CHALMERS

Enhancing the MARV with Multi-Camera Support for Improved Remote Operation over 5G

Bachelor's thesis in Systems and Control

Shada Al-Wakkal, Othman Belal, Gabriel Bengtsson, Anna Lithell, Oskar Ulstein

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Bachelor's thesis 2024

Enhancing the MARV with Multi-Camera Support for Improved Remote Operation over 5G

Shada Al-Wakkal, Othman Belal, Gabriel Bengtsson, Anna Lithell, Oskar Ulstein

CHALMERS

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2024

Enhancing the MARV with Multi-Camera
Support for Improved Remote Operation over 5G

Shada Al-Wakkal
Othman Belal
Gabriel Bengtsson
Anna Lithell
Oskar Ulstein

Supervisor: Viktor Lindström, Department of Electrical Engineering
Examiner: Petter Falkman, Department of Electrical Engineering

Bachelor's Thesis 2024
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The MARV with the installed camera system.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Abstract

The Marine Autonomous Research Vehicle (MARV) project aims to improve the efficiency of SSRS rescue missions. This report details the enhancement of the MARV with a multi-camera system for improved remote operation over a 5G network. The primary objective is to expand the field of view and enhance situational awareness through a system that supports a quad-camera setup in order to provide 360° coverage in future development of the MARV. The system streams high-definition video with low latency via the Secure Reliable Transport (SRT) protocol and GStreamer framework. A waterproof enclosure and connectivity infrastructure were installed, including a 5G router and network switch. The solution uses H.265 video compression for optimal quality under varying 5G network conditions. A user-friendly web-based interface was developed to be implemented in further development of the MARV.

Keywords: Autonomous Research Vehicle, Remote Operation, 5G, SRT, GStreamer, H.265, MARV, UI.

Acknowledgements

We would like to extend a wholehearted thank you to Ericsson and their employees Magnus Castell, Bengt-Erik Olsson, Annicka Ericson, and Henrik Sahlin. Your support during the test day at Lindholmen and in the audience of our presentation was highly appreciated. We especially appreciate the useful insights to our project. We warmly thank Hugo Drakeskär for helping us with hardware assembly and for additional support during our test day at Lindholmen. We would also like to thank Fredrik Falkman at the Swedish Sea Rescue Society for providing us with the necessary equipment. A special thank you goes out to our examiner Petter Falkman, whose support has enabled this project greatly. At last a special thank you to our supervisor Viktor Lindström. You have helped us tremendously throughout this project with your enthusiasm, knowledge and excellent guidance within this subject.

Shada Al-Wakkal, Othman Belal, Gabriel Bengtsson, Anna Lithell and Oskar Ulstein, Gothenburg, May 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis, listed in alphabetical order:

API	Application Programming Interface
ASIC	Application-specific integrated circuit
AV1	AOMedia Video 1 (A video encoding format)
ACU	Autonomous Control Unit
CASE	Centre for Advanced Systems Engineering
CSI2	Camera Serial Interface 2
CPU	Central Processing Unit
DbW	Drive By Wire
DC-DC	Direct Current to Direct Current (Converter)
FOV	Field of View
FPS	Frames per Second
FFmpeg	Fast Forward MPEG (Media Processing Framework)
Gige Vision	Gigabit Ethernet Vision (Camera Interface Standard)
GMSL2	Gigabit Multimedia Serial Link 2
GPU	Graphic Processing Unit
GStreamer	GStreamer (Open-Source Multimedia Framework)
H.264	MPEG-4 Part 10 Advanced Video Coding
H.265	High-Efficiency Video Coding
HRP	Husqvarna Research Platform
HTML5	Hypertext Markup Language 5
IMU	Inertial Measurement Unit
IPv4	Internet Protocol version 4
LAN	Local Area Network
MARV	Marine Autonomous Research Vehicle
MPEG-TS	MPEG Transport Stream
NAT	Network Address Translation
P2P	Peer-to-peer

ROS 2	Robot Operating System 2
RTC	Real-Time Communication
SDP	Session Description Protocol
SRT	Secure Reliable Transport
SSRS	Swedish Sea Rescue Society
UDP	User Datagram Protocol
UI	User Interface
WebRTC	Web Real-Time Communication

Contents

List of Figures

1
Introduction

The Marine Autonomous Research Vehicle (MARV) project started as a collaboration between the Swedish Sea Rescue Society (SSRS) and Chalmers. The primary objective was to equip a personal watercraft with autonomous capabilities. An autonomous rescue vehicle in the SSRS fleet could potentially optimize resource allocation within the organization. In 2022 a bachelor project introduced a driver-less solution using 5G technology provided by Ericsson. An operator on land was now able to control the MARV remotely. A webcam was installed on the MARV to demonstrate the potential of live video streaming over the 5G network. Building upon these foundations, this bachelor's thesis project conducted in the spring of 2024 aims to develop a system with enhanced video streaming capabilities.

1.1 Background

The MARV is a product of a collaborative project between SSRS, Ericsson, and Chalmers University of Technology. The MARV is a specialized test-bed platform that is centered around a customized personal watercraft outfitted with multiple sensors and an advanced control system, see Figure 1.1.

Figure 1.1: Picture of the MARV without additional equipment

1.1.1 Collaboration Overview

The Swedish Sea Rescue Society (SSRS) is the originator of the MARV concept. Upon complete development of the MARV, SSRS plans to deploy the MARV in their operations. Its autonomous capabilities can be utilized to increase the organization's efficiency regarding routine rescue missions, such as providing gas when someone runs out of fuel. Ericsson contributes with providing the team working on the MARV with communication technologies, notably providing 5G network coverage. Connection to a mobile network is essential for the remote operation capabilities of the MARV. Ericsson will host testing of the MARV in their sea backyard located at the company's site by Lindholmspiren in Gothenburg, Sweden.

1.1.2 Proposed System Enhancements

Autonomous systems may require human intervention in unpredictable marine environments. The MARV can therefore not rely solely on automation to function. Past projects have made it possible for a land-based operator to remotely control the MARV over the 5G network [1]. An operator can navigate the MARV by using a game controller from Xbox while receiving a live video feed from a front-facing web-

cam positioned on the MARV. However, this system is constrained by limitations in its field of view (FOV) and vulnerability to water [1]. To address this, the proposed enhancement involves integrating multiple cameras around the MARV and thereby expanding the visual coverage. While conceptually straightforward this initiative presents several technical challenges. These include managing increased bandwidth requirements while ensuring a clear image and video streaming in real-time. Delivering the solution in the form of an intuitive interface is essential for the operator to effectively assess and interpret the expanded visual feed [2].
The initial phase of this enhancement will be trialed on the MARV-HRP, which is a scaled-down version of the MARV utilizing similar hardware and ROS 2 software setup. Successful implementation of this prototype will pave the way for its application on the MARV.

1.2 Aim

The project aims to further enhance situational awareness and operational capabilities of the MARV by integrating multiple camera streams into a single 360° field of view, visualised through a suitable, easy-to-use interface. The suggested solution should act as a foundation for further related projects.

1.3 Objectives

The following objectives align with the project's aim and provide more detail to the intended outcomes of the project.

1. **Install a new camera system**
   The camera system should provide a view from the front, sides, and back of the MARV.
2. **Prevent salt water accumulation on camera housing**
   The camera system should be encased. An external mechanism will be implemented in order to get rid of water stains, grime and salt accumulation on the lens.
3. **Stream live video from multiple cameras on the MARV**
   Software providing streaming capabilities will be developed and

should allow streams from multiple cameras at the same time.

4. **Compress camera data streams**
   The data streams needs to be processed in a suitable format to allow live streaming over the 5G network.
5. **Enhance system security**
   Only one user at a time should be able to view and access the video stream.
6. **Develop algorithm for prioritizing data streams**
   Prioritizing camera streams is necessary to decrease the amount of data transmitted from cameras when they're of less interest.
7. **Enable adaptive resolution**
   The system should enable adaptive resolution to prevent packet loss when live streaming if varying 5G connection in the archipelago occurs.
8. **Develop a user-friendly interface**
   A user-friendly interface is crucial in order for a land-based operator to control and monitor the MARV in a safe way.

The camera performance requirements and goals within the scope of the project are listed in Table 1.1.

Table 1.1: 720p corresponds to a video with HD-quality, while 1080p is equivalent to full-HD quality video. Within the scope of this project, the waterproof rating is based on the International Standard Ingress Protection (IP) rating, where IP68 corresponds to protection against dust, dirt and sand and the capability to endure a water depth of 1.5 m [3].

Camera	**Requirement**	**Goal**
Camera resolution (p)	720	1080
Latency (ms)	sub 500	sub 250
Frames/second (fps)	25	60
Waterproof rating	IP68	IP68

1.4 Limitations

The project is time limited, spanning from the 16th of January 2024 to the 10th of May 2024. Together, the entire project group have approximately 100 dedicated hours per week for the project. Therefore, the project exploration will be limited to the aims and objectives listed in section 1.2 and section 1.3.
The project will utilize self-manufactured products only when no suitable option is available. The suggested solutions are therefore specifically tailored for the MARV project system and its components. Subsequently, the recommended solutions are not intended for other systems.
The project will not focus on developing our own security mechanisms since such precautions require much time and effort. Instead existing communication protocols with integrated security mechanisms will be used. The project aims to ensure the video stream is only accessible for one user at a time.
The project will not focus on developing nor enhancing the autonomous capabilities of the MARV, such as the control pad or ROS 2 system. Ethical considerations related to the autonomy of the MARV and security issues related to the control pad will therefore not be discussed further.
The project budget will exceed the given amount of 5000 SEK and is limited by reason.

2
Theory

2.1 The MARV

The MARV is a test-bed for developing and testing new navigation algorithms for controlling smaller marine vessels in coastal environments. Key technical components of the MARV are described in the following subsections, each highlighting a different aspect of the vehicle's design and capabilities.

2.1.1 Drive by Wire (DbW) System

The Drive By Wire (DbW) system of the MARV provides efficient power management and enables control over the vessel. Its capabilities of providing way-point following has been proven in sea trials of past projects. The DbW system is composed of several components including the Database, Power Distribution Unit, Nozzle Control Unit, Throttle Control Unit, and the Autonomous Control Unit (ACU) [4].

2.1.2 Autonomous Control Unit (ACU)

Central to the MARV's autonomous capabilities is the Autonomous Control Unit (ACU), based on the NVIDIA Xavier NX platform. It is the control hub of the MARV and is responsible for the control system, handling of all the ROS 2 Nodes as well as gathering telemetry from the radar and the IMU [4].

2.1.3 Inertial Navigation System (INS)

Integration of the SBG Ellipse2-D Inertial Navigation System (INS) enhances the MARV's navigation by providing positioning and orientation data. This system is very important for the vehicle's navigation and control tasks, ensuring its operations are precise and reliable.

2.1.4 ROS 2

A suite of ROS 2 nodes has been deployed on the ACU to support the MARV's operational framework. These nodes offer functionalities such as power management, scenario handling, data logging, and system management, which facilitate efficient algorithm development and system operation [4].

2.1.5 The MARV-HRP

The MARV-HRP (Husqvarna Research Platform) is a test-bed with similar capabilities as the MARV used for prototype testing. The test-bed consists of a converted Husqvarna Automower utilizing the same hardware and ROS 2 software setup as the MARV.

2.1.6 Aluminium Structure for Testing

Past projects related to the MARV have used an aluminium structure mounted behind the driver's seat in order to install different types of testing equipment. This structure will be reused for this project and act as a foundation for installing the necessary hardware components, such as the camera.

2.2 Hardware Overview Diagram

The diagram in fig 2.1 represents the integrated system architecture of the MARV, designed by Noel Danielsson and Viktor Lindström for their Master thesis in 2021. It provides an overview of how the different hardware components are integrated and connected together. For further reading and understanding of the system overview in figure 2.1, please consult their Master thesis [4].

2.2.1 Power Management

Ensures the power distribution to the Central Processing Units (CPUs), including an auxiliary power unit and a DC-DC charger for voltage regulation. Auxiliary power unit acts as a backup to support the primary power supply, ensuring that functions remain operational even if the main power system encounters issues.

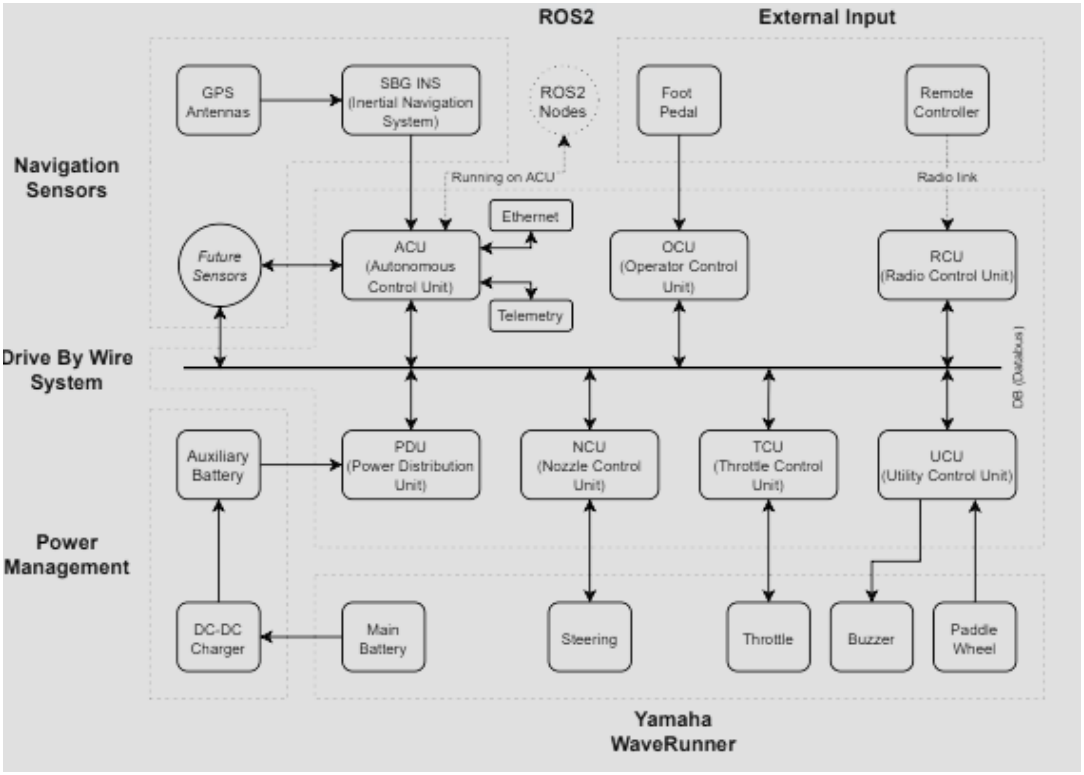


Figure 2.1: The MARV testbed system overview, showing how every part of the system is connected.

2.2.2 REACH

REACH is a custom built embedded computer based on the NVIDIA Xavier NX Development Kit. It is built to be rugged in order to survive marine environments and the vibrations associated with vehicular applications. The computer runs NVIDIA JetPack which is an alternative version of Ubuntu Linux. The I/O port configurations can be varied.

2.2.3 Camera System

A camera system was installed in the Bachelor thesis project conducted in 2022. According to their report, one camera of model Sandstorm S3Y3CAM was installed [5]. The camera was a USB connected 2012 model web camera, capturing video with 480p resolution and 30fps. The main goal of the solution was to demonstrate the possibility of streaming video from the MARV and did not focus on acquiring a high-quality stream [6].

2.3 UI

Relevant to the UI design are multiple design patterns, concepts and templates listed below.

2.3.1 Spatial and Habitual Memory

The patterns Spatial and Habitual Memory patterns both emphasize the importance of consistency on different levels. Spatial memory discusses the user's spatial memory towards the UI, showcasing the importance of components remaining in certain locations throughout pages. Habitual Memory pattern utilizes the user's digital literacy, particularly online habits, to make the design familiar to the user. With familiarity and developing a spatial memory regarding components, users can navigate the UI with more ease [7].

The design concept *Button groups* applies both design patterns by grouping buttons with similar functions together. Spatial Memory is applied by having similar functionalities in a certain area, which hints to the users the whereabouts of the desired button. Habitual Memory is applied by utilizing frequently used button groups in their commonly used position, which aids website navigation prompting functionality search in correct regions [7].

2.3.2 Canvas Plus Palette and Centre Stage

To help the user navigate to the most important component of the UI, the Canvas Plus Palette pattern advises placing an empty canvas on its centre stage with an adjacent menubar. The canvas maintains the heaviest visual weight, while the menu bar offers a manipulation tool operating on it. With this design, the benefits of Spatial Memory can be acquired due to the fixed location of the menubar [7].

The Centre Stage pattern is closely related to the canvas plus palette pattern, emphasizing that the most crucial component should be placed in the centre of the UI, achieving the same benefits as the canvas in Canvas Plus Palette [7].

2.3.3 Flow

In the context of UI, flow is defined as a mental state where the user is focused on a task, conducting it without disturbances. The UI aids in task execution by removing unnecessary distractions, thus simultaneously improving user satisfaction and effectiveness [7].

2.3.4 Visual Hierarchy

Visual Hierarchy is the practice of assigning visual weight to components based on their importance. The most important components should have the heaviest visual weight, with visual weight decreasing with lesser significance, thus creating a hierarchy of components. This concept contributes to a more accessible UI by naturally directing attention to crucial information or functionality [7].

2.3.5 Safe Exploration

Safe exploration emanates safe user navigation by providing an "escape hatch" that redirects the user to a familiar page, as well as encouraging exploration without causing dire side effects, promoting accessibility and positive emotions in user experience [7].

2.3.6 Colors, Pointer, Text, and Icons

Colour combinations used in UI should be harmonious and express the desired emotion clearly. Colours in the range of blue and white are associated with calmness. Colours should also highlight the visual weight of a component [7].

2.3.7 Web Accessibility Guidelines

For an accessible UI, the design needs to adhere to guidelines regarding web accessibility. The guidelines state that colours should indicate that the component is interactive, as well as hint at whether a component is active or inactive. Components and elements should also adhere to a high contrast between them, with a minimum ratio of 4.5:1 [8].
Pointers, texts and icons should also aid in visual communications. The text should be short and targeted. Icons should be clear and

universally acknowledged. The pointer should change form over components to signify interactivity and activity status [9].

2.3.8 Target Audience

The target audience for the design of the MARV UI is defined as the members of the SSRS. As of 2024-04-04, there is no available information regarding the age of its members. Necessary UI design choices include colour usage that conveys calmness. General UI design should strictly adhere to the web guidelines regarding accessibility [9].

2.4 Network Communication

In this section a brief overview of different network components related to the project is provided.

2.4.1 LAN

A local area network (LAN) is a computer network within a limited geographic area, such as a private home or at an office. Today most LANs are connected to a wide area network (WAN) or the Internet, however they can also be used as an isolated network where their sole purpose is resource sharing [10].

2.4.2 Ethernet

Ethernet is the dominant standard for LAN communication. It provides a collection of standardized methods and implementations used for network communication [11]. Using Ethernet cables is one of the simplest ways to create a LAN, where the most common connector for the cables is RJ45. RJ45 connectors allow Ethernet cables to easily connect to different devices in a LAN.

2.4.3 SFP

Small Form-factor Pluggable (SFP) provides a network interface for devices and can be used in telecommunication and data communication applications [12]. Using SFPs can be advantageous since different types of transceivers can equip individual ports, meaning networks can be designed according to one's need.

2.4.4 Router and Modem

A router and modem is needed in order to connect a device to the internet. Routers are responsible for sending and receiving data between networks while keeping track of connected devices IPv4 addresses. Multiple devices can then connect to the same internet connection. A router is connected to a modem responsible for communication with an ISP (internet service provider).

In this project the E-Lins H685 series 5G router with Ethernet is used. Model H685f-W has the functionality of both a router and modem used to establish an internet connection. According to their product description, the router provides a mission-critical connectivity in challenging environments and is protected against humidity [13]. The router will be installed on the MARV enclosed in brackets, see Figure 2.2.

Figure 2.2: The router situated within grey brackets

2.4.5 5G

5G is the fifth generation in mobile communication technology. It provides lower latency, greater bandwidth, and faster speeds than previous generations. 5G can operate over a much wider range of

frequencies compared to 4G, from less than 1 GHz up to 100 GHz [14]. Lower frequencies provider longer coverage but also lower bit rates. The higher frequencies are necessary to provide high bit rates, up to 20 Gb/s, and super low latency with the trade off of shorter coverage. By combining different frequency ranges, 5G can provide both high bit rates and good coverage.

2.4.6 Expericom

Expericom is Ericssons's internal mobile network provider [15]. It provides 5G coverage at Chalmers in the CASE lab and down by the harbor at Lindholmen in Gothenburg. Expericom is used for testing purposes and innovation-related work.

2.5 Streaming Protocols

A streaming protocol describes how data should be packaged and sent between two users on the internet. This section describes two different video streaming protocols, WebRTC and SRT.

2.5.1 WebRTC

WebRTC is an open, royalty-free platform [16]. It provides a collection of protocols, standards and APIs that can be used to enable RTC (Real-Time Communication) capabilities for browsers and apps. The WebRTC functionality of passing data is based on UDP (User Datagram Protocol).

WebRTC establishes a peer-to-peer (P2P) connection between two peers who wish to communicate with each other. When an WebRTC connection is established, two peers can take turns sending and receiving data in real-time securely [17]. WebRTC utilizes the SDP (Session Description Protocol) to initiate a "handshake" between the two peers before a connection is established.

Establishing a connection between users on the same LAN is relatively easy, since no routing or address translation is required. However, firewalls and NAT (Network Address Translation) might interfere with the SDP handshake when two peers are connected to different

networks. Therefore an external server must be set up to handle the data transmission between the two peers [17].

2.5.2 SRT

SRT (Secure Reliable Transport) is an open source video transport protocol. It is a modified version of the UDT (UDP-based Data Transfer) protocol and is designed to optimize video streaming across unreliable networks [18]. Haivision recently announced the upcoming release of SRT Peer-to-Peer with universal HTML5 support.

SRT supports a wide range of features, such as low latency recovery of packet loss, timing recovery of video and audio streams, end-to-end encryption and a simplified firewall traversal [19]. It is also content agnostic, meaning it does not rely on any specific data format, codec, frame rate or resolution for data transmission. SRT has built-in mechanisms for AES data encryption [20].

Data control and packet loss recovery is implemented in a few ways. Buffer sizes are configured in milliseconds to control the amount of time it takes to transmit data. When a timeout error occurs, periodic NAK (Negative Acknowledgment) packets are sent instead of re-transmitting data, which uses less bandwidth [18].

SRT adjusts its packet period by measuring the input stream bit rate. This feature allows SRT to dynamically adapt to changes in the network. Control packet timestamps are included in each packet to accurately time the data flow, a feature that is utilized by prioritizing the latest packets if congestion occurs.

2.5.2.1 SRT Configuration

SRT has a wide range of configurable parameters. For the scope of this project the focus lies on the parameters that affect the latency of the stream. The latency parameter defines how long the sender and receiver buffer should be. An estimate of how long this buffer should be for a stable transmission can be calculated by the following method.

1. Measure the RTT (Round Trip Time). RTT is the time it takes for one package to travel back and forth between two nodes.
2. Measure the worst-case loss rate. This is the worst percentual data loss during a 60 second period.

$$\text{data loss} = \frac{\text{retransmitted data in bytes}}{\text{transmitted data in bytes}}$$

3. Finding the right RTT multiplier. It is a value defined by SRT and depends on the RTT and packet loss rate.
4. The latency parameter is then estimated by

$$\text{latency} = \text{RTT multiplier} \cdot \text{RTT}$$

For more information see Hivisions configuration guide [21].

2.6 Web Application

Web applications are programs stored on a server, delivered through a browser interface over the internet. The front-end functionality, such as UI design, is usually done in languages such as CSS, HTML and JavaScript. The back-end can be programmed in Python.

2.6.1 HTML5

HTML5 is the latest standard for web-based applications. The standard was developed by the World Wide Web Consortium (WC3) to prevent proprietary technologies [22]. To allow communication between two peers, the HTML5 standard is supported by WebRTC streaming and thereby H.264 encoding [23].

2.7 Multimedia Frameworks

Multimedia frameworks are used to handle digital media, such as video content. They provide useful APIs and functions used to process data before transmission between two users. Within the scope of this project, they are essential in order to process, send and receive video streams.

2.7.1 GStreamer

GStreamer is an open-source, pipeline-based framework with a plugin architecture used to create multimedia applications. Depending on what codecs and other functionalities are of interest, different plugins can be added and removed easily. Data can flow between two peers through the pipeline without an intermediary server [24]. The entire GStreamer framework is distributed into different packages, where the SRT functionality is provided through the package gst-plugins-bad. According to their website, the gst-plugin-bad package is a set of plugins "that need more quality" [24].

2.7.2 FFmpeg

FFmpeg is a command-line tool, providing an open-source collection of libraries and pre-made programs used for handling media content [25]. Since FFmpeg focus on delivering finished tools for media processing, it's limited in terms of customization. On the other hand, FFmpeg has a large community providing extensive documentation on its features.

2.8 Encoding

Encoding refers to the act of transforming something into a format suitable for message transmission or simplifying data into a more concise form [26]. This project focus on video encoding since raw video is too large to send over 5G in real-time.
Nvenc is a Hardware encoder built in to many NVIDIA GPUs and SOCs. Hardware encoders are ASIC's that run the supported encoding algorithms much faster than any general CPU.

2.8.1 H.265

H.265 is a newer standard of H.264 and was proposed to reduce the bit-rate with 50% compared to H.264 [27]. The increased performance results from improved algorithms with added complex processing. To achieve more complex processing, H.265 encoding consists of technologies patented by different owners. As a consequence, using H.265 encoding often requires a license. The deployment of H.265 compatible devices was therefore slow at a beginning, however it has seen

increasing support throughout the years. Hardware encoding capabilities has been a part of NVIDIA NVenc encoder since the third generation (2015) [28].

