University of South-Eastern Norway

www.usn.no

FMH606 Master's Thesis 2023

Industrial IT and Automation

Development and Testing of ASTERIX Category 240 Radar Display

Eivind Knudsen

Faculty of Technology, Natural sciences and Maritime Sciences

Campus Porsgrunn

University of South-Eastern Norway

www.usn.no

Course: FMH606 Master's Thesis, 2023

Title: Development and Testing of ASTERIX Category 240 Radar Display

Number of pages: 59

Keywords: Marine Radar Display, ASTERIX Category 240, Julia, Scan conversion, Plan Position Indicator

Student: Eivind Knudsen

Supervisor: Hans-Petter Halvorsen

External partner: Sea Surveillance AS a part of Seabrokers Group

Summary:

The primary objective of this master's thesis is to design, develop, and evaluate an open-source radar display application that effectively processes and visualizes network data in compliance with the EUROCONTROLs ASTERIX Category 240 standard. This project aims to provide a user-friendly and versatile tool for researchers, developers and maritime professionals to analyse and interpret radar data with ease.

Preface

This thesis has been prepared as the final part of the study program Master of Science in Industrial IT and Automation at the University of South-Eastern Norway at Porsgrunn. The thesis task was “Development and Testing of ASTERIX category 240 Radar Display”.

I would like to express my gratitude to Sea Surveillance and Seabrokers Group for their support and permission to undertake this project. I am appreciative of the guidance, insights and constructive feedback provided by my project supervisor from USN, Mr. Hans-Petter Halvorsen.

Bergen, 15.05.2023

Eivind Knudsen

Contents

Abbreviations

Abbreviation	Definition
ASTERIX	All Purpose Structured EUROCONTROL Surveillance Information Exchange, standard for exchange of information.
Azimuth	Angle of antenna in horizontal plane relative to north
CW	Continuous Wave
ECDIS	Electronic Chart Display and Information System
EM Wave	Electromagnetic Wave
FMCW	Frequency Modulated Continuous Wave
FRN	Field Reference Number
FSPEC	Field Specification
IP	Internet Protocol
LAN	Local Area Network
LSB	Least Significant Bit
NM	Nautical Mile, unit of distance (1852 meters)
Octet	A set of 8-bits containing information, single or multiple of which are used in constructing a data item as defined in ASTERIX [1]
OSI model	Open Systems Interconnection model [2]
Packet	A UDP datagram containing binary data
PPI	Plan Position Indicator, is a type of radar display
PRF	Pulse repetition frequency, is the number of pulses of a repeating signal in a specific time unit.
RCS	Radar cross section, is a measure of how detectable an object is by radar

Abbreviation	**Definition**
REPL	Read-Eval-Print Loop, allows quick and easy evaluation of Julia statements
RF	Radio frequency
SAC	System Area Code, assigned to a geographical area [1]
SIC	System Identification Code, assigned to system [1]
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UTC	Coordinated Universal Time

1 Introduction

In this project, a high-resolution commercial marine radar supplied by Sea Surveillance AS is utilized to obtain real-time radar data in the CAT240 format via Ethernet. An open-source radar display application has been developed as a cost-effective solution for monitoring and analysis purposes. This application offers an accessible alternative for users seeking to effectively manage and evaluate radar data without incurring the expenses associated with proprietary software. A simplified system diagram is depicted in Figure 1.1.

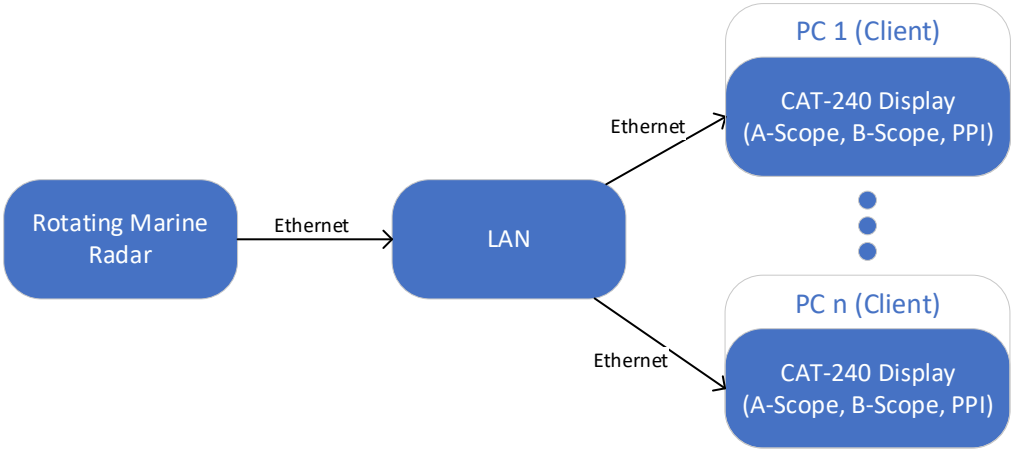


Figure 1.1: A radar-based system where CAT240 video data is directly output to an Ethernet network. The radar sensor collects and processes raw data, converting it into CAT240 video format. The data is then transmitted through the Ethernet interface for real-time analysis or display on a standard PC running the application developed in this project.

1.1 Background

Marine radar is an indispensable tool for navigation, ensuring the safety of life at sea and efficient maritime operations. Traditionally, radar systems relied on analog interfaces for video output or proprietary digital signals. In recent years, however, the EUROCONTROL Specification for Surveillance Data Exchange ASTERIX Category 240 Radar Video Transmission standard has emerged [1] [3], streamlining the encoding and exchange of radar video data across various systems via ethernet. This standard has been widely adopted by numerous radar manufacturers, enhancing interoperability and data sharing capabilities among different maritime stakeholders.

A standard marine scanning radar emits pulses radially from a rotating transmit/receive antenna. In order to display a radar image on a screen, the original radar video data from the sensor must be acquired and undergo a scan conversion process, which transforms the polar coordinates into rectangular coordinates. With the growing adoption of the open ASTERIX Category 240 standard, many commercially available radars now output the radar video signal in compliance with this standard. Specialized surveillance data processing software to decode the CAT240 data and scan convert the images are available on the market, however, there are currently no open-source display solutions available.

1.2 Sea Surveillance AS

Sea Surveillance AS is a software and high-end solutions company located in Bergen, Norway. It provides turnkey solutions for the maritime and offshore industries.

Sea Surveillance AS is a part of Seabrokers Group, a Norway-based company with a strong presence in the maritime and offshore industries. The group has been in operation for more than four decades, demonstrating a robust and consistent growth throughout its history.

1.3 Objective

The main objective of this master's thesis is the development and evaluation of a PC-based marine radar display application that complies with the previously mentioned CAT240 standard. The application is designed to receive, process, and display radar video data from a variety of sources, such as shipborne radars and coastal radar stations, serving as a valuable tool for troubleshooting and testing radar video output.

Notably, the developed application will be open source, providing unrestricted access to the code for use, modification, and distribution. Considering the current lack of open-source marine radar display applications conforming to the CAT240 standard, this project aims to deliver a cost-free resource for diagnosing and testing CAT240 radar video output, while promoting adaptability and innovation through its open-source nature.

1.4 Report Structure

Chapter 2 provides the fundamentals of maritime radar technology.

Chapter 3 introduces the various methods of distributing radar video.

Chapter 4 offers an overview of the key software tools utilized and the selected programming language, explaining the reasoning behind the selection of programming language.

Chapter 5 provides the requirements and design of the project.

Chapter 6 gives an overview of the hardware configuration.

Chapter 7 provides insights into the software development process.

Chapter 8 offers a summary of the CAT240 Video Display Application that has been developed.

Chapter 9 discusses the project outcomes.

Chapter 10 suggest opportunities for further enhancements and developments.

Chapter 11 presents a conclusion, addressing the project's objectives and goals.

2 Fundamentals of Maritime Radar Technology

This chapter provides an overview of the fundamentals of radar technology, covering its historical development, basic principles, key components, and various types of radar systems. Additionally, it introduces the different radar display types and discusses the performance parameters and limitations of radar systems in the maritime context. This background information will help readers gain a solid understanding of radar technology and its significance in the development of a PC-based marine radar display application compliant with the CAT240 standard.

2.1 Historical Development of Radar Technology

The history of radar technology has been marked by continuous innovations and advancements, particularly in the context of maritime applications. This sub-chapter highlights the key milestones in the development of maritime radar technology [4], from its early beginnings during World War II to the sophisticated systems in use today.

2.1.1 Early Beginnings: World War II and Shipborne Radar Systems

During World War II, the first shipborne radar systems were developed to provide naval forces with improved navigation and target detection capabilities. These early systems played a crucial role in enhancing situational awareness at sea and increasing the effectiveness of naval operations.

2.1.2 Post-War Era: Emergence of Commercial Marine Radar

Following the end of the war, radar technology was adapted for civilian maritime applications, leading to the development of commercial marine radar systems. These systems were designed to enable safer navigation and collision avoidance for commercial shipping, improving overall maritime safety and efficiency. Automatic Radar Plotting Aid, also known as ARPA radars, emerged in the 1960s, substantially reducing the need for manual calculations and enhancing situational awareness for mariners.

2.1.3 Modern Marine Radar Systems

Over the past few decades, marine radar technology has continued to evolve, incorporating various advancements that have further improved its performance and functionality. Some notable developments include:

- Digital signal processing: The introduction of digital signal processing in marine radar systems has led to enhanced target detection, tracking, and discrimination capabilities.

- Advanced display technology: Modern marine radar displays, such as the plan position indicator, have greatly improved the presentation and interpretation of radar data, providing navigators with more accurate and actionable information.
- Multifunction radar systems: The development of multifunction radar systems has allowed for the integration of various data sources, such as GPS, AIS, and electronic chart systems, enabling more comprehensive and informed decision-making for maritime operators.

2.2 Basic Radar Principles

Radar, an acronym derived from the term radio detection and ranging, embodies its primary function. Radar systems operate based on a set of fundamental principles that allow for the detection, processing, and display of information about surrounding objects. This sub-chapter provides a concise overview of the basic radar principles, including the radar range equation, the Doppler effect, and target resolution.

2.2.1 Echo Principle

Radars employs the echo principle in detecting objects or targets by transmitting a radio energy pulse and receiving a portion of the reflected energy, known as an echo. This process is similar to the reflection of sound waves off geographical features. Essential aspects to bear in mind include [4]:

- Echoes are never as loud as the original pulse.
- The likelihood of detecting an echo depends on the energy and duration of the transmitted pulse.
- To identify nearby targets, short pulses are required to prevent the echo from being obscured by the transmitted pulse.
- A sufficiently long interval between transmitted pulses allows time for distant echoes to return.

While the sound analogy is useful, it has limitations due to differences in the character and behavior of radio and sound waves, such as radio waves' much higher speed. The time between the transmission of a pulse and the reception of its corresponding echo depends on the pulse's speed and the distance it has traveled. By knowing the pulse speed and measuring the elapsed time, the target's range can be calculated. For radar ranging, radio waves travel approximately at the speed of light, 300.000 kilometers per second. This value enables a simple relationship between target range and elapsed time for pulse transmission and echo reception as given by Equation (2.1).

$$R = \frac{ct}{2} \tag{2.1}$$

Where:

- R [m] is the range between the radar and the target.
- c = 299 792 458 [m/s] and is the speed of light in vacuum.

- t [s] is the round-trip time duration it takes for the radar signal to travel from the transmitter to the target and back to the receiver.

2.2.2 Radar Range

The radar range equation is a fundamental equation used to estimate the maximum range at which a radar system can detect a target. The equation relates the power transmitted by the radar, the antenna gain, the target's radar cross-section, the sensitivity of the receiver, and other factors to determine the detection range. The radar range equation is given by Equation (2.2):

$$P_r = \frac{P_t G_t G_r \lambda^2 \boldsymbol{\sigma}}{(4\pi)^3 R^4} \tag{2.2}$$

Where:

- P_r [W] is the received power by the radar.
- P_t [W] is the transmitted power.
- G_t is the gain of transmit antenna (unitless).
- G_r is the gain of receive antenna (unitless).
- λ [m] is the wavelength of the carrier.
- σ [m^2] is the mean RCS of the target.
- R [m] is the range from radar to target.

The equation assumes that the radar system operates under ideal conditions, and actual performance may vary due to factors such as environmental conditions, interference, and system limitations.

2.2.3 Pulse Repetition Frequency and Pulse Repetition Interval

Pulse repetition frequency refers to the number of pulses transmitted per second by the radar system. Pulse repetition interval is the time interval between successive pulses [5]. PRF and PRI are essential parameters that influence the radar system's maximum unambiguous range and velocity measurement capabilities.

2.2.4 Doppler Effect

The Doppler effect is a phenomenon that occurs when there is relative motion between a radar system and a target. As the radar waves reflect off the moving target, their frequency changes [5]. This frequency shift can be used to determine the target's radial velocity and is the basis for Doppler radar systems, which can detect and track moving objects more effectively.

2.2.5 Target Resolution

Target resolution refers to the ability of a radar system to distinguish between two closely spaced targets [4]. There are two types of target resolution: range resolution and angular resolution.

Range resolution: Range resolution depends on the radar pulse width and the bandwidth of the transmitted signal. A shorter pulse width or a larger bandwidth result in better range resolution, enabling the radar to separate closely spaced targets in range.

Angular resolution: Angular resolution depends on the antenna's beamwidth. A narrower beamwidth leads to better angular resolution, allowing the radar system to differentiate between closely spaced targets in azimuth or elevation.

2.3 Components of a Radar System

Radar systems consist of several key components that work together to detect and display information about surrounding objects. This sub-chapter provides an overview of the primary components found in a radar system, including the transmitter, antenna, receiver, signal processing and display unit. The radar antenna emits a microwave signal, which is then reflected and picked up by a receiving device. The electrical signal picked up by the receiving antenna is called echo signal or return. The radar signal is generated by a powerful transmitter and received by a highly sensitive receiver. The operating principle of a radar system is illustrated in Figure 2.1.

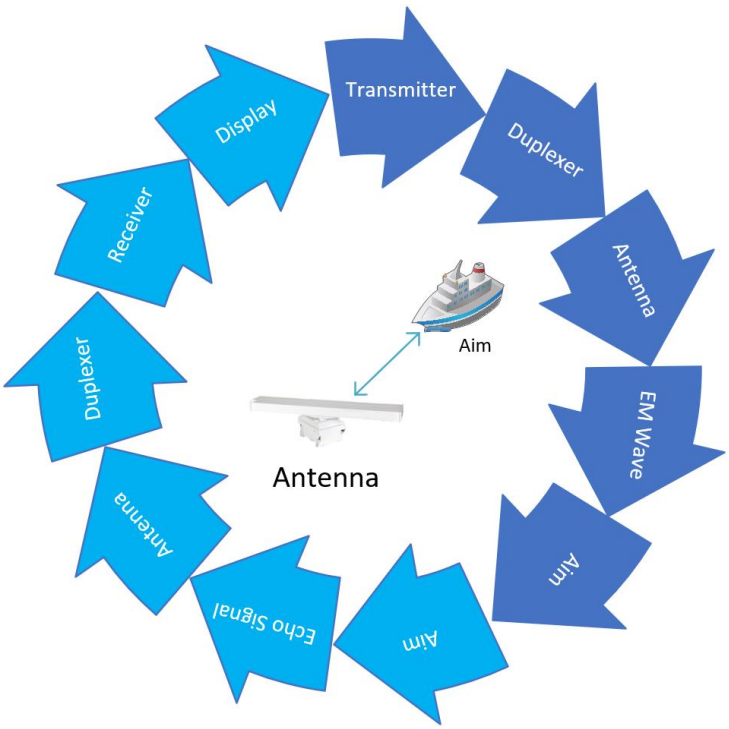


Figure 2.1: The block diagram illustrates the operating principle of a radar system. The transmit path is represented by dark blue boxes, while the receive path is depicted by light blue boxes.

2.3.1 Transmitter

The transmitter is responsible for generating and amplifying the radar signal [4]. It produces high-frequency electromagnetic waves, which are then directed towards the target using the antenna. The power and frequency of the transmitted signal determine the radar system's range and resolution capabilities.

2.3.2 Duplexer

In radar systems using a single antenna for both transmission and receiving signals, a device is needed to prevent the direct transmission of RF energy from the transmitter to the receiver [4]. Two common devices used for this purpose are the duplexer and the circulator. A duplexer is an electronic switch that alternates the antenna connection between the transmitter and the receiver. A circulator is a passive, non-reciprocal device that allows RF signals to pass through in one direction while isolating the other ports. In a radar system, it directs the transmitted signal from the transmitter to the antenna and then directs the received signal from the antenna to the receiver, effectively separating the transmit and receive paths.

