastika, and I. K. A. P. Utama, ‘Investigation of fuel consumption on an operating ship due to biofouling growth and quality of anti-fouling coating’, *IOP Conf. Ser.: Earth Environ. Sci.*, vol. 339, no. 1, p. 012037, Oct. 2019, doi: 10.1088/1755-1315/339/1/012037.

[16] M. Barisic, J. Nowak, and Y. Parrondo Martinez, ‘Remote Diagnostic Services As Enablers For Intelligent Shipping’, presented at the SNAME Maritime Convention, OnePetro, Oct. 2017. Accessed: May 10, 2024. [Online]. Available: https://dx.doi.org/

[17] ‘NMEA2000 C++ library’. Accessed: Jan. 30, 2024. [Online]. Available: https://github.com/ttlappalainen/NMEA2000

[18] ‘W2K-1’, Actisense. Accessed: May 11, 2024. [Online]. Available: https://actisense.com/products/w2k-1-nmea-2000-wifi-gateway/

[19] ‘NMEA Reader’, Actisense. Accessed: May 11, 2024. [Online]. Available: https://actisense.com/acti_software/nmea-reader/

[20] ‘OpenSkipper | Display and process NMEA 0183, NMEA 2000 & AIS data from nautical instuments’. Accessed: May 11, 2024. [Online]. Available: http://openskipper.org/

[21] S. Krile, D. Kezić, and F. Dimc, ‘NMEA Communication Standard for Shipboard Data Architecture’, *NAŠE MORE : znanstveni časopis za more i pomorstvo*, vol. 60, no. 3–4, pp. 68–81, Oct. 2013.

[22] M. Di Natale, H. Zeng, P. Giusto, and A. Ghosal, *Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice*. New York, NY: Springer, 2012. doi: 10.1007/978-1-4614-0314-2.

[23] H. K. Kondaveeti, N. K. Kumaravelu, S. D. Vanambathina, S. E. Mathe, and S. Vappangi, ‘A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations’, *Computer Science Review*, vol. 40, p. 100364, May 2021, doi: 10.1016/j.cosrev.2021.100364.

[24] A. Maier, A. Sharp, and Y. Vagapov, ‘Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things’, in *2017 Internet Technologies and Applications (ITA)*, Sep. 2017, pp. 143–148. doi: 10.1109/ITECHA.2017.8101926.

[25] Y. A. Badamasi, ‘The working principle of an Arduino’, in *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*, Sep. 2014, pp. 1–4. doi: 10.1109/ICECCO.2014.6997578.

[26] M. A. Budiman, J. T. Tarigan, and A. S. Winata, ‘Arduino UNO and Android Based Digital Lock Using Combination of Vigenere Cipher and XOR Cipher’, *J. Phys.: Conf. Ser.*, vol. 1566, no. 1, p. 012074, Jun. 2020, doi: 10.1088/1742-6596/1566/1/012074.

[27] ‘History’, National Marine Electronics Association. Accessed: Apr. 26, 2024. [Online]. Available: https://www.nmea.org/history.html

[28] ‘National Marine Electronics Association’. Accessed: Apr. 26, 2024. [Online]. Available: https://web.nmea.org/atlas/web-content/196

[29] *NMEA 2000*, IEC 61162-3. Jul. 2014. [Online]. Available: https://webstore.iec.ch/preview/info_iec61162-3%7Bed1.2%7Db.pdf

[30] ‘NMEA 2000 Explained - A Simple Intro [2023]’, CSS Electronics. Accessed: Feb. 16, 2024. [Online]. Available: https://www.csselectronics.com/pages/nmea-2000-n2k-intro-tutorial

[31] ‘Manufacturer With Nmea 2000 Certified Product(S)’, National Marine Electronics Association. Accessed: May 11, 2024. [Online]. Available: https://web.nmea.org/certified-products/results?affcode=Manufacturer%20with%20NMEA%202000%20Certified%20Product(s)

[32] ‘NMEA Certified Products’, National Marine Electronics Association. Accessed: Apr. 26, 2024. [Online]. Available: https://web.nmea.org/certified-products/search