2.8.2 MPEG-TS

MPEG transport stream (MPEG-TS) is a container format used for transporting and storing digital data. MPEG-TS is a part of MPEG 2 which is an old international standard for digital television which was approved in 1994 [29]. It is still being widely adopted and it is the container type SRT originally was designed for [18].

2.9 Lens Selection

In order to select the optimal lens for the project, multiple technical aspects should be considered.

- **Focal Length:** The distance from the lens to the sensor when the subject is in focus, typically measured in millimeters (mm).
- **Field Distance:** The distance from the camera to the subject, which significantly affects the field of view.
- **Sensor Size:** The size of the camera's sensor dictates the angle of view that the camera can capture.

2.10 Ethical Considerations

In this section a brief overview of how autonomous vehicles might be of interest for military applications is given. Projects related to the development of autonomous robots are heavily funded by military organizations [30]. To keep soldiers out of harms way motivates governments to fund the development of new weapons. In the war between Ukraine and Russia, Ukraine has supposedly been using modified personal water crafts as bomb drones [31].

3
Method

3.1 Project Resources

The team had access to the CASE-lab at Chalmers, where system development and solution implementation was conducted. Visual Studio Code, PyCharm and Visual Studio were used during software development. GitHub Copilot and ChatGPT was used for debugging purposes and provided software design suggestions. The pipeline was designed and tested directly in a command prompt. The project utilized the Ericsson mobile network Expericom, providing 5G coverage. All developed code within the project was stored in a private repository on the hosting platform Github.

3.2 Hardware Design and Integration

The following section describes the design and integration of hardware conducted within the project.

3.2.1 Design of System Overview

To create a comprehensive idea of the project's design and technical components, a detailed system overview diagram was designed using Google draw.io. The diagram was constantly updated during the ongoing development of the MARV. The system overview is essential to visualise the inter-connectivity between various devices and tools and how the data flows between different components. See overview diagram in Figure 4.1.

3.2.2 Auxiliary Management Unit (AMU)

To process the vast amount of data that the cameras produce, it was decided early on to add a secondary REACH-unit. This unit

is equipped with a 10 Gb network card as a single camera can output 1 Gb/s and it also needs bandwidth to upload the compressed stream. The AMU was mounted by our supervisor using a bracket he designed.

3.2.3 Switch

To be able to aggregate the data from the different Ethernet devices a switch was needed. The supervisor chose an Allied Telesis IE220-10GHX as it had the required amounts of ports, was compact and ran on 48-57V DC. It has two 10G SFP+ ports. As the cabling used RJ45 connectors, two MikroTik S+RJ10 adapters was bought in order to connect the copper cabling with the SFP+ port. The switch is driven by the DC-DC converter which takes the input 12V to 48V.

3.2.4 Installation of Routers

The routers described in section 2.4.4 was fixated on top of the DC-DC charger by using 3D-printed brackets designed by the project supervisor, see Figure B.2 and Figure 2.2. The routers can support six antennas at the same time, where two antennas can be used for 4G, 5G and Wi-Fi respectively.

3.2.5 Integration of Connectivity Components in Waterproof Enclosure

A waterproof case was ordered by the project supervisor in order to isolate electronic components completely from water. Multiple hardware components have been installed in the waterproof case, such as the switch, DC-DC power supply and the router, see Figure 4.3. The customized bracket fixed the DC-DC converter, router and switch to the enclosure, see Figure B.4. On one side of the waterproof enclosure, nine mounting connectors were installed.

These connectors were installed to provide an isolated environment for the Ethernet connectors. As the 10G link demands better signal isolation, an 8a mounting connector was chosen for port 10. This was wired up by the research assistant Hugo Drakeskär. Additionally, three patch cord flex connectors was installed. These connectors were mounted externally to the mounting connectors. The primary func-

tion of these flex connectors is to safeguard the connections outside the waterproof case against water.

On the lateral sides of the waterproof case, both the right and left sides are equipped with three flange adapters These adapters facilitate the connection of SMA socket connectors to type N coaxial connectors, ensuring a stable link. Internally, each flange adapter is connected to the router via Telegartner male SMA to male SMA coaxial cables On either sides there are three flange adapters, one for the 5G antenna, one for the 4G antenna and one for the Wi-Fi. Externally connected to these flange adapters are the respective antennas, each securely housed within a protective white tube. This housing is important as it shields the antennas from water intrusion and adverse weather conditions, thereby preserving their functionality and performance, see Figure B.5 and Figure B.6.

3.2.6 Camera Selection

The selection of an appropriate camera is important for the visual data acquisition capabilities of the MARV. The Basler ace 2 Pro model a2A1920-51gcPRO was chosen by the supervisor based on its advanced features that cater to the project requirements for high resolution and frame rate, along with robustness suitable for marine environments [32].

3.2.7 Lens Selection

The parameters listed in 2.9 was provided to an online field of view calculator. In the calculations, a focal length of 28 mm was used. Based on the results shown in Figure B.1 a suitable lens was selected by the project supervisor. The lens ordered was the Ultra Wide, No Distortion Lens model MY23F developed by Thiea technology and is pictured in Figure 3.1 [33]. The specifications for the lens is provided in 4.1.4.

3.2.8 Camera Enclosure

To ensure the Basler ace 2 Pro camera's functionality in marine environments, a custom-made waterproof enclosure was ordered by the

Figure 3.1: The camera and its lens

project supervisor. The enclosure provides robust protection against water ingress, thereby guaranteeing the camera's operational integrity during maritime deployment. Since it is custom made it does not have a specific IP rating, meaning it is difficult to verify if it can be classified as IP68 mentioned in 1.1.
For the camera to function effectively within the enclosure, the design incorporates a transparent window for visibility. The enclosure includes an entry point for cables to power supply the camera. Additionally it is very important that the enclosure is waterproof to protect the internal components from moisture and water exposure. Tubes for a future air pressure system has been mounted on the enclosure to prevent salt water accumulation on the lens, see Figure 4.4c and Figure 4.4d.

3.3 Software Development Prerequisites

Before any software was developed, GStreamer and SRT was installed for Ubuntu and Windows systems. Visual Studio was then configured to allow GStreamer functionalities. Appendix E provides full details on what installations were done and how they were completed in order to start software development. To develop the UI, the online design application Figma was used.

3.4 5G Modem Configuration

The project utilized two E-lins H685 5G routers to allow communication over the 5G network. Both was configured to communicate via the Expericom mobile network. The client modem was configured to operate with a dynamic IPv4 address while the MARV's modem was configured to be static on Ericsson's private 5G network. Using static IPv4 addresses for the MARV is desirable because the IP never changes, allowing for easy remote access. Both routers were configured to use NAT. This allows multiple clients to connect to the same network. All private nodes get assigned an unique private IP and the router have one public IP which handles all traffic between the private and public network. The router then manages and routes all incoming and outgoing traffic with the private nodes. To get past firewalls and route the traffic to the correct node, port forwards has been assigned.

The following ports have been forwarded on the MARV:

- For camera streams: 7000-7003 routed to the AMU
- For control of the MARV: 1337 routed to the ACU
- For access with SSH and SCP: 22 to both the ACU and AMU

The following ports have been forwarded on the receivers router:

- For camera streams: 7000-7003 routed to all hosts
- For control of the MARV: 1337 routed to all hosts

Static IP leases have been configured on the MARV to ensure that the local IPv4 addresses of the ACU and AMU always stays the same.

3.5 SRT Streaming

After building the SRT library and configuring the modems, the next step was to establish a connection between an SRT source and sink.

3.5.1 Using One Computer

Two command prompts were opened on a computer. The following command was executed in one command prompt to configure the SRT sender:

```
./srt-live-transmit udp://:1234 srt://127.0.0.1:4201 -v
```

In the second command prompt, an SRT receiver was specified by executing the following command:

```
./srt-live-transmit srt://4201 udp://127.0.0.1:4000 -v
```

The receiver side indicated if the connection was accepted while the sender printed a message stating the connection was established. This method was tested for both Ubuntu and Windows.

3.5.2 Using Two Computers

A REACH computer and a Windows computer was connected to the same LAN. After establishing a connection to the same Wi-Fi, the addresses for the wireless LAN IPv4 connection were retrieved for the two computers. For the REACH computer, this was done by executing the ”ifconfig” command in a command prompt. Similarly, the ”ipconfig” command was executed on the Windows computer. The following command was executed on the REACH to initiate it as a sender.

```
./srt-live-transmit udp://:1234 srt://192.168.1.94:4201 -v
```

Where ”192.168.1.94” was the Windows computer’s wireless IPv4 address on the LAN. To set up the Windows computer as the receiving side, the following command was executed.

```
./srt-live-transmit.exe  srt://192.168.1.94:4201
   udp://127.0.0.1:4000 -v
```

3.6 Video Streaming Pipeline

After successfully establishing an SRT connection between two computers, the next step was to send data.

3.6.1 FFmpeg

Once FFmpeg was built on the REACH, the ”FFmpeg example with SMPTE bars test video source” from SRT CookBook was executed [34]. The sender sent a video defined by executing the following command in a terminal.

```
ffmpeg -f lavfi -re
        -i smptebars=duration=60:size=1280x720:rate=30
        -f lavfi -re \
       -i sine=frequency=1000:duration=60:sample_rate=44100
          -pix_fmt yuv420p \
       -c:v libx264 -b:v 1000k -g 30 -keyint_min 120
          -profile:v baseline \
       -preset veryfast -f mpegts
          "srt://127.0.0.1:4200?pkt_size=1316"
```

The receiver was initialized in another command prompt by executing the following code.

```
srt-live-transmit srt://<port> file://con | ffplay -f mpegts
   -
```

The example worked, proving packages of data can be sent between two peers using the FFmpeg framework for SRT communication. No additional development using FFmpeg was done, see Section 5.2.3 for further discussion.

3.6.2 GStreamer

In order to send data, a pipeline was created by using the GStreamer framework. Designing different pipelines was done through an trial-and-error approach in a command prompt similar to the method described in 3.9.1. The GStreamer tool gst-launch-1.0 was used to prototype different pipeline configurations. When a pipeline configuration proved to be successful, a video was broadcast through the gst-launch-1.0 tool. The working pipeline configurations can be seen in 4.2. A

full list of the pipeline configurations that were explored is listed in Appendix G.

3.6.3 Encoding Pipeline Design

To capture images from the Basler camera via the camera interface GigEVision, the gst-plugin-pylon plugin was chosen. The plugin was chosen as it had tailored features adapted for the Basler camera. However, gst-plugin-pylon is not supported by Basler. The original idea was therefore to place the images directly from the camera into the video memory. This solution did not work out and the team lacked the expertise to fix it. Instead the raw video was first placed in the primary memory before being transferred to the video memory for conversion and encoding. The NVIDIA H.265 encoder plugin for Linux was chosen for encoding as it made sense according to the background provided in (2.8).

3.6.4 Implementation of the Pipeline

After a working prototype pipeline was in place the implementation phase began. The pipelines were implemented using the GStreamer API for C development. A flexible solution was created and allows further possible implementations to be made. Future possible implementations are GUI integration as well as choosing and prioritizing video streams. Currently the implemented pipeline solution retrieves statistics from SRT. The full code is provided in Appendix C.1. A parser script was implemented in Python to parse and plot the statistics retrieved from SRT, see C.2.

3.7 UI Development

The UI development consisted of four phases: research, low-level prototyping, high-level prototyping, and implementation.

3.7.1 Research

The research phase involved studying literature relevant to the target audience, UI design, and web accessibility, as well as to form an understanding how these concepts relate to one another. This phase also

included examining existing video streaming UIs, such as YouTube, through the lens of UI design concepts. The ultimate purpose of this phase was to identify existing solutions and concepts that can be integrated to the layout of the project's UI.

3.7.2 Low-level Prototyping

The prototyping phase was related to forming a sketch of the basic design of the UI in Figma. Figma is a tool for designing user interfaces. The sketch consisted of concepts, patterns, and templates without aesthetic details.

3.7.3 High-level Prototyping

Once cleared in the low-level prototyping stage, aesthetic details and button interaction effects were added to the design in the high-level prototyping phase in Figma.

3.7.4 Implementation

The implementation phase translated the design in Figma into code in PyCharm with the website's general appearance were written in HTML language while the aesthetic properties and button interaction effects were in the CSS language.

3.8 Web Application

The team decided to implement the solution in a web application to allow easy integration of the UI design. However, the results were not satisfactory. Instead the team opted for the solution described in 3.6.4. In this section, an overview of the web application development is provided.

3.8.1 Integration of UI design

Once the UI design was complete, a web application was developed in Python. The functionality was based on the flask framework, providing useful APIs for connecting HTML files.
To start the web application, a computer executed the application script "app.py" in a command prompt. While the script was running, a

web application was launched in Google Chrome. The web application was served by a local host on port 5000. The code is provided in Appendix C.3.

3.8.2 Integration of Pipeline

After a working web application was in place, the next step was to stream video to the browser through a GStreamer pipeline. Many different solutions and approaches were explored. Unfortunately, none proved to be working well. A somewhat working implementation were done in Python. Using OpenCV and with their implementation of GStreamer it was possible to receive data from a pipeline and display one picture at a time in the browser, see Figure 3.2.

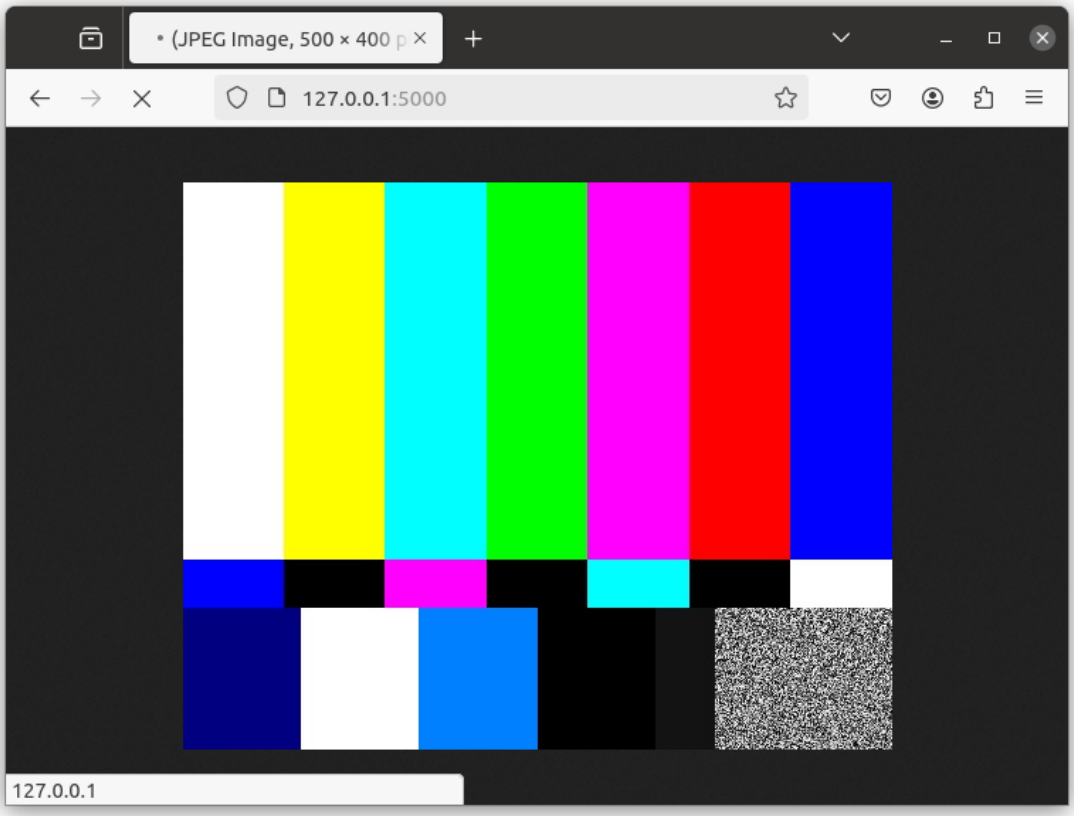


Figure 3.2: GStreamer pipeline sent to a web page

This proved to be heavy on the CPU and would only work for low resolution streams. The code can be found in Appendix G.1. A full discussion on what went wrong and why is provided in section 5.2.5 and 5.4.3.

3.9 Testing

Continuous testing was done throughout the project. The tests can be divided into two categories. Prototype testing was carried out before implementing a solution in the final application. These tests were carried out continuously during software development using the Expericom network at the CASE lab. One day was spent testing the entire system of the MARV at Lindholmen.

3.9.1 Latency Test

In Figure 4.6 a camera recorded the screen of a smartphone displaying a stopwatch. Simultaneously the camera streamed the video to a client's computer screen. The video stream was sent through a pipeline and was broadcast over the CASE lab's 5G network and then received and decoded by the client computer. The sender sent the video by executing the following code in a command prompt.

```
gst-launch-1.0 pylonsrc width=1920 height=1080 ! 'video/x-raw,␣
    width=1920,␣height=1080,␣framerate=30/1,␣format=YUY2' !
    nvvidconv ! 'video/x-raw(memory:NVMM),␣width=1920,␣
    height=1080,␣framrate=30/1,␣format=(string)NV12' !
    nvv4l2h265enc bitrate=2000000 ! mpegtsmux ! srtsink
    uri=srt://:7000 sync=false mode=listener latency=0
```

The receiving pipeline was configured by executing the following in a command prompt.

```
gst-launch-1.0 srtsrc uri=srt://10.81.248.3 mode=caller
    sync=false latency=0 ! decodebin ! autovideosink sync=false
```

SRT was configured to have a zero minimum latency and the clock sync ability was disabled for both SRT and the autovideosink element in the pipeline. The result from this test is provided in 4.3.

3.9.2 Testing Latency of H.265 Encoding and Decoding

By running the arguments below before the gst-launch-1.0 command GStreamer prints detailed latency from each of the element. However it does not print latency from the first and last item in the pipeline. Its printed in nanoseconds.

```
GST_DEBUG="GST_TRACER:7" GST_TRACERS="latency(flags=element)"
```

3.9.3 Test Day at Lindholmen

One day was dedicated to test SRT streaming capabilities in maritime environments. The camera and water-proof enclosure was mounted on top of the testing structure described in 2.1.6. New SIM-cards were provided by Ericsson, adapted solely for 5G network connection. However, several issues occurred during the day.
Two routers stopped working once given the new SIM-cards. The camera did not work as expected during the test. Instead the pipeline of the sender was reconfigured to use GStreamer's testvideosrc pipeline element. The resolution and frame rate of the test video were configured to match the settings of the camera.
All SRT statistics were extracted from the SRT sockets using GStreamer. Latency calculations were done using the method described in 2.5.2.1. The worst case loss rate was calculated by using the following python script.

```
length = len(parsed_data["bytes-sent"])
packet_loss_rate = 0
for i in range(0, length-60):
    bytes_sent_60s = parsed_data["bytes-sent"][i+60] -
       parsed_data["bytes-sent"][i]
    print(bytes_sent_60s)
    bytes_resent_60s =
       parsed_data["bytes-retransmitted"][i+60] -
       parsed_data["bytes-retransmitted"][i]
    print(bytes_resent_60s)
    tmp = bytes_resent_60s/bytes_sent_60s*100
    if tmp > packet_loss_rate:
        packet_loss_rate = tmp

print("Worstcase packet loss rate = " +
   str(packet_loss_rate) + "%")
```

Three complete tests where conducted. All test were configured to use video feeds with 1080p and 30fps. SRT's minimum latency were configured to 150 ms.
The first test involved streaming a single video stream from the MARV to a land-based operator. There was some issues concerning the control-pad which resulted in the MARV driving around slowly in circles.

During the second test, the MARV streamed four camera streams simultaneously. By now the control pad was working as expected and the MARV could drive around the harbor. The route and speed of the MARV during the test run is listed in A.1.
The last test was conducted with the same settings as test 2. The MARV was held still in the beginning of the test and then took a similar route as test 2.

4
Results

4.1 Hardware

In this section the results concerning the hardware implementation is provided.

4.1.1 Hardware System Overview

Figure 4.1 represents the final version of the hardware system. For detailed explanations of the components see 2.1 and 3.2. The camera feed is transported over gigabit-Ethernet, aggregated at the switch and then sent to the AMU. The AMU then compress it and send the camera feed to the router via the switch.

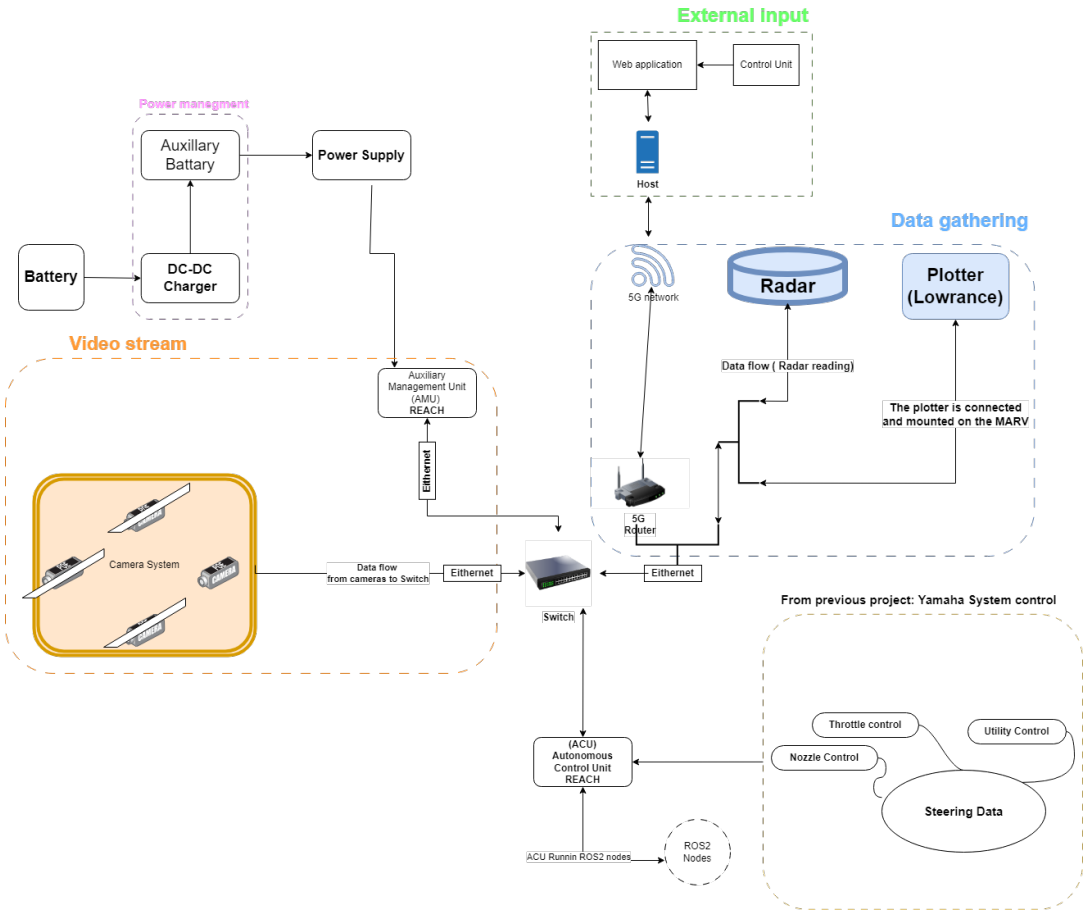


Figure 4.1: The MARV system overview, showing how different components of the system are integrated

4.1.2 Waterproof Enclosure

The enclosure is shown in Figure 4.2. The end product is very compact and in turn leads to narrow bending radii which is sub-optimal for cable health. The wiring diagram in Figure 4.3 shows a rough estimation of the Ethernet ports wiring and where the external ports are located.

4.1.3 Camera Specification

In Figure 4.4a and Figure 4.4b the Basler ace 2 Pro camera and its enclosure is shown. The enclosure isolates the camera completely from weather environments as mentioned in section 3.2.8. In Figure 4.4d and Figure 4.4c, the air pressure system is mounted on the front side of the enclosure.

Figure 4.2: A picture of the Waterproof case while being worked on.

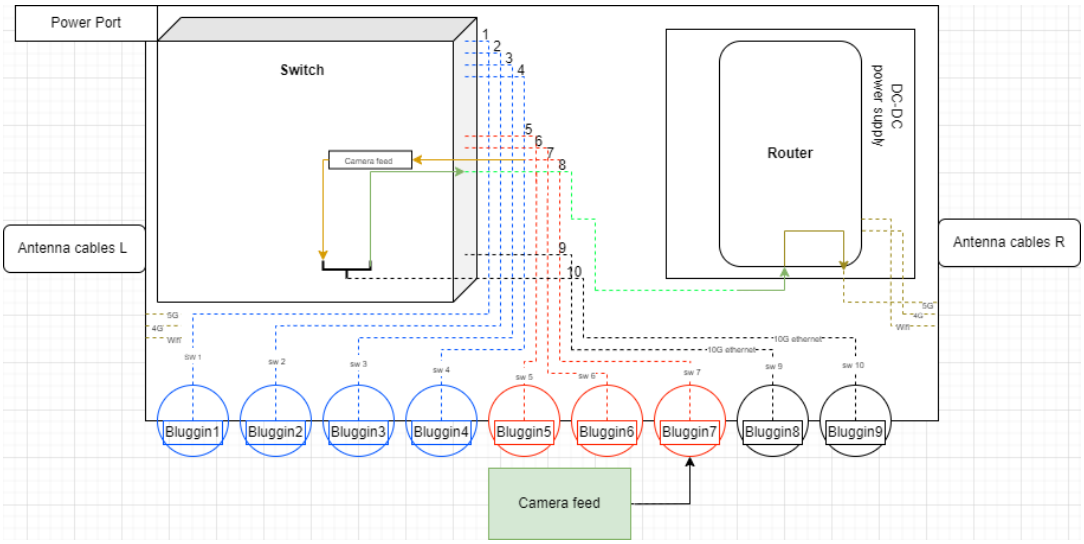


Figure 4.3: Waterproof case connectivity diagram

Technical Specifications The Basler ace 2 Pro camera specs.