2.3.3 Antenna

The antenna plays a dual role in a radar system, transmitting the electromagnetic waves generated by the transmitter and receiving the reflected signals from the target. Antennas come in various designs and configurations, with their shape and size influencing the radar's beamwidth, gain, and directivity.

2.3.4 Receiver

The receiver's primary function is to detect and process the radar echoes or reflected signals from the target. It amplifies the weak signals received by the antenna, filters out noise, and converts the analog signals into digital data for further processing by the display unit.

2.3.5 Display

The signal processing and display unit is responsible for analyzing and interpreting the radar data. It applies various algorithms to the digital data received from the receiver to extract meaningful information, such as target range, bearing, and velocity. The processed data is then displayed on the radar screen, providing the operator with a visual representation of the surrounding environment.

2.4 Types of Radar Systems

Marine radar systems are essential tools for navigation, collision avoidance, and maintaining situational awareness at sea. Several types of radar systems have been developed for marine applications, each with specific advantages and use cases [6]. This sub-chapter provides an overview of three primary types of marine radar systems. The difference in transmitted signal is depicted in Figure 2.2.

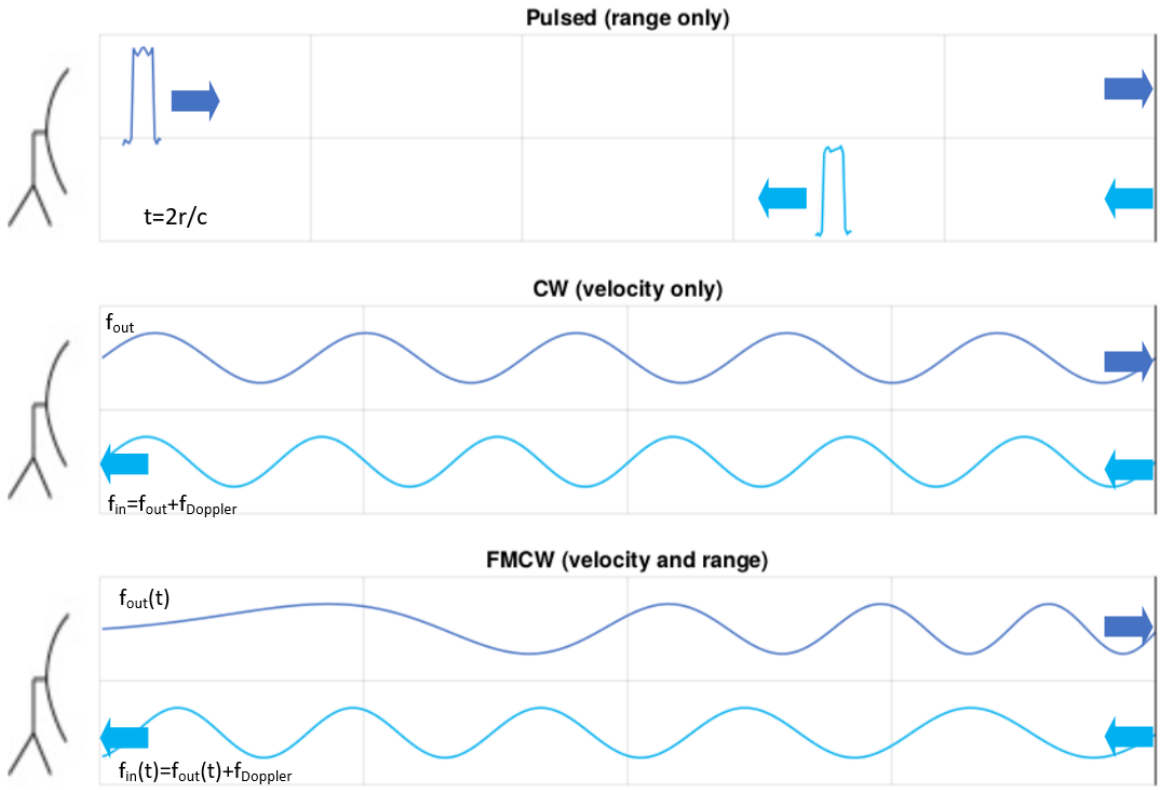


Figure 2.2: The functional principles of different radar types are outlined schematically. A Pulsed radar calculates the range based on the round-trip time of a single pulse (first row). The Continuous Wave (CW) radar assesses velocity by analyzing the Doppler frequency shift (second row). Finally, the Frequency Modulated Continuous Wave (FMCW) radar (third row) examines the frequency disparity between transmitted and received signals, providing both velocity and range data.

2.4.1 Pulse Radar

Pulse radar is the most common type of marine radar system. It works by transmitting short pulses of radio waves and measuring the time it takes for the echoes to return after reflecting off targets. Pulse radar systems provide excellent range resolution and are highly effective in detecting and tracking targets at varying distances.

- X-Band Radar: X-band radar systems operate at a frequency range of 8 to 12 GHz and are known for their high-resolution imagery and relatively short range. They are well-

suited for navigating in congested waterways and detecting small targets, such as buoys and other vessels.

- S-Band Radar: S-band radar systems operate at a frequency range of 2 to 4 GHz, offering longer range capabilities compared to X-band radars. They are less affected by rain and other environmental factors, making them suitable for open-sea navigation and long-range target detection.

2.4.2 Continuous Wave Radar

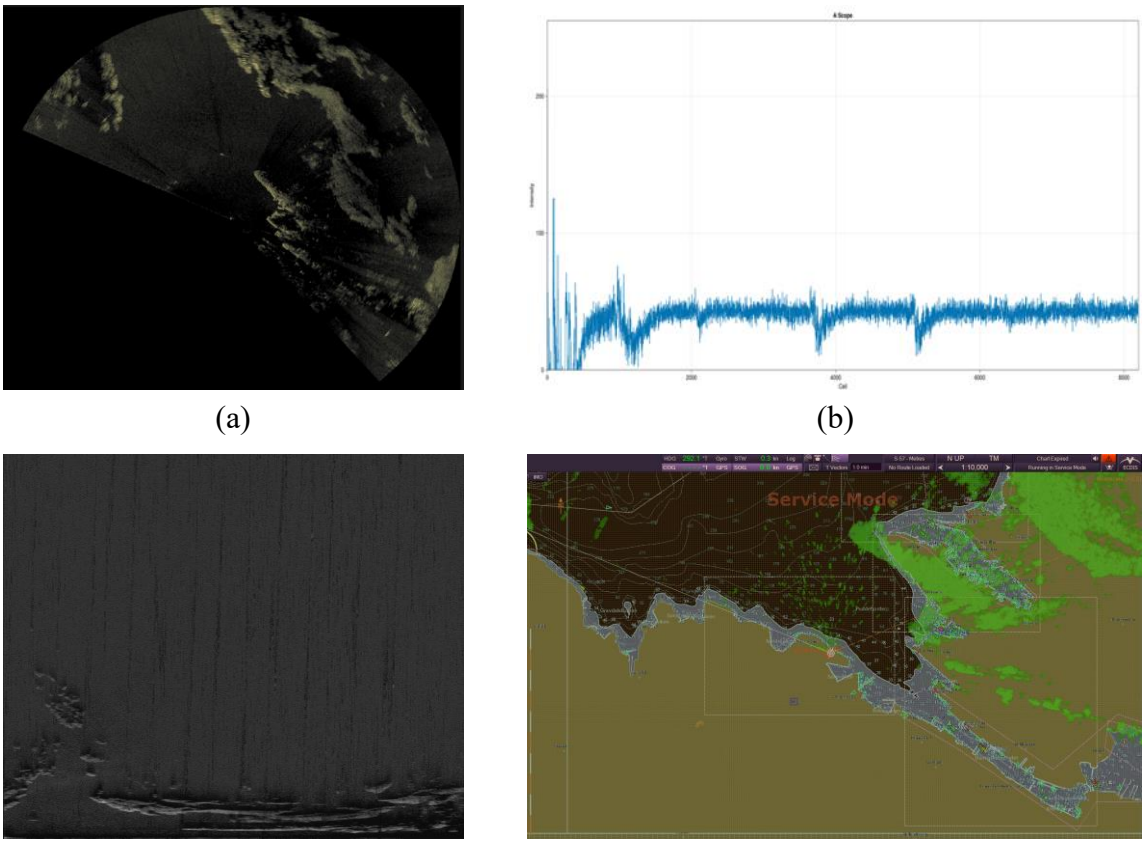
Continuous Wave radar systems transmit a continuous signal at a constant frequency, rather than short pulses, and measure the frequency shift (Doppler effect) of the returning echoes to determine the target's radial velocity but cannot measure the range. Plain CW radar systems are employed in applications like police radar guns, where the primary focus is on measuring the target’s velocity rather than determining its range.

2.4.3 Frequency Modulated Continuous Wave Radar

Frequency Modulated Continuous Wave radar systems, also known as chirp radars, transmit a continuous wave signal with a frequency that changes over time. By analyzing the difference in frequency between the transmitted and received signals, FMCW radars can determine both the range and radial velocity of targets. FMCW radar systems are often used for short-range navigation, docking, and collision avoidance due to their high-resolution capabilities and relatively low power requirements.

2.5 Radar Displays

Radar displays play a vital role in presenting the information gathered by radar systems in a visually intuitive and interpretable manner. This sub-chapter provides an overview of the primary types of radar displays illustrated in Figure 2.3, with a focus on their use in maritime applications.



(a) (b)

(c) (d)

Figure 2.3: Illustration of four common radar display types:
(a) PPI; (b) A-Scope; (c) B-Scope; (d) ECDIS.

2.5.1 Plan Position Indicator Display

The Plan Position Indicator display is the most commonly used radar display in maritime applications. It presents radar data in a top-down, two-dimensional view, with the radar's own position at the center of the display [7]. The PPI display allows navigators to see the range and bearing of targets relative to their vessel, enabling accurate navigation and collision avoidance.

2.5.2 A-Scope Display

The A-Scope display is a one-dimensional radar display that shows the received signal strength as a function of range. In maritime applications, A-Scope displays can be used to analyze radar echoes and assess target characteristics, such as distance and signal strength [7]. However, the A-Scope display does not provide bearing information, limiting its usefulness for navigation purposes.

2.5.3 B-Scope Display

The B-Scope display, also known as the Range-Azimuth display, is a two-dimensional radar display that shows target information in terms of range and azimuth [7]. The B-Scope display is similar to the PPI display but presents the data in a rectangular grid instead of a polar coordinate system. The horizontal axis typically corresponds to the azimuth, and the vertical axis indicates the range, the B-Scope display is less frequently utilized because the PPI display offers a more user-friendly approach to evaluating target locations and motion.

2.5.4 Electronic Chart Display and Information System

Electronic Chart Display and Information System is a computer-based system that integrates radar data, electronic navigational charts, and other data sources, such as GPS and AIS. ECDIS provides a comprehensive and dynamic representation of the vessel's surroundings, enhancing situational awareness and navigational capabilities [7]. In many modern maritime applications, ECDIS is integrated with radar displays, allowing for seamless data fusion and interpretation.

2.6 Performance Parameters and Limitations of Radar Systems

Understanding the performance parameters and limitations of radar systems is essential for maximizing their effectiveness in maritime applications. This sub-chapter provides an overview of the key performance parameters and the inherent limitations of radar systems in marine environments [4].

2.6.1 Performance Parameters

Several crucial performance parameters for radar systems include the following:

- *Range* is the maximum distance at which a radar system can reliably detect targets. Factors affecting radar range include transmitted power, antenna gain, target size, mounting height, line of sight and sensitivity of the receiver.
- *Resolution* refers to the radar's ability to distinguish between closely spaced targets. Two types of resolution are essential in marine applications: range resolution and azimuth resolution. Range resolution depends on the pulse width and signal bandwidth, while azimuth resolution depends on the antenna beamwidth.

- *Accuracy* is the measure of how closely the radar system can determine the target's true position, velocity, and other parameters. In maritime applications, accuracy is crucial for navigation and collision avoidance.
- *Detection Probability* is the likelihood that a radar system will detect a target of a given size, shape, and composition at a specific range. Factors affecting detection probability include target size, radar cross-section, and system noise.

2.6.2 Limitations of Radar Systems

Constraints affecting radar systems encompass:

- *Radar Clutter* refers to unwanted echoes from non-target objects, such as sea waves, rain, snow, ice and landmasses. In maritime applications, clutter can mask target echoes and reduce the radar's detection capability.
- *Multipath Propagation* occurs when radar signals are reflected off the sea surface or other obstacles, causing multiple signal paths to reach the radar receiver. This phenomenon can result in ghost targets and distorted target positions.
- *Doppler Ambiguity* arises in continuous-wave radar systems when the target's radial velocity is incorrectly estimated due to the limited frequency resolution of the system. This limitation can lead to incorrect target tracking and positioning.
- *Radio Frequency Interference* occurs when other radio frequency sources, such as other radar systems or communication devices, interfere with the radar's operation. Radio frequency interference can degrade radar performance and cause false targets or loss of target detection.

3 Radar Video Distribution

This chapter examines radar video distribution, highlighting the traditional methods, the use of the ASTERIX CAT240 standard, and Ethernet multicast technology. The benefits and limitations of each approach will be explored, along with their practical applications in the maritime industry. Some of the key benefits of CAT240 radar, with respect to the requisite hardware, are illustrated in Figure 3.1.

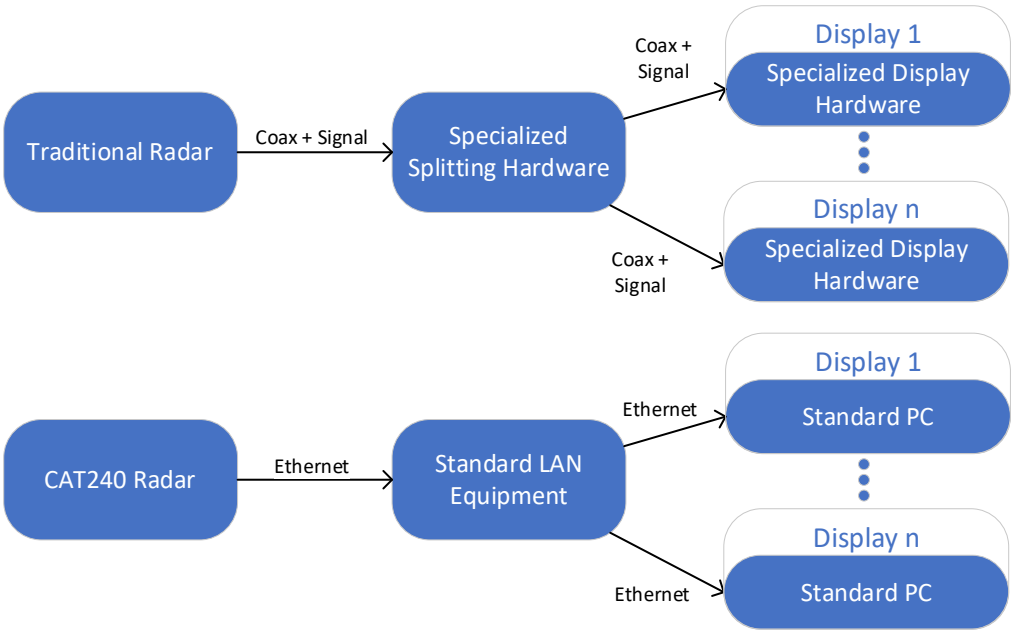


Figure 3.1: Overview of the specialized hardware necessary for a traditional radar system at the top, while demonstrating the utilization of commercially available off-the-shelf cables, switches, and PCs for a CAT240 radar at the bottom.

3.1 Traditional Methods

Traditional radar video distribution methods involve the use of analog signals transmitted over coaxial cables. These methods have been widely used in the maritime industry for several decades. However, they come with several limitations, including signal degradation, limited scalability, and the need for dedicated cables and hardware. An example of such a setup using a VisionMaster FT radar and specialized hardware from Sperry Marine is shown in Figure 3.2.

