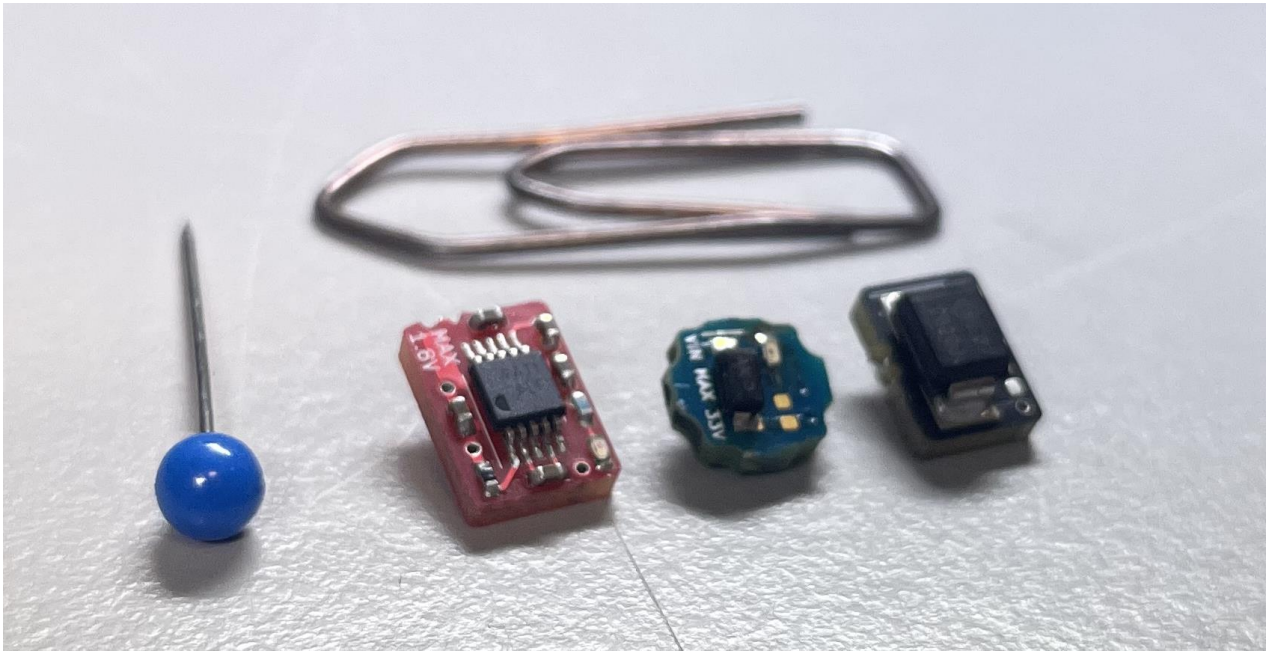


Optical probing of DC/DC converter voltages for sequencing verification

Bachelor Thesis in Electrical Engineering

Arvid Karlsson
Awaz Nabi Pour

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2023
www.chalmers.se

Abstract

The purpose of this project is to develop an application that can verify start-up and shut-down sequencing of DC/DC converters and that can be downloaded on a mobile phone for Ericsson employees to use on their company provided phones. It was concluded that a combination of using JavaScript together with HTML and CSS was the best way to create a program that can measure pixel values of LEDs on probe opto-modules. The probe opto-modules are placed on the circuit board by the user and then recorded during start-up and shut-down. To acquire accurate values from the recorded videos the super slow-motion camera feature, with a frame rate of 960 FPS, on a Samsung S22 mobile phone was used. Due to storage limitations on the phone, the video could only be recorded for about 0,5s with the super slow-motion feature.

The result was a functioning application that can be downloaded on both mobile phones as well as computers. The final application page consists of a header with five buttons, a canvas where the recorded video is displayed and a plot window at the bottom of the page to show the sequencing of the voltages. This application could reduce the workload on Ericsson employees, allowing more time for higher priority tasks than voltage measurement and contribute to an overall decreased environmental harm since far less instruments and components are necessary.

There are only parts of the program added throughout this report, such as functions and other relevant lines of code to make the text easier to understand. The entire code can be viewed by Ericsson employees only.

Abstrakt

Syftet med detta projekt är att utveckla en applikation som kan verifiera uppstart och avstängnings sekvensering av DC/DC omvandlare och som kan laddas ned på mobiltelefoner så att anställda på Ericsson kan använda applikationen på sina företagstelefoner. Det fastställdes att en kombination av JavaScript tillsammans med HTML och CSS var det bästa sättet att skapa ett program som kan mäta pixelvärdena av LED-lampor på probe opto-moduler, som användaren placerat på ett kretskort, från en inspelad video. För att få så exakta värden som möjligt från de förinspelade videorna användes super slow-motion kamerafunktionen, med en bildfrekvens på 960 FPS, på en Samsung S22 telefon. På grund av lagrings begränsningar på telefonen kunde videon endast filmas i ungefär 0,5s med super slow-motion funktionen.

Resultatet blev en applikation som kan laddas ned både på telefonen samt till dator. Slutversionen i detta arbete består av en huvuddel med fem knappar, en canvas där den inspelade videon kan visas samt ett grafritarfönster där sekvenseringen av spänningarna visas. Denna applikation kan lätta på arbetsbelastningen för Ericsson anställda, frigöra mer tid för annat viktigt arbete och minskar företagets negativa miljöpåverkan.

Det är endast delar utav programmet som är tillgängliga i denna rapport, så som funktioner och andra relevanta stycken av kod för att göra det enklare att förstå texten. Hela programmet finns tillgängligt för Ericsson anställda endast.

Abbreviations

- PWA: Progressive Web App
- OpenCV: Open-Source Computer Vision Library
- RGB: Colour model – Red, Green, Blue
- HTML: Hyper Text Markup Languages
- CSS: Cascading Style Sheets
- POM: Probe Opto-Module
- LED: Light Emitting Diode
- URL: Uniform Resource Locator
- ROI: Region of Interest
- FPS: Frames per Second

Acknowledgements

We would like to thank Camilla Karlsson and Fredrik Larsson for trusting us with this project and welcoming us when we first arrived at Ericsson. We also want to thank Sverker Sander and Björn Olsson who are the founders of this project and provided guidance to us as we continued to develop it. Our experience at Ericsson Electric Power Department has been nothing but great.