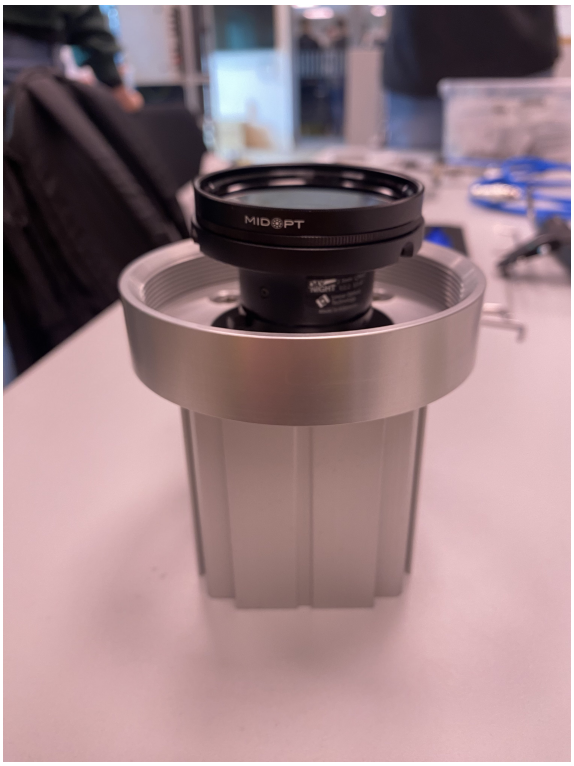
- **Resolution:** 1920 x 1200 pixels, which allows for detailed visual data capture.
- **Frame Rate:** Up to 51 fps at full resolution, offering smooth video streams.
- **Sensor Type:** Sony's Pregius CMOS rolling shutter sensor with high light sensitivity.
- **Interface:** GigE for reliable high-speed data transfer.
- **Lens Mount:** C-mount, providing a wide range of lens options.
- **Size:** 62.2 x 29 x 29 mm (including lens mount and connectors).
- **Weight:** Less than 105 g, adding minimal load to the vehicle.

4.1.4 Lens Specification

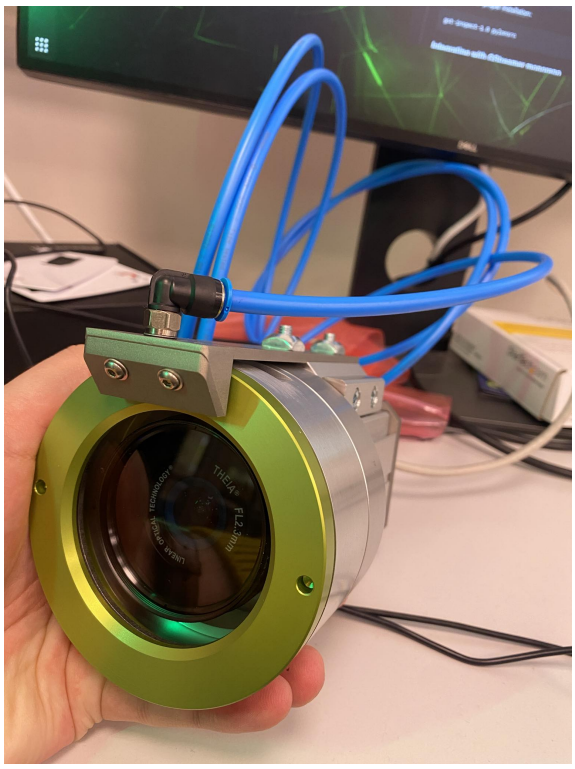
The MY23F lens offers a focal length ranging from 2.3 mm to 2.8 mm, providing a wide horizontal field of view between 115° and 83°, a vertical field of view between 90° and 66°, and a diagonal field of view from 139° to 106°.
The lens features a maximum aperture of F/1.8 and a minimum aperture of F/8.0, allowing for excellent low-light performance and versatile depth of field control. The MY23F lens is able to operate in temperatures ranging from -20°C to +60°C.
In addition to the lens, a linear polarizer filter PR032 was ordered by the project supervisor. The filter is used to eliminate and reduce glare

Figure 4.4: Four pictures of the camera and its enclosure

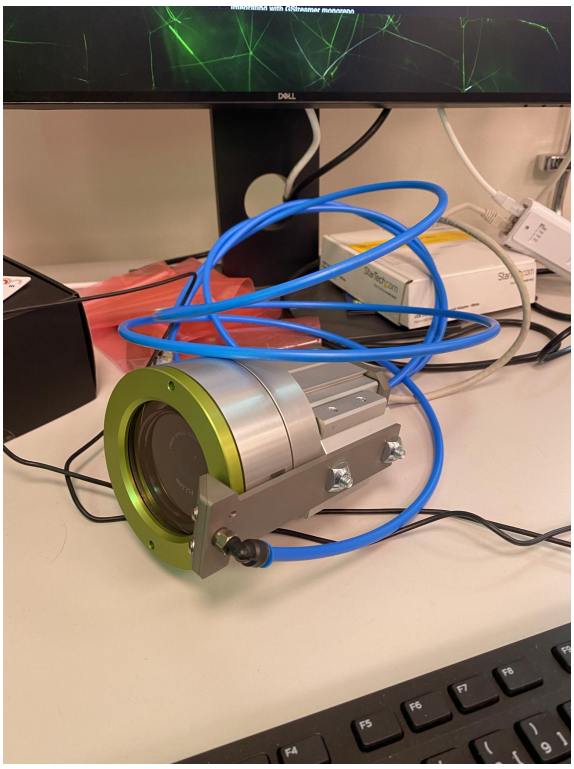
(a) The camera and its enclosure



(b) The camera mounted in the enclosure



(c) The camera in the enclosure with the air pressure system connected



(d) The camera in the enclosure with the air pressure system connected

from the maritime environment on the camera [35].

4.1.5 Hardware Installation on the MARV

A three-legged tower made of aluminum that was designed in the previous projects is mounted on the MARV's rear seat. On the sides of the structure, there are antennas for 5G, 4G, and Wi-Fi. At the top of the structure, there's a forward-facing Basler ace 2 Pro camera enclosed in a waterproof enclosure. Directly beneath the camera is the waterproof case that houses the switch, DC-DC power supply, and router, see figure 4.5a and Figure 4.5b.

(a) The Camera and the Waterproof case on the MARV

(b) The tower that holds the camera and the Case

4.2 Gstreamer Pipelines

The sending pipeline of the script is described by the following pseudo code.

```
pylonsrc->cap(YUY2)->nvidconv->cap(NV12)-> nvv4l2h265enc ->
   mpegtsmux -> srtsink
```

The receiving pipeline is modelled according to the following pseudo code.

```
srtsrc->tsparse->tsdemux->h265parse->d3dllh265->autovideosink
```

The following pipeline was tested where the frame buffer from the camera is put directly into the GPU's memory, however only half of the video was received.

```
    gst-launch-1.0 pylonsrc !   video  /x-raw(memory:NVMM),
       width=1920, height=1080, framerate=30/1,
       format=YUY2 ! nvvidconv !
         video  /x-raw(memory:NVMM), width=1920, height=1080,
       framerate=30/1, format=(string)NV12  !
       nvv4l2h265enc bitrate=2000000 ! h265parse ! queue !
       h265parse ! nvv4l2decoder ! nv3dsink -e
```

4.3 Prototype Testing

As seen in the Figure 4.6, there is a difference in the time displayed on the two screens. The difference between the two clocks is the total latency from when the phone displayed the time to when it is displayed on the client's computer. In this example, the measured latency was 270 ms.

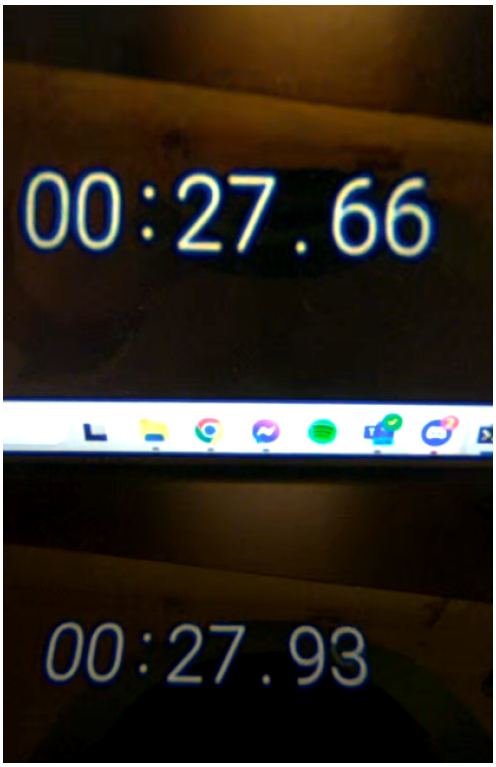


Figure 4.6: Latency test in CASE

4.3.1 Test Results Of Encoding Latency

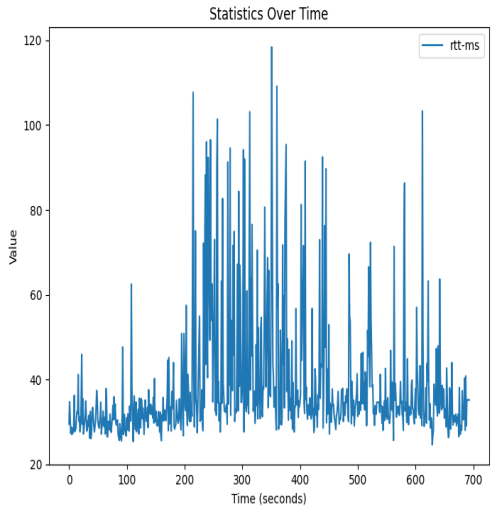
The info below reveals that the encoder takes about 18 milliseconds and the conversions adds another 22 milliseconds. It was obtained using the methodology described in 3.9.2.

```
element=(string)nvv4l2h265enc0, src=(string)src,
   time=(guint64)17783296
element=(string)h265parse0, src=(string)src,
   time=(guint64)376864
element=(string)h265parse1, src=(string)src,
   time=(guint64)360544
element=(string)nvvconv0, src=(string)src,
   time=(guint64)12975776
element=(string)nvvconv1, src=(string)src,
   time=(guint64)8022240
```

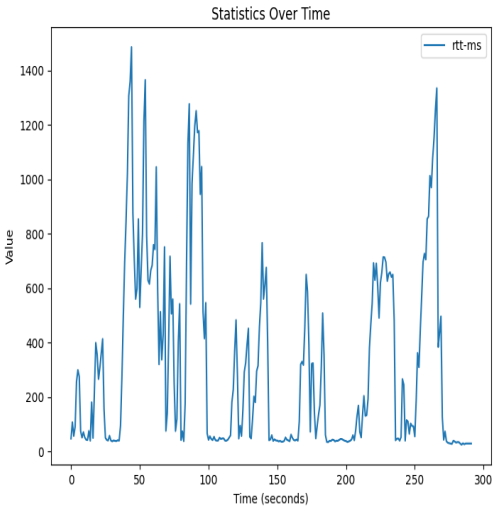
4.4 Test day at Lindholmen

The following section presents the data collected from the test day at Lindholmen.

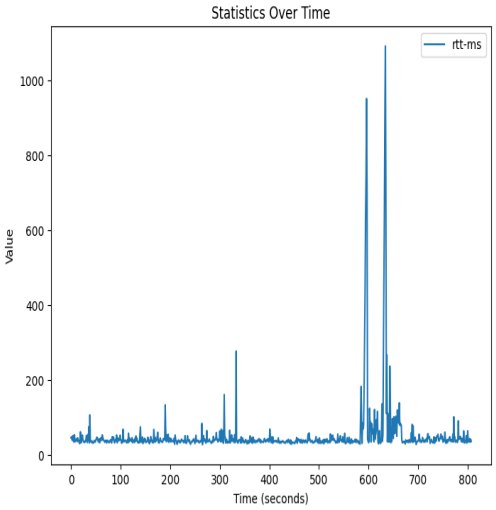
4.4.1 Round Trip Time



(a) Test 1: one video stream



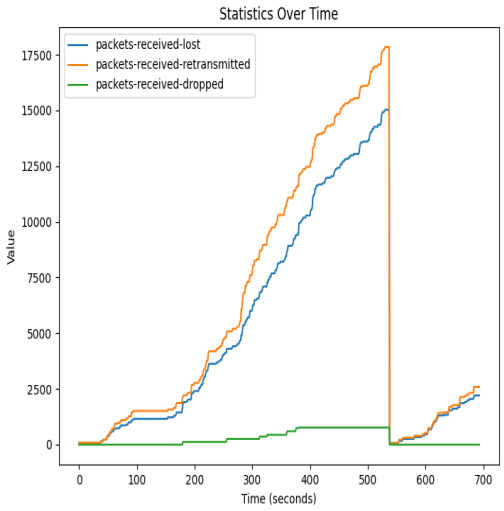
(b) Test 2: four video streams



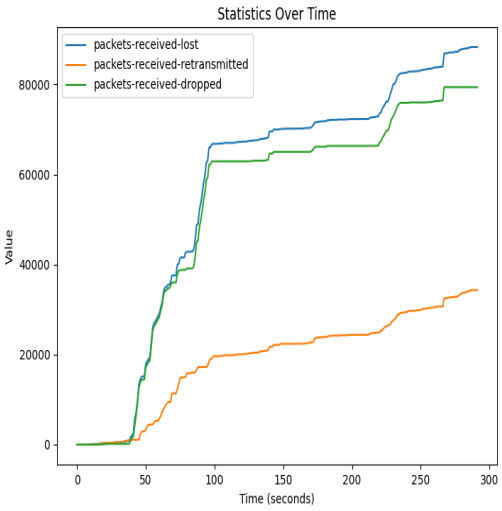
(c) Test 3: four video streams. The MARV was standing still at the beginning

Figure 4.7: RTT in ms

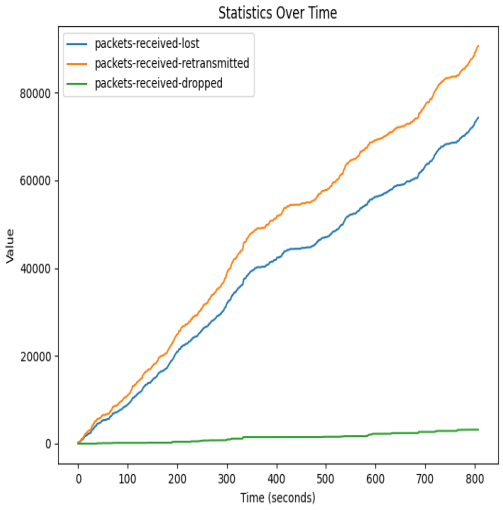
4.4.2 Packet Statistics



(a) Test 1: one video stream



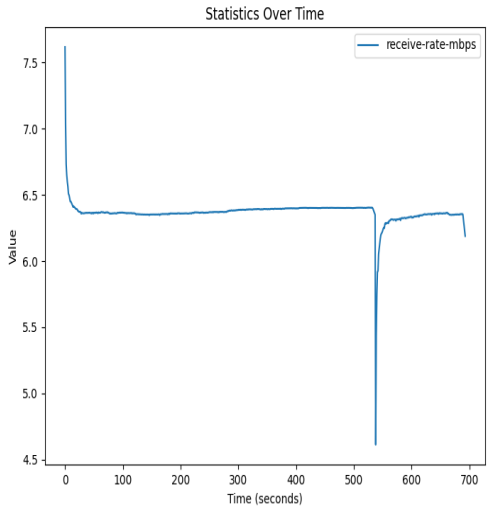
(b) Test 2: four video streams



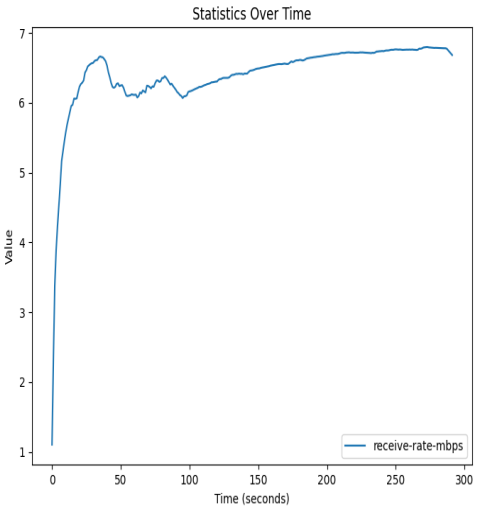
(c) Test 3: four video streams. The MARV was standing still at the beginning

Figure 4.8: Packet statistics for client

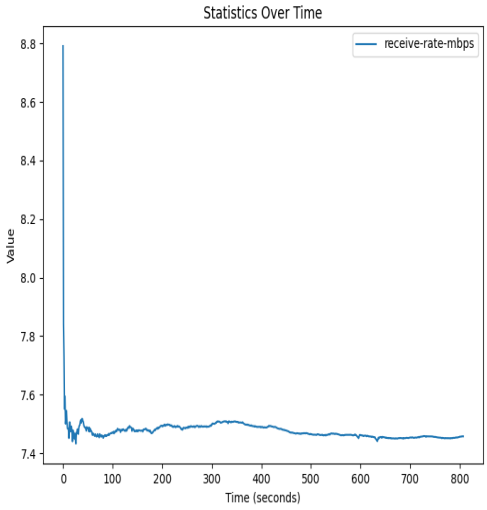
4.4.3 Receive Rate



(a) Test 1: one video stream



(b) Test 2: four video streams



(c) Test 3: four video streams. The MARV was standing still at the beginning

Figure 4.9: Receive rate for client in Mbits/s

4.4.4 Configuration of SRT Latency Parameter

See section 2.5.2.1 for more info about these calculations.

Table 4.1: Measures to determine minimum latency

	Test 1	Test 2	Test 3
Worst-case lost rate (%)	2.1	9.4	4.8
RTT multiplier	4	6	5
mean RTT (ms)	38.7	329	52.2
Minimum latency (ms)	116	1974.1	261.1

4.5 Final UI Layout

The final layout of the UI is showcased in Appendix D, where all the application screens and interactive button effects are shown.

4.5.1 Application Screens

The background colour of the Home view application screen is a white-grey type, while the background colours of the focus view screens are a stronger grey hue. See Figure 4.10

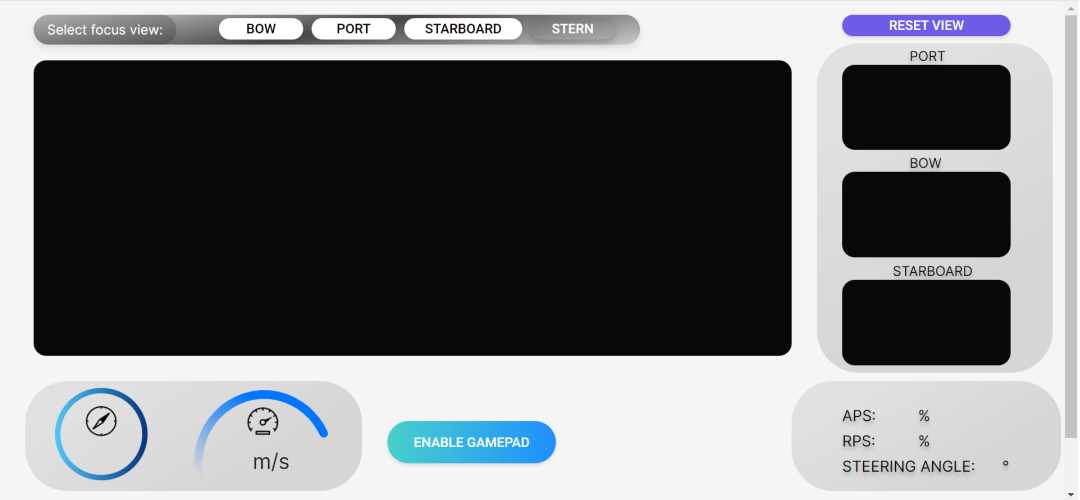


Figure 4.10: The home page of the UI

In the Home view, the main screen displays camera streams from the Bow, Port, and Starboard directions, with the streams from each direction divided into three equal-sized "boxes". The smaller screen, located on the right, shows the camera stream from the Stern view.

In focused views, the main screen displays the camera stream from the selected view, while the three screens positioned in a card to the right, vertically underneath each other, showcase the remaining views with each corresponding direction view label presented in black capital letters above the screen.

4.5.2 Navigation and Control

The navigation panel consists of a thin black bar with the text prompt "Select focus view" in white, along with four navigation buttons. The text has a shadow underneath it on its right side. The buttons are grey and the button labels are in black bold text. In a focused view, the selected button is black with a white bold label. The buttons navigate to their corresponding page, where the desired camera stream direction is shown on the main screen.

The cards with steering information and control units are located underneath the screens, with the "ENABLE GAMEPAD" button between them. Both the cards have a grey background. The first card, placed at the left side of the UI, displays a compass with information regarding the MARV's direction. The compass is encircled by a gradient blue eclipse, with a compass icon. In the same card, adjacent to the compass, the speed of the MARV is displayed. The velocity is showcased in m/s, along with an icon of a speedometer. The gradient blue line on the speedometer is dynamically animated, the length and colour intensity increasing with higher momentum. The second card presents details about the MARV's APS, RPS, and steering angle metrics, displayed in black text. The cards have the same size in height, the width of the card placed to the left of the "ENABLE GAMEPAD" has a bigger height.

4.5.3 Button Design and Effects

The "ENABLE GAMEPAD" button has the biggest height and width out of all the buttons. Its colours are gradient blue, with a brighter and more intense hue. The button has the label "ENABLE GAMEPAD" displayed in white bold text. On "click" action, the "Disable Gamepad" button comes into view with equal size, but with a gradient grey colour and the label "Disable Gamepad" showcased in white. The buttons ac-

tivate, and deactive, the gamepad steering for the MARV respectively.

The "RESET VIEW" button is only available on focus view application screens, and redirects to the Home view. The button has a dark blue colour with the same height as the navigation buttons but with a bigger width. The label "RESET VIEW" is written in white bold text. The "RESET VIEW" button is positioned at the same height as the navigation bar, located above the card with the grouped screens.

4.5.4 Interactive Effects

All buttons are equipped with interactive effects. The cursor on all buttons changes layout to "click" animation while hovering above the buttons. When pressing down on the buttons, the button presses in, when releasing the button, the button returns to its initial position and layout, creating a "bouncing" effect.

The navigation buttons while hovering transform to a gradient dark grey hue, with the label displayed in white. These effects are the same when pressing the button. While the button is active, the cursor does not change form while positioned above it, along with having the same appearance as in the hovering effect.

The "RESET VIEW" button while the cursor is hovering above it alters to a white-grey background color and the label is presented in black, along with a light black shadow displayed underneath it. this effect is consistent when pressing down on the button.

The "ENABLE GAMEPAD" button label transforms to black while the cursor hovering on it, and a light black shadow becomes visible underneath it. After pressing down on it, the button is replaced with the "Disable Gamepad" button. The "Disable Gamepad" button has a black shadow underneath it by default.While the cursor hovers over the button, it transforms to a lighter gradient grey hue. After pressing down on the "Disable Gamepad" button, the "ENABLE GAMEPAD" button reappears.

5
Discussion

5.1 Hardware

In the early stages of the project many possible solutions were discussed. After much deliberation a choice of camera was made and the rest of the system was adapted accordingly. In this section, a discussion regarding the hardware implementation is provided.

5.1.1 The Four Camera Solution

Different camera solutions were discussed in the beginning of the project. Using one single 360° camera or installing four cameras in each direction were the two realistic suggestions. In the end, the team opted for the four camera solution. Using four cameras instead of one provides more options regarding software customization. For example, if one camera stream is of less interest, different software solutions can be implemented in order to decrease the bandwidth used from that particular stream.

5.1.2 The Camera

The use of an FOV calculator was helpful in choosing the lens as it eliminated any guesswork. The GigEVision interface provided plenty of options of how to implement the transfer of images from the camera to the AMU. It is also compatible with Pylon camera software allowing integration with the existing systems including ROS 2.

5.1.3 The Camera Housing

The camera housing designed by our supervisor was a success. The team did not have the means to get an official IP rating, however it survived the entire test day without any signs of water intrusion.

In addition to the camera housing, the supervisor designed an air-pressure system to prevent salt water accumulation on the window. This was not ready in time for the test, as an air compressor had to be installed in order for the system to work. The camera housing weight surpasses 1 kg which needs to be considered in future project if additional cameras are to be installed.

5.1.4 The Switch, The Router and The Enclosure

There were no opportunities to test the IP rating of the enclosure. However, there were no signs of water intrusion by the end of the test day at Lindholmen. Unfortunately encasing the hardware components was done at the expense of signal quality. The 10G ports worked fine when connected directly to the switch, but the long cables and the couplers used within the enclosure meant that the signal loss was too great to establish a connection using the SFP+ adapters. The team lacked the time required to diagnose exactly what caused the signal drop, but the 10G adapter in the AMU is believed to be largely responsible. Other causes may be the couplers, flat panel connectors or the short bending radii of the internal Ethernet cables. A separate problem was that the switch required a license to unlock SFP+ functionality. This lead to unnecessary time spent debugging adapters.

5.2 Software Design

The following section provides a discussion concerning certain design choices made during software development.

5.2.1 Choosing SRT for Streaming

One of the first decisions made within the project was whether to choose WebRTC or not. Previous projects related to the MARV used WebRTC to stream video from a web camera. Since the camera component was upgraded to one with higher streaming capabilities, the new system needs to handle a significant increase in data compared to the previous solution. SRT is content agnostic, meaning H.265 encoding is possible. Using H.265 seemed to be a promising solution in order to decrease the amount of data transmitted by the video stream. The flexibility of SRT could facilitate the process of enhancing the appli-

cation further in the future by for example prioritizing video streams and enable adaptive resolution. Since SRT is built to operate over unreliable networks while streaming low latency, high-quality video (2.5.2), it seemed to provide a solid foundation for this project.