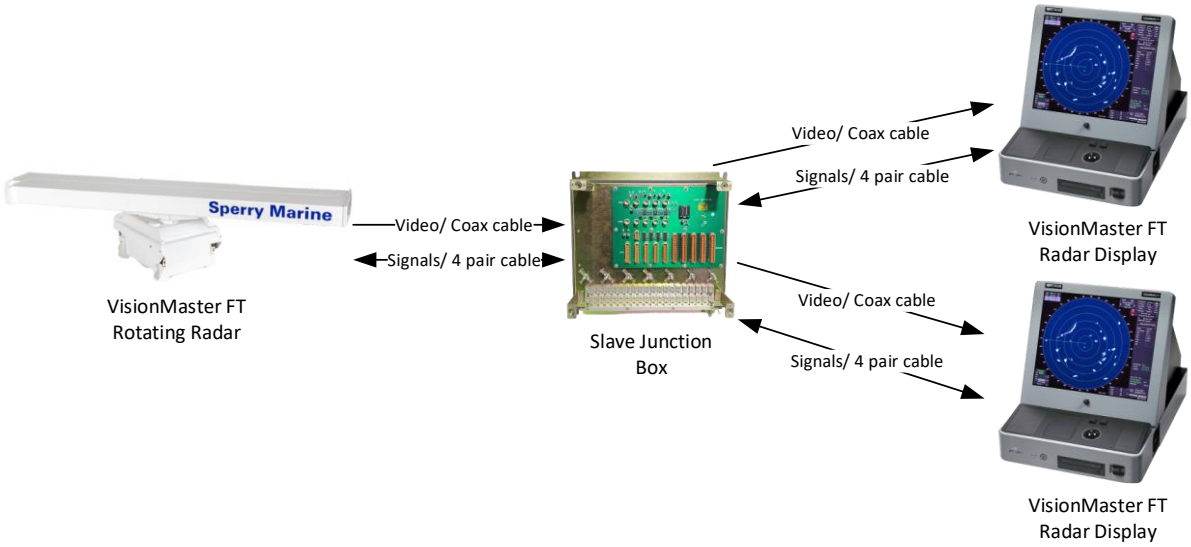


Figure 3.2: An example of a hardware configuration featuring a VisionMaster FT marine radar, a device known as a slave junction box that serves as a splitter, and two connected displays, all provided by Sperry Marine. To maintain simplicity, the power cables have not been included in the representation.

Advantages of traditional methods:

- Simplicity: Analog video distribution is relatively straightforward, involving minimal setup and configuration.
- Compatibility: Many legacy radar systems still rely on analog video distribution, making it a viable option for certain applications.

Disadvantages of traditional methods:

- Signal degradation: Analog signals are susceptible to noise and interference, resulting in a decline in video quality over long distances.
- Limited scalability: Expanding the system with additional radar sources or displays requires the installation of more cables and hardware, which can be expensive and cumbersome.

3.2 ASTERIX CAT240 Standard

Since being specified as a standard in 2009, numerous radar manufacturers have chosen ASTERIX CAT240 as their preferred network video standard. Major brands such as Furuno, Simrad, Terma and Sperry Marine all deliver radars that feature CAT240 output. This compatibility enables the radar video to be processed by any display applications supporting this format. Using a computer-based software application, it is possible to interface directly with the radar through commercial off the shelf Ethernet components for video reception, eliminating the need for costly specialized hardware to distribute or split the signal from the radar. An example of such a setup using a VisionMaster Net radar from Sperry Marine together with COTS network components is shown in Figure 3.3.

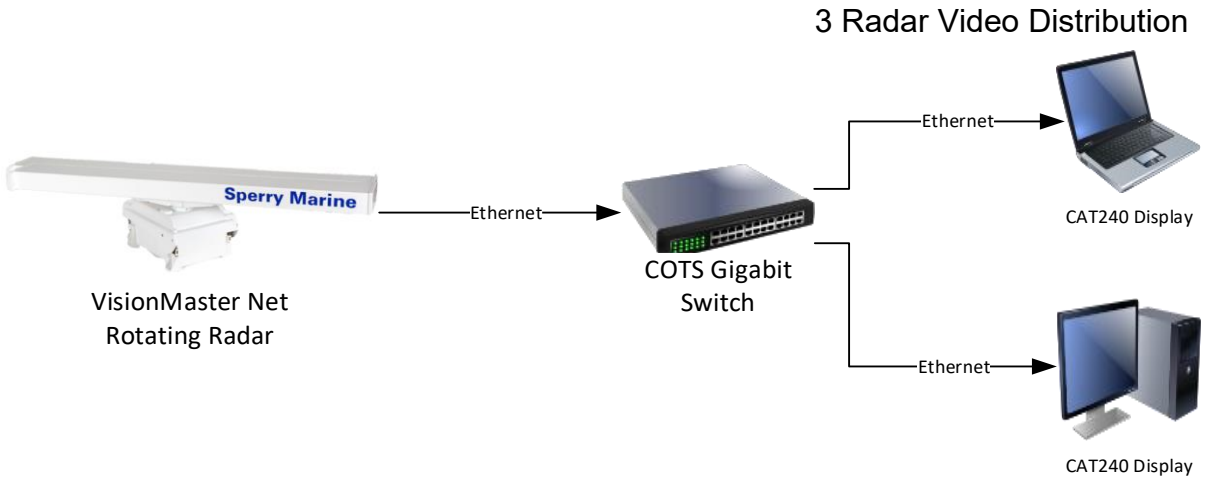


Figure 3.3: Hardware configuration of the system showing two computers connected, each running the CAT240 display application. However, any number of computers can be connected.

CAT240 standard streamlines the exchange of radar video between a data source and one or more destinations. The video data is in polar format, comprising a sequence of radar returns at varying angles, each containing a set of samples representing the radar video at increasing range. For instance, a radar with 1.8kHz pulse repetition frequency, and a range digitized into 8192 samples for each pulse results in a CAT240 stream with 1800 messages per second, where each message encodes 8192 samples of video data, using high definition this give approximately 130Mbps.

Each radar return, as defined by the CAT240 standard, consists of a header and a data block. The header contains information such as the angle represented by the block, radar video range, time of day, and more. A single video segment is included in each ASTERIX Video Message as depicted in Figure 3.4. A sector can be composed of either a single or multiple segments, depending on the number of cells representing the range for that sector. It is worth mentioning that the starting range for a sector, i.e., the turning unit, is not necessarily required to be from 0. ASTERIX defines that the Radar Video must always be stabilized relative to North.

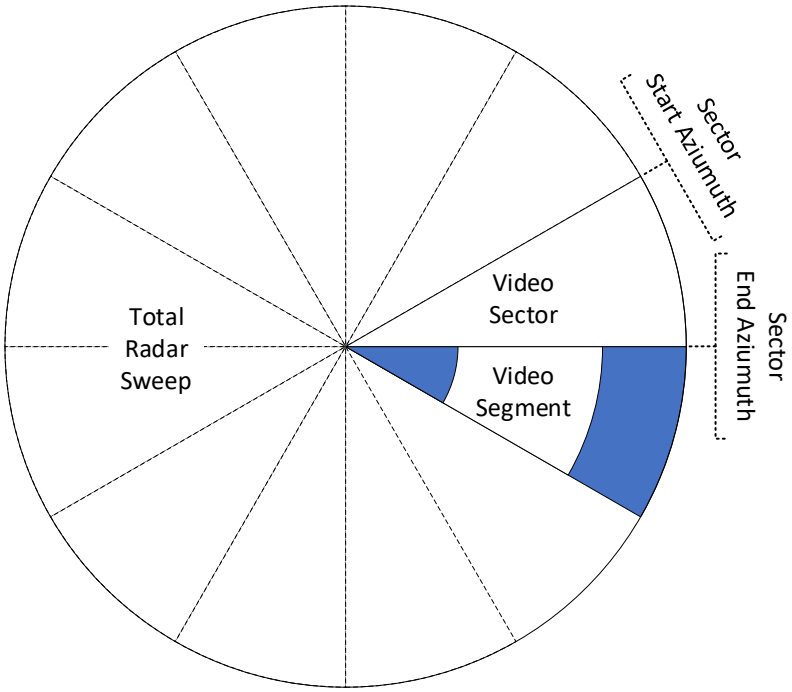


Figure 3.4: Illustration of a Video Sector and a Video Segment.

To obtain the range in meters for a cell within the data stream, Equation (3.1) should be utilized.

$$range = \frac{\big(c(cell\ duration)(start\ range + cell\ position - 1)\big)}{2} \tag{3.1}$$

Where c = 299 792 458 m/s is the speed of light in vacuum.

The ASTERIX specification focuses on the Presentation and Application layers (layers six and seven) as outlined by the Open Systems Interconnection Reference Model Standard [4]. The CAT240 standard establishes a unified format for exchanging radar data within the maritime domain, fostering interoperability among various systems and platforms. This standardized approach enables the efficient transmission and sharing of radar video data across diverse networks.

Advantages of using ASTERIX CAT240:

- Improved data quality: Digital radar data is less susceptible to noise and interference, resulting in higher-quality video.
- Scalability: The standard allows for easier integration with other systems and the addition of new radar sources or displays.

- Interoperability: The use of a common data format ensures compatibility between various maritime systems and platforms.

Disadvantages of using ASTERIX CAT240:

- Compatibility issues: Some legacy radar systems may not support the ASTERIX CAT240 standard, requiring upgrades or modifications.

3.3 Ethernet Multicast

The ASTERIX Specification does not cover the lower telecommunication support layers (OSI model layers one to five). Transmission of surveillance information encoded with ASTERIX can be achieved through any suitable communication medium, such as packet-switched Wide Area Networks or Local Area Networks. Ethernet multicast, a network technology enabling efficient distribution of radar video data to multiple recipients at once as depicted in Figure 3.5, employs UDP multicast to alleviate network congestion and improve overall system performance.

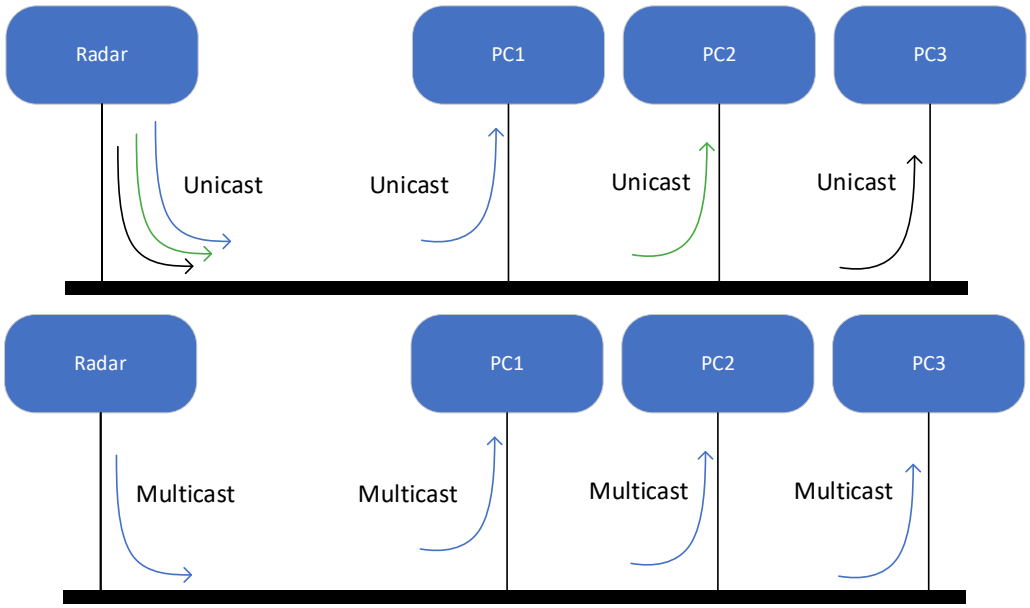


Figure 3.5: The fundamental distinction between unicast and multicast communication: The top section of the figure demonstrates unicast, featuring separate one-to-one data transmissions between individual sender-receiver pairs. The lower section showcases multicast, with a single sender efficiently transmitting data simultaneously to multiple receivers through optimized network infrastructure.

Advantages of using Ethernet multicast:

- Efficiency: Ethernet multicast conserves network bandwidth by sending data to multiple recipients simultaneously, reducing network congestion.
- Scalability: The technology can accommodate large numbers of radar sources and displays without significant impact on network performance.

- Flexibility: Ethernet multicast can be easily integrated with existing network infrastructure and other communication systems.

Disadvantages of using Ethernet multicast:

- Complexity: The setup and configuration of Ethernet multicast can be more complex than traditional analog video distribution methods.
- Security concerns: The multicast nature of the data transmission may expose the radar video data to unauthorized recipients if proper security measures are not implemented.

3.3.1 Network Load Calculations

The calculations below estimate the network load when implementing a CAT240 interface. Note that this primarily considers the video payload's impact, which is by far the most significant factor. It does not account for all effects of protocol headers or the splitting of datagrams larger than the Maximum Transmission Unit. The data rate is calculated using Equation (3.2).

$$Rate = \frac{NpB}{10^6} \tag{3.2}$$

Where:

- Rate is the data per time unit [MB/s]
- N is the number of video cells (unitless)
- P is the Pulse repetition Frequency [Hz]
- B is the size of video cell in bytes [B]

Assuming a network bandwidth of 125 MB/s for Gigabit Ethernet, Equation (3.3) can be applied to determine the network load. Table 3.1 presents some sample values calculated using Equations (3.2) and (3.3).

$$Network\ Load = \frac{Rate}{Network\ Bandwidth} 100\% \tag{3.3}$$

Table 3.1: Video Message Network Load Calculations

Video Message Frequency [Hz]	Video Cells per Sector	Video Resolution [bits]	Bytes per Video Cell	Number of IP Packets required	Rate [MB/s]	Network Load [%]
785	1024	4	0.5	1	0.40	0.32
1800	1024	4	0.5	1	0.92	0.74
785	8192	8	1	6	6.43	5.14
1800	8192	8	1	6	14.75	11.80
3000	8192	16	2	11	49.15	39.32
5000	1024	16	2	2	10.24	8.19

4 Software Tools

This chapter presents an overview of the essential software tools employed in the development of the application, elucidating not only their main functionalities but also the underlying reasoning for their selection. Through an exploration of the importance and capabilities of these tools, valuable insights into their contributions to the successful execution of the project are provided. Table 4.1 provides a summary of the main software tools employed in this project, with detailed descriptions of the first five in subsequent subchapters.

Table 4.1: Overview of main software tools used in this project.

Name	**Description**
Julia	High-performance, dynamic, modern programming language.
Visual Studio Code	Versatile, extensible code editor platform.
Pluto	Interactive, reactive Julia programming environment.
Wireshark	Powerful network protocol analyser tool.
Colasoft Packet Player	A network packet player enables the replay of captured Wireshark files onto a LAN.
Microsoft Project	Comprehensive project management software solution.
Visio	Versatile diagramming and vector graphics application.

4.1 Julia Programming Language and Frameworks

Julia was chosen as the primary programming language for the development of the CAT240 radar video processing application due to its performance-oriented design and ease of use [8]. Its syntax is similar to MATLAB and Python, making it easy to learn for users who are familiar with these languages. Julia is a high-level dynamic programming language designed for numerical and scientific computing, with a strong focus on performance. Compared to MATLAB and Python, Julia has shown to be faster and has a lower memory usage for a range of tasks such as linear algebra and numerical integration. A general guide as to how fast Julia is compared to a wider array of languages is found as general benchmarks on Julia's own website [9]. Furthermore, upon examining the codebase for NumPy, which is Python's fundamental library for scientific computing, it becomes apparent that crucial performance-driven components are implemented in C [10]. This implies that pure Python may not deliver high performance.

The Makie library was selected as the primary GUI library for the signal processing application due to its advanced graphical capabilities and ease of use. Makie provides an intuitive interface for creating complex visualizations and has a wide range of tools for data exploration and analysis. It is also designed to be fast and efficient, which makes it an ideal choice for developing high-performance GUIs for signal processing applications. [11]

The combination of Julia and Makie enables the development of a CAT240 processing application that provides advanced data analysis and visualization capabilities while maintaining high performance and ease of use. The use of Julia and Makie for GUI development in the CAT240 processing application ensures that users can enjoy both advanced data analysis and visualization capabilities as well as a high level of performance, which are essential for the development of a user-friendly and efficient CAT240 processing application. For example, processing algorithms must be able to deal with the increasing amount of data generated by modern radars, which can be several gigabytes per minute. The processing must also be done in real-time to provide timely information to the operator.

One potential disadvantage of using Julia for development is its relatively small user community compared to more established languages like Python and MATLAB. However, this drawback is getting smaller every day as an increasing number of developers discover the benefits of Julia for scientific computing and data analysis.

Additionally, Julia's popularity has been growing rapidly in recent years, and it has been adopted by major companies and organizations including Cisco, Pfizer, IBM, and NASA are notable science-driven efforts requiring high performance that are using Julia [12]. As Julia continues to gain traction in the scientific computing community, its user community is expected to grow and the availability of resources and support for Julia development will continue to improve.