Johan Nicander at Ericsson “Garaget” deserves a big thank you for his expertise. He dedicated a lot of his time to help us with server- and programming issues as well as being someone to discuss problems and ideas with. We also want to thank Bruna Dos Santos Arujo who was very helpful and guided us through many of our programming issues.

Finally, we would like to thank Thomas Hammarström for agreeing to be the examiner of this project and helping us through the writing process.

Arvid Karlsson & Awaz Nabi Pour, Gothenburg, June 2023

Table of contents

1. Introduction

In this chapter an introduction to the project is provided, which purpose is to develop a method to confirm the voltage sequencing of DC/DC converters at Ericsson electric power department. First, the background as to why the project started is discussed, followed by the purpose of the project and how to achieve it. Last, the delimitation of the project is presented, such as phone camera limitations and time constraints.

1.1. Background

This project started in 2014 as an investigation into an alternative way of verifying start-up and shut-down sequencing of DC/DC converters. The reason as to why this project started was to investigate a new way to measure voltages on DC/DC converters. This could lead to more efficient work for the employees at the Ericsson electric power department as they can focus on other more important tasks.

Ericsson produces their own power supply to most of their products which puts a lot of responsibility on the electric power department. There are strict specifications sent to the department from other users within the company, stating the time and amplitude of DC/DC converter start-up and shut-down sequencing. At any of their circuit boards there could be up to 50 different voltage levels to supply other components that needs to be sequenced in a specific order. If it was possible to use a mobile phone to confirm the sequencing of the whole board at once there would be considerable time saved.

1.2. Purpose

The purpose of this project is to make voltage sequencing verification less of a time-consuming task for the employees of the electric power department at Ericsson. To make the process user friendly one of the main goals is to use a mobile phone to measure the voltage sequencing. Since voltage is not easy to measure with a mobile phone, small LEDs are fitted onto a circuit board where the testers would usually measure the voltages. By doing it this way it is possible to utilize the mobile phone camera to measure pixel values and confirm the timing of when voltage is applied across every measuring point. To achieve this, there are a few procedures that needs to be carried out [1]:

- Study DC/DC converter sequencing requirements in Ericsson product design specifications.
- Measure onboard DC/DC converter sequencing with available probe opto-modules.
- Develop algorithms for measuring of light intensity.
- Program an Android application that presents results alongside sequencing requirements.

1.3. Delimitations

Since user compatibility is important, the devices used to film the circuit boards are limited to company provided mobile phones, mainly the latest Samsung or iPhone model. This means that camera quality is limited, both in its ability to capture pixels and in filming slow motion videos.

User error will also factor when filming the LEDs, there is no specific setup in place to hold the mobile phone still for example.

Due to company rules and regulations an open server was not an option to use, it had to be closed to Ericsson employees only to comply with safety standards and prevent the risk of the server being infiltrated.

The timeframe of the project is limited to approximately 2 study periods, or 16 weeks, with other courses running parallel at 50 %.

2. Theory

This chapter provides necessary theoretical knowledge to understand the method that was used to achieve the resulting application, starting with Ericsson sequencing specification and the manufactured probe opto-modules. After that the software is explained, such as the programming languages, IDEs and libraries that are used to help simplify the code. Finally, the pixel value measurement is explained. Abbreviations used within the report are explained in the "abbreviations" chapter.

2.1. Sequencing Verification

Ericsson products are subject to requirement specifications that they need to adapt to. Among these requirements are voltage sequencing of both shut down and startup of DC/DC converters, which can be seen in table 1. These timing requirements are critical and on any given circuit board there could be up to 50 different voltage levels each with their own timing. The sequencing requirements could say that one specific output of the board should be applied with a voltage level during the first 0-25 milliseconds of startup and another output should be applied no more than 15 milliseconds after the first one [2]. The specific sequencing times matter a lot since certain circuits need to be active before others to enable correct behaviour.

Table 1: Voltage sequencing specification. [2], Adapted with permission.

Rail A	Limit	Rail B	Min A→B	Max A→B	Result	PASS/ FAIL	Notes
RAIL_1 RAIL_2	5.8V LOW	RAIL_6 RAIL_3 RAIL_22 RAIL_23	25ms	75ms	75ms 67ms 67ms 67ms	PASS	seq7 seq8 seq9 seq10
RAIL_3	0,90V	RAIL_4	20ms		33ms	PASS	seq12
RAIL_4		RAIL_5	20ms		27ms	PASS	seq13
RAIL_5		RAIL_7 RAIL_24	20ms		26ms 21ms	PASS	seq14 seq15
RAIL_6 RAIL_7		RAIL_25 RAIL_9 RAIL_8	20ms		102ms, 27ms 97ms, 22ms 101ms, 27ms	PASS	seq16, seq19 seq17, seq20 seq18, seq21
RAIL_8 RAIL_9		RAIL_10 RAIL_11 RAIL_12 RAIL_13	20ms		26ms, 29ms 26ms, 30ms 32ms, 35ms 26ms, 30ms	PASS	seq22, seq26 seq23, seq27 seq24, seq28 seq25, seq29
RAIL_10 RAIL_11 RAIL_12 RAIL_13		RAIL_14	20ms		26ms 26ms 21ms 25ms	PASS	seq30 seq31 seq32 seq33
RAIL_14	High	RAIL_15	100ms		101ms	PASS	seq34
RAIL_15	High	RAIL_16 RAIL_17 RAIL_18 RAIL_19 RAIL_20 RAIL_21	TBD	TBD	28ms 24ms 13ms 32ms 9.8ms 9.8ms	PASS	seq35 seq36 seq37 seq38 seq39 seq40
RAIL_16 RAIL_17 RAIL_18 RAIL_19 RAIL_20 RAIL_21		RAIL_26	20ms		30ms 33ms 45ms 25ms 47ms 47ms	PASS	seq41 seq42 seq43 seq44 seq45 seq46
RAIL_15		RAIL_27	TBD	TBD	211ms	N/A	seq47

2.2. POM

POMs, or probe opto-modules, are used to indicate that a voltage is applied across outputs on the circuit boards. They are fitted with LEDs which lights up as voltage is applied. The POM circuit was designed by employees at the power department as a part of the initial project. They were manufactured at the Ericsson production line in Borås and were assembled by hand. Their size is very small which makes it possible to fit a lot of them onto one board, for example one of the POM dimensions are 5.5 times 5.5 millimetres. They were created in a range of different voltage levels to be compatible with most of the levels used in everyday business, with the lowest one being 0.9-1.8 V and the highest one being 2.5-63 V [3].