5.2.2 SRT Performance

As shown in Figure 4.6 the total latency for the entire streaming pipeline was around 270 ms. This is close to our target of 250 ms. The minimum latency was configured to 0, meaning there was no buffer and lost packages could not be recovered by SRT. This configuration of SRT worked well on the 5G network in CASE, but would not work in real life for unreliable networks. Unfortunately the result is based solely on observation since the SRT statistics was not retrieved for this test.

In the tests at Lindholmen there were a quite significant difference between the tests. Between the first and second test there was a significant increase in dropped packets when streaming four videos. For comparison the first test had a worst-case loss rate of 2.1% compared to the second test with 9.4%. This indicates that the minimum latency is set too low which means that SRT does not have enough time to re-transmit the lost packages. Since the SRT latency was configured to 150 ms and the mean RTT was over 300 ms this is expected. When streaming only one video, the minimum latency was also set to 150 ms but only had a few dropped packets. However the mean RTT for this test was only around 40 ms which gives enough time to re-transmit most lost packages. The RTT plots can be seen in Figure 4.7 and packet statistics in Figure 4.8.

A new test was conducted to investigate why streaming four videos led to such an increase in RTT and packet drops. This time the MARV was standing still in the beginning of the test and after a while it took a similar route as in the second test. As can be seen in Figure 4.7c the RTT is low and stable while the MARV is standing still but spikes when in motion. This could be explained by multiple reasons. It could be due to driving outside the reach of Ericsson's 5G network, it could be due to radio shadows from the buildings around or due to the fast

movements of the MARV.

The estimated minimum latency for the tests were calculated to 116 ms, 1974 ms and 251 ms respectively, see Table 4.1. These times adds up to the total latency of the system which means that the total latency would be quite high. Note that the estimation for the minimum latency value are a recommendation for achieving high quality consistent stream delivery. Since this application is dependant on low latency it would probably be beneficial to lower the video quality for increased latency performance.

Theses tests are inconclusive and more testing would be necessary. Since a test video was utilized for streaming it was hard to evaluate the quality of the video feed. It would be interesting to investigate how low the minimum latency could be tuned to get as low latency as possible and still maintain acceptable image quality.

5.2.3 Selecting the GStreamer Framework

The first attempts of sending data using the SRT protocol was done by using FFmpeg. The trials proved to be successful while FFmpeg seemed to be a promising and user-friendly development tool. However, further development using FFmpeg was discontinued. The reason behind this was the application's need for customization. In order to send a large amount of data over any network, it has to be encoded in a suitable format. Using SRT and opting for a solution with H.265 encoding limited the options drastically regarding what tools FFmpeg had to offer (2.7.2). On the other hand, the GStreamer framework provided the necessary plugins to package data in the desired format, while providing tools for SRT streaming (2.7.1). From this point and onwards, further development of SRT streaming was therefore conducted using the GStreamer framework.

5.2.4 Video Compression Using H.265

The H.265 codec was a good choice as the latency for encoding including converting between different formats was around 40 milliseconds (4.3.1). However this could theoretically be improved if the video buffer from the cameras could be put directly into the GPU's mem-

ory. This was for our implementation hindered by the bug mentioned in Section 4.2. This could not only eliminate a conversion-step but also hopefully lower the latency from the camera into the memory. This latency could not be measured with the performed tests.

5.2.5 Opting for a Web Application

Using the flask framework proved to be an easy way to integrate the UI design in a web application. Meanwhile SRT is an extension of UDT as mentioned in Section 2.5.2. Since SRT lacks P2P functionality and the pipeline uses H.265 encoding, the team realized the video cannot be streamed directly to a web application, see Section 2.6.1. From this point there were two available implementation options in order to stream video to a web application. The first one involved converting SRT to Web-RTC. Web-RTC provides RTC capabilities for video streaming in browsers and is supported by the HTML5 standard, see 2.5.1 and 2.6.1. This could potentially facilitate pipeline integration into a web application. However, this option would make the implemented SRT pipeline and weeks of progress worthless. The other option was to use the current pipeline setup in combination with Python plugins, aiming to provide GStreamer functionalities for web applications. The team decided to go forward with the second option, which unfortunately didn't turn out as expected. See Section 5.4.3 where further discussion on what went wrong and why is provided.

5.3 Design Choices of UI

The main purpose of the design choices is to enable flow (2.3.3) and to provide a web-accessible UI with a positive user experience.

The text font size and colour all meet the minimum web accessibility contrast ratio (2.3.7) against their respective background, which is crucial for the project's target audience. The colours are chosen to convey a sense of calmness and neutrality, which are also necessities for the target audience (2.3.8). All caps text is utilized to visually communicate a button's active functionality, for instance, to change the focus view. The labels are short and clear, suitable for the target audience, by adhering to the web accessibility guidelines.

5.3.1 Application Screens

The change in hue between the Home view and the focus view is to indicate to the user that a new application screen is active, which is a requirement for web accessibility (2.3.7).

The UI components are static in position through the application screens, with the dynamic instances being found between the Home view and the focus views, where the "RESET VIEW" button is added along with the three vertical screens - making the design in accordance with the Spatial Memory pattern (2.3.1).

5.3.1.1 Camera Streams Screens

The size of the screens adheres to the Habitual memory pattern (2.3.1), as well as the Canvas Plus Palette (2.3.2) and Centre stage patterns (2.3.2). The screen sizes can be found in the popular video streaming site YouTube which allocates the highest visual hierarchy (2.3.4) rank to the main screen, which is also used in our UI, hence the utilization of the Habitual memory pattern - whereas the screen sizes are also in accordance with the Canvas Plus Palette and Centre stage pattern. The navigation bar is classified as the palette. The positioning of the screens also adheres to these patterns by positioning the main screen in the centre, which also can be found in YouTube's UI.

5.3.1.2 Navigation Panel

The navigation panel mimics the design and position of the YouTube search bar, which yields advantages of the Habitual memory pattern (2.3.1). Moreover, the design and position communicate via visual hierarchy (2.3.4) that the navigation panels adhere to the camera stream screens through their close position and smaller visual weight in comparison with the screens. The navigation buttons' position and layout create a sense of *Button groups*, in alignment with the Spatial memory pattern (2.3.1). The text aids the user in decoding the functionality of the navigation buttons, following the web accessibility guidelines (2.3.7) regarding UI text.

5.3.1.3 Control and Steering Cards

The steering information and control unit cards are of the same height, which effectively conveys the same rank they have in the visual hierarchy (2.3.4). The icons are chosen from a library with universally recognized icons, which strengthens the UI's visual communication, hinting at the type of values presented, satisfying the guideline requirements (2.3.7) for icons.

5.3.1.4 Stand-Alone Buttons

The size of the "ENABLE GAMEPAD" button indicates to the user its gravest importance in comparison with other buttons, which aligns with Visual Hierarchy (2.3.4). Its gradient colour of a lighter blue is still in the calmness spectrum, but also indicates action and alertness, hinting at its functionality of enabling the steering gamepad, fulfilling the requirements of web accessibility regarding colours. The "Disable Gamepad" button's size and position allude to the user its related functionality to the "ENABLE GAMEPAD" button, while its gradient grey colours and lower caps text provide a visual clue of its functionality as an "inactivate" button. The "Disable Gamepad" button follows the same design principles and guidelines as the "ENABLE GAMEPAD" button.

The "RESET VIEW" main functionality is to be an escape hatch, emanating the Safe Exploration (2.3.5) pattern. Its dark blue colour is chosen to showcase calmness, hinting to the user that they can safely navigate back to the Home view, which satisfies web accessibility guidelines (2.3.7). Its width and position convey the related functionality it has to the navigation buttons, while also emphasising its higher rank in the visual hierarchy (2.3.4).

5.3.2 Interactive Effects

The interactive effects of the UI are designed to provide feedback on user activity, mainly by enforcing visual clues regarding button interaction. Diverse effects are used in the UI that adhere to the guidelines of web accessibility. The interactive effects are also consistent through the UI, which is in adherence with the Spatial memory pattern. All

interactive effects are designed after the web accessibility guidelines (2.3.7).

5.3.2.1 Animations

The bouncing effects of the buttons are designed after real-world button interaction, delivering various forms of visual feedback. The close relatedness between button interaction online and real-world yields an advantage for the targeted user audience by providing a sense of familiarity. Furthermore, the bouncing animation gives instant visual feedback that clearly indicates to the user that the action is registered, prominently revealing activity status.

5.3.2.2 Shadows and Colours

Shadows and colour changes are deployed to enhance usability by giving visual clues to the user. The shadows help highlight the button upon hover, showcasing clearly the area of interaction. Furthermore, the change in colour upon hovering or pressing down on the button achieves the same effect as the shadows. The buttons obtaining a darker colour when active improves readability by highlighting their status.

5.3.2.3 Cursor

The cursor changing layout to "click" animation provides visual clues on the interactive functionality of the button. Moreover, the cursor remaining in a neutral form on active navigation bar buttons helps indicate to the user what focus view they are on through visual communication.

5.4 Reflections on Implementation Strategies

The following section provides reflections regarding project management.

5.4.1 Conducting a Vision-based Project

Part of the difficulties faced within the project might have been caused by the nature of the project itself. Generally speaking, there are three types of projects. Some come with an instruction manual describing

each and every step of the work that needs to be done. The second type is driven by a specific goal in mind that should be accomplished within a certain time frame. The third and last type is more exploring in nature, where the project aim is stated in broad terms whether something can be accomplished or not.
This project have been a mix of the second and third type. There was a specific goal in mind, to stream low-latency high-quality video over the 5G network from the MARV. However, actually implementing the solution proved to be more challenging since the team opted for a solution based on new technologies no one had a lot of previous experience with. Taking this into account, the project therefore turned out to be more of a vision-based project, meaning there was no certainty whether the implemented solution would work or not.

5.4.2 Being a Self-managed Team

The team decided on choosing no team leader in the beginning of the project. The reason behind this decision was that having no designated project leader would help distribute an even workload between the participants. Three teams were initiated for hardware, software and UI development respectively. When the hardware components were delayed and arrived later than anticipated, there was not a lot of work to do within hardware development. Since the project was initially more exploring in terms of possible solution implementations, a reorganization of team resources should have been made once it became clear the hardware development was delayed. A team leader with insights of how the entire project development was going might have been able to identify this issue at an earlier stage and reorganize project participants and resources accordingly.

5.4.3 Underestimating the Simple

The team decided to build a web-based application with back-end developed in Python. Because of time constraints towards the end of the project, this seemed to be the easiest way to implement the pipeline design through an application with integrated UI design based on HTML and CSS. Since Python is one of the most common programming languages, implementing the pipeline design in Python would facilitate for future MARV-related projects to use what this project managed

to produce. However, the pipeline framework GStreamer is usually implemented in C.
Therefore the team had to rely on plugins and modules aiming to integrate GStreamer functionality in Python. The problem was that because of the multiple promising tools that seemed to do what needed to be done, the team didn't consider the possible outcome of what would happen if none proved to be working for the specific environment of the project. A complete analysis of how the different plugins and modules worked under the hood was not made, meaning the task of integrating the pipeline in Python was underestimated. Looking back, opting for a desktop-based application with back-end programmed in C right away would have saved a lot of time.

5.5 Ethical Considerations

In this section ethical considerations regarding the implemented solution of the project is discussed.

5.5.1 Impacts on the 5G Network

Two project objectives that the current solution failed to achieve were the development of an algorithm for prioritizing data streams and the implementation of adaptive resolution (1.3). Without these in place, the MARV will transmit an unnecessary amount of data while competing with other devices on the same network. This could potentially lead to disruptions or slow down data transmission for all connected devices. Additionally, the absence of adaptive resolution further limits the current solution's ability to adjust the video quality based on available bandwidth. This could severely impact the solution's live streaming capabilities. A suggestion for further improvement of the system is therefore to implement adaptive resolution as well as enabling prioritization of data streams.

5.5.2 Technology with Potential Military Applications

As mentioned in Section 2.10, there have been reported cases of autonomous water crafts being used in warfare. As a result, there is a security concern regarding the potential theft of the MARV with the

intention of it being utilized for acts of terrorism. Security measures have therefore been taken related to the storage of the MARV.

An objective of this project was to develop a secure software system. A lot of progress has been made by choosing SRT with built-in mechanisms for data encryption while streaming video. In order for the built-in encryption mechanisms to work, an SRT password has to be configured (2.5.2). Due to time constraints, the implemented solution does not include a password. Future projects related to the MARV should focus their efforts to solve this issue as soon as possible.

5.6 Suggestions for Further Development

There is a myriad of possible future implementations concerning the MARV. In this section, a handful of those who are the most closely connected to this project are mentioned.

5.6.1 Integrate UI design and Remote Control Capabilities

Currently the solution is implemented in a desktop application, executed in a command terminal. It does not support any further user input, which is why the UI design developed within this project needs to be implemented in a future application. With an implemented UI design, the remote control capabilities of the MARV can be integrated alongside the video streaming functionalities in the same application.

5.6.2 Install Air Compressor System for Camera Housing

The current camera enclosure supports an external mechanism to get rid of water stains and grime on the lens. In order for it to work an air compressor needs to be installed on the MARV.

5.6.3 Acquire Remaining Cameras

The team decided together with the project supervisor that one camera would be sufficient in order to test if the implemented solution using SRT works. The remaining cameras needs to be ordered and installed in upcoming projects to obtain a 360° FOV.

5.6.4 Build Reliable Camera Streaming Infrastructure

Some of this project's objectives were not accomplished. A suggestion for further development is therefore to start working on an algorithm for prioritizing data streams alongside the ability to adapt resolution of the video. In addition to this, an SRT password should be set-up for the current solution to allow AES encryption (2.5.2).

5.6.5 Further Testing The UI on The Target Audience

It is recommended to further test the UI on the target audience to ensure that the UI is intuitive and easy to use and to potentially discover faults that can be further improved.

6
Conclusion

In conclusion, the hardware development of the project was mostly successful, thanks to the heavy involvement by the project supervisor. As for the software developed, the current solution shows promise in regards to both streaming and compression, however it needs more testing and optimizations to be marked as complete. There is also clear support for development of bandwidth-optimizing tools. The UI was not tested by a large sample size but it follows convention so it should be satisfactory.

References

[1] H. Björk, L. Johansson, M. Lamm, A. Logren, and J. Osterman, *Säker fjärrstyrning via 5g för marv - marine autonomous research vehicle - undersökning och implementering av nätverksbaserad fjärrstyrning av en vattenskoter*, https://odr.chalmers.se/items/891d0fca-3976-467b-972c-046568de6869, (Accessed: Jan. 29, 2024), 2022.

[2] C. Kreitzberg, *The intuitive interface*, https://ux.princeton.edu/learn-ux/blog/intuitive-interface, Accessed: 2024-02-20, Feb. 2017.

[3] I. E. Commission, *Ip ratings*, https://www.iec.ch/ip-ratings, Accessed: 2024-04-04.

[4] N. Danielsson and V. Lindström, "Development and verification of an autonomous personal watercraft testbed - a small marine craft testbed for making future research and testing within autonomous navigation in a coastal environment easily accessible," M.S. thesis, Chalmers University of Technology, 2020. [Online]. Available: https://hdl.handle.net/20.500.12380/304353.

[5] manuall.fi. "Oldcamera." (), [Online]. Available: https://manuall.fi/sandstrom-s3y3cam-verkkokamera/ (visited on Apr. 29, 2024).

[6] H. Björk, L. Johansson, M. Lamm, A. Logren, and J. Osterman, *Säker fjärrstyrning via 5g för marv - marine autonomous research vehicle - undersökning och implementering av nätverksbaserad fjärrstyrning av en vattenskoter*, 2022. [Online]. Available: https://hdl.handle.net/20.500.12380/304640.

[7] J. Tidwell, *Designing Interfaces: Patterns for Effective Interaction Design*. Beijing: O'Reilly, 2020.

[8] DIGG - Myndigheten för digital förvaltning, *Använd tillräcklig kontrast mellan text och bakgrund*, https://www.digg.se/webbriktlinjer/alla-webbriktlinjer/anvand-tillracklig-kontrast-mellan-text-och-bakgrund, Senast uppdaterad: 8 april 2024, DIGG - Myndigheten för digital förvaltning, Apr. 2024.

[9] W3C Web Accessibility Initiative (WAI), *Understanding sc 1.4.13: Content on hover or focus (level aa)*, https://www.w3.org/WAI/WCAG21/Understanding/content-on-hover-or-focus.html, Updated 20 June 2023, 2023.

[10] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 8th Global Edition. Pearson, 2020.

[11] "Ieee standard for ethernet," *IEEE Std 802.3-2022 (Revision of IEEE Std 802.3-2018)*, pp. 1–7025, 2022. DOI: 10.1109/IEEESTD.2022.9844436.

[12] S. Committee, "Specification for sfp (small formfactor pluggable) transceiver," *SFF Committee documentation*, 2001.

[13] L. E-Lins Technology Co. "H685 smart cute 5g router with ethernet." (), [Online]. Available: https://www.e-lins.com/EN/H685-5G-Router.html (visited on Apr. 30, 2024).
[14] "5G wireless access: an overview," Tech. Rep., Apr. 2020.
[15] P. Tech. "Magnus castell på ericsson: "tänk 'sandboxing', prototyping och fail fast, fix fast, learn fast"." (), [Online]. Available: https://peopletech.wiseit.se/magnus-castell-manager-of-expericom-new-business-concept-development-pa-ericsson-tank-sandboxing-prototyping-och-fail-fast-fix-fast-learn-fast/ (visited on Apr. 30, 2024).
[16] N. Blum, S. Lachapelle, and H. Alvestrand, "Webrtc: Real-time communication for the open web platform," *Communications of the ACM*, vol. 64, no. 8, pp. 50–54, 2021.
[17] J. Cui and Z. Lin, "Research and implementation of webrtc signaling via websocket-based for real-time multimedia communications," in *Proceedings of the 2015 5th International Conference on Computer Sciences and Automation Engineering*, Atlantis Press, 2016, pp. 374–380, ISBN: 978-94-6252-156-8. DOI: 10.2991/iccsae-15.2016.72. [Online]. Available: https://doi.org/10.2991/iccsae-15.2016.72.
[18] "A Technical Guide to Implementing Open-Source SRT," Tech. Rep., Oct. 2018.
[19] L. Nikols, *Srt: Everything you need to know about the secure reliable transport protocol*, Mar. 2023. [Online]. Available: https://www.haivision.com/blog/all/srt-everything-you-need-to-know-about-the-secure-reliable-transport-protocol/.
[20] National Institute of Standards and Technology, *National Institute of Standards and Technology: Block Cipher Techniques*, Computer Security Resource Center (CSRC) website, Accessed: 2024-03-15, May 2017. [Online]. Available: https://csrc.nist.gov/projects/block-cipher-techniques.
[21] L. Nikols, *How to configure srt settings on your video encoder for optimal performance*, Jan. 2022. [Online]. Available: https://www.haivision.com/blog/all/how-to-configure-srt-settings-video-encoder-optimal-performance/.
[22] S. J. Vaughan-Nichols, "Will html 5 restandardize the web?" *Computer*, vol. 43, no. 4, pp. 13–15, 2010. DOI: 10.1109/MC.2010.119.
[23] J. K. Nurminen, A. J. Meyn, E. Jalonen, Y. Raivio, and R. Garcıa Marrero, "P2p media streaming with html5 and webrtc," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2013, pp. 63–64. DOI: 10.1109/INFCOMW.2013.6970739.
[24] GStreamer. "What is gstreamer." (), [Online]. Available: https://gstreamer.freedesktop.org/documentation/application-development/introduction/gstreamer.html?gi-language=c (visited on Apr. 29, 2024).
[25] FFmpeg. "About ffmpeg." (), [Online]. Available: https://ffmpeg.org/about.html (visited on Apr. 29, 2024).
[26] C. U. Press, *Cambridge dictionary*. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/encoding (visited on May 10, 2024).

[27] M. A. Layek, N. Q. Thai, M. A. Hossain, *et al.*, "Performance analysis of h.264, h.265, vp9 and av1 video encoders," in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2017, pp. 322–325. DOI: 10.1109/APNOMS.2017.8094162.

[28] Nvidia, *Video encode and decode gpu support matrix*. [Online]. Available: https://developer.nvidia.com/video-encode-and-decode-gpu-support-matrix-new (visited on May 10, 2024).

[29] B. G. H. Atul Puri Robert L. Schmidt, "Overview of the mpeg standard," in *Multimedia Systems, Standards, and Networks*, A. Puri, Ed., Dekker, 2000, pp. 87–88, 93. [Online]. Available: https://books.google.se/books?hl=en&lr=&id=PBH4jUto1HAC&oi=fnd&pg=PA87&dq=related:qgVwYLLCSdAJ:scholar.google.com/&ots=FLoIEMbuDf&sig=EPsojRuuwU5QiEJkTJ12yOWgFTQ&redir_esc=y#v=onepage&q&f=false.

[30] R. Sparrow, ""just say no" to drones," *IEEE Technology and Society Magazine*, vol. 31, no. 1, pp. 56–63, 2012. DOI: 10.1109/MTS.2012.2185275.

[31] K. Airaksinen, *Teorin: Obemannade vattenskotrar bakom attacken mot bron till krim*, https://www.svt.se/nyheter/utrikes/ukrainas-nya-vapen-vattenskoter, (Accessed: Feb. 06, 2024), Jul. 2023.

[32] B. AG, *A2a1920-51gcpro - camera specifications*, https://docs.baslerweb.com/a2a1920-51gcpro, Accessed: 2024-04-04.

[33] Theia Technologies. "Ultra-wide no-distortion lens my23f." (), [Online]. Available: https://www.theiatech.com/lenses/ultra-wide-no-distortion-lens-my23f/?lens=MY23F (visited on May 9, 2024).

[34] SRTLab. "Ffmpeg." (), [Online]. Available: https://srtlab.github.io/srt-cookbook/apps/ffmpeg.html#ffmpeg-example-with-smpte-bars-test-video-source (visited on Apr. 30, 2024).

[35] MidOpt. "Pr032 infrared (ir) longpass filter." (), [Online]. Available: https://midopt.com/filters/pr032/ (visited on May 9, 2024).

[36] Haivision, *Building srt on linux*, https://github.com/Haivision/srt/blob/master/docs/build/build-linux.md, commit: 04b7c00, 2023.

[37] Haivision, *Building srt on windows*, https://github.com/Haivision/srt/blob/master/docs/build/build-win.md, commit: 32d8382, 2023.

[38] Microsoft, *Quick start: Windows*, https://github.com/microsoft/vcpkg?tab=readme-ov-file#quick-start-windows, commit: 1921ac8, 2023.

[39] GStreamer, *Download gstreamer*, https://gstreamer.freedesktop.org/download/, version: 1.24.1.

A

Testday at Lindholmen

A.1 Route and speed of the MARV

Note that the map in Figure A.1 is old, the land structure pictured in the middle to the right does not exist. In the same figure, red color indicates high speed while blue indicates low.