Overall, while the smaller user community may have been a potential disadvantage in the past, the growing popularity of Julia and its increasing adoption by major organizations suggest that this drawback is becoming less relevant with time.

4.2 Visual Studio Code – Code Editor

Visual Studio Code is a highly popular, lightweight, and versatile code editor developed by Microsoft. It is designed to support various programming languages and development frameworks through the use of extensions, which provide additional features and functionality [13]. With a wide range of features such as syntax highlighting, code navigation, debugging, and built-in Git support, Visual Studio Code has emerged as a preferred development environment for many programmers across various platforms.

One such extension for VSCode is the Julia extension, which brings first-class support for the Julia programming language. This extension offers numerous features that enhance the overall development experience for Julia users, including syntax highlighting, code completion, snippets, integrated REPL, linting, and debugging capabilities. Additionally, it

provides an environment in which developers can easily manage Julia packages and execute Julia code within the editor.

By utilizing the Julia extension for Visual Studio Code, developers can streamline their workflow and take full advantage of the powerful features offered by both the Julia language and the VSCode editor. This combination provides an optimal environment for creating, testing, and refining Julia applications, enabling developers to be more productive and efficient in their programming endeavors.

4.3 Pluto – Reactive Notebook

Pluto is an innovative, open-source notebook environment designed specifically for the Julia programming language. It provides an interactive and user-friendly platform for writing, executing, and visualizing Julia code, making it an excellent tool for exploring, prototyping, and sharing ideas [14].

With their reactive nature, Pluto notebooks provide a unique live-coding experience. Upon modification of a code cell, the notebook automatically updates and re-evaluates any dependent cells to ensure the output aligns with the latest changes. This reactive process promotes rapid iteration and exploration, fostering a deeper understanding of the code and its underlying concepts.

Pluto notebooks are highly useful for various tasks, including:

1. Data exploration and analysis: Importing, processing, and visualizing data is seamless with Julia's powerful libraries. This makes Pluto an excellent tool for comprehensive data analysis and exploration.
2. Experimentation and prototyping: The interactive nature of Pluto notebooks enables quick testing of ideas, experimentation with diverse approaches, and iteration on code, thereby accelerating the development process.
3. Educational purposes: The live-coding environment of Pluto is well-suited for educational settings, promoting a hands-on approach to understanding programming concepts and algorithms.
4. Documentation and collaboration: Pluto notebooks, being easily shareable, provide a convenient medium for project collaboration or presentation of research findings, complete with executable code and visualizations.

In conclusion, Pluto notebooks offer an engaging and intuitive approach to working with Julia code, fostering creativity, productivity, and collaboration. By leveraging Pluto's unique features and the power of the Julia language, data exploration, idea testing, and sharing of work can be achieved more effectively.

4.4 Wireshark – Network Packet Analyzer

Wireshark, a free and open-source packet analyzer, serves as a valuable tool for network troubleshooting and analysis. It can capture packet data directly "from the wire" during an active network connection and store it in a file [15]. As demonstrated in Figure 4.1, there is a ASTERIX dissector integrated into the Wireshark package, further enriching its capabilities.

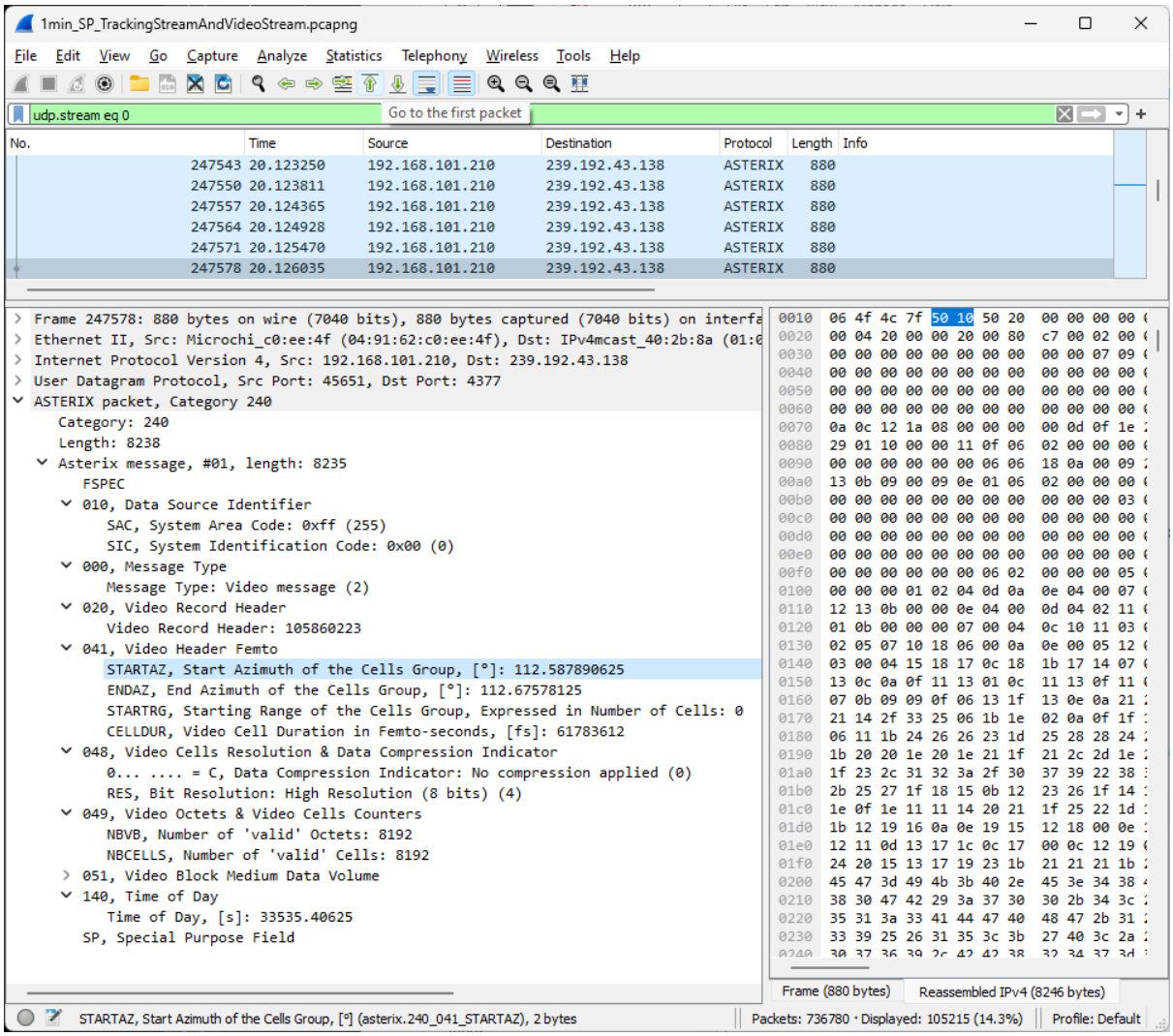


Figure 4.1: A detailed screenshot from Wireshark featuring built in ASTERIX decoder activated, showcasing the raw hexadecimal values alongside the interpreted information within a Category 240 packet.

4.5 Colasoft Packet Player – Network Packet Player

Colasoft Packet Player is a versatile packet replay tool that enables users to open, and playback network packet trace files captured by various sniffer software, including Wireshark. In addition to supporting the original interval when transmitting packet files, the software also allows users to loop the transmission [16]. As illustrated in Figure 4.2, this application streamlines the reproduction of network packets and enables comprehensive testing capabilities using pre-existing data.

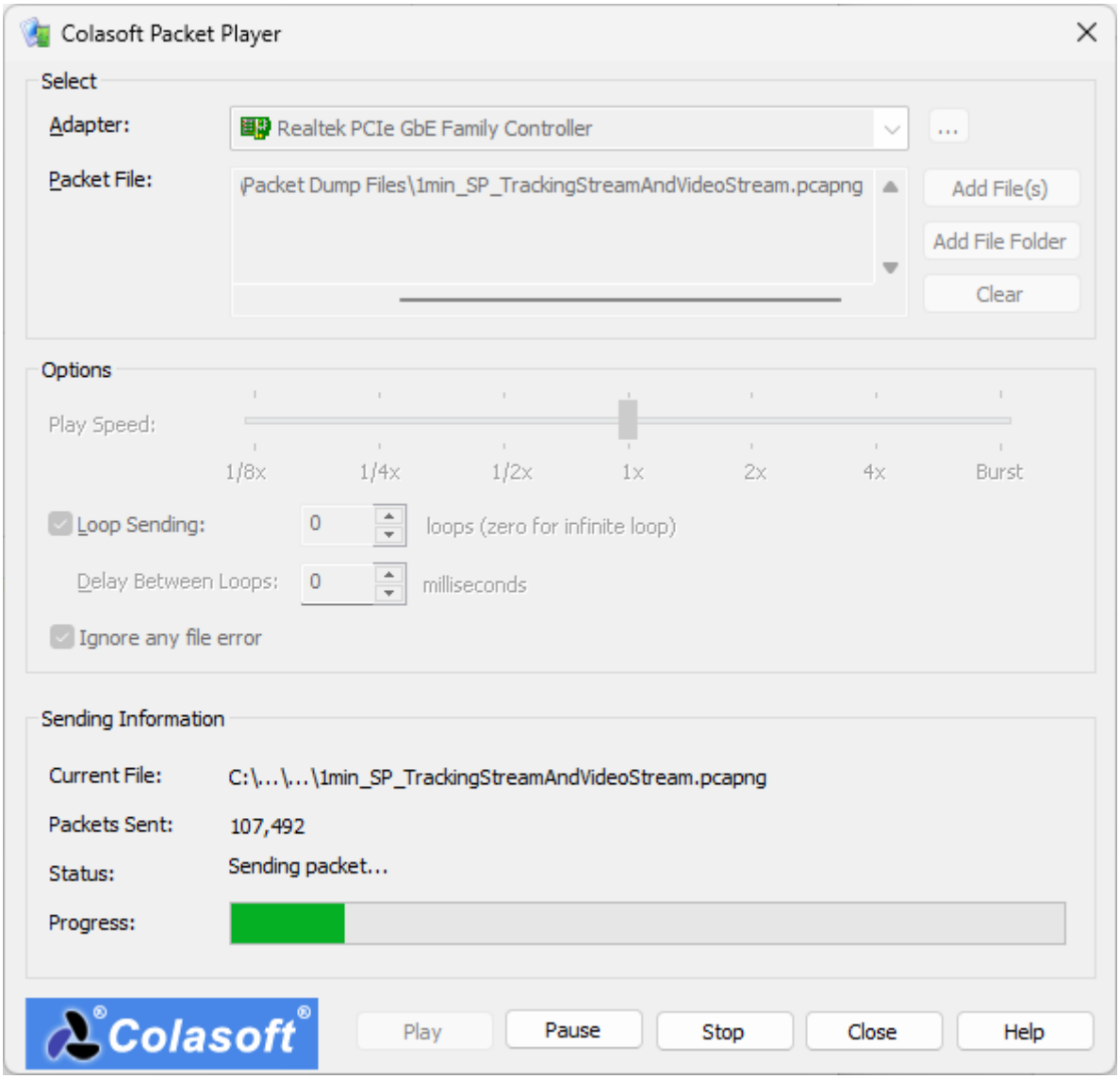


Figure 4.2: Screenshot of Colasoft Packet Player used to open captured packet trace files from Wireshark and play them back in the network.

5 Requirements and Design

This chapter provides a detailed account of the requirements for the application, as well as an explanation of the design decisions implemented to successfully achieve the objectives of the task at hand. Figure 5.1 shows a simplified sketch of the data flow pertaining to several key requirements.

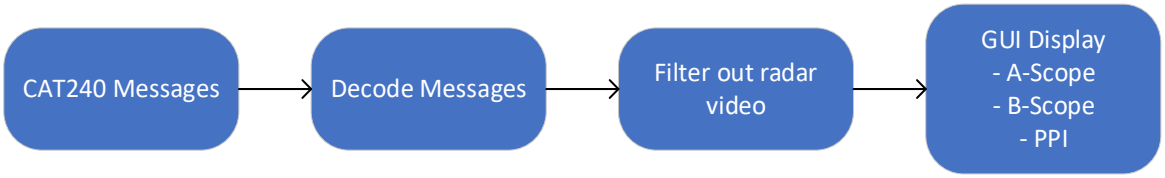


Figure 5.1: A simple sketch of data flow pertaining several key requirements.

5.1 Application Requirements

The subsequent points list the detailed application requirements for this project:

- Read multicast packets (UDP datagram containing binary data) from the radar via Ethernet.
- Decode received binary ASTERIX CAT240 packets comprising header and data [3] [1].
- Present data in a Radar A-scope, comparable to the display of an oscilloscope.
- Present data in a Radar B-scope, a Cartesian diagram that represents a 2-D “top down” view of space where the x-axis represents the azimuth, and the y-axis represents the range.
- Present data in a Radar PPI-display (plan position indicator), which is a scan-converted polar coordinate display with the radar position at the origin.
- The rotating radar available for development and testing purposes in this project has several options for the output video stream, including a factory-configured detailed stream running at 1800 pulses per second (PRF = 1,8kHz in short pulse length), and the radars range is digitized into 8192 samples for each pulse. Then the CAT240 stream represents a set of 1800 messages per second, where each message encodes 8192 samples of video data (approx. 130Mbps). The solution must be capable of processing and displaying the data live, with high performance/ bandwidth.
- Provide information on the range and bearing.
- Configurable radar video input source and decoding settings.
- Video processing, such as gain, threshold and interpolation.
- Possibility to input a static heading for azimuth offset (rotation) of the display.
- The application shall be open-source, which means that all its dependencies should also adhere to open-source licensing.

5.2 Design

Modular software design is a design approach that emphasizes dividing a system into separate, independent modules based on functionality. Each module represents a specific operation or a subset of the system's overall functionality.

In modular design, the entire software system is conceptualized as a set of interacting but self-contained modules. Each module has its own separate and distinct role and is designed to perform that role in an autonomous manner. The modules interact with each other through well-defined interfaces and protocols, without needing to know the internal details of how the other modules work. The main modules in the chosen design are depicted in Figure 5.2.

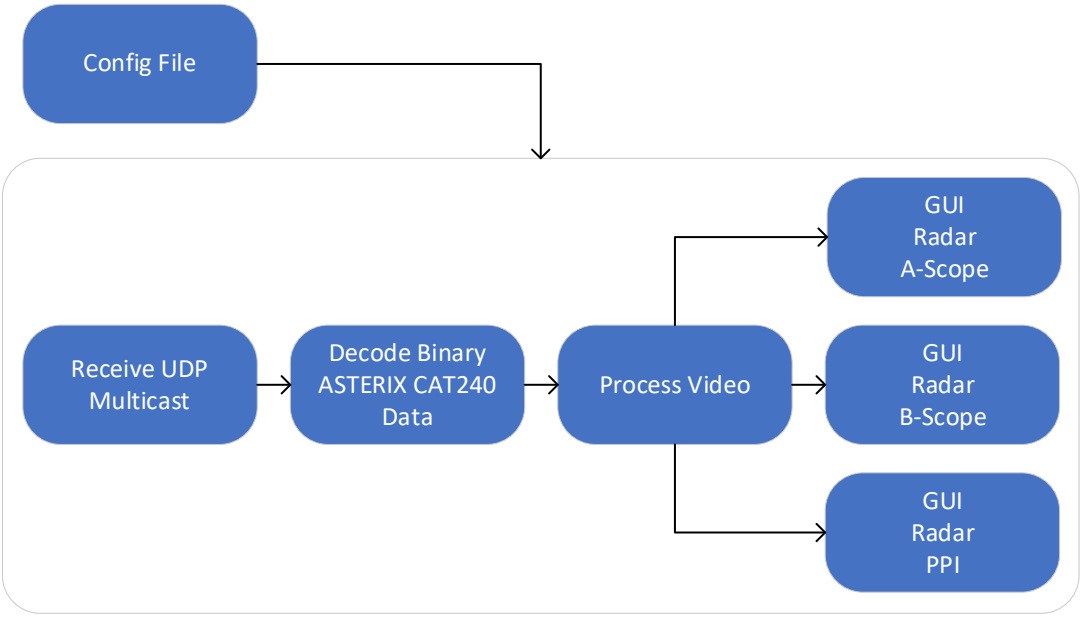


Figure 5.2: Modular software design.

The main program begins by retrieving necessary configurations, such as the multicast IP address and port from the configuration file. Subsequently, a UDP socket is opened to listen for incoming packets. The received data is decoded and assessed for compliance with the CAT240 standard. If the data passes validation, it is processed and displayed in the selected display type. A flow chart of the application is depicted in Figure 5.3. This visual representation outlines the overall application flow structure.

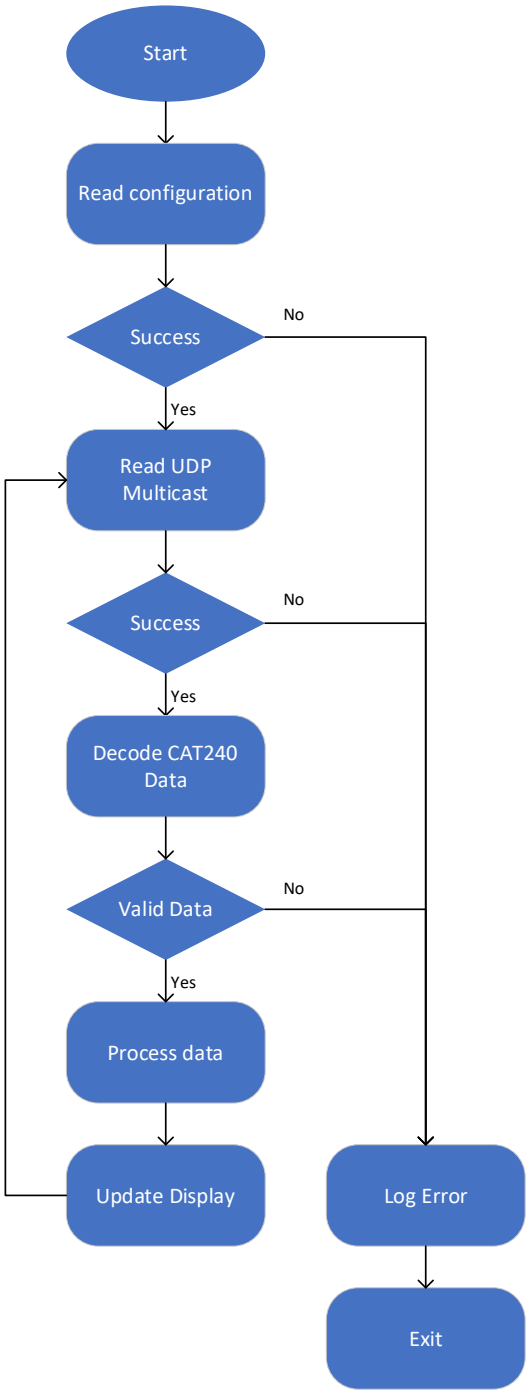


Figure 5.3: Flow chart for the Julia application.

6 Hardware Configuration

A high-resolution navigation radar operating at 28 rpm has been employed, generating 1800 CAT240 packets per second when transmitting in short pulse length. Each packet contains 8192 range cells with a resolution of 8 bits. The hardware has been installed according to the configuration depicted in Figure 6.1.

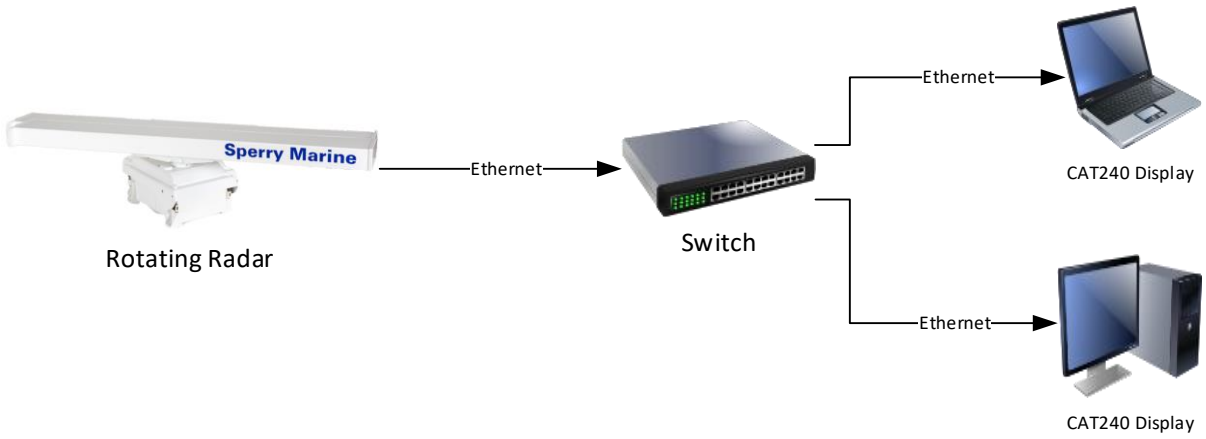


Figure 6.1: Hardware configuration of the system showing two computers connected, each running the CAT240 display application. However, any number of computers can be connected.

The radar is strategically positioned near Bergen port with an unobstructed sea view, at a latitude of 60.39126N and longitude of 5.29285E, as illustrated in Figure 6.2.

Figure 6.2: Picture of the radar utilized to the left, and a map displaying the installation location to the right.

6.1 Data Collection

Real-time CAT240 radar data was captured and stored as .pcapng files using the network protocol analyser, Wireshark. The corresponding files are available from the 'wireshark_packets_dump' directory within the project's GitHub repository. The package dumps facilitate reproducibility and testing of the display application, as though it were connected to a live radar, by utilizing tools like Colasoft Package Player to replay the packet dump files.

6.2 Hardware details

The radar in the test setup employs an 8-foot antenna, and a 25kW X-band transmitter. Table 6.1 outlines the primary specifications for the radar antenna, while Table 6.2 provides detailed information on the transceiver. Additionally, key characteristics of the receiver can be found in Table 6.3 [17]. The information presented has been sourced from the manufacturer's VM Net Ship's manual volume 1 provided by Sperry Marine.

Table 6.1: Radar antenna specification, the test setup employs an 8-foot antenna.

Parameter	Aperture Size		
	1.2m (4ft)	1.8m (6ft)	2.4m (8ft)
Horizontal Beam Width, -3dB (maximum)	2.0°	1.3°	1.0°
Vertical Beam Width, -3dB (nominal)	24°	24°	24°
Sidelobes within 10° of Beam (minimum)	-23dB	-23dB	-23dB
Sidelobes outside 10° of Beam (minimum)	-30dB	-30dB	-30dB
Gain (nominal)	29dB	30dB	31dB
Polarisation	Horizontal	Horizontal	Horizontal
Limiting Relative Wind Speed	100kt	100kt	100kt

Table 6.2: Transmitter specifications, the test setup utilizes a 25kW X-band transmitter.

Parameter	X-band	S-band
Magnetron Frequency	9410MHz ± 30MHz	3050MHz ± 10MHz
Magnetron Peak Power (nominal)	10kW or 25kW	30kW
Pulse Length/PRF (nominal)	0.05µs/1800Hz (Short Pulse) 0.25µs/1800Hz (Medium Pulse) 0.75µs/785Hz (Long Pulse)	
Pulse Generator	Solid-state with Pulse Forming Network driving the Magnetron.	

Table 6.3: Receiver specifications for the receiver used in the test setup.

Parameter	Detail
Type	Logarithmic, with Low Noise Front End (LNFE)
Tuning	AFC/Manual
IF (Intermediate Frequency)	Centred at 60MHz
IF Bandwidth (nominal)	20MHz on short and medium pulses 3MHz on long pulse
Noise Factor (nominal)	5.0dB
Dynamic Range (nominal)	80dB

7 Software Development

This chapter outlines the methodology employed in constructing the software and highlights its essential components.

7.1 Agile Software Development

Agile software development is a contemporary approach to building software that emphasizes adaptability, collaboration, and iterative progress. In contrast to traditional methodologies, which often follow a rigid and linear process, agile development allows for flexibility and rapid response to changing requirements and conditions.

Agile software development is based on several key principles that differentiate it from traditional waterfall methodologies [18]. One such principle is iterative and incremental development. Agile development is characterized by short, iterative cycles called sprints, usually lasting between one to four weeks. During each sprint, a functional subset of the software is developed, tested, and integrated, allowing for rapid feedback and continuous improvement as depicted in Figure 7.1.

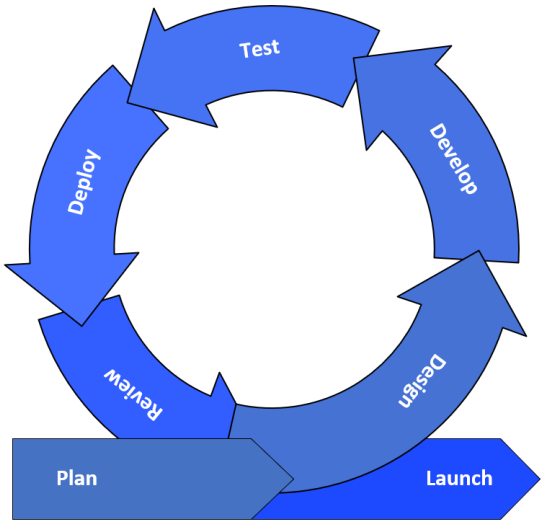


Figure 7.1: Illustration of an agile sprint.

Another important aspect of agile methodologies is adaptability. Agile development prioritizes responsiveness to changing requirements and conditions. As the project progresses, adjustments can be made to the development plan based on new information or shifting priorities, ensuring the final product remains relevant and valuable.

Collaboration is also a cornerstone of agile development. This approach fosters a collaborative environment where cross-functional teams work closely together, sharing knowledge and expertise. This close interaction promotes effective communication, shared understanding, and faster problem-solving. However, this part is not key in a team of one.

Lastly, customer involvement is a crucial component of agile development. In this process, customers or stakeholders are actively involved throughout the project, providing input and feedback at every stage. This ensures that the final product aligns closely with their needs and expectations.

7.2 Data from Radar

Radar video distribution employs the User Datagram Protocol for transmission. Messages are sent within UDP/IP packets to the designated IP address and port specified in the radar settings. Since UDP is not a reliable transport protocol, the receiving implementation should account for the possibility of out-of-order and missing packets.

When sending UDP packets, there might be instances where a single message needs to be split and transmitted across multiple packets to adhere to the Maximum Transmission Unit limits of the network. In cases where jumbo frames are not supported, message frequencies exceeding 1,092 Hz arriving at Windows PCs may be susceptible to corruption, as detailed in [1]. It is crucial to thoroughly consider the implications of this issue during system design.

In order to receive data within the application, the Sockets package [19], which is a part of Julia's standard library, is utilized. This package offers functions to bind to a UDP socket and read data transmitted over a Local Area Network. By binding the socket to host IP 0.0.0.0 as shown in Figure 7.2 the socket will listen on the specified port on all available Ethernet adapters.

```
function receive_udp(chan::Channel, s::UDPSocket; group::IPv4, port::Integer)
…
      ret = Sockets.bind(s, ip"0.0.0.0", port, reuseaddr=true)
…
```

Figure 7.2: Code snippet, listen on all devices.

7.3 Decode CAT240 Data

The decode module is responsible for decoding CAT240 data in accordance with the standard. A messages field specification requires a minimum of one octet and a maximum of two octets. The presence of a second octet is indicated by the field extension indicator, which is represented by the least significant bit in the first octet being set to one. Messages are composed of data items arranged in a sequence determined by the Field Reference Number FRN data items are transmitted only if the corresponding FSPEC bit is set to one. A flow chart illustrating the process can be found in Figure 7.3. Additionally, an overview of the data, including actual example data obtained from a test radar system, is presented in Table 7.1. For more in-depth information it is referred to the standard documentation [1] [3].

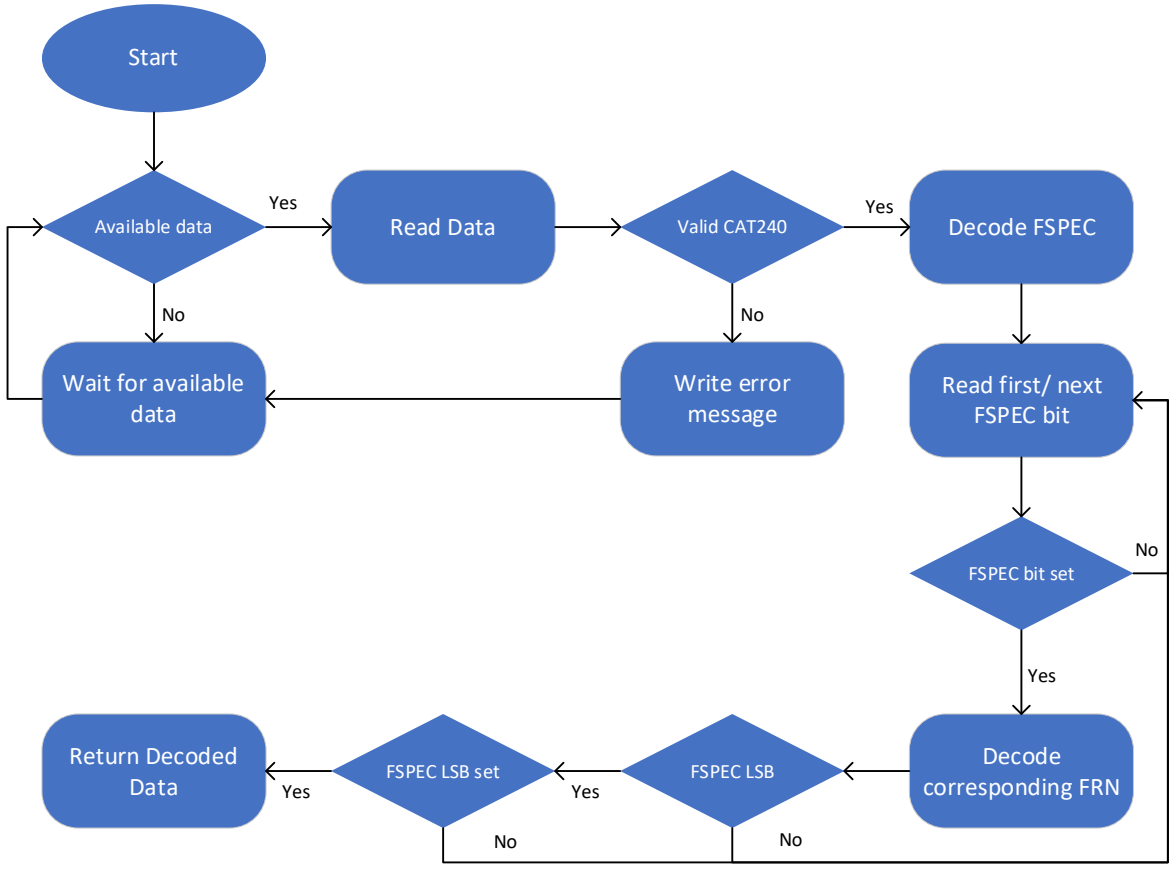


Figure 7.3: Decode data module flow chart.

Table 7.1: Overview of CAT240 video message data items and format, the example data is gathered from the actual test radar system.

FRN	Length in Octets	Data Item	Description	Format	Example	Value (Hex)
-	1	Data Category	CAT	CAT	240	f0
-	2	Length Indicator	Total Length in Octets	0 – (2^{16}-1)	8238	20 2e
-	1 or 2	Field Specification	FSPEC	Binary	11100111 10101010	e7 aa
1	1	Data Source Identifier	SAC	0 – (2^{8}-1)	255	ff
	1		SIC	0 – (2^{8}-1)	0	00
2	1	Message Type	Message Type	001/002	2	02
3	4	Video Record Header	Sequence Identifier	0 – (2^{32}-1)	1465687	00 16 5d 57
4	1	Video Summary	Field Repetition Factor	0 – (2^{8}-1)		
	m		Characters in ASCII	CHAR		
5	2	Video Header Nano	Start Azimuth	0 – (2^{16}-1)		
	2		End Azimuth	0 – (2^{16}-1)		
	4		Start Range	0 – (2^{32}-1)		
	4		Cell Duration [ns]	0 – (2^{32}-1)		
6	13-14	Video Header Femto	Start Azimuth	0 – (2^{16}-1)	0	E7 60
	15-16		End Azimuth	0 – (2^{16}-1)	91	00 58
	17-20		Start Range	0 – (2^{32}-1)	0	00 00 00 00
	21-24		Cell Duration [fs]	0 – (2^{32}-1)	61783612	03 ae be 3c
7	1	Video Cells Resolution and Data Compression Indicator	Compression Indicator	0/1	0	00
	1		Bit Resolution	1,2,3,4,5,6 (1,2,4,8,16, 32 bits)	4 (8bits)	04
8	2	Video Octets and Video Cell Counters	Number of Valid Octets in Video Block	0 – (2^{16}-1)	8192	20 00
	3		Number of Valid Cells in Video Block	0 – (2^{24}-1)	8192	00 20 00
9	1	Video Block Low Data Volume	Repetition Factor (n)	0 – (2^{8}-1)		
	4n		Video Block	4n Octets		
10	1	Video Block Medium Data Volume	Repetition Factor (n)	0 – (2^{8}-1)	128	80
	64n		Video Block	64n Octets	0 0 84 … 10 2 3	00 00 54 … 0a 02 03
11	1	Video Block High Data Volume	Repetition Factor (n)	0 – (2^{8}-1)		
	256n		Video Block	256n Octets		
12	3	Time of Day	1/128 s Elapsed since last Midnight as UTC	0 – (2^{24}-1)	7644707	74 a6 23
13	1+	Reserved Expansion Field	RE	Binary		
14	1	Special Purpose Field	Special Purpose Field Length	0 – (2^{8}-1)	11	0b
	q		Specific Information Beyond the Standard	Binary	-	00 37 9d 84 00 00 00 6d 00 00

7.4 Testing During Development

Testing is a crucial aspect of software development, as it helps verify that the code functions as intended and should be conducted regularly throughout the development process. Identifying and addressing bugs, flaws, and errors early on makes them easier to fix. This kind of testing is referred to as functional testing.

