BUSKERUD AND VESTFOLD UNIVERSITY COLLEGE

CDPBrowser

Mobile Software Engineering

BUSKERUD AND VESTFOLD
UNIVERSITY COLLEGE

Hovedoppgave

OPPGAVE NR	DA-2014-04

Institutt for ingeniørutdanning i Vestfold

Besøksadr.: Raveien 215
3184 Borre
Tlf: 33031000, Fax: 33031100

TILGJENGELIGHET

- ☐ **ÅPEN**
- ☐ **ETTER AVTALE**
- ☐ **KONFIDENSIELL**

NORSK OPPGAVETITTEL	CDP Browser
ENGELSK OPPGAVETITTEL	CDP Browser
FORFATTER(E)	Stian Fanebust Per Torgrim Frøstrup Thorbjørnsen Kim Eskedal Utheim
DATO LEVERT	
ANTALL SIDER	
OPPDRAGSGIVER	ICD Software Ltd
REF. /VEILEDER(E)	Geir Varholm Thomas Nordli
3 STIKKORD	Mobile device, Android, Qt
SAMMENDRAG	

BUSKERUD AND VESTFOLD
UNIVERSITY COLLEGE

SAMMENDRAG (forts.)

1 Abstract

The aim of this project is to answer the problem:

What considerations need to be taken in to account when designing and creating a prototype browser on smart devices for "CDP-controllers"?

The "CDP controller" is a computer running software developed with "CDP" *(i.e. software developed by ICD Software Ltd)*, containing logics for process control. This "CDP controller" can manage the operation of any number of things like; the operation of a crane, a draw bridge or everything in a car.

To produce the prototype for this project, there will be focus on identifying viable way for "Industrial Control Design" *(ICD)* to develop applications or "Apps" for mobile devices. This will be done through research in to mobile software engineering concepts and how these are implemented in available platforms. After which they are weighted against each other in concern to "ICD Software Ltd" current and future plans.

Another focal point of this project will be the "Graphical User Interface" (GUI). The GUI need to give a good overview of the different controllers on the network and their corresponding components and members *(e.g. signals, parameters, alarms and messages)*.

Implementation to produce a working prototype requires the project team to learn a new coding language and IDE. The experience acquired in this process will help to identify different possibilities on how to produce the desired prototype.

The scope of this thesis will cover collecting data from controllers about their inherent objects and how to change their key attributes. Given the projects time limit and resources any other interaction with the controllers through the prototype, will have to be the subject of derived projects.

The principal reason for this project is the expanding market for mobile devices, and how they become more and more common in the interaction with programmable units. As such, any byers of the "CDP" software might come to expect an inherent possibility to interact with their developed controllers from a mobile device.

BUSKERUD AND VESTFOLD
UNIVERSITY COLLEGE

Skjemaet skal leveres sammen med besvarelsen.

Obligatorisk erklæring

Jeg erklærer herved at min:

Eksamensbesvarelse i emnekode:	FE-BAC3000	**Fakultet:**	TekMar

1. er utført av undertegnede. Dersom det er et gruppearbeide, blir alle involverte holdt ansvarlig og alle skal undertegne blanketten.
2. ikke har vært brukt til samme/en annen eksamen ved HVE eller et annet institutt/ universitet/høgskole innenlands eller utenlands.
3. ikke er kopi eller avskrift av andres arbeid, uten at dette er korrekt oppgitt.
4. ikke refererer til eget tidligere arbeid uten at dette er oppgitt.
5. har oppgitt alle referanser/kilder som er brukt i litteraturlisten.

Jeg/vi er kjent med at brudd på disse bestemmelsene er å betrakte som fusk og behandles i hht. §18 i Forskrift om eksamen og studierett ved HBV og U-loven Kap. 4 § 4-7.

Dato:		Sted:	

Underskrift[1]:		Kand.nr.:	

Ved gruppebesvarelse må alle gruppas deltagere undertegne[1]:

Underskrift[1]:		Kand.nr.:	
Underskrift[1]:		Kand.nr.:	
Underskrift[1]:		Kand.nr.:	

[1] Hvis erklæringen leveres inn elektronisk via Fronter er underskrift ikke nødvendig, skriv da inn navn.

BUSKERUD AND VESTFOLD
UNIVERSITY COLLEGE

Hovedprosjektets tittel:

Forfatternes navn:

Stian Fanebust, Kim Utheim, Per Torgrim Thorbjørnsen

Kurs/avdeling:

Dato: ________________________

Rett til innsyn, kopiering og publisering av hovedprosjekt

Biblioteket og avdelingene får ofte forespørsler om kopi eller innsyn i hovedprosjekt. Biblioteket ønsker å gjøre gode hovedprosjekt tilgjengelig ved å publisere dem i papirutgave og legge dem på internett. Høgskolen trenger studentenes tillatelse til dette.

Hovedprosjektet vil fortsatt være forfatterens åndsverk med de rettigheter det gir.

Høgskolens bruk vil ikke omfatte kommersiell bruk av studenters hovedprosjekt.

Tillater du/dere at din/deres hovedprosjekt blir publisert både i papir og nettutgave ?

___ ja ___ nei

Signatur av alle forfattere:

BUSKERUD AND VESTFOLD UNIVERSITY COLLEGE

Preface

The project team consists of three computer engineering students at Høgskolen i Buskerud og Vestfold (Buskerud and Vestfold University College). One team member, Stian Fanebust, works at ICD Software. He pitched the idea of this project to his co-worker/boss and they agreed on it. Stian then came to Kim and Torgrim since they had worked together on previous assignments this school year. Both responded positive to the idea and formed this team. While Stian was the only one with prior experience with Qt, no one on the team had ever written in QML.

This report describes how the team worked when developing a mobile application in an unfamiliar development framework:

Learning-

-about the development framework and its programming languages

-about the requirements and restrictions of mobile development

-how to communicating with a third party application

-how to efficiently display information on small screens

-to develop efficient and good-looking user interface

BUSKERUD AND VESTFOLD UNIVERSITY COLLEGE

Table of contents

Table of figures

1 Introduction

The idea of creating a browser application, dedicated to mobile devices, was presented to "ICD software Ltd" by the project team leader. The browser would serve as an added tool to ease the service of applications created with the company's software "CDP" *("Control Design Platform")*. This software is used to automate and control production processes as well as operations of heavy equipment *(e.g. cranes and draw-bridges)*, also known as "Industry Control Systems *(ICS)*" [18]. The company accepted this idea as a welcome addition to their software package. Thereby assigning the project team with a task to exploring how this could be done and the challenge of producing a prototype.

This task requires research in the field of mobile software engineering to get an idea of which techniques are available *(e.g. sandboxing, MVC)*, and how different platforms in todays marked *(e.g. Windows, iOS and Android)* adopt these. The research might help identify viable ways to develop mobile device software, keeping in mind current and future needs of the company. As the company has expressed a desire to investigate an extended use of their current integrated development environment *(IDE)* in regards to their current solutions. This is a major factor in the decision of what IDE to use for mobile device software development.

In the planning phase of the project it was made clear that the application programming interface *(API)* for a planned extension to the "CDP" software was not yet available for use. This API will provide functionality to access software built with "CDP" without having to run the entire "CDP engine" or through the current built inn web server. In the initial feasibility study, this new API was meant to be a big part of the project. As it was unavailable during the larger part of this project, no more investigation has been done in to the use of this new API. None the less, this expected new extension have a great impact on decisions and choices made during the course of this project as it's considered a priority to accommodate this.
As the current "CDP engine" version do not support any of the new mobile devices and all applications build with the "CDP" software have a built in web server. All this projects exploration in to communication will focus on the web server.

1.1 Background

With the increased use of mobile devices in all aspects of society such as to increase productivity of workers and control everyday objects *(e.g. easy access to documentation, car heating, locking your house)*. Customers either buying new software or ordering a project containing electronic control systems, might come to expect some capability of interacting with this through a mobile device. Therefore it might be considered important for "ICD Software Ltd", to investigate ways of incorporating mobile devices with their products.

1.2 Statement of problem

What considerations need to be taken into account when designing and creating a prototype browser on smart devices for "CDP-controllers"?

For the development of the prototype browser the project team has divided the problem into three main challenges:

- How to develop an application for mobile devices?
- How to design the browser for mobile devices?
- How to use the IDE to achieve the selected design?

1.3 Project goal

The goal of this project consists of three parts. First of is the need to determine a viable way for "ICD Software Ltd" to develop software for mobile devices. This is relevant information to help the company make informed decisions for future development. As well as give this project a starting platform in which to proceed with further analysis and development.

The second goal is to identify design possibilities such as transitional effect to make an informed decision on how to design this projects "CDP browser". As well as give the company information on what is widely used designs on mobile devices for future reference and consideration.

The third and final goal is to use an IDE selected from the first part to realize the design choices from the second part in a working "CDP browser" prototype. This will help to identify unclear or previously unknown factors in the development of mobile device software.

1.4 Process goal

One process goal is to learn about mobile software engineering such as; what different platforms are available? What is the major difference between them? What is needed to start developing on them? This will then give the project team basic knowledge about mobile device software engineering that, in turn, will help the project team make informed decisions in regards to mobile device software development.
The second goal is to get experience in learning a new IDE and thereby identifying different techniques on how to do this. This will then help the project team members more quickly adapt to new IDEs in the future.
A third goal is to prepare the project team members for future team work by experiencing how to collaborate on a larger task. All the while discovering practical application of theoretical knowledge acquired during education.

1.5 Scope

The project will include learning mobile software development concepts and how they are used on different platforms. It will also include identifying what platforms are available in the market, as well as pros and cons of the individual platforms, followed by selecting platforms considered most viable for the company and this region of the world (i.e. Europe). After which, one will be deemed most viable for "ICD Software Ltd" and used throughout this project.

The design process will keep its focus on intuitive use, with the goal of making the browser a good tool when working with "CDP controllers". Furthermore the issue of, how to present dynamic and possibly large amounts of date on mobile device touch screens, will require some attention. This is due to the variety of mobile devices with each its own dpi *(dots pr. inch)*, and the ability to show both "landscape" and "portrait" mode.
To conform to this projects time limit, there will be little focus on the purely esthetic look of the "CDP browser" prototype.

For the implementation stage, the main focus will be on how to realize the imagined design as well as establishing communication with a "CDP controller". Realizing the design will require learning a new IDE and how to implement this on the "CDP browser".
As the API of the new addition to the "CDP" software is not available, it will not be part of the scope of this project. However it will be kept in mind to facilitate an ease the implementation of this new API when it is ready. As such this project will use a web server, which is a part of every application developed with "CDP", for communication between a "CDP controller" and the "CDP browser" prototype. The communication will use the web servers built-in "RESTful"[19] API.

Security the project team considers a major issue for any future implementation of the "CDP browser", but will not be covered in this project. This is considered to be the next step in creating a working product from "ICD Software Ltd".

1.6 Outline

This report start by giving a short introduction to what a "CDP controller" is, explaining the basics that are relevant to this report. Followed by clarifying a few common mobile software engineering concepts, and a look in to platforms that incorporate these. This is then concluded with a selection of platform and IDE to be used in this report. After which the chosen IDE will be introduced with some of its relevant features.

The design chapter explains how the project team worked and reasoned to achieve a desired design through; an analysis of the existing "CDP controller" and exploring design techniques in existing mobile devices followed by, combining these in to a design.
The implementation chapter discusses the way the project team worked to learn a new IDE and language, as well as some aspects of the creation of the prototype.
In conclusion of this report there is a review of project in concern to project goals, followed by a summary.

2 Method

2.1 About

This chapter starts by presenting the project teams work method throughout the project, followed by presenting the different approaches during each phase. The project can be divided in to three main phases in regard to the work that was carried out.

1. Data collection.
2. Design.
3. Implementation.

2.2 Work method

The work in this project has been done following a modified, more informal, version of “SCRUM” [20]. There were weekly meetings to discuss what have been done, where to go from here and divide workload until next meeting. The work was delegated to each team member with a focus on what each member expressed a desired to investigate or do. This was done to keep the work interesting for each member and the idea that, if the work is interesting the result is better.
The informal role of “Scrum Master” was given to one team member, Stian Fanebust, because he had the general idea for project. And as such, initially the better overview of what needed to be done. Yet the progression of work throughout the project was considered more of a democracy, to allow a free flow of ideas in the interest of the learning process.

2.3 Data collection

The first phase consisted of gathering information about anything the project team considered relevant to complete the project. This gathering of information was divided in to three major areas, one for each team member:

- Explore mobile engineering concepts in general to get an overview.
- Identify what is available in the “Qt” IDE and how to start using it for mobile software development.
- Investigate how the “CDP controller” works, and what is relevant information to this projects goal.

Each area was intended to give the project team a better understanding of what the project would encompass. And help to form a basis from which to proceed to the next phase.

2.4 Design

The second phase was the GUI design of the “CDP browser” prototype. This phase was divided in to three parts to be carried out in order, each part enabling a better basis to make decisions on focus for the next part.

1. Analyze what and how data is available in current solutions from a “CDP controller”.
2. Explore which design techniques exist for mobile devices and possible applicable uses for these.
3. Combine data from the analysis with design techniques to make a design for the “CDP browser”.

2.5 Implementation

The third phase was the development process; this process consisted of learning a new IDE and using this IDE to implement the imagined design. Several sources were used to learn the new IDE several, some of which are:

- Running and analyzing example applications that follow the IDE installation.
- Referencing the IDEs documentation while creating small prototype applications.
- Watching tutorials the IDEs own channel on “YouTube.com”[5] to learn best practices and coding techniques.

As there were no clear guidelines or specifications on how the “CDP browser” would work or look, it was decided to use an approach that would clarify these aspects. The choice for the development process was a combination between “extreme programming *(XP)”* [21] and “rapid prototyping”. “XP prototyping” in contrast to ordinary rapid prototyping approach have an even more unfinished result containing only the most important aspects. It should also be shown to the users or customer under development. The XP approach gave more freedom in working individually and made it possible to always change the ideas and design after meetings and discussions within the group.

3 CDP (Control Design Platform)

3.1 About

This chapter is meant to give an understanding what the "CDP controller" is and how it works, by explaining key features of its internal objects and infrastructure. This understanding is vital to understanding decisions made in this project.

Information in this chapter is derived from the "CDP System Manual" [23] located on the project CD or downloadable with the "CDP" software at "ICD Software Ltd" home page[22]. The final chapter is a compilation of communications with senior software developer Nils Petter Eftedal at "ICD Software Ltd".

3.2 CDP

CDP or "Control Design Platform" is software developed by "ICD Software Ltd", a company in the "ICD Industries" group. CDP software is a complete development framework with a main goal of simplifying development and implementation of control systems. This is achieved partly by providing one platform used to develop controllers for multiple "operating systems" (OS) *(i.e. Windows, Linux or On Time RTOS-32)*. These controllers also support multiple existing protocols *(e.g. CAN bus, Modbus, Ethernet)* for communication between controllers and external equipment. Communication between controllers is handled entirely by the CDP framework with its customized "CDP Messenger" protocol. That is used either by internal services on one computer or over Ethernet between computers. The logical structure of a typical network of CDP-controllers is illustrated in **Feil! Fant ikke referansekilden.** below.

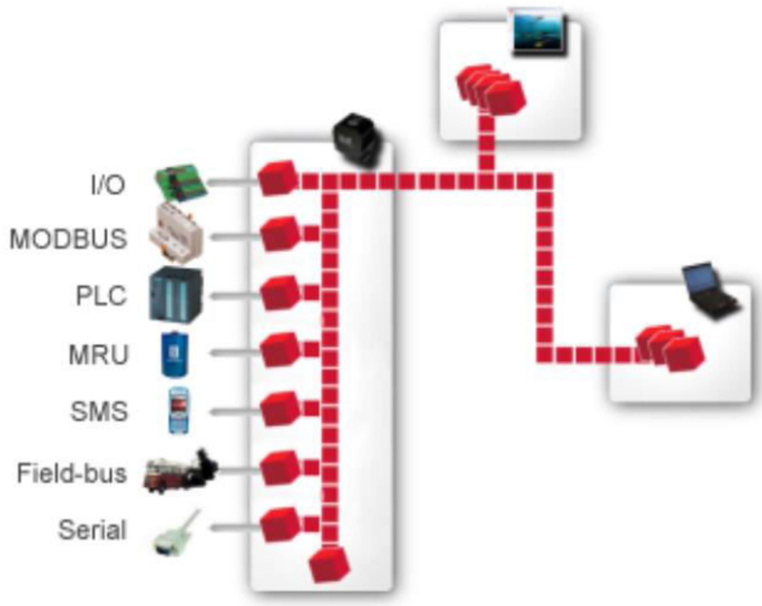


Figure 1 : Different physical locations interfacing to external devices. ("ICD system manual")

3.3 The CDP controller

The “CDP controller” is a computer running software developed with CDP to handle signals either subscribed to from another controller, or received from any external devices *(e.g. PLC, any I/O unit)* that can communicate with the “CDP controller” through supported protocols. These signals then trigger internal logic to either change the value of an I/O, or set an internal signal/parameter on the “CDP controller” itself. These internal signals/parameters can then be subscribed to by other “CDP controllers” through the “MessengerIOServer” to create a network of “CDP controllers”.

The controller consists of 5 main parts or “components”:

1. CDPEngine. The core program, handling logic and functionality.
2. Messenger. The CDP Messenger communication service provider.
3. MessengerIOServer. Signal and property transmission on CDP Messenger protocol.
4. WebServer. Web interface to all CDP components.
5. User “components”. User components that contain user developed objects.

All CDP developed applications consist of “components” that are arranged hierarchically, and each “component” has some basic «CDP members” available to them (e.g. Signals, CDPSignals, Properties, Alarms, Parameters, Messages and subcomponents). These objects are all identified by a unique name that is used to access them.
(e.g. AHCRemotePanel.AHCPanel.Setup.DrumSetupPage.DrumHelp)

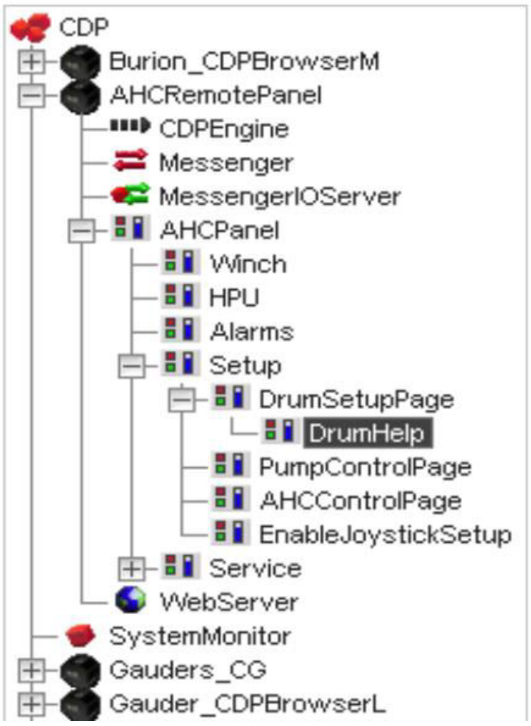


Figure 2 : Hierarchical display of a controller in a “CDP Browser”. (“ICD system manual”)

3.3.1 CDPEngine

This is the core of any “CDP controller” since it handles all the logic and functionality provided by CDP. It operates in loops for checking and updating objects with frequencies specified by the user up to and including 1 kHz.

3.3.2 Messenger and MessengerIOServer

These are the service providers and the servers that handle all internal and external communication between “CDP controllers”. They are developed by “ICD Software Ltd” to handle the controllers’ communication.
The communication is message based, thus every component connects to the “Messenger” service. Through this service they can access other components and their members. What a message should do once it reaches its designated component is specified on the receiving component itself.

3.3.3 WebServer

Each “CDP controller” has a web server that provides an interface for all its “components”. The interface can be connected to by supported web browsers and used to manipulate “components” and «CDP members”. This web server uses “JavaScript” to extract and present xml scheme in a standard web services, this limits the frequency at which the values can be updated and displayed to a maximum of about 1Hz. As the “CDP controller” itself can operate and update values between other controllers at a rate of 1000Hz if it uses the CDP Messenger protocol, this is an issue when third-party applications want access to the “CDP controller” which at the moment only can be done through the web interface.

3.3.4 GUI

To provide the “CDP controller” with an option for “Human Machine Interface (HMI)”, the controller can have its own “Graphic User Interface (GUI)”. This is provided by an “Application Programming Interface (API)” as a plugin *(“CDP2Qt”)* to the “Qt” IDE. This API gives a GUI created in “QtDesigner” access to the controllers “MessengerIOServer”, where it can receive and update information at the controllers inherit frequency *(up to 1 kHz)*. However this method for HMI requires a running controller *(CDP Engine)* on the device where the GUI is to be run.

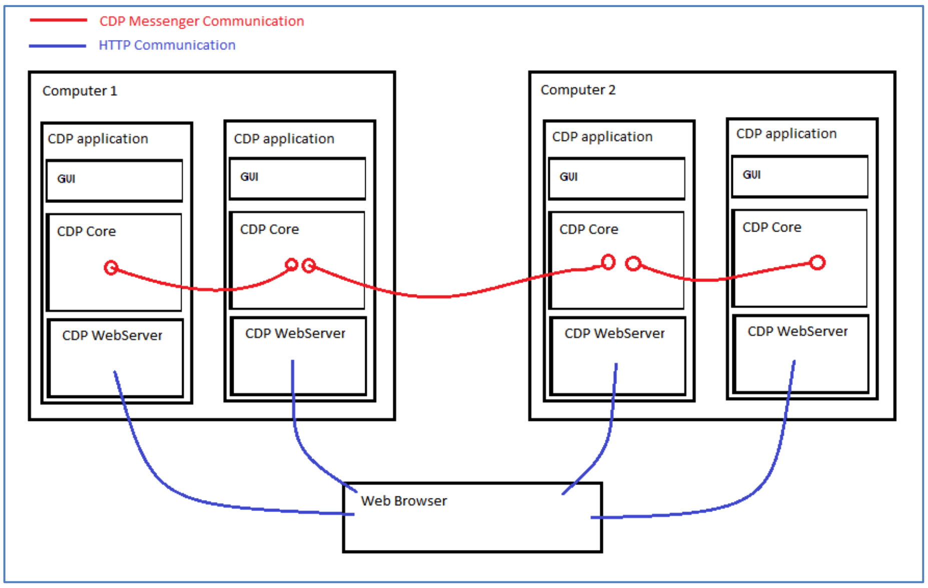


Figure 3 : CDP Controller basic communications paths.

3.4 The CDP horizon

Enhancing the “CDP” software is a constant ongoing process, but at the moment “ICD Software Ltd” main focus is on a major new release of “CDP” called “CDP Studio”, expected in the summer of 2014. Most of the specifics concerning this release are yet to be announced, but the project team has been given access into plans relevant to this project.

This release will contain a major remake of the “CDP” interface for developing controllers. It will also add a new API for C++ to access controllers without having to run the entire “CDP Engine”. This new API will open up a whole new range of possibilities for communicating with and accessing controllers, and in the long run render the existing “Web server” obsolete. But in the transitional phase new controllers might contain both, and there are plans to build a plugin module for older controllers to access this new API.

Software developed using this API will have two major advantages;
First of all not having to run the “CDP Engine” to gain access to a “CDP controllers” objects, give a freer range of platforms on where to run software accessing “CDP controllers”. Secondly without having to rely on the services provided by the “Web Server” any third-party software can access values on the controller using a much higher frequency.

Today GUI programming is done in a part of “Qt” called the “QtDesigner”. This part of the IDE produces a form file *(i.e. *.ui)* which is basically an xml file. The “CDP2Qt” plugin provides ready widgets that can be used in these forms for easy access to controller objects through “routing” identified by unique object names.

As the new “CDP Studio” release is closely interconnected with “Qt”,”ICD Software Ltd” has a desire to take advantage of another module within “Qt” called “QtQuick”. This is the standard library of the “QML engine” which is an engine to run visual application *(QML applications).* “QtQuick” provides a “QML” API with types to create user interfaces, and “C++” API to extend QML applications with “C++” code. At present this module is accessible from “QtDesigner” through built-in “Qt” functionality, but this means that an application has to run a second “QML engine”. This might be acceptable for powerful systems, but is less desirable on smaller systems. That is one reason why ”ICD Software Ltd” want to explore the possibility of creating pure QML applications for connection to their controllers using the new API.

4 Mobile software engineering concepts

4.1 About

Even though different companies use different methods in developments of applications for mobile devices, there are some common concepts that most implementations share. As well as some limitations these devices present, such as less memory, slower CPU and limited battery life [13]. The different mobile devices also differ much in screen size and resolution/pixel density and IO possibilities. This introduces new considerations when designing and developing software for such devices.
This chapter will shed some light on the some of these concepts.

4.2 Security

Security is always important in all kinds of software engineering. But it could be argued that mobile devices require even more consideration regarding security. As they contribute to an increased security risk by the intimate and personal nature of mobile devices (especially phones). They are used to store personal information that if it were to fall in to the wrong hands, would make it easy to hijack the identity of the owner. And thereby bypass company security measures to either steal sensitive information or perform sabotage. Another consideration is the wide distribution of applications developed by everyone from amateurs to professional software engineers. This increases the chance of installing malicious or poorly developed software that might compromise the integrity of any network.

This makes companies strive to make systems that makes the application secure by default not depending on the developers skill. One such system is app-stores like Apple App Store, Google Play and Windows Store. These have automatic and manual testing, certification and developer subscriptions. This helps control the applications and make sure it is properly tested and do not contain malicious software. It also takes care of the financial and copyright aspect of deployment. These app-stores also make sure that the application is installed and uninstalled properly without harming the phone or device in anyway. Another part of the system is integrated security measures in the development environment, as well as the operative system.

Another common concept for security is to restrict software’s authority over the mobile device. One technique implementing this concept is “sandboxing” [1]. It restricts one applications processes from interfere with the processes of any other application, or the OS itself, without explicit permission by the OS and/or the user. E.g. A application wants access to the contact list on the phone. It will not gain access without prompting the user to give it permission to access this part of the mobile device.

4.3 Native Language

Native language indicates the use of “low level” language like C/C++ compiled to machine code for the specific architecture. It is uncommon to use native language when programming for mobile devices. The standard practice is to use scripting language or high level language which can access extensions or components from native code. Using native code introduce more complexity [2] which makes it harder to maintain correctness, maintainability and scalability in the system. Another issue with the use of native language is that it has to be compiled for specific hardware which makes it less portable.

However unconventional use of IO and the need to perform extensive computations like graphics processing might require a higher level of precision from the programmer. In these cases it is necessary to use native language for a more direct access to resources on the device such as memory management. A common misconception is that programming on the native language will always give the highest performance. Because of automatic optimizations in the development environment, and the IDE is often specified towards not using native language. There is no guarantee that using native language will speed up the program. Using native code gives the developer more control at the cost of an increased complexity in the program.

4.4 Managed Language

Another common practice is to use a runtime API, which is a library of low-level or native routines precompiled in a binary and accessible at runtime. These routines can then be called at runtime from a higher level language or script. This helps the developer lessen the complexity of the code, as well as freeing the developer from the responsibility of dealing with low level features of the device and OS. One memory managing technique is "garbage collection" [4], an algorithm for freeing and controlling memory. This is done by a technique called reference counting which counts all the variables pointing to an allocation of memory. If it reaches zero it gets put into a list of objects/memory that can be automatically freed at some optimized time. This concept frees the developer from having to include this in his code. Languages that depend on runtime libraries are often called managed languages such as C# or Java. [3]

4.5 Localization

The concept of localization represents a common technique used to ease the translation of an application between spoken languages, as well as make it possible to change the language on the application at runtime, without any additional programming required.
This technique stores every string representation of text in the entire application in one single file. Everywhere the application shows text there is only a reference to a part of the file containing the string. As such, to change the spoken language of an application, one simply changes the file. And as all the strings are kept in a file containing only them, translation is simpler than having to traverse the entire code looking for them.
This technique is implemented in xCode, iOS and Qt among others.

4.6 Emulators

There are many manufacturers of mobile devices in today's market *(e.g. Samsung, Apple, Microsoft)*, each with a large selection of mobile devices available. All with a large selection of display sizes, resolutions and functionalities as well as ever evolving software.
Testing compatibility of an application on all existing physical devices would require a waste amount of resources and time. To ease this process most SDK's provide emulators that provide a simulated environment in which to test the application. These emulators can present the environment of any existing mobile devices. And if the software is updated or a new device is presented on the market, one can simply download a new environment to match this device. [5]

4.7 Design Patterns, MVC, declarative and reactive languages

Common software architecture used to implement user interfaces in mobile engineering is the "model view controller *(MVC)*". This architecture is designed to separate what the user see on a GUI *(view)* from the underlying data *(model)* and controller *(underlying logic of the application)*. And in doing so, facilitate using one "model" of data on several "views". Most IDEs for mobile development facilitate designing the application in this way.

The user interface, or GUI, is the appearance of the application including everything from the background graphic to interaction options such as buttons and text boxes. These graphic elements need a way to communicate with the model and controller to give a user control of the application. This is often done with data-binding; the data-binding method connects graphic elements to data/objects in the model/controller. The connection is a two way method to let the interface reflect changes in the model/controller and changes made by the user update the model/controller. How this is implemented varies from different SDK's but it is a common concept in mobile development.

Data-binding is often used in programming languages that follows the "declarative" programming paradigm. These languages are designed to let the programmer declare the wanted result and not how to achieve these. As such, they are well suited for interface or GUI programming in regards to easing the process of quickly creating an interface.

This could also be done with languages following the observer pattern like Java. The observer pattern is the concept of making objects keep a list of observers and calling them if any state changes, this fits well with the MVC pattern and data-binding. [13]

Another programming paradigm is "reactive" programming, this is a more general case of the data-binding or observer pattern were the language itself is based around continuously updating data flow. Often used in hardware description languages since it describes how circuit flows, but it is also used in languages for user interface such as QML. [6]

5 Target platforms

5.1 About

Manufacturers of mobile devices incorporate several different platforms in their products. This chapter will take a look at a selection of different platforms to identify important aspects of these. One of which is their approach to software development. The research gathered is used to get an overview of how the different platforms approach mobile software development. This information is useful to get an understanding of what to expect of an IDE; what languages it use and which design patterns that the IDE use and recommends? This is done to help in the process of choosing a viable IDE for "ICD Software Ltd" to expand their mobile device development.
The conclusion will present the selected platform and IDE for this project.

5.2 General

The decision on which platforms to research was made on the basis of their market share. The most used platforms were; Windows, iOS and Android. These three together ads up to 97.2% of the market share.

Worldwide Smartphone Sales to End Users by Operating System in 2013 (Thousands of Units)

Operating System	2013 Units	2013 Market Share (%)	2012 Units	2012 Market Share (%)
Android	758,719.9	78.4	451,621.0	66.4
iOS	150,785.9	15.6	130,133.2	19.1
Microsoft	30,842.9	3.2	16,940.7	2.5
BlackBerry	18,605.9	1.9	34,210.3	5.0
Other OS	8,821.2	0.9	47,203.0	6.9
Total	**967,775.8**	**100.0**	**680,108.2**	**100.0**

Figure 4: Operating system on SmartPhones by market share.

Source: Gartner (February 2014) (http://www.gartner.com/newsroom/id/2665715)

5.3 Windows

Windows got the smallest market share of the selected mobile platforms. Windows applications are commonly developed with the use of C#, Visual basic, C++ for components extensions and XAML for user interface. For Windows OS development the most common IDE to use is "Visual Studio", but any IDE that support the used language can be used. There are several choices of languages when developing windows application:

- C# and Extensible Application Markup Language (XAML)
- JavaScript and HTML5
- Microsoft Visual Basic and XAML
- Visual C++ component extensions (C++/CX) and XAML
- C++/CX and Microsoft DirectX

C# is not compiled to machine code but uses the .NET runtime library. Any native development is done with C++, such as; to create components and interact with DirectX. DirectX is often used in game development that needs to take advantage of the graphics accelerator. C# is usually accommodated by XAML or JavaScript/html for programming and declaring the user interface.

When using Visual basic and C#, the application is not compiled to machine code. Instead it takes advantage of the runtime library. This is done to take advantage of this methods increased security, and to get access to higher level functionality like garbage collection. It is also possible to use the WinRT (windows runtime) for developing native with C++ and CX (component extension).

XAML(extendable application markup language) is a markup language made by "Microsoft", it is used for the GUI design of windows applications. It's basically a XML file and its function is to separate the GUI from the logic of the application *(controller).* XAML can control the GUI, data-binding and can also be compiled into BAML files.

The application is packaged in XAP files for distribution and easy installation on mobile devices. The XAP file which is ZIP compressed can be uploaded to the windows phone store, where it will be certified, licensed and checked [7]. There are possibilities to manually install application on windows phones, but "Windows Phone Store" is supposed to be the only legitimate source of app acquisition [5]

5.4 iOS

The second largest market share is claimed by iOS. Programming for this platform is done in "Objective C". "Objective C" is also a "low level" language which makes the applications high on performance and very efficient if well programmed, this is especially beneficial to devices with low memory and slow processor. The usual and recommended way of developing for IOS, is to use the IDE xCode and the language objective-C. xCode is not available for Windows or any other OS than OSx for mac. Since the OSx only works on macs it is also needed to buy mac hardware. For deployment of the application on the Mac app store its required to sign up with developer account and pay annual fee. It's not even possible to test your app without this subscription. It is also possible to make application with the QT IDE for iOS but it still requires xCode, subscriptions and a mac. The QT application gets ported over to xCode for compilation and deployment. One easy way for developing and deploying apps on the marked is to develop for iOS. Even though the setup is easy and fast, it comes with a prize of special hardware, software and even subscription for deploying the finished app. The developer also faces certain restriction on what is allowed and what is not allowed dictated by Apple Company. [8]

One of the advantages of developing for iOS is that it's easy to setup and gives access for deploying on a huge marked in the app store. Were payments and other administrative tasks is use friendly for the developer. The stern restriction of the apple products also helps making the apps secure and bug-free even if non-professionals make them. The restrictions and regulation for the products made by Apple could also prove to be an advantage for example by using software and hardware and everything made by the same company that produces the device. This makes the developing always up to date and compatible for all the features that the device might have. No mysterious errors or compatibility issues for the IDE setup since everyone is using the same hardware and software.

5.4.1 Memory management in xCode

xCode is an IDE it contains the SDK for apple products such as OSX and IOS its downloaded from Mac app store its free to download but only on the newest OSX (which today is Mac OSX 10.8 Mountain Lion) and an Intel-based Mac. xCode works similar to many other IDE's that have been explored, but it has a storyboard feature which makes it simple to make GUI. It works by defining pages/windows and the transition between them, as the user is tapping on buttons. GUI design is stored in nib files (actually xml). Building the UI is simple with Apples tools by hooking up UI object in the nib files to the declaration in code. [9] xCode uses LLVM compiler with Clang front end so that it works with Objective-C. Objective-C is a programming language influenced by C but is an object orientated language. It is the main programming language for apple products and is therefore also used with iOS apps. The iOS SDK contains all the tools needed for Development like the iOS API Coccoa touch.

5.4.2 Objective-c and low level programming

Using such a low level language makes the apps close to the hardware and will have less overhead and could give more efficiency compared to say java. A language such as Objective-C for apps could introduce more complexity than needed and could make dangerous bugs which would leave the user's phone exploitable for attacks. Objective-C gives access to memory management which could make buffer overflow attack possible and other bugs. But like other companies, Apple uses techniques to sustain the safety and security of the applications, like automatic analyzing code and deployment restrictions.

One of those security measures is part of the xCode IDE and the Clang front end that analyze the code for errors and bugs [10]. The Clang automatic analysis is called Clang static analyzer, it is similar to compile errors, but it also detects bugs which could happen at runtime. It is much more advanced and analyzes the semantic of the code and search deep for any bugs that is usually found in debugging and testing of the program. There are also restrictions in how the application is distributed [8]. It can only be distributed through the Mac app store which is controlled by Apple Company and gives them the opportunity to check all application for dangerous faults and security threats. In contrast to Android application which could be distributed everywhere

for memory management the iOS uses ARC (automatic reference counter) which is a compiled variant of garbage collection[11], its role is the same and it is supposed to automate the memory management by freeing memory that does not have any referee, It doesn't do the reference counting and de-allocation at runtime, but instead do it when the application is compiled. It is supposed to be more efficient on mobile devices since it gives less overhead by the garbage collector and it is more deterministic and will not cause any unexpected runtime errors. Objective C is also a runtime orientated language [12] which means that it have the possibility to execute code at runtime which could give an abstraction layer with higher level functionality. This gives the developer the possibility to use lots of the memory management techniques used on the other developments platforms.

5.5 Android

Android is an OS used on mobile devices. Its owned by Google and its open source. The official way for development on the Android platform is to use Java. Part of the Android OS is a special Java virtual machine (VM) called Dalvik [14], designed for mobile devices. The VM gives a layer of abstraction on top of the native programming environment. This helps the application stay secure and prevents the developers from doing lower level memory management. It also makes it easier for portability and compatibility for the many different devices and functionally it may have E.g. localization and special I/O functionalities. Oppose to IOS development there is no requirements of any subscription for deploying applications on phone or any devices

5.5.1 Eclipse ADT bundle for Android

For development and deployment it needs Eclipse ADT bundle downloaded from the official android website [15]. It comes with SDK with all the tools needed for making applications. Eclipse is open source and free IDE that are usually forked in different direction, specified for certain tasks. The Eclipse ADT bundle is such a forked version specified for android development. It have ADT (Android Developer Tools) built in. Eclipse ADT bundle uses java compiled for the Dalvik virtual machine that is part of the Android OS. It can also be used together with html, html 5, CSS, and javascript. ADT Bundle from the android.com website provides everything that is needed for development such as emulator and SDK tools and does not need any further setup

Making apps with Java and Eclipse is often the recommended way when develop apps for Android. Other option is to use native languages such as C/C++ (example with Qt IDE) and then make programs that are running directly on the devices processor oppose to Java code that is processed on Dalvik. Applications is developed from everyone from amateurs to professionals and can be easily downloaded from some app-store or equivalent. It raises the problem of dangerously bugs and general quality of the software which in professional development would be carefully inspected and tested. One could assume the trend of "apps" is towards quantity, based on how app-stores and many IDE's for mobile development are designed (made user-friendly for ordinary people without previous experience). Therefore the use of virtual machine and java gives appropriate protection, security and sandbox properties that are needed. Other problems that Dalvik solves are the compatibility of different devices. They differ in many ways from pixel density to screen size even different CPU architecture such as ARM or x86. They also have lots of extra functionality like GPS, gyro, touch, landscape mode, camera, etc. Dalvik and even the Eclipse IDE give lots of help in making the applications compatible with differences in hardware. One example of this is how the project folders are structured. In the "res" folder there is subfolders named drawable-hdpi, drawable-ldpi, drawable-mdpi ect. These folders should contain pictures with high resolution to low resolution. So that when the app is deployed on a device with high resolution screen it will get the high res picture. This is a simple way in making compatible apps without any extra programming with the help of eclipse IDE.

The java programming uses lots of libraries made specifically for android and it is somewhat different than programming with "ordinary" java for desktop. For example it does not have any main function, but have onCreate, onResume, onPause, onDestroy instead. The application is also compiled for Dalvik VM not machine code. It goes from .java (source) and uses javac (compiler) to .class (byte code) and dx (converter) to get .dex files (dalvik executable) then zip , aapt to .apk file (android package). All this compiling, conversion and compression is done by Eclipse with one button.

java → class → dex → apk

It also uses more files than ordinary java development. Like XML for UI and manifest file. Eclipse gives the opportunity to use xml file for GUI design just like windows uses XAML. Xml manifest [17] is used for permissions and overall control over the application. If the application needs specific permission for example: controlling the address book or contact info of a phone or sending sms. It needs to be specified in the xml manifest and the user who download the program have to agree that the program may access such information and control it. It also must contain information about all the components used in the project (the activities, services, broadcast receivers, and content providers).

5.6 Pure Web Applications

Perhaps the simplest way of making and distributing applications is by doing a web application. It could basically be a website which uses CSS, html and javascript etc [16] to be designed specifically for a mobile device and look like an ordinary native app. It is also possible on most mobile devices to bookmark this website and make an icon for it just like ordinary app, even removing the browser bars. Other way is to make some code open a webview object and fetch the html, html5, CSS, javascript from a folder downloaded with the app example the asset project folder in android development with eclipse IDE. By including the files will make the app work even without Internet connection which would be needed if the app had to visit an URL. This approach has of course some downsides.

One of the downsides is that it gives the developer less control over the device IO and low level functionalities, it have more overhead by needing some browser engine to run it and is less efficient on using the mobile resources. It is harder to sell since it is not part of any app store. Does not have the same feel as native apps, won't get access the native UI.

Positive sides is that it do not need any administrative choirs for deployments or inclusion in app store, it is very accessible since all it needs is that the device have browser engine, also easy and fast to develop. In this project the webserver is already responding with HTML, CSS and javascript and one solution could be to edit the CSS for different mobile devices, this would also make it necessary to edit the source code of the webserver for detection of the mobile platform and to store different CSS files. This way of making an application would not be depended of any OS or platform

5.7 Conclusion

Many of the platforms and IDE shares common overall techniques and concept in developing for mobile devices, but they have some differences in the details of implementations. Some like Apple gives at default more power to the developer but restrict its submissions to the app store by strict rules and regulations. Others use a layers of abstraction on top of the platform and wants the applications to be to be run by runtime libraries or using virtual machines like Dalvik. Still custom runtime function could be made or programmed in native language, without the layer of abstraction if wanted. Both Windows and Apple have dedicated app-stores which is controlled and makes sure the application are safe and the transactions/administrative aspect of licensing and payments are in order. Android owned by Google is open source and could get applications from everywhere. Even though it also have an app-store controlled by Google

Even though the different options are taken into consideration, the ICD Company already uses Qt IDE and C++ for their main development; this is taken into account when deciding the IDE and platform. It would be possible to choose another platform if the research would show that it would be a better choice. It is still important part of project goal to do research and gather information about the different platforms and companies. Example: Windows phone, iOS, Android and their respective commonly used IDE: Visual Basic, xCode, Eclipse, Qt. The pure web based approach is maybe the simplest solution for the project, since the application is going to fetch data from webserver anyway. Because of payments subscriptions and other restrictions of windows and iOS, the choice of platform and IDE fell on Qt and android. Android had the most marked share and gave more freedom, and Qt was the most suitable IDE of reason stated above.

ICD software is using the Qt IDE for their main software and because of this the IDE for the project should be Qt for Android development. Android is also the OS that have the highest marked share and is opens source which gives more freedom in deployments of the application. No fees, no subscription and no involvement from external company. Java and web based solution would also be a good alternative for android and it is the default way for android development. But the company already uses Qt and could use our project as references for further development for mobile devices in the future. That is maybe the most important reason for choosing Qt for the development environment together with Android.

	Windows phone	IOS	Android	
			Java, Eclipse	Qt, C++ and qml
Security and memory management	- .NET framework - winRT	- ARC - objective-c runtime - clang static analyzer	- Dalvik VM	-Qml runtime
Deployment restrictions	- Windows phone store	- apple app-store -Subscription fee - strict deployment policy	- Google play - everywhere	
Market share	3.2%	15.6%	78.4%	
other				- same IDE as the ICD company use

Figure 5 : Concept implementation pr. OS.

6 Qt

6.1 About

This chapter will give a basic overview of the selected IDE for this project. This will be done to clarify possibilities this IDE give the programmer, as well as limitations.

6.2 Framework

Qt is a multi-platform development framework designed for developing applications with a graphical user interface.

The original developers of Qt, Eirik Chambe-ENg and Haavard Nord, began development of Qt in 1991. They founded the company Trolltech three years after development had begun. The first editions of Qt had two editions, Qt/Windows for Windows and Qt/X11 for Unix, but with the release of Qt 3.0 in 2001, Qt added support for Mac OS X. In 2008, Nokia acquired TrollTech ASA and made it Nokia's main development platform for its devices. Nokia eventually sold Qt to Digia in 2011, Digia now owns all remaining Qt business. [26]

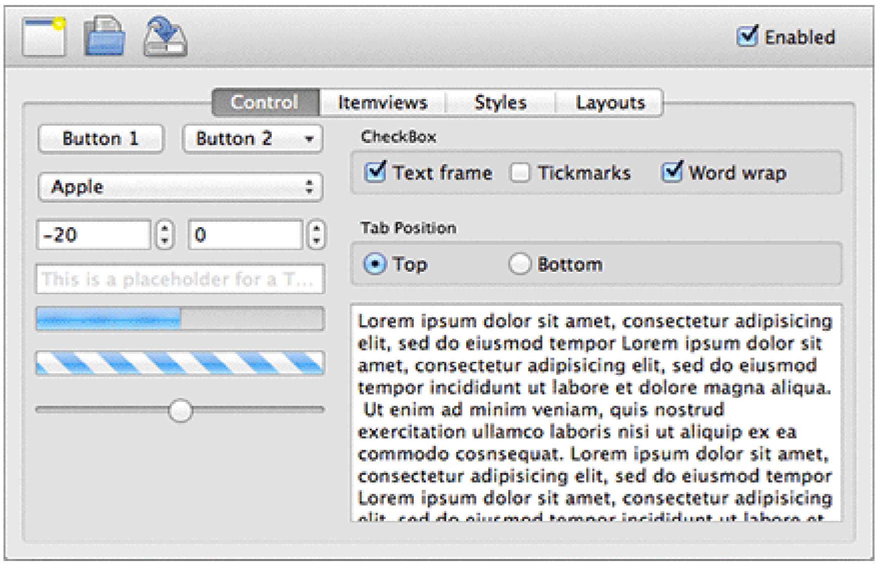
The main programming language in Qt is C++, but it can support other programming languages with language bindings. These bindings are community built and not supported officially. In 2011, Qt introduced Qt Quick, and with it QML. QML is a JavaScript based object oriented, declarative programming language used for GUI development.
Qt runs on all major desktop platforms (Windows, Linux/Unix and OS X) and it can compile applications to them along with mobile platforms; such as Android, iOS and BlackBerry and Windows RT.

6.3 Qt Quick

Qt Quick is a Qt module to help developers create intuitive, modern, fluid user interfaces used in mobile devices. It provides all the necessary types for creating user interfaces with QML. QtQuick uses the markup language QML to describe user interfaces.

Qt Quick Controls

Qt Quick Controls provides Qt Quick with a set of controls used to build complete interfaces. With Qt Quick Controls developers can easily build menus, input components and layouts with a native look and feel among multiple platforms.



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo cosnsequat. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et

Figure 6: Gallery example from QtQuick Controlls. [36]

Qt Widgets

Qt Widgets provides Qt applications with a set of classic desktop-style user interfaces. While Qt Quick aims at mobile devices, Qt Widgets mainly aims at desktop environments like Windows, Linux and Mac OS X. Another contrast to Qt Quick is that Qt Widgets do not have its own UI language, but uses C++.

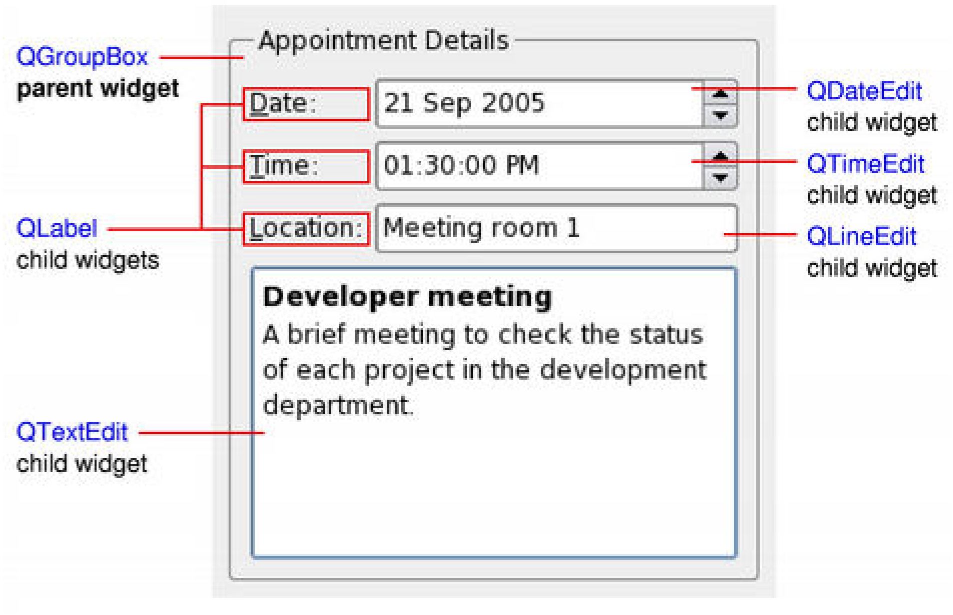


Figure 7 : Example showing elements in Qt Widgets [37]

Web content

Displays web content with the WebKit layout engine. The user interface is written with standard web technologies such as in HTML, CSS and JavaScript.

There is two different webkit APIs in Qt: Qt WebKit Widgets, a C++ API used to render and interact with web content and Qt WebKit, a module that displays web content in a QML view called WebView.[27]

6.4 Qt Creator

Qt Creator is a cross platform IDE part of the Qt application development framework. [28]

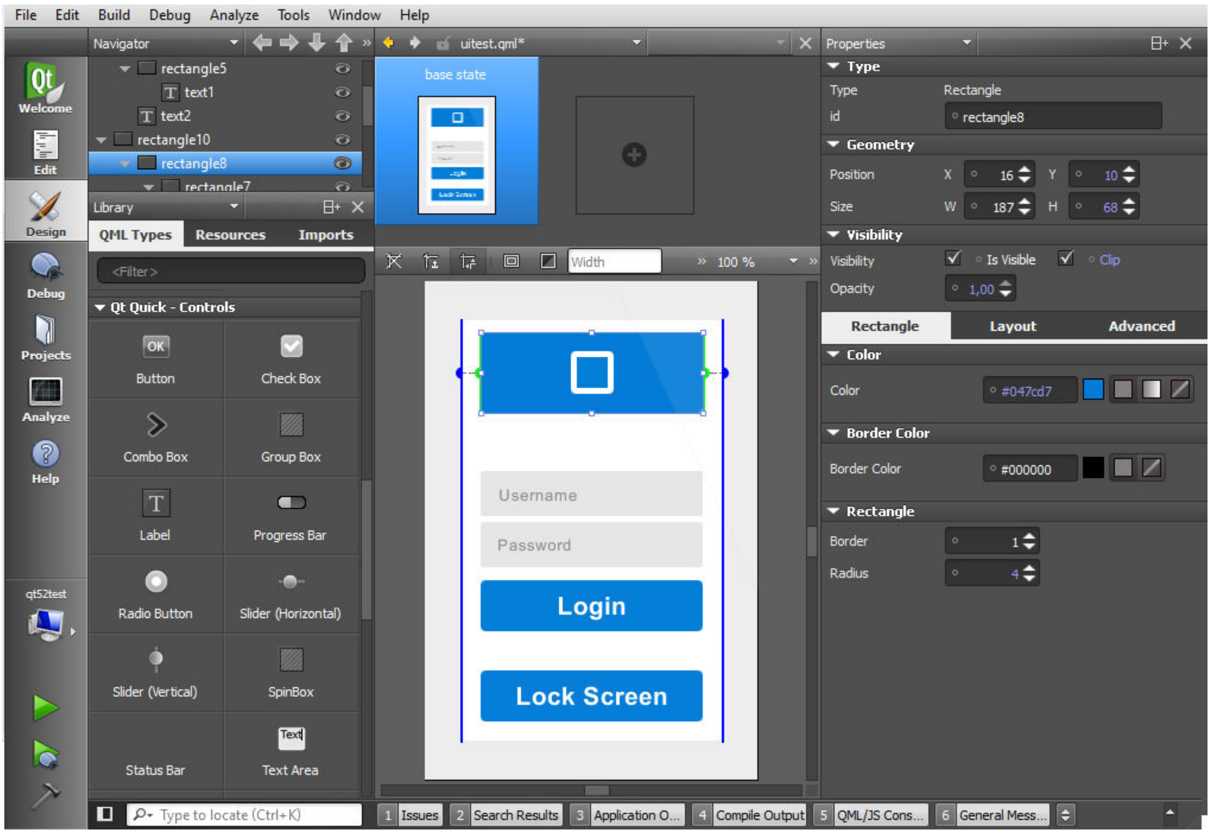
6.4.1 Edit

Qt Creator contains a code editor with support of several languages and syntax highlighting. As editing source code are a core element in application development, the editor is Qt Creators main component.