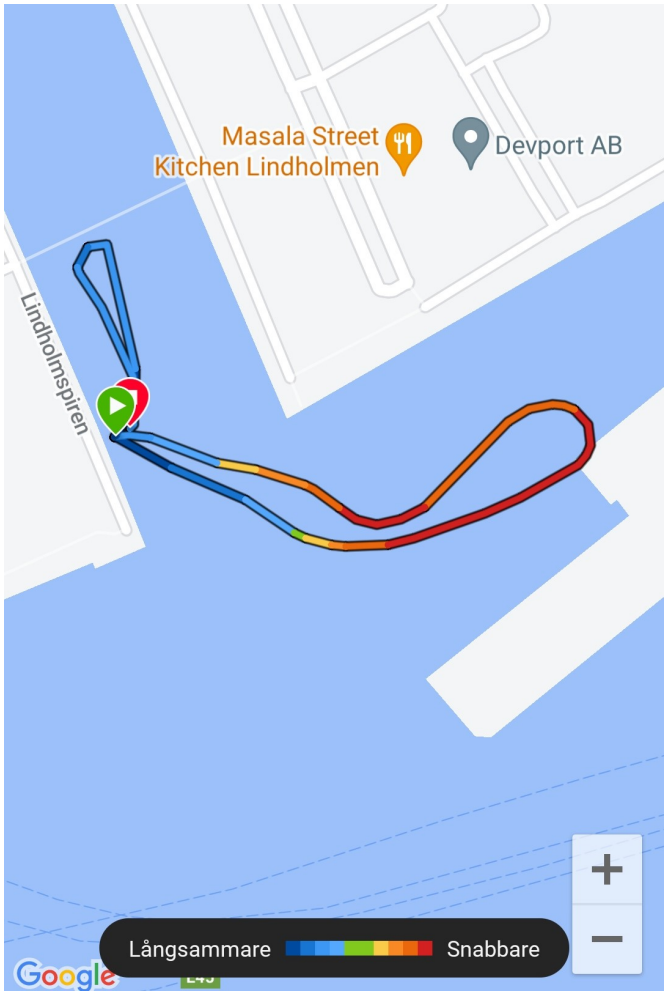


Figure A.1: The route of the MARV during the test run at Lindholmen

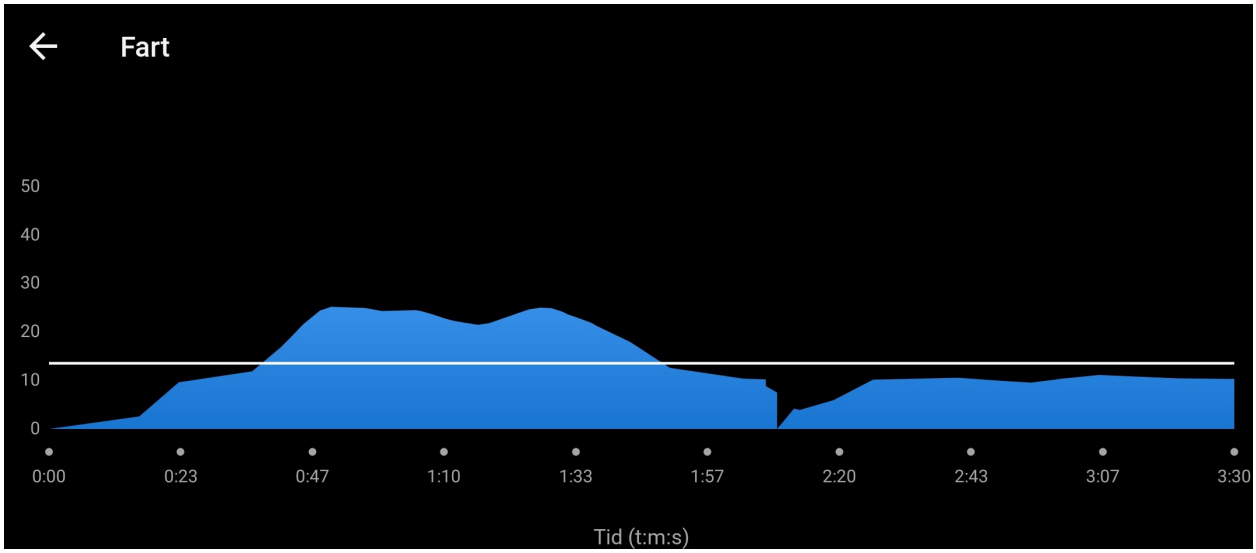


Figure A.2: Speed of the MARV during the test run at Lindholmen

B

Hardware design

B.1 Field of view calculation

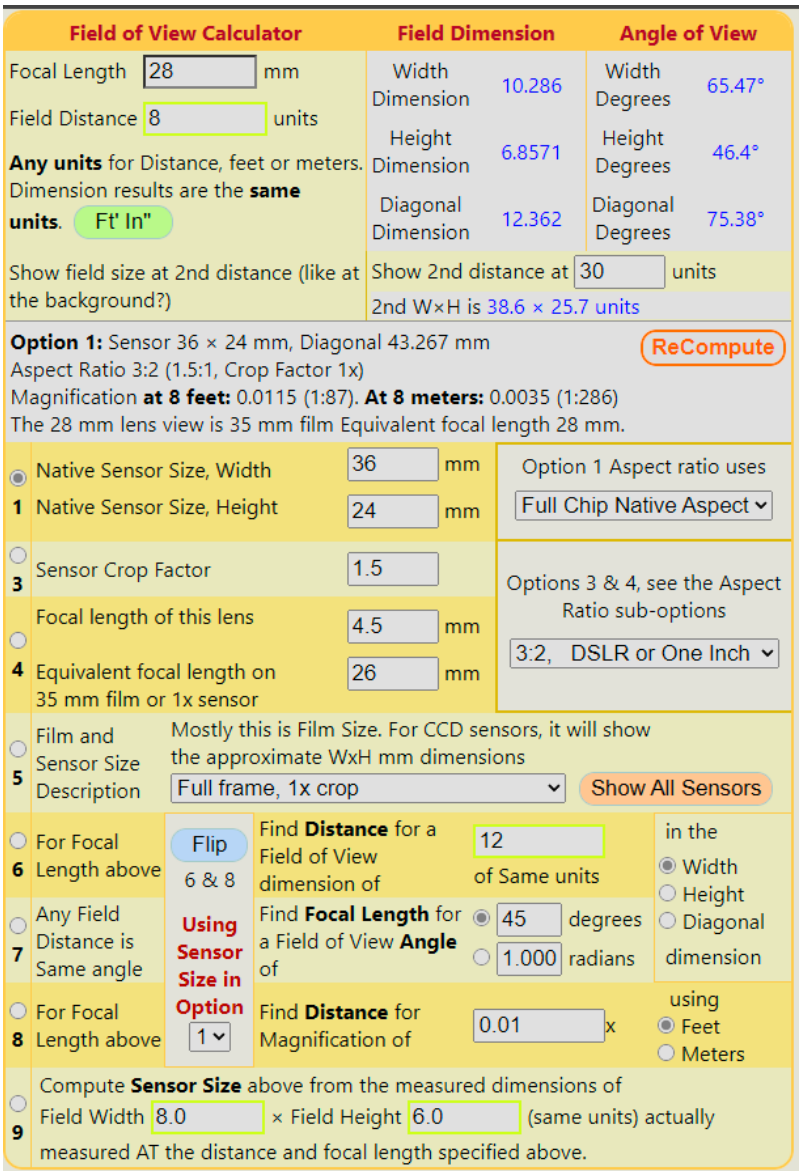


Figure B.1: Field of view calculation for a camera lens based on the parameters listed in Section 2.9

B.2 Brackets design for hardware components

B.2.1 DC-DC charger and router brackets

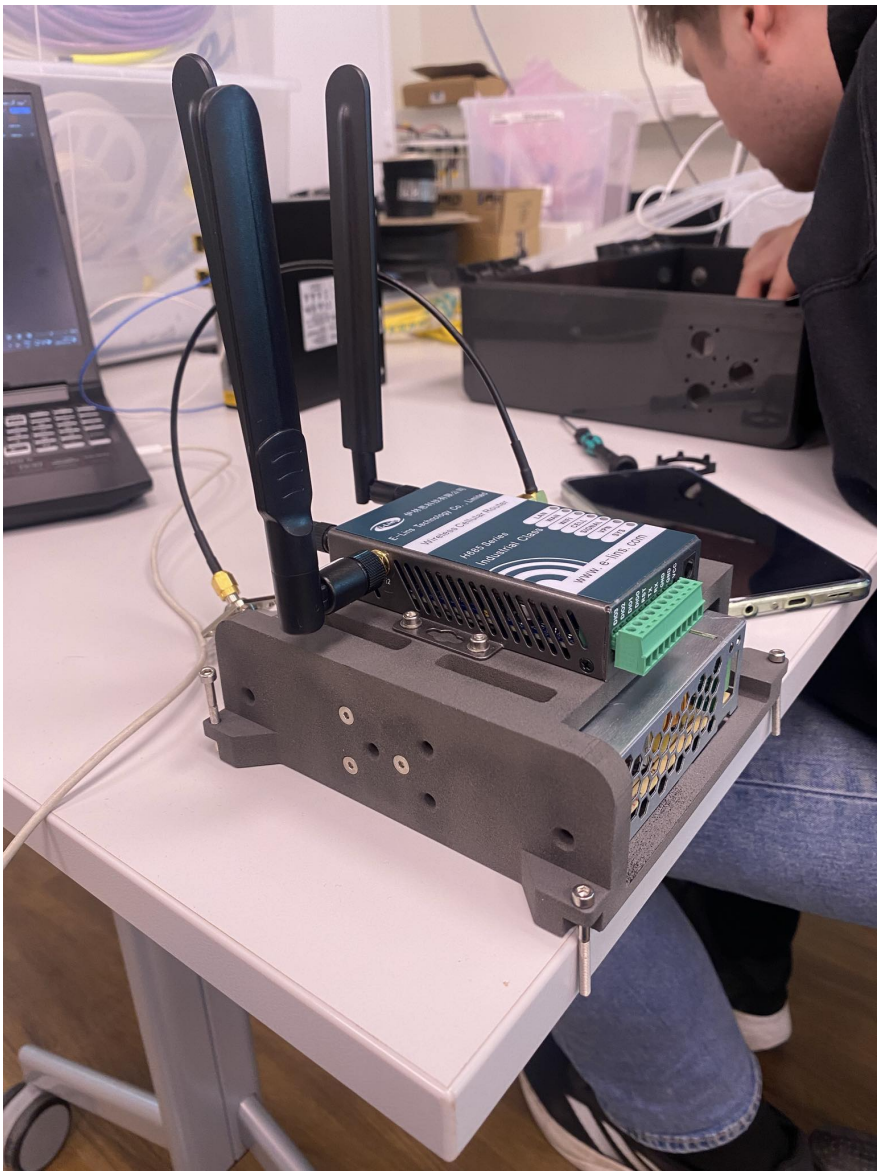


Figure B.2: The DC-DC charger and router bracket

B.2.2 The switch bracket

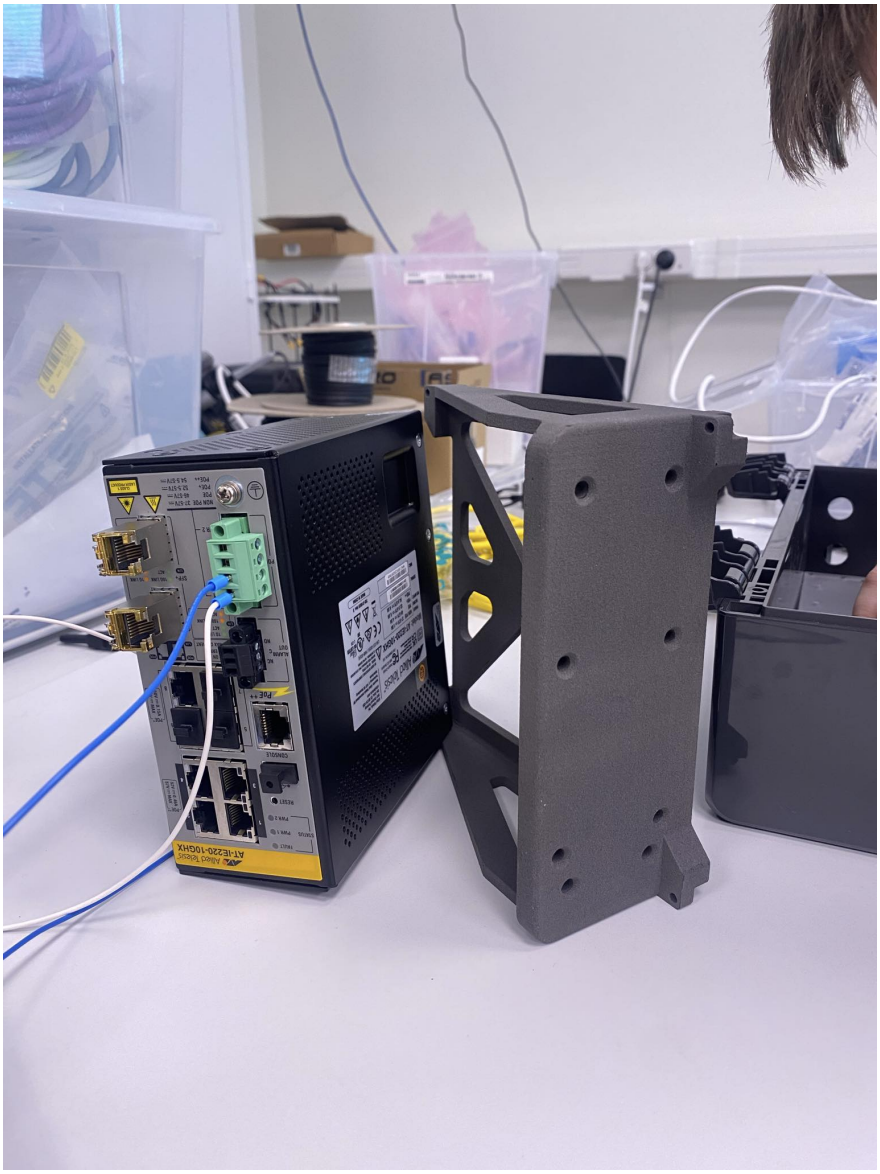


Figure B.3: The Switch bracket

B.2.3 The waterproof case and its components upside down

Figure B.4: The components within the case in an upside down position

B.3 Router antennas

Figure B.5: 4G/LTE and Wi-Fi antennas installed in the small white tubes

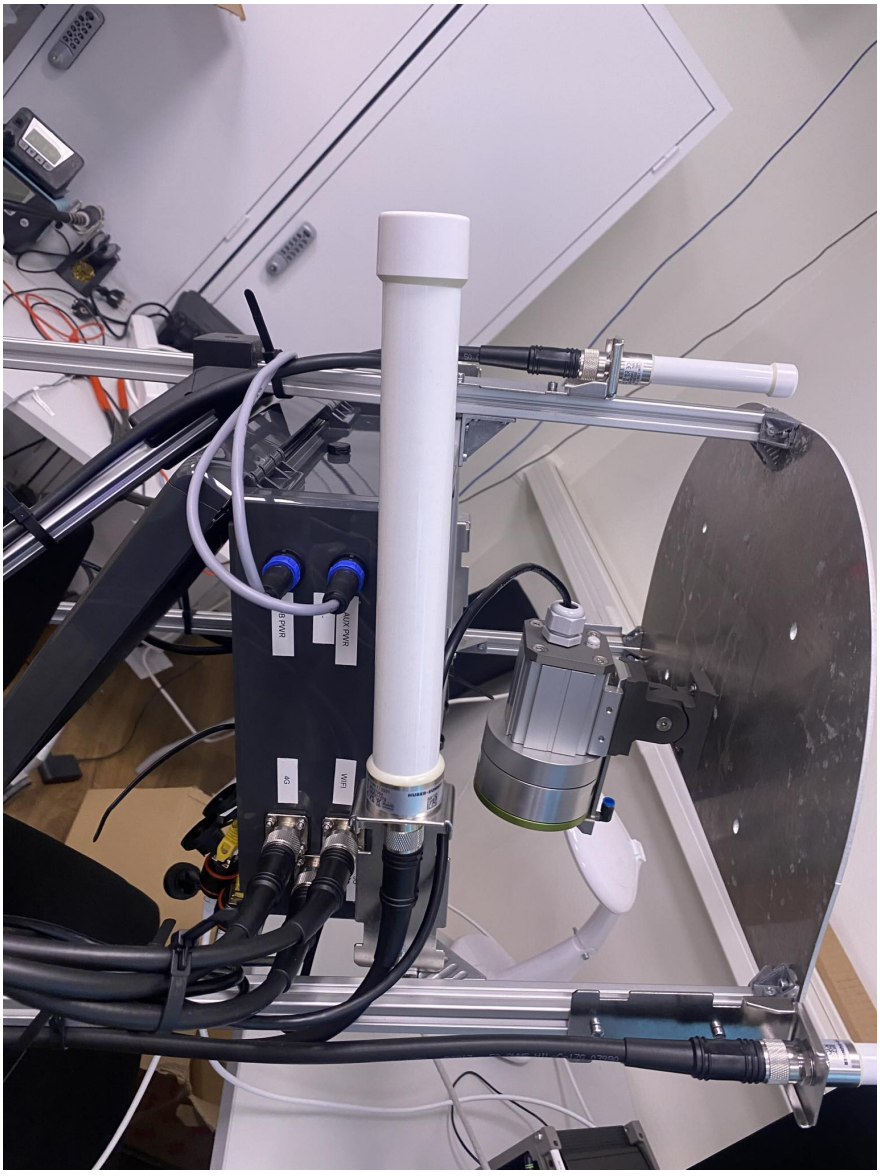


Figure B.6: 5G antennas are installed in the larger white tube

C
Software design

C.1 Implementation of pipelines

C.1.1 Sender

```c
#include <gst/gst.h>
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

pthread_t dot_thread;

//Structure to contain all our information, so we can pass it to
    callbacks
typedef struct _CustomData {
    GstElement* pipeline;
    GstElement* source;
    GstElement* nvidConv;
    GstElement* h265enc;
    GstElement* mpegtsmux;
    GstElement* sink;
} CustomData;

//Save srt stats to local file
void* print_stats(void* srt) {
    int time_step_s = 1;
    g_print("start collecting srt stats");
    FILE* fptr;
    // Create a file
    fptr = fopen("/srt_log.txt", "w");


    while (TRUE) {
        sleep(time_step_s);
        GstStructure* stats;
        g_object_get(srt, "stats", &stats, NULL);
        if (stats) {
            fprintf(fptr, "%s", gst_structure_to_string(stats));
        }
    }
    fclose(fptr);
```

```
36      pthread_exit(0);
37  }
38
39  int
40  main(int argc, char* argv[])
41  {
42      CustomData data;
43      GstBus* bus;
44      GstMessage* msg;
45      GstStateChangeReturn ret;
46      gboolean terminate = FALSE;
47
48      /* Initialize GStreamer */
49      gst_init(&argc, &argv);
50
51      /* Create the elements */
52      data.source = gst_element_factory_make("pylonsrc", "source");
53      data.nvidConv = gst_element_factory_make("nvvidconv", "nvidConv
        ");
54      data.h265enc = gst_element_factory_make("nvv4l2h265enc", "
        h265enc");
55      data.mpegtsmux = gst_element_factory_make("mpegtsmux", "
        mpegtsmux");
56      data.sink = gst_element_factory_make("srtsink", "sink");
57
58      /* Create the empty pipeline */
59      data.pipeline = gst_pipeline_new("send-pipeline");
60
61      //Check if all element is created corectly
62      if (!data.pipeline || !data.source || !data.nvidConv || !data.
        h265enc || !data.mpegtsmux || !data.sink) {
63          g_printerr("Not all elements could be created.\n");
64          return -1;
65      }
66
67      /* Build the pipeline */
68      gst_bin_add_many(GST_BIN(data.pipeline), data.source, data.
        nvidConv, data.h265enc, data.mpegtsmux, data.sink, NULL);
69
70      //Link elements except pylonsrc and nvvidconv
71      if (!gst_element_link_many(data.h265enc, data.mpegtsmux, data.
        sink, NULL)) {
72          g_printerr("Failed to link elements (mpegtsmux -> sink).\n"
        );
73          gst_object_unref(data.pipeline);
74          return -1;
75      }
76
77
78      //Adding caps for pylonsrc and nvvidconv and then link with the
         filter. This is to define framrate, format and resolution
79      GstCaps* caps1;
80      gboolean link_ok1;
81      caps1 = gst_caps_new_simple("video/x-raw",
82          "format", G_TYPE_STRING, "YUY2",
83          "width", G_TYPE_INT, 1920,
```

```
84          "height", G_TYPE_INT, 1200,
85          "framerate", GST_TYPE_FRACTION, 30, 1,
86          NULL);
87
88      link_ok1 = gst_element_link_filtered(data.source, data.nvidConv
    , caps1);
89      gst_caps_unref(caps1);
90
91
92      GstCaps* caps2;
93      gboolean link_ok2;
94      caps2 = gst_caps_new_simple("video/x-raw(memory:NVMM)",
95          "format", G_TYPE_STRING, "(string)NV12",
96          "width", G_TYPE_INT, 1920,
97          "height", G_TYPE_INT, 1200,
98          "framerate", GST_TYPE_FRACTION, 30, 1,
99          NULL);
100
101     link_ok2 = gst_element_link_filtered(data.nvidConv, data.
    h265enc, caps2);
102     gst_caps_unref(caps2);
103
104     //Check if linking was successful
105     if (!link_ok1 || !link_ok2) {
106         g_warning("Failed to link elements!");
107     }
108
109
110     /* Modify the source's properties */
111     g_object_set(data.sink, "uri", "srt://:7000", NULL);
112     g_object_set(data.sink, "latency", 125, NULL); //Define srt
    latency buffer
113     g_object_set(data.sink, "mode", 2, NULL); //Set srt to listener
     mode
114     g_object_set(data.sink, "sync", 0, NULL);
115     g_object_set(data.h265enc, "bitrate", 3000000, NULL);
116
117
118
119     /* Start playing */
120     ret = gst_element_set_state(data.pipeline, GST_STATE_PLAYING);
121     if (ret == GST_STATE_CHANGE_FAILURE) {
122         g_printerr("Unable to set the pipeline to the playing state
    .\n");
123         gst_object_unref(data.pipeline);
124         return -1;
125     }
126
127     //Start new thread for srt logging
128     pthread_create(&dot_thread, NULL, print_stats, data.sink);
129
130     /* Listen to the bus */
131     bus = gst_element_get_bus(data.pipeline);
132     do {
133         msg = gst_bus_timed_pop_filtered(bus, GST_CLOCK_TIME_NONE,
134             GST_MESSAGE_STATE_CHANGED | GST_MESSAGE_ERROR |
```

```
    GST_MESSAGE_EOS);
135
136         /* Parse message */
137         if (msg != NULL) {
138             GError* err;
139             gchar* debug_info;
140
141             switch (GST_MESSAGE_TYPE(msg)) {
142             case GST_MESSAGE_ERROR:
143                 gst_message_parse_error(msg, &err, &debug_info);
144                 g_printerr("Error received from element %s: %s\n",
    GST_OBJECT_NAME(msg->src), err->message);
145                 g_printerr("Debugging information: %s\n",
    debug_info ? debug_info : "none");
146                 g_clear_error(&err);
147                 g_free(debug_info);
148                 terminate = TRUE;
149                 break;
150             case GST_MESSAGE_EOS:
151                 g_print("End-Of-Stream reached.\n");
152                 terminate = TRUE;
153                 break;
154             case GST_MESSAGE_STATE_CHANGED:
155                 /* We are only interested in state-changed messages
     from the pipeline */
156                 if (GST_MESSAGE_SRC(msg) == GST_OBJECT(data.
    pipeline)) {
157                     GstState old_state, new_state, pending_state;
158                     gst_message_parse_state_changed(msg, &old_state
    , &new_state, &pending_state);
159                     g_print("Pipeline state changed from %s to %s:\
    n",
160                         gst_element_state_get_name(old_state),
    gst_element_state_get_name(new_state));
161                 }
162                 break;
163             default:
164                 g_printerr("Unexpected message received.\n");
165                 break;
166             }
167             gst_message_unref(msg);
168         }
169     } while (!terminate);
170
171
172
173     /* Free resources */
174     gst_object_unref(bus);
175     gst_element_set_state(data.pipeline, GST_STATE_NULL);
176     gst_object_unref(data.pipeline);
177     return 0;
178 }
```

C.1.2 Receiver

```
#include <gst/gst.h>
#include <Windows.h>
#include <stdio.h>

//Structure to contain all our information, so we can pass it to
    callbacks
    typedef struct _CustomData {
        GstElement* pipeline;
        GstElement* source;
        GstElement* tsparse;
        GstElement* tsdemux;
        GstElement* h265parse;
        GstElement* gpudecoding;
        GstElement* sink;
    } CustomData;

//Handler for the pad-added signal
static void pad_added_handler(GstElement* src, GstPad* pad,
    CustomData* data);

DWORD WINAPI print_stats(void* srt) {
    int time_step_ms = 1000;
    FILE* fptr;

    // Create a file
    fptr = fopen("/srt_log.txt", "w");

    while (TRUE) {
        Sleep(time_step_ms);
        GstStructure* stats;
        g_object_get(srt, "stats", &stats, NULL);
        if (stats) {
            fprintf(fptr, gst_structure_to_string(stats));
        }
    }
    fclose(fptr);
    return 0;
}

int
main(int argc, char* argv[])
{
    CustomData data;
    GstBus* bus;
    GstMessage* msg;
    GstStateChangeReturn ret;
    gboolean terminate = FALSE;

    /* Initialize GStreamer */
    gst_init(&argc, &argv);

```

```
50      /* Create the elements */
51      data.source = gst_element_factory_make("srtsrc", "source");
52      data.tsparse = gst_element_factory_make("tsparse", "mpegparse")
        ;
53      data.tsdemux = gst_element_factory_make("tsdemux", "demux");
54      data.h265parse = gst_element_factory_make("h265parse", "parse")
        ;
55      data.gpudecoding = gst_element_factory_make("d3d11h265dec", "
        decode");
56      data.sink = gst_element_factory_make("autovideosink", "sink");
57
58      /* Create the empty pipeline */
59      data.pipeline = gst_pipeline_new("recive-pipeline-main");
60
61
62      if (!data.pipeline || !data.source || !data.tsparse || !data.
        tsdemux || !data.h265parse || !data.gpudecoding || !data.sink) {
63          g_printerr("Not all elements could be created.\n");
64          return -1;
65      }
66      //data.tsdemux has unknown sink pads. It needs to be link
        during runtime.
67      /* Build the pipeline */
68      gst_bin_add_many(GST_BIN(data.pipeline), data.source, data.
        tsparse, data.tsdemux, data.h265parse, data.gpudecoding, data.
        sink, NULL);
69      if (!gst_element_link_many(data.h265parse, data.gpudecoding,
        data.sink, NULL)) {
70          g_printerr("Failed to link elements (h265parse ->
        gpudecoding -> sink).\n");
71          gst_object_unref(data.pipeline);
72          return -1;
73      }
74
75      if (!gst_element_link_many(data.source, data.tsparse, data.
        tsdemux, NULL)) {
76          g_printerr("Failed to link elements (source -> tsparse ->
        tsdemux).\n");
77          gst_object_unref(data.pipeline);
78          return -1;
79      }
80
81      /* Modify the source's properties */
82      g_object_set(data.source, "uri", "srt://10.81.180.3:7000", NULL
        );
83      g_object_set(data.source, "latency", 125, NULL);
84      g_object_set(data.source, "mode", 1, NULL); //This sets the
        srtsrc to caller mode
85      g_object_set(data.sink, "sync", 0, NULL);
86
87      /* Connect to the pad-added signal */
88      g_signal_connect(data.tsdemux, "pad-added", G_CALLBACK(
        pad_added_handler), &data);
89
90      /* Start playing */
91      ret = gst_element_set_state(data.pipeline, GST_STATE_PLAYING);
```

```
92      if (ret == GST_STATE_CHANGE_FAILURE) {
93          g_printerr("Unable to set the pipeline to the playing state
    .\n");
94          gst_object_unref(data.pipeline);
95          return -1;
96      }
97
98      //Start thread for logging srt stats
99      HANDLE thread = CreateThread(NULL, 0, print_stats, data.source,
        0, NULL);
100
101     /* Listen to the bus */
102     bus = gst_element_get_bus(data.pipeline);
103     do {
104         msg = gst_bus_timed_pop_filtered(bus, GST_CLOCK_TIME_NONE,
105             GST_MESSAGE_STATE_CHANGED | GST_MESSAGE_ERROR |
    GST_MESSAGE_EOS);
106
107         /* Parse message */
108         if (msg != NULL) {
109             GError* err;
110             gchar* debug_info;
111
112             switch (GST_MESSAGE_TYPE(msg)) {
113             case GST_MESSAGE_ERROR:
114                 gst_message_parse_error(msg, &err, &debug_info);
115                 g_printerr("Error received from element %s: %s\n",
    GST_OBJECT_NAME(msg->src), err->message);
116                 g_printerr("Debugging information: %s\n",
    debug_info ? debug_info : "none");
117                 g_clear_error(&err);
118                 g_free(debug_info);
119                 terminate = TRUE;
120                 break;
121             case GST_MESSAGE_EOS:
122                 g_print("End-Of-Stream reached.\n");
123                 terminate = TRUE;
124                 break;
125             case GST_MESSAGE_STATE_CHANGED:
126                 /* We are only interested in state-changed messages
     from the pipeline */
127                 if (GST_MESSAGE_SRC(msg) == GST_OBJECT(data.
    pipeline)) {
128                     GstState old_state, new_state, pending_state;
129                     gst_message_parse_state_changed(msg, &old_state
    , &new_state, &pending_state);
130                     g_print("Pipeline state changed from %s to %s:\
    n",
131                         gst_element_state_get_name(old_state),
    gst_element_state_get_name(new_state));
132                 }
133                 break;
134             default:
135                 g_printerr("Unexpected message received.\n");
136                 break;
137             }
```

```
138             gst_message_unref(msg);
139         }
140     } while (!terminate);
141
142
143
144     /* Free resources */
145     gst_object_unref(bus);
146     gst_element_set_state(data.pipeline, GST_STATE_NULL);
147     gst_object_unref(data.pipeline);
148     return 0;
149 }
150
151 /* This function will be called by the pad-added signal */
152 static void pad_added_handler(GstElement* src, GstPad* new_pad,
    CustomData* data) {
153
154     GstPad* sink_pad = gst_element_get_static_pad(data->h265parse,
    "sink");
155     GstPadLinkReturn ret;
156     GstCaps* new_pad_caps = NULL;
157     GstStructure* new_pad_struct = NULL;
158     const gchar* new_pad_type = NULL;
159
160     g_print("Received new pad '%s' from '%s':\n", GST_PAD_NAME(
    new_pad), GST_ELEMENT_NAME(src));
161
162     /* If our converter is already linked, we have nothing to do
    here */
163     if (gst_pad_is_linked(sink_pad)) {
164         g_print("We are already linked. Ignoring.\n");
165         goto exit;
166     }
167
168     /* Check the new pad's type */
169     new_pad_caps = gst_pad_get_current_caps(new_pad);
170     new_pad_struct = gst_caps_get_structure(new_pad_caps, 0);
171     new_pad_type = gst_structure_get_name(new_pad_struct);
172     if (!g_str_has_prefix(new_pad_type, "video/x-h265")) {
173         g_print("It has type '%s' which is not h265 video. Ignoring
    .\n", new_pad_type);
174         goto exit;
175     }
176
177     /* Attempt the link */
178     ret = gst_pad_link(new_pad, sink_pad);
179     if (GST_PAD_LINK_FAILED(ret)) {
180         g_print("Type is '%s' but link failed.\n", new_pad_type);
181     }
182     else {
183         g_print("Link succeeded (type '%s').\n", new_pad_type);
184     }
185
186 exit:
187     /* Unreference the new pad's caps, if we got them */
188     if (new_pad_caps != NULL)
```

```
189         gst_caps_unref(new_pad_caps);
190
191     /* Unreference the sink pad */
192     gst_object_unref(sink_pad);
193 }
```