Moreover, it is imperative to confirm that existing code continues to work correctly after new modifications are introduced. This form of testing is known as regression testing.

Throughout the rest of this document, the term "testing" will encompass both functional and regression testing. There are two primary approaches to conducting tests: manual and automated.

7.4.1 Manual Testing

Manual testing involves interacting with the developed software, a process referred to as exploratory testing. Developers should frequently engage in this activity, as it is the sole means of confirming that the software behaves as anticipated. It is also crucial for the end-user or customer to test the software using real-life scenarios, as their usage patterns may significantly differ from those of the developer.

7.4.2 Automated Testing

Manual testing can be labour-intensive and may yield inconsistent results due to human error. Consequently, scripted tests are employed to examine the system's underlying functionality more reliably. These tests produce consistent results, enabling precise identification of the failing system component. Automatic tests generally fall into two categories: unit tests and integration tests [20].

Unit tests evaluate whether a single code component functions as expected. For instance, a unit test may verify that a specific function returns the correct value or carries out the desired action. Unit tests typically belong to the "white-box testing" group, in which the code's internal workings are known to the test writer. In contrast, integration tests evaluate the compatibility and functionality of various system components. For example, an integration test might involve reading data through an Ethernet adapter and verifying that the correct actions are carried out. Integration tests can be classified as either white-box or black-box tests, with the latter indicating that the test writer is not familiar with the code's internal workings.

This project utilizes both unit tests and integration tests to ensure comprehensive and reliable evaluation of the system. A snippet demonstrating some of the automated test results is shown in Figure 7.4.

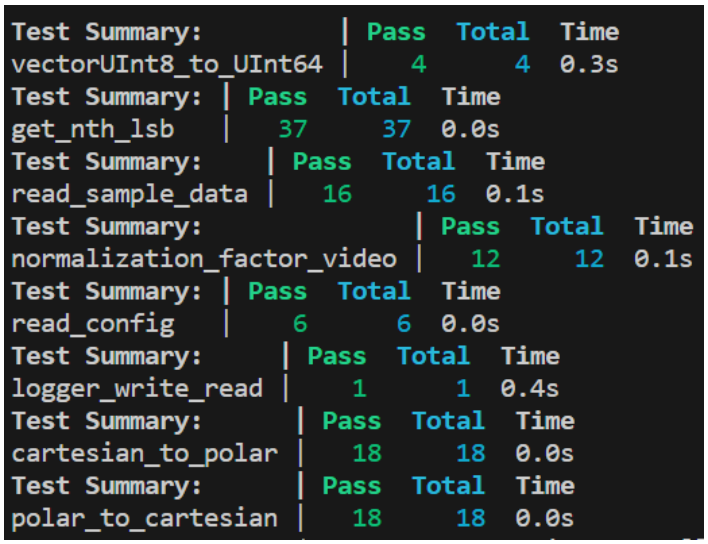


Figure 7.4: The following snippet demonstrates the automated test results generated by running the 'runtests.jl ' file, accessible within the 'test' directory of the project's GitHub repository.

7.5 Logging

File-based logging is implemented using a package called LoggingExtras, which serves as a convenient wrapper around the Logging module included in Julia's standard library [21]. This approach allows for tracking the history and progress of computations through a log of events. Events are generated by incorporating logging statements within the source code. Example of such statements is showcased in Figure 7.5, and the corresponding output in the log file is depicted in Figure 7.6.

```
@warn("Warning: Potential issue detected")
@info("Information: Routine operation")
@error("Error: Critical failure occurred")
@debug("Debug: Diagnostic information")
```

Figure 7.5: Example of various logging levels to categorize and filter messages based on their severity or importance during execution.

```
┌ Warning: 2023-05-09 10:01:29 Warning: Potential issue detected
└ @ Main d:\Source\USN\RadarDisplay\examples\logger\logger_example.jl:3
┌ Info: 2023-05-09 10:01:29 Information: Routine operation
└ @ Main d:\Source\USN\RadarDisplay\examples\logger\logger_example.jl:4
┌ Error: 2023-05-09 10:01:29 Error: Critical failure occurred
└ @ Main d:\Source\USN\RadarDisplay\examples\logger\logger_example.jl:5
┌ Debug: 2023-05-09 10:01:30 Debug: Diagnostic information
└ @ Main d:\Source\USN\RadarDisplay\examples\logger\logger_example.jl:6
```

Figure 7.6: Example log file output.

These logging statements enable the classification of events according to their severity or importance, including warnings, informational messages, errors, and debug messages. Additionally, it contains information about when and where in the code it was executed. This

structured approach simplifies monitoring and debugging throughout the development process. Furthermore, the logs can prove to be an invaluable resource post-deployment, assisting in monitoring the program's status and identifying any potential issues.

7.6 Naming Convention and Style Guide

The official Julia style guide found here [22], offers a comprehensive set of recommendations to help write clean, readable, and consistent code. Although these guidelines may not be suitable for every situation, they generally serve as helpful suggestions and an effort to follow the guidelines has been made. Naming conventions have also been chosen to align with those used in Julia's base library, ensuring intuitive code for other Julia users.

7.7 Source Control

Source control represents a cornerstone in modern software development practice, as it preserves the historical record of all changes made to the project files. This functionality allows for the rollback to a previous version of the code, should errors or issues arise. For this project, Git is utilized as the source control mechanism, operating in conjunction with the cloud-based service, GitHub. Given that packet dump files for this project exceed GitHub's file size limit of 100.00 MB, the Git Large File Storage, also known as Git LFS [23], extension is employed to manage these larger files. This solution allows the GitHub repository to contain pointers to these large files, effectively managing their storage and accessibility.

7.8 Performance

BenchmarkTools.jl [24] is a robust Julia package specifically designed for assessing the performance of Julia code. Its primary offering is the `@benchmark` macro, which runs a code block multiple times and collects detailed statistics about its execution time. This package has been instrumental in assessing and optimizing the performance of various functions within the application. Figure 7.7 and Figure 7.8 present an example of the code and its resulting output. In this context, 'Cat240' is a module and 'Cat240Message' is a struct. By inputting binary data, the constructor validates if the data conforms to the CAT240 standard and, if valid, populates the struct fields accordingly.

```
@benchmark cat240message = Cat240.Cat240Message($binary_data)
```

Figure 7.7: This code snippet benchmarks the performance of the process for validating and decoding binary data adhering to the CAT240 standard.

```
BenchmarkTools.Trial: 10000 samples with 87 evaluations.
 Range (min … max):  740.230 ns … 52.876 µs  ┊ GC (min … max):  0.00% …  0.00%
 Time  (median):       1.038 µs              ┊ GC (median):     0.00%
 Time  (mean ± σ):     1.342 µs ±  1.863 µs  ┊ GC (mean ± σ):  14.86% ± 11.93%

  740 ns          Histogram: log(frequency) by time        10.6 µs <

 Memory estimate: 8.97 KiB, allocs estimate: 13.
```

Figure 7.8: Result of a performance benchmark trial.

To evaluate the computational demands of the developed PPI application, it was tested on a standard modern computer. Figure 7.9 provides a snapshot of the application's performance as monitored through the task manager.

Name	Status	18% CPU	25% Memory	0% Disk	16% Network	4% GPU
Julia Programming Language		16.0%	1,064.3 MB	0 MB/s	0 Mbps	4.2%
ASTERIX Category 240 Radar Display						

Figure 7.9: A snapshot from the task manager, showcasing the PPI application's performance on a modern computer equipped with an Intel i5-11400 CPU and a GeForce RTX3070 GPU.

7.9 Deployment

To initiate the application, it's a prerequisite to have Julia installed on your computer, with a minimum required version of 1.8.5. The 'Project.toml' file lists the necessary package dependencies, which can be conveniently installed using Julia's built-in package manager with the 'instantiate' command [25]. Additionally, valid CAT240 data must be accessible on the network, located at the IP address and port detailed in the 'config.JSON' file. Once these conditions are met, execute any of the three files named after the display type available in the 'display' folder within the GitHub repository.

7.10 PPI

In the radar scan conversion process, the polar coordinates of radar data, represented as range-azimuth, are transformed into Cartesian coordinates, represented as x-y. The geometric transformation can be expressed as Equation (4.1) and (4.2).

$$x_i = r_i \sin\theta_i \tag{4.1}$$

$$y_i = r_i \cos\theta_i \tag{4.2}$$

There is not a direct relationship between radar samples and the output image using Cartesian coordinates, two-dimensional interpolations are required to prevent unwanted visual artifacts, such as the Moiré patterns illustrated in Figure 7.10.

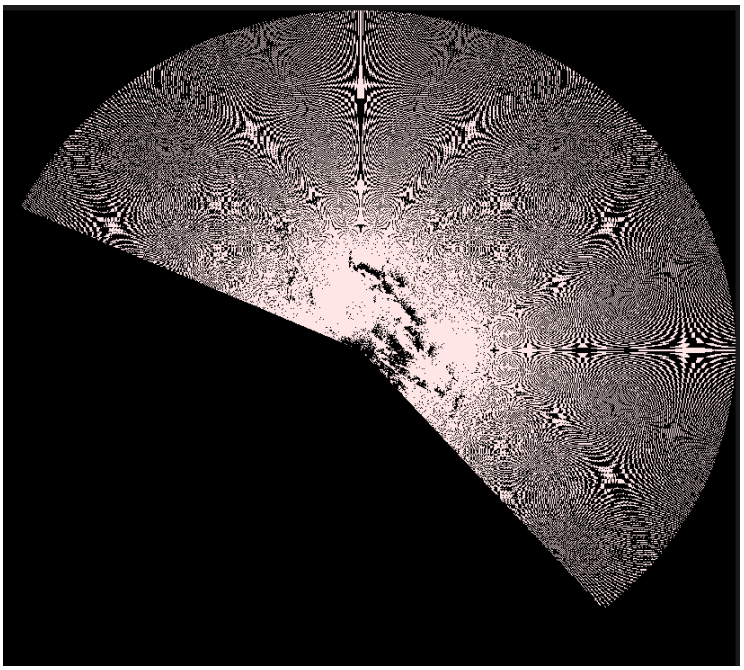


Figure 7.10: PPI display without interpolations illustrating the Moiré pattern.

8 CAT240 Radar Display Application

In this chapter, the developed application will be presented, highlighting the user configuration, various radar displays and processing functions accessible to the user during operation.

8.1 General

The application has been developed using the Julia programming language. The complete source code is accessible in the dedicated GitHub repository, which can be found at the following link: GitHub - USN231216/RadarDisplay.

Visualizations in the application are created using Makie, with GLMakie as the backend. GLMakie leverages OpenGL to render graphics efficiently, ensuring smooth performance. To take full advantage of these capabilities, users should have an OpenGL-enabled graphics card that supports OpenGL version 3.3 or higher.

8.2 Configuration

The application features a JSON configuration file, which enables users to conveniently modify connection settings and other essential parameters using any text editor. Figure 8.1 displays the available parameters within the configuration file.

```
{
    "Radar": {
        "Network": {
            "ip_address": "192.168.1.10",
            "port_number": 8080
        },
        "Video": {
            "azimuth_offset": 12.3
        }
    }
}
```

Figure 8.1: Parameters in the JSON configuration file. Syntax highlighting is provided when using e.g., Visual Studio Code as text editor.

8.3 A-scope

The A-scope display offers real-time visualization of video blocks found in CAT240 messages, akin to an oscilloscope presentation. Each frame comprises a group of video cells that correspond to a video radial, as demonstrated in Figure 8.2. Utilizing the sliders positioned beneath the plot, users can adjust the gain, effectively multiplying all cell amplitudes by the selected value, and modify the offset, which directly adds to all cell amplitudes. Additionally, the start azimuth and end azimuth sliders enable users to filter and display only specific azimuth angles for a more tailored viewing experience.

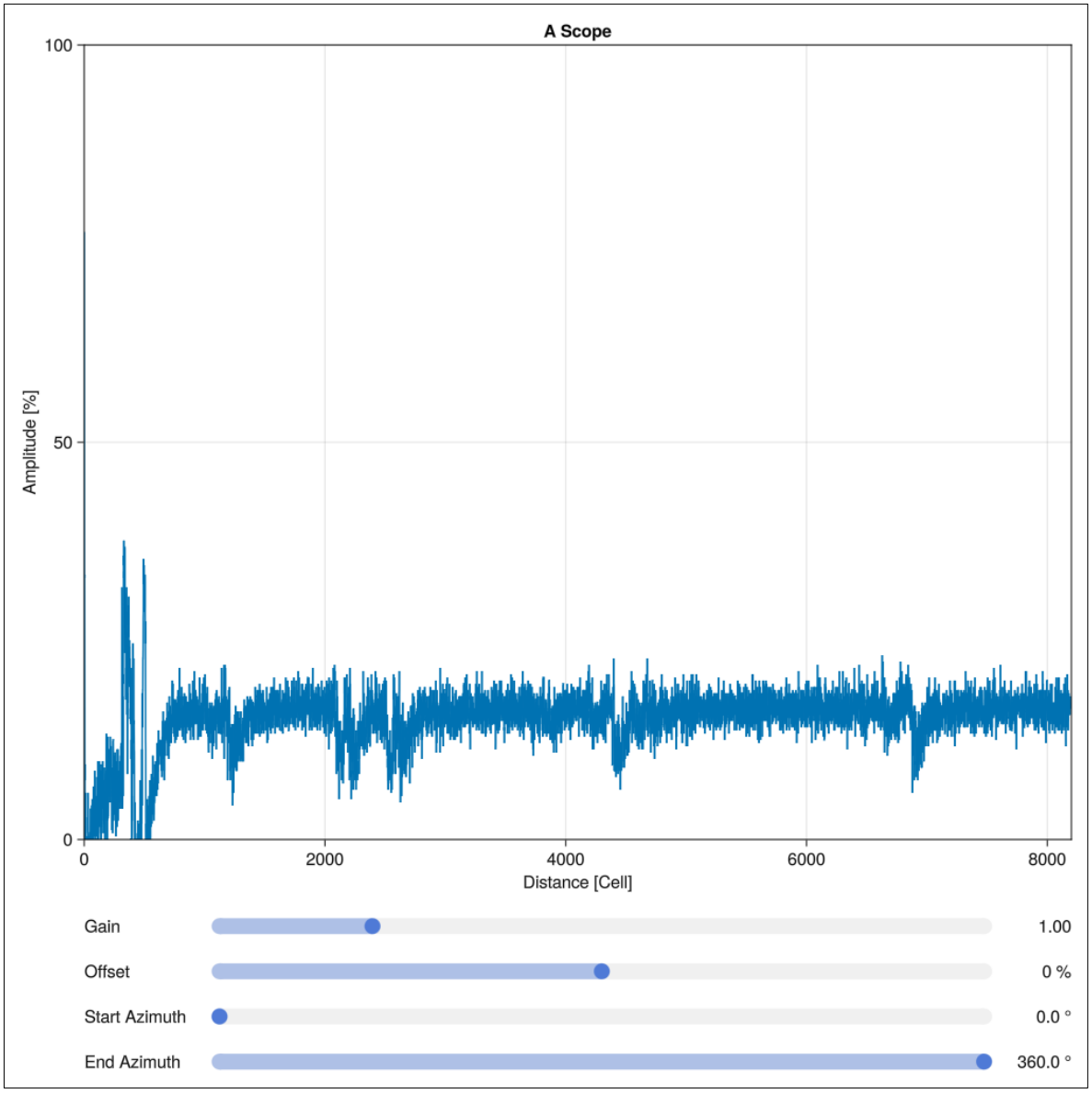


Figure 8.2: A-scope showing a group of video cells corresponding to a video radial.

