NTNU

Kunnskap for en bedre verden

Bacheloroppgave

IE303612 Bacheloroppgave i automatisering/data

Web-based Data Visualization

120257 120258 130724

Totalt antall sider inkludert forsiden: 89

Innlevert Ålesund, 27/5/16

Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. **Manglende erklæring fritar ikke studentene fra sitt ansvar**.

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1.	**Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.**	☒
2.	**Jeg/vi erklærer videre at denne besvarelsen:** • ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • ikke refererer til andres arbeid uten at det er oppgitt. • ikke refererer til eget tidligere arbeid uten at det er oppgitt. • har alle referansene oppgitt i litteraturlisten. • ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	☒
3.	**Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen.**	☒
4.	**Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert i Ephorus, se Retningslinjer for elektronisk innlevering og publisering av studiepoenggivende studentoppgaver**	☒
5.	**Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter NTNUs studieforskrift.**	☒
6.	**Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider**	☒

Publiseringsavtale

Studiepoeng: 20

Veileder: Hans Georg Schaatun og Que Tran

Fullmakt til elektronisk publisering av oppgaven

Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven §2).
Alle oppgaver som fyller kriteriene vil bli registrert og publisert i Brage med forfatter(ne)s godkjennelse.
Oppgaver som er unntatt offentlighet eller båndlagt vil ikke bli publisert.

Jeg/vi gir herved NTNU i Ålesund en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	[✓ ja]	nei
Er oppgaven båndlagt (konfidensiell)? (Båndleggingsavtale må fylles ut)	ja	[✓ nei]
- Hvis ja: **Kan oppgaven publiseres når båndleggingsperioden er over?**	ja	nei
Er oppgaven unntatt offentlighet? (inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13)	ja	[✓ nei]

Dato: 27/05/16

BACHELOR THESIS - REPORT

HØGSKOLEN I ÅLESUND

Aalesund University College

TITLE:
Web-based Data Visualization

CANDIDATE(S): **120257, 120258, 130724**			
DATE: 27.05.16	COURSE CODE: IE303612	COURSE TITLE: BACHELOR THESIS	RESTRICTION:
STUDY PROGRAM: Automation Engineering		PAGES: 86 APPENDIX: 3	LIBRARY NO.:

SUPERVISOR(S):
Hans Georg Schaathun, Que Tran

SUMMARY:

The purpose of this report is to describe the process of creating web-based data visualization aimed at data collected from marine operations.

The work focuses on visualizing data in interactive graphs and illustrations. In addition, map and videos were implemented in order to enhance the understanding of the numeric data. These elements can give a better view of the ship's conditions and surroundings. All of the components are being synchronized and controlled by a common timeline. Data was stored in a MySQL database. The server-side language PHP was used for retrieving and providing the correct data for the different visualization elements.

The finished application is meant as a prototype for illustrating some of the possibilities for data visualization on the web. The idea is that Ulstein and the ONSITE-project can find inspiration from the application, and use it in further developments.

PREFACE

This report describes the accomplishment of our bachelor thesis. The bachelor thesis marks the completion of the course Automation Engineering at NTNU Aalesund.

Working on this thesis has been challenging, and has led to a greater understanding of data visualization, web development, interaction design and how to work as a team.

We would like to give our thanks to Ulstein Group and the ONSITE project for providing us with such an interesting thesis, and necessary IAS-data. Our contact at Ulstein, André Keane, has provided us with guidance and feedback through the project period.

We also want to thank our supervisors, Hans Georg Schaathun and Que Tran, for their valuable guidance throughout the project period.

In addition, we want to thank Snorre Hjelseth and Kjetil Nordby from AHO for giving us advice on how to develop good interaction design.

TABLE OF CONTENTS

Table of Figures

Table of Tables

ABSTRACT

The purpose of this report is to describe the process of creating web-based data visualization aimed at data collected from maritime operations.

The work focuses visualizing data in interactive graphs and illustrations. In addition, map and videos were implemented in order to enhance the understanding of the numeric data. These elements can give a better view of the ship's conditions and surroundings. All of the components are being synchronized and controlled by a common timeline. Data was stored in a MySQL database. The server-side language PHP was used for retrieving and providing the correct data for the different visualization elements.

The finished application is meant as a prototype for illustrating some of the possibilities for data visualization on the web. The idea is that Ulstein and the ONSITE-project can find inspiration from the application, and use it in further developments.

CONCEPTS

Visualization	Visual representation of data
HTML	A markup language for describing web pages
CSS	Style sheet language used for describing the presentation of the document written in a markup language (HTML)
JavaScript	The programming language of HTML and the Web
Software Library	Generally a collection of pre-written code, classes, procedures, scripts, configuration data and more that can be added to a program to achieve more functionality
API	Specific functions and methods that are present in a library
Client-server model	Client-server model is a program relationship in which the client sends requests and the server listens to those requests and responds the required task
Web Application	A client-server application running in a web browser

D3	JavaScript library for visualizing data
Canvas	JavaScript Charting Library for visualizing data
SVG	Scalable Vector Graphics is a two-dimensional XML based graphics format, with support for animation and interaction.
jQuery	JavaScript library designed to simplify the client-side scripting of HTML
AJAX	Technique for updating only parts of a web page
Popcorn	JavaScript library for creating time-based interactive media on the web
Leaflet	JavaScript library for creating interactive maps
Leaflet.hotline	A Leaflet plugin for drawing gradient along polylines
Google Maps	An online map application with its own JavaScript API
Windyty	Worldwide animated wind and weather map
SQL	A programing language for databases
MySQL	An open source database management system based on SQL
Localhost	Means this computer. Always refers to the loopback IP address. Is used to communicate with the local machine
PHP	Server-side scripting language
PDO	A PHP extension for interfacing with a database
Apache	Open source HTTP server
CSV	File format consisting of plain text with commas separating the values

JSON	A simple text based standard for data exchange
GeoJSON	Format for encoding a variety of geographic data structures, based on JSON
Software Architecture	The high level structure of a software system
StackOverflow	A collaborative questions and answer web site for software developers
Scrum	An agile project management framework for software development

ABBREVIATIONS

HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
D3	Data-Driven Documents
SVG	Scalable Vector Graphics
SQL	Structured Query Language
PHP	Hypertext Preprocessor (formerly known as Personal Home Page)
PDO	PHP Data Objects
AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
CSV	Comma Separated Values
JSON	JavaScript Object Notation
IAS	Integrated Automation System
AHO	The Oslo School of Architecture and Design
NTNU	Norwegian University of Science and Technology

1 INTRODUCTION

1.1 Background

The main topic of this thesis is data visualization, and the focus relies on solutions based on HTML5. HTML5 is interesting as a platform for data visualization because it can be used by anyone on any device with a modern web browser with no extra installations needed. Using the three cornerstone technologies of the web, HTML5, JavaScript and CSS, one can create interesting applications. HTML5 can be used as a platform and structure design, JavaScript for interactivity and CSS for styling. The use of data visualization seems to be a popular and growing field, and there are several visualization libraries for JavaScript that use HTML and CSS technology. It is now possible to visualize information in new ways that better relay information. These new ways of presenting data on the web can influence the cognitive psychology of people, and affect how data is interpreted.

Design and development of new ships build on the experiences from operations executed by earlier models. The behavior of ships can be documented by doing field studies. The operational data collected from field studies can be notes, videos or pictures. Context data collected from field studies are mostly large quantities of numeric data extracted from the ships IAS and DP systems. The challenge is how to interpret the context of numeric values from all the different sensors, as well as videos and notes. Rendering these data into well-presented visualizations can simplify the process of analyzing these data, making better use of all the information available. By understanding what happens to the ship during different operations, new solutions for improvements can be made.

1.2 Aim

Ulstein Group, as a part of the ONSITE project, assigned the issue for this bachelor thesis to NTNU Aalesund. ONSITE is a research project between NTNU Aalesund, AHO, DNV GL, Pon Power and Ulstein Group. The ONSITE projects' goal is to develop better methods for systematically documenting characteristics of earlier ship designs.

During this project, the group will explore the use of HTML5 for creating web-based data visualizations. Use of the D3 JavaScript library as a tool for visualizing various numeric data was suggested in the bachelor description. At the end of the project, the group will present a prototype of how different visualizations can be put together to create a broader picture of the data's entirety. Ulstein Group has not specified many demands regarding the accomplishment of this thesis. They would like to see something creative that can inspire new ways of analysing data. That made this an open assignment that was formed during the project.

We have formulated two challenges that apply for this project

- Determining whether or not HTML5 a suitable platform for data visualization
- Creating a prototype consisting of examples for data visualization on the web, aimed at data collected from maritime operations

1.3 Report Structure

This report will cover the process of exploring and mapping the possibilities for data visualizations on the web. It will also cover the possibilities of using video and map elements to enhance the understanding of the data. This project has consisted of practical work with many different components all being put together to a single application. Because of this, the group with advice from our academic supervisor, decided on following a custom layout in this report.

The first part of the report presents the different software technologies and materials that were used during development.

Chapters 4-7 will focus on the development of the different components created during the project period, as illustrated in *Figure 1* Overview of Technical Solutions.

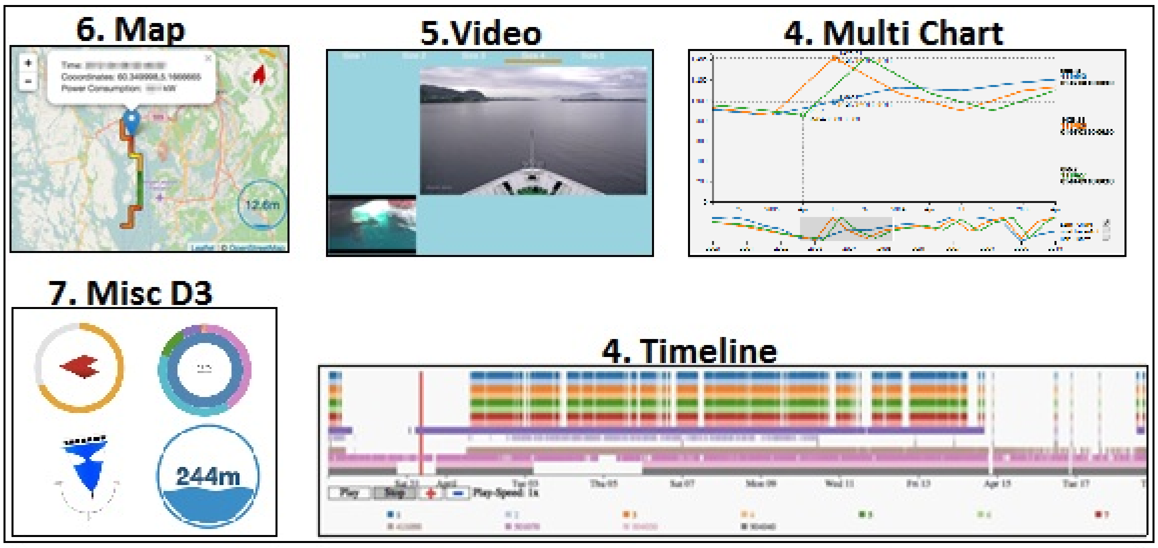


Figure 1 Overview of Technical Solutions

Chapter 8 will cover the integration of all the different components into a single application. Every chapter will contain a small subchapter for reviewing and discussing the component or components mentioned.

The development and the results of this project will be discussed and concluded at the end of the report.

2 SOFTWARE STANDARDS AND MATERIALS

In this chapter we will discuss the different aspects of the software platform we decided to use, and the parts they played in the development of the application. Project management will also be referred too, as well as the development platform and production platform.

2.1 Software Architecture

Software architecture regards all significant decisions that are hard to change, the organization of the software and the blue print of how each major part of the system is connected. It deals with larger aspects of the composition of the code.

Martin Fowler defines software architecture as decisions that are hard to change. “What is architecturally significant can change over a system's lifetime, and in the end architecture boils down to whatever the important stuff is." (Fowler, 2002).

Good software architecture should have loose coupling, empower the user and be adaptable to future changes. [1]

In the development of the project the group set up a three-tier architecture for the application. The three-tier architecture consists of the data tier at the bottom, the logic tier in the middle, and the presentation tier on the top. The data layer in this project is the MySQL database. This is where all the relevant data for the application is stored. The logic tier consists of PHP scripts that sends queries and operates on the data from the database tier before passing it to the presentation tier. The presentation tier is what the user is presented with when looking at the screen. This is where the JavaScript and HTML use the data provided by the logic tier to create visualizations.

Figure 2 illustrates the software architecture of this project:

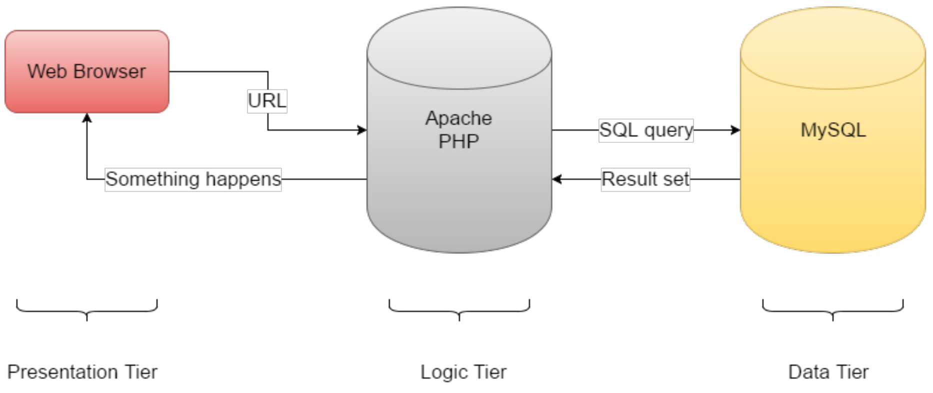


Figure 2 Software Architecture

[1] https://msdn.microsoft.com/en-us/library/ee658098.aspx

2.2 Software Technologies

2.2.1 HTML5

HTML is the language used for the presenting and structuring of content on the World Wide Web. HTML, CSS and JavaScript are the three cornerstone technologies of the Web.

HTML elements are the foundation of a web page, allowing for the implementation of text, videos, audio and other objects to be embedded on the web page. CSS is a style sheet language that handles the presentation of the HTML on the webpage. JavaScript is used to manipulate the elements on the HTML page. (Flanagan, 2011)

HTML5 introduced many new syntactic features to handle multimedia and graphical content on the web. There was also focus on cross-platform with mobile devices, with the inclusion of low energy features designed for smartphones and tablets.

HTML5 allow for the creation of interactive web applications. Since HTML5 runs on most modern web browsers, there are billions of users that can access such applications. HTML5 along with JavaScript has the largest outreach of any programming language. (Freeman, 2011)

HTML5 Media

HTML5 open up more possibilities for media to be embed natively in a webpage without the use of plugins. Displaying media on the Web have quickly adapted the HTML standard. The cross-platform adaptation ensures that videos can also be viewed on smartphones and tablets. The web browser can restrict the formats of the media on the page since not all browsers support the same codecs. (Pfeiffer, 2010)

HTML5 Video Support

HTML5 contains a video element for the playing of video and audio. Most modern web browsers, such as Chrome, Firefox, IE, Safari and Opera, support the HTML5 video standard. Browsers that usually do not support HTML5 video are older browsers. Even though most modern browsers support HTML5, not all of them support the same video codecs.

The WebM container format uses only the VP8 and VP9 video codecs and the Vorbis or Opus audio codecs. Chrome, Opera and Firefox support WebM. Safari and Internet Explorer can be supported with the install of an add-on.

The Ogg container format uses the Theora video codec and the Vorbis audio codec. Chrome, Firefox and Opera support the format. Safari can be supported by the install of an add-on. Internet Explorer does not support the Ogg container format by any means.

The MP4 container format uses the H.246 video codec and AAC audio codec. Internet Explorer, Safari and Chrome support it. Firefox can support the format in some cases, but only with a third party decoder. (Mozilla developers, 2016)

Using at least two of the file container types is recommended so all modern browsers will be supported. (Pfeiffer, 2010)

JavaScript

JavaScript is the most common programming language on the web. Having first class functions along with it having prototypical inheritance makes it supportive of both object oriented programming and functional programming styles. It does not contain any I/O functionality, such as networking and storage. JavaScript relies on the host environment to embed these. It is a client-side programming language that interacts with the HTML by manipulating its documents. It is what adds functionality to HTML elements. (Flanagan, 2011)

JavaScript Libraries

D3 Data-Driven Documents	D3 (Data-Driven Documents) is a JavaScript library for the creation of interactive and dynamic data visualization for the world wide web. It uses the HTML, SVG and CSS standards.[2]
jQuery write less, do more.	jQuery is a JavaScript library used to simplify the client side scripting of HTML. Is one of the most popular JavaScript libraries in the world. If often used in to simplify functions that would require more effort to create with plain JavaScript.[3]
Leaflet	Leaflet is a lightweight open source JavaScript library used for the creation of web based map applications. It has a well-documented API and an extensive catalog of plugins and extensions. It was used for the development of the Map application.[4]

[2] https://d3js.org/
[3] https://jquery.com/
[4] http://leafletjs.com/

	Leaflet.Hotline is a plugin that extends on the functionality of the Leaflet Function Polyline. It makes it easy to gradate the polyline in colors corresponding to a value.
Popcorn.js	Popcorn is a JavaScript library design for HTML 5 media development. It normalizes the HTMLMediaElement events, properties and methods into an API. Used in correlation of media centered web sites/applications.[5]

Table 1 JavaScript Libraries

CSS

Cascading Style Sheets is a language that is used to define the style of documents written in HTML or XML. The principle is that the HTML or XML document exclusively describes the structure and sematic of the web page, while the layout, colors and other styling information are described in the CSS. (Flanagan, 2011)

2.2.2 AJAX

AJAX stands for Asynchronous JavaScript And XML. It is a collection of web development techniques. It is used to update parts of a web page asynchronously by exchanging small amounts of data with the server behind the scenes. By using AJAX one can update parts of the web page without needing to refresh the whole page.[6]

2.2.3 SVG

Scalable Vector Graphics is a two-dimensional markup language that supports animations and interactions. Since it is vector-based, it will always remain its shape and definition when scaled, and will not become unclear like pixel-based image formats. It is well suited for use on the web, since it will retain its quality no matter what screen size and resolution it will be displayed at. SVG files can be created and edited in text format, though it is more commonly used with drawing software like Adobe Illustrator instead of a text editor. (Flanagan, 2011)

[5] http://popcornjs.org/
[6] https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started#What's_AJAX

2.2.4 MySQL and PHP

PHP and MySQL are used in order to create a dynamic, database-driven web page. MySQL with PHP is the most popular database system.

MYSQL is a database management system (DBMS) for relational databases. It is open-source and easy to use. MySQL uses a standard form of the SQL data langue. It is a database system that can be used on the web and runs on a server. It contains structured collections of data, normally text, numbers or binary files. These data are organized by the DBMS. The data are stored in tables consisting of columns and rows.

MySQL is the most popular database system used with PHP. The causes for the popularity is primarily that MySQL is easy to use, free and maintain a high performance. The reason PHP was chosen for the server side language is that it is easy to learn and offers a great deal of freedom. PHP has for a long time been seen as an important part of web development, and learning it seemed like a useful skill. There are a lot of good resources on the web for learning it.

PHP combined with MySQL are cross-platform. By incorporating a database into a web application, some of the data generated by PHP can be retrieved from MySQL. This further moves the site's content from a static basis to a flexible one, which leads to more dynamic web sites. (Ullman, 2012)

PHP ... is a widely-used Open Source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. (PHP, 2001)

This definition is long, but descriptive as to what PHP is. The following will give a more detailed description of the concepts included in the definitions. PHP is a *server side language, which* means that everything PHP does occurs on a server. A Web Server application, like Apache or Microsofts's IIS, is required and all PHP scripts must be accessed through a URL. That it is a *scripting language* means that the server reads the PHP code and then processes it according to its scripted direction. That PHP can be embedded into HTML means that you can drop in some PHP code wherever you need it in a standard HTML page. By using PHP tags, PHP code can be added to a page. (Ullman, 2012)

```
<?php
    PHP code;
?>
```

PDO

PDO is short for PHP Data Objects, and offers a robust method of interfacing with a database. Other extensions that could be used are MySQLi or the old MySQL. The old

MySQL extension is not recommended to implement in new web applications, so the choice was between PDO and MySQLi.

The reason why PDO extension was implemented in this project is that it is easier to change to another database later. PDO provides a data-access abstraction layer, which means that, regardless of which database you are using, you use the same functions to issue queries and fetch data. PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases. So, for switching to another database, PDO makes the process easy. The only changes that must be made are the connection string and a few queries. With MySQLi, the entire code must be rewritten – queries included. (Waterson, u.d.)

2.3 Development platform

2.3.1 Software

	AMPPS is a software bundle from Softaculous enabling Apache, MySQL, MongoDB, PHP, Perl, Python and Softaculous auto-installer on a desktop. This program was used for setting up the localhost that was used during the development. We used AMPPS for simulating the Apache server for a MySQL database, and accessing the data through PHP. phpMyAdmin is a graphical user interface that were used for managing and administrating the MySQL database.[7]
Web Browser	A web browser is an application for navigating the World Wide Web. The browser compiles and runs HTML codes. The main browser used in this project was Google Chrome.
Sublime Text	Sublime Text 2 is a source code editor that natively supports many languages, such as python, html and JavaScript. It is lightweight and easily accessible. It has a lot of features that make it a good tool for the development of code. Column selection, multi-select editing and syntax highlighting are some of the features.
Adobe Illustrator	Adobe illustrator is a vector graphics editor that allows for the creation and editing of SVG elements for the application.
Dropbox	Dropbox is a cloud-based file hosting service used for sharing files. The desktop application makes it easy to access files, through the file system of the operating system.