C.2 Parser for SRT statistics

```
import tkinter as tk
from tkinter import ttk
import matplotlib.pyplot as plt


# Parse the data.
def parse_data():
    print("parsing")
    file_path = "/srt_log.txt"
    with open(file_path) as file:
        data = file.read()

    data = data.replace("\\", "")
    data = data.replace('"', "")
    data = data.replace("<", "")
    data = data.replace(">", "")

    entries = data.split(';')  # Split the data into individual entries
    parsed_data = {}
    for entry in entries:
        if not entry.strip():
            continue
        fields = entry.split(',')  # Split each entry into key-value pairs
        for field in fields:
            if field == "application/x-srt-statistics" or field == "":
                continue
            try:
                key_value = field.split('=')  # Split key and value
                key = key_value[0].strip()  # Remove whitespace around the key
                value = key_value[1].split('(')[-1].split(')')[1]  # Remove object type
            except IndexError:
                break

            # Convert value to appropriate type if possible
            try:
                if '.' in value:
                    value = float(value)
                else:
                    value = int(value)
            except ValueError:
                pass  # If conversion fails, keep the value as string

            if key in parsed_data:
                parsed_data[key].append(value)
```

```
45              else:
46                  parsed_data[key] = [value]
47      print("parsing done")
48      return parsed_data
49
50
51  def plot_stats(data, selected_stats):
52      time = range(
53          len(data[selected_stats[0]]))  # Assuming the length of one
        statistic's data is the same for all chosen stats
54
55      plt.figure(figsize=(8, 6))
56      # plot all selected stats
57      for stat in selected_stats:
58          plt.plot(time, data[stat], label=stat.strip())
59
60      plt.xlabel('Time (seconds)')
61      plt.ylabel('Value')
62      plt.title('Statistics Over Time')
63      plt.legend()
64      plt.show()
65
66
67  def on_plot_click():
68      selected_stats = [stat_name for stat_name, stat_var in zip(
       stat_names, stat_vars) if stat_var.get() == 1]
69      if selected_stats:
70          plot_stats(parsed_data, selected_stats)
71
72
73  def on_refresh_click():
74      global parsed_data
75      parsed_data = parse_data()
76      length = len(parsed_data["bytes-sent"])
77      packet_loss_rate = 0
78      for i in range(0, length-60):
79          bytes_sent_60s = parsed_data["bytes-sent"][i+60] -
       parsed_data["bytes-sent"][i]
80          print(bytes_sent_60s)
81          bytes_resent_60s = parsed_data["bytes-retransmitted"][i+60]
        - parsed_data["bytes-retransmitted"][i]
82          print(bytes_resent_60s)
83          tmp = bytes_resent_60s/bytes_sent_60s*100
84          if tmp > packet_loss_rate:
85              packet_loss_rate = tmp
86
87
88      print("Worstcase packet loss rate = " + str(packet_loss_rate) +
        "%")
89      rtt_mean = sum(parsed_data["rtt-ms"]) / len(parsed_data["rtt-ms
       "])
90      print("RTT mean: " + str(rtt_mean))
91      rtt_multiplier = int(input("Enter RTT Multiplier: "))
92      print("Srt latency buffer should be set to: " + str(
       rtt_multiplier*rtt_mean) + " ms")
93
```

```
94
95  # Load data
96
97  parsed_data = parse_data()
98  stat_names = list(parsed_data.keys())
99
100 # Create GUI
101 root = tk.Tk()
102 root.title("Plot Statistics")
103
104 main_frame = ttk.Frame(root, padding="10")
105 main_frame.grid(column=0, row=0, sticky=(tk.W, tk.E, tk.N, tk.S))
106
107 stat_vars = []
108 for idx, stat_name in enumerate(stat_names):
109     stat_var = tk.IntVar(value=0)
110     stat_vars.append(stat_var)
111     ttk.Checkbutton(main_frame, text=stat_name, variable=stat_var).
    grid(column=0, row=idx, padx=5, pady=5, sticky=tk.W)
112
113 plot_button = ttk.Button(main_frame, text="Plot", command=
    on_plot_click)
114 refresh_button = ttk.Button(main_frame, text="Refresh", command=
    on_refresh_click)
115 plot_button.grid(column=0, row=len(stat_names), pady=10)
116 refresh_button.grid(column=1, row=len(stat_names), pady=10)
117
118 root.mainloop()
```

C.3 Integration of UI design

The following code shows how a simple flask application was built to integrate HTML files.

```
from flask import Flask, render_template

app = Flask(__name__)

# navigate to http://localhost:5000 in browser

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/bow/')
def bow():
    return render_template('bow.html')

@app.route('/port/')
def port():
    return render_template('port.html')

@app.route('/starboard/')
def starboard():
    return render_template('starboard.html')

@app.route('/stern/')
def stern():
    return render_template('stern.html')


if __name__ == '__main__':
    app.run(debug=True)
```

C.3.1 Integration of SRT to WEB

```
import cv2
from flask import Flask, Response

app = Flask(__name__)

cap = cv2.VideoCapture("videotestsrc ! video/x-raw,width=500,height=400 ! videoconvert ! appsink", cv2.CAP_GSTREAMER)
#cap = cv2.VideoCapture("srtsrc uri=srt://192.168.1.95:20000 ! decodebin ! appsink", cv2.CAP_GSTREAMER)

def generate_frames():
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')


@app.route('/')
def index():
    return Response(generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')


if __name__ == '__main__':
    app.run(debug=True)
```

D

Final UI Layout

D.1 Application Screens

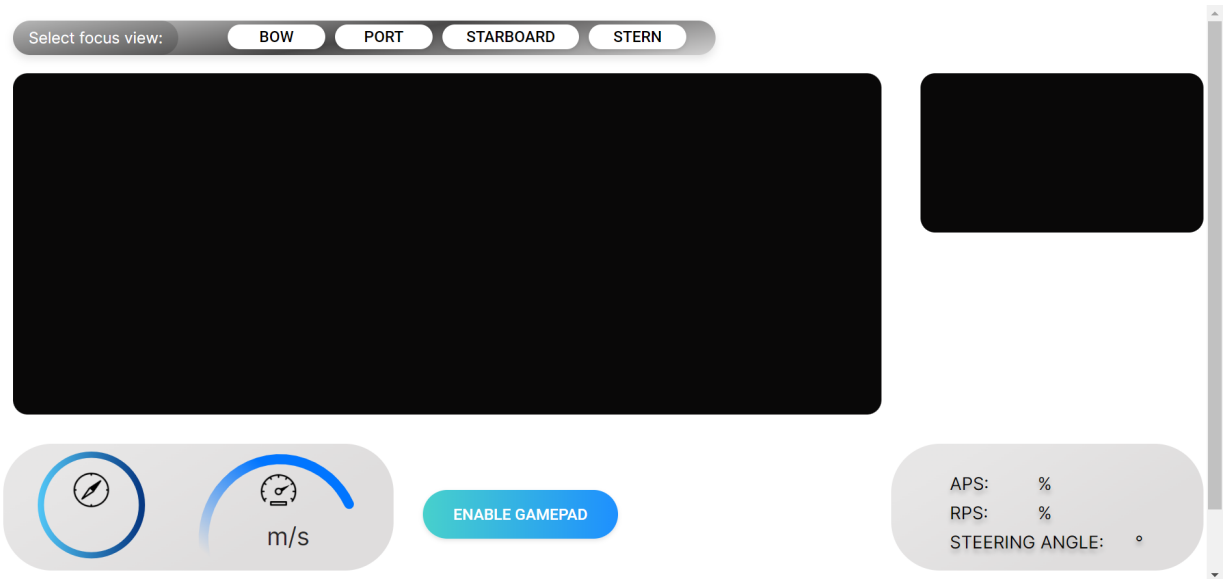


Figure D.1: The application screen for the Home view.

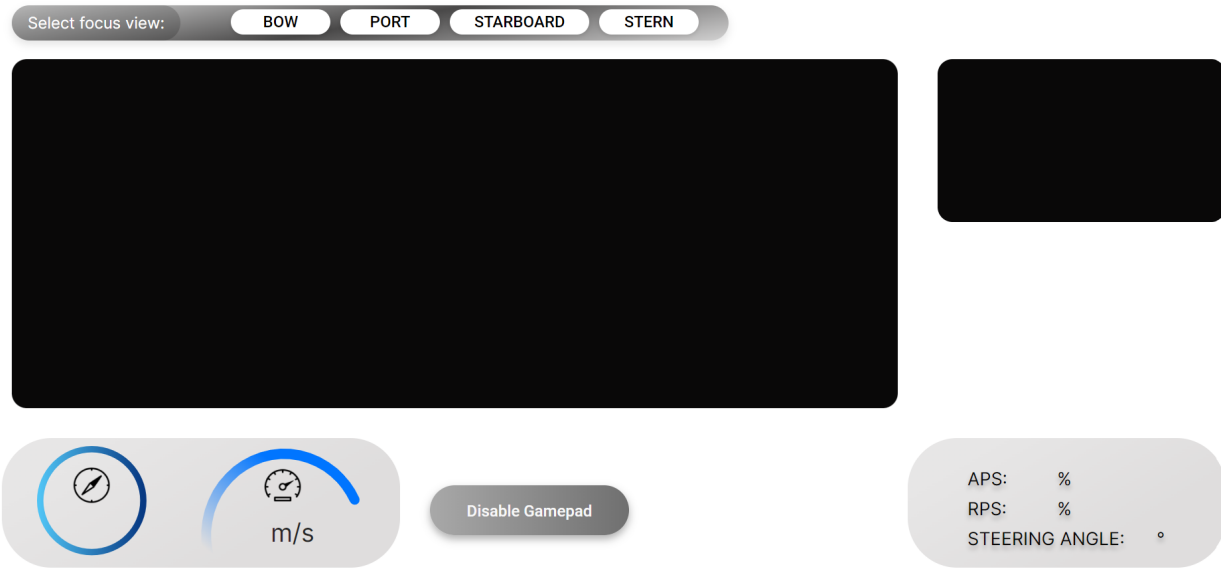


Figure D.2: The application screen for the Home view when the "ENABLE GAMEPAD" button is active, making the "Disable Gamepad" button visible. This functionality is available on all application screens.

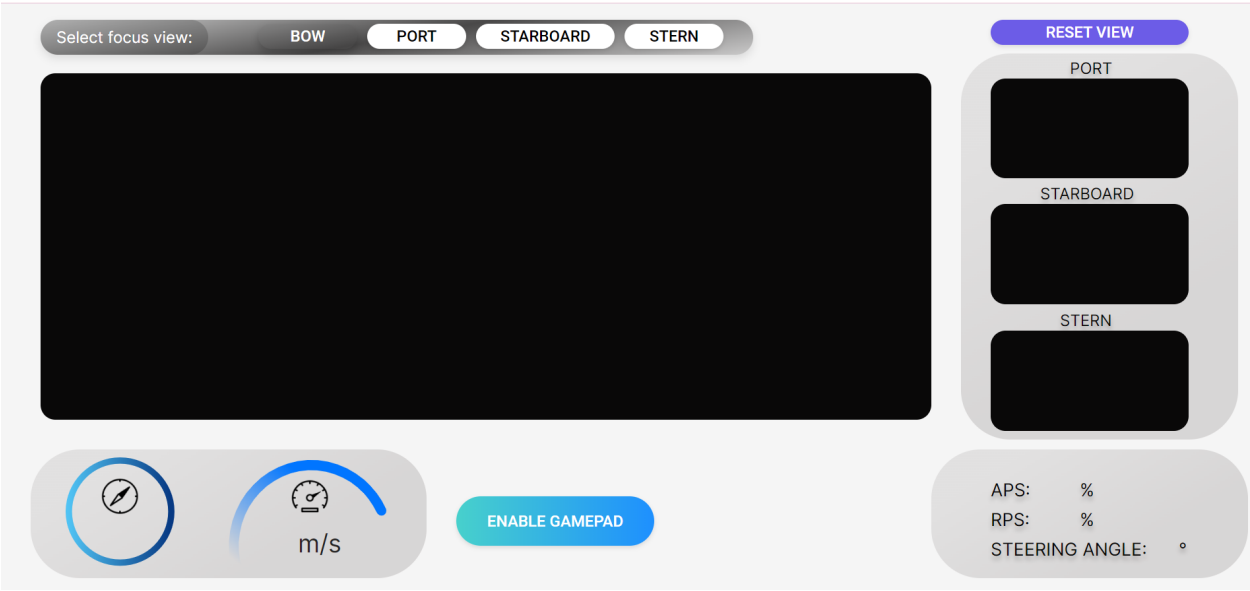


Figure D.3: The application screen for the Bow view.

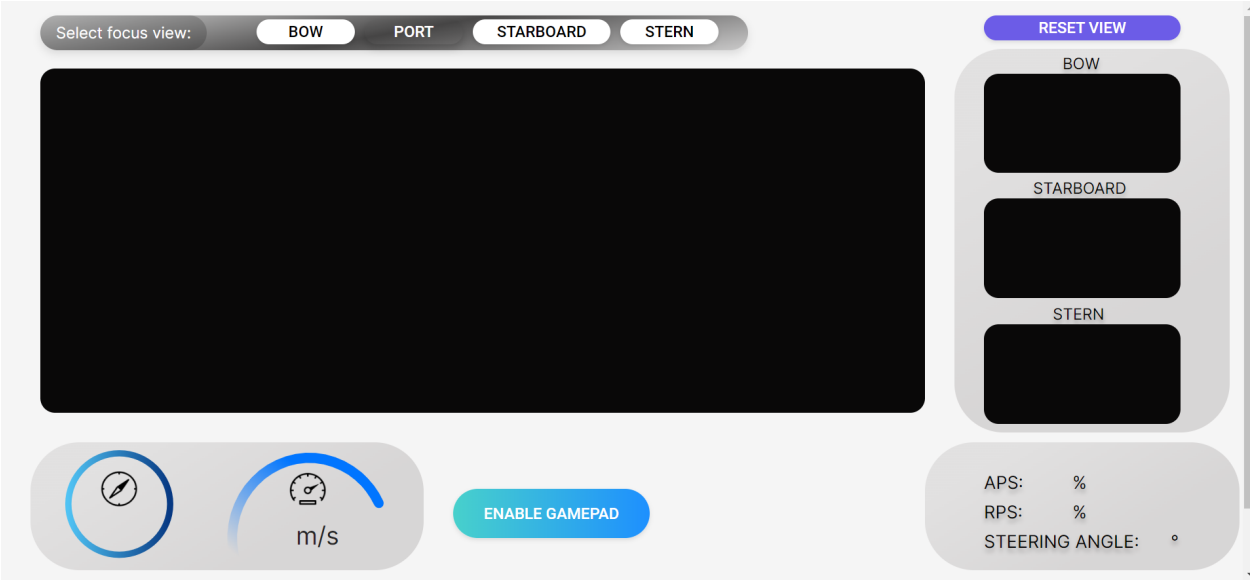


Figure D.4: The application screen for the Port view.

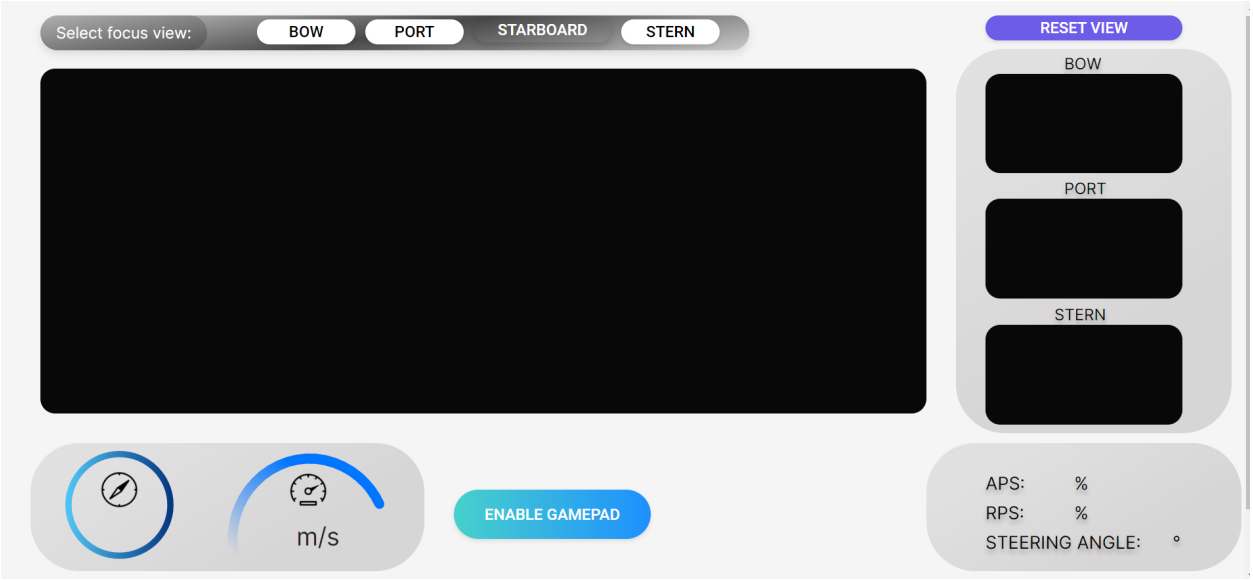


Figure D.5: The application screen for the Starboard view.

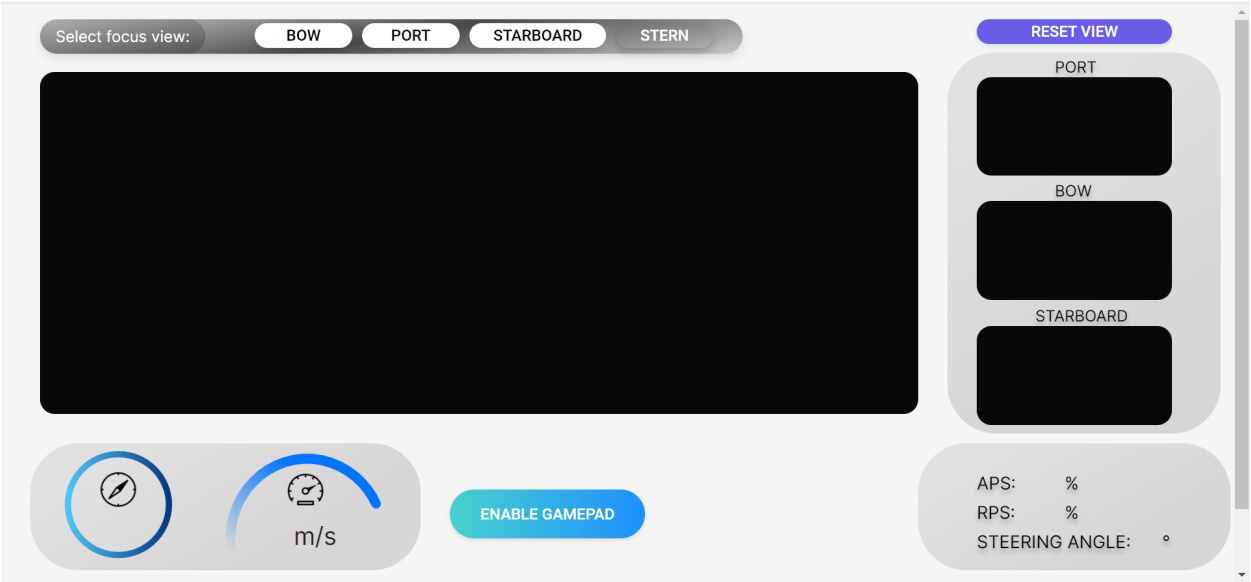


Figure D.6: The application screen for the Stern view.

D.2 Interactive Effects

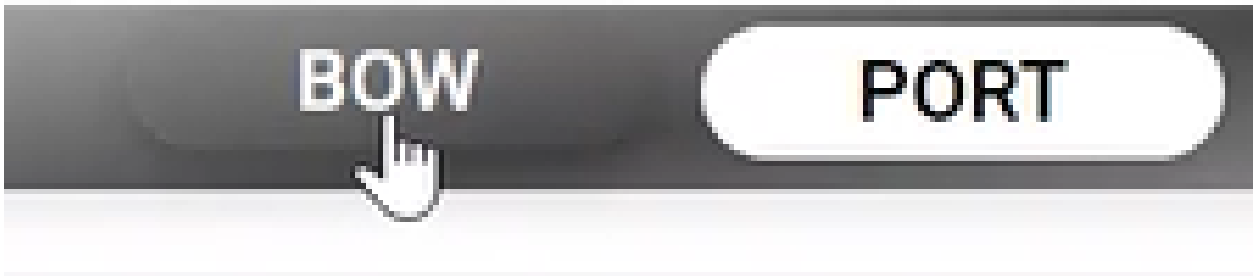


Figure D.7: The effects on the navigation button when the cursor is hovering on it, here showcased on the "BOW" button.

Figure D.8: The first part of the effects when pressing down on the navigation button, is displayed on the "BOW" button.

Figure D.9: The second part of the effects when pressing down on the navigation button, displayed on the "BOW" button.

Figure D.10: The interactive effects when the navigation button is active, demonstrated on the "BOW" button.

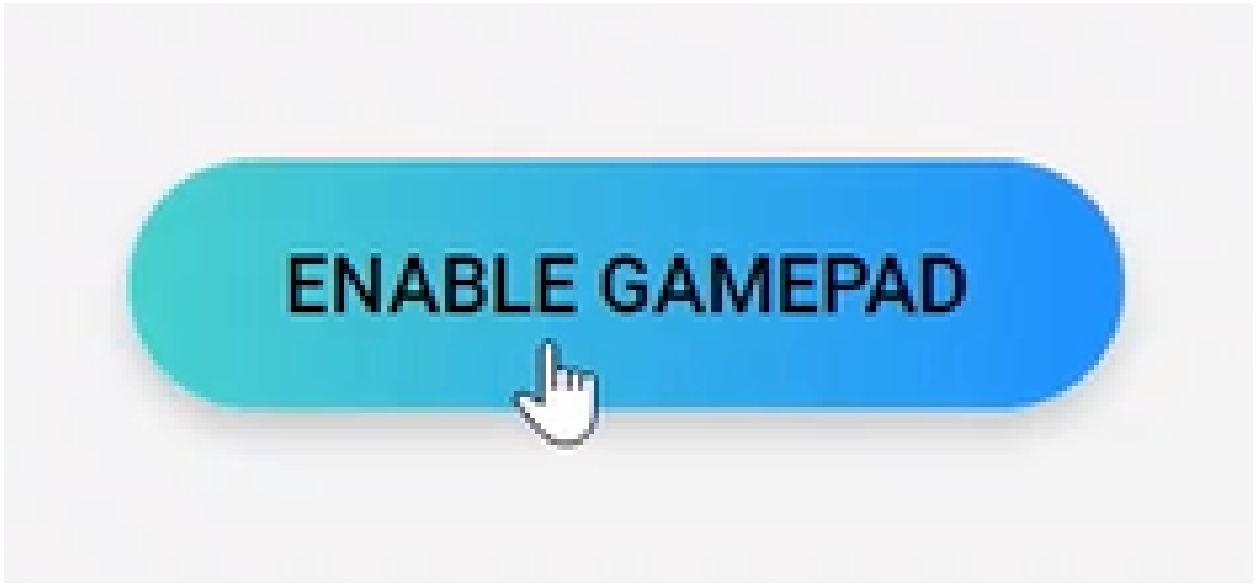


Figure D.11: The interactive effects when the cursor is hovering on the "ENABLE GAMEPAD" button.