8.4 B-scope

The B-Scope presents an image similar to a Cartesian plot, offering a two-dimensional "top-down" view of the area. In this representation, the horizontal axis corresponds to the azimuth measurement, while the vertical axis depicts the range measurement. Signals appear as distinct bright spots, with their intensity indicated by the colormap on the right side of Figure 8.3. The functionality of the interactive sliders for floor level and ceiling level are demonstrated in Figure 8.4.

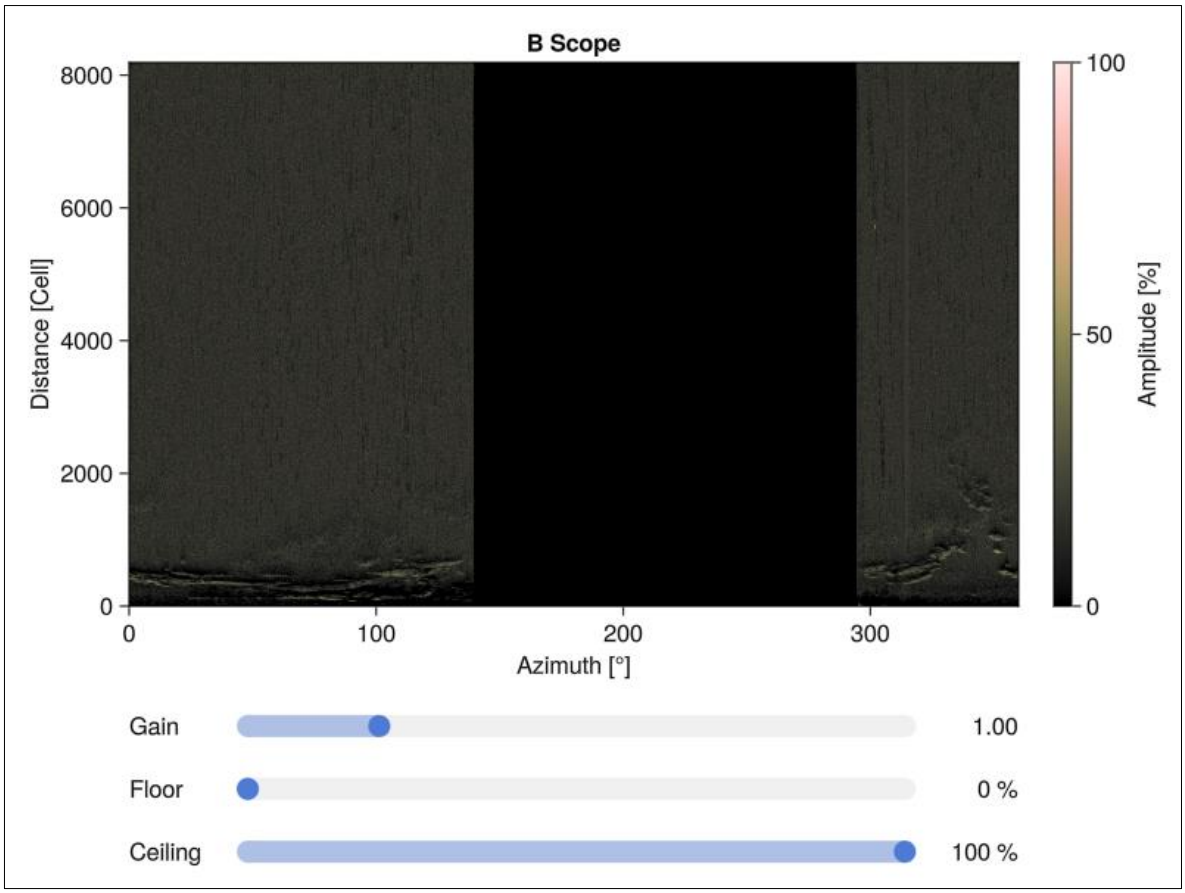


Figure 8.3: B-scope display of the application, with adjustable sliders for interactive parameter settings.

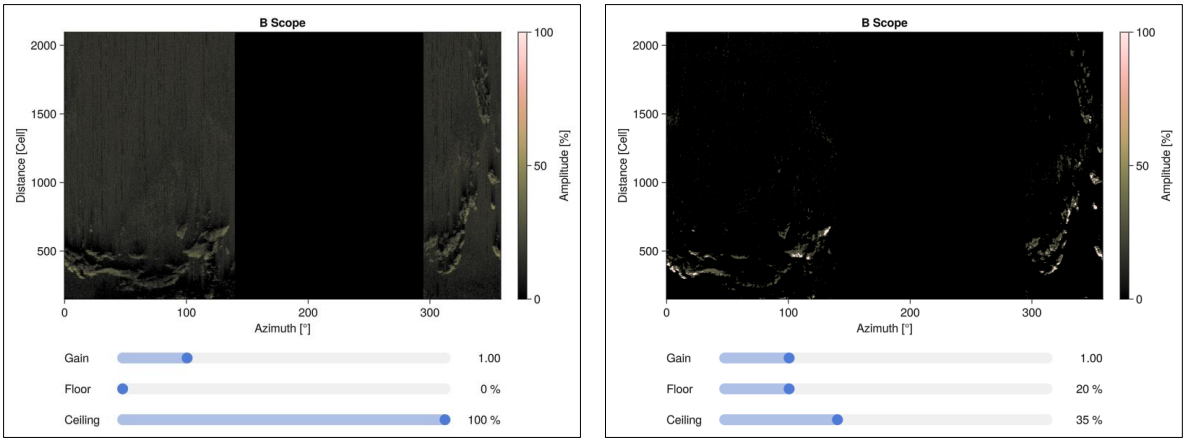


Figure 8.4: B-scope display of the application with zoom. On the left side no floor or ceiling filter are active. While on the right-side floor and ceiling filters are active as indicated by the slider positions.

8.5 PPI

The live scan converted image can be viewed by employing the Plan Position Indicator, or PPI, script. The radar's location is at the centre, marked by 0 on both the X and Y axes.

Users can adjust various parameters using corresponding sliders, such as gain, which amplifies each pixel according to the specified parameter. The floor filter eliminates pixels with an intensity value below the slider setpoint, while the ceiling filter adjusts pixel values above the setpoint to the maximum value, represented as 100% or white on the colour bar located to the right of Figure 8.5. Moreover, an azimuth offset slider is available to rotate the entire image by applying a rotation matrix multiplication as illustrated in Figure 8.8. The floor and ceiling filters are showcased in Figure 8.6 and Figure 8.7. Figure 8.9 presents the radar video manually superimposed on a local area map for a more contextual view. Furthermore, specifics derived from the decoded CAT240 data are documented in a log file, a sample is presented in Appendix A.

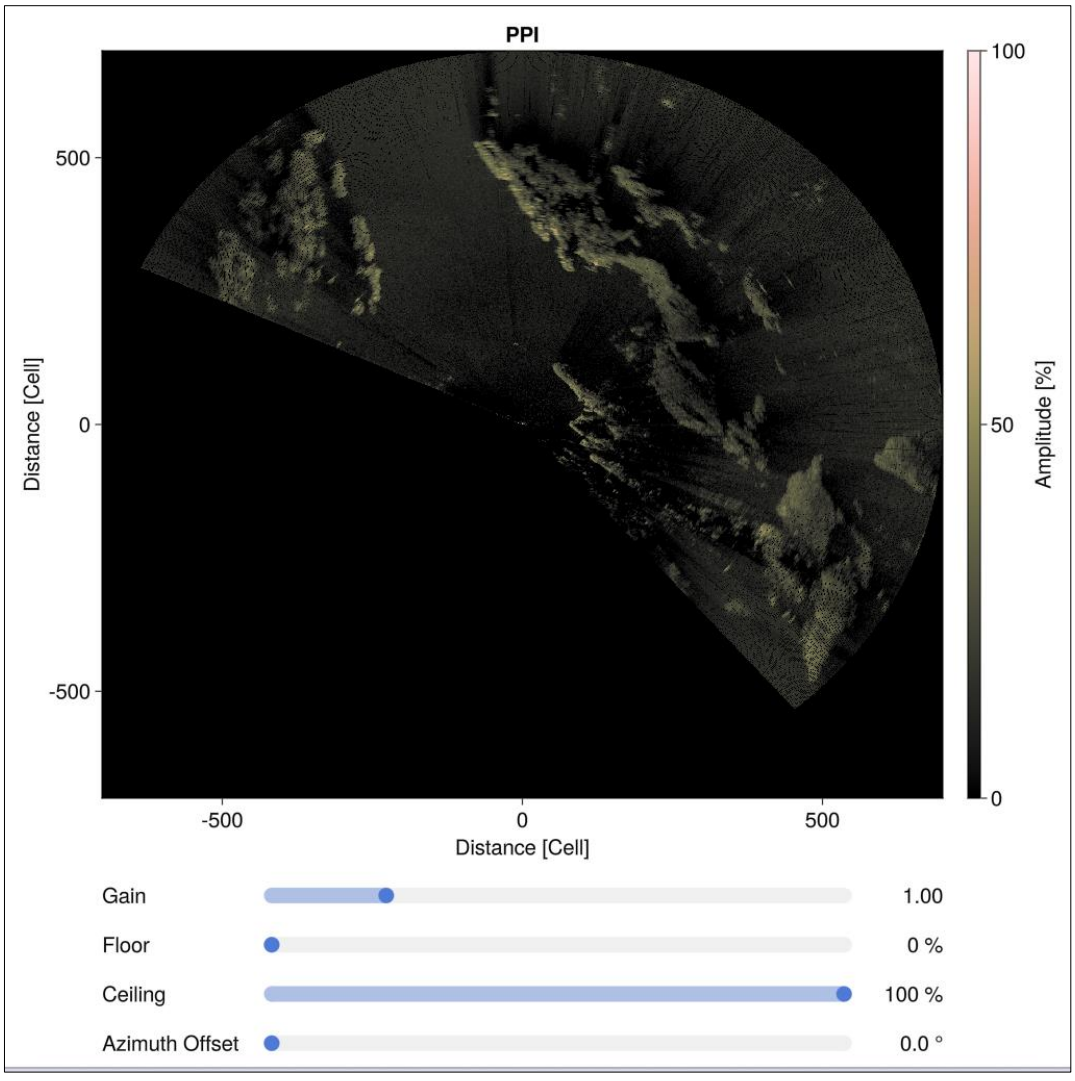


Figure 8.5: PPI display with sliders below and a colorbar to the right.

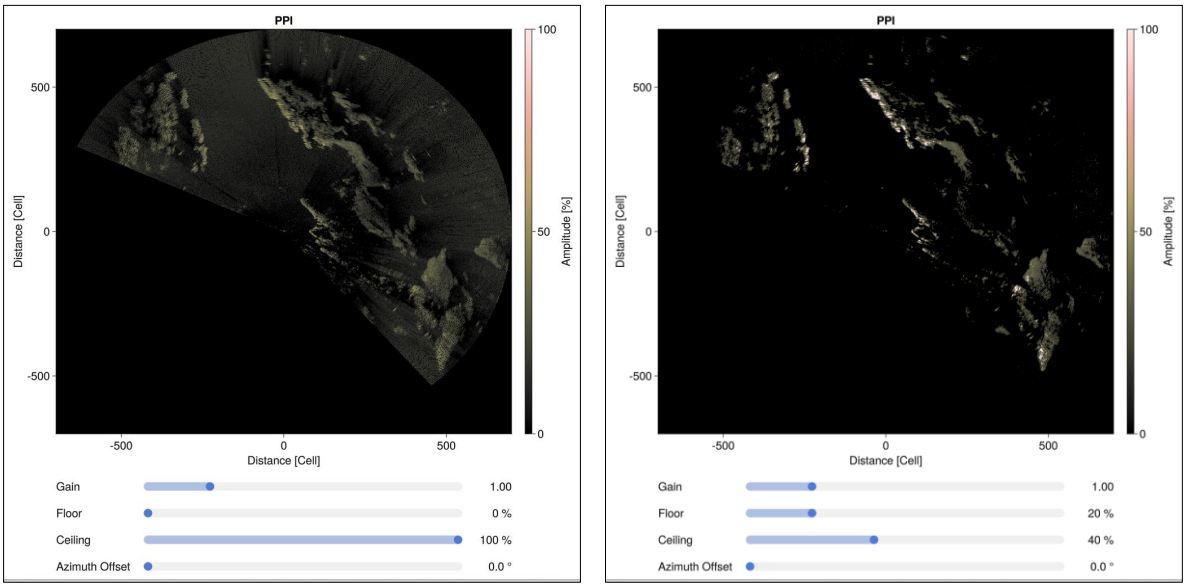


Figure 8.6: The left PPI displays the data with no floor or ceiling filters applied. The right image, on the other hand, shows the result after the application of floor and ceiling filters, as determined by the respective slider positions.

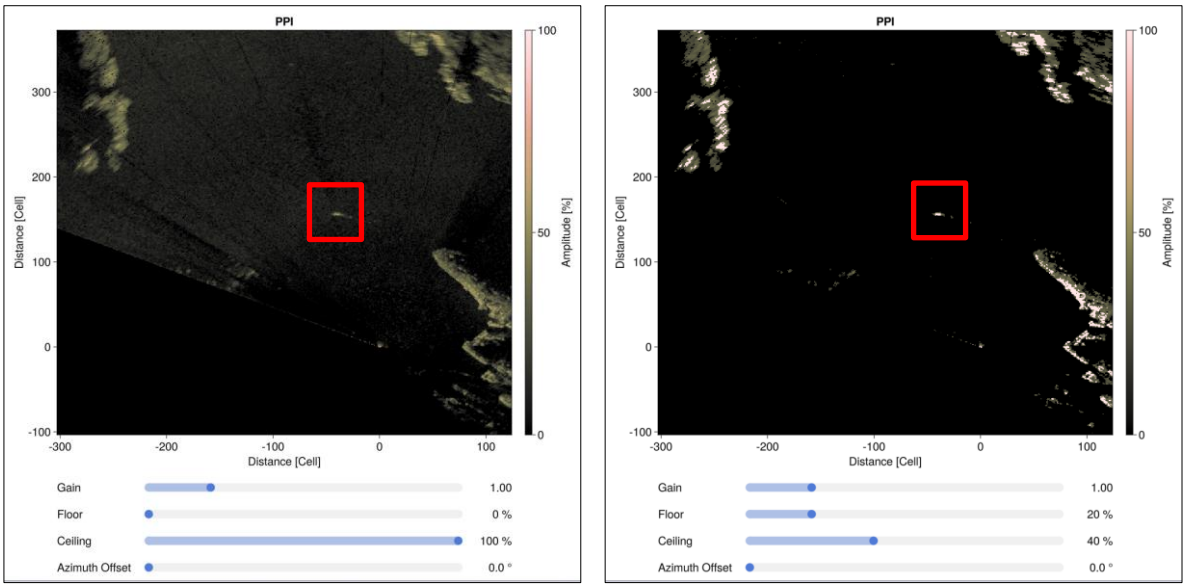


Figure 8.7: This zoomed-in image features a smaller vessel, with the same filter settings as applied in Figure 8.6. To highlight the vessel's location, a red square has been manually superimposed.

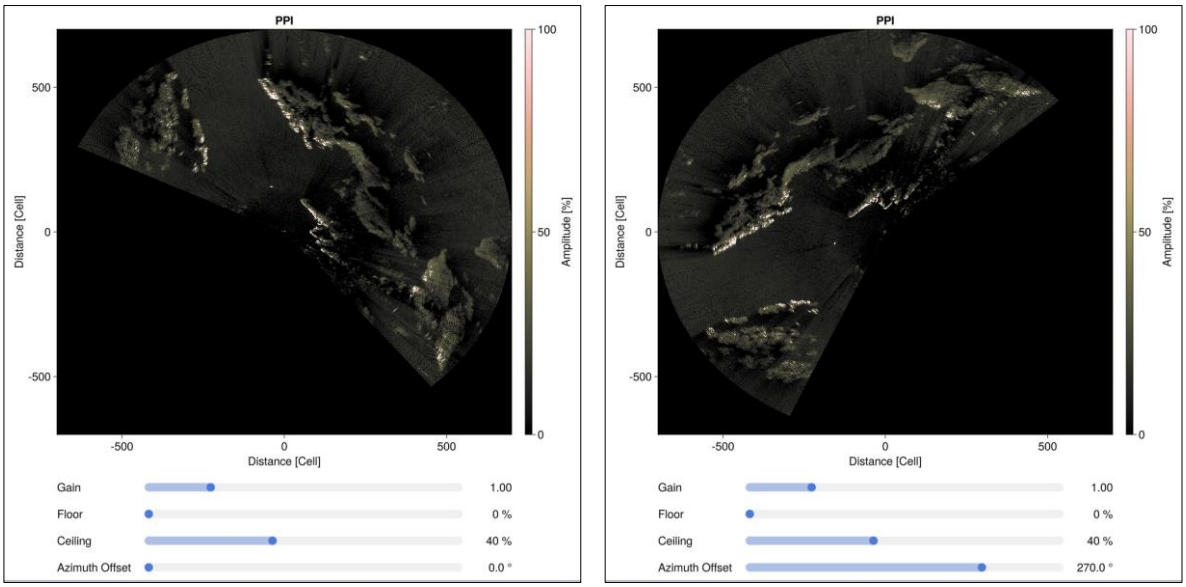


Figure 8.8: The left image is displayed without any azimuth offset, while the right image demonstrates the application of a 90-degree counter-clockwise azimuth offset.

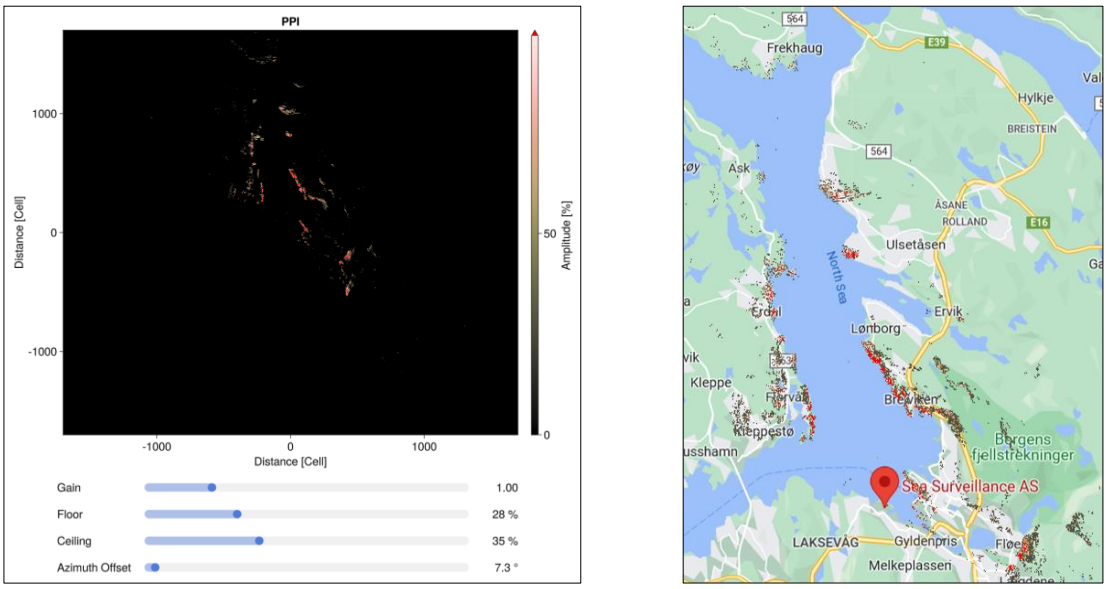


Figure 8.9: Radar video manually superimposed on a map of the local area, with adjustments made to the video colors for enhanced visibility.

9 Discussion

In this project, the main objective was to create a free open-source tool with a user-friendly interface to read and display radar video data in the CAT240 format, which arrives binary via Ethernet UDP multicast. Due to the high amount of data produced by high-resolution radar, three different high-level programming languages were considered. MATLAB was ruled out early on due to its licensing cost, and Python emerged as the main option due to its available libraries. However, the customer's preference for a single-language implementation and the performance advantages of Julia, mentioned in chapter 4.1, led to the selection of Julia over Python.

To develop the application, Agile methodologies were used, prioritizing sprint cycles and continuous user feedback. Although Kanban was initially adopted for organizational purposes, it was found to provide little value for a solo developer and was used sparingly. Regular reflection on ways to enhance effectiveness, following the Agile manifesto's 12th principle, led to the decision to discard Kanban. A Lean's Gantt project plan was used to track overall progress and remaining tasks.

The end result of the project is a fully functional application that meets the primary goal set by the requirements and user feedback. The application's main tasks involve reading and decoding CAT240 data, processing it, and displaying it in various forms of standard radar displays. User input generates instant visual feedback on a live radar video display. Additionally, as an open-source software package, the application is designed to be a foundation for future extensions. The codebase is accessible on GitHub, allowing for future modifications or improvements.

10 Further Work

While the current application has successfully addressed the specified requirements and demonstrated its utility for CAT240 radar testing and analysis, there remain opportunities for further enhancements and developments. This chapter outlines potential avenues for future work, which could improve the application's performance, expand its features, and maximize its impact on the field.

- **Performance Optimization:** Further optimization of the application's performance could lead to faster processing times and a more responsive user interface, ensuring a smoother user experience. Techniques such as parallel processing, caching, and algorithmic improvements can be explored to achieve these enhancements.
- **Improved GUI:** The user interface could be further enhanced by incorporating additional visual elements such as range rings and real-time cursor positioning. Moreover, transitioning to vector graphics for rendering could eliminate the need for interpolation and facilitate a lossless visualization of each video cell in the digitized CAT240 video, thereby offering a more precise and detailed view.
- **Additional Data Formats:** To increase the application's versatility, support for additional data formats could be integrated, allowing users to work with various radar systems and data types seamlessly. This expansion would broaden the application's appeal and make it more widely applicable in the radar community.
- **Advanced Analysis Tools:** Incorporating advanced analysis tools and algorithms into the application could provide users with deeper insights into the radar data. Features such as automatic target detection, tracking, and classification could be developed to enable more sophisticated data analysis.
- **User Customization:** Allowing users to customize the application's interface and settings to suit their preferences could improve its usability and overall user satisfaction. Implementing features such as configurable display layouts, colour schemes, and data visualization options would cater to diverse user needs.
- **Integration with External Tools:** To further increase the application's utility, integration with external tools and platforms used in the radar domain could be considered. This would enable seamless data exchange and collaboration among users and facilitate a more streamlined workflow.
- **User Feedback and Iterative Improvement:** Continuous collection of user feedback and subsequent refinements of the application based on their suggestions would ensure that it remains relevant and effective in meeting users' evolving needs.

By exploring these avenues for further work, the CAT240 display application can continue to evolve, providing even greater value to the radar community and fostering ongoing innovation in the field.

11 Conclusion

In this master's thesis project, a CAT240 display application was created utilizing modern software development methodologies and tools. The project was carried out in close cooperation with the external partner, Sea Surveillance AS.

The developed application addresses a difficulty encountered by certain CAT240 radar users, who often struggle to examine and assess data quality without specialized paid tools. To mitigate this issue, the application offers a user-friendly interface that enables users to visualize data through familiar radar displays, such as A-scope, B-scope, and PPI. Furthermore, the application incorporates readily accessible processing functions via sliders, delivering instantaneous feedback for enhanced user experience.

The application is designed as an open-source project, promoting adaptability and extensibility for other developers to modify and enhance its features according to their needs.

The delivered application effectively addresses the stated requirements, and hopefully it will serve as a valuable tool for CAT240 radar testing and analysis in its current state. Moreover, it lays a foundation for future enhancements and developments.

The open-source source code for the application can be found at the following link:

GitHub - USN231216/RadarDisplay

References

[1] EUROCONTROL, "Specification for Surveillance Data Exchange Part I," [Online]. Available: https://www.eurocontrol.int/publication/eurocontrol-specification-surveillance-data-exchange-part-i.

[2] ISO/IEC 7498-1:1994, "Information Processing Systems - OSI Reference Model - The Basic Model," 2000.

[3] EUROCONTROL, "CAT240 - EUROCONTROL Specification for Surveillance Data Exchange ASTERIX," [Online]. Available: https://www.eurocontrol.int/publication/cat240-eurocontrol-specification-surveillance-data-exchange-asterix.

[4] A. G. Bole and W. O. Dineley, Radar and ARPA Manual, Newnes, 2016.

[5] M. O. Kolawole, Radar Systems, Peak Detection and Tracking, Newnes, 2002.

[6] G. S. Rao, Microwave and Radar Engineering, Pearson India, 2014.

[7] C. Wolff, "Radartutorial.eu," [Online]. Available: https://www.radartutorial.eu/index.en.html.

[8] A. Edelman, J. Bezanson, S. Karpinski and V. B. Shah, "Julia: A fresh approach to numerical computing," 2017. [Online]. Available: https://doi.org/10.1137/141000671.

[9] Julialang.org, "Julia Micro-Benchmarks," Julialang.org, [Online]. Available: https://julialang.org/benchmarks/.

[10] Numpy, "NumPy is the fundamental package for scientific computing with Python.," [Online]. Available: https://github.com/numpy/numpy.

[11] S. Danisch and J. Krumbiegel, "Makie.jl: Flexible high-performance data visualization for Julia," 2021. [Online]. Available: https://doi.org/10.21105/joss.03349.

[12] JuliaHub, "Julia Case Studies," [Online]. Available: https://juliahub.com/case-studies/.

[13] Microsoft, "Visual Studio Code," [Online]. Available: https://code.visualstudio.com/.

[14] "Pluto: Simple, reactive programming environment for Julia," [Online]. Available: https://plutojl.org/.

[15] "WireShark: Network Analyzer," [Online]. Available: www.wireshark.org.

[16] "Colasoft Packet Player," [Online]. Available: https://www.colasoft.com/packet_player/.

[17] Sperry Marine, *VM Net Ship's Manual,* 2020.

[18] A. Cockburn, Agile Software Development: The Cooperative Game, Second Edition, Addison-Wesley Professional, 2006.

[19] Julia Documentation, "Julia Standard Library - Sockets," [Online]. Available: https://docs.julialang.org/en/v1/stdlib/Sockets/.

[20] Testim, "Unit Test vs. Integration Test: Tell Them Apart and Use Both.," [Online]. Available: https://www.testim.io/blog/unit-test-vs-integration-test/.

[21] Julia Documentation, "Julia Standard Library - Logging," [Online]. Available: https://docs.julialang.org/en/v1/stdlib/Logging/.

[22] Julia Documentation, "Style Guide," [Online]. Available: https://docs.julialang.org/en/v1/manual/style-guide/.

[23] Git Large File Storage, "An open source Git extension for versioning large files," [Online]. Available: https://git-lfs.com/.

[24] BenchmarkTools.jl, "BenchmarkTools Manual," [Online]. Available: https://juliaci.github.io/BenchmarkTools.jl/dev/manual/.

[25] Julia Documentation, "Working with Environment," [Online]. Available: https://pkgdocs.julialang.org/v1/environments/#Using-someone-else's-project.

Appendices

Appendix A Master's Thesis Description

Appendix B Log File Example

University of South-Eastern Norway

Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: Development and Testing of ASTERIX category 240 Radar Display

USN supervisor: Hans-Petter Halvorsen

External partner: Sea Surveillance AS

Task background:

Sea Surveillance AS is a part of Seabrokers Group (https://www.seabrokers.no), based in Bergen, Norway operating since 2003.

A typical marine scanning radar emits pulses radially from a rotating transmit/receive antenna. To display a radar image on a display requires that the original radar video from the sensor is acquired and scan converted, from polar coordinates to rectangular coordinates. An open standard for radar video distribution has emerged in recent years and a number of commercially available radars output the video signal digitally using the ASTERIX category 240 standard. Specialised surveillance data processing software to decode the category 240 messages and scan convert the images are available, however not open source (as far as we know). The motivation to make this software is to have a free open-source tool that can be used for troubleshooting and display of the radar signals. For example if the radar is running with 2000 pulses per second (prf = 2kHz) and the radars range is digitised into 1000 samples for each pulse then the CAT-240 stream represents a set of 2000 messages per second, where each message encodes 1000 samples of video data.

Task description:

Main Task: Develop software to decode and display radar video distribution messages that use the ASTERIX format for transportation of Radar Video data as defined in the ASTERIX category 240 standard [1] and the general ASTERIX standard [2].

A commercial navigation grade marine radar scanner capable of outputting ASTERIX category 240 radar video with a view over Bergen harbor will be made available by Sea Surveillance and should be used for development and testing of the application.

The following sub tasks should be done in this project:

- Explore Marine Radar Applications in general and in context of this project. Common use cases
- Explore ASTERIX category 240 standard in general and in context of this project
- Defining requirements, design and implement a system for monitoring of category 240 radar video. The system needs to be flexible and configurable so it can be used in different scenarios and use cases. Examples:
    - Read and decode CAT-240 packets comprising header and data [1][2]

    - Present data as a Radar A-scope on selected radial(azimuth), comparable to the display of an oscilloscope
    - Present data as Radar B-scope, cartesian diagram. 2-D "top down" representation of space. Where the x-axis represents the azimuth, and the y-axis represents the range.
    - Present data as Radar PPI-display (plan position indicator). Scan converted polar coordinate display with the radar position in the origin. The most used and known radar display.
    - Provide information on the range, bearing
    - Configurable radar video input source, and decoding settings
    - Video processing. E.g. gain, threshold, interpolation
    - Possibility to input a static heading for azimuth offset(rotation) of the display.
- Explore network requirements in context of category 240 radar in general and this project
- Testing, Installation and Deployment of System
- Include necessary scientific aspects, methods, and analysis
- The system needs to be properly documented so it possible for others to use and extend the system after the project is finished

References:

[1] EUROCONTROL, "EUROCONTROL-SPEC-0149-0240: CAT240 - EUROCONTROL Specification for Surveillance Data Exchange ASTERIX," Available: https://www.eurocontrol.int/publication/cat240-eurocontrol-specification-surveillance-data-exchange-asterix.

[2] EUROCONTROL, "EUROCONTROL-SPEC-0149 EUROCONTROL Specification for Surveillance Data Exchange Part I," Available: https://www.eurocontrol.int/publication/eurocontrol-specification-surveillance-data-exchange-part-i.

Student category: IIA, both campus and online, but also for industry master students that want to take a project outside their own company.

The task is suitable for online students (not present at the campus): Yes. Most of the work can be done online.

Practical arrangements: None

Supervision:

As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc).

Signatures:

Supervisor (date and signature): Hans-Petter Halvorsen

Student (write clearly in all capitalized letters): EIVIND KNUDSEN

Student (date and signature): 26.01.2023 E. Knudsen

Appendix B Log File Example

```
┌ Info: 2023-05-06 20:23:57 Application started.
└ @ Main ~\RadarDisplay\src\display\a-scope.jl:5

┌ Info: 2023-05-06 20:23:57 receive_udp, thread id:
│   Threads.threadid() = 1
└ @ Main ~\RadarDisplay\src\sockets\receive_udp.jl:4

┌ Info: 2023-05-06 20:23:57
│ First received message decoded successfully as ASTERIX
│ cat:                    240
│ len:                    8238
│ fspec:                  59306 (bit representation: 1110011110101010)
│ sac:                    255
│ sic:                    0
│ msgtype:                2 (VideoMessage)
│ msgindex:               105842118
│ start_az:               3.0311 rad (173.6719 deg)
│ end_az:                 3.0327 rad (173.7598 deg)
│ start_rg:               0
│ cell_dur:               61783612
│ c:                      0 (No compression applied)
│ res:                    4 (High Resolution - 8 bits)
│ nb_vb:                  8192
│ nb_cells:               8192
│ rep:                    128
│ Length videocells:      8192
│ time_of_day [s]:        33525.3
│ special_purpose_length:11 bytes (including this byte)
│ special_purpose_field: UInt8[0x00, 0x0f, 0xa4, 0x9a, 0x00, 0x00, 0x00, 0x6d,
0x00, 0x00]
└ @ Main ~\RadarDisplay\src\display\a-scope.jl:73

┌ Info: 2023-05-06 20:23:57
│ FRN - Item types included in first received message:
│  1 - Data Source Id
│  2 - Message Type
│  3 - Video Record Header
│  6 - Video Header Femto
│  7 - Video Cells Resolution & Data Compression
│ FX - Field Extension indicator
│  8 - Video Octets & Video Cells Counters
│ 10 - Video Block Medium Data Volume
│ 12 - Time of Day
│ 14 - Special Purpose Field
└ @ Main ~\RadarDisplay\src\display\a-scope.jl:74
```