Figure 1: Probe opto-modules. [4], Adapted with permission.

2.3. PWA

A PWA, progressive web app, is a web-based application that can be installed onto any device. Once the PWA is installed on a phone it will look like and work just like any other application and will among other things: have an icon on the home screen, open its own window on the device separate from the web browser as well as be functional offline. Many popular applications such as Microsoft office 365, Hulu, Instagram lite and YouTube music are examples of PWAs. PWAs are available to download by opening the link to the web app, but they are also available online to download from application platforms like Apple's app store and google play store [5].

2.4. Visual Studio Code

Visual studio code is a broadly used code editor and is supported by many coding languages such as JavaScript, Python and C++. It supports development operations such as debugging, and task running and provides a broad range of necessary tools for developers. Visual studio Code has many different features such as, live sharing, debugging and built in source control [6].

2.5. HTML

HTML is the most used markup language to create and structure the content of a webpage. In HTML a page is built sectionally using tags such as <head></head> and <body></body>. Within these tags an array of different elements can be introduced to create a website or application such as file inputs, buttons, scrollbars, titles, and images. HTML code is written as a text file and web browsers can interpret the tags and elements which is displayed on the page [7].

2.6. CSS

CSS code is a stylesheet language which is used together with HTML as a way of formatting the looks and style of a webpage. It can be used to change font size and style, colours of different elements and alignment of elements within the page for example. CSS is helpful since it allows the creator to separate the content creation which is done in HTML from the appearances of the content. CSS code is written using selectors which can target different elements in the HTML code. The creator can for example target the body element of the HTML file and in the CSS code define the font style of everything in the body to be times new roman [8].

2.7. JavaScript

JavaScript is a programming language that can be used to incorporate more complex features and functions to webpages. This makes it possible for webpages to do things like react to a click of a button or take the user to another page which gives webpages more functionality and broaden the user experience. While HTML and CSS help creating the shell for a webpage, JavaScript is what makes the webpage function and react on user interaction.

JavaScript can be written directly into HTML code by using <script></script> tags. It can also be written in its own file that later can be called in the HTML file. Together, CSS, HTML and JavaScript can create a fully functioning webpage with many different types of complex features [9].

2.8. OpenCV

OpenCV is a library of algorithms and functions designed to make programming easier when working with visuals such as camera and video capture. It was initially created for C++ code but has since been converted into multiple programming languages, such as python and JavaScript. According to [10] the OpenCV library has more than 2500 optimized algorithms that ranges from computer vision to machine learning algorithms.

2.9. Fabric

Fabric is a JavaScript based framework that makes it possible to create interactive geometrical shapes like circles, rectangles, and polygons [11]. By using Fabric, it is among many other things, possible to change the size, rotation, and position of the different shapes by clicking on the screen. Fabric also makes it possible to manipulate HTML canvases by creating an interactive object on top of a canvas element [12].

2.10. Plotly

Plotly is a JavaScript charting library with a large range of chart types including statistical graphs, financial charts, 3D charts and much more [13]. Plotly enables highly customizable graphs and plots which makes it easier for the developer to create user friendly content. Every chart within the library is interactive, meaning that the user can zoom in and out or mark data-points within the plot for example.

2.11. Python

Python is a high-level programming language. It launched in 1991 as an option to other existing programming languages such as C, C++, or Java. The main goal was to make it easier from users to write complex code using fewer lines [14]. Python is a general-purpose language that can be used in many different cases and projects, from self-driving cars to machine learning and AI. Python is popular among developers due to its large community and range of easily accessible function libraries. There are many forums dedicated to Python programming which also makes it easy for beginners to ask for help and guidance [15].

2.12. Pixel Value Measurement

Within the scope of the project was the ability to measure light intensity to be able to verify a change in voltage applied across specific points of the circuit boards. However, there was found to be an easier way of doing the verification which was to measure pixel values. The way it is done here is to use the functionality of the OpenCV library. In the library the function “mean ()” [16] calculates the mean value of a pixel or multiple pixels within a 3-channel (RGB) image. This makes it easy to determine when a LED is turned on or off due to the difference in RGB value. The video is converted to grayscale to make the distinction even more clear. A LED which is turned on displays a bright grey light which approximately has a RGB value [200, 200, 200] while a turned off LED displays a dark grey light that has a RGB value around [50, 50, 50] [17]. This is shown in table 2. It is important to realize that these values are ideal, in reality the values will vary depending on lighting within the room that the video is being filmed, the angle of the mobile phone camera and the distance between the camera and the circuit board.

Table 2: RGB values. [17].

Red	Green	Blue	Colour
50	50	50	Dark Gray
120	120	120	Medium Gray
200	200	200	Light Gray
0	0	0	Black
255	255	255	White

3. Method

In this chapter, the method of creating the finished program is explained in the chronological steps that it was performed. Starting with initial work which included research and testing different platforms. When initial research was finished and programming languages were decided the graphical design planning, how the application was going to be used, took place. Finally, the actual programming began in JavaScript, HTML and CSS.

3.1. Initial work

The first objective was to investigate how the mobile phone application was going to be created. As the project started in Matlab and a functioning algorithm had already been created there, the initial work became more focused on converting the code from Matlab to a mobile application. It quickly became clear that this was not the most effective way to continue as Matlab's compatibility with phone application was limited.

After further research, Python code seemed to be able to create web-applications which then could be used on mobile phones, not as an app but rather as a website. Python was also extremely well compatible with OpenCV which seemed to be the best way to measure the LED pixel values.
The connection was simply done by one line of code, followed by a reading of the video file through OpenCV.

```
# Import the OpenCV library
import cv2
# Read the video file using OpenCV
cap = cv2.VideoCapture(root.filename)
```

It was then possible to manipulate the video file with existing functions from OpenCV. The algorithm was created in Python and almost became a complete and fully functioning program. While the algorithm was being finished the connection between Python and the web proved to be more difficult than what was initially believed.

More research was done and after collaboration with Ericsson developers it was determined to write the algorithm in JavaScript and use HTML and CSS as the visual presentation since this combination is by far the most widely used combination for web applications. To make it work on a phone it was decided to turn the program into a PWA [18] using an existing template [19]. This way it could be downloaded from a link without having to go through Appstore or google play store. OpenCV is not as compatible with JavaScript as it is with Python but a sufficient part of the OpenCV library have been converted to JavaScript.