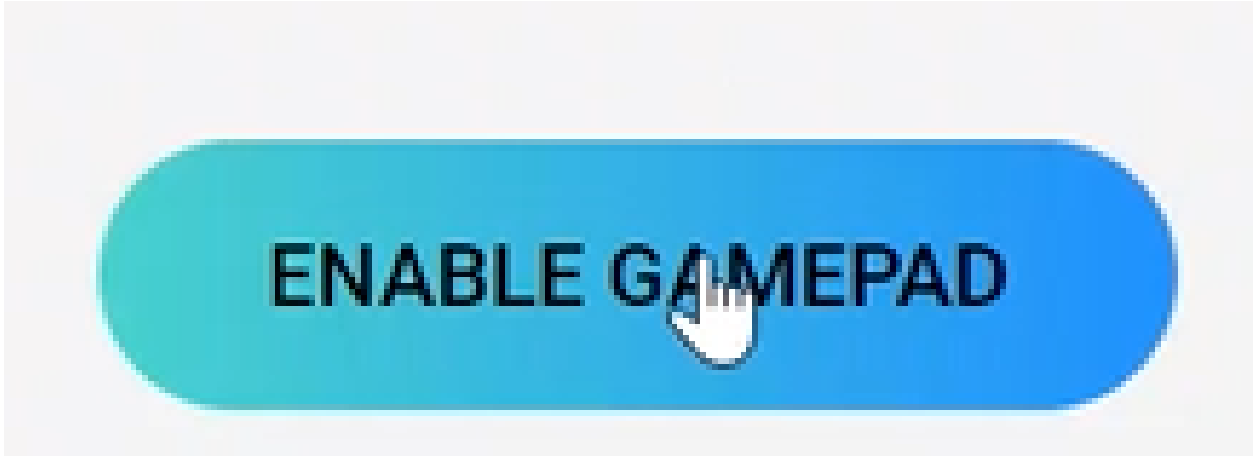


Figure D.12: The interactive effects when the cursor is pressing down on the "ENABLE GAMEPAD" button.

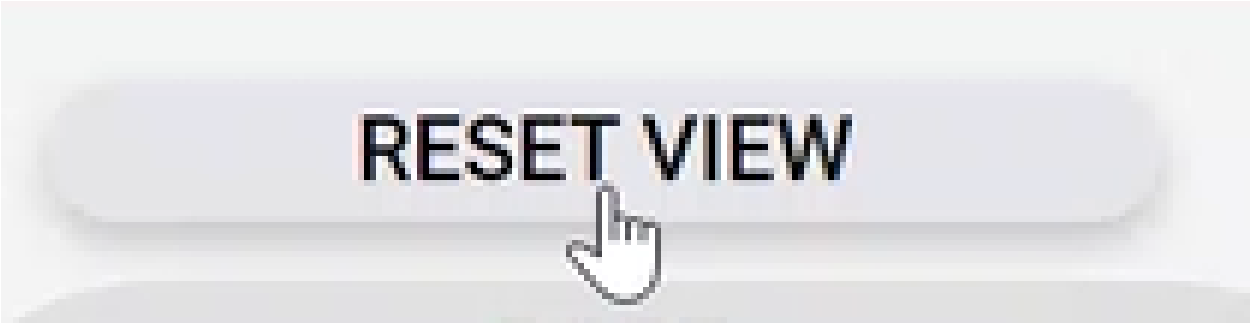


Figure D.13: The interactive effects when the cursor is hovering on the "RESET VIEW" button.

Figure D.14: The interactive effects when the cursor is pressing down on the "RESET VIEW" button.

Figure D.15: The interactive effects when the cursor is hoovering on the "Disable Gamepad" button.

Figure D.16: The interactive effects when the cursor is pressing down on the "Disable Gamepad" button.

E

Software prerequisites

E.1 SRT installation on Ubuntu

The SRT library was installed on the REACH by following the guidelines outlined by Haivision in their GitHub repository "Building SRT on Linux" [36]. The necessary prerequisites was to install CMake and OpenSSL-devel on the REACH. After the installation was complete, the following commands were executed in the command prompt.

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install tclsh pkg-config cmake libssl-dev build-
    essential
./configure
make
```

E.2 SRT installation on Windows

The SRT library was set up on a Windows computer by following the guidelines provided by Haivision in their GitHub repository "Building SRT on Windows" [37].

1. **SRT dependencies**
   The SRT build dependencies CMake, Git and Visual Studio were installed. Regarding external library dependencies, OpenSSL is used by SRT as default cryptographic library. It was installed using the library manager VCpkg. Finally, the Standard C++ thread library was used for threading by specifying the following CMake option.

   ```
   -DENABLE_STDCXX_SYNC=ON
   ```

2. **Install library manager VCpkg**
   The library manager VCpkg was installed by following the guidelines provided by Microsoft in their GitHub repository "vcpkg"

[38]. VCpkg was installed globally to avoid path issues. The installation was completed by executing the following command.

```
git clone https://github.com/microsoft/vcpkg
cd vcpkg
.\vcpkg\bootstrap-vcpkg.bat
```

Once installed, VCpkg was used in the remaining steps of the SRT installation process.

3. **Install OpenSSL**
   The command line tool OpenSSL was installed using VCpkg by executing the following commands in the command prompt.

```
.\vcpkg install openssl --triplet x64-windows
vcpkg integrate install
-DCMAKE_TOOLCHAIN_FILE=
%VCPKG_ROOT%\\scripts\\buildsystems\\vcpkg.cmake
```

4. **Clone SRT source code** Using a git client, the source code for SRT was retrieved from GitHub. Note the branch name was set to ”v1.5.3”, which was the current latest release when the repository was cloned.

```
git clone --branch v1.5.3 https://github.com/haivision/srt.git
    srt
```

5. **Build the SRT library**
   The current working directory was changed to ”srt/build” in the command prompt. A separate folder was created for the generated build files, separating them from the source files. This was done by executing the following in the command prompt.

```
cmake -S . -B build -DCMAKE_TOOLCHAIN_FILE=
%path to your  vcpkg
cmake --build build --config Release
```

After generating the build files, the SRT library was built by executing the following command.

```
cmake --build .
```

E.3 Verify successful SRT build

The test script ”srt-live-transmit.exe” was located among the SRT build files. The current working directory pathway was changed, corresponding to the pathway of the script. The following command was then executed in a command prompt.

```
.\srt-live-transmit.exe udp://:1234 srt://:4201 -v
```

If the SRT library was installed properly without any errors, the following output was generated.

```
Media path: 'udp://:1234' --> 'srt://:4201'
Opening SRT target listener on :4201
Binding a server on :4201 ...
 listen...
```

E.4 GStreamer installation

In order to stream video between two peers using a pipeline, the GStreamer framework for multimedia applications was utilized. GStreamer was installed on the REACH by executing the following command in a terminal.

```
apt-get install
libgstreamer1.0-dev
libgstreamer-plugins-base1.0-dev
libgstreamer-plugins-bad1.0-dev
gstreamer1.0-plugins-base
gstreamer1.0-plugins-good
gstreamer1.0-plugins-bad
gstreamer1.0-plugins-ugly
gstreamer1.0-libav
gstreamer1.0-tools
gstreamer1.0-x
gstreamer1.0-alsa
gstreamer1.0-gl
gstreamer1.0-gtk3
gstreamer1.0-qt5
gstreamer1.0-pulseaudio
```

On Windows, the MSVC 64-bit runtime installer (version 1.24.1) for Visual Studio 2019 CRT Release was installed from the GStreamer website [39].

E.5 GStreamer Visual Studio configuration

In order to implement a pipeline design in C, the IDE Visual Studio for Windows had to be configured properly. A new console app project was created in Visual Studio. Under ”Configuration Properties”, the following directories were added as ”Additional Include Directories” for ”C/C++”.

```
C:\gstreamer\1.0\msvc_x86_64\lib\glib-2.0\include
C:\gstreamer\1.0\msvc_x86_64\include\gstreamer-1.0
C:\gstreamer\1.0\msvc_x86_64\include\glib-2.0\
C:\gstreamer\1.0\msvc_x86_64\include\glib-2.0\glib
```

The following directory was added as "Additional Library Directories" under the "General" properties for "Linker".

```
C:\gstreamer\1.0\msvc_x86_64\lib
```

Finally, the following libraries were configured to be "Additional Dependencies" for the "Input" properties of the "Linker".

```
gobject-2.0.lib
glib-2.0.lib
gstreamer-1.0.lib
```

E.6 FFmpeg installation

A static build of FFmpeg for arm64 was downloaded from the FFmpeg website. After installation, the tar file was unpacked by executing the following in a command prompt.

```
tar -xf ./ffmpeg-release-arm64-static.tar.xz
```

F

Encoding latency test

```
GST_DEBUG="GST_TRACER:7" GST_TRACERS="latency(flags=element)" gst-launch-1.0 videotestsrc ! 'video/x-raw, width=1920, height=1080, framerate=30/1, format=YUY2'! nvvidconv ! 'video/x-raw, width=1920, height=1080, framerate=30/1, format=RGBA'! nvvidconv ! 'video/x-raw(memory:NVMM), width=1920, height=1080, framerate=30/1, format=(string)NV12' ! nvv4l2h265enc bitrate=2000000 ! h265parse ! queue ! h265parse ! nvv4l2decoder ! nv3dsink -e
0:00:00.505360768  3097 0xaaaad8a96b00 TRACE                 GST_TRACER :0:: element-latency, element-id=(string)0xaaaad8a84170, element=(string)capsfilter0, src=(string)src, time=(guint64)97152, ts=(guint64)505277504;
0:00:00.535413376  3097 0xaaaad8a96b00 TRACE                 GST_TRACER :0:: element-latency, element-id=(string)0xaaaad8a4fba0, element=(string)nvvconv0, src=(string)src, time=(guint64)29307712, ts=(guint64)534585216;
0:00:00.536187232  3097 0xaaaad8a96b00 TRACE                 GST_TRACER :0:: element-latency, element-id=(string)0xaaaad8a844b0, element=(string)capsfilter1, src=(string)src, time=(guint64)1421216, ts=(guint64)536006432;
0:00:00.553172608  3097 0xaaaad8a96b00 TRACE                 GST_TRACER :0:: element-latency, element-id=(string)0xaaaad8a50290, element=(string)nvvconv1, src=(string)src, time=(guint64)17041600, ts=(guint64)553048032;
0:00:00.553372544  3097 0xaaaad8a96b00 TRACE                 GST_TRACER :0:: element-latency, element-id=(string)0xaaaad8a847f0, element=(string)capsfilter2, src=(string)src, time=(guint64)226944, ts=(guint64)553274976;
NvVideo: H265 : Profile : 1
NVMEDIA: Need to set EMC bandwidth : 846000
NVMEDIA: Need to set EMC bandwidth : 846000
0:00:00.572429024  3097 0xaaaad8a96b00 TRACE                 GST_TRACER :0:: element-latency, element-id=(string)0xaaaad8a84170, element=(string)capsfilter0, src=(string)src, time=(guint64)140032, ts=(guint64)572295136;
0:00:00.583386368  3097 0xaaaad8a96d80 TRACE                 GST_TRACER :0:: element-latency, element-id=(string)0xaaaad8a59950, element=(string)nvv4l2h265enc0, src=(string)src, time=(guint64)30016896, ts=(guint64)583291872;
0:00:00.589594464  3097 0xaaaad8a96b00 TRACE                 GST_TRACER :0:: element-latency, element-id=(string)0xaaaad8a4fba0, element=(string)nvvconv0, src=(string)src, time=(guint64)17150208, ts=(guint64)589445344;
0:00:00.589766080  3097 0xaaaad8a96b00 TRACE                 GST_TRACER :0:: element-latency, element-id=(string)0xaaaad8a844b0,
```

```
    element=(string)capsfilter1, src=(string)src, time=(guint64)
    253376, ts=(guint64)589698720;
14 0:00:00.597458304  3097 0xaaaad8a96b00 TRACE             GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a50290,
    element=(string)nvvconv1, src=(string)src, time=(guint64)
    7610272, ts=(guint64)597308992;
15 0:00:00.597623968  3097 0xaaaad8a96b00 TRACE             GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a847f0,
    element=(string)capsfilter2, src=(string)src, time=(guint64)
    232160, ts=(guint64)597541152;
16 0:00:00.606963776  3097 0xaaaad8a96d80 TRACE             GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a59950,
    element=(string)nvv4l2h265enc0, src=(string)src, time=(guint64)
    9302208, ts=(guint64)606843360;
17 0:00:00.609491392  3097 0xaaaad8a96d80 TRACE             GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a5f7f0,
    element=(string)h265parse0, src=(string)src, time=(guint64)
    2574144, ts=(guint64)609417504;
18 0:00:00.610436672  3097 0xaaaad8a96920 TRACE             GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a6a110,
    element=(string)queue0, src=(string)src, time=(guint64)971936,
    ts=(guint64)610389440;
19 0:00:00.610763072  3097 0xaaaad8a96b00 TRACE             GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a84170,
    element=(string)capsfilter0, src=(string)src, time=(guint64)
    101376, ts=(guint64)610607488;
20 0:00:00.628545664  3097 0xaaaad8a96b00 TRACE             GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a4fba0,
    element=(string)nvvconv0, src=(string)src, time=(guint64)
    17771392, ts=(guint64)628378880;
21 0:00:00.628711168  3097 0xaaaad8a96b00 TRACE             GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a844b0,
    element=(string)capsfilter1, src=(string)src, time=(guint64)
    264512, ts=(guint64)628643392;
22 0:00:00.636832288  3097 0xaaaad8a96b00 TRACE             GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a50290,
    element=(string)nvvconv1, src=(string)src, time=(guint64)
    8023232, ts=(guint64)636666624;
23 0:00:00.636970752  3097 0xaaaad8a96b00 TRACE             GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a847f0,
    element=(string)capsfilter2, src=(string)src, time=(guint64)
    246528, ts=(guint64)636913152;
24 0:00:00.645408640  3097 0xaaaad8a96d80 TRACE             GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a59950,
    element=(string)nvv4l2h265enc0, src=(string)src, time=(guint64)
    8421856, ts=(guint64)645335008;
25 0:00:00.645878112  3097 0xaaaad8a96d80 TRACE             GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a5f7f0,
    element=(string)h265parse0, src=(string)src, time=(guint64)
    498176, ts=(guint64)645833184;
26 0:00:00.646118592  3097 0xaaaad8a96920 TRACE             GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a6a110,
    element=(string)queue0, src=(string)src, time=(guint64)222368,
    ts=(guint64)646055552;
27 0:00:00.649269856  3097 0xaaaad8a96b00 TRACE             GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a84170,
```

```
    element=(string)capsfilter0, src=(string)src, time=(guint64)
    114080, ts=(guint64)649171456;
NvMMLiteOpen : Block : BlockType = 279
NvMMLiteBlockCreate : Block : BlockType = 279
0:00:00.661232512  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a4fba0,
    element=(string)nvvconv0, src=(string)src, time=(guint64)
    11916128, ts=(guint64)661087584;
0:00:00.661432736  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a844b0,
    element=(string)capsfilter1, src=(string)src, time=(guint64)
    244160, ts=(guint64)661331744;
0:00:00.661901664  3097 0xaaaad8a96920 TRACE                 GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a69550,
    element=(string)h265parse1, src=(string)src, time=(guint64)
    15763328, ts=(guint64)661818880;
0:00:00.670579392  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a50290,
    element=(string)nvvconv1, src=(string)src, time=(guint64)
    9111040, ts=(guint64)670442784;
0:00:00.670884896  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a847f0,
    element=(string)capsfilter2, src=(string)src, time=(guint64)
    383744, ts=(guint64)670826528;
0:00:00.678990272  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a59950,
    element=(string)nvv4l2h265enc0, src=(string)src, time=(guint64)
    8098688, ts=(guint64)678925216;
0:00:00.679402624  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a5f7f0,
    element=(string)h265parse0, src=(string)src, time=(guint64)
    399488, ts=(guint64)679324704;
0:00:00.681769696  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a84170,
    element=(string)capsfilter0, src=(string)src, time=(guint64)
    111008, ts=(guint64)681689248;
0:00:00.693922656  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a4fba0,
    element=(string)nvvconv0, src=(string)src, time=(guint64)
    12085920, ts=(guint64)693775168;
0:00:00.694085824  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a844b0,
    element=(string)capsfilter1, src=(string)src, time=(guint64)
    244480, ts=(guint64)694019648;
0:00:00.702328960  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a50290,
    element=(string)nvvconv1, src=(string)src, time=(guint64)
    8187104, ts=(guint64)702206752;
0:00:00.702506912  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a847f0,
    element=(string)capsfilter2, src=(string)src, time=(guint64)
    248672, ts=(guint64)702455424;
0:00:00.710215072  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
     :0:: element-latency, element-id=(string)0xaaaad8a59950,
    element=(string)nvv4l2h265enc0, src=(string)src, time=(guint64)
    7664288, ts=(guint64)710119712;
```

```
43 0:00:00.710724320  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a5f7f0,
   element=(string)h265parse0, src=(string)src, time=(guint64)
   487712, ts=(guint64)710607424;
44 0:00:00.713341984  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a84170,
   element=(string)capsfilter0, src=(string)src, time=(guint64)
   111456, ts=(guint64)713264608;
45 0:00:00.725071232  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a4fba0,
   element=(string)nvvconv0, src=(string)src, time=(guint64)
   11624480, ts=(guint64)724889088;
46 0:00:00.725261920  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a844b0,
   element=(string)capsfilter1, src=(string)src, time=(guint64)
   268256, ts=(guint64)725157344;
47 0:00:00.733487392  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a50290,
   element=(string)nvvconv1, src=(string)src, time=(guint64)
   7559392, ts=(guint64)732716736;
48 0:00:00.733961568  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a847f0,
   element=(string)capsfilter2, src=(string)src, time=(guint64)
   1097056, ts=(guint64)733813792;
49 0:00:00.741818688  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a59950,
   element=(string)nvv4l2h265enc0, src=(string)src, time=(guint64)
   7914240, ts=(guint64)741728032;
50 0:00:00.742133824  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a5f7f0,
   element=(string)h265parse0, src=(string)src, time=(guint64)
   361728, ts=(guint64)742089760;
51 0:00:00.743831168  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a84170,
   element=(string)capsfilter0, src=(string)src, time=(guint64)
   135712, ts=(guint64)743666624;
52 0:00:00.755104736  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a4fba0,
   element=(string)nvvconv0, src=(string)src, time=(guint64)
   11293600, ts=(guint64)754960224;
53 0:00:00.755250688  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a844b0,
   element=(string)capsfilter1, src=(string)src, time=(guint64)
   222272, ts=(guint64)755182496;
54 0:00:00.762941152  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a50290,
   element=(string)nvvconv1, src=(string)src, time=(guint64)
   7648352, ts=(guint64)762830848;
55 0:00:00.763105632  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a847f0,
   element=(string)capsfilter2, src=(string)src, time=(guint64)
   219648, ts=(guint64)763050496;
56 0:00:00.772081056  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a59950,
   element=(string)nvv4l2h265enc0, src=(string)src, time=(guint64)
   8930016, ts=(guint64)771980512;
```

```
57 0:00:00.772435168  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a5f7f0,
   element=(string)h265parse0, src=(string)src, time=(guint64)
   408224, ts=(guint64)772388736;
58 0:00:00.772612320  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a84170,
   element=(string)capsfilter0, src=(string)src, time=(guint64)
   172256, ts=(guint64)772523200;
59 0:00:00.783842112  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a4fba0,
   element=(string)nvvconv0, src=(string)src, time=(guint64)
   11154560, ts=(guint64)783677760;
60 0:00:00.784020704  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a844b0,
   element=(string)capsfilter1, src=(string)src, time=(guint64)
   244128, ts=(guint64)783921888;
61 0:00:00.791277600  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a50290,
   element=(string)nvvconv1, src=(string)src, time=(guint64)
   7197824, ts=(guint64)791119712;
62 0:00:00.791415552  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a847f0,
   element=(string)capsfilter2, src=(string)src, time=(guint64)
   238240, ts=(guint64)791357952;
63 0:00:00.800367392  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a59950,
   element=(string)nvv4l2h265enc0, src=(string)src, time=(guint64)
   8942592, ts=(guint64)800300544;
64 0:00:00.800741472  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a5f7f0,
   element=(string)h265parse0, src=(string)src, time=(guint64)
   391840, ts=(guint64)800692384;
65 0:00:00.800786048  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a84170,
   element=(string)capsfilter0, src=(string)src, time=(guint64)
   122912, ts=(guint64)800651328;
66 0:00:00.812140768  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a4fba0,
   element=(string)nvvconv0, src=(string)src, time=(guint64)
   11341248, ts=(guint64)811992576;
67 0:00:00.812282336  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a844b0,
   element=(string)capsfilter1, src=(string)src, time=(guint64)
   226016, ts=(guint64)812218592;
68 0:00:00.819525600  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a50290,
   element=(string)nvvconv1, src=(string)src, time=(guint64)
   7161632, ts=(guint64)819380224;
69 0:00:00.819693120  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a847f0,
   element=(string)capsfilter2, src=(string)src, time=(guint64)
   221408, ts=(guint64)819601632;
70 0:00:00.827870848  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a59950,
   element=(string)nvv4l2h265enc0, src=(string)src, time=(guint64)
   8179360, ts=(guint64)827780992;
```

```
71 0:00:00.828168064  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a5f7f0,
   element=(string)h265parse0, src=(string)src, time=(guint64)
   342208, ts=(guint64)828123200;
72 0:00:00.828803648  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a84170,
   element=(string)capsfilter0, src=(string)src, time=(guint64)
   116448, ts=(guint64)828709664;
73 0:00:00.840097568  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a4fba0,
   element=(string)nvvconv0, src=(string)src, time=(guint64)
   11190432, ts=(guint64)839900096;
74 0:00:00.840241696  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a844b0,
   element=(string)capsfilter1, src=(string)src, time=(guint64)
   277056, ts=(guint64)840177152;
75 0:00:00.847462432  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a50290,
   element=(string)nvvconv1, src=(string)src, time=(guint64)
   7176640, ts=(guint64)847353792;
76 0:00:00.847585632  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a847f0,
   element=(string)capsfilter2, src=(string)src, time=(guint64)
   184352, ts=(guint64)847538144;
77 0:00:00.856493920  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a59950,
   element=(string)nvv4l2h265enc0, src=(string)src, time=(guint64)
   8866272, ts=(guint64)856404416;
78 0:00:00.856639584  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a84170,
   element=(string)capsfilter0, src=(string)src, time=(guint64)
   120384, ts=(guint64)856541920;
79 0:00:00.856890336  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a5f7f0,
   element=(string)h265parse0, src=(string)src, time=(guint64)
   437280, ts=(guint64)856841696;
80 0:00:00.867946176  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a4fba0,
   element=(string)nvvconv0, src=(string)src, time=(guint64)
   11204192, ts=(guint64)867746112;
81 0:00:00.868153120  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a844b0,
   element=(string)capsfilter1, src=(string)src, time=(guint64)
   337056, ts=(guint64)868083168;
82 0:00:00.875395776  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a50290,
   element=(string)nvvconv1, src=(string)src, time=(guint64)
   7190752, ts=(guint64)875273920;
83 0:00:00.875525152  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a847f0,
   element=(string)capsfilter2, src=(string)src, time=(guint64)
   201696, ts=(guint64)875475616;
84 0:00:00.878548192  3097 0xaaaad8a96920 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a6a110,
   element=(string)queue0, src=(string)src, time=(guint64)
   199172064, ts=(guint64)878496768;
```

```
85 0:00:00.880448800  3097 0xaaaad8a96920 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a69550,
   element=(string)h265parse1, src=(string)src, time=(guint64)
   1869504, ts=(guint64)880366272;
86 0:00:00.881176896  3097 0xaaaad8a96920 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a6a110,
   element=(string)queue0, src=(string)src, time=(guint64)
   170524224, ts=(guint64)881131648;
87 0:00:00.881803904  3097 0xaaaad8a96920 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a69550,
   element=(string)h265parse1, src=(string)src, time=(guint64)
   628672, ts=(guint64)881760320;
88 0:00:00.882481536  3097 0xaaaad8a96920 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a6a110,
   element=(string)queue0, src=(string)src, time=(guint64)
   140345504, ts=(guint64)882435264;
89 0:00:00.883182144  3097 0xaaaad8a96920 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a69550,
   element=(string)h265parse1, src=(string)src, time=(guint64)
   673248, ts=(guint64)883108512;
90 0:00:00.883766528  3097 0xaaaad8a96920 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a6a110,
   element=(string)queue0, src=(string)src, time=(guint64)
   111323840, ts=(guint64)883712576;
91 0:00:00.884787104  3097 0xaaaad8a96b00 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a84170,
   element=(string)capsfilter0, src=(string)src, time=(guint64)
   143968, ts=(guint64)884658912;
92 0:00:00.885077856  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a59950,
   element=(string)nvv4l2h265enc0, src=(string)src, time=(guint64)
   9535360, ts=(guint64)885010976;
93 0:00:00.885411328  3097 0xaaaad8a96d80 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a5f7f0,
   element=(string)h265parse0, src=(string)src, time=(guint64)
   352512, ts=(guint64)885363488;
94 0:00:00.885577024  3097 0xffff7c116060 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a75fe0,
   element=(string)nvv4l2decoder0, src=(string)src, time=(guint64)
   223629184, ts=(guint64)885448064;
95 0:00:00.886147104  3097 0xaaaad8a96920 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a69550,
   element=(string)h265parse1, src=(string)src, time=(guint64)
   2363104, ts=(guint64)886075680;
96 Pipeline is PREROLLED ...
97 Setting pipeline to PLAYING ...
98 0:00:00.887353984  3097 0xaaaad8a96920 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a6a110,
   element=(string)queue0, src=(string)src, time=(guint64)86600576,
    ts=(guint64)887292960;
99 0:00:00.887950144  3097 0xffff7c116060 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a75fe0,
   element=(string)nvv4l2decoder0, src=(string)src, time=(guint64)
   7517536, ts=(guint64)887883808;
100 0:00:00.887942976  3097 0xaaaad8a96920 TRACE                 GST_TRACER
       :0:: element-latency, element-id=(string)0xaaaad8a69550,
```