Table 2 Development Software

[7] http://www.ampps.com/

2.3.2 Hardware

The entirety of the project was developed on consumer end laptops. There was no need for expensive equipment. When the data amount increases an independent database should be acquired.

2.3.3 Project management

Project management regards finding techniques for organizing and conducting different tasks as a team in order to achieve specific goals. Project management concerns the processes such as planning, controlling and initiating in order to achieve those goals and to meet the project requirements.

No specific project management model was followed during this project. Since the scope of this project was broad and the group small, we decided that good communication between the members of the group, supervisors and contacts would replace the strict structure of most project models. That *Figure 3* illustrates

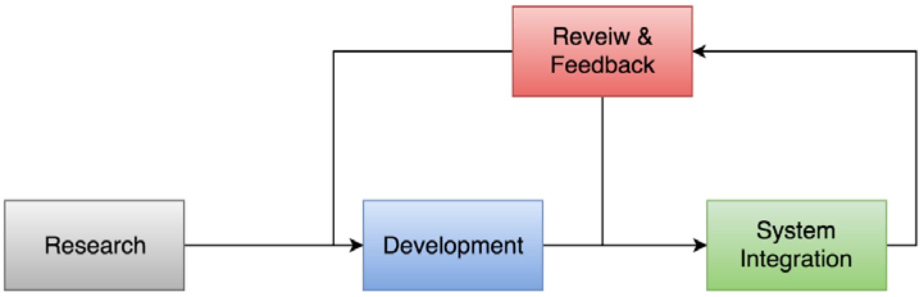


Figure 3 Project Workflow

Weekly meetings with our supervisor has been valuable in order to get feedback on the work done, and help setting a focus for further work. Also, feedback from both contacts at AHO and Ulstein, have been crucial. During this project, we have met with our contact at Ulstein four times, in addition to exchanging mails. We have also attended several Skype meetings both with Ulstein and AHO. These feedbacks along the way have helped us steer the focus of our work towards something that would be of interest.

The group worked in iterative stages, in a way that might resemble the Scrum model. Each group member would show of their progress to the others as soon as possible. This allowed for fast and efficient communication within the group, and ideas and concerns were addressed during the unveiling of the progress made. In the start of the project, members would work on their own specific tasks. As the project went along and each member’s prototypes got more and more refined, the frequency of having the other team members’ direct input, increased. When the system implementation phase started, the group worked even closer, and almost every development in the project from that point was a team effort.

There were little to no structured meetings between the group members during the project. The team instead worked together almost all the time. This tight and open working environment somewhat negated the need for meetings since decisions about the development would be dealt with as they came up. This working environment increased communication between the group, and made it easy to keep track of other group members' progress, obstacles and goals.

2.4 Production Platform

What is needed for running this application is:

- a web server
- a database that can be accessed through PHP's PDO
- a web browser

The group recommends using a MySQL database and Google Chrome as the web browser since that was used during development.

3 BACK-END

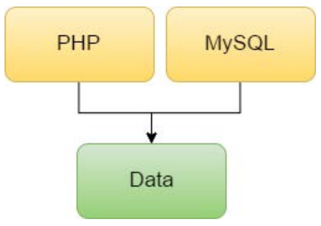


Figure 4 Back-End Related Technologies

The data being used in this project is collected from vessels. The data is collected from vessels' integrated automation systems (IAS). IAS provides integrated control and monitoring of vessels' different processes. We received two datasets during this project.

- IAS data from Anonymous PSV
- IAS data from Ulstein PSV

The two datasets differ quite a bit regarding several matters. The data from Ulstein have better integrity, which includes consistent values at a high sample rate. The main difference in the two datasets is the setup. The data from Ulstein consist of several signals listed in different columns so that each row has one point from each signal. The data from Anonymous PSV on the other hand, only have one signal per row, and uses one column to identify the signal.

As described in chapter *2.2.4*, the datasets have been implemented into a local database, and accessed with PHP scripts. Some JavaScripts load the PHP scripts in order to use and implement the data in visualization elements. In this chapter, we will describe details about the data and how they are setup, and how the data is retrieved and implemented in the different visualization elements.

Data that give information about power consumption, location, weather conditions have been essential. Below is a table that describes which data that were implemented in which elements of the application.

Element	Data
Timeline	all data that is visual on the website
Multi Chart	selected IAS data from database
Map	latitude and longitude + total power consumption
Simple Line Graph	power consumption
Sunburst	power consumption
Ship SVGs	pitch and roll
Arrow and Radial Bar	wind direction and wind speed
Liquid Fill Gauge	water depth

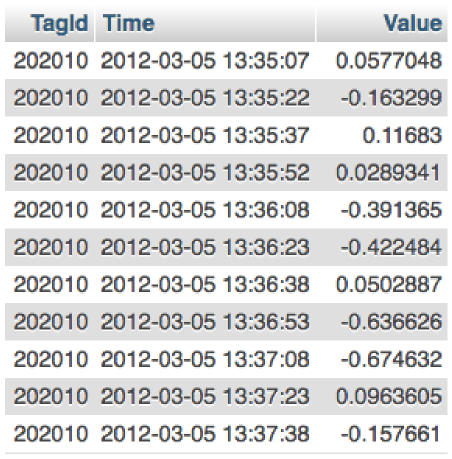
Table 3 Data for Visualization Elements

3.1 Data from Anonymous PSV

The first dataset the group got access to was through the university, which had access to a database containing data from a PSV vessel. It was IAS-data collected in 2012, and with little known information about what vessel the data belonged to. The reason why we could access the data so easily, was because of the lack of information about the data's origin. The dataset contained IAS data from the vessel. The data was sampled at a 4-minute rate, and contained some NULL-values and some missing values. Some data points did not contain any data. The data integrity was therefore not optimal.

This data was very useful to get started with, and was implemented in the visualizations that were developed in order to get a clear view of how they work. The data was essential for the creation of the back-end solution in the project.

The data was set up in two tables as shown below:

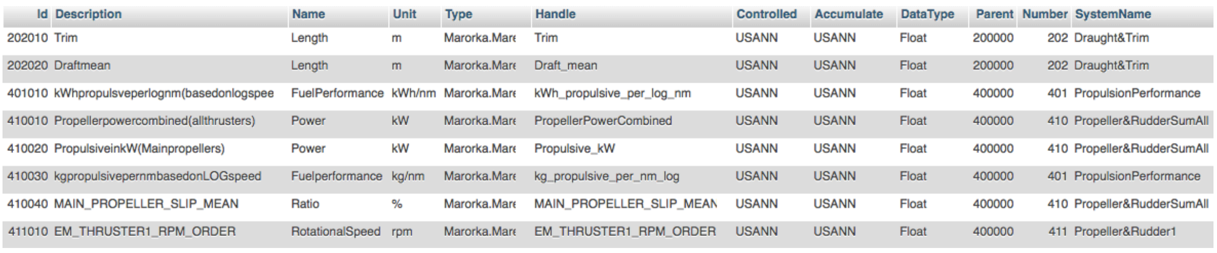


TagId	Time	Value
202010	2012-03-05 13:35:07	0.0577048
202010	2012-03-05 13:35:22	-0.163299
202010	2012-03-05 13:35:37	0.11683
202010	2012-03-05 13:35:52	0.0289341
202010	2012-03-05 13:36:08	-0.391365
202010	2012-03-05 13:36:23	-0.422484
202010	2012-03-05 13:36:38	0.0502887
202010	2012-03-05 13:36:53	-0.636626
202010	2012-03-05 13:37:08	-0.674632
202010	2012-03-05 13:37:23	0.0963605
202010	2012-03-05 13:37:38	-0.157661

One table named *data* containing three columns named *TagId, Time* and *Value*:

Figure 5 Table: data

Another table named *name* containing columns named *Id, Description, Name, Unit, Type, Handle, Controlled, Accumulate, DataType, Parent, Number, SystemName*:



Id	Description	Name	Unit	Type	Handle	Controlled	Accumulate	DataType	Parent	Number	SystemName
202010	Trim	Length	m	Marorka.Mare	Trim	USANN	USANN	Float	200000	202	Draught&Trim
202020	Draftmean	Length	m	Marorka.Mare	Draft_mean	USANN	USANN	Float	200000	202	Draught&Trim
401010	kWhpropulsveperlognm(basedonlogspee	FuelPerformance	kWh/nm	Marorka.Mare	kWh_propulsive_per_log_nm	USANN	USANN	Float	400000	401	PropulsionPerformance
410010	Propellerpowercombined(allthrusters)	Power	kW	Marorka.Mare	PropellerPowerCombined	USANN	USANN	Float	400000	410	Propeller&RudderSumAll
410020	PropulsiveinkW(Mainpropellers)	Power	kW	Marorka.Mare	Propulsive_kW	USANN	USANN	Float	400000	410	Propeller&RudderSumAll
410030	kgpropulsivepernmbasedonLOGspeed	Fuelperformance	kg/nm	Marorka.Mare	kg_propulsive_per_nm_log	USANN	USANN	Float	400000	401	PropulsionPerformance
410040	MAIN_PROPELLER_SLIP_MEAN	Ratio	%	Marorka.Mare	MAIN_PROPELLER_SLIP_MEAN	USANN	USANN	Float	400000	410	Propeller&RudderSumAll
411010	EM_THRUSTER1_RPM_ORDER	RotationalSpeed	rpm	Marorka.Mare	EM_THRUSTER1_RPM_ORDER	USANN	USANN	Float	400000	411	Propeller&Rudder1

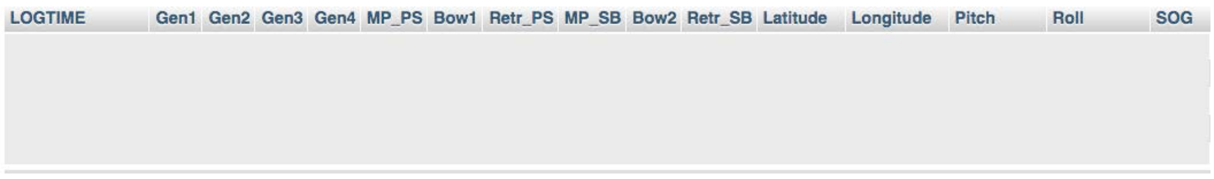
Figure 6 Table: name

The way this setup works is that data from each sensor on the ship has its own TagId/Id. The TagId is logged at every timestamp along with the value of the data. The TagId in the Data table is then matched up with the Id in the Name table in order to get additional information about the data.

3.2 Data from Ulstein

The data we received from Ulstein was from a field study from one of their vessels. This dataset had a sample rate of 5 seconds, and contained no missing values. The dataset we received was only a small part of a larger dataset collected from the IAS system. The group mainly received this dataset because it contained position data. The dataset also included data for power consumption from generators and data for pitch and roll, which was implemented in our visualization. This data is confidential.

The data from Ulstein had a different setup than the first received data. The setup of the data from Ulstein is shown on *Figure 7*:



LOGTIME	Gen1	Gen2	Gen3	Gen4	MP_PS	Bow1	Retr_PS	MP_SB	Bow2	Retr_SB	Latitude	Longitude	Pitch	Roll	SOG

Figure 7 Ulstein Data setup

All the data are contained in one table. The data are ordered by columns and sampled at the same timestamps. The name of the columns describes what signal it is.

There are advantages and disadvantages to both setups. By using an ID combined with the data, more information and descriptions can be supplied to the data. By having the data in separate columns, instead of in one joined "Value" column, it may sometimes be easier to retrieve the desired data using PHP.

3.3 Combined Data

To make our vision for visualization work, we needed the following data:

- position data(latitude and longitude)
- power data(power consumption from generators/thrusters)
- weather data(wind speed, wind direction, wave height/water depth)
- data for pitch and roll
- videos that are collected in a similar timeframe

We have received all these data, but not from the same source. In order to create a timeline that consists of all these data synchronized, we needed to combine them. Since the data from the two sources were from different years, it would not be optimal to view them in the same timeline. Since we needed data from both the received datasets, we needed to find a way to make common timestamps for the two datasets.

Since the setup determines how the data are retrieved with PHP, we had to choose one of the setups to continue with. The setup for the first data was kept. The data from Ulstein had to be adapted to that setup. This was executed by combining the data we needed from Ulstein with the timestamps from the data from Anonymous PSV. Finally, unique IDs were added to each data point.

3.4 Retrieving Data

The work done in this chapter is based on the guide (A Web Developing Cat, 2012).
Further in this chapter, the JavaScripts that create different visualizations will be called client-side-scripts and the PHP scripts that retrieve data will be called server-side-scripts. In this subchapter, the process of what happens when the user requests a new dataset will be explained and reviewed.

The relationship between the GUI(form), the client-side-scripts and the server-side-scripts and is shown *Figure 8* Data Retrieval Flowchart***.***

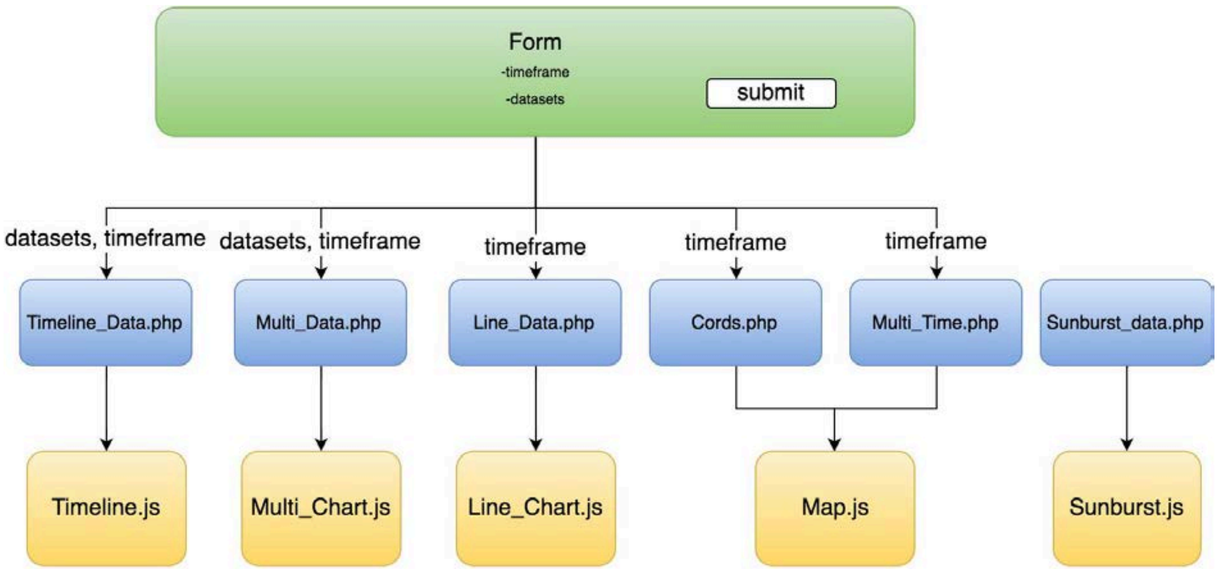


Figure 8 Data Retrieval Flowchart

The client-side-scripts import the server-side-scripts in order to retrieve and use the correct data in the visualization elements.

3.4.1 GUI

The first server-side-script we created was for the client-side-script that contained the Multi Chart. Some basic functionality that was implemented in the server-side-script is that the user can:

1. Select which datasets to view in the multi chart
2. Select start date and time interval

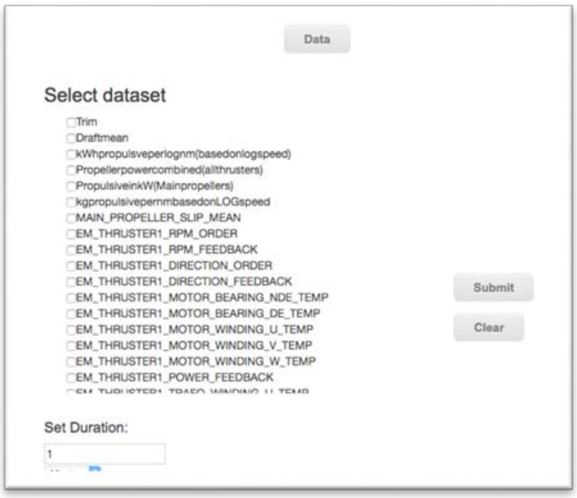
This functionality lets the user limit the amount of data that is implemented in the data visualization. The reason why it was important to establish this functionality is that the database contains large amounts of data that can be both heavy and time consuming for the visualization components to run. The components run perfectly with a small set of data, but when the amount of data increases, they use more time to build the visualizations and some functionality such as zooming and mouse over interactivity lags.

Another aspect was that it is very handy is to have the opportunity to select which data to visualize from the database. By doing this, it will be possible for users to go back in time and view data collected from a field study, and figure out what happened at a given time. This lets the user view desired data in a desired timeframe.

Form

In order to create the mentioned functionality, a form was implemented. The form lets the user select the desired datasets and set a time frame for those data. The form appears on the website when the *Data* button is pushed.

The two screenshots below illustrate how the form works.



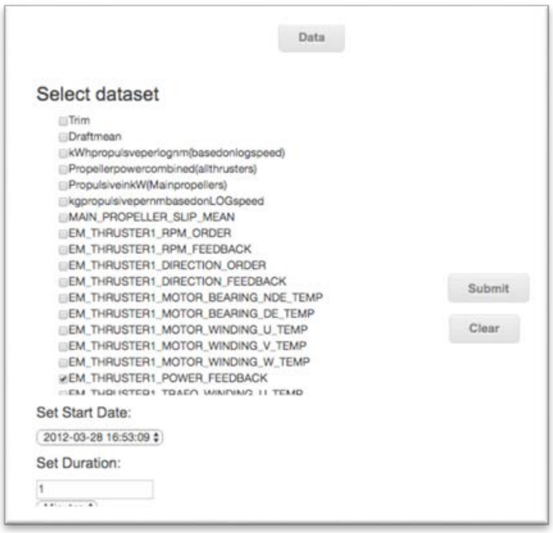
Select dataset

- ☐ Trim
- ☐ Draftmean
- ☐ kWhpropulsveperlognm(basedonlogspeed)
- ☐ Propellerpowercombined(allthrusters)
- ☐ PropulsiveinkW(Mainpropellers)
- ☐ kgpropulsivepernmbasedonLOGspeed
- ☐ MAIN_PROPELLER_SLIP_MEAN
- ☐ EM_THRUSTER1_RPM_ORDER
- ☐ EM_THRUSTER1_RPM_FEEDBACK
- ☐ EM_THRUSTER1_DIRECTION_ORDER
- ☐ EM_THRUSTER1_DIRECTION_FEEDBACK
- ☐ EM_THRUSTER1_MOTOR_BEARING_NDE_TEMP
- ☐ EM_THRUSTER1_MOTOR_BEARING_DE_TEMP
- ☐ EM_THRUSTER1_MOTOR_WINDING_U_TEMP
- ☐ EM_THRUSTER1_MOTOR_WINDING_V_TEMP
- ☐ EM_THRUSTER1_MOTOR_WINDING_W_TEMP
- ☐ EM_THRUSTER1_POWER_FEEDBACK

Set Duration:

1

Figure 10 Form Before Selections



Select dataset

- ☐ Trim
- ☐ Draftmean
- ☐ kWhpropulsveperlognm(basedonlogspeed)
- ☐ Propellerpowercombined(allthrusters)
- ☐ PropulsiveinkW(Mainpropellers)
- ☐ kgpropulsivepernmbasedonLOGspeed
- ☐ MAIN_PROPELLER_SLIP_MEAN
- ☐ EM_THRUSTER1_RPM_ORDER
- ☐ EM_THRUSTER1_RPM_FEEDBACK
- ☐ EM_THRUSTER1_DIRECTION_ORDER
- ☐ EM_THRUSTER1_DIRECTION_FEEDBACK
- ☐ EM_THRUSTER1_MOTOR_BEARING_NDE_TEMP
- ☐ EM_THRUSTER1_MOTOR_BEARING_DE_TEMP
- ☐ EM_THRUSTER1_MOTOR_WINDING_U_TEMP
- ☐ EM_THRUSTER1_MOTOR_WINDING_V_TEMP
- ☐ EM_THRUSTER1_MOTOR_WINDING_W_TEMP
- [✓ EM_THRUSTER1_POWER_FEEDBACK]

Set Start Date:

2012-03-28 16:53:09

Set Duration:

1

Figure 9 Form with Start Date Options

The first screenshot is the first sight that meets the user. The user selects the desired datasets by clicking on the checkboxes, and then pressing the submit button.

When this is done, the next screenshot appears. What has happened is that the server-side-script for the form has collected all available timestamps for the selected datasets. The reason why this is done is so that the user cannot select a time frame where no data exists. Now, the user can set a start date and duration for the data. The start date is selected in a dropdown menu. The duration is set by typing a number and selecting a time unit from the dropdown menu. When the submit button is pressed again, all server-side-scripts will be updated with new data.

3.4.2 Checking the Conditions

The form is contained in a function called getForm() that appears when the Data button is pressed. Inside the getForm() function, the functions getNames(), getTime() and getUnit() appears. getNames() contains an array that contains the selected datasets. By using the $_GET() and isset() functions, the getForm() function checks if the array returned from getNames() contains values. The getForm() calls the getTime() function if the conditions are met.

```
if(isset($_GET['dataset'])){
    getTime();
}
```

Further, the same logic applies for the getTime() and getUnit() functions.

There are several server-side-scripts that checks if the parameters for the form is set. Those server-side-scripts are Timeline_Data.php, Multi_Data.php, Cords.php and Map_Time.php. These scripts check if the parameters are set and conditions are met in the same way as the getForm() function does.

3.4.3 SQL Query

The way the server-side-scripts get the data is by sending SQL queries to the database requesting the specific data. Below is an example of such a query.

```
try{
  $stmt = $db->prepare("SELECT DISTINCT data.Time, data.Value, data.TagId
    FROM data
    WHERE data.TagId IN(".implode(',',$id).") AND
     data.Time BETWEEN :time AND DATE_ADD(:time, INTERVAL :dur MINUTE)
     AND data.Value IS NOT NULL
     ORDER BY data.Time
   ");
      $stmt->bindParam(':time', $time);
      $stmt->bindParam(':dur', $dur);
      $stmt->execute();
      $data = $stmt->fetchAll(PDO::FETCH_ASSOC);
     }
      catch(PDOException $ex) {
      echo $ex;
  }
```

Take notice of the keywords *SELECT*, *FROM* and *WHERE*. These keywords form the basis of how data is retrieved from the database. What happens in this code snippet is that the columns named *Time*, *Value* and *TagId* are retrieved from the table named *data*. The columns are collected between the start date and duration parameters set by the user. It is data that are selected in the dropdown menu that are retrieved. This is where we use the TagId to check which datasets where selected by the user. Only data that does not contain null-values are retrieved, and then the Time-column orders them.

Detail: the implode function is used to convert the int value of TagId to a string, so that the JavaScript can use it.

3.4.4 Data for each Element

Now that it is known how the form works, and how SQL queries takes place, we can take look at how the data is retrieved and applied for the different visualization elements.

Multi Chart

The Multi_Chart.js loads the data from the server-side-script as a CSV file by using the d3.csv() function. The location.search contains the URL that is being set when the user have made the selections on the form.

```
d3.csv("data/Multi_Data.php"+location.search, type, function(error, data) {
var negativeValues = false;
 data.forEach(function(d){
     if(d.price < 0){
      negativeValues = true;
     }
 });
```

Result output:

```
id,date,price     411090,28-Mar-12-16-53-09,37.8     411070,28-Mar-12-16-53-09,41.4
411090,28-Mar-12-16-57-10,39.6   411070,28-Mar-12-16-57-10,40   411090,28-Mar-12-17-
01-10,39  411070,28-Mar-12-17-05-11,37.8  411090,28-Mar-12-17-05-11,40.2  411090,28-
Mar-12-17-09-12,37.1   411090,28-Mar-12-17-13-12,37   411070,28-Mar-12-17-13-12,39.1
411090,28-Mar-12-17-17-13,35.8 ...
```

Some date parsing had to be done in order to get the correct format in the client-side-scripts. The SQL timestamp had to be set to a Unix timestamp. The first thing that needed to be done was to set the timezone settings in the data file. This was set to Oslo/Europe. Second, the strtotime() function is used. The strtotime() function can parse about any English textual datetime description into a Unix timestamp. Finally, the data must be parsed to a D3 datetime object in JavaScript.

Map

When working with map APIs, the formats of the data have mostly been preferred in GeoJSON, TopoJSON or as multi-dimensional arrays. Since the data in this project are retrieved from a MySQL database, the easiest way seemed to be to output the data as arrays.

Two arrays are being output as results from two separate server-side scripts: Cords.php and MapTime.php. The Cords.php outputs an array containing data for latitude, longitude and total power consumption from generators. The MapTime.php collects all timestamps for the data in Cords.php. The timestamps are needed in order to set the marker on the map in accordance with the timeline. The reason why there are two separate arrays is that the first array did not work when the timestamps were added in the same one.

Cords.php

Since the data is setup with all values in the same column, and separated by IDs, multiple functions had to me made for retrieving for latitude, longitude and power. Three functions were made: getLatitude(), getLongitude() and getPower(). Below is the getLatidue() function shown.

```
function getLatitude(){
    global $db,$time, $dur;
    try {
      $stmt = $db->prepare('SELECT data.Value
        FROM data
    WHERE data.TagId = "5" AND data.Time BETWEEN :time AND DATE_ADD(:time, INTERVAL
:dur MINUTE)
     AND data.Value IS NOT NULL
     ORDER BY data.Time
    ');
      $stmt->bindParam(':time', $time);
      $stmt->bindParam(':dur', $dur);
      $stmt->execute();
      $data = $stmt->fetchAll(PDO::FETCH_ASSOC);
     }
      catch(PDOException $ex) {
      echo $ex;
    }
    $lat_array = array();
    foreach($data as $val){
      $lat = $val['Value'];
      $lat= str_replace(',', '.', $lat);
      array_push($lat_array, $lat);
    }
    return $lat_array;
  }
```

The way it works is that a SQL select query is made, retrieving the data from Value where the TagId = 5. The data between the selected start date and time interval is retrieved. All the values are being pushed into an array, and the decimal separator in the values are changed from "," to ".". At last, the array containing the values is returned. The same happens for the functions getLongitude() and getPower().

What happens next is that all the 3 arrays are being joined to one array named $joined, and then returned as an array containing JavaScript objects. The json_encode() function is used to return the values in the array with a JSON representation.

MapTime.php

The MapTime.php outputs an array containing all the timestamps for the data that is selected in Cords.php. The timestamps are the ones that are logged with the latitude values. The reason why this works is that it is known that longitude and power have the same timestamps.

Timeline

The server-side-script for the timeline is exactly the same as the one for multi chart. The only difference is that some values that are meant to be displayed at the timeline at all times, are added. These values are added into the server-side-script by making the query always select the values that belong to the following IDs:

ID	DESCRIPTION
1	Gen1
2	Gen2
3	Gen3
4	Gen4
5	Latitude
6	Longitude
7	Pitch
8	Roll
904030	EM_out_WIND_DIR_REL
904040	EM_out_WIND_SPEED_REL
501070	WATER_DEPTH

Table 4 IDs for Timeline Elements

The table shows the values for components that are always visible on the web page, so it can be useful to see if they are containing values or not at all times.

Sunburst

The server-side-script for the sunburst does not collect data within the whole time frame set by the user in the form. Sunburst needs a dataset consisting of descriptions and values at a given time. It is basically intended to be a static figure. The reason for this is that sunburst uses the description from the data to build and draw the component. This description had to be added in the database. That means that every signal that is going to be displayed in a sunburst, need to have a description added to it.

In this case, we had to create this description and match it to the associated IDs. For instance, the description for ID=1 is Power-Gen1.The output from the server-side-script for sunburst looks like this:

```
Power-Gen1,-2 Power-Gen2,736 Power-Gen3,759 Power-Gen4,3
```

Power describes the inner ring, and Gen1, Gen2, Gen3 and Gen4 describes the outer arc elements that form the second ring of the component.

3.5 Information Security

Data security, information security or IT security is a discipline connected to the key terms confidentiality, integrity and availability. It regards the practice of defending information.

Confidentiality
Integrity
Information Security
Availability

Figure 11 CIA Triad

Data integrity describes the accuracy and consistency of the data. Integrity applies for ensuring that the information is correct, valid and complete.

Data availability refers to how one can access the data. It concerns ensuring that information is available within those availability demands that are set.

Confidentiality applies for ensuring that information only is available for those who are intended to have access to the data. (wikipedia, 2016)

What had to be considered regarding information security was to keep any corporate secrets and to protect the privacy rights of employees and operators. The information security becomes more important when the amount of data increases since it will be easier to find context from the data. The data could reveal sensitive information like position and time. The data could also reveal human errors during operations, something that can make owners, staff or others involved with the operation look bad.

Information security has not been thoroughly handled in this application, since the development has focused on running the application in a closed network. Anyone who has access to the source code of this application will have access to the data. The PHP solution currently connects to the database with a full access account. This is something that should be changed to a read-only account. Additionally, we could add further restrictions by only allowing the host server to access the database.

Another issue regarding information security is SQL injections. SQL injection is a technique where users can inject SQL commands into SQL statements via web page inputs. By doing this, attackers may extract, manipulate or delete information from the database. It exists methods for protecting the database against SQL injections with PHP, and this is something worth looking into if high protection is necessary.

3.6 Review

The back end solution described in this chapter is well suited for the application, but has room for improvement. We are satisfied with the choice of using a MySQL database with PDO connection and PHP as scripting language, since it was easy to learn, use and modify. The connection and SQL queries work well for the application. Since our task was to create and find possibilities for data visualization on the web, our main focus has been on front-end web development. We wanted to retrieve data from a database, since it would add functionality to the application to make it more dynamic. The back-end solution as it is now, works for letting the user retrieve a new dataset and refresh all the components on the web page. Still, there are several aspects that could be improved or extended.

The way the user makes inputs is through the form described in *3.4.1*. This form consists of the desired functionality, but could have a more user-friendly appearance. Some immediate changes that may improve the usability of the form is by putting the checkboxes in a dropdown menu, and by letting the user input the time frame by selecting date and times from a calendar. Also, sorting the data in relation to each other and separating them as groups, could help the user orient in the data. Instead of listing all available dataset from the database in a long, long list.

The different data setups have proved to have their advantages and disadvantages. The setup of the data from Anonymous PSV gives more information that can be valuable. The unique IDs also proved to be very useful in order to separate signals. It was useful in the form in order to let the user select the desired datasets. The IDs was necessary for the multi chart in order to separate the different signals, and plot them as separate lines. When working with the map, the IDs in the setup made it troublesome.

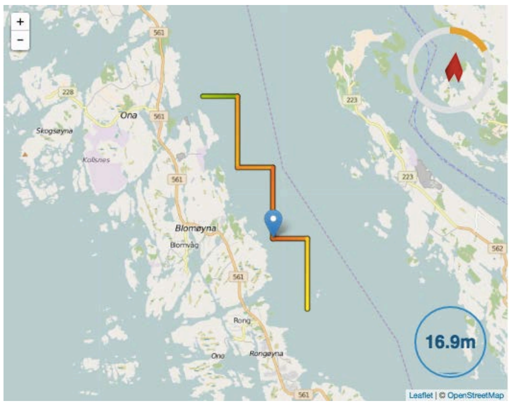


Figure 12 Map with Zigzag Tracing

The solution of providing data for the map as arrays is less than optimal. The reason why the solution works is that it is known that there are values at all times, and that they all have the same timestamps. If this were not the case, the data would either be cut or skipped in order to all fit in the same array. The fact that the signals were listed in the same column in the database, and separated by IDs, made things complicated. If the signals were stored in separate columns, it would be easier to retrieve the data and push them directly into a single array. The solution now requires a function for each signal that returns the values as an array. Each array is then being pushed to a joint array.

The reason why the tracing is zigzag is that the data for latitude and longitude are logged more often than they are updated. Latitude is updated before longitude, and that is the reason why the tracing is zigzag. The actual path is on the right corners of the tracing. The group has not investigated this issue since our focus has been to explore and identify opportunities for visualizing different data. This seems to be a challenge regarding how the data is sampled.

4 GRAPHICAL REPRESENTATION OF DATA

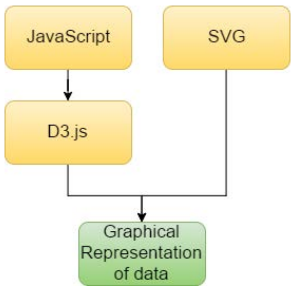


Figure 13 D3 Related Technologies

To visualize some of the data we received, we took advantage of the D3 library for JavaScript. The first we heard of D3 was from the project description given to us by Ulstein Group and the ONSITE-project. Using the D3 library had quite a steep learning curve. Luckily, D3 has a large community owning up to its open-source status. Thousands of questions have been answered on StackOverflow, which provided us with much needed guidance. D3 also has a gallery with more than 900 examples for inspiration. These examples range from the very basic, to the extremely sophisticated and often interactive.

The work on the multi chart began in the startup phase of the project. At that point we did not have much experience with D3 and did not know what was realistic to achieve. The work started simple by trying to create a line chart and work our way from there. We took inspiration from a lot of different examples made public but also created our own solutions to problems, as they occurred. *Figure 14* shows the final result of the multi chart.

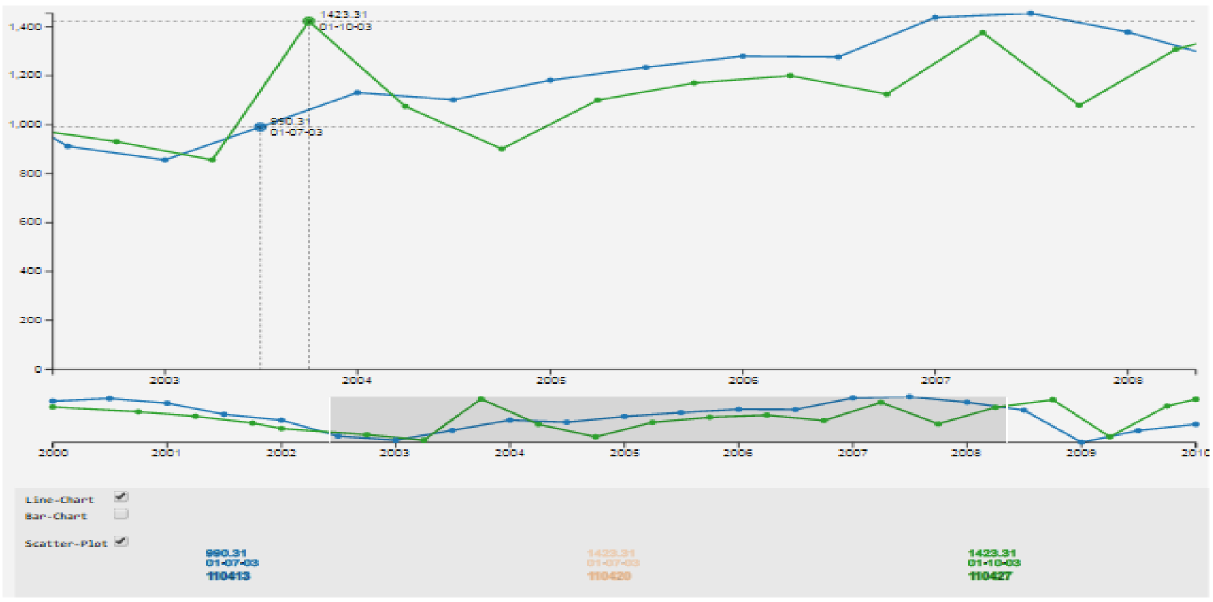


Figure 14 Complete Multi Chart

The multi chart takes advantage of the zoom, pan and brush behavior. Tooltips lock on to the closest segments of data on each line, relative to the position of the mouse pointer. The user can select which data to display by clicking the legends in the bottom of the element. By clicking the checkboxes, the user can decide if the data is shown as a line chart, scatter plot or bar chart.

With the multi chart finished and a lot of other components starting to take shape, we wanted a way to connect them all together. This idea formed the basis for the timeline. As well as connecting components together, the timeline should serve to give the user a larger picture of the data. In the timeline, a bar represents every point of data. This allows the user to see exactly which data is available, or what the sample rate of the data is, at any given time. A large red bar referred to as the time bar, serves as the reference point in the timeline. Clicking the timeline or using the play function of the timeline, will update the position of the time bar. *Figure 15* shows the result of the timeline.

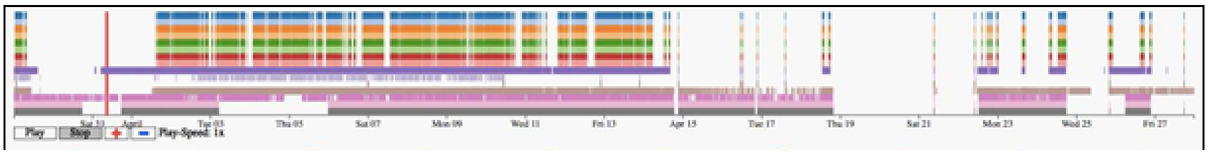


Figure 15 Timeline Complete

4.1 *Multi Chart*

4.1.1 Creating a Path

The process of making a line chart that plots a single path was pretty straightforward and a Google search displayed bundles of relevant examples. This solution was inspired by an example provided by Mike Bostock[8].

The first step was creating a SVG element and defining the width, height and margins of the element. The next step was making variables that set the scale of our chart. These variables are typically referred to as x and y, and represent the x- and y-scale. The purpose of these variables is to make sure SVG element will be scaled to fit the entire length of our data. The domain of the x and y is set when the data is loaded. In our case where the x domain will be represented by dates, the data representing timestamps first needs to be stored as JavaScript date objects. The string value is converted into a date object by the d3.time.format(specifier) constructor. The specifier parameter tells how the date string is formatted. A timestamp like "Jan 2000" in the data would correspond to a ("%b %Y") specifier. Where %b represents the abbreviated month name and %Y represent year with century as a decimal number. Since the x domain will represent the entire length of time our data is sampled, we applied the d3.extent()

[8] https://bl.ocks.org/mbostock/3883245, Updated: (Jan 13, 2016), Author: Mike Bostock

function. This function returns the highest and lowest date from the data and these values are used for our x domain. The y domain is represented by numeric values. Unless the data contains negative values, using the d3.extent() function in the y domain would give an incorrect picture of the data. The values would be correct but it is considered best practice to display the data from zero at the bottom of the y domain to the largest value amongst the data, at the top. In order to find the largest value, the function d3.max is used. We also had to check for negative values and in case of negative values in the y domain, it should be defined by the d3.extent() function. This is to ensure that no values are left out of our scope. Making sure the x- and y-scales are correct is a crucial part of making the line chart. In order to create paths on our SVG element, we simply select it and append a "path".

The path element represents the outline of a path that can be used as a clipping path, filled, stroked or any combination of the previous. The "d" attribute is what defines the path data. To simplify the construction of the "d" attribute D3 comes with a number of helper functions for generating path data. The path generator we used was the d3.svg.line() function. By setting .x and .y of the line() we can call this path generator where we set the "d" attribute of the path element. The path will then be created as a single element for the entire path between all our data points.

To put our data better into context, we constructed an x- and y-axis. Luckily D3 comes with an axis generator as well. We used the d3.svg.axis() generator to create both the x- and y-axis. This generator also comes with some additional and optional parameters. In our multi chart component we used d3.svg.axis().scale(x).orient("bottom") this way the x-axis is scaled with respect to x-domain and oriented in the bottom of our window. For the y-axis the approach is the same d3.svg.axis().scale(y).orient("left"), the only difference is that the y-axis is being scaled with respect to y-domain and oriented to the left. Attributes like .ticks, which is the number of ticks on the axis or .tickFormat, which is what the function name suggests, can also be added. By default, the number of ticks is set to 10. With x- and y-domain defined and the path and axis created, we can have a look at the result in *Figure 16*.

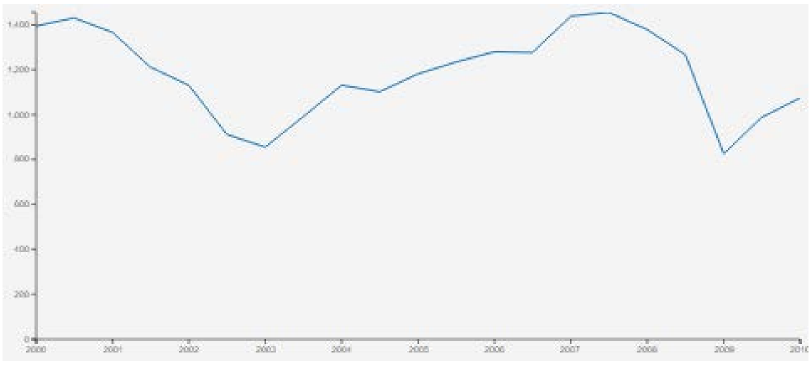


Figure 16 Simple Line Chart

4.1.2 Creating Multiple Paths

Creating multiple paths took some searching for similar examples. How this component handles the problems of creating multiple paths is based on example posted by D3noob[9].

The big difference when creating multiple paths is that the component has to be able to differentiate between the data. If the component is unable to differentiate between the different data, it will only create one path from all the data and it will probably not make much sense. In order to solve this problem we took advantage of the d3.nest() function. d3.nest() is about taking a flat data structure and turning it into a nested one. This lets us break apart the data by a categorical variable and look at the statistics or details for each group. Because of the way our early data was structured we decided to use nest to group the data by their Id. In the example below it is shown how nesting can be used to change the structure of the data.

```
var data = [{id: 110413, date: “Jan 2000”, price: 1394.46},
           {id: 110413, date: “Jan 2001”, price: 1323.46},
           {id: 110413, date: “Jan 2002”, price: 1310.46},
           {id: 110420, date: “Jan 2000”, price: 1100.46},
           {id: 110420, date: “Jan 2001”, price: 630.46},
           {id: 110420, date: “Jan 2002”, price: 880.46},
           {id: 110427, date: “Jan 2000”, price: 1731.46},
           {id: 110427, date: “Jan 2001”, price: 500.46},
           {id: 110427, date: “Jan 2002”, price: 932.46}];
```

The array above called data shows what the original array of data looks like.

```
var dataNest = d3.nest().key(function(d){
    return d.id;
)}
.entries(data);
```

Above we can see how the nesting is executed. The data is in this case grouped by their Id. The array called dataNest shows how the structure of the array has changed after the nesting process.

[9] http://www.d3noob.org/2014/07/d3js-multi-line-graph-with-automatic.html Posted: (08.07.2014), Author: D3noob

```
dataNest = [{key: 110413, values: [{id: 110413, date: “Jan 2000”, price: 1394.46},
                                   {id: 110413, date: “Jan 2001”, price: 1323.46},
                                   {id: 110413, date: “Jan 2002”, price: 1310.46}]},
            {key: 110420, values: [{id: 110420, date: “Jan 2000”, price: 1100.46},
                                   {id: 110420, date: “Jan 2001”, price: 630.46},
                                   {id: 110420, date: “Jan 2002”, price: 880.46}]},
            {key: 110427, values: [{id: 110427, date: “Jan 2000”, price: 1731.46},
                                   {id: 110427, date: “Jan 2001”, price: 500.46},
                                   {id: 110427, date: “Jan 2002”, price: 932.46}]}];
```

By grouping the data together in this manner we can easily draw paths for the elements in the array by iterating through each nest. In this case we would be able to create three different paths. Structuring the data this way is also crucial for setting the “id” attribute of elements we create. By adding the key value of the data to the “id” field of each element created, a lot of the groundwork is being done. Selecting elements by their id and then color and place them, or remove them will be an easy task if the groundwork is done properly. *Figure 17* displays our chart when nesting data with three different Id values.

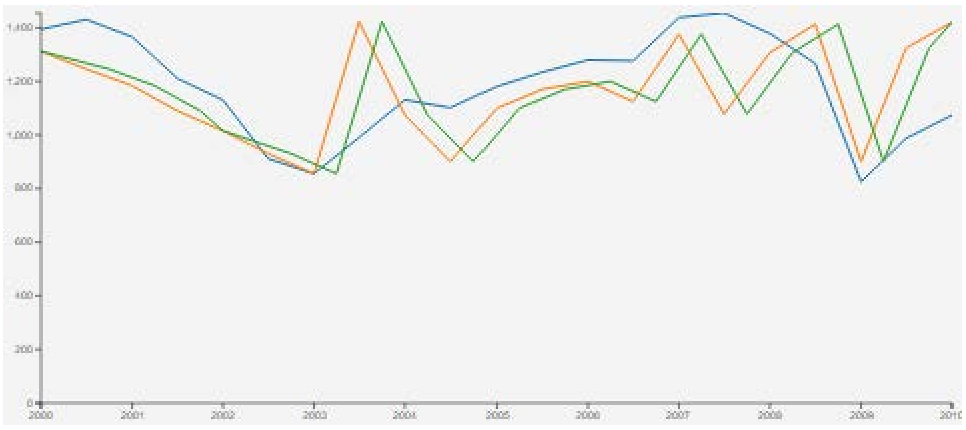


Figure 17 Multi Line Chart

Coloring each line in a unique color based on the path id, is an attempt at creating more structure in the component. For the actual coloring we are using a D3 color scale. d3.scale.category10() constructs a new ordinal scale with a range of ten categorical colors. The d3.scale.category() constructors comes in different forms like 10, 20, 20b and more. It is also possible to make your own color scale but for our purposes this was more than sufficient.

The way the data has been grouped together by the nest function also helps make the component more versatile since most of the operations will be done for each element in the data. Adding more points of data with a new Id will simply create another path and all other elements that come with the path, such as legends or circles for a tooltip function.

4.1.3 Grouping Elements

The SVG group element is used to group elements together and any transformation applied to the group element is applied to all of the child elements contained inside. This lets us handle multiple elements as a single big one. Instead of doing transformations or scaling on every single element, we can group them together and treat the whole thing as a picture that can be scaled, moved or rotated.

Instead of making multiple SVG elements for our line chart, timeline with brush and our control part of the component (covered later in the text), we have divided our SVG element into groups. The reason we have chosen to divide the elements into groups within a single SVG element is because we felt that this was a tidy way of doing it. If we by a later occasion we would like to run multiple different scripts on the same web page, shorting down the number of elements in need of placement would be preferable.

4.1.4 Implementing the Brush Behavior

If data is being displayed for a long period of time it would make it hard to spot small or rapid changes in the numeric values. After looking at an example provided by Mike Bostock[10], we decided on making a brush function for our component. By making a duplicate of our already existing paths in another group below the main chart, we found that we could make a small timeline that displays the entirety of the data. We created another group element with a new x and y domain. The new group element is shorter in height and displays the same paths as in the original line chart, only in a more compressed manner and without the y-axis as a guideline. *Figure 18* shows the brush being used.

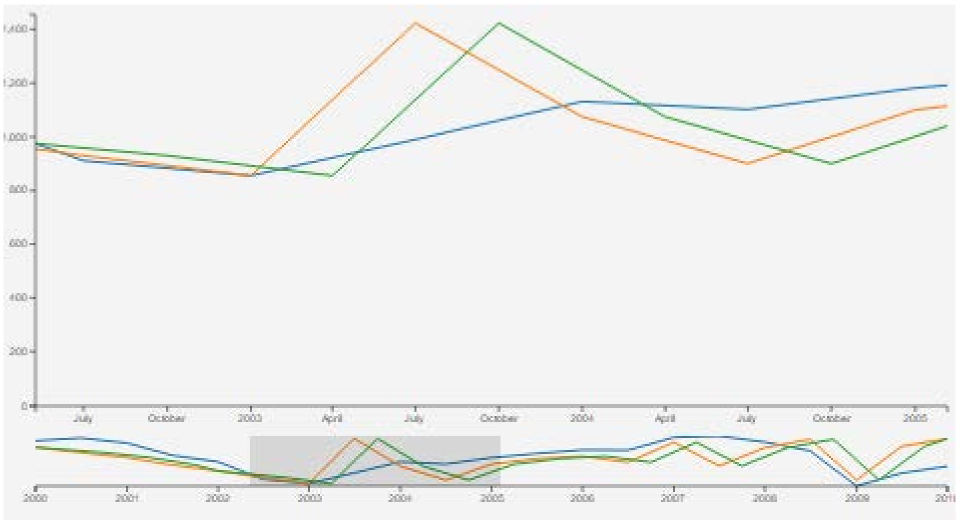


Figure 18 Multi Line Chart with Brush

The reason we picked this way of zooming was because it gives the user an overview of the data even when the user is zooming. It is easy to get a little disoriented while

[10] http://bl.ocks.org/mbostock/1667367 Posted: (03.01.2016), Author: Mike Bostock

zooming in the traditional way, but with the brush function and the corresponding timeline it should make it easier for the user to pick the right scope straight away.

Both the top chart and the small timeline where our brush is fitted have its own x domain. We named the scale of the top chart x and the scale of the small timeline x2. The x2 domain always stays the same, this way the small timeline will always show the entire dataset. The x domain on the other hand is being scaled to fit the brush extent. If the brush is empty (not extended) both the x domain and x2 domain will be the same.

One downside of using the brush function is that the amount of zoom is somewhat limited. Because the brush extends over an area of our SVG element it is limited by the pixel width of the element. This means that it is not possible to extent the brush over a smaller area than 1 pixel. In case of big datasets where 1 pixel might represent a full day of data sampled every hour, a brush function would simply not be the best fit. To solve this problem and trying to get the best from two different behaviors, we have also implemented a zoom function that will be mentioned later.

When using the brush, we access information about the brush through the extent() method of the brush object. The information returned by the extent() method depends on the scales of the brush. In our case we only brush in one dimension, along the x-axis. When the brush only scales in one dimension, the brush extent() method will return a two element array of the form[min, max].

As long as the scale has a valid invert() method the min and max values in the array will be converted into our data domain values. If we were using ordinal, threshold or other scales that does not have an invert() method, the return value of the extent() method would return the values in the coordinate system that is in effect for the brush element.

4.1.5 The Tooltip

Implementing a tooltip function was the next logical step in improving the functionality of the component. The idea was to make it possible to position the mouse pointer over the chart to display the values for that point. After doing some research and looking at an example provided by Mike Bostock[11], we decided on making the tooltip lock on to the closest points of data. By finding the closest point of data in the x-axis from where the mouse pointer is located, we eliminate the problem with having to hit the exact data point or rather a circle representing the point, in order to display the numeric values. With 2000 samples on the chart, hitting the right one might prove difficult.

The first step to making a tooltip was to pinpoint the location of the mouse pointer. Creating and translating a transparent rectangle to fit on top of the SVG element allowed us to make event handlers for when the mouse was positioned within the

[11] https://bl.ocks.org/mbostock/3902569 Updated: (08.02.2016), Author: Mike Bostock

rectangle. The three event handlers created was mouse over, mouse out and mouse move. The mouse over and mouse out event handlers only sets the visibility of the tooltip, this way the tooltip is only displayed as long as the mouse is on top of the element. The mouse move event handler is really the one doing all the work, and it starts by getting the position of the mouse pointer.

The D3 method d3.mouse(this)[0] returns the x coordinate in the rectangle for the mouse pointer. Since the rectangle is scaled to fit the SVG element, these values are identical for both the rectangle and SVG element. Using the x.inverse() method (x being our scale) these values are returned as date objects instead of coordinates. At this point we have the date location of the mouse pointer, but we still need to find the closest data segments. To find the closest segments we took advantage of the d3.bisector() method. The underlying idea behind bisect is that you want to insert a new value into an array and want to know how the array would partition it. Let us say we have an array of [4,5,9,10,12] and want to know where 7 would fit in this array. In other words, we want to know what the index of the number 7 would be if it is inserted into this sorted array. This is exactly what we are doing with the date object we got from the mouse position. We then proceed comparing the date we got from the mouse position with the date of the data placed before and after in the array to see which one is closest. This operation is done for every element in each nest (each path) of the array, something that makes it possible to make display a tooltip for every path we have created at the same time.

Each tooltip consists of three circles, two striped lines, date and a numeric value. As the closest data segment to the mouse pointer position is found, position of the circles is set to the relative position of the data in our SVG element. The striped lines are being drawn in a 90-degree angle from the x- and y-axis to the relative position of the data. The reason behind the striped lines is to make it easier for the user to read the values from the x- and y-axis. The numeric values are displayed to the right of each circle. Because of the limited space when working with a lot of samples we have also decided to display the numeric values in the control group of the SVG element. The control group will be mentioned later. *Figure 19* displays a picture of the tooltip in the chart.

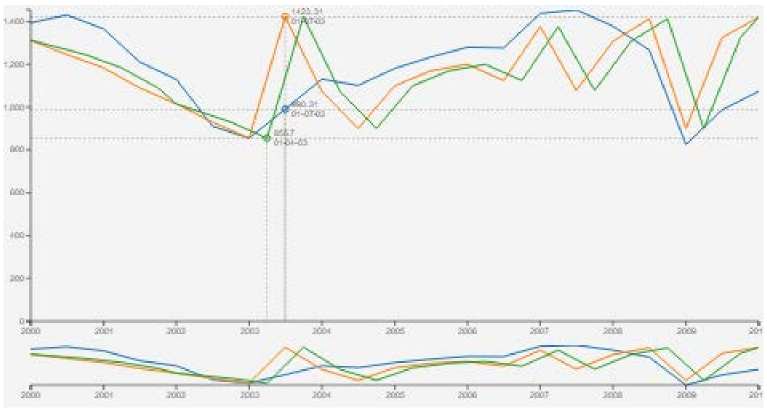


Figure 19 Multi Line Chart with Brush and Tooltip

4.1.6 The Zoom and Pan Behavior

The zoom function made on this chapter is based on the example by Mike Bostock[12] Additional zoom was implemented because of the limitations of the brush functions. This way we could keep the tidiness of the brush but still be able to zoom deep enough into the data. To do this we took advantage of the d3.behavior.zoom() constructor. This constructs an event listener that handles all the zoom gestures (mouse over and drag) on the element we applied the zoom onto. The d3 zoom behavior actually consists of both zooming and panning behavior. The reason for this is that when zooming in on a specific point, we also need to move the scope so that the point we zoomed in on, is still in the picture. The same applies when zooming out. The d3 zoom behavior provides two different properties within the D3 event object, d3.event.scale and d3.event.translate. The d3.event.scale returns or sets the current zoom scale. If the behavior was defined with a scale extent, this will limit the scale. The d3.event.translate returns or sets the x- and y-coordinate translations. This is contained in a two dimensional array.

The two main ways of resizing within the zoom event listener is either through the transform scale attribute, or by doing the math and using the d3.event.scale to recalculate the size of the objects drawn on the screen. The challenge when implementing the zoom together with the brush was that they had to work together.

Our solution was to make the zoom event handler set the extent of the brush when zooming, and make the brush set the scale of the zoom when brushing. The result of this is that the area extended by the brush will decrease as we zoom in. When the user gets to the point where it is no longer possible to zoom with the brush, the user can simply keep scrolling on the mouse wheel to continue zooming further. The brush does not get any smaller than one pixel wide and will therefore still be displayed on the timeline as a guidance marker. With this solution, the user can brush a desired area for further inspection, and if the user wants to look even deeper, zooming the remaining distance can be done with the mouse scroll. We also changed the event of the double click on the left mouse button. Now it resets the scale and translation of the zoom, returning to the original scale or the area that was brushed.

4.1.7 Additional Visualization Options

Changing the type of chart displayed within the SVG element was not very complicated. The data was there and we only needed to change the structure of the objects being drawn. The process of creating the new bar and circle objects is basically the same as when creating paths. While iterating through the nested array of data, we create each object and give it the same ID as the rest of the nest. The ID is crucial for further

[12] http://bl.ocks.org/mbostock/3892928 Updated:(13.01.2016) Author: Mike Bostock

selections like coloring or removing certain paths. After creating the objects, we place them by their values, with respect to the scales of our SVG element.

By making buttons trigger the events, the user can create or remove the different types of objects. The additional chart types we decided on were bar chart and scatter plot. It would often be preferable to see more than one type of graph at a time. A line chart shows the estimates and big picture, while a scatter plot can show the exact time and value of the data. When the line chart and scatter plot are being shown simultaneously, it gives us both the small and the big picture. In other cases, there might be too much data and the points of the scatter plot will merge together. In that case, we want to be able to remove it.

Even though changing the chart type in itself is not all that complicated, our component was starting to get big at this point. There are a lot of functions where the changes had to be implemented, and event handlers created. The first step was making the creator and remover functions. These functions were set to run on various click events. The zoom and brush functions had to be able to scale the new bar chart and scatter plot in the case that these were selected. The event handlers for the legend clicks, mentioned in the control part, had to be able to handle the new bar and circle objects in the same way they did the old path objects. *Figure 20* shows the multi chart displaying the data as a scatter plot and a line chart, simultaneously.

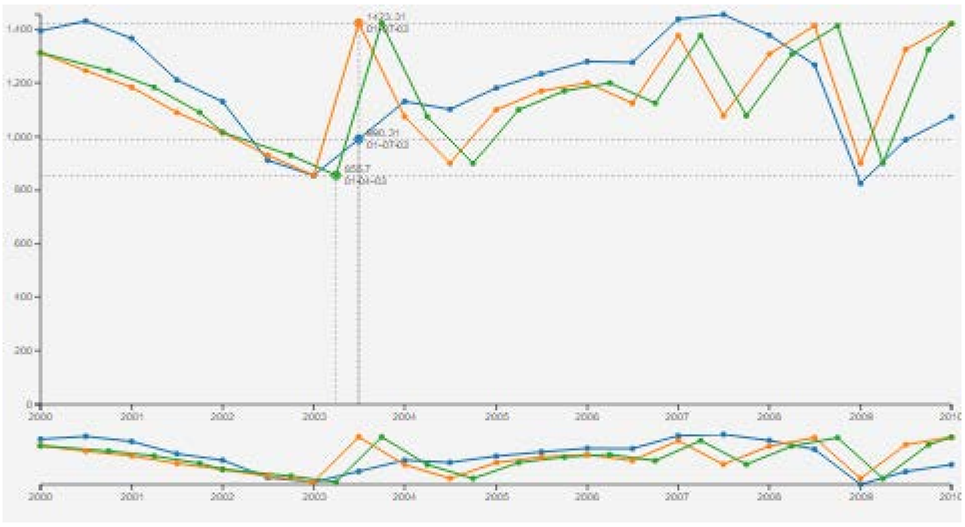


Figure 20 Multi Chart

4.1.8 GUI

To handle user input we made a new group element called controls. Controls contain three checkboxes for the different chart types, the dates and values of the tooltips and the legends made for each nest in the data.

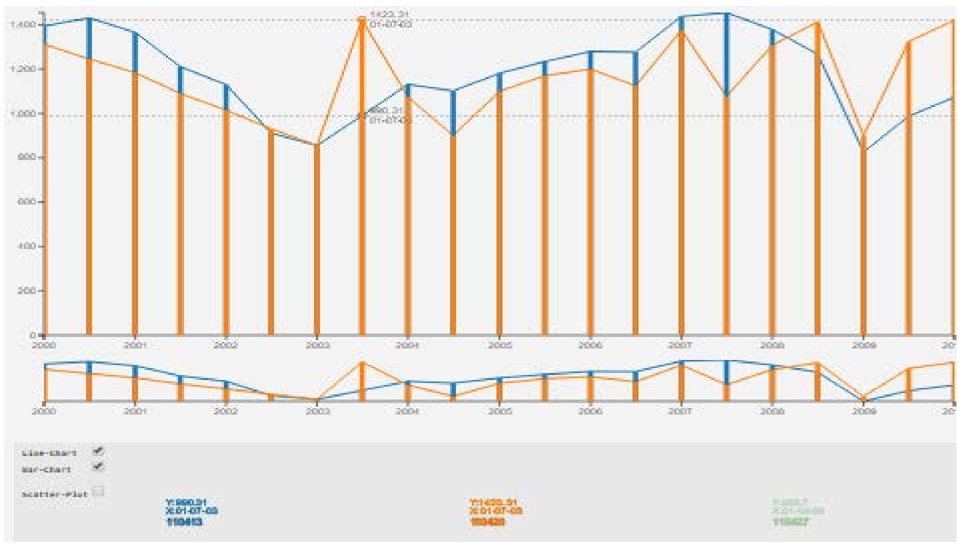


Figure 21 Multi Chart with Controls

In *Figure 21*, the chart is showing the data as both a bar chart and a line chart, for two different nests of data. The colored numbers grouped together are from top to bottom, the value, date and ID of the nest. The colored number on the bottom is what we refer to as the legend, and its number represents the signal's ID. By clicking the legend, the user can hide or show all the objects created from this part of the data. What we actually do is just to change the opacity of all the objects associated with this ID, turning them invisible instead of removing them. One *Figure 21*, one of the legends is almost transparent and is harder to spot than the rest. This is a nest of data that was available, but the user has chosen not to display it by clicking its legend.

The date and value of the tooltips are displayed above the legends as well as in the chart. This is because they might overlap if the data from different nests are close together. We decided on keeping the display in the chart because it works fine when working with a single nest or nests that are spread far apart.

In the top left corner of the controls group, the three checkboxes for the different chart types are located. The checkboxes are actually written in HTML within the script. Checkboxes was not an option in D3 unless we made them ourselves. Because of that we skipped the extra work and used the checkboxes from HTML, which looked pretty good.

Making click events in D3 is really simple. It is possible to create event handlers for just about any object created. To make a click event we just need to select the object and add the .on("click") event handler. The full line would look something like selector.on("click", "event"). The selector represents the object we want to append the event handler to. The "event" is what actually happens when triggered. In most cases "event" would represent a function call.

4.1.9 Containing the Elements

A clip-path was used to remove any shapes that were positioned outside our desired window. Because we implemented behavior for zoom, pan and brush, sometimes our shapes would be displayed outside of the axis labels. To solve this, we made a clip-path covering the area between the inner borders of the axis labels. A clip-path is the path of an SVG shape that can be used in combination with another shape to remove any part of the shape that does not intersect with the clip-path.

Imagine a long straight tunnel with cars driving through it. Now, imagine you are standing outside and looking into the tunnel through a window in the wall. The glass window represents the clip-path and the cars driving through the tunnel would represent our shapes. The cars would only be visible to us when driving past the window. That is basically how the clip-path works.

4.2 The Timeline

4.2.1 Creating the Timeline

The timeline is built in a very similar manner to the multi chart explained in *4.1.1*, and we took advantage of the experience gained from creating the multi chart. We started by creating the SVG element, setting the scales and domains, and appending an axis to the element. A lot of the data displayed on the timeline will be loaded on startup. Data from the static components on the webpage, like sunburst or pitch- and roll-models, will always be represented on the timeline. In addition to these static components, any data loaded into the multi chart is also loaded into the timeline. When collecting data, the height of the bars is set by the total length of the data divided by the height of our SVG element. This way we ensure that the bars created from the data segments will fill the entire height of the timeline. As we loop through the data nests, we place the bars along the x-axis. For each nest, we increase the position along the y-axis with the height of the bars. The width of the bars created from each data segments are set to one pixel. By setting the width this small, the timeline might seem a bit “empty” when dealing with small amounts of data, but when dealing with a lot of data the bars will look like a wide coherent line. To give the bars from each nest and the corresponding legend a unique color, we are using a color scale with 20 different values. The timeline is therefore limited to 20 nests of data (20 different signals).

What makes this timeline an active component is the possibility to click on the timeline and jump to that exact point in time. To achieve this, we make a rectangle that is scaled to fit the SVG element. Just as when making the tooltip for the multi chart, this rectangle is used for getting the position of the mouse and enabling click events. A large red bar extending the entire height of the timeline was created to represent the current location on the timeline. This bar will be referred to as the time bar. In the case of a click event, the time bar is moved to the location clicked. The idea is that the position of the time bar is what will determine which data segments all the other components on the webpage are displaying. This will be explained further in *System Integration*.

The timeline also contains the possibility for zooming. We used the same d3.behavior.zoom() constructor to implement the zoom and pan behavior. By changing the function of the double mouse click, this function now resets the scale to 1 and the translation to [0, 0], thus returning to the original scope.

4.2.2 Playing Through the Data

The timeline play function is where the timeline and the multi chart component diverge. When the play button is pressed the time bar is moved in steps of 0.25px in the positive direction of the timeline. The first problem was making the time bar move at the correct speed instead of looping fast through the motion. To solve this, we used the JavaScript window setTimeout() method. The line of code below is an example on how this method can be used.

```
function play(){ setTimeout(function(){ alert("Hello"); }, 3000); }
```

The setTimeout() method calls a function or evaluates an expression after a specified number of milliseconds. In the line of code above, the alert is being executed after 3000 milliseconds have passed. By calling our play function again from within the setTimeout() method, we can make it loop. If no conditions are set, this loop would go on until the system breaks. In case the stop button is pressed, a Boolean variable called playing is set to false. If the play button is pressed, the variable playing is set to true. By using the playing variable as the loop condition, the user can stop the play function at any time by pressing the stop button. Below is what our code looked like at this point.

```
function play(){
    setTimeout(function(){
        if(playing == true){
            play();
        }
        else{
            stop();
        }
    }, 3000);
}
```

The next problem to solve was how to make the play function play the correct length of the timeline. One alternative was to keep checking the data as it goes through the timeline. Then it would stop as soon as the last data point ended. Since we have implemented zoom and pan behavior to our timeline, some of the data segments may appear outside the scope of the timeline. In that case, we would get some strange values for our time bar position and it would not work properly. This could probably be solved with a simple if statement. Another problem with using the data to guide the play function is that it is time sensitive. In the case that the timeline is handling a lot of data, this could prove a problem. It would limit the potential speed of the execution if a lot of operations were executed within the function. Instead, we based the play function on the width of the timeline. One variable was created to hold the number of times the play function had executed. Another variable was created to hold the starting position of the time bar. By comparing these two variables to the step-length of the time bar and the width of the timeline, it is possible to determine when the time bar reaches the end of the timeline. The play function code is displayed below.

```
function play(){
    counter++;
    setTimeout(function(){
        if(counter < (width -startingPosition)/stepLength && playing == true){
            play();
        }
        else{
            stop();
        }
    }, delayTime);
}
```

Two speed buttons were created and works by changing the delay time between executions within a certain range. The piece of code above is a simplification of the original code and only contains the basic operation. Since we wanted the user to be able to pan, zoom and click while the play function is executing, the starting position is changed and the counter is reset within each of these functions. Since our play function is based on the width of the timeline, there are actually two different ways of setting the play speed on the timeline. The first way of doing this would be to press the speed buttons. That would change the time it takes for the time bar to move the distance of the timeline. The second way would be to zoom in or out. When the user zooms, the domain of the timeline changes. If the user zooms inwards, the time domain would be smaller than it was, but the time bar would still use the exact same time from start to finish.

4.3 Review

The finished multi chart works well and can handle most numeric data. The elements in the multi chart are created by nesting the data. Changing between the different chart types works well and is easy to understand. Not every chart type is beneficial to view together, but using combinations like scatter plot and line chart can really improve the viewing experience. The brush function with the corresponding timeline works well to keep an overview of the data, even while zooming. The possibilities of selecting which data to view by clicking the legends work well to give the user additional viewing options.

It would be useful to be able to select which data should be displayed as a bar chart and which should be displayed as a line chart. This would have taken some time to implement and we decided our time was better spent elsewhere.

Zoom and pan behavior in both x and y-axis was also considered. This was not implemented because it made it a lot harder for the user to control the zoom and pan behavior. By adding the option to select whether the x-, y- or both -axis should zoom this problem could have been solved.

Sizes of the different elements could be set more dynamically. For example, now the size of each point in the scatter plot is set to a fixed size. If there are too many points they will overlap and form a thick line. If the number of data segments in each dataset sets the size of the points, it would prevent elements from overlapping.

The timeline works well as an overview of all the data available, and nesting the data creates all elements. This allows the user to see which data is available and loaded for the period displayed in the timeline.

The timeline play function could have been made differently by using an interval method instead of a timeout. This would have improved the accuracy of the time between each function call. This low accuracy may be a source of problems when synchronizing against other components.

To improve the response of the zoom and pan behavior, the number of elements removed and redrawn needs to be decreased. When handling large amounts of data, the zoom and pan behavior is too slow. It might be possible to reduce the number of elements being redrawn by translating and scaling instead.

The idea of basing the time bar movement on the width of the timeline is probably not the optimal solution. One of the reasons this was implemented was because of the low sampling rates of the early data.

An additional feature could be to let the timeline adjust the domain of the timeline when it reaches the end of the element, this would let the play function pan the time domain of the timeline as it plays through it.

The data structure used in both the timeline and multi chart could be achieved in other ways than using the d3.nest() method. An example would be using the D3 iteration method array.map() to nest the data. Using the array.map() method would not require the specific Id column in every row of the CSV file, and would give more freedom to structure the data. Another option would be to get the data in the form of an array from the database, already nested.

5 HTML5 VIDEO

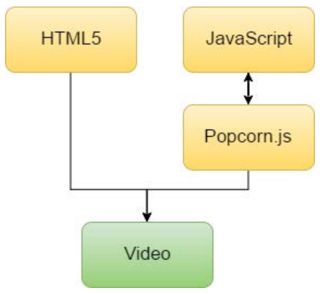


Figure 22 Video Related Technologies

Implementing videos in to the final application opened up many possibilities. It opens up for a different approach to data visualization. Instead of just showing the environmental conditions of the ship as numeric data, letting the user see conditions of the ship, could be interesting.

In relation to HTML5 video, the group researched many different topics.

Synchronizing two videos makes it possible to have a view of the ship from different angles. This lets the user view the interesting aspects of the ship during the traversal at the same time.

Changing the layouts of these videos so that the focus can be shifted from one to the other makes the application more dynamic. The parts that are the most interesting will not be obscured by the size by the other video.

The group also did some research on video metadata and how this could be used in an application built on HTML5. The issue of performance was also investigated, as to may videos running at the same time will slow down the application as a whole.

Overlays on the videos were also looked into. How to do it and, what could be interesting to place over the videos.

5.1 *Video setup*

Videos are implemented in HTML using the Video tag. Within the tag there can be several video attributes. It is recommended to include the height and width attributes for video elements. If they are not set, the browser does not know the size of the video elements and the page will load unpredictably. The video attributes are also modifiable in JavaScript through the use of the document.getElement function. The video source can also be included by the source tag, not only in the “src” attribute of the video tag. The source tag can be put inside the video tag or have the video tag append it as a child in JavaScript. The code snippet below shows how to set and change the source of a video element in JavaScript.

```
// Gets video element by the video tag Id
var video_a = document.getElementById('a'),
video_b= document.getElementById('b');
//Creates a source tag in the html
var source_a = document.createElement('source');
source_a.setAttribute("id","src_a");
var source_b = document.createElement('source');
source_b.id=("src_b");

    // Sets the src(source) attribute of the source tag
    source_a.setAttribute('src',"funny.webm");
    source_b.setAttribute('src','funny.webm');
     // append the source tag to the video tag
    video_a.appendChild(source_a);
    video_b.appendChild(source_b);
```

5.2 *Video Handling*

Due to time constraints, the group was not able to find any well-suited methods for the storing and loading of video files. The group also experienced difficulties with accessing the video files' metadata from JavaScript. A decision was made to simply set the name for each video file to the start date and time of the video. By doing so, we created some reference to the locations of the videos relative to each other. The filename and duration for each video file had to be hardcoded into an array containing all the different video files being used in the application. By converting the filenames back and forth between JavaScript date objects, this solution gave us some room for creativity.

5.2.1 Video Selection

The idea of loading different videos came up as a method needed for future implementation against other components. Since we did not have any sophisticated back-end solution for loading video files, some other measures had to be taken.

The process of locating and loading different videos was done by sorting them in the array containing the video filenames. Before the filenames of the videos could be converted into a date object, some string operations had to be made. By cutting away most of the source address, we are left with only the filenames of the videos.

When the string operations are done, each of the filenames is being converted to a JavaScript date object and pushed into the array together with its corresponding duration. The array is then sorted into an ascending order. The ascending order is crucial for the bisect function to do its work described in detail in *4.1.5*. The function tasked with finding the closest videos then bisects the ascending array based on the input timestamp of its function call. When the bisect value is returned, the function compares the video dates closest to it by both start date (filename) and duration of the videos. When the two closest videos are found, it will call a function that is tasked with loading the videos. The input parameter of the function tasked with loading the videos is an array consisting of two string values. The sting value consists of the file name from the two closest videos.

These values are converted back into the full sting containing source location, before the function checks if the conditions for loading any new videos are met.

The function tasked with loading the videos will first check if each of the video elements already contains an assigned source. If it does not, the function will load the videos. If the video elements already contain an assigned source, the function will check if the sources requested are different from the already assigned sources. If there is no difference between the assigned sources and the requested ones, no action will be taken. If there is a difference between the sources requested and the sources assigned, the corresponding element will assign the new source location and load the new video file. By checking the current sources assigned to the video element against the new sources requested, there will not be any unnecessary new loading of videos already contained within the element.

5.3 Video Styling

The styling of videos in this application has consisted of exploring the possibility of having useful elements as overlay on videos. Also, making the application more dynamic by letting the user change the layout of videos within in frame.

5.3.1 Overlay

Overlays on the video can convey information in an easily digestible way. Having a relevant element overlaying a corner of the video could add valuable information. Overlaying something like the output of the generators would probably not be very useful. The vessel’s speed and heading along with water depth and other information related to navigation would probably be of value.

The process of adding an overlay to the video is quite simple. It is all controlled by CSS by positioning it over the video and making sure it has a z-index value greater than the one for the video. It is important to ensure that the overlay always stays on top off the video element, even if changes are made to the position of the video element.

There were plans to make custom D3 modules to overlay on the video. Elements such as heading and speed could both have been useful options. Due to time constrains these components were not prioritized.

5.3.2 Video Setups in Frame

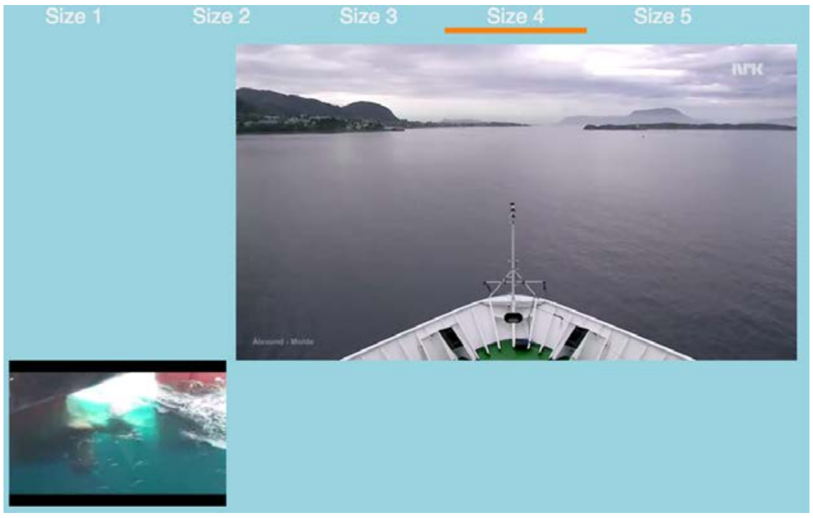


Figure 23 Video Setup in Frame

Having a video frame, in which it is possible to change the layout of the videos, allows the user to find a better viewing experience that suits the situation.

Since the application currently only have two embedded video elements, the amount of different setups the frame needs, is limited. There are now five different styles for the video frame. One where the videos are equally large, one for each of the videos to fill out the whole frame and two where one of the video elements is large and one is small. These styles can be switched between using a navigational bar at the top of the video frame with one button for each setup.

The different styles of the videos are defined in CSS. Each different style has its own CSS class. When the application is first loaded the videos are set to the default style, the one where both videos are the same size. Each button has a function that changes the video tags class name, and the videos' style then changes to that of its new class.

Figure 24 illustrates what happens when a video setup is selected from the navBar.

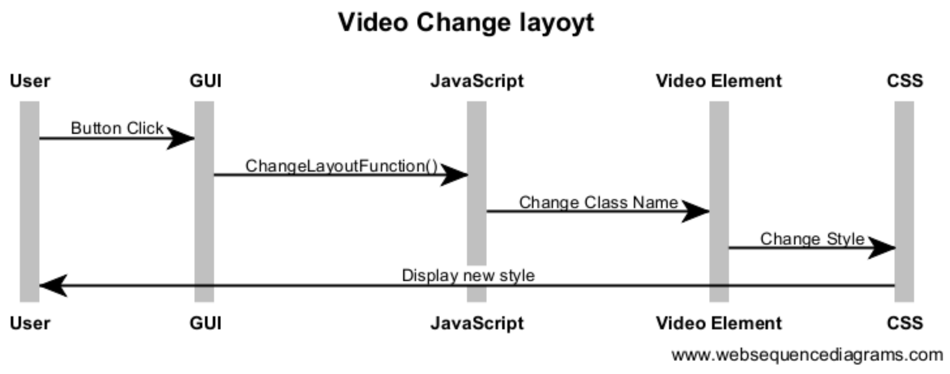


Figure 24 Video Change Layout Sequence Chart

5.4 *Synchronization*

5.4.1 The Issue of Video Desynchronization

Having videos just play at the same time, and having each operation done on one of the videos done to all, is not an ideal solution. The videos will start to desynchronize due to small differences in how the players are being treated by the hardware. The rate of which the videos will desynchronize depends upon a number of factors. Some tests were locally preformed, and the result was varied. In some if the tests, the desynchronization just amounted to a few frames. In other tests, one of the videos was lagging several seconds behind. The desynchronization is a direct effect of performance issues, and the desynchronization rate will increase as the stress on the hardware increases.

5.4.2 Video Libraries

The Popcorn library for JavaScript was used to synchronize the videos against each other. The Popcorn library extends the control over the videos, and makes creating media centered applications easier.

The Synchronize JavaScript library was also considered during this project. This was the easiest way to synchronize videos, but had several drawbacks that made it unfit for the application. One of the things that made the library so advantageous was that it required less coding. Just by loading the videos into the same media class, they would be synchronized. The drawbacks were that if the videos started to go out of sync, the response of the script was to increase the speed of the videos that were lagging behind. The speed increase was quite noticeable and the videos fell out of sync quite often. The major drawback was that if the master videos time was manually set to something less than the other videos, the other videos would just play normally and the master video would be lagging behind them. There might have been a solution to this problem, but to embark on such an effort would have cost much time that was considered better spent elsewhere.

5.4.3 The Solution

The group's solution was the derivative of the script made open by Rick Waldron[13]. In this solution, the two video elements are synchronized by following a master-slave principle. The different video elements will be referred to as element A and B. The master video is the video loaded in element A. The video loaded in element B is set as the slave. Event handlers are created in such a way that any event triggered in element A also will be triggered in element B. By pressing play on element A, both elements are set to play

[13] https://bocoup.com/weblog/html5-video-synchronizing-playback-of-two-videos

and an interval is created. The JavaScript setInterval() method either calls a function or evaluates an expression at a certain interval set in milliseconds. The line of code below shows how this method can be used.

```javascript
var intervalA = setInterval(function(){ syncA(); }, 100);
```

This line of code creates a new interval that will call a function called syncA() every 100 milliseconds until the interval is cleared. By creating the interval when the play button of element A is clicked, the syncA() function will be called continuously ten times every second. The purpose of the syncA() function is to check if the videos are still synchronized. The currentTime property of the HTML5 video either sets or returns the current position (in seconds) of the video playback. By comparing the current time of element A to the current time of element B, we can find the exact offset between the two videos. If the absolute value of the difference between the two current times is larger than 0.1 seconds, the syncA function will set the currentTime property of video B equal to the current time of video A. If the difference in current time between the videos is less than 0.1 seconds, no actions need to be made. We found that the operation of setting the current time attribute of the video element required some time to process. Setting the sync to be any more precise than 0.1 seconds would require calculating the time it takes to do the operation and add that time to the new current time of element B. If no such measures are made, setting the offset to be less than 0.1 seconds would only result in creating an endless loops of synchronizing that would never finish in time. The interval that calls the syncA() function ensures that the videos are continuously being checked and corrected if they are out of sync. The interval that calls the syncA() function is cleared when the pause button of video element A is pressed, or when the video finishes playing.

Figure 25 displays a picture of the simplified synchronizing process.

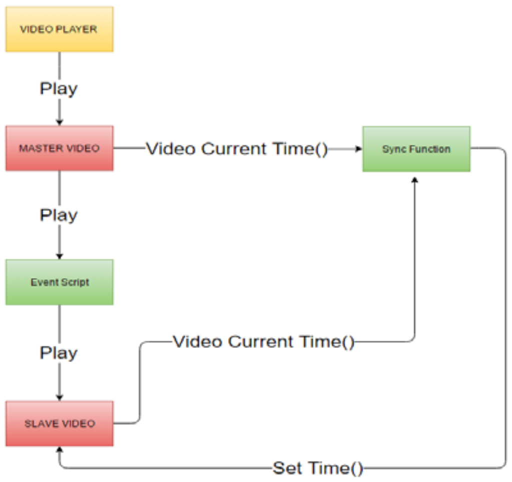


Figure 25 Video Synchronization Flowchart

5.5 Metadata

When relating to videos, there are two kinds of metadata:

- Type 1 that is added when the video is created
- Type 2 that is added manually

The one that would be most interesting is type 1. Metadata type 1 often contains information relating to the date of creation, location, author and camera description. This could be used to place the video on both the timeline and map, as well as give the video an ID of which camera it was filmed on. The HTML5 video element only gets metadata directly relating to the playing of the video such as dimension, duration and text tracks. Using metadata type 1 for sorting videos does not seem to be possible using only JavaScript and HTML5.

Metadata type 2 is data that is added later. This can be incorporated into HTML5 and JavaScript in the use of the HTML5 <Track> element. Tacks can be used in several ways to enrich videos. There are possibilities to add subtitles, chapters, captions as well as metadata type 2. Giving the tracks the information that would be beneficial to the application lets us access it through HTML and JavaScript. The problem here, as well as with video, is that the video and tracks are not visible to the application before they are loaded into the element. So to assign the videos to a specific place on the timeline or map would still prove to be challenging.

5.6 Performance

Video performance is dependent upon the videos' resolution and refresh rate, and also the computer the videos are being played on. High resolution and refresh rates put a bigger load on the hardware. Demanding video operations would require more than average consumer-end hardware.

Doing operations, such as changing the size of video elements during playback, may also impact the performance of the video and create a performance bottleneck. If there are problems with the performance of a video, the video will start to stutter. If the video stutters too much, it will become unwatchable.

Since it is rarely useful to run more than one video at full resolution, an option would be to have the application transcode the video to a lower resolution. The amount of videos on screen could dictate the resolution of the videos. If there is one video on screen, the video will run at full resolution, two would run at half of the resolution and so on. If there is a need to have one of the videos in full resolution, the other videos could be removed. This would ensure that performance would remain stable, and since the videos would take less space on the screen, the lower resolution would not impact the viewing experience.

5.7 Review

The finished video application synchronizes two videos. Adding a third video is no problem when following the setup provided in this solution. It has a custom video frame to allow for the selection of different viewing experience.

Since only two videos are currently being run by the application, the frame works well enough for the time being. The number of videos currently being shown in the frame made it easy to create pre-defined layouts for viewing the videos. The video frame has the same aspect ratio as the videos we wish to display. The frame makes it hard to utilize the space when we try to display only two videos in the same frame. When the two videos are set to be of equal size, there is a lot of empty space. The only way to make us of the space would be to either have four, or way more, videos within the frame or by resizing the frame itself.

Video handling, while not optimal, works for this application. Its current nature is the byproduct of this application being built as a proof of concept. It is quite rigged and making changes to which videos are being used in the application requires hardcoding the names, in the form of timestamps, and the durations of the videos. This solution also makes it impossible for two videos to start at the exact same time because it would cause a naming problem.

Video performance is currently not a problem for the application, but it might present a challenge if the system is expanded. The tests that were performed showed us that it is possible to have up to seven videos playing at full resolution before performance issues begins to occur. It seems that transcoding the videos, while giving benefits to the application as a whole, is not of dire importance at the moment.

6 MAPS

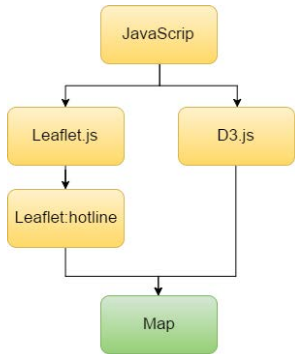


Figure 26 Map Related Technologies

Exploring different map APIs have proved that maps can represent a useful tool for visualizing location data with different approaches. Vessels' location and path can be displayed. Markers and information popups can be implemented to give more information about the current location of the vessels. We have explored how location data can be combined with data for power consumption. We have also looked at the use of visualizing weather data in combination with maps.

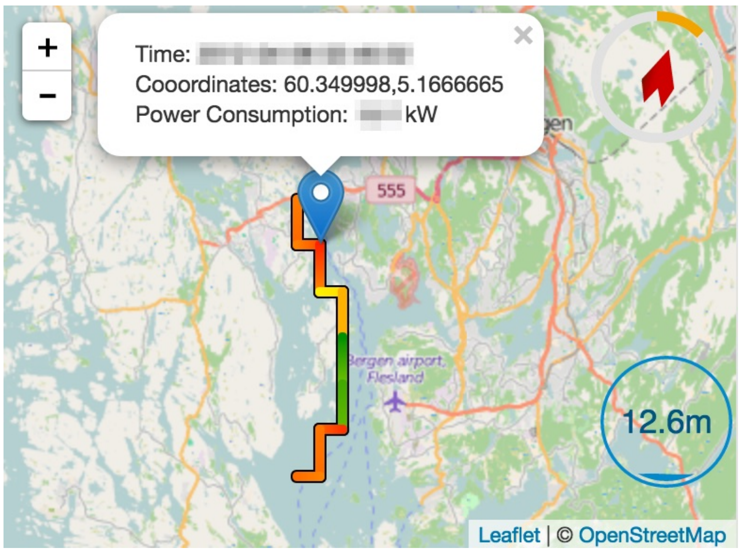

Time: [redacted]
Cooordinates: 60.349998,5.1666665
Power Consumption: [redacted] kW

Figure 27 Map

On the *Figure 27*, the marker shows the vessel's current location. Clicking on the marker will reveal a popup that contains relevant information on the vessel's current state. The gradating line shows the ships path, where it has been a where it is going. The gradating color illustrates the power consumption outputted by the generators. Some elements representing weather conditions are added as overlays on the map.

6.1 *Map APIs*

6.1.1 Google Maps

An API that was explored and considered for use in this application is Google Maps[14]. One of the advantages of this API is that it provides access to Google's own satellite photos, road maps and hybrid maps. Google Maps is free until you exceed 25,000 map loads per day for 90 consecutive days. The reason why Google Maps was not implemented in this project was because the gradient coloring of the path was difficult.

6.1.2 Leaflet

Leaflet.js was chosen as the map API in this visualization because of its ease of use and good online documentation.

The Leaflet API has an abundance of plug-ins and extensions that proved to be useful in combination with maps. A plug-in called Leaflet.Hotline made the process of gradating the path on the map with the color corresponding to a numeric value quite simple.

Leaflet is built up using layers. Each element that is added to the map is presented as a layer. These layers include a map tile layer, which selects the background map, and any other element, which is later added to the map, gets its own layer. There are many map tiles to choose from and they all give different purpose to the map considering how they look. One can use a road map provided by Open Street Maps or a National Geographic world map. The code below shows how to add a map tile.

```
// Creates a new tile layer, and give it a link to a map layer
    var layer = L.tileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    maxZoom: 19, // sets the maz zoom level
    attribution: '© <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'
});

    var map = L.map('map').addLayer(layer); // Creats the map and and adds the tile layer to it.
```

6.2 *GPS data*

The GPS data that is being used in this component can be handled in a number of different ways.

The most common way to handle GPS data for web-based map APIs is by using the GeoJSON file format. The GeoJSON file format is an offshoot from the file format JSON. It contains objects with latitude and longitude values, along with geometry type and other relevant information. The geometry types, such as line, polygon and point, decide what will be displayed on the map. The GeoJSON file format is very useful for one-time visualization on a website, but proves challenging when used in a dynamic application. It

[14] https://developers.google.com/maps/documentation/javascript/

is therefore necessary to change the GeoJSON file for different visualizations one would like to display on the map.

```
{
"type":
        "Feature", "geometry": {
                "type": "Point",
                "coordinates": [125.6, 10.1]
        },
        "properties": {
                "name": "Dinagat Islands"
        }
}
```

Another possibility is to use CSV files with a GeoCSV-plugin for Leaflet. The plugin loads a CSV file to the GeoJSON layer in Leaflet. This is a lightweight solution since a GeoJSON file can be as much as four times larger than a CSV file containing the same data. This solution will run into the same problems as the original GeoJSON solution, since there is no way to specify the geometry type using the plug-in alone.

The approach used in this visualization was to output the retrieved data from the database as a multi-dimensional array, as described in *3.4.4*. The array contains values for latitude, longitude along with the total power consumption of generators.

6.3 Drawing Paths using D3

It is possible to draw a path on the map using the geometry type, points, combined with the D3 library. D3 is easy to combine with Leaflet, and the possibilities for creating interesting visualizations and animations on maps are many. By using the file formats GeoJSON, TopoJSON, JSON or CSV, this is a good and easy solution. A prototype was made using this method with a GeoJSON file containing the geometry type, points. The prototype was successful at drawing a path through the points. Both the shape and size was retained even as the map was zoomed in and out. This solution proved to be hard to use when it came to color gradation. The conventional way of gradating a line in D3 was very convoluted and hard to understand for novice developers[15]. The modifications made to it to make it gradate from a value, was unsuccessful. There was also an attempt to gradate a line chart of the value that would dictate the gradation and use the offset of this to dictate the color gradation in CSS[16]. This led to an unstable line that would randomly change its colors. In the end, the effort to find the source of these problems was deemed to great and the focus of gradating the line was moved over to the leaflet.hotline plugin. The prototype can still be useful in other circumstances where gradation is not of importance.

[15] https://bl.ocks.org/mbostock/4163057
[16] http://bl.ocks.org/d3noob/3e72cafd95e1834f599b

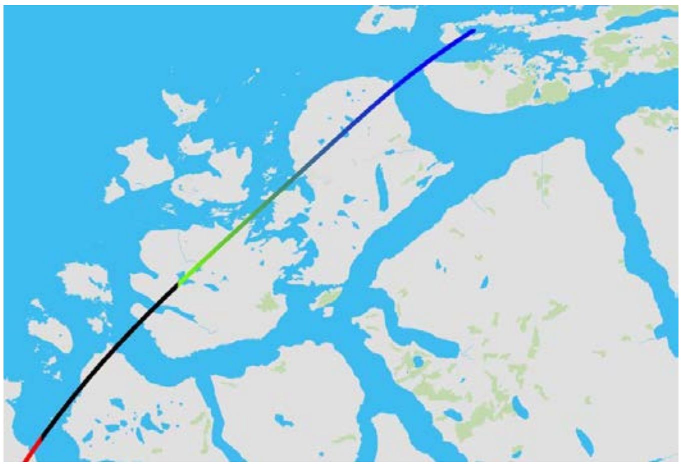


Figure 28 Unfinished D3 Map Prototype

6.4 Tracing and Color Gradient

To achieve color gradation determined by a third value along the ship's path, the plug-in Leaflet.Hotline was used. This plugin extends the polyline function in Leaflet. Three colors were selected for symbolizing a low, medium and a high value. For this application, the color combination of green, yellow and red was used to visualize low, medium and high respectively. The value for each color is being mapped between 0.0 and 1.0. Where the color values are put can change the way information is interpreted from the application. In this visualization, the green value is set to 0.0, yellow to 0.5 and red to 1.0.

In addition to mapping the colors in a logical color scale, we discovered that setting the minimum and maximum values for the color gradient can be done in several ways. Every value below the minimum value will be assigned the lowest mapped color, in this case green. Values over the maximum value will be assigned to the highest mapped color, in this case red. Where to set the values, will also affect the outcome. Some of the ways these values can be set are listed below.

- Static, theoretical values can be set in script
- The user set the desired values as inputs
- The user can select type of ship/vessel as input, which corresponds to vales for different ships that is set static in the script
- The highest and lowest values from the data can be set as max and min values

By setting the values to the theoretical correct max and min values, a theoretical view of the data will be presented. By following this approach, there may be room for misinterpretations of the information since different vessel have different theoretical values. A small vessel would not have the same values for power consumption as a large vessel. This solution will be best if the application is suited for a certain type of vessel.

If the maximum and minimum values are set to the highest and lowest value from the dataset, a view of the data in a specific situation will be presented with the colors mapped between that specific dataset. This solution may raise inconsistencies in the sense that it is hard to get a grasp on what the values actually mean. An operation might be demanding, but if the values have little variation during that time, the colors will be mapped between those values and indicate that the operation was not demanding. This would again distort the other color values to reflect this, and values that are considered high will have a lower color value that it should have.

By letting the user input and define the maximum and minimum values that suits a specific observation, will prevent misguided information from the visualization. If the user does not have the needed information for defining these values, an alternative way could be to select vessel type, which corresponds to some values that are reasonable for the different vessels.

6.4.1 Creating the Color-Gradated Path

In order to create a line that represents the vessels path with color gradation using. A variable must be declared. This variable calls the hotline function with the data as an array as parameter. In this function, the options for setting the maximum and minimum values for the gradated is made along with the color mapping. Finally, these options are being applied to the data and drawn on the map as a color-gradated path.

```
// Creats a new layer with the hotline function, which extends the polyline function

    var hotlineLayer = L.hotline(latlng, {
     min: x, // Minimum value for the color gradation
     max: z, // Maximum value for the color gradation
     palette:{ // The color ponts for the mapping
                0.0: '#008800', //green at 0%
                0.5: '#ffff00', //yellow at 50%
                1.0: '#ff0000' //red at 100%
            },
    weight: 5, // Thickness of the line
    outlineColor: '#000000',
    outlineWidth: 1,
    smoothFactor: 0 // How much to simplify the
// polyline on each zoom level. More means better performance and smoother look, and
 less means more
//accurate representation.

    }).addTo(map); // adds the layer to the map

    var bounds = hotlineLayer.getBounds(); // Retruns the Lat Lng bounds of the path

    map.fitBounds(bounds); // Sets a map view that mostly contains the whole world w
ith the maximum zoom level possible.
```

Other options in the hotline function's setup are available. It contains the ability to select the weight of the line, the outline weight and the outline color. These options can be used along with a smooth factor, which will remove hard edges on the line. Using the smooth factor in this case will cause the line to not reach all the coordinates and will give a false sense of the data. Lastly, the bound of the map will be set to the bounds of the line. This ensures that each time the application loads the map, the right dimensions for displaying the path are set.

6.5 Marker

A marker can be added to the map as a layer. The marker receives the current coordinates of the map, and is then added to the map at that position.
Adding a popup to the marker is done with the bindPopup function. This function accepts HTML elements and strings.

```
var marker= L.marker(latlng[4]).addTo(map).bindPopup('<p>'+S+'<br />'+A+'<br />'+'Co
ordinates: '+C).openPopup();
```

The marker can be customized with the icon function. The icon function can change the size of the marker, and use other graphical elements to display the location.

```
var myIcon = L.icon({
    iconUrl: 'ship-icon.png',

    iconSize: [20,40]

});
```

6.6 Weather Conditions on Map

Incorporating weather data to the map, might increase the application’s area of use. Weather data for wind, wave and currents would be useful to see in combination with location data. It can provide more detailed information about the actual conditions during operations. The weather data may also reveal causes for anomalies found in the numeric data. This can give a better description of how weather conditions affect the different processes of the vessel.

A good way to display this would be with a heat map layer on the map. There are several examples of how to achieve this with D3, but the biggest obstacle here is the data. Open weather data is available, and the most common file formats for these data are GRIB, netCDF and HDF. These file formats are complex, but can be read by appropriate software without the user knowing the files’ structural details. Valuable data for visualizing weather conditions is often restricted to the GRIB (Gridded Information in Binary) file format. GRIB is the format used by metrological institutes (Norwegian Meteorological Institute, Météo et climat, etc.) around the world to transport and manipulate weather data. GRIB files contain one or more data records, arranged as sequential bit streams. Each record has a header, followed by binary data in packs. It contains information about:

- The qualitative nature of the data (field, level, date, etc.)
- Meta information of the header
- The methods needed to decode the data packs
- Layout and geographical data of the grid the data is to be plotted on.

The time and effort for implementing this in our web-based data visualization seemed extensive, and was not prioritized.[17]

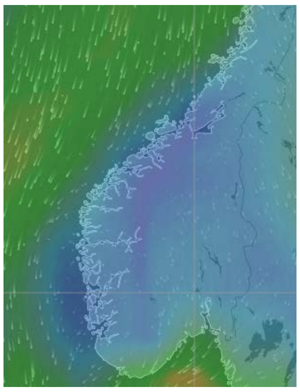


Figure 29 Windity

An example of weather conditions visualized with D3 is a wind map of Tokyo[18].

Other, more advanced applications on the web such as Windity[19] that display these conditions in an interesting and detailed way and served as inspiration for what the group wanted to do. Similar approach was made to the very similar "earth" application[20].

Given a lot more time, the group would have liked to delve deeper in to this issue and try to find a solution suitable for the use in this application. As a substitute, D3 elements relating to wave high and wind speed was laid over the map. Read more about them in the next chapter.

6.7 Review

In its finished state, the map component adds good value to the application. The color-gradated line illustrates both the vessel's path as well as power consumption from the generators. The marker indicates the ship's current location. The marker's popup gives the user an easy way to access the exact information and values. These elements combined provide useful information in this visualization. They make it easier for the user to grasp the vessels operation in a bigger scale. It is possible to interpret from the map where and when operations were demanding and when they were less so.

Implementing a directional arrow as marker on the map would make it easier for the user to see the vessel's heading. In the application's current state, it is hard to understand which direction of the path the ship is heading. The only way one can figure that out is by playing the timeline or investigating the data in the multi chart. This is considered as an easy implementation that would have been implemented if the group had more time.

[17] https://climatedataguide.ucar.edu/climate-data-tools-and-analysis/common-climate-data-formats-overview

[18] http://air.nullschool.net/
[19] https://www.windyty.com
[20] https://github.com/cambecc/earth

Even though D3 proved hard to work with, especially in the context of the gradating line, it should still not be neglected. It has excellent data management capabilities, and D3 offers a lot of possibilities in terms of visualization that leaflet just does not have on its own.

The matter of file formats when working with maps is something that must be considered. Maps can be restrictive in the use of their functions by certain data types. A good solution could be to retrieve and output location data combined with other useful data in the GeoJSON format.

The formats for weather data proved to be complex to work with. They became an obstacle that hindered the group in doing map-related weather visualizations that could be useful. If more time were available, the group would have taken a second approach to try to use these data. This is something that would add more value to the map as an application for data visualization.

7 STATIC COMPONENTS

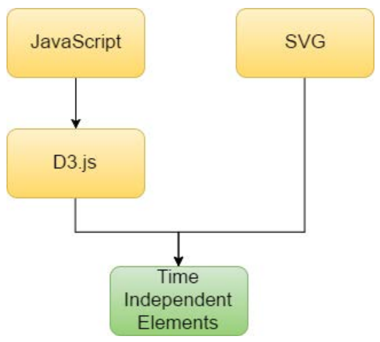


Figure 30 Static Components Related Technologies

Different elements that do not visualize in a historical time frame have been implemented in this visualization. These components are static, and display something at a given time. Some of the elements are found in D3 gallery, and other elements are simply SVG elements imported to the visualization as images. We have explored different examples from the D3 gallery, and found that some of the examples could be useful in our application given the right area of use.

Sunburst is a layout that can be useful for visualizing contributors of a process. We have explored different scenarios and processes that can be valuable to display in a sunburst layout. The sunburst that has been implemented in this visualization is used to describe how much each generator contributes to the power consumption.

Since the idea of using open climate data and creating a heatmap were scraped, we wanted to find alternative ways to visualize some weather conditions.

An example of a radial progress bar was implemented and combined with a directional arrow. The combination of a direction and a value was something we could vision several uses for. We used the combination to visualize direction and speed of the wind.

Another D3 example that was implemented is called liquid fill gauge. This component seemed suitable for visualizing either water depth or wave height.

The group also had a desire of visualizing pitch and roll, and created SVG elements of ships viewed from the front and profile that were rotated.

7.1 Sunburst

Sunburst is a component that can be used for visualizing contributors of a process. It is interactive, interesting to watch, and gives a view of how several elements are connected. The sequence sunburst is developed by Key Rodden using the D3 library.[21] Below, two examples of how the sequence sunburst can be used for visualizing data from maritime operations are shown.

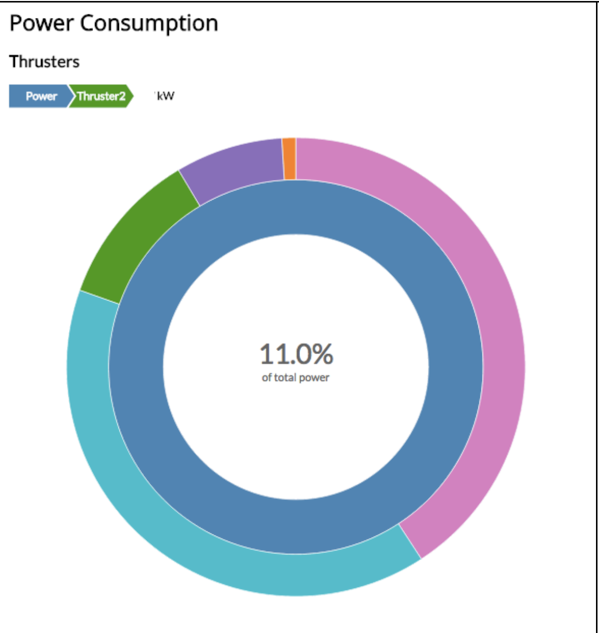


Figure 31 Sunburst Power Consumption

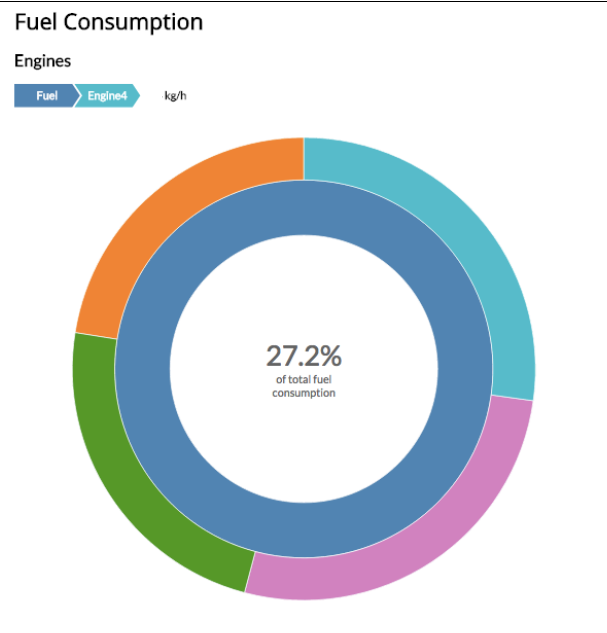


Figure 32 Sunburst Fuel Consumption

These are just some examples of what it can be used for. The component displays information when the mouse pointer enters some of the component's arcs. The step names and values are being displayed in the top left corner when the mouse pointer enters. The percentage that describes how much that step arc contributes to the whole process appears in the middle.

As mentioned in chapter *3.4.4*, the sunburst needs a description for each value in order to build itself. The data outputted from the server-side-script looks like this:

```
Power-Gen1,1
Power-Gen2,2
Power-Gen3,3
Power-Gen4,4
```

The client-side-script for the sunburst loads the CSV data and converts it to a JSON structure as shown below.

```
d3.text("data/sunburstData.php?timestamp="+timestamp, function(text){
  var csv = d3.csv.parseRows(text);
  var json = buildHierarchy(csv);
  createVisualization(json);
});
```

[21] https://bl.ocks.org/kerryrodden/7090426

The parseRows function removes the header file if there is any in the CSV output.

The buildHierarchy function takes the CSV and transforms it into a hierarchical structure suitable for a partition layout. The first column is a sequence of step names, from root to leaf, separated by hyphens. The second column consists of values that are being compared to the other values. This function returns a JSON setup.

The createVisualization function takes the JSON variable, and uses it to draw and set up the visualization. This function also activates the mouse interaction.

7.2 Directional Arrow and Radial Bar

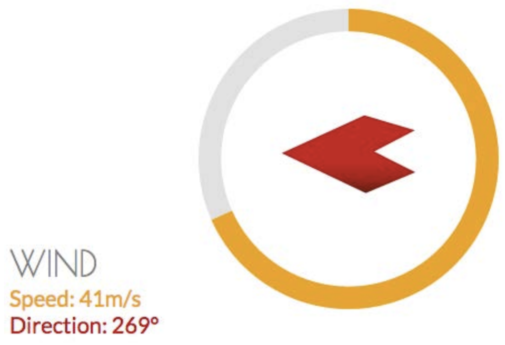


Figure 33 Wind Direction and Speed

This component was created for visualizing both a direction and a value that has a connection to each other. On *Figure 33* Wind Direction and Speed, the component is used for describing wind conditions. The arrow in the middle points out the direction of the wind, and the radial bar around describes the wind's speed.

The component can be used for visualizing other data. Some examples of scenarios where this component may be useful are for visualizing:

- direction and speed for engines
- direction and power consumption of thrusters
- the vessel's heading and speed

The illustration is assembled by two components: an arrow and a radial bar. The arrow is an SVG element created in Adobe Illustrator, and the radial bar is an SVG element created using the D3 library.

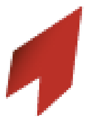


Figure 34 Directional Arrow

The directional arrow is an SVG element that is being loaded into the index file as an image by using the <img> tag. The reason why it is being loaded as an image instead of an SVG element is that it is easier to rotate an image using CSS.

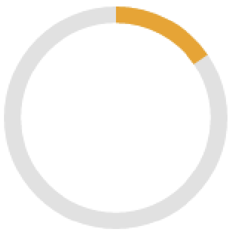


Figure 35 Radial Bar

The radial bar is an SVG element developed by Brightpoint[22] using the D3.js library. It is created by using the d3.svg.arc() function, which constructs a new arc generator. That means that it returns a generated path data for a closed solid arc.

22 http://www.brightpointinc.com/download/radial-progress-source-code/

7.3 Liquid Fill Gauge

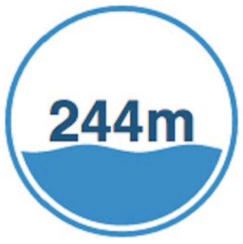


Figure 36 Liquid Fill Gauge

The liquid fill gauge was developed by Curtis Bratton using the D3 library[23]. It is a simple, but descriptive component. It has an animation feature that makes the blue area act like a wave using a sinus-function.

The component was implemented in our visualization for visualizing water depth and/or wave height.

7.4 Pitch and Roll

Pitch and roll gives a description of how weather conditions affect the vessel. In order to visualize pitch and roll, two SVG elements illustrating vessels from front and profile were created. These SVG elements were created in Adobe Illustrator, and imported to the apllication as images. These images are being rotated using CSS in a JavaScript function.

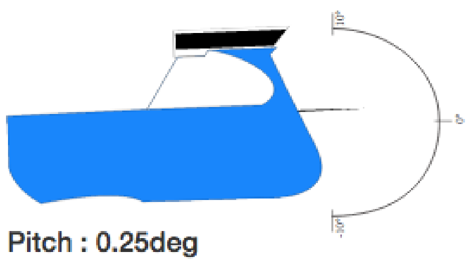


Figure 38 Pitch

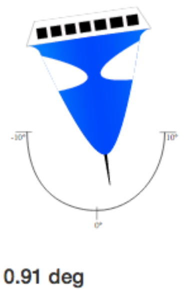


Figure 37 Roll

7.5 Review

Trying to find different elements to use with suitable data have been interesting. We are left with several ideas for further expansion for some of the elements. These elements are results of exploration in the D3 library. The radial bar and the directional arrow is a component we can vision in several areas of use.

Since the idea of visualizing weather conditions as a heatmap was scraped, some of these elements were implemented as substitutes. What is positive about these elements is that they use actual, registered values instead of values from weather forecasts. In that way, these elements give more accurate information about the conditions. Still, a heatmap would provide a bigger picture of the conditions, as these elements only display the values at one point at a given time.

23 http://bl.ocks.org/brattonc/5e5ce9beee483220e2f6

8 SYSTEM INTEGRATION

With a lot of different separate components, the group wanted the experience of trying to put them all together in one working application. The timeline described in chapter *4.2* was set as the cornerstone for connecting all the different components into a single application. By connecting all the components to a single time reference, the information displayed by each component could be put into a larger context.

During the first attempt at interconnecting all the components, we tried to connect the timeline and the video components through a switching master-slave relationship. This made for an overly complicated GUI and a second attempt at connecting the applications was made.

During the second attempt, we remade both the video component and the timeline. This was necessary to simplify how the components interacted, and to simplify the GUI.

With a number of different components in our application we decided on putting some effort into the design and layout of the application. This was introduced at a late stage and was outside of our main scope at the start of the project. Thanks to some good advice from Kjetil Nordby at AHO, we made some simple implementations to improve the interaction design and layout of the application.

How the back-end and the front-end solutions of this application is connected was introduced in *3 Back-End*. The components that have their own server-side-script are the timeline, multi chart, map, line graph and sunburst. In this chapter, we will describe how these components connect the other components in the application.

The result of the full application with all components synchronized to the timeline is illustrated in *Figure* 39. It also displays the finished layout and design of the application.

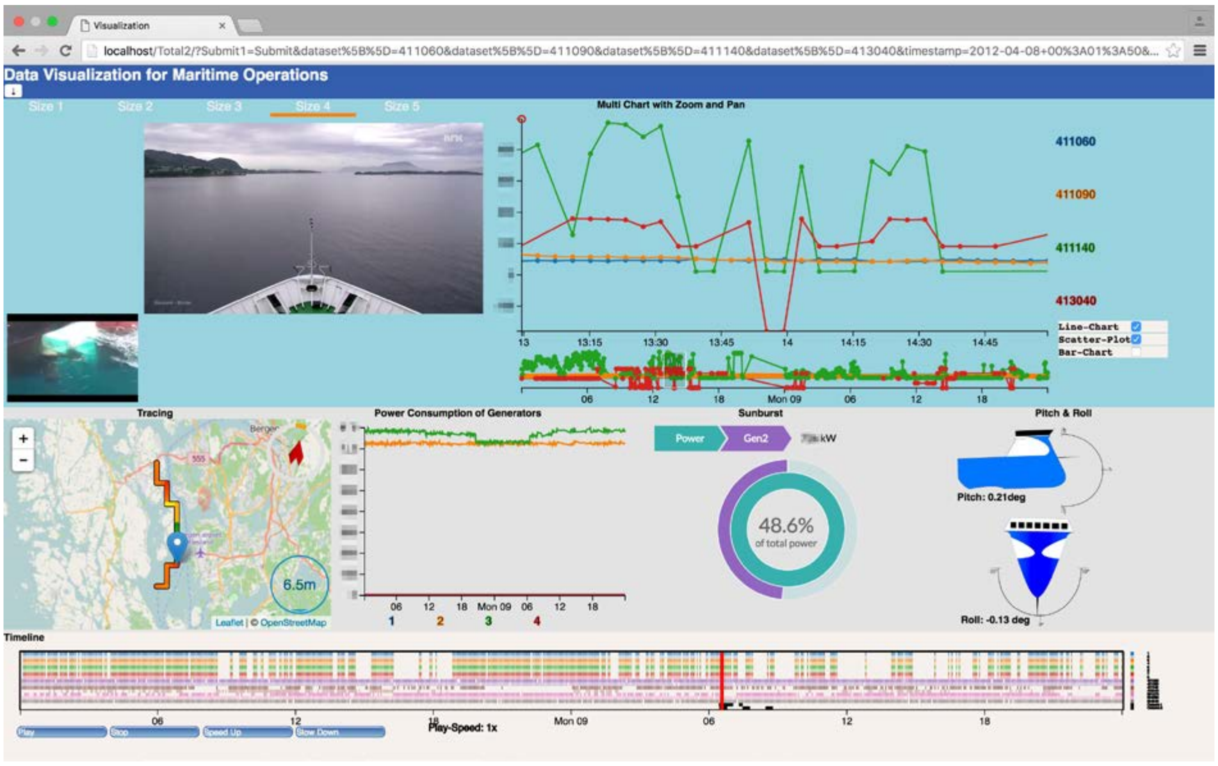

Visualization
localhost/Total2/?Submit1=Submit&dataset%5B%5D=411060&dataset%5B%5D=411090&dataset%5B%5D=411140&dataset%5B%5D=413040&timestamp=2012-04-08+00%3A01%3A50&...
Data Visualization for Maritime Operations
Size 1
Size 2
Size 3
Size 4
Size 5
Multi Chart with Zoom and Pan
411060
411090
411140
413040
Line-Chart
Scatter-Plot
Bar-Chart
Tracing
Power Consumption of Generators
Sunburst
Pitch & Roll
Power
Gen2
kW
48.6%
of total power
Pitch: 0.21deg
Roll: -0.13 deg
6.5m
Leaflet | © OpenStreetMap
Timeline
Play
Stop
Speed Up
Slow Down
Play-Speed: 1x

Figure 39 The Finished Application

8.1 First Attempt

8.1.1 Video

The general idea was that each HTML5 video element would represent a specific camera on the ship, or from the field study. By sorting the video files by the camera used and placing them on the timeline based on their start date and duration, we make sure that no elements representing the video files are drawn on top of each other. Since it cannot exist multiple video files from the same camera in the same period of time, we ensure that the video files are sorted properly. Sorting which video files go where would be hard without any specific signal id, file location or in this case the camera name. Since the timeline contains the date representations of the video files, it will sort through the data and find the closest videos in any instance where the time bar is moved. By finding the closest videos, we can continuously update the videos in the HTML5 video elements based on where the time bar is located on the timeline.

In this prototype we have put together two HTML5 video elements as shown in *Figure 40*.

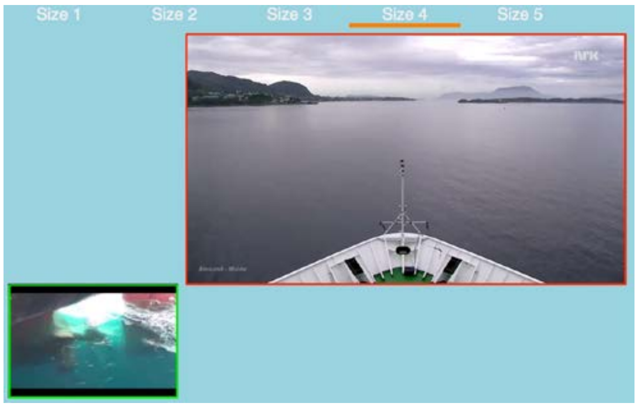


Figure 40 Video Synchronization with Master and Slave

On startup, both video elements will be displayed with a green borderline. This signals that neither of the videos acts as a master to the other. By clicking one of the video elements, the border color will be set to red. This signals that the red video is now master to the green one. By clicking the red one again the master status will be removed. By clicking the slave video, the master status will be transferred to that video element. This allows the user to easily change the master video without a lot of instructions on how to do so. All actions done on the master video will be transferred to the slave. If a master video is chosen while one of the videos are playing, the slave video play status will be set equal to the master video play status. In all events, the current time of the slave video will be set relative to the current time of the master video.

The time offset between the videos are checked ten times per second. If the offset between the two videos is more than 0.2 seconds, the slave video will be set to the relative current time of the master video. If we set the offset between the videos as small as 0.5 or 0.1 seconds, the synchronization operation would not finish in time. This would only cause a loop of synchronization and the slave video would never run smoothly. The compromise was as mentioned above, an offset of 0.2 seconds between the master and the slave.

Because the video elements require a very specific framerate in order to run smoothly, we decided on letting them share control of the timeline whenever a video is playing. The alternative would be to try to match the specific framerate of the video from our timeline play function, something that would prove difficult.

Whenever play is being pressed on a video element (or both by using master), the timeline is stopped and an interval is set using the setInterval() method. This interval will call the sync() function ten times a second until the interval is cleared by pressing pause on the video element, or play on the timeline. The sync() function will then call a public function within the timeline script, this public function will be referred to as VideoOverride() and takes a timestamp as parameter. The VideoOverride() then moves the time bar based on the parameter value and updates all the other components on our web page. The parameter value is derived from the start date added to the current time of the video. Whenever pause is pressed on the video element, the interval is cleared. Because of the ability to toggle between master and slave, we actually have four different sync functions. One for each acting master and one for each solo video element.

While the video element is playing, it is still possible to adjust the time in the video, or changing videos by pressing the timeline. It is also possible to change the position of the time bar in the timeline by pressing the timeline of the video element. Whenever the timeline play button is being pressed, the videos are stopped and are instead set to show screenshots as the time bar progresses through the video. While either the timeline or the video is playing, a function for finding the closest videos are being called. The closest videos will be loaded based on the position of the time bar in the timeline.

Figure 41 is a representation of how the videos might be placed on the timeline.

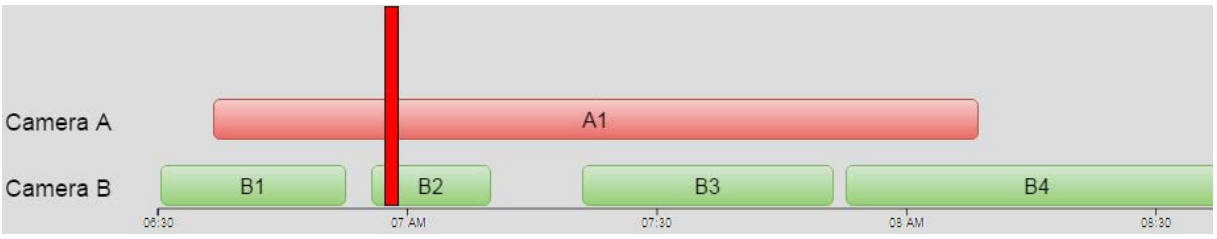


Figure 41 Video Placement Timeline

The red vertical bar is the timeline time bar. The boxes B1 – B4 represents video files filmed by camera B. The box A1 represents a video file filmed by camera A. As mentioned earlier we have two video elements in our web page. One element will load video files filmed by camera A and the other video element will load video files filmed by camera B. These two elements will be referred to as video element A and video element B.

Given the scenario in *Figure 41*, say the user were to select the video element A as master. When the user presses play on the video element A, the videos A1 and B2 would start playing from where the time bar is located. When the time bar reaches the end of video B2, the video B2 would stop. When the time bar is just past the middle of the space between videos B2 and B3, video element B will load and pause video B3. When the time bar reaches the start of the video B3, the video B3 will start playing. It will continue like that until it reaches the end of video A1, which is about 20 minutes into video B4. The progress in the videos can be adjusted at any time by pressing the timeline or the timeline in the video element, thus changing the position of the time bar.

Another scenario would be for the user to select video element B as master. From *Figure 41* this would mean that the master has currently loaded video B2. When the time bar reaches the end of video B2 the video would pause and nothing else would happen. In order for the user to continue viewing the videos, he/she would either have to select video element A as new master, or click on the timeline close to videos B1, B3 or B4, thus loading a new video file in the master element.

As already mentioned, the video elements continuously load the videos closest to the time bar. If the time bar is outside the range of the video as play is pressed on the video element, the time bar would jump to the start of the closest video and start playing. If the user were to start both the videos at different times without using the master system, the time bar will jump between the two different time values of the videos.

Synchronizing the two video elements against the timeline took a lot of work, and was clearly the hardest component to implement with the timeline. Numerous event handlers had to be created for all the different scenarios. Because the video current time is always relative to the timestamp of the video, some of the if statements are almost unreadable.

8.1.2 Interaction

All the components on the web page act as slaves to the timeline, the exception being the video elements as mentioned in *8.1.1*. All the components contain a public function. This allows the timeline to call each of the other components whenever they should update their current values. *Figure 42*, shows a model of how the different components communicate.

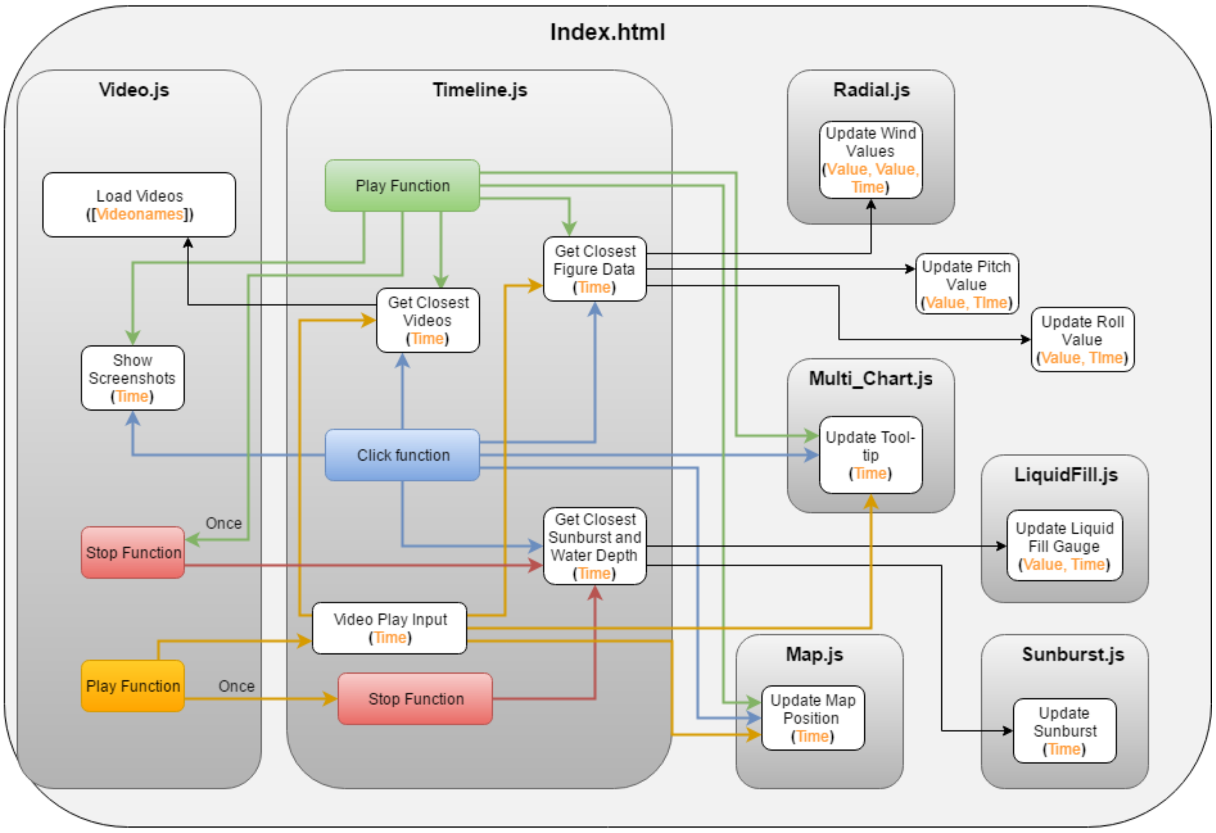

Index.html
Video.js
Load Videos ([Videonames])
Show Screenshots (Time)
Stop Function
Once
Play Function
Once
Timeline.js
Play Function
Get Closest Videos (Time)
Get Closest Figure Data (Time)
Click function
Get Closest Sunburst and Water Depth (Time)
Video Play Input (Time)
Stop Function
Radial.js
Update Wind Values (Value, Value, Time)
Update Pitch Value (Value, Time)
Update Roll Value (Value, Time)
Multi_Chart.js
Update Tool-tip (Time)
LiquidFill.js
Update Liquid Fill Gauge (Value, Time)
Map.js
Update Map Position (Time)
Sunburst.js
Update Sunburst (Time)

Figure 42 Communication Overview

Figure 42 only shows how the functions in the different scripts interact with each other. There are a lot more functions and operations being executed in each script, not being shown in *Figure 42*. Each gray box represents a script. Each box within the script represents a function. All functions with an input parameter are white. The input of the white functions is displayed within the parentheses. Key functions directly connected to the GUI is colored. The function names in *Figure 42* explain roughly what the function does.

The multi chart and map components both contain their own data for the entire length of the timeline. The map contains all the position data, and the multi chart contains all the data for whatever it is showing. Each time the time bar is moved, the update functions within the multi chart and map components are being called. These update functions only require a timestamp parameter. The map and multi chart components will then do the bisection (explained in detail in section 4.1.5) themselves, and find the closest data to the timestamp. The map component will then update the marker location on the map. The multi chart will update the positions of the tooltips. Both components will update in any event the time bar is moved.

The components showing wind speed and direction, pitch and roll only shows data for a specific point in time. These components do not contain any data, except for the time they are showing. Instead, the timeline component does the operation of finding the closest data segments for them. When the closest data segments and timestamps for the segments are found, the public functions within these components are called. All these components are being updated in any event that update the time bar location.

The sunburst and the liquid fill gauge are being updated by using AJAX. AJAX will update the entire content of the div containing the component. The timeline handles the bisecting, and sends the closest data values to the liquid fill gauge. The sunburst on the other hand, is only passed a timestamp from the timeline. The sunburst still needs to collect the data itself, from the database. Because all of this is time consuming, neither the sunburst nor the liquid fill gauge is updated by the play function of the timeline or video. Instead they are updated by click on the timeline, or when the stop function is pressed while playing.

The video Component overrides the timeline through the public “video play function” shown in *Figure 42*. The “video play function” within the timeline component is being called ten times a second from the video component, if a video is playing. The updating of all the other components happens through that public function. The function first moves the time bar, then updates all the other components except the sunburst and the liquid fill gauge. Ten times a second might be a bit excessive for most IAS data, but can easily be limited by only updating the components every “X” times the function is called.

8.1.3 Interaction Problems

The Sunburst and the Liquid Fill Gauge are not yet compatible with the timeline play function. This is because they do not yet contain any update functions. Instead, the entire div elements containing the components are being reloaded by using AJAX. Making AJAX requests within a function looping at a high frequency would quickly exceed the browser’s limit for concurrent connections to the same domain. If the browser limit is exceeded, pretty much everything in the application would stop working properly. Instead, the Sunburst and the Liquid Fill Gauge are only updated when clicking the timeline, or when stopping the play function. Making functions for updating the values of these two components is possible, but would require more time than we can afford to spend this late in the project.

The real problem during this phase of the project was the timeline's lack of ability to play in a real time scale. Because of the way the timeline is set up, there is no way of getting it to play in a real time scale. The reasons why are better explained in section *5.4.2*, but can easily be summed up to bad design choices. To try to make up for this problem, the video component and the timeline component switches between being master and slave. This solution actually worked to some degree by making the video component overly complex. Even though the solution of letting the video component synchronize the timeline worked, it also made for an overly complicated GUI.

A better solution would be to remake the timeline play function in such a way that it is referenced to real time. The videos could then be synchronized to the timeline, which would make the structure of the application as a whole, much better. The video component would also have had to be restructured, but it would be an easy task compared to how the video component is structured now.

8.2 Second Attempt

In this chapter we will cover the "last minute" implementations made to improve the interaction between the different components. This was our second and final attempt at making the components interact in a decent manner.

8.2.1 The Timeline

Our main problem during the first implementation was the timeline's inability to play in a real time scale. To solve this problem, we made an entirely new play function for the timeline.

The new play function uses the setInterval() method instead of the setTimeout() method. The setInterval() method is mentioned briefly in *5.4.3*. The big difference between the setTimeout() and the setInterval() methods is the accuracy between executions. A timeout executes a certain amount of time after the setTimeout() is called. An interval executes a certain amount of time after the previous interval was initiated.

Say we create a looping function using the setTimeout() method. The function will then wait 1000ms, call the function and execute the commands, which take a few milliseconds, wait 1000ms and call again. This makes the waiting period a bit more than 1000ms, and can cause a big offset over time.

The interval on the other hand, will attempt to execute exactly 1000ms after the last interval initiated. If the JavaScript is busy doing something, like handling the last interval, the interval is remembered, and is executed as soon as the control is returned to the browser. So, intervals actually try to “catch up” to get back on schedule whenever they are running behind. If the intervals start to stack up, the old interval will simply be discarded.

The movement of the time bar in the old timeline play function was based on pixels. In the new function it is based on time. The interval calling the play function is set to execute every 100ms. Every time the play function is executed, the position of the bar in the time domain is increased by 100ms, and most of the other components are being updated. The timeline also contains a counter. This is to have the option to limit how often certain functions are being called. Most of the components do not need to be updated ten times per second. By using the counter to only update some functions every fifth time the function executes, some stress is taken off the play function.

When changing the speed of the timeline, we actually increase the length of each jump the time bar takes. 10 x play speed would increase the jump length to 1 second instead of 100ms. This function could perhaps be improved with some mathematics. At the same time, if the user wants higher speed, it is probably because the samples are far apart or span over a long period of time. If the user wants to play at high speed, it is probably not very interesting to see every single sample. This, and because we were pressed for time, are the reasons the speed up function works as it does. The options available in play speed are from 0.25x to 1000x real time.

8.2.2 Interaction

All components on the web page are slaves to the timeline with no exceptions. All the components contain a public function. This allows the timeline to call each of the other components whenever they should update their current values. *Figure 43* shows a model of how the different components communicate.

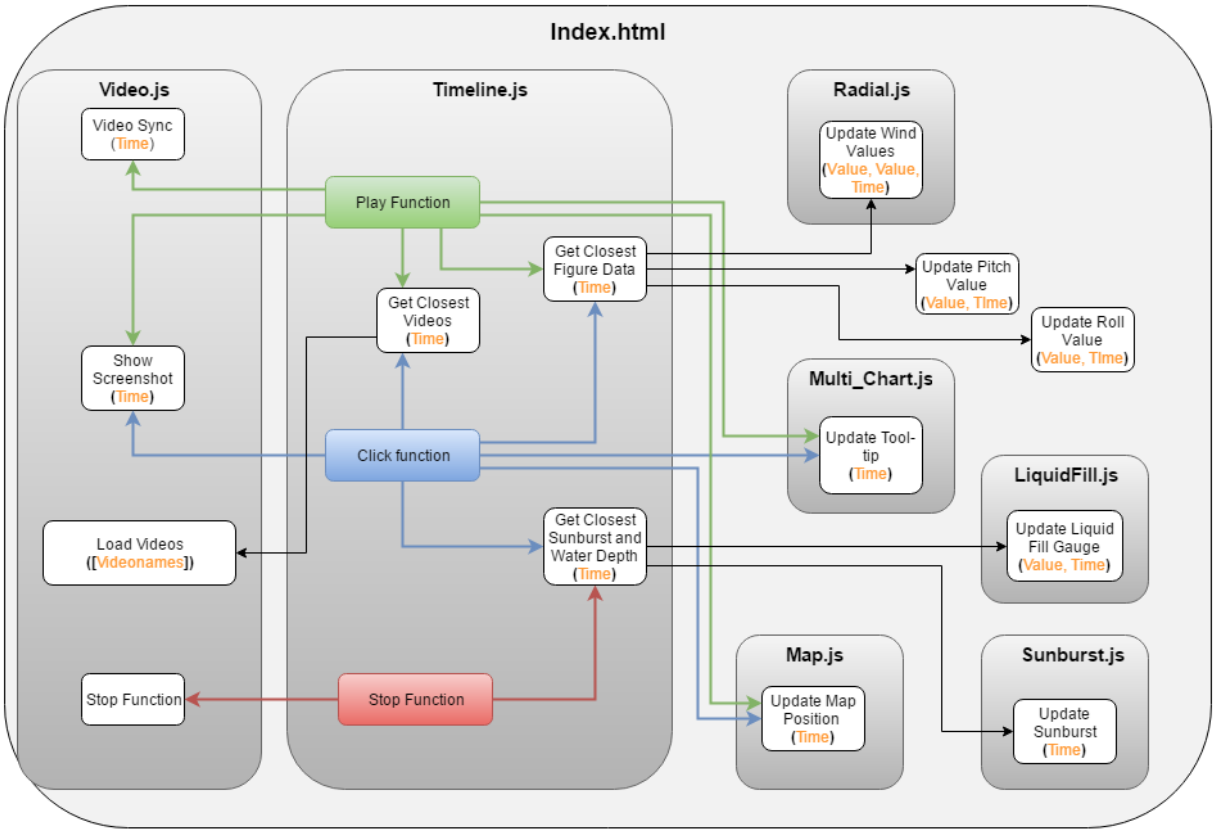

Index.html
Video.js
Video Sync (Time)
Show Screenshot (Time)
Load Videos ([Videonames])
Stop Function
Timeline.js
Play Function
Get Closest Videos (Time)
Get Closest Figure Data (Time)
Click function
Get Closest Sunburst and Water Depth (Time)
Stop Function
Radial.js
Update Wind Values (Value, Value, Time)
Update Pitch Value (Value, Time)
Update Roll Value (Value, Time)
Multi_Chart.js
Update Tool-tip (Time)
LiquidFill.js
Update Liquid Fill Gauge (Value, Time)
Map.js
Update Map Position (Time)
Sunburst.js
Update Sunburst (Time)

Figure 43 Improved Communication Overview

Figure 43 only shows how the functions in the different scripts interact with each other. There are a lot more functions and operations being executed in each script, not being shown in *Figure 43* . Each gray box represents a script. Each box within the script represents a function. All functions with an input parameter are white. The input of the white functions is displayed within the parentheses. Key functions directly connected to the GUI is colored. The function names in the figure explain roughly what the function does.

The Video Component

Because the timeline now plays in a real time scale, a lot of the problems concerning video synchronization were eliminated. Most of the code in the video component could be removed since it mostly handled user input. The video component now works as a slave for the timeline. The videos will be played when the time bar on the timeline is within the period of the video. Videos are started or stopped as the timeline enters or exits their period.

A function within the video component is responsible for making sure the videos are in sync with the timeline, this function will be referred to as the video Sync() function. The video Sync() function is called from the timeline play function with a timestamp parameter. The video Sync() function then checks the input parameter against the time of the videos. The time of the videos are given as current time of the videos added to the start date of the videos. If the difference between the timestamp received from the timeline and the time of either of the videos is too large, the video Sync() function will make adjustments. The adjustments are done by setting the time of the video or videos lagging equal to the timestamp received from the timeline. The accuracy of the video Sync() function can be increased by taking into account the time the video “set current time” operation takes to execute.

All other components interact just as they did in the first attempt *8.1* The only difference being that the videos can no longer master the other components through the timeline.

8.3 *Interaction Design and Layout*

The design and layout of this application was developed from this this mock-up.

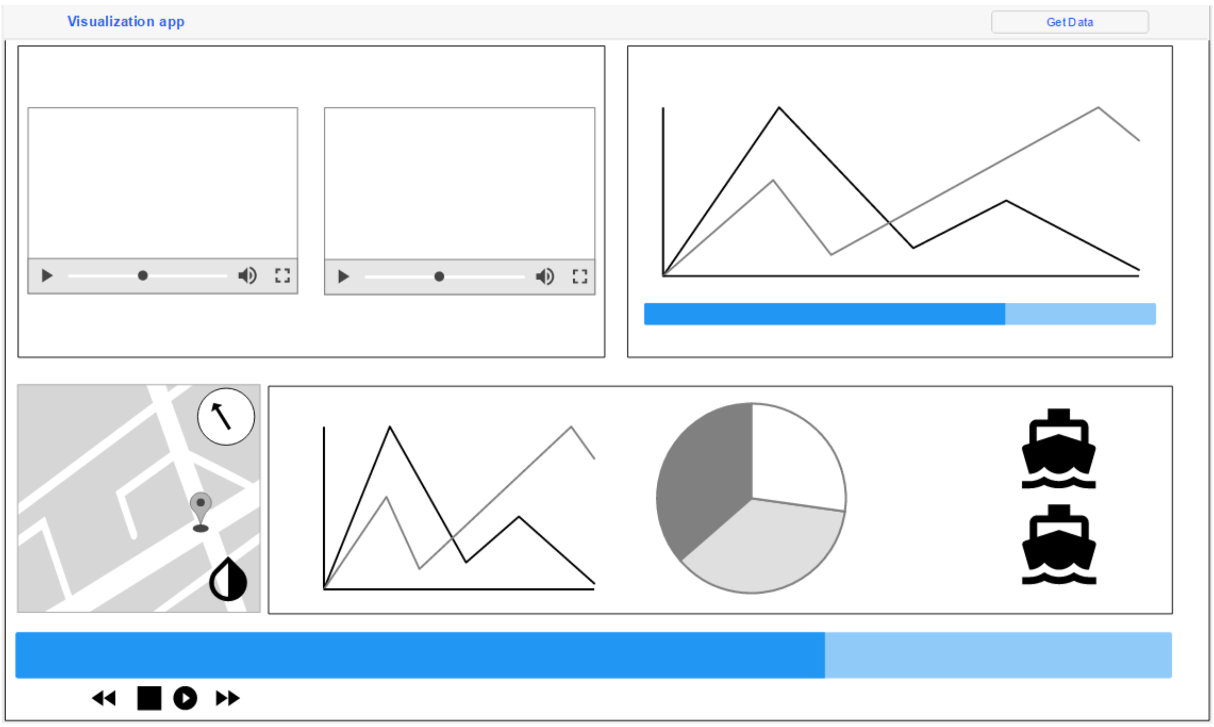


Figure 44 Layout Mockup

During this project, the group has focused on creating a visualization that is both interactive and dynamic. The focus has been to use the concept of human-computer interaction to create and provide interaction design and user experience in a human-centered way.

An interactive system has the purpose of processing information. We have considered how the graphic design and the presented information influence the user. The goal has been to amplify human cognition in users by enabling them to see patterns, trends and anomalies in the data in order to gain insight. This can help the user to enhance discovery, decision-making and explanation of phenomena regarding the data. This visualization is developed for experts and analysts in order to help them understand and make sense of large amounts of dynamically changing data. The idea is to let them detect patterns in the data, and draw interferences from them. (Benyon, 2014) (Jenny Prece, 2015)

To reach these goals, we have focused on methods for using interactive techniques with presentations of large quantities of data. Creating good information layout, and reducing search for information has been two key elements on the way toward the goals.

"Overview first, zoom and filter, then details on demand."[24] This is Ben Schneiderman's information-seeking mantra. The group has followed his idea, and had the intension of creating the visualization in a way that lets the user drill down into the data.

The way we have followed this thinking is by creating a timeline that gives an overview of all the data that is visible on the web page. The timeline illustrates all visual components, and lets the user know at what times the different data is available. The user can zoom in to get a better view of every available data sample.

Other components, such as the multi chart, sunburst and map also gives the user selection options. The multi chart gives the user an overview of the data with the ability to zoom in and filtrating which data to view. Details are revealed when the mouse pointer is positioned over the element. The map lets the user zoom, and further details are revealed when clicking the marker. The sunburst provides details when the mouse pointer is positioned over the element.

As for the application as a whole, we have considered how different elements are connected. We have considered how texts and fonts correspond to each other. The layout of the page is intended to place elements in such a way that they supplement each other's content. The application has all the elements lined up with an appropriate amount of space between them to ensure that the user is not distracted away from the relevant information on display. The different data are color coded to make sure they are distinct and differentiable.

8.4 Review

System Interaction

Synchronizing data against the timeline worked well for most of the different components. The problems regarding the Sunburst and the Liquid Fill Gauge components are still as they were described in section *8.1.3*. Starting with the second attempt at

[24] http://www.codingthearchitecture.com/2015/01/08/shneidermans_mantra.html

system interaction, we aimed to make a new timeline play function and a new video component. This was successful and the group is satisfied with these last minute implementations. With the new timeline play function, the GUI is a lot easier to understand and the communication between the components is more straightforward.

It would be possible to set a reference in the timeline to an external clock. This could be used to check how far the timeline has drifted out of real time scale since it started. We are not sure if it would be preferable to adjust the time of the timeline according to an external clock, as this could cause the timeline to skip parts of the data.

As the application is now, videos are only played when the speed of the play function is set to 1x. If the speed of the timeline is any higher or lower than 1x, the videos will only show screenshots. It would be possible to change the video speed as the timeline speed changes, all the way up to at least 10x. This was not implemented because of time restraints.

Reviewing Interaction Design of Components

Because the group was more focused on the technical solutions, there was no real focus on design before the system integration.

In finalizing the project, we have considered how well the different elements present information. Feedback from Kjetil Nordby at AHO has been valuable during this process.

We have realized that some of the elements are not self-explanatory enough and do demand some kind of user guidance to be fully utilized. Elements such as the sunburst may not be as well suited of displaying the ratio between the different contributors, as originally thought. It might be easier to compare different contributions when aligned in for example a bar chart, according to Kjetil. Also the component for measuring the wind speed and direction is somewhat misgiving. Using a circle for displaying wind speed was simply a poor choice of model. To put a theoretical maximum value for speed may be misgiving and hard for the user to relate to. A circle would be more suited for representing percentage values. Given more explanation and detail, the element could still be useful in the application.

9 DISCUSSION

This report proves that it is possible to use the web as a platform for data visualization. There are many opportunities for incorporating different web APIs. This project has consisted of finding and combining different APIs in order to create a visualization that is suitable for data collected from maritime operations. Aspects of the solution can be referred to as a proof of concept, such as the video synchronization. The aim has been to identify opportunities for web-based data visualization suitable for the maritime industry rather than creating a full, ready-to-use application for data visualization.

Some questions we reflect upon in this chapter are listed below.

- What is the value of the work?
- What potential consists for further development?
- What have we learned from this project?

9.1 Value of the Work

When we review the results from this project, we can either consider the result of the project as a whole, or the worth of each technical solution. The details of the technical solutions are being reviewed in the relevant chapters.

At the end of the project we met with our contacts, André Keane and Ulrikke Brandt, and let them test the application to give us some feedback.

The map, multi chart, videos and timeline were considered to be the most interesting components of the application. These were components that may be worth developing further. The interaction between the different components was interesting, and more cross interaction would improve this concept. The timeline as solo master seemed to be a bit strict. Letting other components have the possibility of controlling the timeline as well as each other, would be a useful addition.

When we review the value of this work, we consider feedback from Ulstein and AHO along with our personal opinions. We have considered some of the elements in the application to be more fundamental in the process of data visualization than others. Our opinion is that this project serves as a marker on how HTML5 and different web API's can be used to create web-based visualizations. Linking together different data through a common timeline has hopefully enhanced how combining visualizations can give a broader picture of the data as a whole. A well-presented visualization of data has the potential of making the process of analyzing data easier. It may help the process of finding and interpreting context of data. We believe that some of the visualization elements such as the multi chart, map, video implementation and the principle of the timeline can be useful in the process of making data analysis easier. By combining location and videos with numeric data, the understanding of operations can be enhanced. It can help reveal details about operations that would be hard to understand from the data alone. The components together tell a deeper story about the ship's operations. We see potential in creating cross interaction between the components, and letting them affect each other.

Hopefully, some of the solutions we have come up with during this project are original, and will inspire further work.

9.2 Possible Improvements

Moving forward with the project, there are several features that have the potential of improving the current version application.

Increasing Performance

The system lags when handling large amounts of data. Especially the zoom behavior of the timeline and the multi chart, are demanding and will cause desynchronization. Reducing the number of elements being moved or redrawn will improve this problem. Instead of hiding unused elements by making them transparent, these elements should be removed to reduce the number of elements being manipulated.

The fastest way to increase the performance would be to adjust the visualizations for the human eye. Take the timeline as an example. Say the timeline visualizes a year of data sampled every five seconds. Assuming that there is no missing data, this would sum up to 6 307 200 elements. Our SVG element is at most 2000 pixels wide and each bar is 1 pixel wide. With the width of each bar set to 1 pixel, the largest possible number of elements placed without any overlapping would be 2000. Any more bars would not be visible to the human eye. By not allowing elements to overlap, the same results can be achieved by drawing only a fraction of the elements. When the group realized this fundamental flaw in resource management, there was not enough time left to implement the solution. This solution is something that would greatly improve the speed of both the timeline and the multi chart.

The Canvas JavaScript library should be explored as a possible replacement for D3 for a timeline such as this. If there is no need to interact with each element created, Canvas could handle the large amounts of data better.

Video Handling

A better option for sorting and loading video files is needed. Accessing the metadata was only explored using either HTML or JavaScript. Later, we learned that it is possible to access metadata using for example PHP. Setting up a database could be used for storing large amounts of videos.

9.3 Ideas for Future Development

Some of the ideas that were discussed during the project period, we consider as good implementations for this application.

Annotations

Allowing the user to write annotations while analyzing the data could increase the overall value of the application. The annotations could be given a timestamp and added to the timeline, represented as a bar. The annotations could relay information between different users and be loaded by the timeline in the same manner as the current data.

Determining Operational Phases

The possibility of automatically dividing ship's operations into different phases based on data is something that has been discussed in relation to this project.

Dynamic Web Page

Setting the elements of the application movable and resizable manually could make it easier for the user to focus on relevant elements for a given scenario.

9.4 Learning Outcome

The group had experience with object-oriented programming and industrial control systems prior to this project. Having no experience with web development gave the process of developing this application a steep learning curve.

During this project, the group learned much about web development. The project gave experience in many aspects of web development, mostly regarding front end, but also some back end. The project's open development nature made it challenging to determine early on what was important to have in the finished application and what was not. This made it hard to visualize an end-goal at the early stages, and the development were an open task with the possibility of taking different paths.

The development methodology that was followed by the group was not always the strictest considering project management, but fostered an environment of learning. The way the project was developed, enhanced the group's understanding of web development. Much time was spent on researching different problems in order to gather information. This has given a broad understanding of how web development works. The group also explored subjects that may have had promising utilities for the application, but in the end proved to be of little to no use. A lot of time was spent on experimenting with prototypes to better understand certain elements, and trying to apply them for use in the application.

Tasks that proved to be larger and more comprehensive than originally presumed were put on hold. If there were time, development would start again after the core functions of the components were implemented. Since the amount of features the group wanted to include in the project was extensive, many features were postponed, and some did not make it in the final build of the application.

The group has done considerable growth in both the understanding of how web technologies work as well as the skills needed to develop them.

9.4.1 Communication

Since several parties have invested interest and involvement in this project, communication was necessary. This resulted in a lot of constructive feedback during the project, but also different ideas and opinions that could lead the project in multiple directions. It could sometimes be difficult to decide which feedbacks to focus on. It has been a valuable experience, and given us an idea of what it is like to work in projects with multiple parties involved.

CONCLUSION

HTML5 proved to be a good platform for this kind of application. The possibilities of incorporating different web APIs and combining them to one application make HTML5 an interesting platform for data visualization.

The accessibility of HTML5 is the main reason why it is considered a useful tool. The fact that it can be used by anyone on any device with a modern web browser with no extra installations needed makes it desirable for gathering and displaying data. HTML5 has the possibility of incorporating server-side scripting languages such as PHP in order to retrieve data dynamically from a database. The possibility of having large amounts of data to back up an application strengthens the accessibility of HTML5 even more. HTML5 is what provides a close cooperation between the dynamically retrieved data and visualization elements.

The prototypes developed during this project gives a marker as to how web developed visualizations can be formed.

The most interesting components are the timeline, multi chart, map and video. These are components the group believe are worth developing further. The timeline works well as a navigational tool that binds all the different components together. The group is satisfied with the principle of the timeline, not only acting as a common time reference, but also conveying information about the data used in the visualization.

Elements that were not so efficacious were the static components. These did not add as much information to the application. They were not detailed enough, so the information the user could gather from them, were limited. This does not discard the whole concept of having static visualizations, but a different approach to them is necessary to make them useful in the application.

10 REFERANSER

A Web Developing Cat. (2012, Desember 13). *PHP, MySQL Tutorial for beginners – connecting to the database with PDO*. Retrieved from http://webdevelopingcat.com/: http://webdevelopingcat.com/php-mysql-tutorial-for-beginners-connecting-to-the-database-with-pdo/

Benyon, D. (2014). *Designing Interactive Systems.* Person Education.

Flanagan, D. (2011). *JavaScript: The Definitive Guide(6th edition).* o'reilly.

Fowler, M. (2002). *Patterns of Enterprise Application Achitecture 1st edition.* Addison-Wesley Professional.

Freeman, A. (2011). *The Definitive guide to HTML5.* Apress.

Jenny Prece, Y. R. (2015). *Interaction Design 4th edition.* John Wiley & Sons.

Mozilla developers. (2016, jan 19). *Mozilla developer*. Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats

Pfeiffer, S. (2010). *The Definitve Guide to HTML5 Video.* Apress.

PHP. (2001). *What is PHP?* Hentet fra php.net: http://php.net/manual/en/intro-whatis.php

Ullman, L. (2012). *PHP and MySQL for Dynamic Web Sites (4th edition).* Peachpit Press.

Waterson, K. (u.d.). *Introduction to PHP PDO*. Hentet fra Phpro: http://phpro.org/tutorials/Introduction-to-PHP-PDO.html

wikipedia. (2016, may 26). *wikipedia*. Hentet fra information security: https://en.wikipedia.org/wiki/Information_security

APPENDIX