6.4.2 Design

The Design mode provides the user with two visual editors, Qt Quick Designer and Qt Designer. Qt Designer uses a "what you see is what you get" workspace, where a user can drag and drop UI components, and position and resize them.

It is easy to switch between the editor and designer. QML written in the editor can be seen in the Qt Quick Designer and source code of elements created in the designer can be seen in the editor.



Type: Rectangle
id: rectangle8
Geometry
Position X 16 Y 10
Size W 187 H 68
Visibility
Visibility: ✓ Is Visible ✓ Clip
Opacity: 1,00
Color
Color: #047cd7
Border Color
Border Color: #000000
Rectangle
Border: 1
Radius: 4

Figure 8 : Quick Designer screen shot.

6.5 QML

QML is a highly readable, declarative, user interface specific programming language designed for mobile devices. Being declarative means the programmer tell the computer what result they want, but not how the computer should accomplish it. QML is usually complimenting C++, where GUI is written in QML and the underlying application written in C++, but QML can also be executed alone with QML Runtime. QML runtime allows applications to consist solely of QML. [29]

6.5.1 Syntax

A simple “Hello World”-application would look something like this:

```
import QtQuick 2.2
import QtQuick.Controls 1.1

ApplicationWindow {
    title: qsTr("Hello World")
    width: 640
    height: 480

    Text {
        text: qsTr("Hello World")
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.verticalCenter: parent.verticalCenter
    }
}
```

Figure 9 : "Hello world" source code

There are two objects in this “Hello World” application, the main application window (ApplicationWindow), a Qt Quick Controls Windows object, and a Text object as its child. Their type, followed by a pair of braces, specifies each object. Inside the braces the programmer can specify property values and child objects. In this example the properties are title, width and height; and the child object is the Text object. Properties can be written line-by-line and all or some on a single line.

```
ApplicationWindow {
    title: qsTr("Hello World")
    width: 640
    height: 480
}
```

Figure 10 : One per line

```
ApplicationWindow {
    title: qsTr("Hello World")
    width: 640; height: 480
}
```

Figure 11 : Multiple per line

```
ApplicationWindow {
    title: qsTr("Hello World"); width: 640; height: 480
}
```

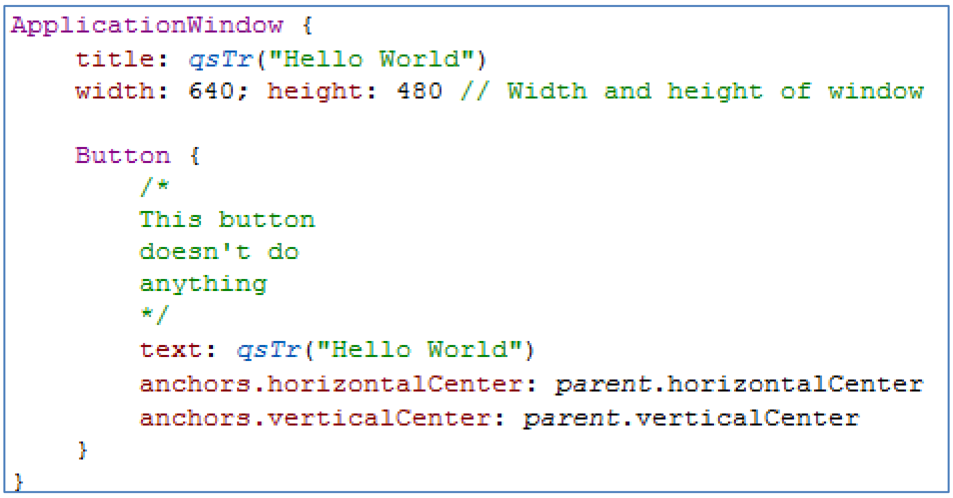
Figure 12 : All on single line

A semicolon must separate properties when written on a single line.

While objects can have many child objects, there can only be one root object.

6.5.2 Comments

Comments are optional text written directly in the source code. They are used to explain parts of code to the author or others reading the code. When compiled or executed, the comments are ignores. Comments on a single line starts with // and comments on multiple lines are written inside /* and */



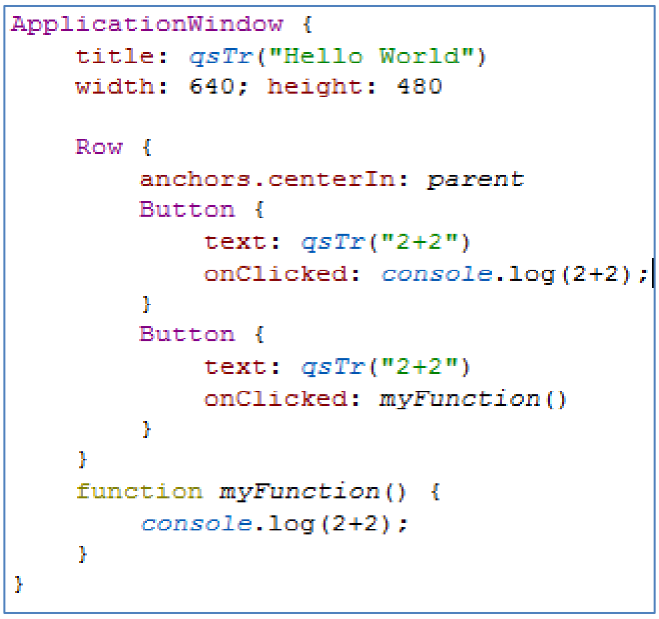
```
ApplicationWindow {
    title: qsTr("Hello World")
    width: 640; height: 480 // Width and height of window

    Button {
        /*
        This button
        doesn't do
        anything
        */
        text: qsTr("Hello World")
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.verticalCenter: parent.verticalCenter
    }
}
```

Figure 13 : Commenting examples

6.5.3 JavaScript

Qt QML provides the ability for running JavaScript expressions in QML and C++.

Signal Handlers in QML allows JavaScript to be executed when a signal is raised.
As an example, the Button has an onClicked signal handler that will run code when the button is pressed.



```
ApplicationWindow {
    title: qsTr("Hello World")
    width: 640; height: 480

    Row {
        anchors.centerIn: parent
        Button {
            text: qsTr("2+2")
            onClicked: console.log(2+2);
        }
        Button {
            text: qsTr("2+2")
            onClicked: myFunction()
        }
    }
    function myFunction() {
        console.log(2+2);
    }
}
```

Figure 14 : Signal handlers example.
Both these buttons will echo “4” to the application console output.

7 GUI Design

7.1 About

This chapter will describe the different steps of this projects GUI design process and give a detail account of reasoning behind the decisions made. The first part will contain initial thoughts on what is needed to produce a viable design for the prototype. Followed by an explanation of how the work to realize these thoughts was carried out.

The second part begins with analyzing the existing design for the "CDP controller" and the exploration of different design options used in existing applications. Followed by how this information is used in this project to create a design for a prototype.

In conclusion of the chapter there are mentioned considerations that a derivative of this project might benefit from taking in to account when designing the GUI.

7.2 Initial thoughts

The first thing needed when creating a viable design for this projects prototype is to find out how the existing "CDP browser" and web server look, and how they present their data. This will give an overview of what "ICD Software Ltd" considers vital information to present in a browser. Then analyze which parts of these GUIs work well and which parts might have room for improvement.

This analysis should also determine which of the data presented are absolutely necessary at any given time. *(e.g. When you look at a components list of signals, what is pertinent information? The full unique name, the short name, the value? Are the components parameters needed in the same display?)*This will help determine how to divide the existing views in to logical units that can be displayed separately, without losing important information. As mobile devices have a much smaller screen than a regular computer, too much information on one screen would be perceived as clutter and confuse the user. Furthermore there is the purely physical aspect of it, more information on the screen means smaller font that is hard to read, and smaller objects on the screen are hard if not impossible to navigate using a touch screen.

A natural second step would be to explore what solutions are used and available on mobile devices today. The frequency at which each solution is used will give an indication of what would give the user a familiar feel, and thereby give an intuitive way of navigating the application and looking at data. During this exploration there should also be a focus on how existing solutions can be adapted to suit this project data?

The exploration of existing solutions might not uncover any viable ways of displaying the data needed for this project. In such case, the project team will take a closer look at the "QtQuick" library. This might uncover existing classes that can be adapted and combined to create the desired way of displaying data for this project.

Mobile devices add some limitations to design options in regard to screen size, limiting how much information can be displayed at any given time. Yet it may also present new design options that are not available on a normal computer. *(e.g. screen orientation sensor for changing how data is displayed)*

7.3 Methodology

For the initial analysis of a “CDP controllers” web server user interface, the project team has been given access to a controller that follows all “CDP” installations. This is the “FunctionGenerator”, which is a “CDP controller” that has functionality to generate signals for any “CDP controller” to subscribe to or access its controlling parameters.

As for investigation of what is available on today’s mobile devices, the project team gained access to smart phones and tablets from different manufactures (e.g. Samsung, Apple) that contain different operating systems (i.e. Android, iOS). Combining this with the project team experience;

- Private users for five to seven years.
- One Bachelor of Arts.
- One “Telecommunications” officer for eight years.
- One software developer using “CDP2Qt” for two years.
- All have two years at university collage with computer engineering.

This gives the project team a wide basis in which to make decisions on; what solutions are frequently used on mobile devices? What solutions work well and are intuitive? Which solutions do not work as well as could be expected?

Each operating system comes with a wide selection of applications *(App’s)* that are available to download. To find solutions suitable for this project, the main focus will be on applications that collect and show dynamic data *(e.g. RSS feeds, Facebook, messages)*.

For the initial design, the project team members have decided to rely on the product of their analysis of the “CDP controllers” web interface and the exploration of existing solutions for mobile devices. A decision not to include more experienced engineers in the early stages of GUI design was in the interest of the learning process. Giving the project team an opportunity to, by their own reasoning, find their own solutions. Not influenced by experienced engineers giving solutions that might be perceived as facts, for reasons none other than the experience of the engineer.

However, after the initial design is ready there is a need to evaluate it. And this is conceived to be best done by an engineer that have experience with “CDP controllers”, and might one day become the user of derivatives from this project.

7.4 CDP web interface analysis.

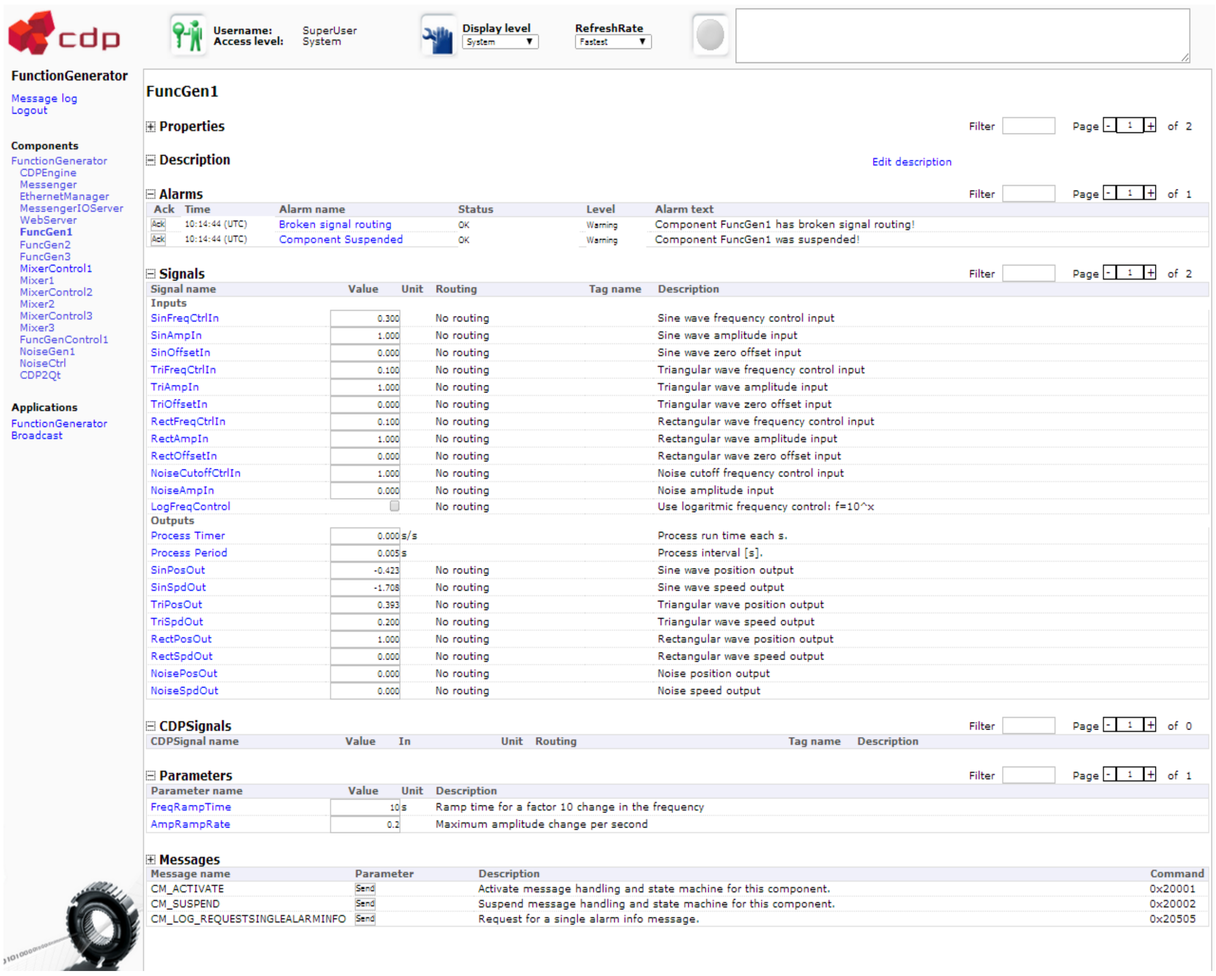
The analysis is done by studying the CDP web interface layout, with focus on the grouping of objects. This information, combined with the project teams' knowledge of "CDP controllers" through work and the "ICD system manual", is then used to identify logical grouping possibilities. The principal reason for this is to make smaller more manageable blocks of information that can easily be shown on a smaller mobile devices screen.

Another aspect of the analysis is the imagined use of this projects prototype. As this impact which information is conceived to be vital to a user.

This analysis is divided in to one chapter for each of the logical groups the project team found during its analysis of visual objects on the web interface.

7.4.1 Initial

The initial impression of the web view of CDP web server is that it presents a lot of different information and data on one page;-this is less than optimal for smaller screens. The design is also divided in to clearly defined groups with rather closely spaced objects. The grouping makes it easy to identify natural logical units within the design, but the lack of space between objects in a group can be problematic when navigating with a touch screen.



Username: SuperUser
Access level: System
Display level: System
RefreshRate: Fastest

FunctionGenerator

Message log
Logout

Components

FunctionGenerator
CDPEngine
Messenger
EthernetManager
MessengerIOServer
WebServer
FuncGen1
FuncGen2
FuncGen3
MixerControl1
Mixer1
MixerControl2
Mixer2
MixerControl3
Mixer3
FuncGenControl1
NoiseGen1
NoiseCtrl
CDP2Qt

Applications

FunctionGenerator
Broadcast

FuncGen1

⊞ **Properties** — Filter — Page 1 of 2

⊟ **Description** — Edit description

⊟ **Alarms** — Filter — Page 1 of 1

Ack	Time	Alarm name	Status	Level	Alarm text
Ack	10:14:44 (UTC)	Broken signal routing	OK	Warning	Component FuncGen1 has broken signal routing!
Ack	10:14:44 (UTC)	Component Suspended	OK	Warning	Component FuncGen1 was suspended!

⊟ **Signals** — Filter — Page 1 of 2

Signal name	Value	Unit	Routing	Tag name	Description
Inputs					
SinFreqCtrlIn	0.300		No routing		Sine wave frequency control input
SinAmpIn	1.000		No routing		Sine wave amplitude input
SinOffsetIn	0.000		No routing		Sine wave zero offset input
TriFreqCtrlIn	0.100		No routing		Triangular wave frequency control input
TriAmpIn	1.000		No routing		Triangular wave amplitude input
TriOffsetIn	0.000		No routing		Triangular wave zero offset input
RectFreqCtrlIn	0.100		No routing		Rectangular wave frequency control input
RectAmpIn	1.000		No routing		Rectangular wave amplitude input
RectOffsetIn	0.000		No routing		Rectangular wave zero offset input
NoiseCutoffCtrlIn	1.000		No routing		Noise cutoff frequency control input
NoiseAmpIn	0.000		No routing		Noise amplitude input
LogFreqControl	☐		No routing		Use logaritmic frequency control: f=10^x
Outputs					
Process Timer	0.000	s/s			Process run time each s.
Process Period	0.005	s			Process interval [s].
SinPosOut	-0.423		No routing		Sine wave position output
SinSpdOut	-1.708		No routing		Sine wave speed output
TriPosOut	0.393		No routing		Triangular wave position output
TriSpdOut	0.200		No routing		Triangular wave speed output
RectPosOut	1.000		No routing		Rectangular wave position output
RectSpdOut	0.000		No routing		Rectangular wave speed output
NoisePosOut	0.000		No routing		Noise position output
NoiseSpdOut	0.000		No routing		Noise speed output

⊟ **CDPSignals** — Filter — Page 1 of 0

CDPSignal name	Value	In	Unit	Routing	Tag name	Description

⊟ **Parameters** — Filter — Page 1 of 1

Parameter name	Value	Unit	Description
FreqRampTime	10	s	Ramp time for a factor 10 change in the frequency
AmpRampRate	0.2		Maximum amplitude change per second

⊞ **Messages**

Message name	Parameter	Description	Command
CM_ACTIVATE	Send	Activate message handling and state machine for this component.	0x20001
CM_SUSPEND	Send	Suspend message handling and state machine for this component.	0x20002
CM_LOG_REQUESTSINGLEALARMINFO	Send	Request for a single alarm info message.	0x20505

Figure 1: Web interface of the controller "FunciontGenerator.FuncGen1" component.

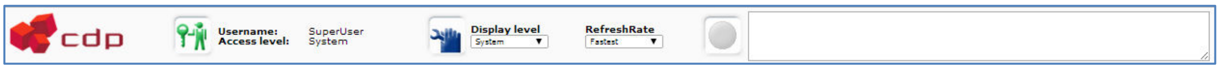
7.4.2 Pr. Connection objects.

These visual objects provide information about options that concern the entire connection. Some of this information is vital in establishing a connection to the "CDP controller" *(e.g. URL, username and password)*. After a connection is established this information is not as relevant to the imagined use of the application, of giving access to the controllers objects such as the value and name of inherent objects. Yet the information still needs to be accessible to the user of the application. As such the project team foresees a need to add "login" page to gain access to "CDP controllers", after which the information can be accessed through a menu or by returning to the "login" page. The menu should also contain a few more options such as:

- An option to log out of the "CDP controller"
- Give access to the "message log"
- Setting the frequency in which the application will pull data from a "CDP controller"

The "pr. Connection" objects from the web interface are identified as:

- All objects grouped in the header of web view including the URL of the browser.
- The small group of objects in the top of the side bar. *(e.g. "Logout")*
- The small group of objects in the bottom of the side bar. *("Applications")*



Username: SuperUser
Access level: System
Display level: System
RefreshRate: Fastest

Figure 15 : The header on the CDP web view.

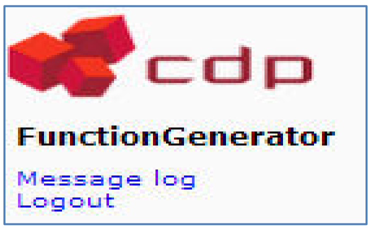


Figure 17 : Top of sidebar on the CDP web view.

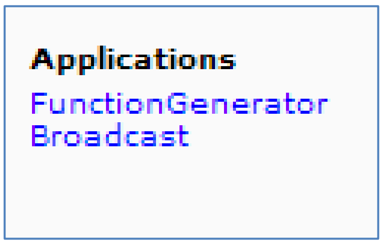


Figure 16 : Bottom of sidebar CDP web view

7.4.3 Navigation objects.

These visual objects provide information about where in the hierarchical structure of the "CDP controller" the active/selected "component" is, and a method for navigating between these "components".
The web interfaces way of doing this is presenting the "CDP controller" and its "components" in a tree structure, with a slightly bolder caption on the active/selected component.

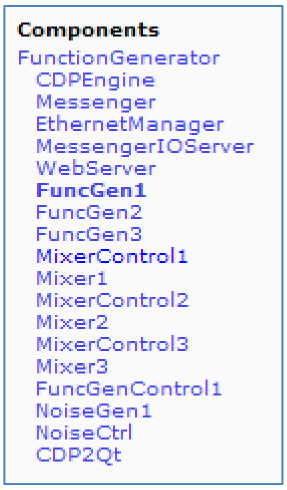

Components
FunctionGenerator
  CDPEngine
  Messenger
  EthernetManager
  MessengerIOServer
  WebServer
  FuncGen1
  FuncGen2
  FuncGen3
  MixerControl1
  Mixer1
  MixerControl2
  Mixer2
  MixerControl3
  Mixer3
  FuncGenControl1
  NoiseGen1
  NoiseCtrl
  CDP2Qt

Figure 18 : Tree structure view of a controller in a web view.

The tree structured view is an easy and intuitive way of navigating the "CDP controller". This is something that could be used on a mobile device, though this structure would need to be modified for a touch screen. One apparent change would be to enlarge the visual objects to be able to "click" them using a finger;-another improvement might be a clearer definition of what is the active "component". With these changes the view would take up a large portion of the small screen on a mobile device. This might become an issue as; the screen should always provide an overview of where the active/selected "component" is located, and navigation may not always have to be shown on screen but should always be easily accessible.

7.4.4 “CDP member” visual objects

These visual objects represent «CDP members” that are inherent to all “components”. A “component” does not have to use all of these «CDP members”, but they are available to all “components”.

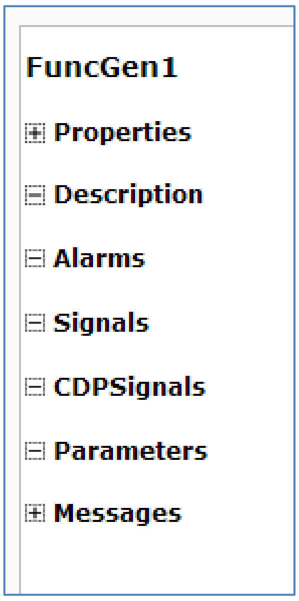
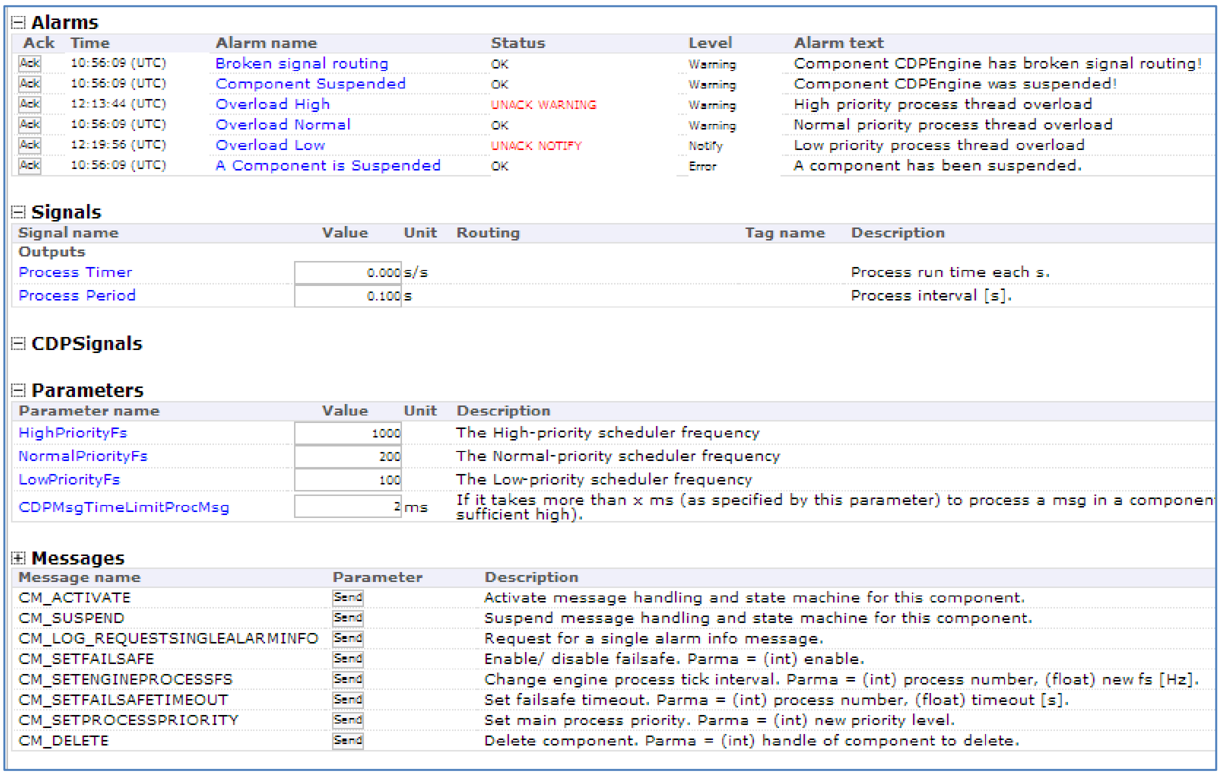

FuncGen1
⊞ Properties
⊟ Description
⊟ Alarms
⊟ Signals
⊟ CDPSignals
⊟ Parameters
⊞ Messages

Figure 19 : List of members on a component in web view.

The “Properties” and “Description” members the project team consider more or less “static”. These «CDP members” are set by the developer of the “CDP controller” to ensure the application runs properly, and are not a part of the “CDP controllers” programmed functionality. As such they are less likely to be changed during runtime. However they might contain information that is relevant during service on the “CDP controller” and therefor need to be accessible, but they can be separated from the rest of the «CDP members”.
The rest of the «CDP members” directly interact with the programmed functionality of a “CDP controller”, therefor it is natural to group these in the same logical unit. They contain input and output data the “CDP controllers” logic use at run time as well as “Messages” to control the inherent state machine.

7.4.5 “CDPObject” visual object

These visual objects have a specific layout dependent on the model they are created from, but all data in the layout is specified by the developer of the controller. These “CDPObjects” are identified by a unique name derived from its parents and its own “ShortName” attribute.



Alarms

Ack	Time	Alarm name	Status	Level	Alarm text
Ack	10:56:09 (UTC)	Broken signal routing	OK	Warning	Component CDPEngine has broken signal routing!
Ack	10:56:09 (UTC)	Component Suspended	OK	Warning	Component CDPEngine was suspended!
Ack	12:13:44 (UTC)	Overload High	UNACK WARNING	Warning	High priority process thread overload
Ack	10:56:09 (UTC)	Overload Normal	OK	Warning	Normal priority process thread overload
Ack	12:19:56 (UTC)	Overload Low	UNACK NOTIFY	Notify	Low priority process thread overload
Ack	10:56:09 (UTC)	A Component is Suspended	OK	Error	A component has been suspended.

Signals

Signal name	Value	Unit	Routing	Tag name	Description
Outputs					
Process Timer	0.000	s/s			Process run time each s.
Process Period	0.100	s			Process interval [s].

CDPSignals

Parameters

Parameter name	Value	Unit	Description
HighPriorityFs	1000		The High-priority scheduler frequency
NormalPriorityFs	200		The Normal-priority scheduler frequency
LowPriorityFs	100		The Low-priority scheduler frequency
CDPMsgTimeLimitProcMsg	2	ms	If it takes more than x ms (as specified by this parameter) to process a msg in a componen sufficient high).

Messages

Message name	Parameter	Description
CM_ACTIVATE	Send	Activate message handling and state machine for this component.
CM_SUSPEND	Send	Suspend message handling and state machine for this component.
CM_LOG_REQUESTSINGLEALARMINFO	Send	Request for a single alarm info message.
CM_SETFAILSAFE	Send	Enable/ disable failsafe. Parma = (int) enable.
CM_SETENGINEPROCESSFS	Send	Change engine process tick interval. Parma = (int) process number, (float) new fs [Hz].
CM_SETFAILSAFETIMEOUT	Send	Set failsafe timeout. Parma = (int) process number, (float) timeout [s].
CM_SETPROCESSPRIORITY	Send	Set main process priority. Parma = (int) new priority level.
CM_DELETE	Send	Delete component. Parma = (int) handle of component to delete.

Figure 20 : CDP members with layout of child objects.

As each “CDPObject” have the same layout within a «CDP member”, it is only natural to separate them in to one logical unit/group for each «CDP member”. This will enable styling of each group individually, and the possibility to keep one group visible at the time, as there is no apparent need to show all “Signals” and “Parameters” at the same time.

7.5 Exploration of existing mobile device design techniques.

7.5.1 About.

This chapter will identify various design techniques commonly used in existing applications on mobile devises. Each technique is analyzed to identify what is its purpose in an application and what they are most commonly used for?

7.5.2 Techniques

Flick.

This is where the screen is dragged either horizontally or vertically to reveal or hide sections of the application.
This solution is widely used in many applications, and gives a smooth and intuitive way of navigating between different logical groups in an application. Furthermore it is good way to hide information that is not needed at that time, but keeps it easily accessible. This is achieved by moving current screen content with the content entering the screen.
For a more application transitional illusion, the current content on screen is kept still while new content move inn and over it.

Flip.

This is where there is an animation that makes it appear as the screen flips over on itself to display what is hidden beneath.
This is a widely used transition in applications to demonstrate that the user is moving to another part of the application with different functionality.

Tabs

These represent banners for groups of information to give the illusion of file folders.
Tabs is a widely used way of grouping information or functions in to distinct pages or views , where only one tab is visible at the time.

Lists

A widely used method for displaying lots of data and is highly customable and dynamic for adding and removing data.

Forms

These present the user with a predefined layout for entering, editing or reading data.
Forms are mostly used to present a user with a well-defined layout to access more static data that only change when a user decides to change them. Not well suited for a dynamic display of ever changing data.

Popups

These give the user an impression of a second window or bubble that presents information. They are smaller than the main screen and if a window is presented, give the impression that no further navigation is possible from this window, only back to last screen. In the case of a bubble popup this often give information about a change in state, and is purely informative or can be clicked to access further information about the state change.

Orientation

The possibility to change orientation *(i.e. landscape or portrait)* on the screen of mobile devices present an intuitive way of changing what is displayed on the screen.
This is most commonly used to either add more detail to a view when switching from portrait to landscape and vice versa. This is done by either keeping the layout of the view and just adding a field or changing the views layout altogether. The later should be used with care as, if poorly executed, it can add confusion to the application rather than be a nice feature.

7.6 Design

Throughout the design phase the main focus has been on how to present and interact with data on a "CDP controller", and how to make a good design for the smaller touch screens of mobile devices. To conform to the scope of this project in regard to its time limit , most of the focus have been on designing the actual "data view". This is the view that will show all objects on a "CDP controller", and is perceived to present the biggest challenge.

The design for the "data view" is influenced by three main factors:

1. What the app need to show?
   This is represented in the analysis of the "CDP web view".
2. How the app should show the data?
   Some choices are presented in the analysis of existing applications for mobile devices.
3. What is available in the chosen IDE to achieve this?
   There are classes available in "QtQuick", and these can be customized to mimic the behavior of known design techniques. Some classes are readily available with the wanted design technique others need more customization. As a result, some design choices might be purely convenient for the purpose of producing a working prototype in the time allowed. It is up to the final review at this projects conclusion to decide the soundness of these choices.

Some consideration has also been given to the idea that there should not be too many "levels" of data on the application. Navigating too many "levels" of data could be perceived as confusing and tedious to frequent users.

7.6.1 The front page

The application should behave somewhat like one expect from any other application on a mobile device, to give the user a familiar look and feel right away.
This could be achieved by presenting a splash screen at start up. This would also give the opportunity to mask startup time, and present some nice company graphics.
After the splash screen, there should be a main page to present the user with a choice of actions. This main page would also add some modularity to the application, with a canvas to present other functionality at a later time. Some regular actions presented at the start of an application is; “Settings” and “App function A”/ “App function B”/…. . As this prototype only will contain the “Browser”, this will be the only “App function” available on the main page along with “Settings” button. *(“Settings” button is purely for illustration purpose in this project)*
Selecting the “Browser” option will move the user to another part of the application; this will be represented by a “flip” animation.

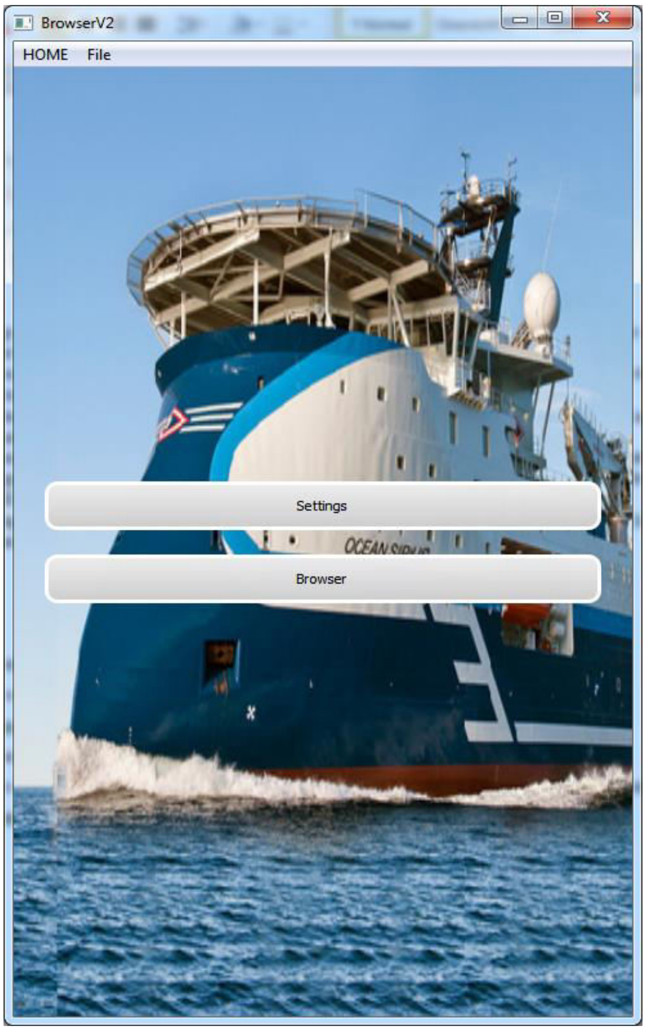
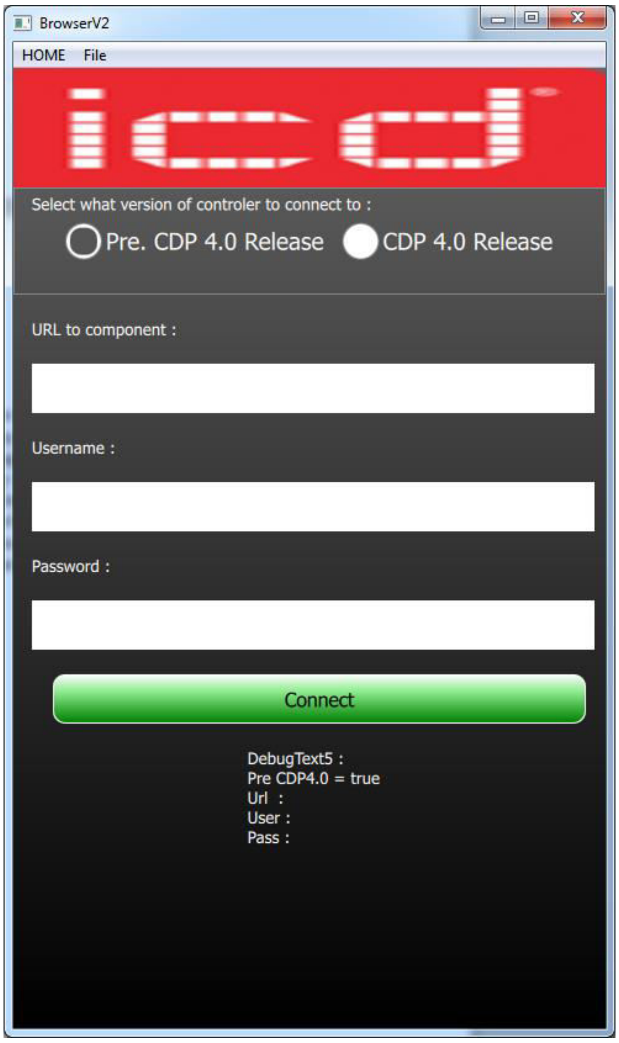


Figure 21 : Main page of the "CDP browser"

7.6.2 Browser "connection view"

For the application to connect to a "CDP controller" there are some information the user need to enter in to the application. This is needed before any information is received from a "CDP controller", and should be designed as a separate page in the app. Thus freeing up space from the "data view" and adds a "level" to the design.
The application will be designed with the future possibility of connecting through two separate protocols in mind, thus needing an option to select what kind of "CDP controller", to connect to. This will be done with a radio button as the first option on this "connection view". Secondly the application needs to know where to connect to. This will be provided by a text input field where the user can enter the URL to the controller. And thirdly text input fields to enter user name and password to authenticate the user and give access to the controller. Upon "connect" another "flip" animation illustrates that the user moves to another part of the application, or one "level" in.



DebugText5 :
Pre CDP4.0 = true
Url :
User :
Pass :

Figure 22 : The "connection view" of the CDP browser

7.6.3 Browser "data view"

This is where all the data on a "CDP controller" will be displayed and interacted with. And as such have to provide all the necessary functionality and every view needed to present this data in a well-organized mater. It will have to display vital information to the user for most common scenarios and give easy access to any other information.

The first decision, with the idea of "less is more" [38], was to separate the "component" and "CDPObjects" in to a simple view and a detailed view. Whereas the simple view only show the most commonly accessed information and the rest of the properties is designated to the detailed view. Thereby only having to show the "CDP members"; Alarms, Signals, CDPSignals, Parameters and Alarms in the main view. For these "CDP members" it was decided to use a "tab view" where each "CDP member" got their own "tab", as there is no need to show two different «CDP members" at the same time. And each «CDP member" has different information and functionality relevant to a simple view list. Their corresponding "CDPObjects" would have similar design within one «CDP member" in a simple view, with the option to select them for a more detailed view of all their properties.

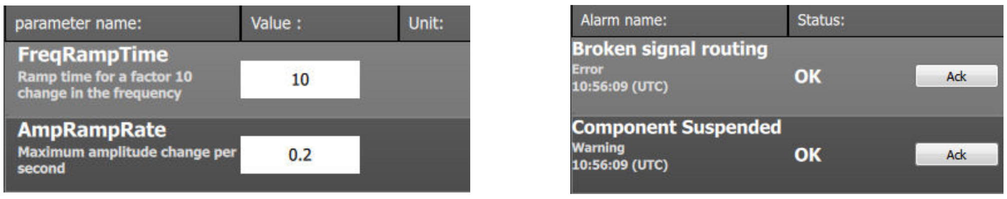


Figure 23 : Design of the CDPObject "Parameter" and "Alarm".

For the "navigation" part of the view, the decision was made to use a hierarchical list of "CDP controllers" and their "components". But as this list can be quite substantial and would take up too much space of the view, it was decided to hide this list out to the left of the main view. It will still be accessible by a "swipe" or "flick" motion, which will give the feeling of staying on the same view, and keep it easily accessible at all times.

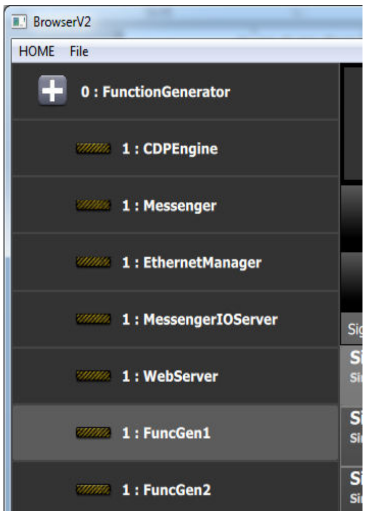


Figure 24 : Design for the "navigation view".

This decision presents the project team with another challange, the user should always be shown where on the "CDP controller" the active/selected "component" is located. This is vital information that is needed to prevent user errors when using the app, and is therefore always relevant. In the "CDP web interface" this was done by the hierarchical navigation list, and as it was decided to hide this, that is not an option for this application. Therefor it was decided to make dedicated "field" or "window" at the top of the main view, with the sole purpose of showing where the user is located in the hierarchy. The main focus in this "windows" design is to be simple, clear and unambiguous, for it to be efficient.

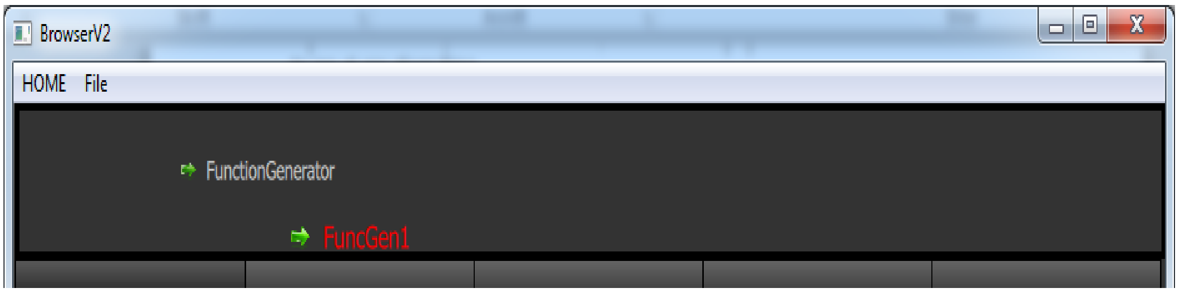


Figure 25 : Design of the "path view".

This new "path view" at the top of the screen presents a new possibility of how to access the components detailed properties. If this "path view" shows the active component, it is intuitive to click it to get more information about it. This information might be presented as a "popup" window, but other options are just as viable. One alternative option is sliding the screen "down" to gain access to it. Another alternative is to add a dropdown menu when clicking the location view. As the project team got limited experience with controllers, this would add some modularity to accommodate unforeseen additions. *(E.g. And a way to give quick access to frequently used properties on a controller.)*

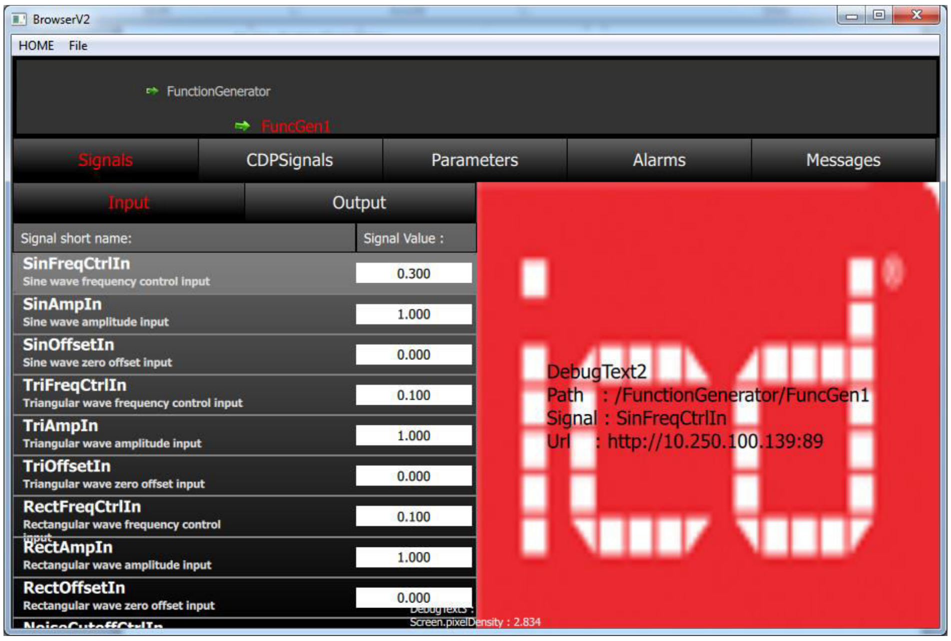
7.6.4 Orientations

Portrait orientation on a smart phone was decided early on to only contain one list of "CDP member" objects at the time. And the top of the screen would be used as a "path view" to show the active "component". The accessibility of the "navigation view" to the left of the screen would be controlled by the ability to flick the screen. But the issue of showing each "CDPObjects" detailed properties still remained.

The options considered were to show a popup window, flip the screen or flick the screen to the left. Both popup window and flipping the screen were considered to be viable options, but the final decision was to use the flicking option. This option was considered to be the best, due to the intuitive way of navigating a "CDPObject", and the notion that it would work well in conjunction with the landscape orientation.

Portrait orientation on a tablet got a bigger screen than the phone and with this the possibility to show more information without it becoming unreadable. For this reason it was considered to show both the "CDP members ""simple list" and "detailed view" at the same time on a tablet. After some quick testing it was decided that for this to be a good solution, it would need a lot more adjustment to the design than time would allow. For this reason it was decided that portrait mode for tablets would look the same as for a smart phone in the interest of producing a prototype within the time frame.

Horizontal orientation was considered to be large enough to show both the simple list and detailed view for both smart phones and tablets. As there are phones with smaller screens and lower resolutions than what is considered viable for this projects purpose, this would only add a device requirement. And as such, in this project there is no focus on designing a view for mobile devices with a screen size less than 3.5" and a resolution less than 960×640.
(The size and resolution of an iPhone 4)
One design option for a horizontal view is to change the entire layout, but the decision was made, for the purpose of this project, adding of the detailed view is a better solution. This would keep a uniform look to the application. Some consideration was made in to adding of the "navigation list" as a constant part of the horizontal view of tablets, but it was later removed to keep a cleaner look to the application.



Signal short name:	Signal Value :
SinFreqCtrlIn Sine wave frequency control input	0.300
SinAmpIn Sine wave amplitude input	1.000
SinOffsetIn Sine wave zero offset input	0.000
TriFreqCtrlIn Triangular wave frequency control input	0.100
TriAmpIn Triangular wave amplitude input	1.000
TriOffsetIn Triangular wave zero offset input	0.000
RectFreqCtrlIn Rectangular wave frequency control input	0.100
RectAmpIn Rectangular wave amplitude input	1.000
RectOffsetIn Rectangular wave zero offset input	0.000

DebugText2
Path : /FunctionGenerator/FuncGen1
Signal : SinFreqCtrlIn
Url : http://10.250.100.139:89

DebugText3 :
Screen.pixelDensity : 2.834

Figure 26 : Design of the horizontal "data view"

7.7 Considerations

Some considerations are not addressed in regards to the design of this application, this is to conform to the projects scope. Yet they should be mentioned in the interest of any future work derived from this project.

The first thing to mention is the color scheme, on this project it is the product of personal preference of the team members. But to create a ready product from the prototype, there are some aspects to the coloring scheme that need to be addressed.
One of which is the pure esthetics of the application, how it looks to the user. Another is “where” and “when” it will be used, as the “where” would dictate any standards it would have to conform to *(e.g. on a* ***spørre en lærer for hmi farge regler*** *shi[?]p or an oil rig)*. And the “when” is as in, time of day, would provide an indication of the need for a bright daytime theme and a dimmer nighttime theme. Furthermore there are some aspects regarding colorblindness that need to be considered. Should the main theme take colorblindness in to consideration, or should there be an option to change theme.

A second consideration is audio, would any part of the application benefit from adding sound? The transition between different parts of the application or if an “alarm” value changes, both might benefit from the adding of sound.

8 Implementation

8.1 Methodology

8.1.1 About

This chapter talks about how the team learned to develop in Qt, and what resources was used.

8.1.2 Learning Qt and QML

Stian was the only on in the group with experience with Qt, but he had never coded in QML. The team learned most of its Qt and QML knowledge through the official Qt Documentations found on Qt Projects homepage or in Qt Creator. Qt Documentation contains information about all APIs, supported platforms, tutorials, update notes and examples. Information found in the Qt Documentation is all one should need to start developing with Qt.¨

In Qt Creator there's a shortcut to access the Qt Documentation. Pressing F1 while hovering over a Qt class or function a window will open on the left containing the Qt Documentation.

Useful information can also be found on Qt Digias official YouTube channel, Qt Studios. Here the developer of Qt themselves can teach further about the Qt framework, QML and C++.[25]

8.1.3 Communicating with the "CDP Controller"

The CDP documentations [24] installed with the "CDP Controller/Function Generator" revealed the webserver used a RESTful API to listen to HTTP requests and sends responses as XML [30]. The CDP documentation lists all available GET commands and all its parameters.

The team used Google Chrome Developer Tools, a set of debugging tools built into the Google Chrome browser, to check what commands are getting sent when navigating the web interface. These commands where used to check what kind of information was received in the XML files.

With this information the team searched for XML parsers in Qt Documentation and found an XML module for QML, called XmlListModel, which creates a read-only model from XML data[31]. This data can be sent to a View (e.g. ListView, PathView, GridView). To test all this out the team built an application capable of listing signals of a predetermined "CDP Controller" component.

To make GET requests the team looked to the XMLHttpRequest API available in JavaScript. It is used to send HTTP requests behind the scenes. Despite its name, the API can receive any textual data, not just XML.

Now the team knew how to send GET requests and they expand the test application with the ability to change a selected signals value. As Qt is cross-platform compatible, the test application was compiled and executed on a Windows desktop for quicker testing.

8.2 The process

8.2.1 Programming choices

ListView

When looking at the existing CDP browser for viable design the team noticed that all the data is presented as lists; lists of components, subcomponents and objects (e.g. signals, parameters, alarms). The Qt Quick library contains a view called "ListView", this element displays information in lists horizontally or vertically. "ListView" is scrollable with flicking since it inherits from Flickable, an item that places its children on a surface that can be dragged and flicked. "ListViews" can also be traversed with scrollbars and arrow buttons.

Alternatives to "ListView" are "GridView" and "PathView".
GridView provides data in a grid view while PathView shows data in a path with the selected item brought to the front.

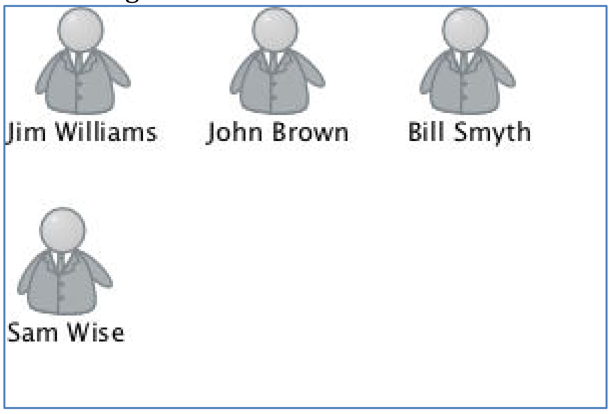


Figure 27 : Example of GridView

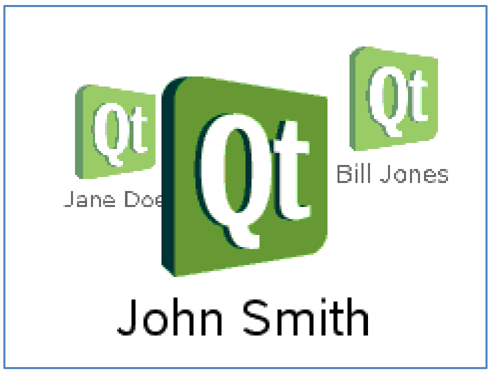


Figure 28 : Example of PathView

A ListView is built from three separate components, the ListView itself which references the other two components, a ListModel that contains the data for the list, and a delegate which contains the formatting of a list element. Lists can be populated by QML types like ListModel, XmlListModel or custom C++ classes.

The ListView delegate provides a template for viewing each element in the ListView. A view delegate should only contain enough QML to view necessary information at that moment. As an example, the only required information in a list of signals could be signal name, a description of the signal and the value of the signal. More information isn't needed in the initial list and any further information about the signal should be gathered upon further user interaction for performance reasons.

XmlListModel

XmlListModel [31] creates read-only data from XML data specified with a “source” property. The “query” property contains the path to the desired elements in the XML feed. The desired elements are gathered in an XmlRole which chooses a variable for each element, the path to the element (with “query” as root) and its data type. The data type must be declared in the XmlRole as this makes caching performance improve drastically.

QML Loader

[33]To assure faster application startup and control over memory usage the application is built modularly, where each component has its own QML file. QML components can be loaded dynamically using a Loader item. With the Loader item the application avoids doing work it don’t need to do, and only loads components when needed. For example a component’s signals don’t need to be loaded before the user asks it to load. The Loader can load a QML file or a QML Component object. Unloading an object or QML file from memory is easy, just change the source property to something else and the previously loaded item gets destroyed automatically.

There’s a lot of information constantly being updated in the application, and to combat lag and freezing Loaders and images can be loaded asynchronous. This means the object will be loaded and compiled in a background thread, making sure other components can start to load before the Loader or image has finished loading.

Property binding

Property binding [34] allows objects to inherit properties from other objects and updating automatically when changed. As an example, a rectangle can inherit its parent’s height and width, making its size dynamic, changing with its parent.

A more efficient way to position elements in QML is with anchors. Each item has seven anchor points that can be used to anchor items to each other, left, horizontalCenter, right, top, verticalCenter, baseline and bottom.

```
Rectangle { id: rect1; ... }
Rectangle { id: rect2; anchors.left: rect1.right; ... }
```

Figure 29 : Two rectangles side by side with rect2 following rect1

Challanges

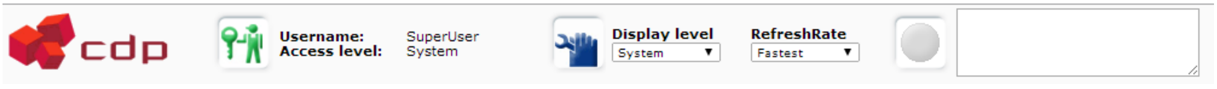
While compiling for Android, two modules didn't import correctly, XML and XmlPatterns. Troubleshooting this problem revealed the option to import them manually from the Qt projects .pro file.

Another problem was with a custom keyboard application that one of the projects team members is using called SwiftKey. Currently SwiftKey can't interact with a QML TextField. This is a registered bug and is scheduled to be fixed in Qt 5.3. [35]

TODO:

The application is missing a detailed view of selected objects or components, and the possibility to change its properties, like the description. A placeholder page is located in the application within object lists (e.g. signals, parameters, messages). Selecting an object and swiping to the left will reveal a blank page where detailed information about the selected object can go.

The application doesn't have the ability to show settings information like user information, access authority, refresh rate and a message log area.



Username: SuperUser
Access level: System
Display level: System
RefreshRate: Fastest

Figure 30 : Screenshot of web interface's top frame

System for user management is missing, but applications start menu contains login views.

9 Konklusjon

9.1 About

Through this projects progression the team worked hard to achieve its goals, and by doing so gained a lot of experience in the field of mobile software engineering. This chapter states the team's opinions of how close to completion on each goal the team came.

9.2 Process goal

- Learn about common mobile software development concepts.
- Get experiences learning a new IDE to increase this learning experience in the future.
- Experience using theoretical knowledge to accomplish a large task through team collaboration.

The team feels that through its research in to mobile software development, it has acquired a good base in which to make informed decisions. These decisions include; what platform to use, available IDEs for selected platform and special considerations needed for it.

Learning the new programming language as well as IDE gave the team many new experiences. And provided the team with new information that might help it more easily identify important aspects of any future learning process. A couple of the most prominent experiences where, the value of running tutorials as well as analyzing existing code. These methods of acquiring knowledge were invaluable in the learning process.

The team work on such a large task was a new experience to all team members. And provided many new and unfamiliar situations, each having to be solved quite differently in a team than when working alone. Situations where there was need for a degree of compromise and diplomacy to reach a mutually agreeable decision. Another situation was the administrative task of delegating work. What should be a priority when performing this task? *(E.g. Individual preference, strengths, weaknesses or work load.)*.
As the work progressed the importance of the theory concerning software development and project work became more and more apparent. *(E.g. Theory from the coarse "Software Engineering(DA-SYS3000)" at "HBV" and books on project work.)*

9.3 Project goal

- Identify a viable way for “ICD Software Ltd” to develop for mobile devices.
- Design a well-defined layout in which to present data from “CDP controllers”.
- Produce a working prototype of the “CDP browser”.

Through the research in to mobile software development, the team found that “Qt” would provide the best development starting point for “ICD Software Ltd”. Information gathered suggested that the Android platform had the biggest market share. And as the company’s current IDE of choice *(“Qt”)* had the possibility to develop for this platform, this IDEs developments possibilities for “Android” was a priority. Further research uncovered that although “Qt” did not support all functionality on mobile devices, it was in the process of being created. And the functionality “Qt” currently supported would suffice for this projects prototype. Furthermore “Qt’s” portability was a factor in the decision, as the current software developed by “ICD Software Ltd” supports multiple platforms. The team considered this a company policy for development, and it would be wise to continue this in the mobile software development process. The sum of these choices, the team feel, gives “ICD Software Ltd” a viable way for developing software for mobile devices.

The team feels the design process’s analytic work method have provided a well-defined and viable layout for the “CDP browser”. The design choices are thoroughly considered and documented. This provides any work derived from this project a good basis to start its work, as well as the opportunity to discuss this projects decision.

The work on the prototype was heavily influenced by the need to learn a new language and IDE. And as the team discovered the more it learnt about the language, the easier it was to find solutions to any design choice or bug that was presented. As such, it is easy to come to the conclusion that with a little more experience with the language, the team could present an even better prototype. However the prototype produced within the time allowed, provided the team with valuable insight and experience. And even though the prototype is far from complete, it still provides the possibility to communicate with a “CDP controller”. Thereby fulfilling the last project goal.

9.4 Summary

Through achieving previously stated goals we feel it have been a satisfactory completion of the project and answered the stated problem.

What considerations need to be taken in to account when designing and creating a prototype browser on smart devices for "CDP-controllers"?

Throughout the process that has been this project, there have been many different challenges. Each with its own set of obstacles, some more challenging than others. But through team work and playing on each other's strengths they were overcome one by one. Either by going round, over, under or sometimes straight through them.
The process veered the team in to unknown territory where it often was hard to distinguish relevant information from everything else. Working through this territory often gave life to new ideas and possibilities which the project could benefit from. As the process was ever evolving, the later stages produced more and more such ideas. All the while there was less and less time to bring these ideas in to life. However some of these ideas deserve some mention:

- Possible "settings" options.
  The setting of variables used in communication with the RESTfull api of the web server.
  This would provide more modularity to the application.
  Setting of detail level of what to show.
  This could give a totally different dynamic and security feature to the application.

- Using the inherent type of "WorkerScript" in the QML code to optimize the feel and effectiveness of the application.
  "WorkerScript" starts a new thread that can carry out instructions. This thread can carry out demanding instructions without interrupting the main thread and GUI.

- The communication with a "CDP controller" is carried out by using QML types, this could be done in pure C++.
  This would remove the load of input/output operations from the main thread as well as give access to the whole C++ library.

10 Bibliography

1. Nokia, (25.jul.2011), "*Windows Phone Platform Security"* Collected 20.05.14 from Developer.nokia.com

2. Richard Grimes, (n/a), "*Managed Or Unmanaged?"* Collected 20.05.14 from http://www.grimes.nildram.co.uk/dotnet/man_unman.htm

3. Microsoft, (n/a), "*What is managed code?"* Collected 20.05.14 from msdn.microsoft.com

4. Wikipedia (n/a), "*Garbage Collection"* Collected 20.05.14 from www.wikipedia.com

5. Microsoft, (n/a), "*Security for Windows Phone"* Collected 20.05.14 from msdn.microsoft.com

6. Wikipedia (n/a), "*Reactive Programming"* Collected 20.05.14 from www.wikipedia.com

7. Microsoft, (n/a) "*App Architecture"* Collected 20.05.14 from msdn.microsoft.com

8. Krishna, (n/a), "*App Store guidelines"* Collected 20.05.14 from http://way2ios.com/development/ios-development-2/app-store-review-guidelines/

9. Apple, (n/a), "*xCode"* Collected 20.05.14 from developer.apple.com

10. Clang-analyzer, (n/a), Collected 20.05.14 from clang-analyzer.llvm.org

11. Wikipedia, (n/a), "*ARC"* Collected 20.05.14 from www.wikipedia.com

12. Colin Wheeler (2010, jan 20) [blogpost] http://cocoasamurai.blogspot.no/2010/01/understanding-objective-c-runtime.html

13. Reza B'Far, 2004, "Mobile Computing Principles: Designing and Developing Mobile Applications with UML and XML", Cambridge University Press

14. Google, "*Dalvik Technical Information",* Collected 20.05.14 from www.android.com

15. Google, “*Get the Android SDK”,* Collected 20.05.14 from developer.android.com

16. Wikipedia, (n/a), “*web application”,* Collected 20.05.14 from www.wikipedia.no

17. Google, (n/a), “*App Manifest”,* Collected 20.05.14 from developer.android.com

18. Wikipedia, (n/a), “*Industrial Control System”,* Collected 20.05.14 from www.wikipedia.com

19. Wikipedia, (n/a), “*Representational_state_transfer”,* Collected 20.05.14 from www.wikipedia.com

20. Wikipedia, (n/a), “*Scrum_(software_development)”,* Collected 20.05.14 from www.wikipedia.com

21. Wikipedia, (n/a), “*Extreme_Programming”,* Collected 20.05.14 from www.wikipedia.com

22. ICDsoftware, (n/a), *“Download”,* Collected 20.05.14 from www.icdsoftware.no

23. ICDsoftware, 2014, *“CDP System Manual”,* V3.5.0, *rev A*

24. ICDsoftware, 2013, “Web server – User manual”, V3.5.0, rev PC2

25. YouTube, (n/a), “QtStudios”, Collected 20.05.14 from www.youtube.com

26. Wikipedia, (n/a), *“Qt (software)”,* Collected 20.05.14 from https://en.wikipedia.org/wiki/Qt_(framework)

27. Qt-project, (n/a), *“User interfaces”*, Collected 20.05.14 from http://qt-project.org

28. Qt-project, (n/a), *“User interface | QtCreator”* Collected 20.05.14 from http://qt-project.org

29. Qt-project, (n/a), *“Indroduction to the QML language”* Collected 20.05.14 from http://qt-project.org

30. W3schools, (n/a), *“The XMLHttpRequest Object”*, Collected 20.05.14 from http://www.w3schools.com

31. Qt-project, (n/a), *“XmlListModel”*, Collected 20.05.14 from http://qt-project.org

32. Qt-project, (n/a), *“ListView”,* Collected 20.05.14 from http://qt-project.org

33. Qt-project, (n/a), *“Loader”,* Collected 20.05.14 from
    http://qt-project.org

34. Qt-project, (n/a), *“Property Binding”,* Collected 20.05.14 from
    http://qt-project.org

35. Qt-project, (n/a), *“[QML/Android] TextField is not working with SwiftKey keyboard ”*
    Collected 20.05.14 from
    https://bugreports.qt-project.org

36. Qt-project, (n/a), *“Qt Quick Controls Overview”*, Collected 20.05.14 from
    http://qt-project.org

37. Qt-project, (n/a), *“Widgets and Layouts”*, Collected 20.05.14 from
    http://qt-project.org

38. Wictionary, (n/a), *“Less Is more”,* Collected 20.05.14 from
    http://en.wiktionary.org

11 Keywords

GUI
Graphical User Interface, how the program looks onscreen.

ICD
Industrial Control Design, company that develop the software.

CDP
Control Design Platform, software to develop controllers.

App
Application, small program running on mobile devices.

Smart devices
Multifunctional media devices *(i.e. smart phones, tablets)*

Mobile devices
same as smart devices

CDP2Qt
Software package adding “CDP” libraries to Qt.

Qt
Software development tool. (IDE)

CDP Controller
Computer that run software to read input, compute the data and then control the outputs according to programmed parameters.

CDP component
Object with strict interface specification. A component is a specific instance of a Model, overriding some of its properties to make it unique. [23]

CDP member
Part of a CDP component with its own unique set of key attributes and layout.

CDPObject
Part of a CDP member using the CDP members attributes and layout.

Extreme programming(XP)
A software development technique, with weigth on programming.

Xp protoptyping
A software development technique based on making prototypes.

SCRUM
Software development technique.

IDE
Integrated development platform.

Sandboxing
Restricting the process interaction with other processes and I/O.

Managed language	Language that is managed by some runtime framework.
Garbage collection	Automatic memory managed technique.
SDK	Software development kit.
NDK	Native development kit.
MVC	Model view controller (design pattern).
BAML	Compiled XAML file to binary.
WinRT	Windows runtime framework.
XAML	Extendable application markup language.
XAP	Packages windows phone app ready for deployment.
XML	Extended markup language.
RESTful	http API that have methods for C,R,U,D.
Java Virtual machine (VM)	For running java bytecode.
Dalvik	Java VM for Android OS.
CSS	Cascading style sheet, defines the look of html file.
Emulator	Emulates an cpu architecture with software.
Gyro	Detection of movement and rotation in a device.
GPS	Detection of geographically location.
Reactive programming	Programming paradigm based on non-sequential flow control.
Observer pattern	Design pattern. in event driven programming.
Event-driven	Objects that listens to an event.
Runtime library/framework	

Compiled code that is used at runtime.

API
Application programming interface.

12 Appendices

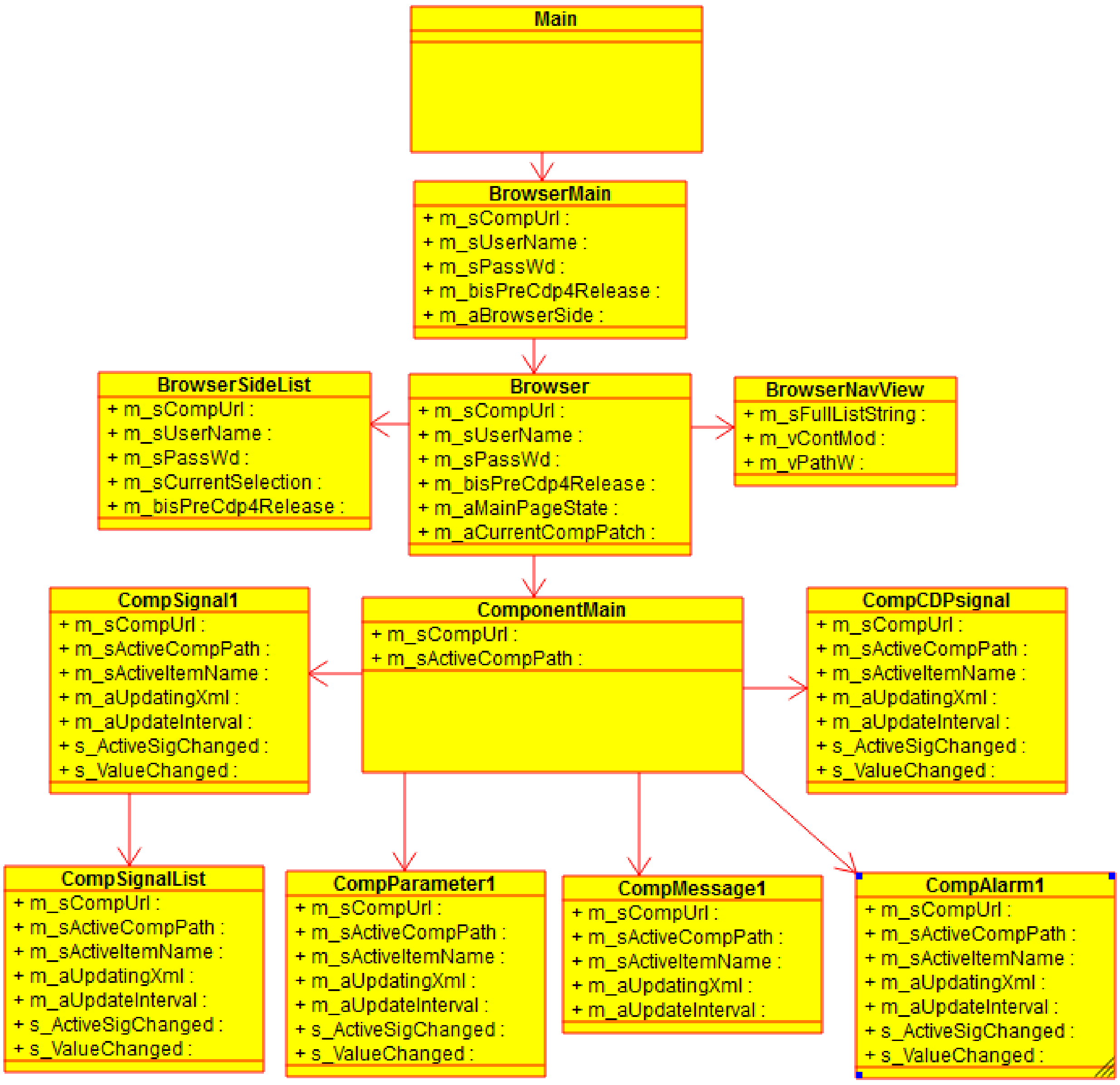
12.1 Appendix A Implementation Issues

Issue Nr :	Short Description:	Type:
1	How to start learning QML and QtQuick ?	Information
2	Make an initial qml application in QtQtuick 1.0.	Training
3	Add functionality to initial application.	Training
4	QtQuick 1.0 Lacks functionality in QtQuick 2.1.	Training
5	Start creating the browser application.	Improvement
~~6~~	~~Create the “navigation view”.~~	~~Improvement~~
	~~- Create a dynamic list.~~	~~Task~~
	~~- Fill the list from a “CDP controller”~~	~~Task~~
	~~- XMLList works on desktop but not on test phone.~~	~~Bug~~
	~~- Make the list behave like a file list tree structure.~~	~~Task~~
	~~- List cannot change individual delegate size on runtime.~~	~~Bug~~
	~~- Make the list expand and collapse delegates.~~	~~Task~~
	~~- Make the list inform the “path view” on active comp.~~	~~Task~~
	~~- Add graphics to delegate, indicate “gotChild”~~	~~Improvement~~
	- Make method expanding/collaps delegate dynamic! - Only support 2 levels atm.	Bug
~~7~~	~~Create the “path view”.~~	~~Improvement~~
	~~- Create dynamic list.~~	~~Task~~
	~~- Create the path of the list~~	~~Task~~
	~~- Add picture to each delegate to show down lvl~~	~~Improvement~~
	~~- Tweak the path to adjust according to screen size~~	~~Task~~
	~~- Create functionality to add/remove from list.~~	~~Task~~
	~~- Make add/remove dynamic only support 6 levels atm~~	~~Bug~~
	- Change path of the list, currently only room for 6 lvl.	Bug
~~8~~	~~Create the “data view”.~~	~~Improvement~~
	~~- Create tabbed view to contain “CDP components”~~	~~Task~~
	~~- Create “detailed view” for “CDP objects” properties~~	Task

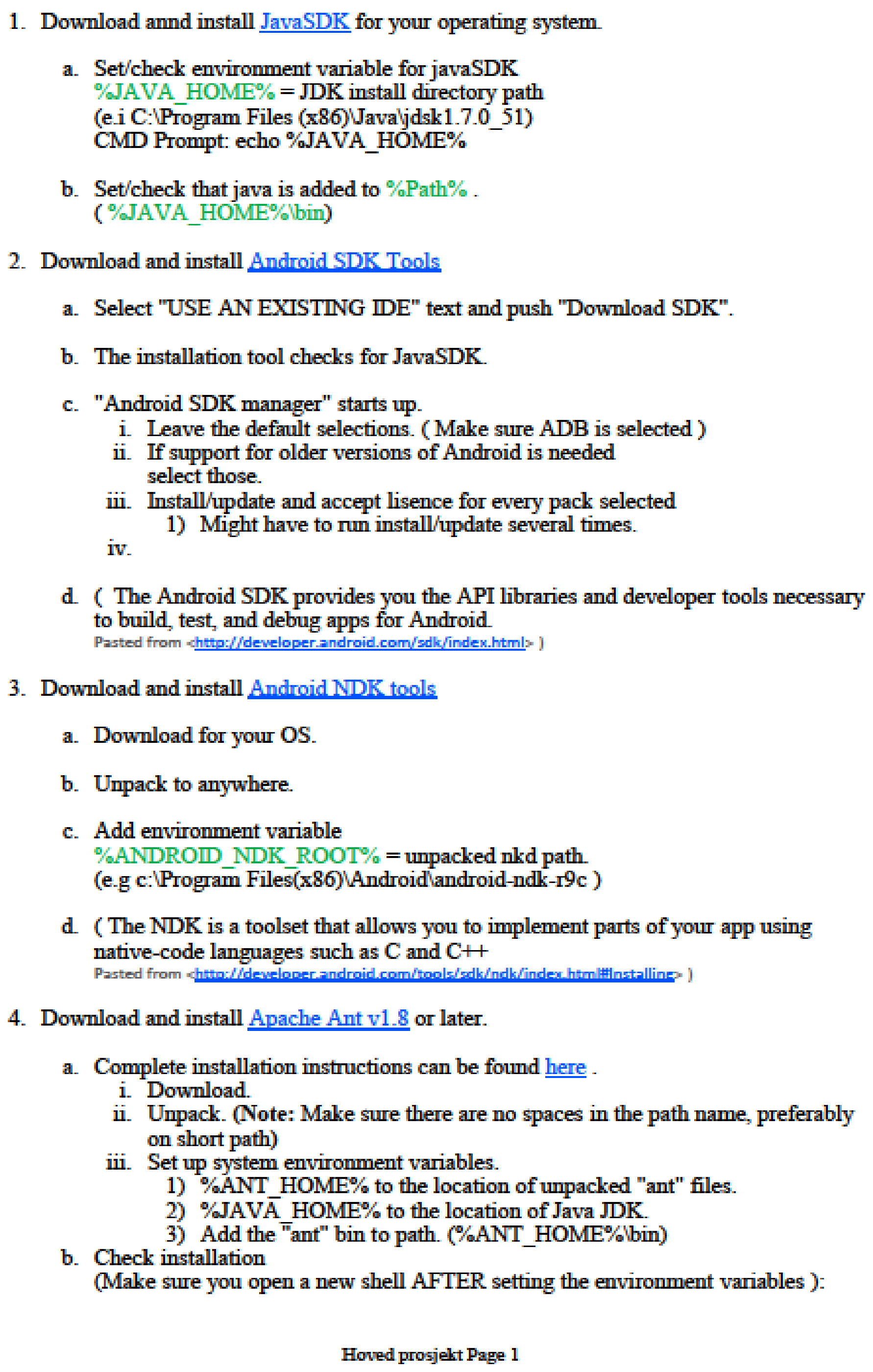
	~~- Create list that fills and update from Signals~~	~~Task~~
	~~- Create list that fills and update from CDPSignals~~	~~Task~~
	~~- Create list that fills and update from Parameters~~	~~Task~~
	~~- Create list that fills and update from Messages~~	~~Task~~
	~~- Create list that fills and update from Alarms~~	~~Task~~
	- Create a view for “CDP component” details.	Task
	~~- Create transition for small screen, “data->detail view”~~	~~Task~~
	~~- Create one list for each of Signals (input/output)~~	~~Task~~
	- When updating input/output list, selection resets.	Bug
	~~- Fetch and update data from app to “CDP controller”~~	~~Task~~
	~~- Create transition “data view -> nav view”~~	~~Task~~
	~~- Make transition not stay in middle position.~~	~~Improvement~~
	~~- Add support for orientation change. (phone/tablet)~~	~~Task~~
~~9~~	~~Create the “connection view”~~	~~Improvement~~
	~~- Add functionality to select “CDP controller” version~~	~~Task~~
	~~- Add functionality to set address of “CDP controller”~~	~~Task~~
	~~- Add functionality to insert username/password~~	~~Task~~
	~~- Add transition to “data view”~~	~~Task~~
	~~- Optimize transition, slow on phone.~~	~~Improvement~~
	-	
	-	
~~10~~	~~Create the “main page”~~	~~Improvement~~
	~~- Add graphic to background~~	~~Task~~
	~~- Add functionality/button to select “browser”~~	~~Task~~
	~~- Add button for “settings” to illustrate possibility~~	~~Task~~
	~~- Add transition to “connection view”~~	~~Task~~
11	Create a splash screen.	Improvement
12	Create a “detail view” for the “CDPObjects”	Improvement
13	Create a “detail view” for the “CDPcomponents”	Improvement
14		
15	Fetch multiple attributes of XML file in QML	Task

12.2 Appendix B UML diagram

This UML diagram displays different QML types and how they are connected in the "CDP browser" prototype.



12.3 Appendix C: Quick installation guide qt android.



1. Download annd install JavaSDK for your operating system.

    a. Set/check environment variable for javaSDK
    %JAVA_HOME% = JDK install directory path
    (e.i C:\Program Files (x86)\Java\jdsk1.7.0_51)
    CMD Prompt: echo %JAVA_HOME%

    b. Set/check that java is added to %Path% .
    (%JAVA_HOME%\bin)

2. Download and install Android SDK Tools

    a. Select "USE AN EXISTING IDE" text and push "Download SDK".

    b. The installation tool checks for JavaSDK.

    c. "Android SDK manager" starts up.
        i. Leave the default selections. (Make sure ADB is selected)
        ii. If support for older versions of Android is needed
        select those.
        iii. Install/update and accept lisence for every pack selected
            1) Might have to run install/update several times.
        iv.

    d. (The Android SDK provides you the API libraries and developer tools necessary
    to build, test, and debug apps for Android.
    Pasted from <http://developer.android.com/sdk/index.html>)

3. Download and install Android NDK tools

    a. Download for your OS.

    b. Unpack to anywhere.

    c. Add environment variable
    %ANDROID_NDK_ROOT% = unpacked nkd path.
    (e.g c:\Program Files(x86)\Android\android-ndk-r9c)

    d. (The NDK is a toolset that allows you to implement parts of your app using
    native-code languages such as C and C++
    Pasted from <http://developer.android.com/tools/sdk/ndk/index.html#Installing>)

4. Download and install Apache Ant v1.8 or later.

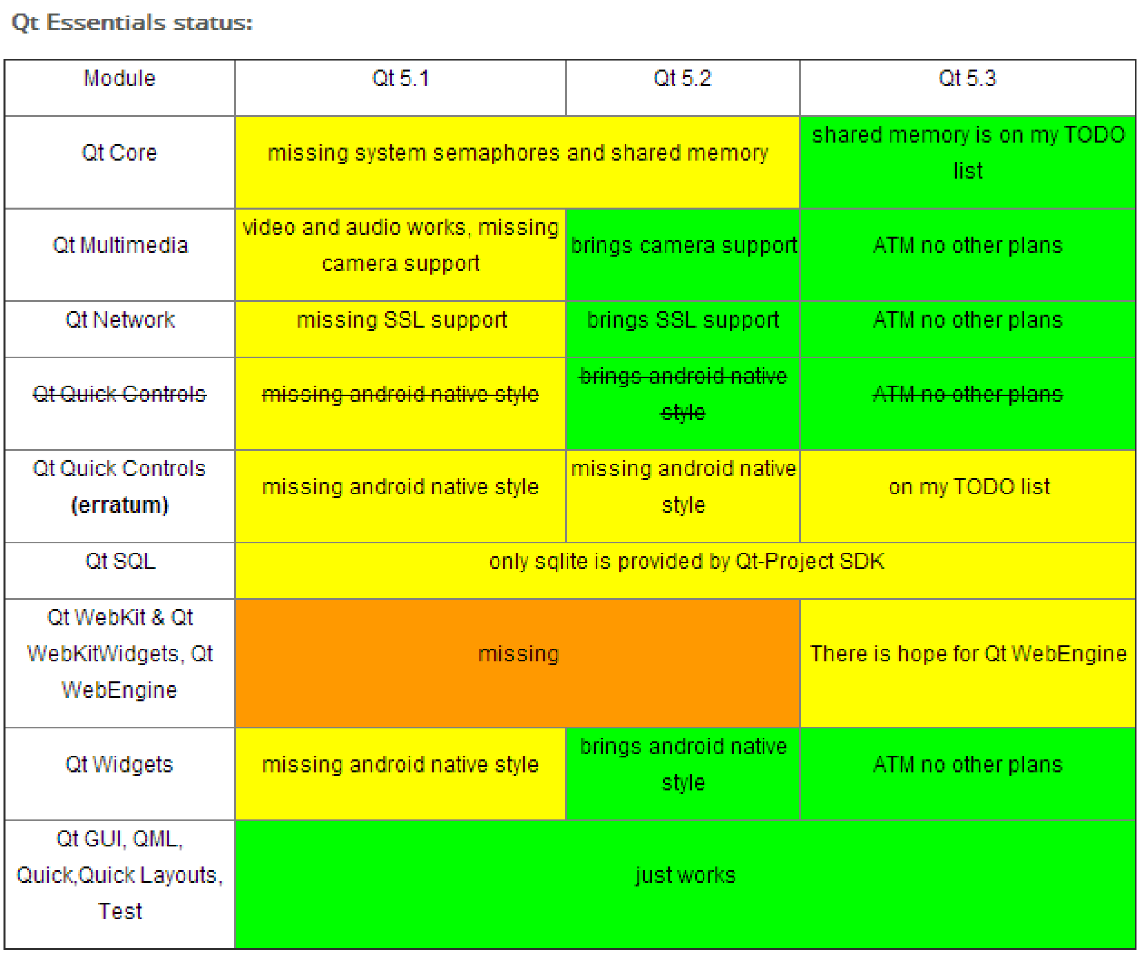
    a. Complete installation instructions can be found here .
        i. Download.
        ii. Unpack. (**Note:** Make sure there are no spaces in the path name, preferably
        on short path)
        iii. Set up system environment variables.
            1) %ANT_HOME% to the location of unpacked "ant" files.
            2) %JAVA_HOME% to the location of Java JDK.
            3) Add the "ant" bin to path. (%ANT_HOME%\bin)

    b. Check installation
    (Make sure you open a new shell AFTER setting the environment variables):

Hoved prosjekt Page 1

12.4 Appendix D: Table of Qt development support for Android.

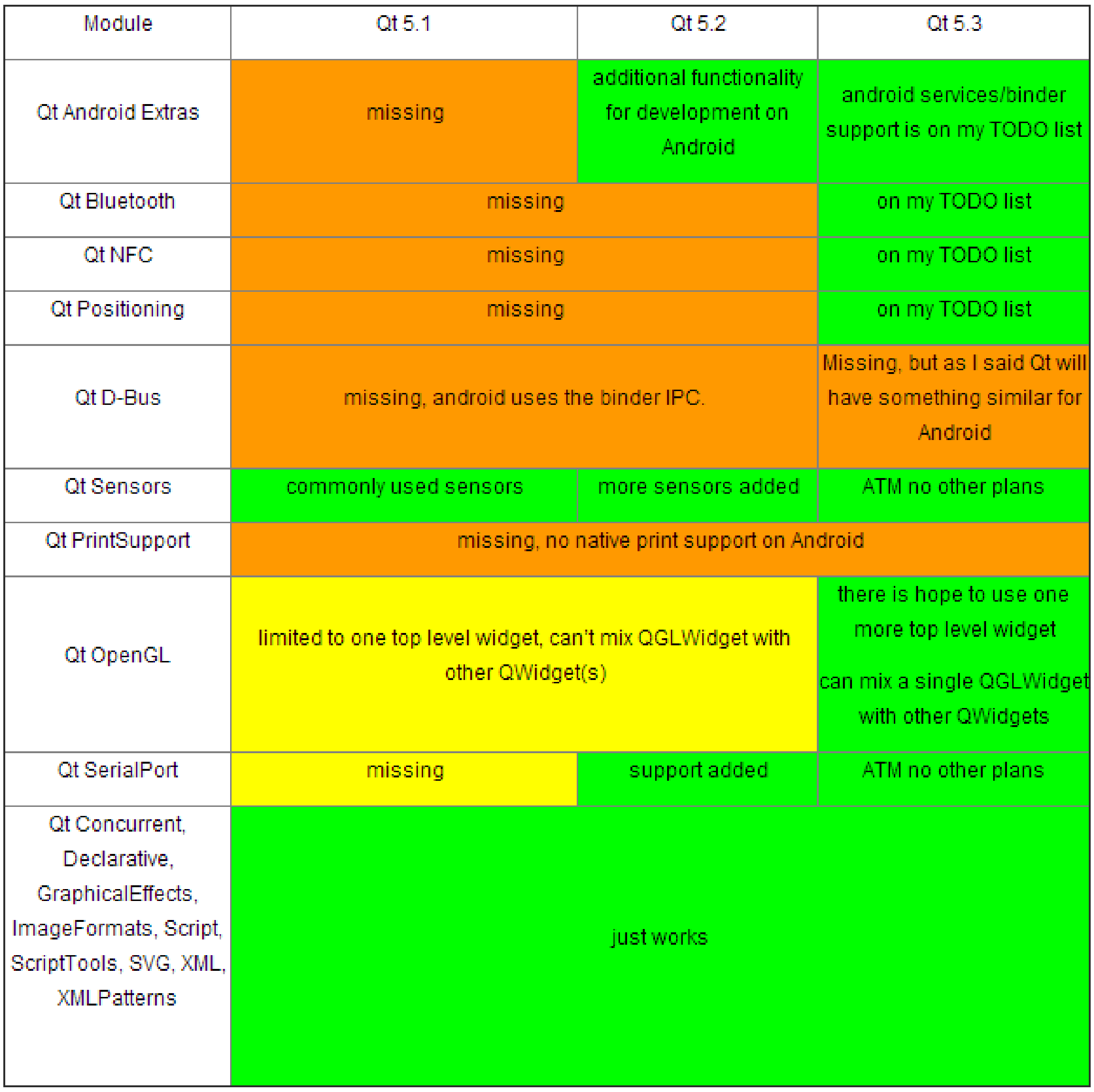
Tables are from https://www.kdab.com/qt-on-android-episode-1/ .



Qt Essentials status:

Module	Qt 5.1	Qt 5.2	Qt 5.3
Qt Core	missing system semaphores and shared memory		shared memory is on my TODO list
Qt Multimedia	video and audio works, missing camera support	brings camera support	ATM no other plans
Qt Network	missing SSL support	brings SSL support	ATM no other plans
~~Qt Quick Controls~~	~~missing android native style~~	~~brings android native style~~	~~ATM no other plans~~
Qt Quick Controls **(erratum)**	missing android native style	missing android native style	on my TODO list
Qt SQL	only sqlite is provided by Qt-Project SDK		
Qt WebKit & Qt WebKitWidgets, Qt WebEngine	missing		There is hope for Qt WebEngine
Qt Widgets	missing android native style	brings android native style	ATM no other plans
Qt GUI, QML, Quick,Quick Layouts, Test	just works		

Qt Add-Ons status:



Module	Qt 5.1	Qt 5.2	Qt 5.3
Qt Android Extras	missing	additional functionality for development on Android	android services/binder support is on my TODO list
Qt Bluetooth	missing		on my TODO list
Qt NFC	missing		on my TODO list
Qt Positioning	missing		on my TODO list
Qt D-Bus	missing, android uses the binder IPC.		Missing, but as I said Qt will have something similar for Android
Qt Sensors	commonly used sensors	more sensors added	ATM no other plans
Qt PrintSupport	missing, no native print support on Android		
Qt OpenGL	limited to one top level widget, can't mix QGLWidget with other QWidget(s)		there is hope to use one more top level widget can mix a single QGLWidget with other QWidgets
Qt SerialPort	missing	support added	ATM no other plans
Qt Concurrent, Declarative, GraphicalEffects, ImageFormats, Script, ScriptTools, SVG, XML, XMLPatterns	just works		