[33] ‘What is the OSI Model? - 7 OSI Layers Explained - AWS’, Amazon Web Services, Inc. Accessed: May 09, 2024. [Online]. Available: https://aws.amazon.com/what-is/osi-model/

[34] ‘Exploring the NMEA 2000 Protocol: Everything You Need to Know’. Accessed: May 10, 2024. [Online]. Available: https://www.linkedin.com/pulse/exploring-nmea-2000-protocol-everything-you-need-know-annamalai-bb1rc

[35] NETGEAR, ‘NETGEAR 4G LTE Modem - LB1120 | Mobile Broadband’, NETGEAR. Accessed: May 11, 2024. [Online]. Available: https://www.netgear.com/home/mobile-wifi/lte-modems/lb1120/

[36] T. Dunning, B. E. Friedman, M. K. Loukides, and R. Demarest, *Time series databases: new ways to store and access data*, First edition. Sebastopol, CA: O’Reilly Media, Inc, 2014.

[37] ‘Boat Monitor with Bilge Sentry | Sensar Marine’. Accessed: May 11, 2024. [Online]. Available: https://sensarmarine.com/en-us/products/boat-bilge-monitor

[38] A. Urke and P. Sivertsen, ‘Infomasjon om sensar teknologi’, presented at the Infomasjon om sensar teknologi, Thormøhlens Gate 41E, 5006 Bergen, Apr. 26, 2024.

[39] ‘Feature Creep: What Causes It and How to Avoid It’, Shopify. Accessed: May 10, 2024. [Online]. Available: https://www.shopify.com/partners/blog/feature-creep

[40] J. C. Pereira and R. de F. S. M. Russo, ‘Design Thinking Integrated in Agile Software Development: A Systematic Literature Review’, *Procedia Computer Science*, vol. 138, pp. 775–782, Jan. 2018, doi: 10.1016/j.procs.2018.10.101.

[41] M. Gunderloy, *Coder to Developer: Tools and Strategies for Delivering Your Software*, 1. Aufl. Newark: Sybex, 2006.

[42] C. Meinel and L. J. Leifer, *Design thinking: understand - improve - apply*. in Understanding Innovation. Berlin London: Springer, 2011.

[43] C. Owen, 'Design Thinking: Notes on Its Nature and Use', *Design Thinking*, 2006.

[44] S. De, 'A Novel Perspective to Threat Modelling using Design Thinking and Agile Principles', in *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, Nov. 2020, pp. 31–35. doi: 10.1109/PDGC50313.2020.9315844.

[45] A. Rai, *The Inner Building Blocks: A Novel to Apply Lean-Agile and Design Thinking for Digital Transformation*. New York, UNITED STATES: Business Expert Press, 2022. Accessed: May 07, 2024. [Online]. Available: http://ebookcentral.proquest.com/lib/hogskbergen-ebooks/detail.action?docID=6957421

[46] 'Scrum Guide 2020 Updates | Scrum.org'. Accessed: Mar. 06, 2024. [Online]. Available: https://www.scrum.org/scrum-guide-2020

[47] M. Holcombe, *Running an Agile Software Development Project*. Newark, UNITED STATES: John Wiley & Sons, Incorporated, 2008. Accessed: Mar. 06, 2024. [Online]. Available: http://ebookcentral.proquest.com/lib/hogskbergen-ebooks/detail.action?docID=427745

[48] P. Hamill, *Unit Test Frameworks: Tools for High-Quality Software Development*. O'Reilly Media, Inc., 2004.

[49] W. Sheikh, 'Teaching C++ programming using automated unit testing and test-driven development—Design and efficacy study', *Computer Applications in Engineering Education*, vol. 30, no. 3, pp. 821–851, 2022, doi: 10.1002/cae.22488.

[50] D. Thomas and A. Hunt, 'Mock objects', *IEEE Softw.*, vol. 19, no. 3, pp. 22–24, May 2002, doi: 10.1109/MS.2002.1003449.

[51] 'Unit Testing — PlatformIO v6.1 documentation'. Accessed: May 11, 2024. [Online]. Available: https://docs.platformio.org/en/stable/advanced/unit-testing/index.html

[52] Chauhan Naresh, *Software Testing - Principles and Practices*. Oxford University Press, 2010.

[53] J. F. Dooley, *Software Development, Design and Coding*. Berkeley, CA: Apress, 2017. doi: 10.1007/978-1-4842-3153-1.

[54] R. Chopra, *Software Testing: A Self-teaching Introduction*. Mercury Learning and Information, 2018.

[55] G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*. Hoboken, UNITED STATES: John Wiley & Sons, Incorporated, 2011. Accessed: May 12, 2024. [Online]. Available: http://ebookcentral.proquest.com/lib/hogskbergen-ebooks/detail.action?docID=697721

[56] ‘Exploring the Voltage Dilemma: Why Do Most Microcontrollers Operate at 3.3V or 5V?’ Accessed: May 02, 2024. [Online]. Available: https://www.linkedin.com/pulse/exploring-voltage-dilemma-why-do-most-operate-33v-5v-zoe-zhang-kndxc

[57] ‘ESP32 CAN-Bus board’, SK Pang Electronics Ltd. Accessed: Jan. 30, 2024. [Online]. Available: https://www.skpang.co.uk/products/esp32-can-bus-board

[58] ‘Arduino’, *Wikipedia*. Jun. 05, 2023. Accessed: May 12, 2024. [Online]. Available: https://no.wikipedia.org/w/index.php?title=Arduino&oldid=23599931

[59] ‘What is an Arduino? - SparkFun Learn’. Accessed: May 11, 2024. [Online]. Available: https://learn.sparkfun.com/tutorials/what-is-an-arduino/all

[60] ‘What is Arduino?’ Accessed: Apr. 30, 2024. [Online]. Available: https://www.arduino.cc/en/Guide/Introduction

[61] ‘About Arduino’. Accessed: May 11, 2024. [Online]. Available: https://www.arduino.cc/en/about

[62] ‘Getting Started with ESP-NOW (ESP32 with Arduino IDE) | Random Nerd Tutorials’. Accessed: May 12, 2024. [Online]. Available: https://randomnerdtutorials.com/esp-now-esp32-arduino-ide/

[63] ‘How can you program microcontrollers using an IDE?’ Accessed: Apr. 01, 2024. [Online]. Available: https://www.linkedin.com/advice/1/how-can-you-program-microcontrollers-using

[64] ‘PlatformIO IDE — PlatformIO v6.1 documentation’. Accessed: Mar. 15, 2024. [Online]. Available: https://docs.platformio.org/en/stable/integration/ide/pioide.html

[65] S. Chatterjee, ‘Why do We Use the Arduino Programming Language? How is it Helpful?’, Emeritus Online Courses. Accessed: May 12, 2024. [Online]. Available: https://emeritus.org/blog/coding-arduino-programming-language/

[66] ‘Getting Started with Arduino IDE 2 | Arduino Documentation’. Accessed: May 12, 2024. [Online]. Available: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2/

[67] R. Tucker, ‘Understanding the Feature Branching Strategy in Git’, Split. Accessed: May 10, 2024. [Online]. Available: https://www.split.io/blog/understanding-the-feature-branching-strategy-in-git/

[68] PlatformIO, *Deutsch: Logo von PlatformIO.* 2017. Accessed: May 12, 2024. [Online]. Available: https://commons.wikimedia.org/wiki/File:PlatformIO_logo.svg

[69] ‘ESP-IDF Programming Guide - ESP32 - — ESP-IDF Programming Guide v5.2.1 documentation’. Accessed: May 10, 2024. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/index.html

[70] ‘Question: Benefits of esp-idf over framework type arduino · Issue #62 · mmakaay/esphome-xiaomi_bslamp2’, GitHub. Accessed: Apr. 08, 2024. [Online]. Available: https://github.com/mmakaay/esphome-xiaomi_bslamp2/issues/62

[71] ‘C++’, *Wikipedia*. May 12, 2024. Accessed: May 12, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=C%2B%2B&oldid=1223530547

[72] ‘A Quick Guide to Embedded Programming Languages’. Accessed: Apr. 23, 2024. [Online]. Available: https://www.deepseadev.com/en/blog/embedded-programming-languages/

[73] ‘How is Python used in embedded systems?’ Accessed: Apr. 23, 2024. [Online]. Available: https://www.tutorialspoint.com/how-is-python-used-in-embedded-systems

[74] Atlassian, ‘What is Git | Atlassian Git Tutorial’, Atlassian. Accessed: May 10, 2024. [Online]. Available: https://www.atlassian.com/git/tutorials/what-is-git

[75] ‘About GitHub Importer’, GitHub Docs. Accessed: Mar. 05, 2024. [Online]. Available: https://docs.github.com/_next/data/tJy4SvGJ_ZWAKf3EDhBjC/en/free-pro-team@latest/migrations/importing-source-code/using-github-importer/about-github-importer.json?versionId=free-pro-team%40latest&productId=migrations&restPage=importing-source-code&restPage=using-github-importer&restPage=about-github-importer

[76] J. Cogswell, ‘Git: Mastering the Basics of Branch Merging’, Dice Insights. Accessed: May 10, 2024. [Online]. Available: https://www.dice.com/career-advice/git-mastering-basics-branch-merging

[77] D. Lettnin and M. Winterholer, Eds., *Embedded Software Verification and Debugging*, 1st ed. 2017. in Embedded Systems. New York, NY: Springer New York : Imprint: Springer, 2017. doi: 10.1007/978-1-4614-2266-2.

[78] ‘ESP32 Wi-Fi & Bluetooth SoC | Espressif Systems’. Accessed: Apr. 23, 2024. [Online]. Available: https://www.espressif.com/en/products/socs/esp32

[79] C. Ruth, ‘The Evolution of Wi-Fi Technology and Standards’, IEEE Standards Association. Accessed: Apr. 23, 2024. [Online]. Available: https://standards.ieee.org/beyond-standards/the-evolution-of-wi-fi-technology-and-standards/

[80] ‘Microcontroller Peripherals’. Accessed: Apr. 23, 2024. [Online]. Available: https://www.labcenter.com/blog/sim-microcontroller-peripherals/

[81] ‘UART, SPI, I2C Serial Communication Protocols And Bluetooth UART Modules’. Accessed: Apr. 23, 2024. [Online]. Available: https://www.linkedin.com/pulse/uart-spi-i2c-serial-communication-protocols-qursc

[82] ‘Ultra Fast Mode UFm’, I2C Bus. Accessed: Apr. 24, 2024. [Online]. Available: https://www.i2c-bus.org/http://www.i2c-bus.org/ultra-fast-mode-ufm/

[83] ‘Kave Oy’. Accessed: Jan. 30, 2024. [Online]. Available: https://www.kave.fi/Apps/

[84] ‘I2C-bus specification and user manual’. NXP Semiconductors, 2021.

[85] ‘CC1312R7 data sheet, product information and support | TI.com’. Accessed: May 03, 2024. [Online]. Available: https://www.ti.com/product/CC1312R7#params

[86] gcampbell-msft, ‘Serial Monitor’. Accessed: Apr. 26, 2024. [Online]. Available: https://learn.microsoft.com/en-us/cpp/embedded/serial-monitor?view=msvc-170

[87] ‘Protocol Buffers’. Accessed: May 07, 2024. [Online]. Available: https://protobuf.dev/

[88] ‘Protobuf vs JSON’, Wallarm. Accessed: May 07, 2024. [Online]. Available: https://lab.wallarm.com/what/protobuf-vs-json/

[89] ‘Time series database (TSDB) explained | InfluxData’. Accessed: Apr. 10, 2024. [Online]. Available: https://www.influxdata.com/time-series-database/#download

11 APPENDICES

Here is a list of the appended documents:

Appendix 1. Vision Document.

Appendix 2. Requirement Specification.

Appendix 3. System Documentation.

Appendix 4. Project Handbook.

Appendix 5. GANTT Diagram.

Appendix 6. Link to GitHub Repository:

https://github.com/h599000/N2K_Insight