Videos to test the final application were filmed in one of the power departments labs using their existing circuit boards. There, the start-up and shut down of a power supply circuit board was filmed using the super slow-motion camera function on a Samsung S22 mobile phone.

3.2. Graphical Design Planning

During early stages of the project a plan of the applications graphical presentation took form. Since the target audience were the employees working in the power department labs it was decided to create some sort of testing involving them as soon as the first working prototype were available. The

key features were clearly defined alongside the supervisors, as can be seen in the purpose part of this report.

Initially the goal of the user experience was quite simple:

- Film the circuit board with POMs attached.
- Mark ROIs (Region of Interest).
- Process the video and measure the LEDs.
- Present the data on a new page in a graph.

However, as the project moved forward changes were made from the initial graphical design plan. Some of the features proved to be very difficult to achieve, such as the fact that a user could not drag their finger on the canvas element to navigate the screen of the phone. The solution became to use the existing scroll-function of the mobile phone. Other changes were made due to better and more practical solutions presenting themselves during the project. It was decided that the graph presenting the results should be presented below the video instead of on a new page to make it easier for the user to remember which ROI was connected to which POM. As far as the aesthetics of the application go, colours, fonts, and style, could be improved upon to.

3.3. Developing a functional webpage using HTML and CSS

HTML was used to create a functional webpage that has all the necessary blocks for the program. The HTML code created a shell for the web application and makes it possible for the user to interact with the program. Text, Buttons, canvases, and plots were all coded in HTML and then presented on the page.

The first part of the HTML code declares what type of document it is and sets the standard language of the program to English. The next part of the code was written inside the <head></head> tags. This part contains metadata and links to external libraries such as OpenCV, Fabric, and Plotly.

```
<!-- Add OpenCV library -->
<script async src="opencv.js" onload="initialize();"></script>

<!-- Add fabric library -->
<script src="https://unpkg.com/fabric@5.3.0/dist/fabric.min.js"></script>

<!-- Add plotly library -->
<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
```

The JavaScript file was also linked here. The top lines of the <head></head> section of the HTML code specify the character set and browser compatibility. The title of the application was thereafter set to “Optoprobing app”. The next <meta> tag set the webpage to the same size as the device that was currently used.

```
<!-- Title -->
<title>Optoprobing app</title>

<!-- Set viewport -->
<meta name="viewport" content="width=device-width, initial-scale=1">
```

After the <meta> tag, there are two rows of code. The first one makes the webpage downloadable and is a crucial part of turning the webpage into a PWA. The second row sets the icon of the application.

```
<!-- To make the web page downloadable -->
<link rel="manifest" href="./manifest.json">
```

The next line of code links the CSS style document to the HTML document. To enable offline usage and faster loading of the page, a service worker was included in the code [20]. The main JavaScript file was linked in the HTML code to enable interaction with the HTML blocks and the main algorithm of the program.

The next part of the HTML code is the <body></body>. Everything that is seen on the page from the user side was coded in this block. At the top of the block, the title of the page was set to "Optoprobing".

```
<!-- Title at top of page -->
<h1 class="top">Optoprobing</h1>
```

Thereafter, an input button was created so that the user could upload a video. The "accept" line in the <input> tag specifies what type of file is allowed to be uploaded.

```
<!-- Add button to upload file -->
<div class="mid">
    <input type="file" id="fileInput" accept="video/*" class="buttons">
</div>
```

A header with informative guidelines and five buttons were then created. The first button initiates the processing of the video through the JavaScript algorithm when the user is finished marking ROIs. The second button was created so that the user could delete the latest drawn ROI if it was placed poorly. The third button is used to reload the page and let the user start from the beginning. The fourth and fifth buttons are used to increase or decrease the size of the ROI.

```
<!-- Adding buttons and guidelines-->
<nav>
    ** Scroll and zoom here or on the right side of the video. Plot appears at the bottom **
    <p></p>
    <button id="processVideo" class="buttons">Process Video</button>
    <button id="deleteROI" class="buttons">Delete latest ROI</button>
    <button id="reloadPage" class="buttons">Restart page</button>
    <button id="sizeUp" class="buttons">Size Up ROI</button>
    <button id="sizeDown" class="buttons">Size Down ROI</button>
</nav>
```

The next part inside the <body></body> tag displays the video and canvas elements. Two canvases were created and placed on top of each other. The first one was used to display the uploaded video and the second one was used to draw ROIs on top on the video. Both canvases were set to the same size to make it seem to the user that there is only one.

```
<!-- Canvas for drawing ROIs -->
<div class="canvas-container">
```

```
        <canvas id="backgroundCanvas" style="position: absolute;" width="720"
height="1280"></canvas>
        <canvas id="canvas" style="border:2px dotted #ffffff;" width="720" height="1280"></canvas>
    </div>
```

The size of the canvas element was set to 720x1280 as this is the standard size of a video filmed with the Samsung s22. The last part of the HTML file plots the result of the pixel measurements. The X-axis of the plot represents time in milliseconds, and the Y-axis represents the mean pixel value.

In many of the parts within the <body></body> tags there is an “class” attribute. This is what is used to tell the CSS document how a specific part of the program should look or behave. The class attribute that is called “top” is used in the CSS document to set the text fixed at the centre of the page, the font size, font style and font colour. The next class attribute, mid, also fixes the file input button to the centre of the page. In the “buttons” class attribute all the button sizes, font sizes, font styles and colours were set. An animation was also added to the buttons so that they react when being clicked on. This was done to make it clear to the user a button has been clicked.

```
buttons {
    margin-left: 10px;  /* leftside margin */
    height: 70px;   /* button height */
    width: 150px;   /* button width */
    font-size: 16px;    /* button font size */
    margin-right: 10px; /* right side margin */
    border-radius: 8px; /* button border thickness */
    text-align: center; /* center button text */
}
```

The <nav> tag holds the entire header including the guidelines and the buttons. In the CSS document everything within <nav></nav> was firstly set to z-index = 1. In a three-dimensional plot the z-axis would be from the screen towards the user. This means that z-index = 1 puts the header and buttons on top of every other element on the page. Secondly, everything in the <nav> tag was set to be “sticky”. The sticky tag makes elements stick to the top of the page when the scroll functions is used. This was necessary to have for the user experience to avoid having to scroll back up to the top when all the ROIs have been marked.

```
/* header */
body nav {
    text-align: center; /* align text to center */
    position: sticky;   /* set positin to sticky */
    top: 0; /* pisition element at top */
    z-index: 1; /* Z index to 1, move forward */
    background-color: rgb(105, 109, 112);   /* set background color */
    width: 730px;   /* set width */
    border: dotted; /* add dotted border */
}
```

3.4. Algorithm in JavaScript

The main function of the application is the algorithm which measures pixel values, this was created using JavaScript. In the JavaScript file variables were initialized, event listeners were set, and Functions were defined.

The first few lines initialize a few variables, some Booleans and an array that were later used in the code. Thereafter the main function “Initialize” was set up. There, an if statement was created to check if the Fabric and OpenCV libraries have successfully been loaded. If not, the function “initialize” were to be called again after 50 milliseconds.

```
// Check if fabric and opencv are loaded
if (typeof fabric === 'undefined' || typeof cv === 'undefined') {
  setTimeout(initialize, 50) //if not loaded, wait 50 ms and check again
  return
}
```

When both libraries have been loaded, the function cv.onRuntimeInitialized runs. This function adds event listeners to the different buttons on the page that reacts when they are clicked on.

```
  document.getElementById('processVideo').addEventListener('click', processVideo1)  //process video button
  document.getElementById('deleteROI').addEventListener('click', deleteROI) //delete latest ROI button
  document.getElementById('reloadPage').addEventListener('click', reloadPage) //reload page button
  document.getElementById('sizeUp').addEventListener('click', largerSize) //Increase size of ROI
  document.getElementById('sizeDown').addEventListener('click', smallerSize) //Decrease size of ROI
```

When uploading a file by using the “choose file” button, the function “handleVideoUpload” is called. This function takes the uploaded video file and creates a URL for it by calling the function “URL.createObjecURL(file)”, where the “file” variable is the name of the uploaded file. The file URL is then assigned to the “src” attribute of the video element.

```
const fileURL = URL.createObjectURL(file) //Create object URL for file
video.src = fileURL //Set video source to object URL
```

When the video file is loaded, the height and width of the canvas and the Fabric canvas are set to the size of the video. Adding ROI boxes is also a feature that was coded in the “handleVideoUpload” function. This was done by adding an event listener to “mouse.up” that reacts when the user clicks and releases the left mouse button, or clicks on the screen using a finger on the canvas.

```
  fabricCanvas.on('mouse:up', () => { //mouse up event listener
    ROIcount++ // Increment the rectangle count
    const pointer = fabricCanvas.getPointer() //get mouse position

      const rect = new fabric.Rect({ //create ROI
            left: pointer.x,  //sex X pos of ROI
            top: pointer.y, //set Y pos of ROI
            width: 20,  //width of ROI in pixles
```

```
            height: 20, //height of ROI in pixles
            left: pointer.x - rectWidth / 2, // Center ROI round mouse position
            top: pointer.y - rectHeight / 2, // Center ROI round mouse position
            fill: 'rgba(0,0,0,0)',  //Set the color of the ROI
            stroke: 'blue', //Set outer edge of ROI to blue
            strokeWidth: 2  //thickness of outer edge
        })
    }
```

The event listeners of the different buttons all have different tasks. If the "Delete Last ROI" button is pressed, the function "deleteROI" is called, which removes the latest drawn ROI on the canvas. If the "Restart" button is pressed the function "reloadPage" is called which reloads the webpage and allows the user to start from the beginning. If any of the size up or size down buttons are pressed the size of the ROI will increase or decrease by 5x5 pixels.

If the "Process Video" button is pressed the function "processVideo1" is called, which sets the Boolean "isProcessing" to true, and then calls the function "playVideoAndMeasureIntensity". This function plays the video on the page through the canvas. By using "cv.VideoCapture", the frames from the video are captured and stored in a variable called "cap". Another variable called "FRAME_RATE" was declared and given the value 960, that is the same frame rate of the phone camera during super slow-motion mode.

```
    const src = new cv.Mat(video.videoHeight, video.videoWidth, cv.CV_8UC4) //create new cv.mat object with video dimentions & 4 chanels
    const dst = new cv.Mat(video.videoHeight, video.videoWidth, cv.CV_8UC1) //create new cv.mat object with video dimentions & 1 chanel
    const cap = new cv.VideoCapture(video)  //create new cv.mat object with video element

    const FRAME_RATE = 960    //set frame rate in fps
```

The program then enters another function called "processVideo". This function checks if the Boolean "isProcessing" is true, if it is, the frames stored in "cap" are converted from RGB to grayscale using another function from OpenCV called "cv.cvtColor". If the video has finished playing, the function stops processing the captured frames.

```
if (isProcessing) { //check if processing enabled
        if (video.ended) return //return if video ended
        cap.read(src) //read current video frame into src matrix
        cv.cvtColor(src, dst, cv.COLOR_RGBA2GRAY) //convert frame color to grayscale
```

Inside of "processVideo" the function "fabricCanvas.forEachObject()" is called. This function loops through all the ROI objects that has been placed on the Fabric canvas and stores them in an ROI matrix. It also measures the mean pixel values inside of the bounding rectangles for every frame and stores them in a matrix called "lightIntensities". Lastly the function deletes the ROI matrix.

```
            const roiMat = dst.roi(roi) //get ROI from grayscale frame
            const meanIntensity = cv.mean(roiMat) //calculate mean intensity in ROI
            lightIntensities.push(meanIntensity[0]) //Add mean intensity to array
```

The next steps in the "processVideo" function displays the grayscale frames on the canvas element in HTML and then calculates the delay until the next coming frame should be processed. This is

calculated based on the FPS and the time between two frames. If the delay is negative, the delay variable is set to 0. The function “setTimeout” is then called so that the next frame can be processed.

The function “catch(err)” was implemented so that if an error occurs, the “cap”, “dst” and “src” variables are deleted.

```
catch (err) {
      console.error(err)  //if error, log to console
      src.delete()  //delete src matrix obj
      dst.delete()  //delete dst matric obj
      cap.delete()  //detete cap obj
    }
```

The next part of the “processVideo” function initiates two variables and one new matrix called “newMatrix”. The function “createNewMatrix” fills “newMatrix” with the measured pixel values for each ROI in separate rows. Lastly the function “createNewMatrix” returns the “newMatrix”.

```
    const newMatrix = createNewMatrix(lightIntensities, ROIcount) // declare matrix that will contain values of lightintensity for all ROIs
    const data = newMatrix.map((row, i) => ({ x: row.map((_, j) => ((j + 1) / FRAME_RATE) * 1000), y: row, type: 'scatter', name: `ROI: ${i + 1}` })) // data for graph
    const layout = { xaxis: { title: 'time [ms]' }, yaxis: { title: 'intensity' } } // layout for graph
```

After this, the values in “newMatrix” are plotted by using plotly. The plot presents each ROI as its own line. The X-axis is presented in time (ms) while the Y-axis presents light intensity.

```
Plotly.newPlot('plot', data, layout)  //plot graph using plotly
```

Another variable was declared: “startTime”. This variable is active when the function “playVideoAndMeasureIntensity” is called and sets the starting time to help calculate the delay variable. Last of all an event listener was declared to check if the page has been loaded successfully, if it has, it calls for the “initialize” function and goes through the main part of the JavaScript algorithm.

4. Results

Over the course of the project multiple ways to achieve the final application were tested. The most significant finding was that JavaScript together with HTML and CSS was the best and most efficient programming languages to use when creating the application. This was also one of the first results that were found. Another early and important result was the decision to make the program a downloadable web-application. This choice had major implications as it made it possible to disregard the preinstalled application store on iPhone and Samsung.

As the algorithm became better and more accurate some of the focus was shifted towards usability. It was difficult to navigate the page if you had not done it many times before. The use of buttons on the screen was found to be helpful. A guiding text was also placed here since it was not obvious that the user could not scroll or zoom on the video itself. The size up and size down buttons were not part of the initial program that was sent out to be tested, but after feedback from users they were added to the application to make it easier to mark ROIs of different sizes. In figure 2 the header holding the buttons and the user guidelines can be seen.

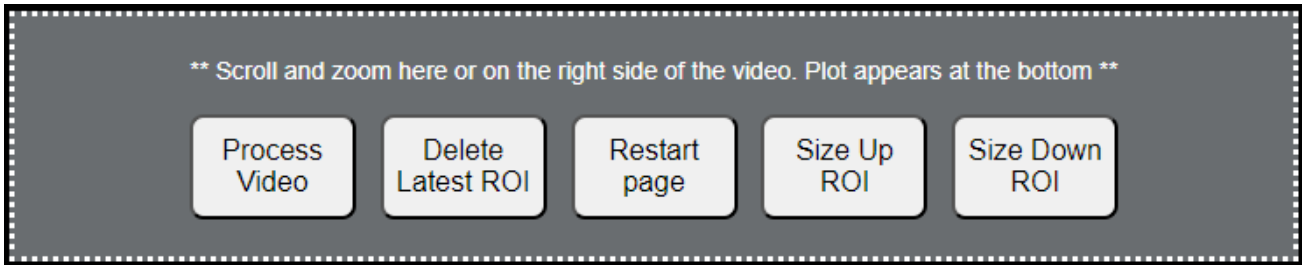


Figure 2: Header with guidelines and buttons

During the testing of the application using the Samsung S22 phone it was found that, in order not to fill the memory storage immediately, a time limitation is put on ultra slow-motion videos. This limitation is not something that the user can change. The Samsung S22 can film approximately 0.5 seconds of 960fps that is played back over approximately 17 seconds [21]. This puts a limit on the time between the first LED to the last LED. This was unknown at first and became a delimitation to the project.

While filming videos with the phone it was also found that external light on the circuit board could affect the camera's ability to register the flashing LED. When a bright light was shining directly and closely to the circuit board the camera could not register the flashing LED at all and thus did not trigger the ultra-slow-motion video to start recording. There was no problem for the camera to react to the flashing light from a LED in regular lit rooms. If the rooms were darker than normal was not an issue for the camera either, however it instead became difficult for the user to mark the regions of interest on the video.

The resulting plot was found to be affected depending on the size of the ROIs. If the user chooses to make the ROIs very large and cover an area that might be unnecessarily big the plot is not as clear as if the user makes the size of the ROI exactly cover the LEDs.

In appendix A, a flowchart of the complete program that is explained in chapter 3 can be seen. The flowchart represents the important functions of the program from start to finish, and shows the different options the user can make while using the application.

One of the main goals of this project was to be able to use a company provided mobile phone to measure voltage sequencing during start-up and shut-down of DC/DC converters. Figure 3 shows the resulting graph that is presented to the user on the mobile phone after filming a DC/DC converter during start-up, marking the regions of interest, and successfully processing the video using the algorithm. It is easy to determine the time of when a specific output of the circuit was applied with a voltage. The plot can be navigated in many ways and zoom can be used to see very short time differences.

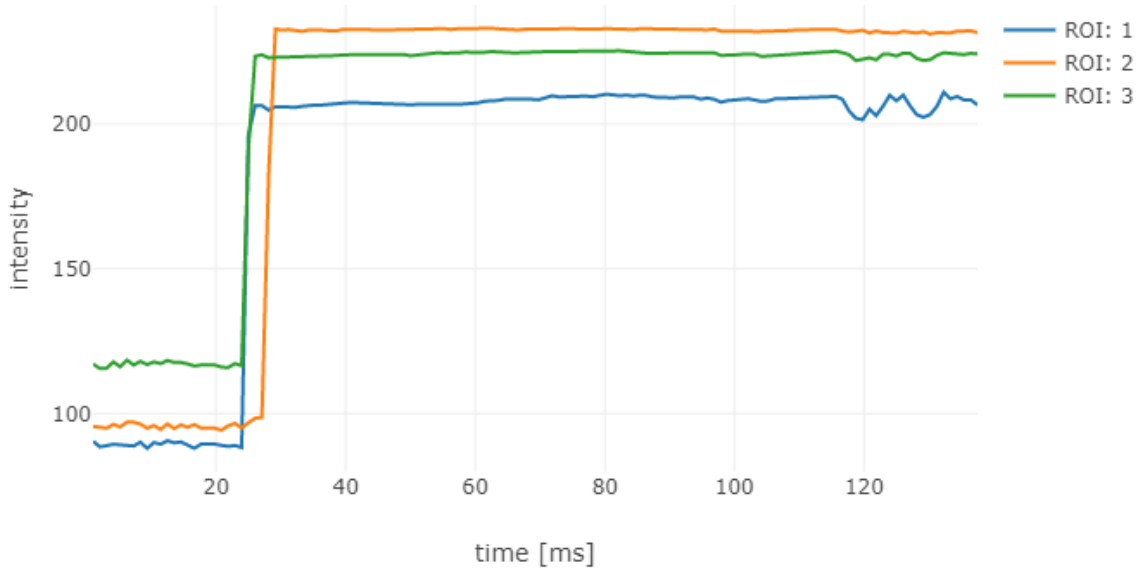


Figure 3: Startup sequencing of DC/DC converter

5. Conclusion

This chapter provides conclusions on the results provided over the course of the project. Flaws in the first algorithms are discussed as well as some issues and compromises done during the graphical design planning. The final program and the user experience is analysed. This chapter also discusses environmental effects and future developments.

5.1. Initial Work

During the initial work, a Python program was developed. The program that was produced could not on its own determine the number of ROIs, therefore the user was prompted to enter the number of ROIs they were intending to draw beforehand. This number was then fixed and could not be changed without rerunning the program. The user also needed to change the code manually to be able to measure and plot the light intensities. These properties were seen as major flaws in the ability for users to easily use the program and to convert the code into a functioning application.

Apart from the flaws mentioned, the rest of the program worked on a computer according to the specification provided by Ericsson. When converting to a phone application, the OpenCV library in python was not compatible with web application usage. This made the library unusable in the program that it was intended for. Other libraries with similar features could not be applied to the already developed program since they lacked ability to manipulate the canvas in which the video was played. This resulted in changing the programming language of the main program to JavaScript, since a translation of OpenCV in python was available in JavaScript. As a result of this, it was concluded that JavaScript was the optimal programming language for the application.

5.2. Graphic design planning

The planning of the graphic design contained some key qualities that had to be a part of the applications as well as some qualities that would make the application more user friendly. One of the most important qualities was that the user should be able to draw ROIs. A predefined size of the ROI box was set so that the user could just klick on the middle of the LED, without having to change the size of the ROI. Although this makes it easier for the user to place the ROIs, it makes the results less accurate. If the video that is being used with the program is filmed from a distance, and the LEDs become too small on the video, the set size of the ROI could become too big. Therefore, after user feedback, two buttons that can change the size of the ROIs were added to the program.

Another issue that relates to the one described above is the fact that the user is not able to zoom in on the canvas that the video is played on. Trying to zoom in by using two fingers on the screen simultaneously as users do on most platforms is not possible and results in drawing ROIs in unwanted places. The solution in this case was to create a small space on the “floating” bar where the user can use two fingers to scroll and zoom in on the page. This is not the most optimal solution but the attempts that were made to apply a two-finger pinch feature were unsuccessful.

5.3. Developing a functional webpage using HTML and CSS

One of the main parts of the page is the displaying of the graph. It is important that it is visible and that nothing else on the page covers any part of it. To solve the issue of the header covering the plot, the position of the plot was set to “absolute”. This resulted in a clear view of the graph, although the user could still accidentally move the graph up or down. A solution to this would be to open the

graph on its own linked page, so that the graph could take up more space on the screen and user error could be reduced. Unfortunately, opening a linked page would result in server difficulties since the program is meant to run on a private Ericsson server. It was therefore concluded that opening a new linked page was outside of the assignment scope.

5.4. Algorithm in JavaScript

The main program code in JavaScript includes the functionality of the buttons on the page. Since the buttons are seen as Booleans, they are a part of the code and set the order of events while the code is running. While developing the code, four buttons were used, but after developing a functioning code, it was concluded that only three buttons were needed. This resulted in changing the code to adapt to three buttons instead of four, merging two buttons into just one. After the user feedback came back however, two more buttons were added to the program to change the size of the ROIs, resulting in five buttons in the final application according to figure 3. This use of buttons was seen as the best way to increase useability of the application with the limited time available.

5.5. Environmental impact

If the application was to be used more widespread, there would be less need for measuring tools such as oscilloscopes and multimeters. At most companies such as Ericsson, many of the staff members already have access to a company phone which would make it easier to implement the application in everyday work. The negative environmental impact of producing measurement tools would become smaller the more the application is used. Another impact that the application might have would be that the application saves time and lightens the workload, which would result in a more effective way of working. This would leave more time for employees to develop new solutions for other things.

5.6. Future development and improvements

There is no shortage of improvements that could be made to the application. A problem in the project was the short time one could film a super slow-motion video on the mobile phone. This puts a limit on the range of DC/DC converters the algorithm can process. There are existing high-speed cameras that could be used for longer videos and connected to computers or tablets to run the film through the algorithm, but these are expensive. The reason to use a company provided mobile phone was to make the program easy to access for everyone who wanted to make use of it, using a high-speed camera and processing equipment would be working in the opposite direction. Luckily most of the DC/DC converters do not have a startup or shut-down sequence longer than 0.5 seconds.’

In the program, the fixed framerate of 960 FPS is used to calculate the time on the X-axis. Although this gives us an approximate time it does not result in precise values. This is also something that can be improved by reading the framerate directly from the video.

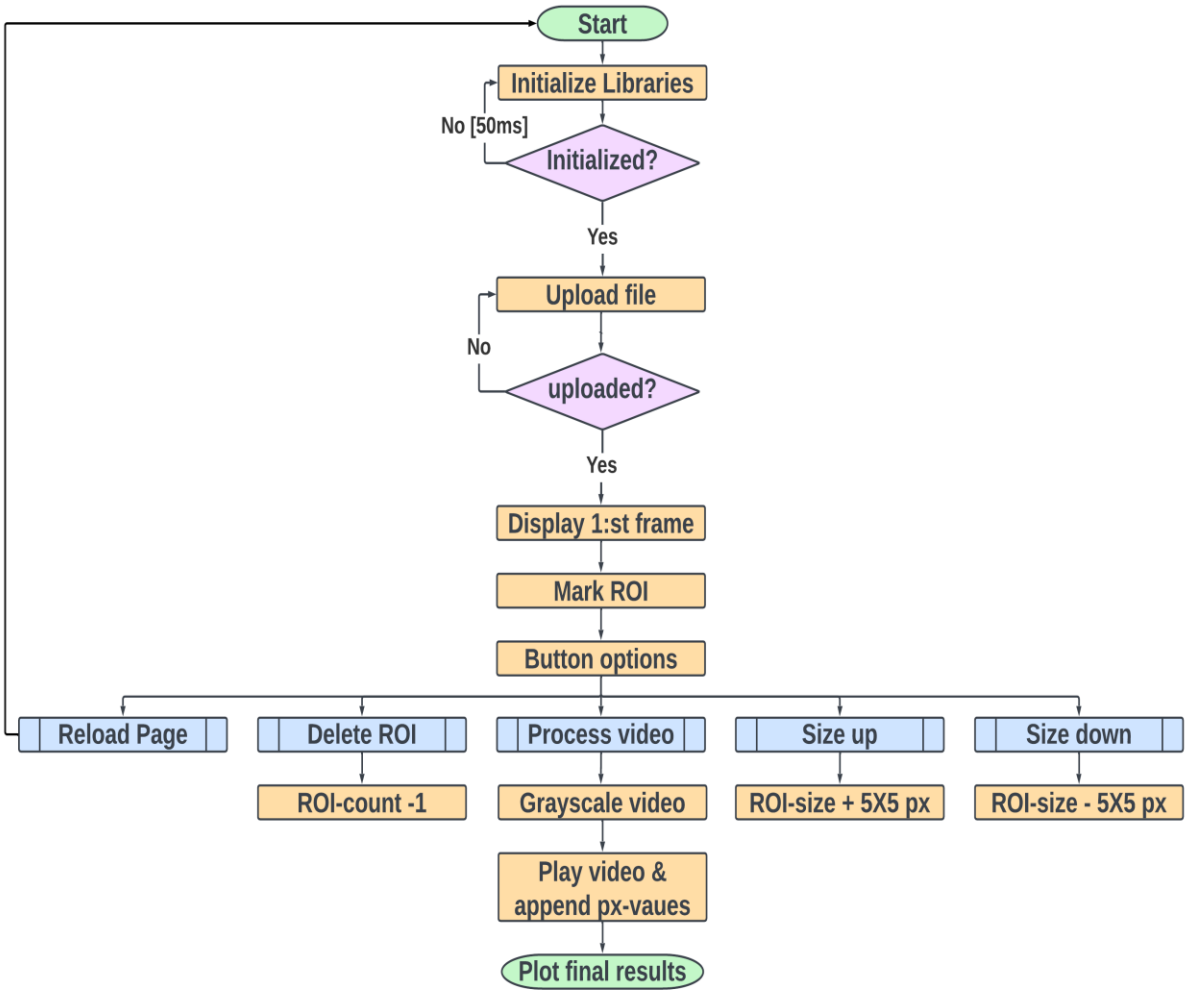
To confirm the functionality of the program tests should be done where the voltage sequencing timings of the application are compared with the timings of an oscilloscope. This would also confirm the functionality of the probe opto-modules as it would be easy to spot if one is not working as supposed to.

Developments could also be made to the user experience. The current version of the algorithm does not allow the user to draw their own ROI boxes, instead the ROI boxes have predefined sizes which

the user can choose from. The user can not draw ROIs in any other shape than squares either. This could be improved by implementing the ability to draw ROIs on the screen using a finger.

If time was not a delimitation an improvement could be to link the application together with the sequencing specifications to make confirmation even easier by displaying the result together with the specification. Another improvement would be to convert the pixel value on the y-axis to the actual voltage value, this does not only confirm the sequencing but also the voltage values from the specification.

Appendix A


Start
Initialize Libraries
No [50ms]
Initialized?
Yes
Upload file
No
uploaded?
Yes
Display 1:st frame
Mark ROI
Button options
Reload Page
Delete ROI
Process video
Size up
Size down
ROI-count -1
Grayscale video
ROI-size + 5X5 px
ROI-size - 5X5 px
Play video & append px-vaues
Plot final results

References

[1] Ericsson, “Optical probing of DC/DC converter voltages for sequencing verification”, thesis scope, unpublished.

[2] Ericsson,” Voltage sequence specification”, unpublished.

[3] Sverker Sander, private communication, 2023.

[4] Ericsson, “POMs”, unpublished.

[5] Web.dev, “Learn PWA,” [Online]. Available: https://web.dev/learn/pwa/ (accessed on: 2023-03-28)

[6] Visual Studio Code, “Learn to Code With Visual Studio Code”, 2023, [Online]. Available: https://code.visualstudio.com/learn (accessed on: 2023-03-28)

[7] W3schools “HTML Introduction”, [Online]. Available: https://www.w3schools.com/html/html_intro.asp (accessed on: 2023-03-28)

[8] W3schools “CSS Introduction”, [Online]. Available: https://www.w3schools.com/Css/css_intro.asp (accessed on: 2023-03-28)

[9] MDN, ”What is JavaScript?”, [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript (accessed on: 2023-04-04)

[10] OpenCV, “About”, [Online]. Available: https://opencv.org/about/ (accessed on: 2023-04-06)

[11] npm, “Fabric.js”, [Online], Available: https://www.npmjs.com/package/fabric (accessed on: 2023-04-06)

[12] Fabricjs, “Introduction to Fabric.js Part 1.”, [Online]. Available: http://fabricjs.com/fabric-intro-part-1 (accessed on: 2023-04-11)

[13] Plotly, “Plotly JavaScript Open Source Graphing Library”, [Online]. Available: https://plotly.com/javascript/ (accessed on: 2023-04-11)

[14] Python, “What is Python? Executive Summary”, [Online]. Available: https://python.org/doc/essays/blurb/ (accessed on: 2023-04-11)

[15] Coursera, “What is Python Used Fore? A Beginner’s Guide”, [Online]. Available: https://coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python (accessed on: 2023-04-11)

[16] OpenCV, “Operations on Arrays”, [Online]. Avalible: https://docs.opencv.org/3.4/d2/de8/group__core__array.html#ga191389f8a0e58180bb13a727782cd461 (accessed on: 2023-04-11)

[17] Web.Stanford, ”Image-6 Grayscale”, [Online]. Available: https://web.stanford.edu/class/cs101/image-6-grayscale.html (accessed on: 2023-04-11)

[18] Johan Nicander, private communication, 2023.

[19] nikkifurls, ”Simplepwa,” *github.com*, [Online], 03-19-2021. Available: https://github.com/nikkifurls/simplepwa (accessed on: 2023-03-20)

[20] Web.dev, “Service workers”, [Online]. Available: https://web.dev/learn/pwa/service-workers/ (accessed on: 2023-04-17)

[21] Samsung, ”Specifications”, [Online]. Available: https://www.samsung.com/global/galaxy/galaxy-s22/specs/ (accessed on: 2023-04-18)

Department of electrical engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

CHALMERS