```
   element=(string)h265parse1, src=(string)src, time=(guint64)
   429248, ts=(guint64)887722208;
New clock: GstSystemClock
```

Listing F.1: Encoding

G

gst-launch-pipelines

```
 export LD_LIBRARY_PATH=/opt/pylon/lib/
Mest basic exempel
gst-launch-1.0 pylonsrc ! videoconvert ! autovideosink

Basler expempel:
gst-launch-1.0 pylonsrc ! "video/x-raw(memory:NVMM), width=1920,
    height=1080"  !  nvvidconv  ! "video/x-raw(memory:NVMM), width
    =1280, height=720" !  fakesink

Debugg basler exempel:
GST_DEBUG=2,pylonsrc:6 gst-launch-1.0 pylonsrc ! "video/x-raw(
    memory:NVMM), width=1920, height=1080"  !  nvvidconv  ! "video/x
    -raw(memory:NVMM), width=1280, height=720" !  fakesink

Stream grabb1:
gst-launch-1.0 -v pylonsrc width=1024 height=512 binningv=2
    binningh=2 ! "video/x-raw,format=GRAY8,framerate=60/1" !
    nvvidconv ! 'video/x-raw(memory:NVMM), format=(string)I420' !
    omxh264enc ! 'video/x-h264, stream-format=(string)byte-stream' !
     h264parse ! qtmux ! filesink location=test.mp4 -e

Nvidia h265 exempel
gst-launch-1.0 nvarguscamerasrc ! \
'video/x-raw(memory:NVMM), width=(int)1920, height=(int)1080, \
format=(string)NV12, framerate=(fraction)30/1' ! nvv4l2h265enc \
bitrate=8000000 ! h265parse ! qtmux ! filesink \
location=filename_h265.mp4 -e

Stream grabb2
gst-launch-1.0 pylonsrc width=1024 height=512 binningv=2 binningh=2
     ! nvvidconv ! nvv4l2h264enc ! h264parse ! matroskamux  !
    filesink location=test.mkv

Egen test
gst-launch-1.0 pylonsrc width=1024 height=512 ! 'video/x-raw(memory
    :NVMM), width=1024, height=512, framerate=10/1, format=UYVY'!
    nvvidconv! 'video/x-raw(memory:NVMM), width=1024, height=512,
    framerate=10/1, format=(string)NV12' ! nvv4l2h265enc bitrate
    =8000000 ! h265parse ! qtmux ! filesink location=test.mp4 -e

Egen test 2:
gst-launch-1.0 pylonsrc format=UYVY ! 'video/x-raw(memory:NVMM),
    width=1920, height=1080, framerate=10/1' ! nvvidconv !
    autovideosink
```

```
29
30 Egen test 3
31 gst-launch-1.0 pylonsrc width=1920 height=1080 ! 'video/x-raw(
       memory:NVMM), width=1920, height=1080, framerate=10/1, format=
       YUY2'! nvvidconv ! 'video/x-raw(memory:NVMM), width=1920, height
       =1080, framerate=10/1, format=(string)NV12' ! nvv4l2h265enc
       bitrate=8000000 ! nvv4l2decoder ! nvvidconv ! ximagesink
32
33 Egen test 4
34 gst-launch-1.0 pylonsrc width=1920 height=1080 ! 'video/x-raw(
       memory:NVMM), width=1920, height=1080, framerate=10/1, format=
       YUY2'! nvvidconv ! 'video/x-raw(memory:NVMM), width=1920, height
       =1080, framerate=10/1, format=(string)NV12' ! nvv4l2h265enc
       bitrate=2000000 ! h265parse ! qtmux ! filesink location=test.mp4
        -e
35
36 Egen test 5
37
38 gst-launch-1.0 pylonsrc ! 'video/x-raw(memory:NVMM), width=1920,
       height=1080, framerate=10/1'! nvvidconv ! 'video/x-raw(memory:
       NVMM)' ! nvv4l2h265enc bitrate=8000000 ! nvv4l2decoder !
       autovideosink
39
40 Egen test 6
41 t-launch-1.0 pylonsrc width=1920 height=1080 ! 'video/x-raw, width
       =1920, height=1080, framerate=10/1, format=YUY2'! nvvidconv ! '
       video/x-raw, width=1920, height=1080, framerate=10/1, format=
       RGBA'! nvvidconv ! 'video/x-raw(memory:NVMM), width=1920, height
       =1080, framerate=10/1, format=(string)NV12' ! nvv4l2h265enc
       bitrate=2000000 ! h265parse ! qtmux ! filesink location=test.mp4
        -e
42
43 Egen test 7
44 gst-launch-1.0 pylonsrc width=1920 height=1080 ! 'video/x-raw,
       width=1920, height=1080, framerate=10/1, format=YUY2'! nvvidconv
        ! 'video/x-raw, width=1920, height=1080, framerate=10/1, format
       =RGBA'! nvvidconv ! 'video/x-raw(memory:NVMM), width=1920,
       height=1080, framerate=10/1, format=(string)NV12' !
       nvv4l2h265enc bitrate=2000000 ! nvv4l2decoder ! queue ! nv3dsink
        -e
45
46 Egen test 8
47 gst-launch-1.0 pylonsrc width=1920 height=1080 ! 'video/x-raw,
       width=1920, height=1080, framerate=30/1, format=YUY2'! nvvidconv
        ! 'video/x-raw, width=1920, height=1080, framerate=30/1, format
       =RGBA'! nvvidconv ! 'video/x-raw(memory:NVMM), width=1920,
       height=1080, framerate=30/1, format=(string)NV12' !
       nvv4l2h265enc bitrate=2000000 ! h265parse ! queue ! h265parse !
       nvv4l2decoder ! nv3dsink -e
48
49 Egen test 9
50 gst-launch-1.0 pylonsrc width=1920 height=1080 ! 'video/x-raw(
       memory:NVMM), width=1920, height=1080, framerate=30/1, format=
       YUY2'! nvvidconv ! 'video/x-raw(memory:NVMM), width=1920, height
       =1080, framerate=30/1, format=RGBA'! nvvidconv ! 'video/x-raw(
       memory:NVMM), width=1920, height=1080, framerate=30/1, format=(
```

```
    string)NV12' ! nvv4l2h265enc bitrate=2000000 ! h265parse ! queue
     ! h265parse ! nvv4l2decoder ! nv3dsink -e
51
52  Camera not open yet, returning src template caps video/x-raw,
    format=(string){ GRAY8, RGB, BGR, YUY2, UYVY }, width=(int)[ 1,
    2147483647 ], height=(int)[ 1, 2147483647 ], framerate=(fraction
    )[ 0/1, 2147483647/1 ]
53
54  Spela upp h265 fil
55  gst-launch-1.0 filesrc location=test.mp4 ! qtdemux ! queue !
    h265parse ! nvv4l2decoder ! nv3dsink -e
56
57  FUNKAR utan grafikminne 1 till fil
58  gst-launch-1.0 pylonsrc width=1920 height=1080 ! 'video/x-raw,
    width=1920, height=1080, framerate=10/1, format=YUY2'! nvvidconv
     ! 'video/x-raw(memory:NVMM), width=1920, height=1080, framerate
    =10/1, format=(string)NV12' ! nvv4l2h265enc bitrate=2000000 !
    h265parse ! qtmux ! filesink location=test.mp4 -e
59
60  Gr n artifakt: grafikminne till fil
61  gst-launch-1.0 pylonsrc width=1920 height=1080 ! 'video/x-raw(
    memory:NVMM), width=1920, height=1080, framerate=10/1, format=
    YUY2'! nvvidconv ! 'video/x-raw(memory:NVMM), width=1920, height
    =1080, framerate=10/1, format=(string)NV12' ! nvv4l2h265enc
    bitrate=2000000 ! h265parse ! qtmux ! filesink location=test.mp4
     -e
62
63  grafikminne fr n fil till fil via en en extra detour.
64  gst-launch-1.0 pylonsrc width=1920 height=1080 ! 'video/x-raw,
    width=1920, height=1080, framerate=10/1, format=YUY2'! nvvidconv
     ! 'video/x-raw, width=1920, height=1080, framerate=10/1, format
    =RGBA'! nvvidconv ! 'video/x-raw(memory:NVMM), width=1920,
    height=1080, framerate=10/1, format=(string)NV12' !
    nvv4l2h265enc bitrate=2000000 ! h265parse ! qtmux ! filesink
    location=test.mp4 -e
65
66  Encoda och decoda h265 via grafikminne, gr n artifakt
67  gst-launch-1.0 pylonsrc !  video  /x-raw(memory:NVMM), width=1920,
    height=1080, framerate=30/1, format=YUY2 ! nvvidconv !
      video  /x-raw(memory:NVMM), width=1920, height=1080, framerate
    =30/1, format=(string)NV12  ! nvv4l2h265enc bitrate=2000000 !
    h265parse ! queue ! h265parse ! nvv4l2decoder ! nv3dsink -e
68
69  Encoda och decoda i h265 p  samma dator, via prim rminne
70  gst-launch-1.0 pylonsrc width=1920 height=1080 ! 'video/x-raw,
    width=1920, height=1080, framerate=30/1, format=YUY2'! nvvidconv
     ! 'video/x-raw(memory:NVMM), width=1920, height=1080, framerate
    =30/1, format=(string)NV12' ! nvv4l2h265enc bitrate=2000000 !
    h265parse ! queue ! h265parse ! nvv4l2decoder ! nv3dsink -e
71
72  Testa gr n artifakt med videotestsrc, ger ingen artifakt
73  gst-launch-1.0 videotestsrc ! nvvidconv ! 'video/x-raw(memory:NVMM)
    , width=1920, height=1080, framerate=30/1, format=YUY2'!
    nvvidconv ! 'video/x-raw(memory:NVMM), width=1920, height=1080,
    framerate=30/1, format=(string)NV12' ! nvv4l2h265enc bitrate
    =2000000 ! h265parse ! queue ! h265parse ! nvv4l2decoder !
```

```
    nv3dsink -e
74
75 testa GRAY8 direkt grafikminne
76 gst-launch-1.0 pylonsrc width=1920 height=1080 ! 'video/x-raw(
    memory:NVMM), width=1920, height=1080, framerate=30/1, format=
    GRAY8'! nvvidconv ! 'video/x-raw(memory:NVMM), width=1920,
    height=1080, framerate=30/1, format=(string)NV12' !
    nvv4l2h265enc bitrate=2000000 ! h265parse ! queue ! h265parse !
    nvv4l2decoder ! nv3dsink -e
77
78 Encoda och decoda med latency test
79 GST_DEBUG="GST_TRACER:7" GST_TRACERS=latency gst-launch-1.0
    pylonsrc width=1920 height=1080 ! 'video/x-raw, width=1920,
    height=1080, framerate=30/1, format=YUY2'! nvvidconv ! 'video/x-
    raw, width=1920, height=1080, framerate=30/1, format=RGBA'!
    nvvidconv ! 'video/x-raw(memory:NVMM), width=1920, height=1080,
    framerate=30/1, format=(string)NV12' ! nvv4l2h265enc bitrate
    =2000000 ! h265parse ! queue ! h265parse ! nvv4l2decoder !
    nv3dsink -e
80 ish 45 ms
81
82 Testa latens f r varje element
83 GST_DEBUG="GST_TRACER:7" GST_TRACERS="latency(flags=element)" gst-
    launch-1.0 pylonsrc width=1920 height=1080 ! 'video/x-raw, width
    =1920, height=1080, framerate=30/1, format=YUY2'! nvvidconv ! '
    video/x-raw, width=1920, height=1080, framerate=30/1, format=
    RGBA'! nvvidconv ! 'video/x-raw(memory:NVMM), width=1920, height
    =1080, framerate=30/1, format=(string)NV12' ! nvv4l2h265enc
    bitrate=2000000 ! h265parse ! queue ! h265parse ! nvv4l2decoder
    ! nv3dsink -e
84
85 Testa latens med bugg via endast grafikminne
86 GST_DEBUG="GST_TRACER:7" GST_TRACERS=latency gst-launch-1.0
    pylonsrc width=1920 height=1080 ! 'video/x-raw(memory:NVMM),
    width=1920, height=1080, framerate=30/1, format=YUY2'! nvvidconv
     ! 'video/x-raw(memory:NVMM), width=1920, height=1080, framerate
    =30/1, format=RGBA'! nvvidconv ! 'video/x-raw(memory:NVMM),
    width=1920, height=1080, framerate=30/1, format=(string)NV12' !
    nvv4l2h265enc bitrate=2000000 ! h265parse ! queue ! h265parse !
    nvv4l2decoder ! nv3dsink -e
87
88 Testa latens med bugga via grafik-prim r-grafik
89 GST_DEBUG="GST_TRACER:7" GST_TRACERS=latency gst-launch-1.0
    pylonsrc width=1920 height=1080 ! 'video/x-raw(memory:NVMM),
    width=1920, height=1080, framerate=30/1, format=YUY2'! nvvidconv
     ! 'video/x-raw, width=1920, height=1080, framerate=30/1, format
    =RGBA'! nvvidconv ! 'video/x-raw(memory:NVMM), width=1920,
    height=1080, framerate=30/1, format=(string)NV12' !
    nvv4l2h265enc bitrate=2000000 ! h265parse ! queue ! h265parse !
    nvv4l2decoder ! nv3dsink -e
90
91 Testa latens fr n prim r direkt till grafikminne
92 GST_DEBUG="GST_TRACER:7" GST_TRACERS=latency gst-launch-1.0
    pylonsrc width=1920 height=1080 ! 'video/x-raw, width=1920,
    height=1080, framerate=30/1, format=YUY2'! nvvidconv ! 'video/x-
    raw(memory:NVMM), width=1920, height=1080, framerate=30/1,
```

```
    format=(string)NV12' ! nvv4l2h265enc bitrate=2000000 ! h265parse
     ! queue ! h265parse ! nvv4l2decoder ! nv3dsink -e
93
94  Testa latency basfall
95  GST_DEBUG="GST_TRACER:7" GST_TRACERS=latency gst-launch-1.0
    pylonsrc ! videoconvert ! autovideosink
96  Ish v ldigt l gt men upplevd r h g
97
98
99
100
101
102
103
104
105
106
107
108 Encoda i h265 via prim rminne, skicka via srt. Mpegtsmux tillf r
    ca 2s latens. Portforward f r port 7000-7003 s anv nd n gon
     av dem f r att str mma srt.
109 gst-launch-1.0 pylonsrc width=1920 height=1080 ! 'video/x-raw,
    width=1920, height=1080, framerate=30/1, format=YUY2'! nvvidconv
     ! 'video/x-raw(memory:NVMM), width=1920, height=1080, framerate
    =30/1, format=(string)NV12' ! nvv4l2h265enc bitrate=2000000 !
    mpegtsmux ! queue ! srtsink uri=srt://:7000
```

Listing G.1: Pipelines

H

Data Sheets

H.1 The camera

The camera data sheet is pictured in Figure H.1.

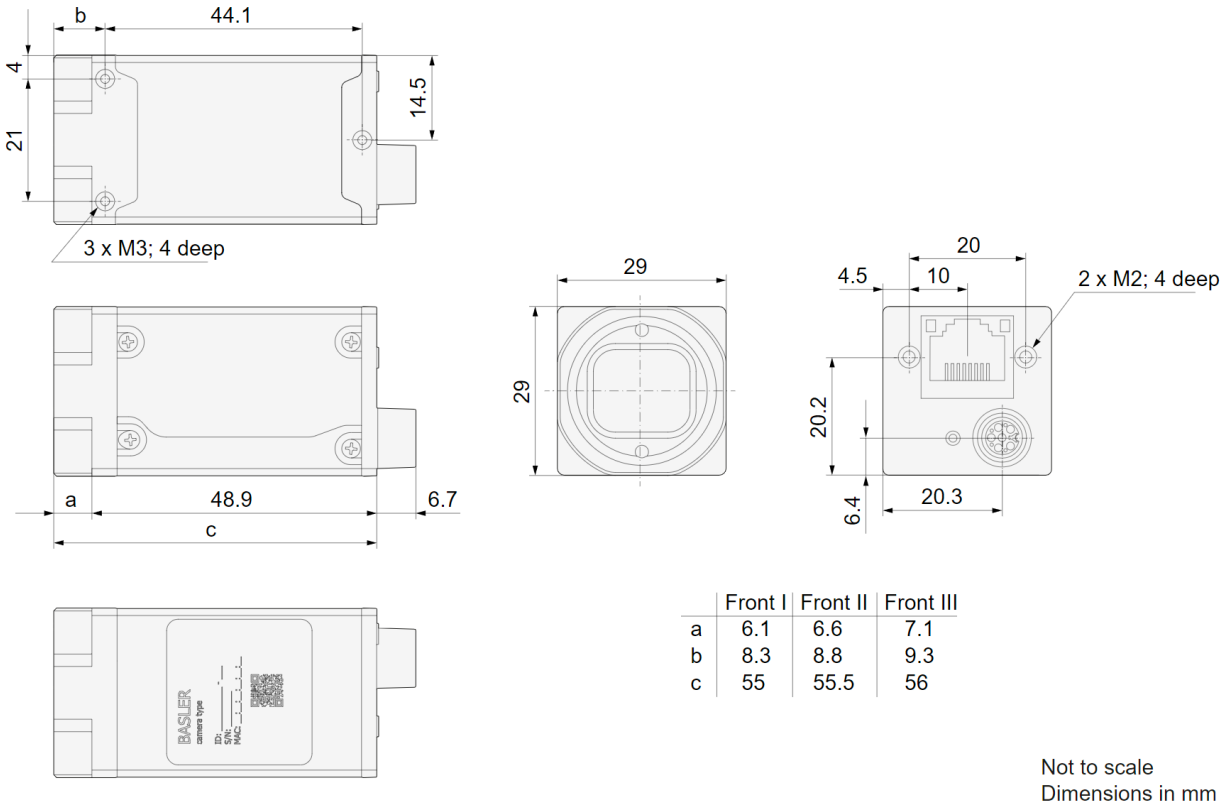


Figure H.1: The Camera used in the project

H.2 Components in the waterproof case

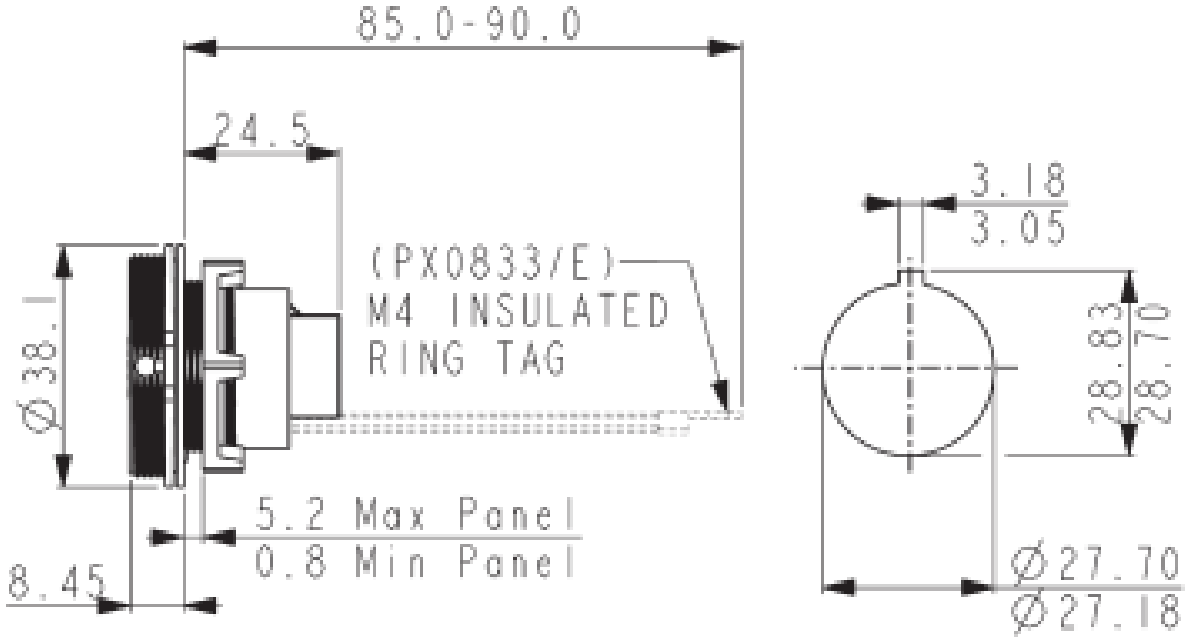


Figure H.2: Mounting connectors datasheet mentioned

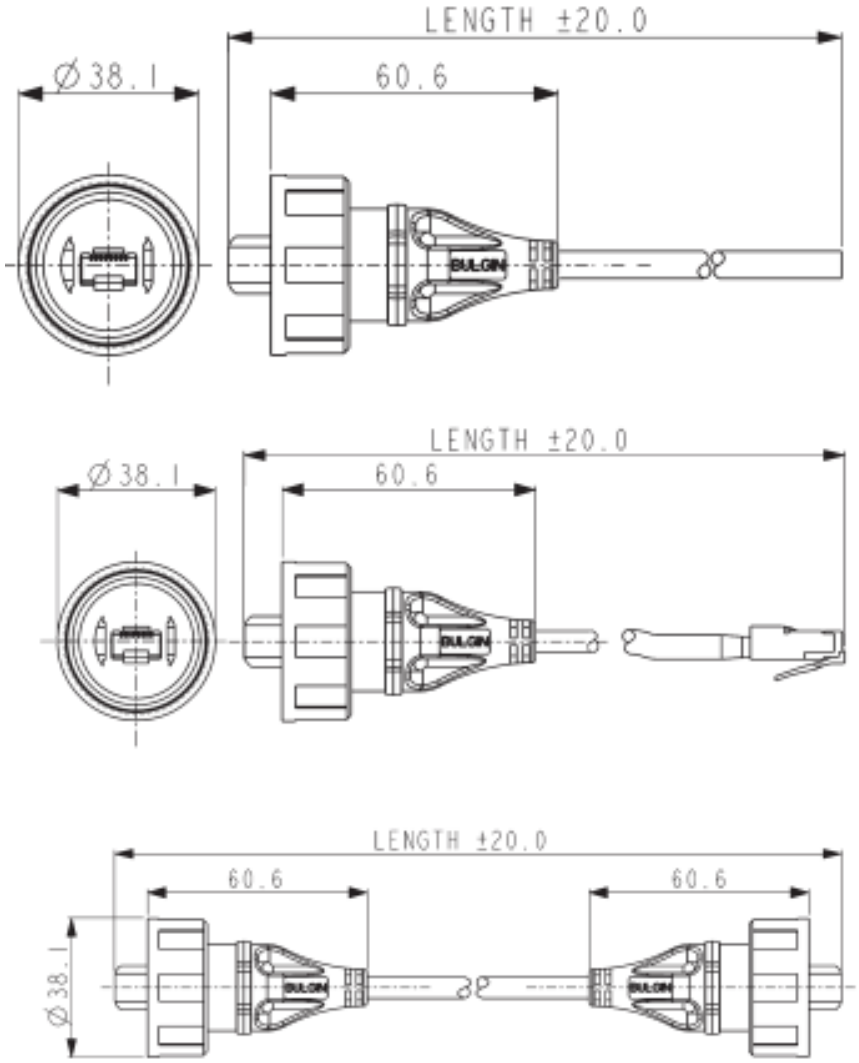


Figure H.3: Patch cord flex connectors datasheet

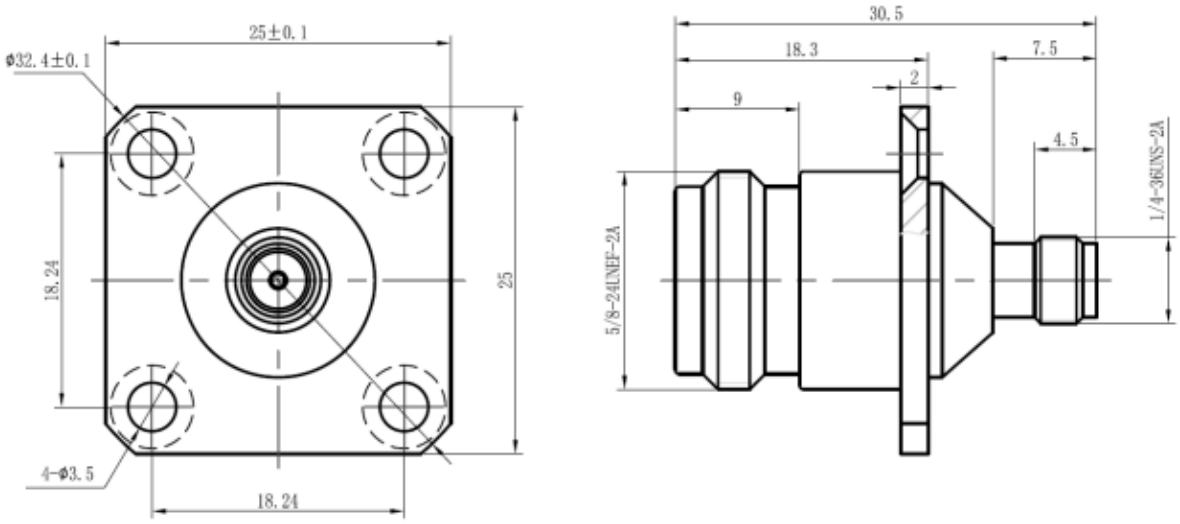


Figure H.4: Flange adapters data sheet

1

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS