CHALMERS
UNIVERSITY OF TECHNOLOGY

Code structure for developing and validation of PLC program

Control and monitoring of a power transformer by implementing a PLC control, with a code structure designed for flexibility and reusability

Master's thesis in Master Programme MPPEN

Niklas Borg

Department of signals and systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Master's thesis 2015:NN

Code structure for developing and validation of PLC program

Control and monitoring of a power transformer by implementing a PLC control, with a code structure designed for flexibility and reusability

Niklas Borg

Department of signals and system
SSY
Chalmers University of Technology
Gothenburg, Sweden 2016

Code structure for developing and validation of PLC program
Control and monitoring of a power transformer by implementing a PLC control, with a code structure designed for flexibility and reusability
Niklas Borg

Supervisor: Kristofer Bengtsson, SSY
Examiner: Bengt Lennartsson, SSY

Master's Thesis 2016:NN
Department of Signals and systems
Division of SSY
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2016

Code structure for developing and validation of PLC program
Control and monitoring of a power transformer by implementing a PLC control, with a code structure designed for flexibility and reusability
Niklas Borg
MPPEN
Chalmers University of Technology

Abstract

In this project a concept for developing and improving the control and monitoring of a power transformer by using a PLC is presented. A PLC system can contribute with several key functionalities which are necessary in the ongoing reconstruction of the power grid, towards the smart grid concept.

However due to a high variation in the design of the power transformer, it is necessary to use a smart code structure. This report presents a possible code structure of the control logic and simulation model, that can be used for validation of the control logic, as well as reducing the developing time for this type of projects. In order to obtain this, the project have been utilizing features from the object oriented programming which enables for easy modification, adding, removing, etc of different functionalities in the PLC program.

In this project the reusability and flexibility in the code structure have been increased by building the application in several layers based on basic components. This is accomplished by using two of the four foundations of object oriented programming, inheritance and encapsulation. This approach will reduce the required time for developing the control logic but also encourage to utilize the benefits of simulation earlier and give the engineer faster feedback.

Keywords: PLC, PLC programming, Object oriented programming, Power transformer, validation, simulation

Acknowledgements

First, I would like to thank everyone working at the department DC, at ABB transformer in Ludvika for the opportunity of me being apart of a interesting development project. I have learn a lot regarding all the different phases in the developing of a PLC program and setup of a new physical control system.

Also i would like to thanks Anders W, Hans-erik and Jhonny, all working at ABB, for contributing with their expertise within the 800xA system continuously during the project.
Furthermore i would like to thank my supervisor Kristofer which have been contributing with input both regarding the report and developing of the control system.

Niklas Borg, Gothenburg, Juni 2016

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Before the implementation of the first programmable logical controller, more common know as PLC, control logic where embedded in the hardware, consisting of relays, mechanical thermometers, etc. During the years, PLC units have become more common in control system and are today widely used in the industry for different purposes [1]. However, still in today's moderns society there are systems being designed with the control logic embedded in the hardware, often due to the high reliability, quality, low cost and history of earlier implementations [2]. If that is the case many benefits are lost which otherwise possible to obtain with the implementation of a PLC in a control system. For example flexibility of modifying the control logic after installation without modifying the hardware [1] [3] .

However, it is not possible to simply connect a PLC to a control system without defining a program containing the control logic. Unfortunately, the engineer can not be 100% sure that a PLC program is working correctly, before validating it [2]. To validate of the PLC program it is most common to build the real physical system but this result in a lot of time spent on debugging and error fixes during the installation phase at the customer [4] [5]. Also one should considering the fact that there are increasing demand of shorten time to market, quality, complexity and functionality of the control system [6][7]. This means that the room for mistakes, is slowly disappearing and it is important to have fast feedback in order to detect errors early [8].

A second approach to validate a PLC program, is to use simulation, which allows for reallocating considerably amount of time spent on debugging the PLC program to before the start of installing the control system [9]. This is one of many reasons why simulation has become such an important part during development of control system. The engineer can save both effort and money during the testing and validating phase of the control system when using simulation [9]. Even though simulation is a major benefit, since it eliminates the need of hardware, it is costly in terms of time during the creation of the simulation model.

Also in a new project it is often necessary to start from scratch due to high customization. Therefore it is desirable to start reusing old code as much as possible from earlier projects. This requires a code structure that considers how to handle variation. Furthermore the possibility of reusing of code also enables for auto generating code based on the components in the library. However there are several aspects to consider before this is possible [10] [11].

1.2 Industrial Context

ABB is a developer and producer of power transformers which is an essential component in the power sector. This component minimizes the energy loss when the produced energy should be transported over long distances by increasing the voltage and has become a vital part for the modern society. The principle with a power transformer is easy, either you increase or decrease the voltage depending of desired outcome. This is achieved by using two coils and a magnetic field that occurs when the current passes through it. This means that losses due to for example resistance in the wires used for the transportation of the energy can be reduced significantly[13]. However there are still minor losses which results in heat and therefore there are need of cooling equipment for this purpose, See appendix A for more information regarding the power transformer itself and figure 1.1 for the appearance of power transformer.

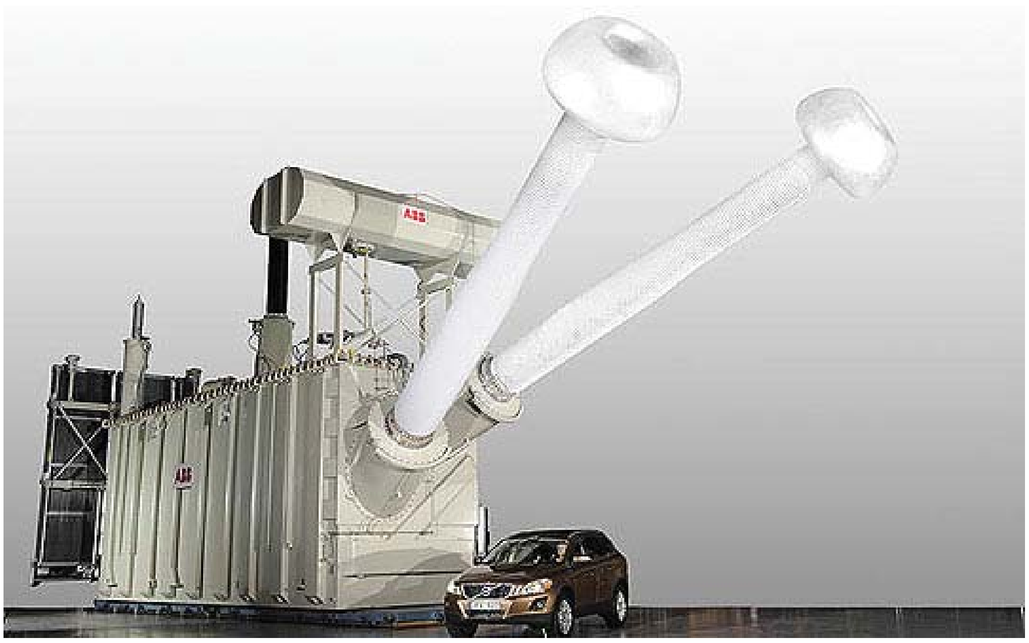


Figure 1.1: Example of a power transformer

A power system is complex and consist of several advanced components such as power transformers, breakers, etc and all of them must collaborate together in order to obtain the desired outcome with a high efficiency.

1.2.1 Smart grid

ABB has increasing demands for better power system solutions that can offer higher capacity, reliability, efficiency and stainability. In order to fulfill this demand, ABB have been considering to integrate their products with a solution called “smart grid”.

This is a power system solution where all the components quickly can be adjusted and optimized in order to minimize the energy loss. This will create a power system which is more stable for disturbances and establish a collaboration between all different components in the system. This means that there will be a system which quickly can adjust depending on the status of the grid [14]. Also considering the fact that there is a ongoing transformation of the power system structure as we know today. New technology enables for smaller companies or even ordinary household to generate energy both for personal use and sometimes even for other household as well.

The smart grid will enable for utilizing the produced energy further and avoid unnecessary losses in the power system. For example there are occasions when the grid is over/underloaded and when this occurs it is suitable to adjust the grid. A overloaded grid means that there is more energy available on the grid than the customers uses and therefore it is better to send the energy to other locations/grids, where it is needed. In order to enable for the use of smart grid, it is necessary with a modern control system that easily can provide the operator with data, both real time and historical. This puts demands on the communication between different parts in the power system and therefore aspects such as flexibility and accessibility are of importance in the control system [12] [14].

1.2.2 Present control system

The present control system consist of two main parts, the control cabinet and the control room which sometimes can be several miles apart, and monitor several important parameters of the power transformer and controls the cooling system. The communication media between the control cabinet and the control room depends on the customer specification but most common are to use several copper wires.

The present control system solution can almost be considered as "hardcoded" in the physical hardware in the control cabinet. For example, the control system that determines when for example the cooling system should be activated, is controlled by mechanical thermometers or relays that monitors parameters like current and temperatures. Modification in the control logic therefore result in a need of travelling to the physical location of the control cabinet and adjust it. Furthermore, every transformer is entailed with respect to the customer specifications, which means that there is a high customization level, see figure 1.2 for a typical control cabinet.

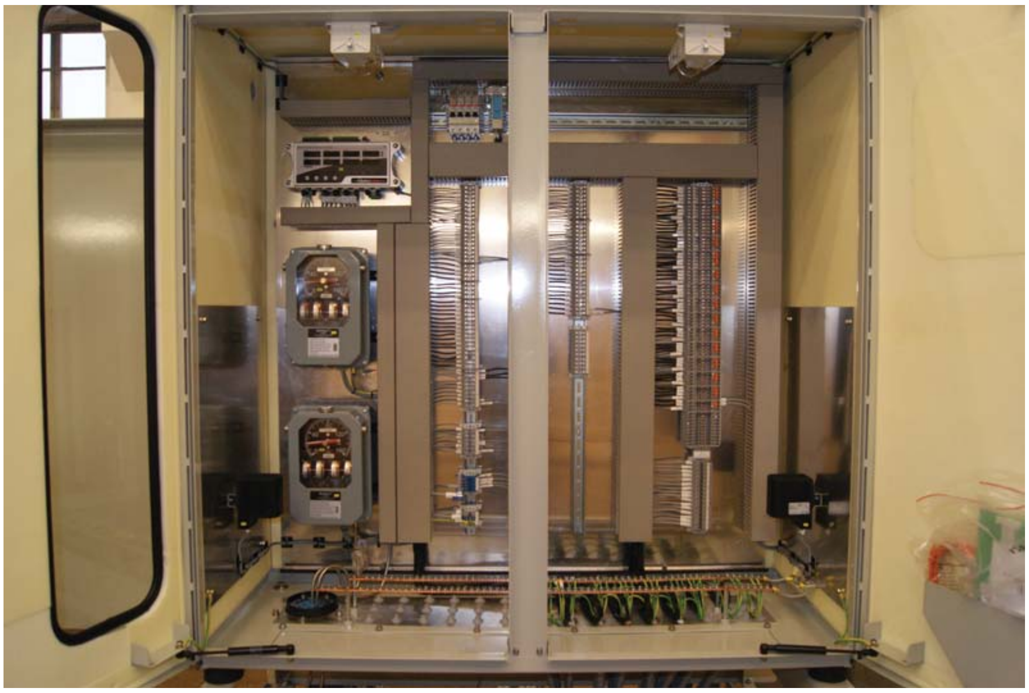


Figure 1.2: Control cabinet ready for delivery, made by ABB

1.2.2.1 Problem area

In order to fulfill all the requirements in the smart grid solution it is necessary to sample both historical and real time data. This data can be used to predict future actions for the control, for example the cooling or discover irregular patterns that indicates a possible error [15]. This means that it is possible to extend the life time of the power transformer since the cooling of the windings will be done earlier. The historical data can also for example be of use in order to have leveled use of the fans located on the transformer. This minimizes the risk of irregular need of maintenance of the cooling system.

One way for obtaining the necessary flexibility and accessibility it is possible to integrate a PLC into the control system. However, due to the importance of the power transformer and price tag it isn't possible to test a new concept on a physical transformer. Therefore it is suitable to use simulation to test and validate the system before the installation on the transformer and by this reduce the cost and time for testing and validation of the control system and also ensure that the solution works correctly. Still one have to consider the high variation level in order to not have a PLC solution that requires a lot of time for the creation of the program.

1.3 Objective

The purpose with this project is to create a concept for the control and monitoring of a power transformer by implementing a PLC, and validate the control logic with a simulation model, and also implement a method for enabling a easy reusing of code from earlier projects and by this enable for auto generation of code.

1.4 Scope

This project will be focusing on the control and monitoring of the power transformer and a code structure that can handle a high customization level during the design of the PLC program and simulation model, in order to reduce developing time. The project aims to enable for reusing code from earlier projects and by this reduce the development time in a new project significantly. The concept of reusing code also seeks for enabling auto generating of code.

1.5 Deliveries

The project aim to deliver the following in the end of the project,

- Easy and understandable PLC logic.
- A solution for easy reuse of PLC and simulation code.
- A clear code structure for easy debugging and error searching
- A concept for controlling and monitoring the cooling equipment, gas, tap changer, leak detection, etc on a power transformer.
- Enable the use of simulation early in a development project for validation.

1.6 Research questions

- Is a PLC a strategic choice for the control and monitoring of a power transformer?
- How can one design the PLC/simulation code structure for increased reusability?

1.7 Contribution

In this thesis a concept for implementing PLC control for a power transformer is presented. This concept have been developed with a code structure that enables reusing old code from earlier projects and by this the creation of a library. The main idea with the library is to reduce the time needed for creating new customized applications which also open up for the possibility of auto generating code.

Simulation have been used in this thesis in order to validate the control logic, that also have been designed with respect for reusability. This does not only enable for fast and easier validation of the control logic but it is also possible to utilize the benefits of simulation earlier in a project and evaluate several different concepts. This results in a possibility of not only reduce the developing time of the control logic and simulation model but also increase the quality due to the fact that the code is well used and tested before.

1.8 Thesis outline

The report have been divided into three main chapters, the first considers PLC system in general, and specifically for the one used in this project and controlling and monitoring a power transformer. chapter 3 considers the structure of the PLC program and how one can enable for easy reusing of PLC code. Chapter 4 contains information regarding the validation of control logic and also how this can be done with respect for reusability. The three chapters are then followed by a conclusion which summarizes this project and suggest future work.

2

Implementing of a programable logic controller in a control system

This chapter contains information regarding the basics of PLC systems, drawbacks and benefits both in general but also specific for the system used in this thesis. Furthermore, the chapter contains information regarding results of implementing a PLC and the user interface that have been designed during this project.

2.1 Programable logical controller system

PLC systems are widely used all around the world for different types of purposes, everything from monitoring and control of traffic lights to heavy machine equipment [1]. A PLC system is usually connected to several other components for example other PLC units, servers, human-machine-interface(HMI), third party systems/componets, etc [3].

PLC developers often offer their customer systems including all these parts, and more, in one complete system. One example of this kind of system is the 800xA developed by ABB which have been used in this project. The 800xA system is used all around the world in different types of applications thanks to its wide application area. One of benefits with using a system such as 800xA is the easy set up of a new control system and the fact that many third party components are already possible to integrate to the system without any extra effort such as Wago, Siemens, etc, see figure 2.1[3].

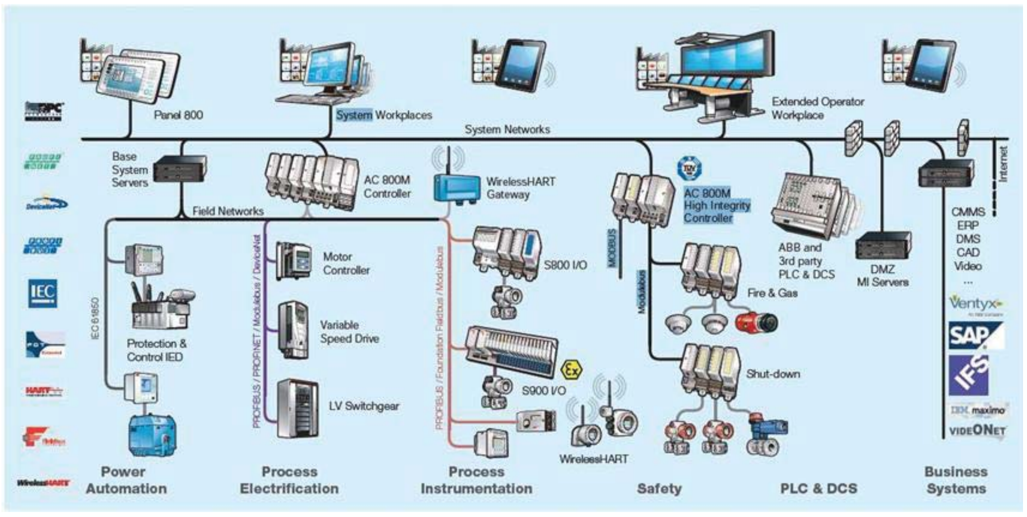


Figure 2.1: Example of a control system structure (800xA)

The communication between the components in the system can be performed in several ways, either following standard protocols such as modbus, profibus, profinet, IEC 61850, etc or adding a own customized protocol. Also it is possible to connect the control system to higher level system via the system network, which also enables for remote access and modification in for example the control logic. This means that it is possible to reduce the need of travelling to the physical location of the PLC to a minimum [18]. However, sometimes it is desirable to have so called stand alone solution which means that a PLC can operate on its own without being connected to other components. If that is the case it is still necessary to travel to the location of the PLC unit if it is necessary to modify the control logic.

2.2 Basics of a PLC

As mentioned before the control logic is downloaded to the PLC. The PLC unit performs a cyclic reading of the program. The cyclic reading means that the PLC read the program row by row and and only in the end of it, or start if it is defined by the engineer, write the values to variables and I/O signals. I/O signals are used for communication with sensors, actuators,etc. The activation of signals is based on the input signals or time events, however how this should be done can be totally customized by the engineer. A engineer can for example define that the PLC should be monitoring different running times for a motor or activate a specific components on a specific day or value [1] [3].

The PLC unit which used in the 800xA system is called AC800m and consist of a CPU unit connected to one or several modules. A module is used for adding functionality to the PLC unit and can for example enabling communication according to a specific protocol or possibility to read or write different types of I/O signals. The AC800m PLC can be seen in figure 3.6 which is connected to a I/O module, on the right, and two communication modules, on the left. The modules are connected to the PLC via an terminal unit. Then, wires are connected to the terminal unit and sensors, actuators, etc. Since there are no direct connection between the modules and wires, it is possible to replace faulty modules without disconnecting the wires from actuators, sensors, terminal unit, etc. This enables for a efficient service of the control system without interrupting the system more than necessary [3].

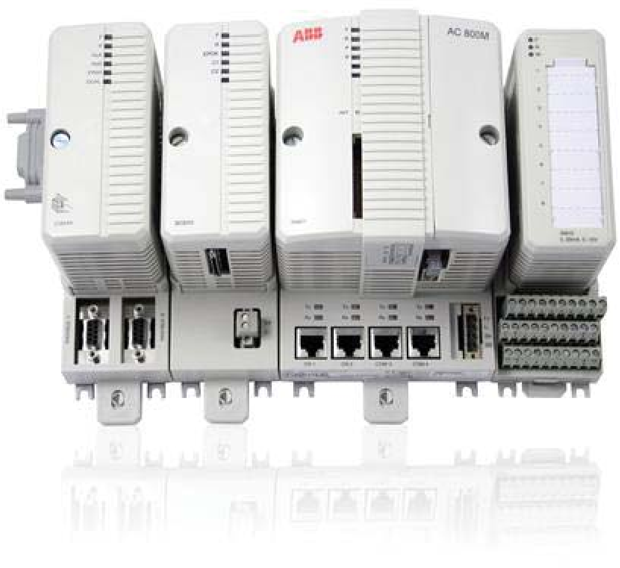


Figure 2.2: PLC with modules

The use of modules also enables for an easy extension of the hardware in the control system. The AC800m PLC allows for 12 modules directly connected to the PLC. When all of these are used one simply add a new "row" of modules. This is achieved by connecting the new "row" via a communication module to the PLC, see figure 3.6. On the new row one can connect further 12 modules, to the control system by defining the address of the modules in the software compact control builder used for programming the AC800m unit, provided by ABB.

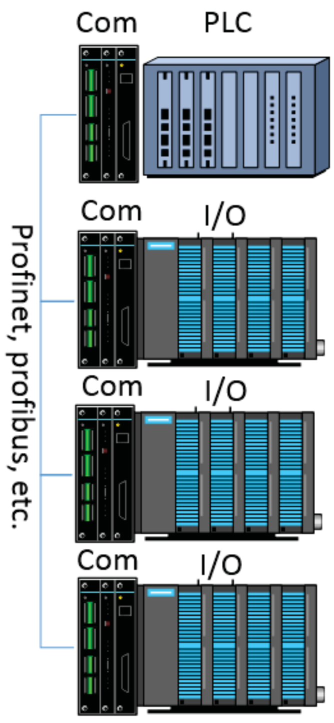


Figure 2.3: PLC with several rows of I/O modules

2.3 Programming language

A PLC program determines the output signals of the control system that can based on several different parameters such as sensors, timers, date/day, counters, etc. The PLC program is written in one of the five languages, used by almost all PLC brands, all described by the industrial standard IEC 61131. These are structured text(ST), sequential function chart(SFC), ladder diagram(LD), function block diagram (FBD) or instruction list (IL). All five can be considered as low level programming langues. Still despite this fact, it is possible to obtain a total customized control logic for every control system and the possibility to monitor both components of the control system and sensors in the system [16] [1] [3]. An example of how the activation of a pump can be defined, see figure 2.4. This example also includes the use of a customized so called function block.

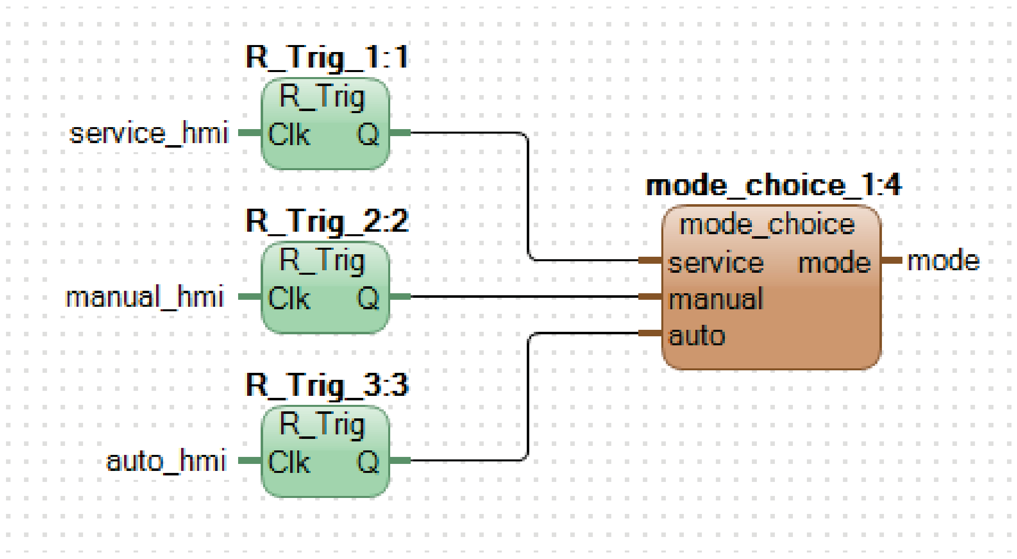


Figure 2.4: FBD code example for choosing mode

A function block contains in and output parameters, see figure 2.4. The output is dependent on the input parameters and the defined logic inside the function block which can be done in any one of the 5 langues, for an example of see figure 2.5. The use of function blocks enables the engineer to save time and effort since the same function can be accomplished simply by importing a new instance of the desired function[3][1]. In figure 2.5 one can notice the use of a data type. This can be considered as a package that contains several variables, for example a simple bool (true or false), arrays, further own customized data types, etc [3].

```
if manual then
   mode.manual:=true;
   mode.service:=false;
   mode.auto:=false;
   TimerReset( service_time);
end_if;

if service then
   mode.manual:=false;
   mode.service:=true;
   mode.auto:=false;
   TimerStart( service_time))
end_if;

if auto or (mode.service and (TimerElapsed( service_tim)>delay)) then
   mode.manual:=false;
   mode.service:=false;
   mode.auto:=true;
   TimerReset( service_time);
end_if;
```

Figure 2.5: Control logic inside the function block seen in figure 2.4

2.4 Implementation of PLC control for improved monitoring and control of a power transformer

In this thesis a concept of implementing PLC control, in order to improve the control system of a power transformer have been tested. The main idea with the design of the system structure in this project have been to demonstrate benefits with a PLC system but also in order to align the control system with the smart grid solution. This means that functionalities such as remote access, historical data, etc have been enabled. The structure of the PLC system used in this thesis can be seen in figure 2.6 and one can notice that only a small part of the 800xA system have been used in this project. Pictures of physical setup of the system can be found in appendix B.

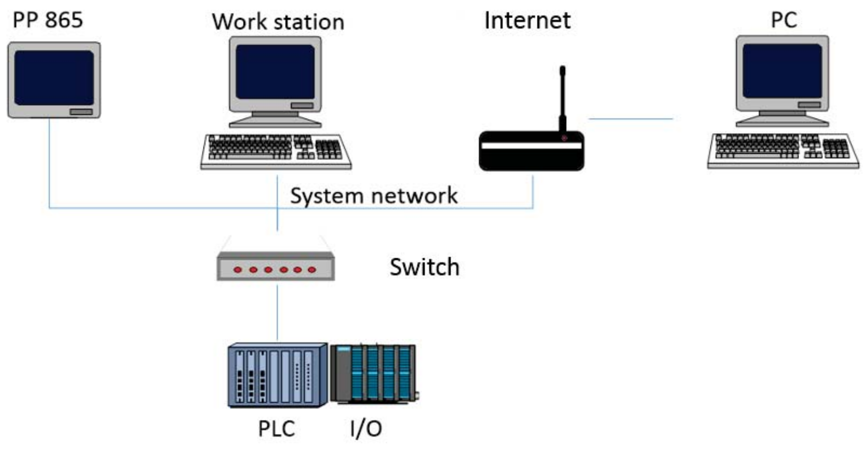


Figure 2.6: PLC system setup

2.4.1 Improved utilizing of the cooling equipment

One can, as mentioned before, measure time continuously for different components in the system. This enables improving the use of the cooling equipment. Utilizing this function it is possible to level the use of for example the pumps and by this prevent the need of service for the pumps to occur on different times.

Furthermore the implementation of a PLC total customization of the control logic. This can for example enable for individual control of the cooling equipment. The cooling equipment is usually divided into several groups consisting of for example 1 pump and 4 fans. In the previous control system a group is fully activated when there are actually maybe need of only the pump and one fan. This means that 3 fans are running unnecessarily and can therefore be considered as waste not only in energy but in running time for service as well. Using the individual control it is possible to extend the time when there are need of service of the cooling equipment and also when it is necessary to activate the exercising running of the different fans.

2.4.2 Improved error detections and service supervision

The implementation of PLC control enables for increasing the assistance for the operator during stressful circumstances when searching or fixing errors in the control system. This could result in a reduced time needed for service or the startup phase. The PLC uses modules that can both receive and send I/O signals and with the 800xA system it is possible to detect if a module is broken, receiving faulty signals or if it is replaced with the wrong type. All this enables for a system that directly detects errors and its exact location and send alarm to the operator, in this case even with article number, type of module/sensor, etc. The main idea have been to provide the operator with all the necessary information for locating the error and if needed order new parts.

The control system can also determine when different parts on the power transformer is need of service thanks to a implementation of this function in the control logic. The PLC constant measures active time for all the components on the power transformer. For example the cooling system which uses electric fans need service within a specified time interval. The PLC sends a signal exactly when it is required to start considering for service on a specific component. This enables for allocating the service event during a planned stop of the power transformer and avoid unnecessary cost due to down powering the station.

Furthermore the PLC can monitor several signals which for example indicates whether fan is started or not. This can for example be utilized in order to secure the cooling of the power transformer when one fan breaks down. If a active fan breaks down, the signal is lost and the PLC can send a alarm is sent to the operator and then a new group/fan is started based on active running time. Also one can define that the PLC should send a alarm which alerts the operator when there are no fans available but there are need of further cooling capacity. This kind of information can then be used in order to decide whether it is worth the cost of shortened life time by running the power transformer at the same load or if one should considering to lower the load.

2.4.3 Improved communication and accessability

At the present control system it is sometimes necessary with several copper wires for establishing a communication between the control cabinet and control room. The implementation of the system 800xA reduces the amount of cables needed since it is possible to transfer information via communication protocols. This means that it is possible to save money and time needed for communication during design, installation and error searching. Also the possibility of connecting the control system to a networks opens for the possibility of remote access, control and modification. This means for example that it is possible for the operator to access the control system via any device with a internet connection, for example smart phone, tablet, laptop, etc. This also enables for a future collaboration with the smart grid solution, since it it possible to provide both real time and historical data.

2.5 Visualization of data and control functions

One of the most important purposes with a PLC systems, except for the control and monitoring, is to present information for the operator. However this isn't done by the PLC itself but rather a HMI system who is designed for this purpose. The PLC sends the information via one of the communication protocols to the HMI which presents the data for the operator. Furthermore it is possible to send commands to the PLC, for example triggering a variable which starts for example a pump. However, the media which is used for visualizing the HMI can vary a lot depending of the customers specification. One can for example use a HMI based on a server which you connect from a ordinary computer using a client or a HTML based interface located on a webserver [1] [3].

During this project the touch panel PP865, included in the 800xA system, have been used. This allows for easy access to the control system with no need of a computer. Furthermore the PP865 enables several other functions such as sampling of data, webserver, vnc server, alarm manager, report generating, email/sms functions, PDF viewer, etc. The PP865 panel is programmed using the software Panel Builder 800 which enables for easy changes and creation of graphic used for visualizing data or control functions for the operator, see figure 2.7 for the working environment in the software. The communication between the PP865 and PLC is performed using ABB's communication protocol called control network which is based on the MMS protocol [3]. Furthermore it is possible to connect the PP865 to the system network. See figure 2.1, which allows for remote modifications in the HMI. Also it is possible to remotely access the HMI via either a VNC client or the web page located on the webserver but it requires that the user is connected to the same network either directly or via VPN.

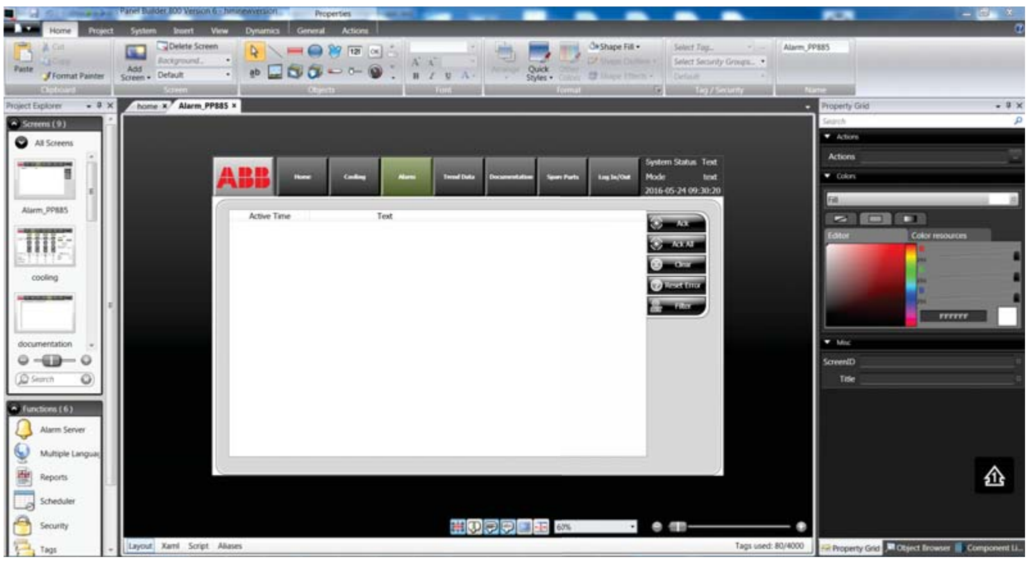


Figure 2.7: Snapshot from Panel Builder 800

The design of the HMI have is to be as easy and fast as possible to use for the operator. One reason to this is that when a alarm goes off it should be easy for the operator to find the needed information or functions fast. In order to achieve this, the information have been divided into several menu's. The start screen can be seen in figure 2.8 and this have been designed to provide the operator with all the necessary information in order to interpret the status of the power transformer. Furthermore due to possibility of easily modify the HMI for different purposes, one can entail the HMI for each customer. For example a picture of the power transformer of each project which can be used to visualize the exact location of for example a faulty pump or adding extra information to the start screen.

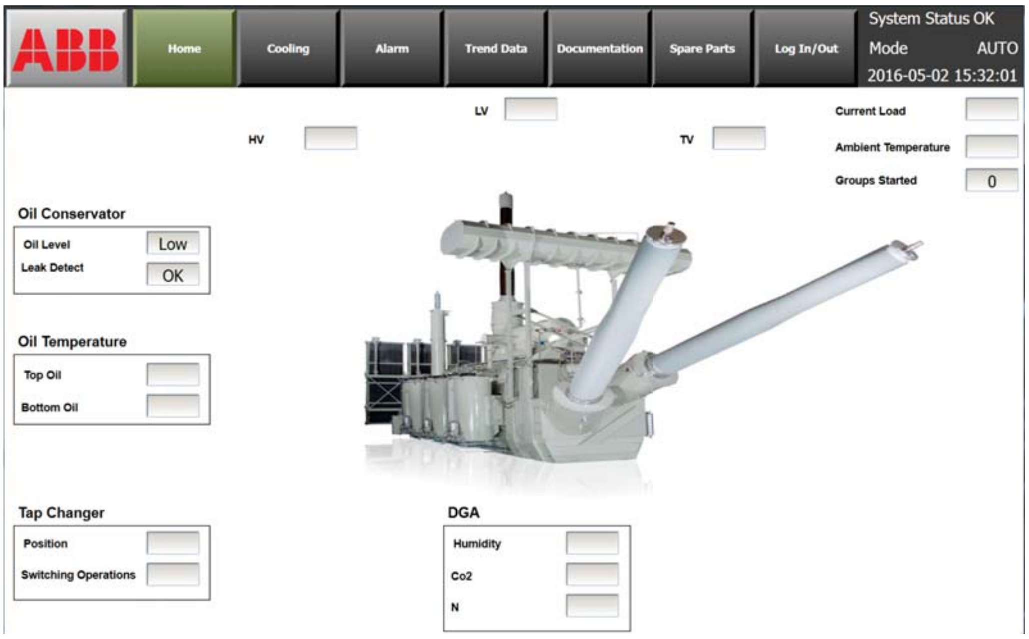
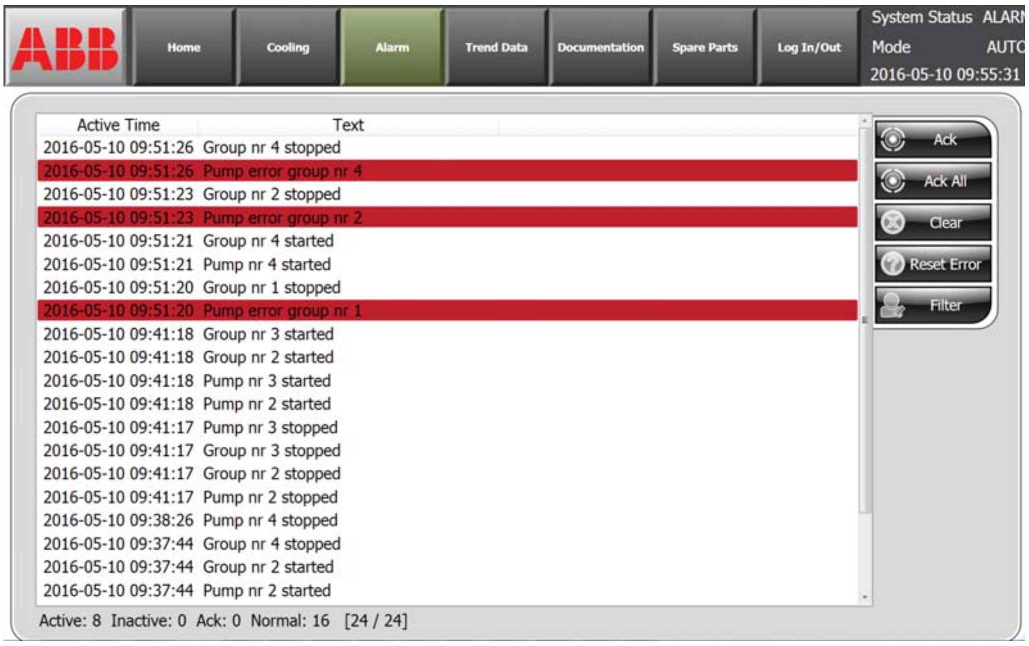


Figure 2.8: Start screen

The alarm manager located in the HMI, which monitors variables in the PLC. When one of them are triggered the HMI sends a alarm. In the alarm manager it is possible to connect the alarm to a text which can be used to describe the error including information such as article number, location, etc. see figure 2.9 for the alarm screen.



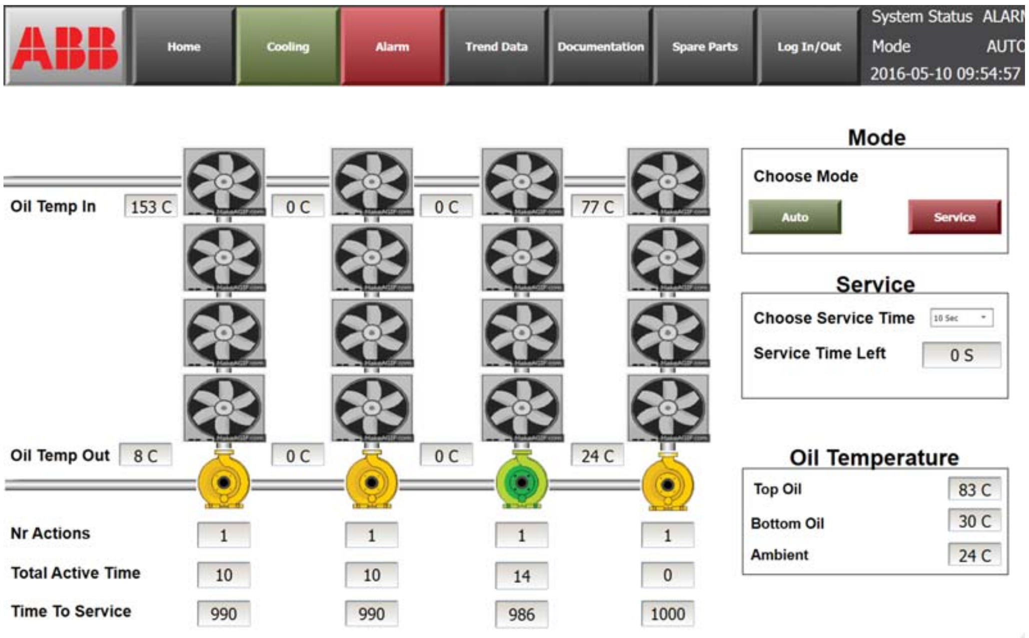
System Status ALAR
Mode AUTO
2016-05-10 09:55:31

Active Time	Text
2016-05-10 09:51:26	Group nr 4 stopped
2016-05-10 09:51:26	Pump error group nr 4
2016-05-10 09:51:23	Group nr 2 stopped
2016-05-10 09:51:23	Pump error group nr 2
2016-05-10 09:51:21	Group nr 4 started
2016-05-10 09:51:21	Pump nr 4 started
2016-05-10 09:51:20	Group nr 1 stopped
2016-05-10 09:51:20	Pump error group nr 1
2016-05-10 09:41:18	Group nr 3 started
2016-05-10 09:41:18	Group nr 2 started
2016-05-10 09:41:18	Pump nr 3 started
2016-05-10 09:41:18	Pump nr 2 started
2016-05-10 09:41:17	Pump nr 3 stopped
2016-05-10 09:41:17	Group nr 3 stopped
2016-05-10 09:41:17	Group nr 2 stopped
2016-05-10 09:41:17	Pump nr 2 stopped
2016-05-10 09:38:26	Pump nr 4 stopped
2016-05-10 09:37:44	Group nr 4 stopped
2016-05-10 09:37:44	Group nr 2 started
2016-05-10 09:37:44	Pump nr 2 started

Active: 8 Inactive: 0 Ack: 0 Normal: 16 [24 / 24]

Figure 2.9: Alarm screen

It is possible to both monitor the cooling equipment with detailed information, see figure 2.10, and control it as well. However in order to activate any of the functions in the HMI it is required that operator log in as a user which enables specified permissions. In the cooling control screen one can easy see whether a pump is running or a fan is activated, since for example the fans is GIF images which rotates when activated. Also in order to visualize if there are any errors, the images change color. As it can be seen in figure 2.10, where pump 1,2 and 4 have failed but 3 is running.



System Status ALAR
Mode AUTO
2016-05-10 09:54:57

Oil Temp In	153 C	0 C	0 C	77 C
Oil Temp Out	8 C	0 C	0 C	24 C
Nr Actions	1	1	1	1
Total Active Time	10	10	14	0
Time To Service	990	990	986	1000

Choose Mode

Choose Service Time 10 Sec
Service Time Left 0 S

Top Oil	83 C
Bottom Oil	30 C
Ambient	24 C

Figure 2.10: Cooling screen

Several test runs have been performed with designed HMI in order to receive feedback. During these runs it didn't take long before the "operator" could operate the control system without help and the HMI have only been receiving positive feedback.

2.6 Drawbacks and benefits with the implementation of a PLC in a control system

As always there are drawbacks with connecting a system to the internet, for example the possibility of someone attacking the system and succeeds with overriding the system and obtain the control of it. If this occurs it is possible to overheat the transformer and accelerate the ageing which result in a shorter life time. However, since the 800xA system is continuously being updated and improved by ABB, there are modern countermeasures available for this concern and therefore it is possible to protect the control from unauthorized control. The continuous improvement also applies for the hardware and software which is used for programming the system. This means that it is possible to utilize the benefits with newest technology that ABB implements in the 800xA system. Also the fact that a big company as ABB is the one responsible for the system, encourages many other smaller companies to adapt their products for easy implementation in the system.

However one have to consider whether it is worth the cost of adding a PLC to a control system. For example some of the designed power transformers doesn't require advanced cooling control, possibility of remote control/modification or storing of historical data. If that is the case, the implementation of a PLC may not be needed, rather the opposite due to extra cost and work if one should be using the AC800m PLC unit. But it is clear that for bigger transformers that requires these functions it is beneficial with the use of a PLC. There also are cheaper "budget" variants of PLC's available on the market, that could be used instead of the AC800m but comes on the cost of functionality and performance. Therefore it is possible to use a PLC even for smaller transformers and enable some of the benefits with a PLC, for example historical and real time data, visualization, remote access, etc.

Also since the PLC is consisting of different types of modules it is possible to add more functionality and hardware. Furthermore it is easily integrate a new system with a previous control system and avoid adding a extra stand alone system. Also the fast service of the control system is useful when components have to be replaced in the control system. There are occasions when there are several companies delivering either complete power system or parts of it to a station. If the last scenario is the case, it is important with the possibility of integrating the control system with a third party system.

The possibility of customizing the control logic for every power transformer gives a clear advantage both technically and in a selling perspective. This means that it is possible to entail for example the cooling conditions and how, when and why a group should start. All this can be done according to customer specification and own experience in order to get the best cooling strategy. Also if the design of the hardware is considered, it is possible to apply comprehensive modifications in the control logic without re-manufacture the control cabinet. This means that the implementation of a PLC enables for flexility in the control system not only during the developing of the control logic but also when it has been finished. Furthermore the possibility of easily apply changes in the control logic encourage for improvements and bug-fixes in the control system even after the delivery to customer. Therefore when new progress are made in the cooling strategy area it is possible to implement this in both old and new control system and ensuring that each power transformer always is using the best strategy available. However before it is possible to start up a PLC system it is necessary with a PLC program which have to be developed to each type of transformer, due to the high variation which demands a code structure that supports reusability and flexibility.

3
Design of control logic with respect for reusability of previous knowledge

A high variation regarding the specification of the control system for a power transformer requires a code structure that contributes with flexibility and reusability. How this can be achieved will be discussed in this chapter, together with relevant research within this topic. Two possible concepts which can be used for enabling reusing of code within this sector will also be presented.

3.1 Background theory

In order to handle the increasing demands on time and quality during development of control system one should start to consider the possibility of enabling for reusing of code. One method is to start using a library which contains code from earlier projects, that can be imported to a project when needed. It is obvious that reusing code is more efficient in terms of time, compared to starting every project from scratch and entail the entire project to one system. The purpose with a library is to allow the engineer to easily reuse previous created code, components, functions, etc if it is needed in the new application. This requires a structure and design of the components in the library with respect for this purpose. Variation is a major problem within the industrial automation sector, obvious, because without variation it would be possible to use the same code over and over again. Variation result in a need of adding extra code in order to obtain the desired control logic and behavior in the system [16].

Adding extra code to a project/library, based on components from earlier projects, there are several factors to take into account, for example so called cross cutting concerns. These concerns are factors/functions that occurs in multiple locations in the program which complicates both the understanding and reusing of code [11]. This means that one modification one location of a application might have other unwanted affects at other locations. However adapting the structure and design of the control logic for this purpose it is possible to ease the reusing of code in a company and reduce the required time for developing the control logic significantly.

However when designing a PLC program it is preferable to have a clear picture of the needed functionalities. Often is the desired logic described in text in the specification, that can sometimes be hard to fully understand due to its abstractness. In order to ease the understanding it is possible to use so called machine states.

3.2 Machine state

The first step during a creation of control logic is to determine different types of functions/parts in the system, possible operations, how they should be triggered and which order[17]. In order to visualize these aspects, in a pedagogical way it is possible to use a so called "machine state", see figure 3.1. A machine state describes the events that occurs when a specified signal is triggered and how they affect each other. This helps the engineer to get a good overview of the system and different functions. The machine state consist of in and out signals, that is connected to either IO signals or other machine states, and the different events dependent on these signals.

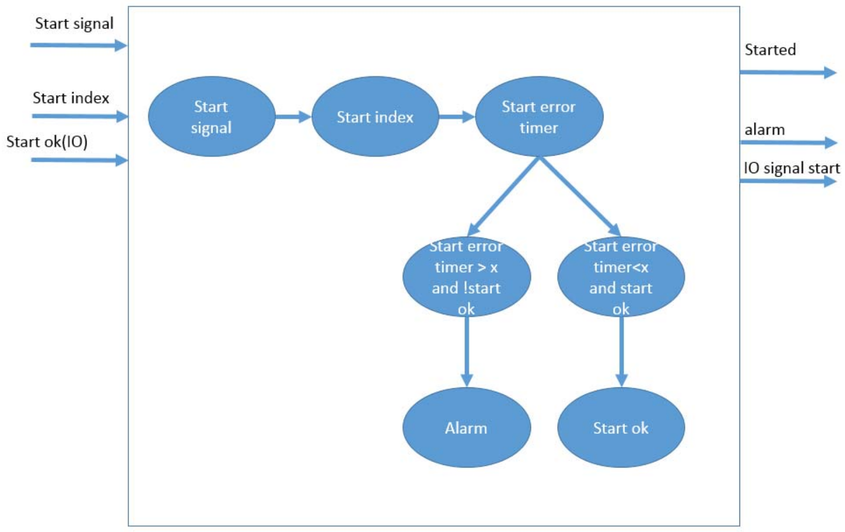


Figure 3.1: Example, machine state

A example of a machine state can be seen in figure 3.1. This machine state describes the part of the application determines which cooling group to start, in this case based on active time, and sends the signal to the next part in the program which executes the start. However as it can be seen there are two possible scenarios during this event, for example if there are no group left to start an alarms should be sent to the operator and if there are groups available then this parts should send the number to the next part of the program.

Even though the visualization with the machine states ease the creation of control logic, it is still possible that a application contains errors due for example human factor, etc. With a machine states it is possible to get an better perspective and understanding of the different parts when programming the control logic or designing the structure of the program.

3.3 Traditional vs object oriented programming

Before starting to programming it is necessary to determining the structure of the PLC program, that have a major influence on the reusability. The design of the PLC program can be very diverse from company to company, both in terms of structure and choice of language. It is common that companies uses their own standards, developed within the firm but there are also cases where no standards are used at all. No standards means that the application has to be created from scratch, both in terms of structure and previous defined functions. Defining a structure of a the PLC code requires that the engineer considerers the control logic in a wide perspective, that often requires several modifications during the development[2] [3].

For the programming of the AC800m there are two approaches, traditional and object oriented programming. The choice depends on the complexity of the needed control logic. For a system with low complexity it is preferable to utilize the traditional programming methodology and vice versa when one should use the object programming. Still worth mentioning is that the features that that defines the object oriented programming approach isn't fully implemented in the PLC units available today [3]. For example the possibility of inherit features are not possible, but the use of some features of object programming is slowly started to be seen in developing softwares for PLC's[19].

The traditional programming utilizes ordinary programs (POU) and functions blocks and is suitable for easier PLC programs. Still using the traditional programming contributes with several drawbacks in terms of reusability. For example a higher use of global variables for the communication between different applications in the program which complicates the reusing, see figure 3.2.

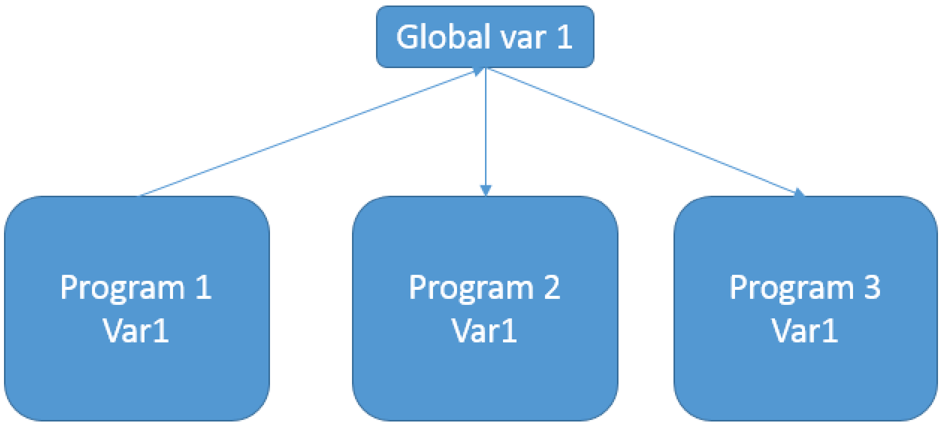


Figure 3.2: Example of structure for traditional programming

Therefore in order to ease the reusing, it is suitable to use the object oriented approach. This can enable for easier encapsulation of different parts in the program. In the programming of AC800m it is possible in the software Compact Control Builder to utilize control modules. A control module can be considered as a super function block which can contains defined control logic with possibility of adding graphic for control, service, etc purposes. It is also possible to combine the use of control modules and function blocks depending on the complexity of one part of the program. Still even though the appearance of the control module is similar as the function block it is to be considered as a small program. Using control modules enables several benefits in terms of reusing, for example it is possible to create the modules in a hierarchy structure and reduce the amount of variables needed in the program[3]. The decrease of variables is possible since one can connect modules directly to each other, see figure 3.3.

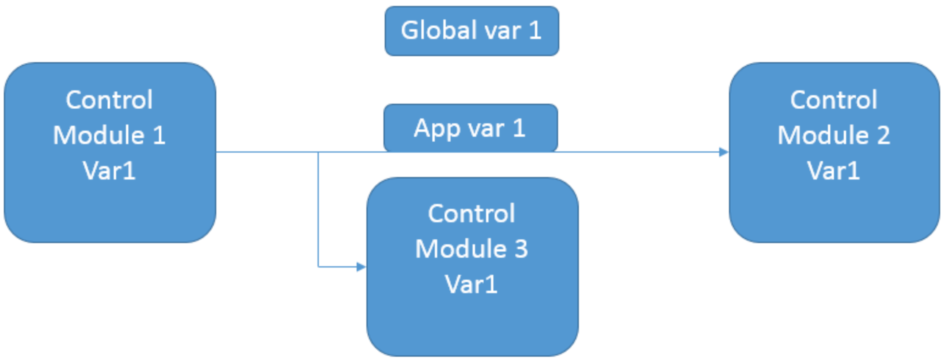


Figure 3.3: Example of structure for object oriented programming(control modules)

3.4 Encapsulation and inheritance of functions in a application

In order to be able to handle the variation it is important to have a flexible code structure for easy and fast changes. This is possible to obtain with several approaches, one is to encapsulate code. This approach, builds the application in different parts which enables for easy modifications and reuse of earlier code. This reduces the time needed for development between different system significantly since one doesn't have to know exactly what happens inside the encapsulated part but rather understand what has happened.

In order to ease the reusing of code from previous projects it is possible to encapsulate functions/programs into several smaller parts. In practice this means that the engineer considers the structure of the program and how it should be designed in order to enable this. One approach as Pineda-Sanchez describes, is to encapsulate for each type of component in a control system and by this build the application using objects of these [20] [19]. For example a relay, certain cable, motor, etc. The encapsulation of program code isn't only limited to hardware in the control system but parts/functions in the control logic are also possible to encapsulate, see figure 3.4.

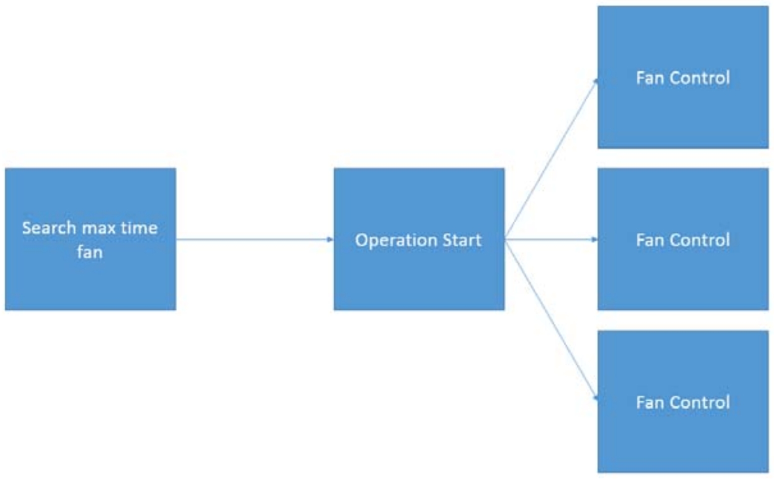


Figure 3.4: Example, encapsulation

Unfortunately in the approach used by Pineda-sanchez utilizes templates for each type, which is copied and paste into the project. Then, one have to manually add the desired control logic in the instance, containing the encapsulated part of the program, since it isn't fully possible to utilize inheritance in the PLC's. However, the code structure for a basic components such as a relay could be similar as in figure 3.16 based on Pineda-sanches method[20] which utilizes the basics of inheritance and encapsulation.

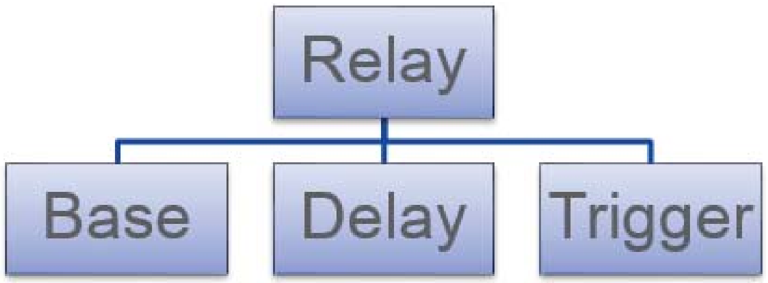


Figure 3.5: Basic of inheritance

The design of the code structure is a time consuming task but there are also other aspects to consider during this phase of a project such as the need of storing data globally [3]. Global variables can for example be used for connecting separated parts which complicates the understanding of the execution order between thee different parts. This could lead to unwanted behavior when changes are applied to one of the parts. Also one have to consider that the execution order of different parts in a PLC program may differ between PLC developers even though this is described by the IEC 61131-3 [7].

The main idea with the encapsulation and inheritance is to enable for easy modification in the control logic and change the behavior without reprogramming the whole application. Also the possibility of reusing code from earlier projects in order to save time and effort. The engineer can swap one part of the program for another or add complete functions to the PLC program and radically affect the outcome in the control system [16]. However the flexibility in the encapsulated part of the program is limited and in order to handle variation it is necessary to add a new type to the library even for minor differences.

3.5 Encapsulation in several layers

In this project, a concept for utilizing the benefits with encapsulation, inheritance(similar as Pineda-Sanchez) and control modules been tested in order to get a flexible and reusable code structure. In order to achieve this, the structure of the code have been considered from scratch and implemented in the new control system proposed in this project for the control and monitoring of the power transformer. However, as mentioned before, even though the encapsulation enables for easy modifications it is limited in terms of variation. The reason to this is that it is necessary to add extra types in the library for small variation. In order to handle this phenomena it is possible to encapsulate the PLC program into several levels,see figure 3.6, possible to obtain with the use of control modules.

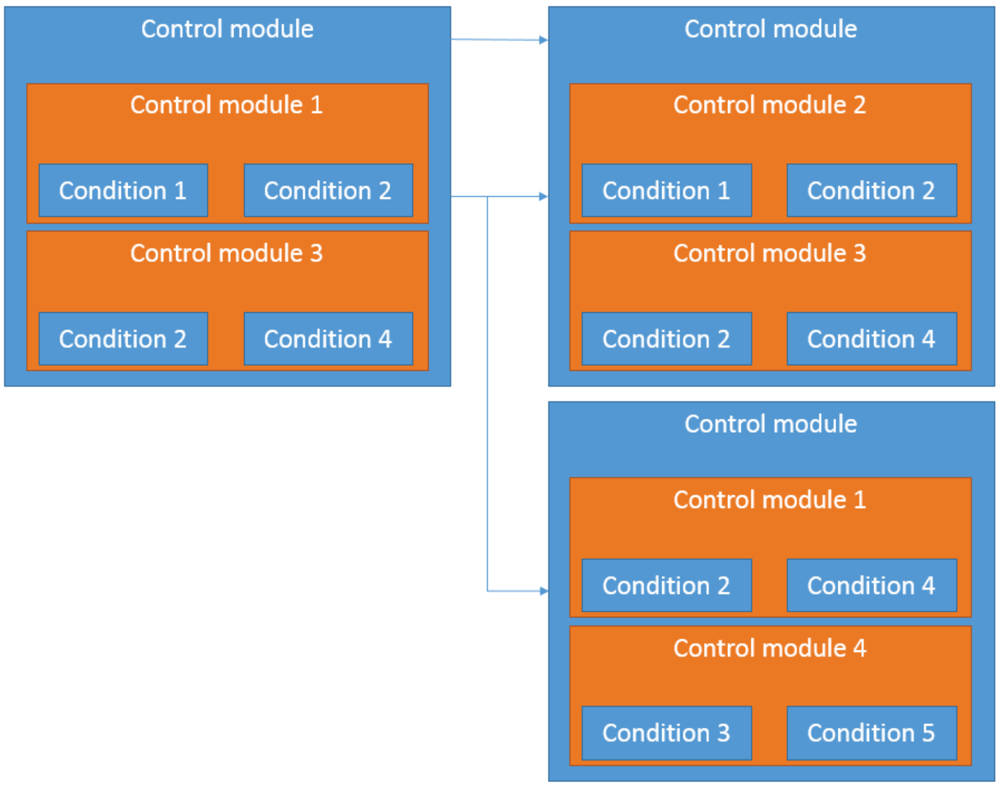


Figure 3.6: encapsulation and "inheritance"

This enables for the possibility of modifying the control logic on a lower level. Comparing this to the encapsulation with only using one level this gives the engineer a clear structure of the PLC program which is beneficial when debugging the PLC program. The reason to this is that it is possible to easily find the part which is most likely to be incorrectly and focus the debugging on this part. An example of this is the control of the cooling equipment. The first level of encapsulation can be performed as in figure 3.7.

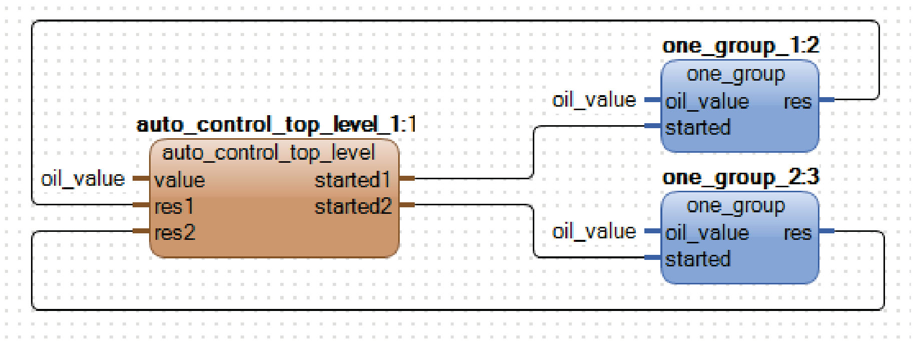


Figure 3.7: Example, encapsulation level 1

The part that determines when a group should be started has been encapsulated into one part. The part of the program that handles the control of the cooling for one group have been encapsulated into another part, including alarms, timers, etc.

This allows for easy adding of groups to the control system since one simple can copy and paste a group and add it. However, the amount of for example fans varies, in order to avoiding adding a extra type to the library have this part been divided into several different levels. This increases the possibility of adding or removing functionality for each group only by using components from the library, see figure 3.6.

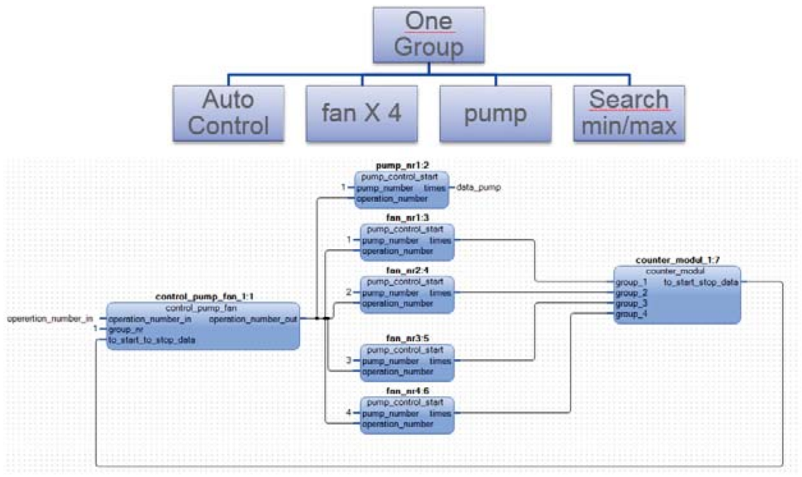


Figure 3.8: Example, encapsulation level 2

The dividing of the group control have now been divided into three subparts. The first part listens to the start signals, from the higher level, and sends it to the for activation of the group which have minimum active time. The last part receives data from the groups considering status, running times, etc. This part then processes this and determines which groups should be started or stopped next time the first part is triggered. In this case it is possible to divide the second part further, see figure 3.9. However at this level it is possible to add extra fans by simply copy and paste the pump control instance. Also it is possible to entail this different for each group, for example one group maybe only have 2 fans and another group have 4.

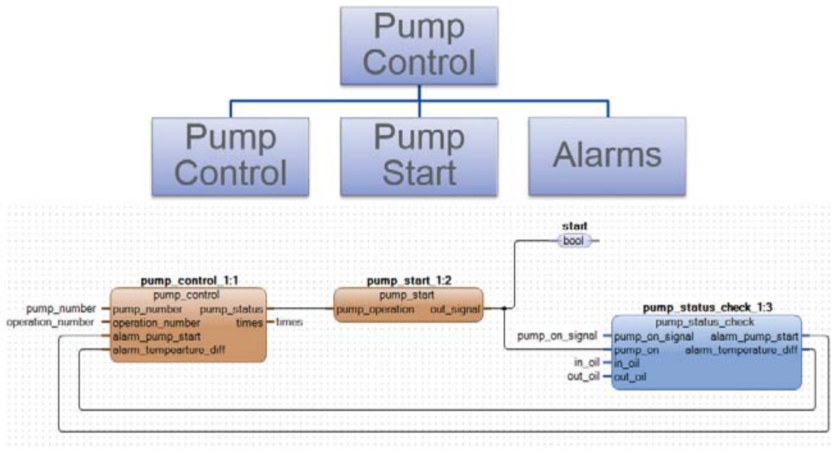


Figure 3.9: Example, encapsulation level 3

This part have been divided to 3 additional parts. The first parts listen to the operation number and determines if it is turn to start or stop. The second part activates the fan/pump and then the last part considers all the alarms which is connected to this part of the program. However in order to enable for easy modification have the first and last part been encapsulated to further subparts, while the pump_start part is down to such a basic level that further encapsulation is unnecessary. The encapsulation of for example the alarm part enables for adding or removing different types of alarm easily and is the reason why this have been encapsulated to further subparts, see figure 3.10.

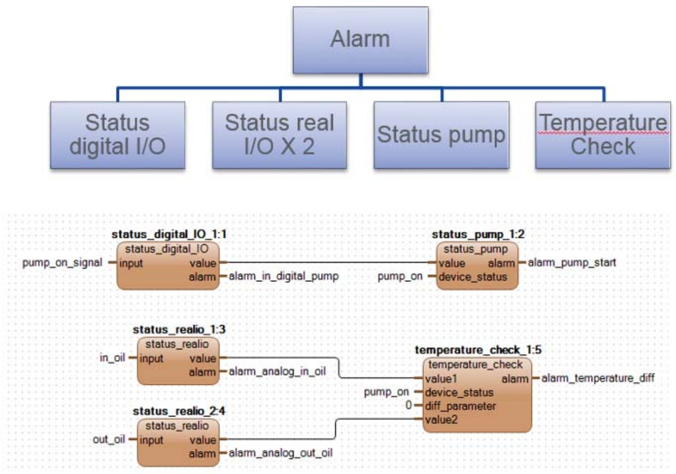


Figure 3.10: Basic of encapsulation level 4

As one can understand this structure enables for modifications with the same approach as in the first encapsulation, in figure 3.7 but with higher flexibility and the need of adding extra types have been reduced since the PLC program consist of basic components on lower levels. This means that adding extra functionality to the control system can be done rapidly and easy. Also in order to ease the building of the application have data types been used as input and output parameters which only enables some of the components in the library to be used together.

The use of instances/objects for each group enables, as mentioned, for customization between the different groups using the same code with the possibility of adding/removing features. For example when a group consist of a higher amount of fans it is desirable to start one by one in order to avoid voltage drop or unnecessary high current during the start of multiple fans. This is done by adding this functionality to the encapsulated part for the triggering of fans, seen in figure 3.9. Since this part is located in one location of the program one can go down one level and add a this functionality.A better overview of the included functions in the PLC program can be found in figure 3.11.

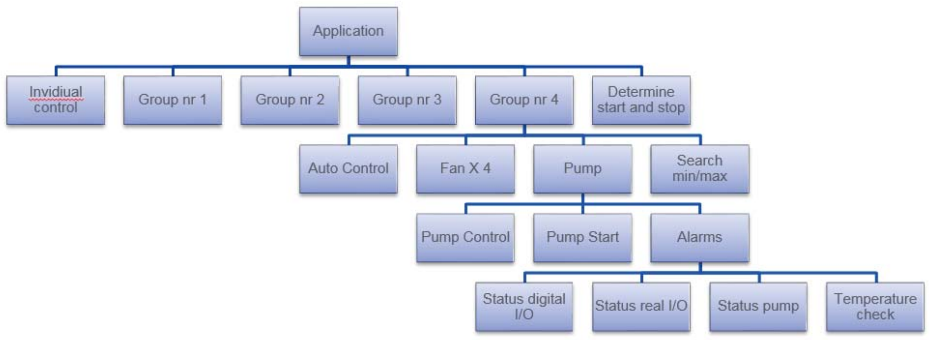


Figure 3.11: Overview of PLC program

The use of levels doesn't only contribute with a easy modification of the PLC program but also eases the debugging of the PLC program when searching for errors. Comparing the approach with several levels to one with only one level, as in figure 3.12 it is clear that the debugging is easier with multiple levels. One reason to this is that one can focus the debugging on the parts that the error is most likely to be located.

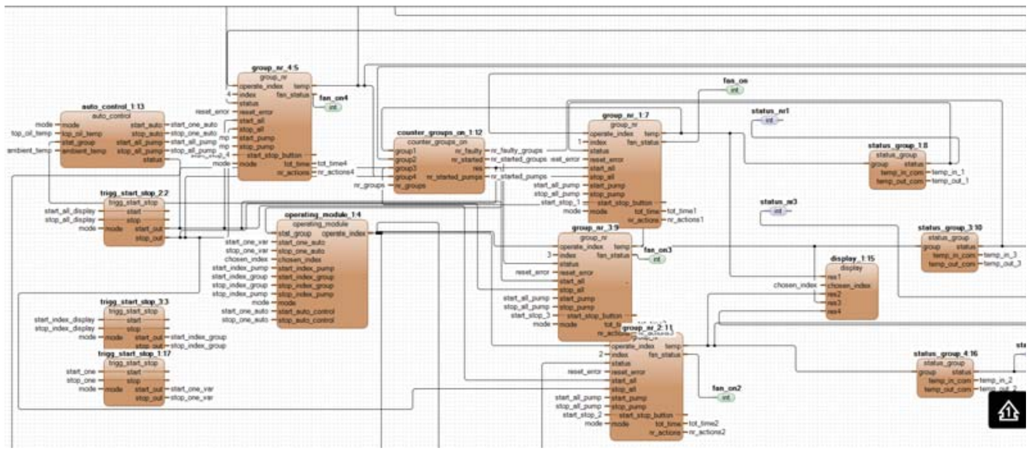


Figure 3.12: Encapsulation with one level

3.5.1 Adding extra functionality

Adding extra functionality is done in two steps, assumed that the necessary logic is already existing. The first step is to add the control module for example a extra fan including alarms, start conditions, etc. This can be done by simply copy and paste a other one in the project see figure 3.13.

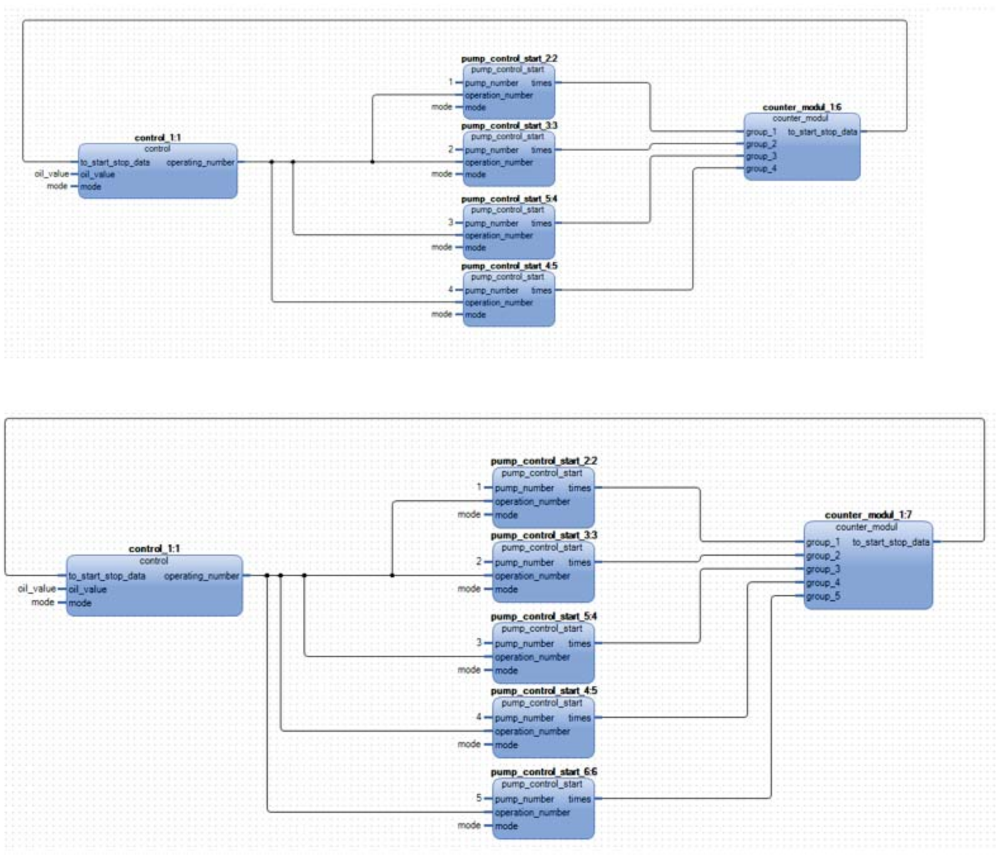
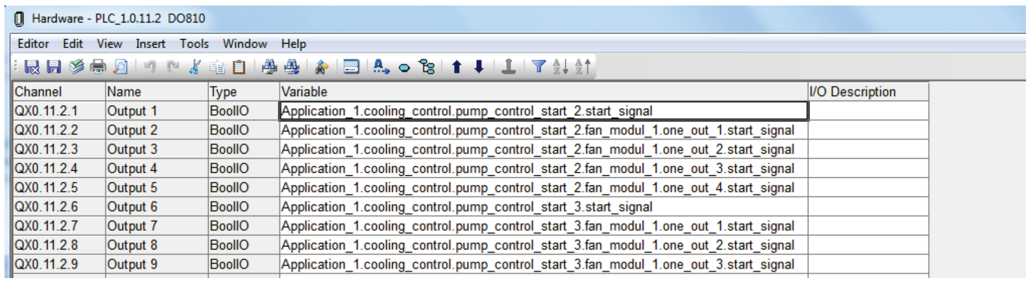


Figure 3.13: Adding a extra fan/pump

The second step is to "map" the start signal in the control module to the I/O signal. This done by specifying the address of the start variable in the control module in the "Hard ware" editor in the software. See figure 3.14.


Hardware - PLC_1.0.11.2 DO810

Editor Edit View Insert Tools Window Help

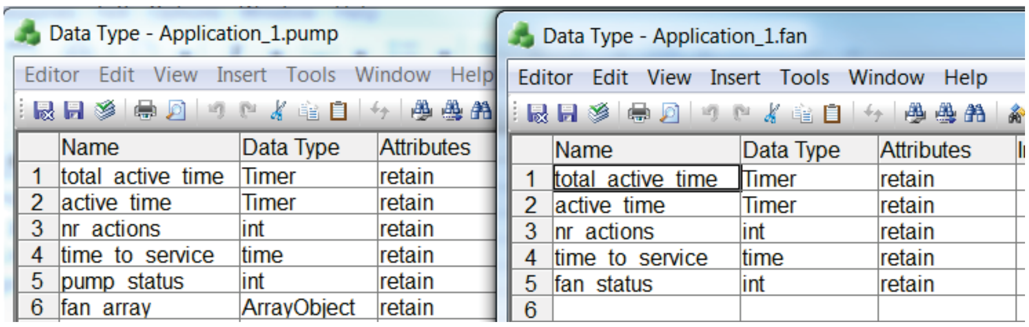
Channel	Name	Type	Variable	I/O Description
QX0.11.2.1	Output 1	BoolIO	Application_1.cooling_control.pump_control_start_2.start_signal	
QX0.11.2.2	Output 2	BoolIO	Application_1.cooling_control.pump_control_start_2.fan_modul_1.one_out_1.start_signal	
QX0.11.2.3	Output 3	BoolIO	Application_1.cooling_control.pump_control_start_2.fan_modul_1.one_out_2.start_signal	
QX0.11.2.4	Output 4	BoolIO	Application_1.cooling_control.pump_control_start_2.fan_modul_1.one_out_3.start_signal	
QX0.11.2.5	Output 5	BoolIO	Application_1.cooling_control.pump_control_start_2.fan_modul_1.one_out_4.start_signal	
QX0.11.2.6	Output 6	BoolIO	Application_1.cooling_control.pump_control_start_3.start_signal	
QX0.11.2.7	Output 7	BoolIO	Application_1.cooling_control.pump_control_start_3.fan_modul_1.one_out_1.start_signal	
QX0.11.2.8	Output 8	BoolIO	Application_1.cooling_control.pump_control_start_3.fan_modul_1.one_out_2.start_signal	
QX0.11.2.9	Output 9	BoolIO	Application_1.cooling_control.pump_control_start_3.fan_modul_1.one_out_3.start_signal	

Figure 3.14: Mapping of I/O signal

Still it is necessary that the engineer imports and connect the different parts of the control logic for different groups of cooling equipment, every new type of installation needs new developed programs. This requires time and effort especially if the comments in the reused code isn't sufficient enough in terms of understanding.

3.6 Encapsulation using objects

In order to improve the code structures ability to handle variation, a second approach was evaluated. This is based on the use of objects together with encapsulation and control modules which allows for reusing the same code without any significant modification. In order to enable this the project have been utilizing data types. For example two data type have been created for the cooling equipment, see figure 3.15.


Data Type - Application_1.pump
Editor Edit View Insert Tools Window Help

	Name	Data Type	Attributes
1	total active time	Timer	retain
2	active time	Timer	retain
3	nr actions	int	retain
4	time to service	time	retain
5	pump status	int	retain
6	fan array	ArrayObject	retain

Data Type - Application_1.fan
Editor Edit View Insert Tools Window Help

	Name	Data Type	Attributes
1	total active time	Timer	retain
2	active time	Timer	retain
3	nr actions	int	retain
4	time to service	time	retain
5	fan status	int	retain
6			

Figure 3.15: Data type for a fan.

The data types is stored in a array which contains information regarding the status of all the cooling equipment. Each pump is stored separately in the array, and include one more array with data types of the fans, and values that the data type contains in each spot of the array can be modified independent of each other. These object, stored in the array, are then sent to different parts of the program in order to determine for example which group to start. The first spot of the array is dedicated as a operation slot, for example which fan/pump to start/stop.

This enables for easy adding of extra functionality, such as a extra fan, by utilizing the same approach described in section 3.5 one can start changing different part of the control logic easily. The different encapsulated parts are connected to by sending the array of object to each other, see figure 3.16. This allows for adding extra parts to the PLC program by simply connecting it to one part and send the array to the next part of the program. Also it is possible to remove unnecessary functionality and without considering the whole application. Furthermore one can add parts to the PLC program that for example only considers one of the fans by sending the array and extract the object and export, it to the array after use if any changes are done, and send it to the next part of the program. This allows for adding customized code for each object. It is important that one consider the order of the parts included in the program since the execution order may have an influence of the control logic.

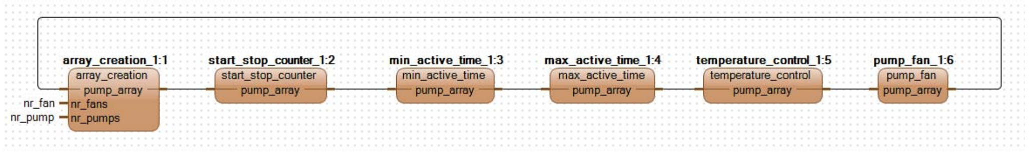


Figure 3.16: Example of application utilizing objects

Also, in this approach it was chosen to utilize for-loops for avoiding modifying parts dependent on the amount of for example fans. This creates a code structure that has embedded logic for adapting for variation in the control logic. However worth mentioning is that utilizing for-loops, objects, etc in a PLC program contributes with a complex debugging and also adds a possibility of entering a deadlock. But the power of using a for-loop enables for easy reusing of code without manually modify it, see below for a example of the search in the array after the group with lowest active time which can be done independent of the amount of groups.

```
GetArray( Array := pump_array,
                Index := 0,
                ArrayElement => operation,
                Status => status );

if (operation.start) then

   for index :=1  to nr_group  do
      GetArray( Array := pump_array,
                Index := index,
                ArrayElement => temp1,
                Status => status );
      if (temp1.status=0) then
         exit;
      end_if;
   end_for;

   for index :=1 to nr_group  do
      GetArray( Array := pump_array,
                Index := index,
                ArrayElement => temp2,
                Status => status );
      if (((temp2.status=0) and ( TimerElapsed(temp2.total_active_time) <
      temp1:=temp2;
      end_if;
   end_for;
   operation.to_start:=temp1.index;
   PutArray( Array := pump_array,
                   Index := index,
                   ArrayElement => temp2,
                   Status => status );
   end_if;
```

3.7 Drawbacks and benefits with reusable code structure

A reusable code structure enables not only for reducing the developing time for the PLC program but also improves other aspects such as quality, reliability, etc. This clearly advocates for a this implementation and one should design the code structure for this purpose. During the design of the code structure would it require extra engineering time but it should be seen as a long term investment. The reason to this is that variation can be seen as a positive influence in a project. Adding extra code in a project will in the future minimize the development time for the next project using the same type of code and increasing the flexibility and functionality of the control system. Still the encapsulation of the different part in the PLC program isn't always obvious. This means that it is still possible that the engineer creates parts including several aspects which, as mentioned, complicates the reusing of code. Luckily during the development of the PLC program for the control and monitoring of the power transformer it was possible to distinguish different parts and encapsulate them cleanly. However it is still possible that the library will be overflowing with similar objects, as previous mentioned, but the created PLC code parts are on a basic level and easy to create and are used to build complex control logic. Also it is possible to use the library for offering a customer standard solutions that can be customized according to their specifications and offer this as a extra service. One can compare this with the ordering of a new car, where it is possible to choose between several standard components and also with the possibility of adding new unique components to an extra cost.

Furthermore the use of levels and encapsulation in the code structure eases the debugging since there are clear distinguishing of different parts in the program. For example if there are problems with the one of the alarms in one group it is easy to locate the problem. During this project this have been helpful when searching for errors and the debugging was easier compared to the approach with only have one level which gives a more unstructured appearance.

Comparing the first and second approach, presented in this report, it is noticeable that the object based approach requires less time and effort for building a application, thanks to the embedded flexility in the code. However considering the time which was spent during this project on debugging in this approach, the extra time that the first approach requires during the creation of the PLC program was worth it. Especially considering the stressful circumstances when one start the installation of the control system, on site at the customer, if it is necessary to start debugging the application. Still, the use of high level programming approaches such as object programming might be the future choice of programming language for PLC systems. Developers of PLC programming softwares are slowly starting to implementing, for example, the basics of object oriented programming in their developing softwares. Also already in the programming of industrial robots have the languages in some brands been switching from the use of low level to high level langues such as java. However, both approaches used in this project was able to relatively easily to reuse old and add new code.

The use of a library decreases the requirement of knowledge/experience within programming since the code is already written, tested and used before. Therefore if all the necessary parts exist in the library it is possible for one with minor knowledge regarding the programming, but with some experience of the developing software, to build a new PLC program. In order to ease this phase it is crucial with a good description of each part in the library and one can utilize machine states for this purpose. Even though the easy creation of a "customized" PLC program it is possible to create a incorrect control logic. A PLC program might for example not fully following the specifications considering alarms, etc. This means that it is still necessary with a validation of the control logic in order to ensure the behavior of the control system.

4
Validation of PLC program

Validation of a PLC program is a crucial part in order to avoid delivering a incorrect system to the customer. This chapter contains information regarding the validation of the PLC program, created during this project, by using simulation. Furthermore the design of the simulation model that have been used done with respect for reusability.

4.1 Background theory

Simulation eliminates the need of hardware and therefore it is possible to perform multiple and fast validations of the control logic. Furthermore it is sometimes desirable to validate that a solution will work during extreme and dangerous circumstances. This might not be possible with the physical hardware due to cost and safety issues. Simulation enables this and the engineer can perform these testes and ensure that the solution will work as well in normal as in extreme conditions[21]. Also, one of the main reason why using simulation because, it is possible to reduce the time needed during the start up of a new system since it is possible to detect hidden errors before starting the installation of a system

Even though the use of simulation enables the engineer to get fast feedback for every concept, the creation of the simulation model is a though and time consuming task. First of all it requires programming skill but also detailed information regarding the system to be simulated, for example how different actuators are triggered by the control system and their effects on the system [22]. However before starting with the simulation model it is wise to evaluate if it is necessary, considering the cost and time needed for programming and gathering information. Furthermore, the creation of a simulation model often started late in a project due to earlier mentioned aspects. Unfortunately, this means that some benefits with the simulation is lost[10].

There are several different types of simulation models, each one with its benefits and drawbacks. Two examples of model types are dynamic and static model. A static model calculates differen states for each time a event occurs. An dynamic model on the other hand calculates the state of the system continuous and is time dependent. The dynamic model consist, usually, of differential equations which describes the behavior of the simulation model.

In order to perform the validation with a simulation model it is possible to for example use virtual commissioning (VC). VC is performed by using either the real physical controller (hardware in the loop (HIL)) or a soft controller (software in the loop (SIL)) and connect to the simulation modelfor example via Open Platform communication(OPC) technology.

4.2 OPC

OPC is used for communication in industrial application and is specified by the OPC foundation. The purpose of using an OPC server is basically to enable the access of data from the PLC. The data from a PLC can easily be accessed by simple using an OPC DA (data access) server and an OPC DA client. The client is used for reading the data from the PLC via the server to desired interface, for example the simulation model. However before the data appears in the server it has to be read from the PLC and this is done by a cyclic synchronization in a specified interval [23].

However the use of an OPC server can sometimes create unwanted problem if there is a time critical application. The reason to this is that a short signal from the PLC might be missed due to the cyclic reading. In order to solve this Carlsson[10] propose a extension of the OPC communication that eliminates this problem.

4.3 Virtual commissioning

The concept of VC is to use the real PLC controller and control logic as it is supposed to be used in the finished installation. The PLC is connected to a virtual simulation model, as mentioned before, via OPC technology see figure 4.1. This means that the engineer can be performing the validation similar as it would be with the real physical system.

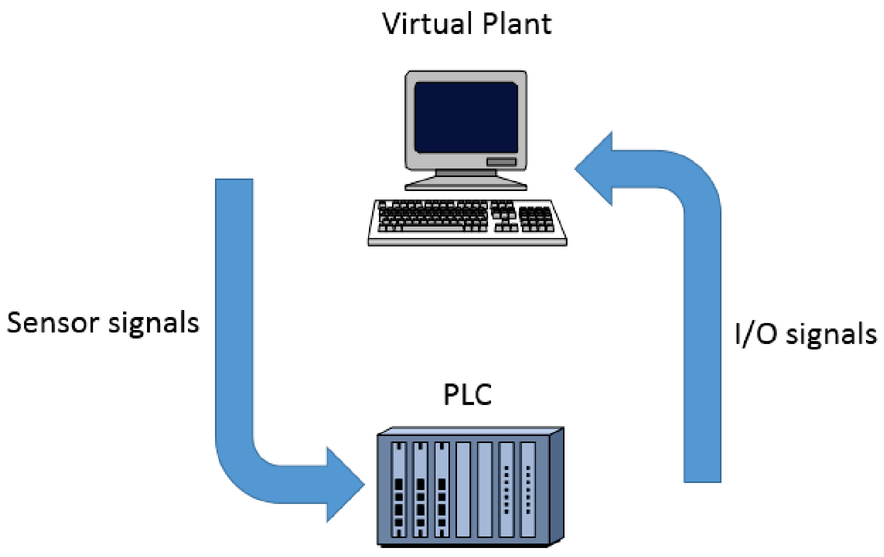


Figure 4.1: Concept of virtual commissioning

One can say that the virtual commissioning consist of two parts, the real controller and the virtual plant, but one can also be using a soft PLC. The real controller includes the PLC and sometimes also the HMI which the operator can use for monitor and control of the system. This part can also include the I/O signals and be connected to a visualization of the virtual plant. For example it can consist of signal lamps and smaller fans that is a smaller version of the real system only for visualization purposes. However the actual I/O signals are never necessary to be used in the virtual commissioning since the communication between the controller and virtual plant is done via the OPC server. This means that it is possible to eliminate the need of hardware. Also it is possible to connect the HMI to the physical or soft PLC, see figure 4.2. This means that the control system can be used for education purposes for operators using with the real HMI and see the effects of the actions in real time without any possibility of jeopardizing any hardware or humans.

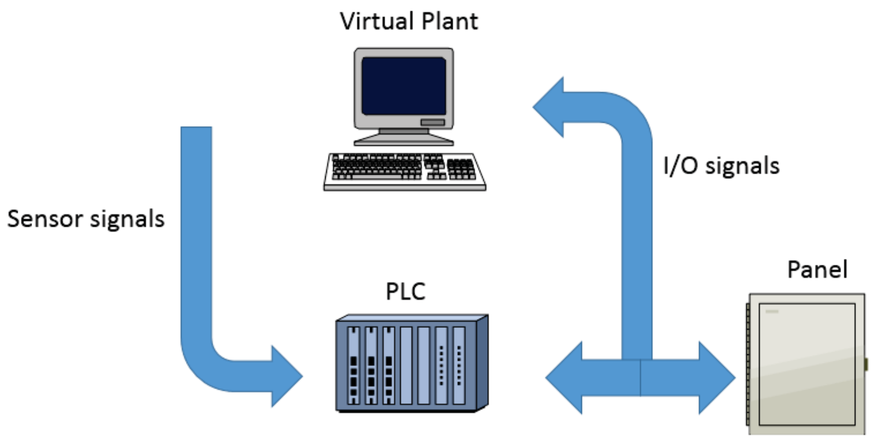


Figure 4.2: Concept of virtual commissioning with control panel

The level of detail in the simulation model depends on the system and customer. For example if it is a production cell including robotics it is necessary with four steps/-parts, process design, physical device modeling, logical device modeling and system control modeling [24]. However if it is for example a process such as a furnace or a power transformer, then the mechanical aspect is not necessary to model in order to validate the control logic. Rather the behavior of the "machine" for example in terms of heat generation within the process .

Furthermore there are some problems that can occur independent if it is in the real and virtual world, for example considering the synchronization between the controller and simulation model or actuators actual state. This can for example lead to that the PLC is performing the wrong actions even though it is the right one according to the PLC program. This is possible to solve with better conditions that has to be fulfilled before the start of an operation. One drawback with this solution is that the improved conditions can result in a deadlock if the PLC and simulation model are unsynchronized.

4.4 Software for creation of simulation model

In order to create a simulation model for validating the PLC code there are several softwares available. The choice of software depends on the system to be simulated, for example if a system contains moving parts and the visualization of the system is important it is suitable to use a software such as 3D-Create, Winmod, experior, etc. These types of software allow the engineer to define how the system should behave virtually, for example the kinematics for customized parts. However when there is no need of visualizing the functions/system it can be suitable to use MATLAB due to the advantages in a mathematical perspective[25].

MATLAB have a add-on called "Simulink" which allows the engineer to connect to either the real PLC or a soft controller easily with a tool called OPC toolbox. This means that for a process that for example controls a cooling system, one can define the generation of heat mathematically in Simulink. This simulation model corresponds to the real world system if the formulaes are correct. By doing this it is possible to runs the system totally virtually and see how for example the control logic will affect the outcome in the system. [25].

Simulink also have possibility to create a simple user interface which can be used to monitor and control the simulation model during the validating of the PLC code. This means that it is possible to testing everything from the basic of the system to generating error signals either automatically or on demand. Simulink enables for a easy set up different scenarios in one run with the use of pulse generation which is triggered at a certain point. However independent of software that is used in a project it is necessary to gather information regarding the system in order to create a simulation model corresponding to the real world system[25], see figure 4.3.

4.5 Creation of simulation model

In this case the system to model is a power transformer. A power transformer is a complicated component that have several factors to consider. However the main purpose with the control system, in question, is to control the cooling equipment. This means that it is necessary to include the heat generation, in the process of transforming voltage up or down, in the simulation model. Cooling of the power transformer is a vital part and if the temperature is to high the ageing of it will be accelerated. Therefore it is desirable to include this as well in the simulation model in order to see the result of different cooling strategies. Furthermore the control system monitors parameters such as gas detection, leak detection, tap changer, load, etc.

4.5.1 Encapsulation of simulation model

It is common that engineer creates one massive model which includes the whole system. This complicates the reusing of code since it is no clear distinguishing between the different parts in the simulation model. Luckily, similar as for the encapsulation of the control logic in the PLC it is possible to utilize the same in the principle in the simulation model in order to ease reusing. This also enables for the possibility of testing different parts of the program separately.

For example in this case it is necessary to include heat generation, generation of error signals,etc. By encapsulating each one of these into several parts it is possible to start reusing them in other projects. For example if the system will be monitoring the gas generation in the power transformer it is possible to extract this part from the standard library and fast validate if the control logic is correct or not for this purpose. Also sometimes there are several error signals each one connected to different part in the program, by designing one type that is applicable for all the error generation it is possible to fast and easy build the simulation model.

4.5.2 Power transformer

The system needed for validating the control logic can be seen in figure 4.3. However depending on the power transformer specifications it can sometimes be necessary to monitor parameters such as humidity, gas, leak detection, etc which also have to be simulated sometimes. This means that variation can be found in both the control logic and simulation model. For more information regarding the equations behind and different parameters, see appendix A.

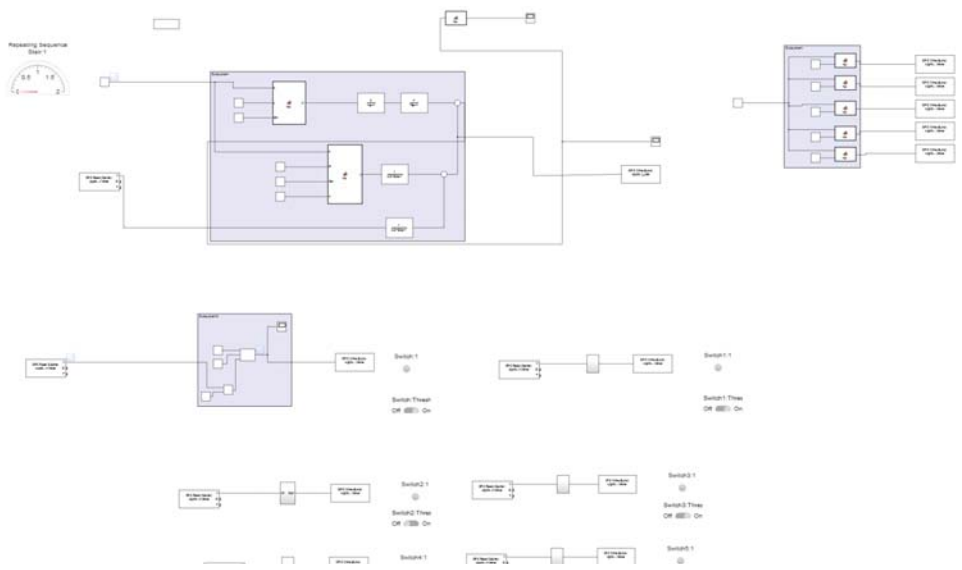


Figure 4.3: Example, Simulink with the OPC toolbox

As it can be seen in figure 4.3 the simulation model have been divided into several parts. In the top can the heat generation be found which calculates the temperatures in the power transformer during different loads. Furthermore in order to validate how the control logic handles a error in one of the cooling groups simulation logic were created for this purpose, see figure 4.4. The simulation model is connected to the I/O signals which triggers the start of for example a pump and then since the control logic continuous monitors if the pump is activated a signals is received from the contactors that activates the pump based on the signals from the I/O signal. If this signal is not received the PLC should switch to the next group. In order to test this the simulation model listens to the activation I/O signal and via the interface in Simulink it is possible to interrupt the signal which is being sent to the PLC which indicates a successful activation of the pump. Furthermore some parameters which are necessary in order to fully validate the control logic such as gas generation have been solved with a more basic solution (not included in figure 4.4). In this case have the gas generation been simulated by using a "bar" which change to different values in order to see that the PLC sends the correct alarms to the operator. The same reasoning have been applied on the tap changer and leak detection in the oil conservator, see figure 4.4.

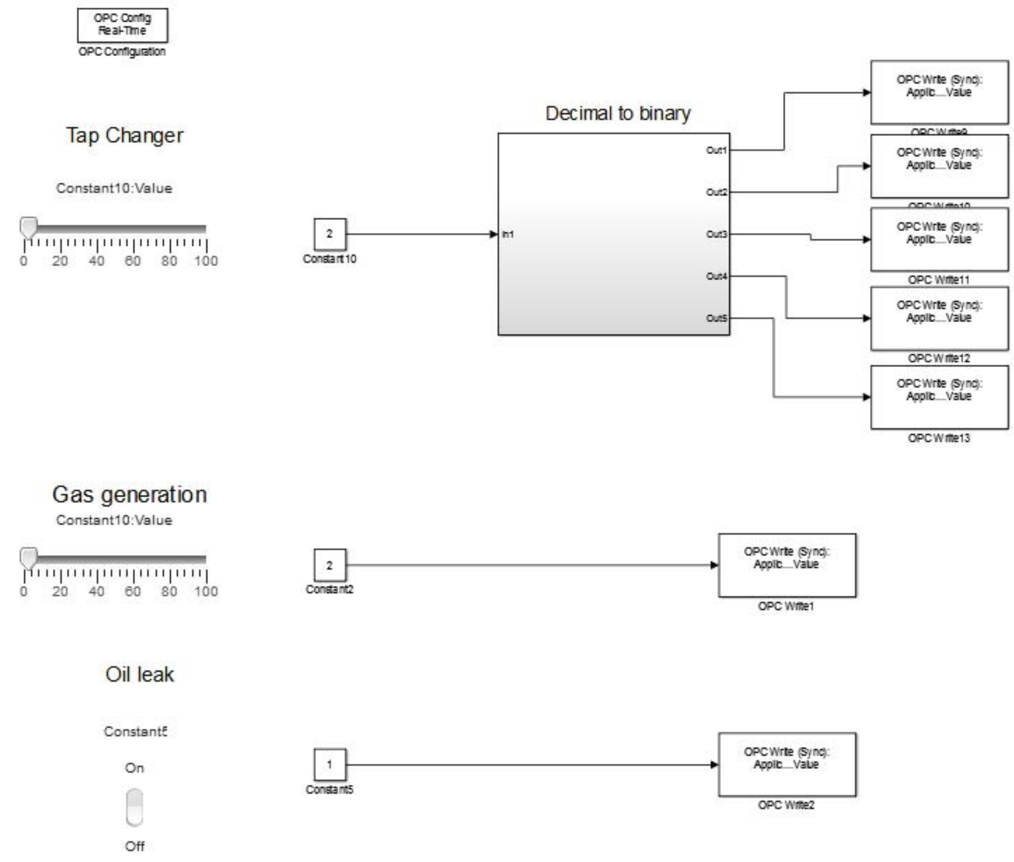


Figure 4.4: Simulation model used for validation of control logic, created in Simulink.

4.6 Validating control logic

Using the created simulation model in Simulink it is possible to connect it to the AC800m via an OPC server and connect the parameters in the simulation model to variables which are connected to I/O signals. It is possible to validate the control logic totally virtually with the 800xA system by using the soft controller. This can be connected to the OPC server and then to the simulation model, similar as the real system. During this project it was the HIL solution preferred since it gives a more realistic impression when validating the control logic since the actual PLC unit is used.

The validation was performed by setting different scenarios in Simulink and connect it to the PLC. Scenarios that was performed was for example during normal conditions with a load capacity of around 80 % to extreme conditions with a load factor of 200 %. Also during the validation were different error signals sent to the PLC, for example if a fan breaks down the PLC should directly switch to the next one with minimum running time. This enable for testing the control logic during operation with faulty cooling equipment and see that the next group will start and also if there are none available to start that the right alarm is sent to the operator.

All the testes that where performed revealed several minor errors in the control logic which, of course, where corrected. After the bug had been corrected a new validation run where performed in order to ensure that the control logic is behaving as intended.

However during this project the creation of the simulation model have been done in parallel with a physical representation of some of the functions on the power transformer. This was due for selling purpose but it is also possible in a limited extent to validate some of the control logic, see figure 4.5. Worth mentioning is that the Visualization hardware could also be sending different parameters to the PLC thanks to several potentiometers. However it is also possible to only use the LED-lamps for displaying when for example a pump motor is started and use the simulation model at the same time.

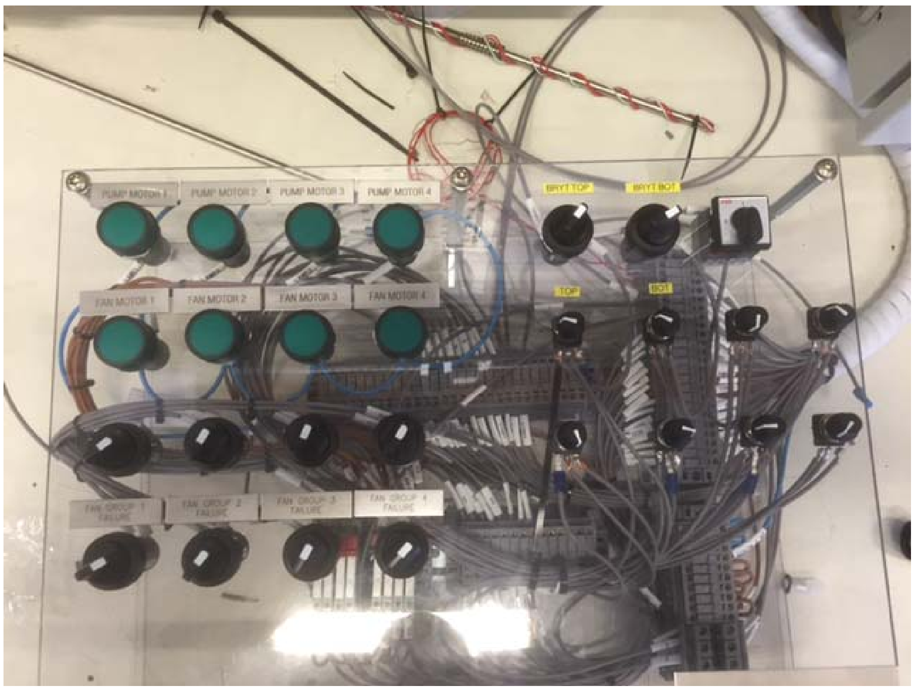


Figure 4.5: Visualization for selling purpose

4.7 Drawbacks and benefits with validating the control logic using a simulation model

It is clear that the use of simulation can reduce the time needed during the start up of a new system since one can identify and eliminate errors in the PLC program. Therefore it is possible to save a significant amount of money otherwise due to the fact of having a physical system in standby waiting for the PLC system to working. Still it is not possible to be absolutely 100 % sure that the PLC program will be behaving as intended with the physical system due to several different unknown aspects. However as time proceeds these unknown aspects can be included in the simulation model library in order to eliminate this problem in the next project. This means the start up time will continuously be decreasing and perhaps in the future be almost eliminated.

One of the drawbacks with the use of simulation is the time that have to be spent on the creation of the simulation model. The encapsulation of different parts in the simulation model enables for decreasing this time. It is important that one evaluates the value that a simulation model might contribute with before starting the creation of a complex simulation part. Sometimes can be enough with a simple solution in order to validate the control logic. Still a simulation model that corresponds exactly to the real physical system gives a more realistic impression and can be used as a selling point when presenting for the customer. Furthermore this allows for training of operators with a new control system without risking the power transformer. However the use of simple representation enables for earlier use of simulation when the engineer are programming the PLC program since more complex simulation are time consuming.

The use of a simulation isn't necessary to be limited to the validation and design phase in a project but rather sometimes can the model be used to improve the running result of the application. The benefits that the simulation model can contribute with is mainly in the future decision making. For example, the simulation model can predict heat generation and improve the uncertainty in by using real data value as parameters. Based on this information it is possible to activate counter measures before there are needed or trigged by the system. This can in some application result in a increased life time of the system since risks such as overheating is minimized. Also it is possible to use the simulation model as a extra security if the sensors that are for example measuring the temperatures break down.

Also, reusing simulation and creating a library enables for testing the PLC program continuously during its development and evaluate several different concept easily. This can lead to that the Still the chosen approach is sensitive for variation in the simulation model since one have to add new types each time this occur. The same approach with several layers haven't been realistic to implement in this case which resulted in a lost flexibility in the code.

5 Conclusion and future work

Is a PLC a strategic choice for the control and monitoring of a power transformer?
It is clear that it is beneficial with a implementation of a programable controller for control and monitoring of a power transformer. First of all it enables for a future collaboration with the smart grid solution since it can provide the operator with detailed information, both historical and real time data. Using this information the operator can start to see irregular patterns and use it as a tool when making a decision. This means that it is possible to avoid one major cost which will otherwise occur if it is necessary to perform a unplanned down powering of the power transformer. It is possible to extent the time between service event that is due to the equipment monitored by the PLC system. A major advantage with a PLC system is the modularity that it offers. One can build the hardware after specifications which means that the cost and functionality is optimized. Also this improves the service of the control system since faulty components easy can be replaced.

Also the flexibility that the PLC system offers in terms of applying changes in the control logic after the final installation is a major advantage. Some conditions might change over time and with a Plc it is easy to adapt the control logic without rebuilding the hardware by simply downloading a updated PLC program. By this it is possible to use the best control logic available even if the system is older since it is always possible to easy apply changes.

How can one design the PLC/simulation code structure for increased reusability?
The creation and validation is time consuming if one have to create the PLC program and simulation model from scratch each time. Therefore it is preferable to implement methods that enables for easy reusing of code, such as in this project with a object oriented code structure. However the object oriented approach isn't fully supported by the PLC softwares but some parts are possible to utilize anyway. During this project it was noticeable that some features from the object programming was powerful but also contributed with a complicated debugging such as the use of objects. However using features from the object oriented programming approach such as encapsulation, "inherence",etc in the code structure. This enables for the engineer to easily modify,add,remove, etc in the code without considering the whole application. In this case this was possible thanks to the clean encapsulation of the different parts in the PLC program and "inheritance" (similar only, not fully applied) for the building of the different parts using control modules.

Furthermore since the same approach of building a library of encapsulated parts have been used for both the creation of PLC program and simulation model one can easily validate the control logic by importing the necessary parts. This does not only increase the quality of the PLC program and validation, since the parts are well tested and used before, but also it lower the requirement for knowledge considering the programming.

The PLC program was created in multiple levels which in this case wasn't fully possible/optimal with the simulation model. The simulation model have been utilizing the encapsulation but only one level. This means that the flexibility of the encapsulated part is lost but the programmer can add different functionalities to the simulation model. A future simulation model should be utllizing the multple level approach but for some function this is unnecessary work due to its entailed appearance.

Future work
The chosen approach in this project wasn't able to handle smaller variation without adding a extra type to the library. However in order to create a code structure that can handle this it might be possible to generate code using a high level programming that can handle this complex problem. Due to time limitations in this project this part haven't been included as intended. Therefore a natural next step for this project would be to investigate how the use of high level programming can generate complete PLC code code with respect for variation between different projects.

Bibliography

[1] Bolton, W. (2015). Programmable logic controllers.

[2] Birgit Vogel-Heuser & Co. (2015). Challenges for Maintenance of PLC-Software and Its Related Hardware for Automated Production Systems Selected Industrial Case Studies.

[3] ABB AB. (2011). Compact Control Builder AC800M configuration student compendium.

[4] Petter Falkman, Erik Helander, Mikael Andersson. (2011). Automatic Generation: A way of ensuring PLC and HMI standards

[5] Wenbin (William) Dai IEEE Member, Jukka Peltola, Valeriy Vyatkin IEEE Senior Member, Cheng Pang IEEE Member. (2014) Service-Oriented Distributed Control Software Design for Process Automation Systems.

[6] Marc H. Meyer and James M. Utterback.(1995). Product Development Cycle Time and Commercial Success.

[7] Alois Zoitl and Valeriy Vyatkin. (2008). IEC 61499 Architecture for Distributed Automation: The "Glass Half Full" View.

[8] Vivek Hajarnavis, Member, IEEE, and Ken Young. (2008). An Assessment of PLC Software Structure Suitability for the Support of Flexible Manufacturing Processes.

[9] Dougall, D. J. (1998) Applications and benefits of real-time I / 0 sl"m um lon for PLC. Netherlands: Elsevier Science lad.

[10] Dahl, M., Bengtsson, K., Bergagård, P., & Fabian, M. (2016). Integrated Virtual Preparation and Commissioning: supporting formal methods during automation systems development.

[11] Kristofer Bengtsson Bengt, Lennartson, Oscar Ljungkrantz, Chengyin Yuan. (2012). Developing controllogicusingaspect-orientedprogramming and sequenceplanning.

[12] ABB AB. (2016). Smart Grid.

[13] Del Vecchio, R. M. (2010). Transformer design principles: with applications to core-form power transformers. Boca Raton, FL: CRC Press.

[14] Clark W.Gellings (2009) The smart grid:Enabling energy efficiency and demand response

[15] Ratings, D. (2012). For transformer monitoring, control and communication.

[16] Marcello Bonfi and Cesare Fantuzzi.(2000) Mechatronic Objects encapsulation in IEC 1131-3 Norm

[17] Johan, R., & Martin, F. (2003). Automatic Generation of PLC Programs for Control of Flexible Manufacturing Cells.

[18] Mader, A. (2000). A classification of PLC models and application.

[19] Benitez Pina, I., Vazquez Seisdedos, L., Villafruela Loperena, L.(1999). INCLUDING OBJECT-ORIENTED PROPERTIES IN THE PLC's PROGRAMMING LANGUAGES.

[20] Manuel Pineda-Sanchez, Marina Pérez-Vázquez, Ángel Sapena-Bañó, Javier Martinez-Román, Juan Pérez-Cruz, Rubén Puche-Panadero, Victor Pérez-Vázquez (2015) Programmable Logic Controllers (PLC) in the Packaging Industry: an Object Oriented Approach for Developing Control Programs

[21] Oppelt, M., & Urbas, L. (2014). Integrated Virtual Commissioning an essential Activity in the Automation Engineering Process.

[22] Chi G, L., & Sang C, P. (2014). Survey on the virtual commissioning of manufacturing systems.

[23] Wolfgang, M., Stefan-helmut, L., & Matthias, D. (2008). OPC Unified Architecture.

[24] Minsuk Ko, Euikoog Ahn and Sang C Park. (2013). A concurrent design methodology of a production system for virtual commissioning.

[25] MATLAB. (2016) MATLAB and Simulink

[26] Starner, C. & Chessin, M. (2010). USING EMULATION TO ENHANCE SIMULATION. 3300 Breckinridge Blvd, Ste 100 Duluth, GA 30096, USA: E2M, Inc.

[27] Henrik, C., Bo,S., Fredrik, D., & Bengt, L. (2012). Methods for reliable simulation-based PLC code verification.

[28] Kleinschmager, S. (2012). spect-Oriented Programming evaluated : A Study on the Impact that Aspect-Oriented Programming can have on Software Development .

[29] Kristofer, B. (2009). Operation specifications for sequence planning and automation design.

[30] Lock-Jo, K., Chang, M. P., Chang, H. L., SangChul, P., & Gi-Nam, W. (2010). Simulation framework for the verification of PLC programs in automobile industries.

[31] Mulman, B., Devinder, T., & Gi-Nam, W. (2008). Generation of PLC Ladder Diagram Using Modular Structure.

[32] Speck, A. (2003). Reusable Industrial Control Systems.

[33] Dessì, Massimiliano.(2009). Spring 2.5 Aspect Oriented Programming.

[34] Martin Obermeier, Steven Braun, and Birgit Vogel-Heuser, Senior Member, IEEE. (2015). A Model-Driven Approach on Object-Oriented PLC Programming for Manufacturing Systems with Regard to Usability

[35] Kevin Hoffman, Patrick Eugster. (2009). Cooperative aspect-oriented programming.

[36] IEC. (2005). IEC 60076-7.

[37] JIAN-FENG CHEN, WEI-WEN WU, ZHI-YAN WANG.(2010). RESEARCH OF AUTOMATED ASSEMBLY LINE CONTROL SYSTEM BASED ON MODULARIZATION

A

Appendix 1- The power transformer

Power transformer is used all over the world in power systems and have a high variation in terms of design both mechanical and electrical. A typical location of the power transformer is at a power plant(sender) where it is used to increase the voltage to a desired voltage, for example 800 KV, and then it is send to another station(receiver) for further distribution to for example households. Transforming the voltage will reduce the energy loss which otherwise will for instance be caused by the resistance in the wires. The wires connects the sender and receiver station togehter[13]. In order to understand the savings that can be done with a power transformer see equation A.3 where it is clear that a higher voltage (U) leads to a decreased loss in the system.

$$R = \rho * \frac{L}{A} \tag{A.1}$$

$$Loss = I^2 * R \tag{A.2}$$

$$\frac{Loss}{P} = \frac{I^2 * R}{U^2} \tag{A.3}$$

In order to simplify the consistence of a power transformer one can say that it consist of two coils, one primary and one secondary. These generates an electric magnetic field when the current passes through the primary coil. When the magnetic field is generated the current passes through as in figure A.1.

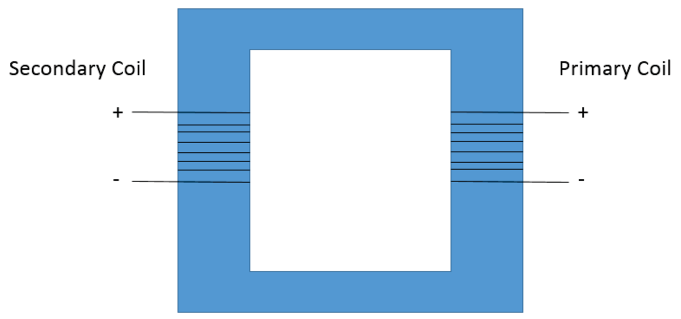


Figure A.1: Example of coil used in a transformer

The voltage that is generated in the magnetic field can be calculated by equation A.4, which describes the relationship between the primary and secondary coil. During this process there is a small energy loss within power transformer even though the efficiency is around 99 % [13]. The energy loss is often in form of heat which is generated in the magnetic field, coil, wires, etc and in order to reduce the risk of overheating the transformer is often some kind of cooling system used, for example oil, air, water, etc. This cooling system is then controlled by a system which can for example is monitoring oil temperatures or the load factor on the power transformer and activate or deactivate the cooling system based on these parameters.

$$\frac{E_1}{N_1} = \frac{E_2}{N_2} \tag{A.4}$$

The heat that is generated in the power transformer increase the so called ageing rate. This is a parameters that defines the degradation of the isolation paper in the coil. The paper is used for reducing the risk of sparking between the wires on the coil which results in a gas generation and if this is to high it can be necessary to repair the power transformer. Therefore is the cooling system a vital part of the power transformer. The cooling system usually consist of four main components, fans, pumps, heat sinks and the cooling media which the coil is immersed in. The cooling system appearance varies between each type of transformer, some are only using pumps and oil while others are all the mentioned components.

A.0.1 Temperature Calculation

It is possible to predict the heat generation within the power transformer due to the losses. This is normally used to compare that the designed power transformer is behaving as intended and according to the specifications before starting the production. In order to perform these types of calculations there are a standard, SS-IEC 60076-7 which describes the heat generation of a power transformer with oil as cooling liquid [36].

In order to calculate the θ_o there are several parameters necessary to determine before which are the following,

- ambient temperature(θ_a)
- Load factor (K) - The present load on the power transformer.
- Hot-spot temperature (θ_h) - Gradient at the load considered .
- Hot spot to top oil(in tank) ($\Delta\theta_{hr}$) Gradient at rated current.
- Top oil(in tank) ($\Delta\theta_{or}$) temperature rise in steady state at rated losses (no-load losses + load losses).
- Hot spot to top oil(in tank) ($\Delta\theta_h$) gradient at the load considered.
- Average oil time constant (τ_0).
- Ratio of load losses at rated current to no-load losses (R).
- Exponential power of total losses versus top-oil (in tank)(x) temperature rise (oil exponent).

- Exponential power of current versus winding temperature rise (winding exponent)(y).
- Winding time constant (τ_W).
- Thermal model constant (k_{xy}).

These parameters can either be determined by recommendations found in standards or it is possible to use so called finger prints which is collected during the testing of the power transformer. The finger print data is unique for every power transformer and can be used to predict how the heat is generated when it is active with more precision since it is adapted for that power transformer[36].
There are two possible scenarios for the heat generation, either it increases or decreases. These two scenarios can be described in two differential equations which can be seen in equation A.5 for increase and A.8 for the opposite.

Heat increase

$$\theta_h(t) = \theta_a + \left(\Delta\theta_{or} * \left(\frac{1+R+K^2}{1+R}\right)^x - \Delta\theta_{oi}\right) * f_1(t) + \Delta\theta_{hi} + (H_{gr} * K^y - \Delta\theta_{hi}) * f_2(t) \tag{A.5}$$

$$f_1(t) = \left(1 - e^{(-t)/k_{11}*\tau_0}\right) \tag{A.6}$$

$$f_2(t) = k_{21} * \left(1 - e^{(-t)/k_{22}*\tau_0}\right) - (k_{21} - 1) * \left(1 - e^{(-t)/\tau_0/k_{22}}\right) \tag{A.7}$$

Heat decrease

$$\theta_h(t) = \theta_a + \Delta\theta_{or} * \left(\frac{1+R*K^2}{1+R}\right) + \left(\Delta\theta_{oi} - \Delta\theta_{or} * \left(\frac{1+R*K^2}{1+R}\right)^2\right) * f_3(t) + H_{gr} * K^y \tag{A.8}$$

$$f_3(t) = e^{(-t)/(k_{11}*\tau_0} \tag{A.9}$$

B

Appendix 2- Photos of hardware

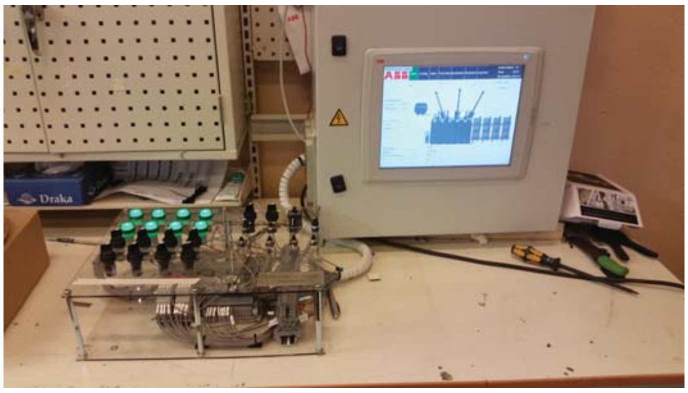


Figure B.1

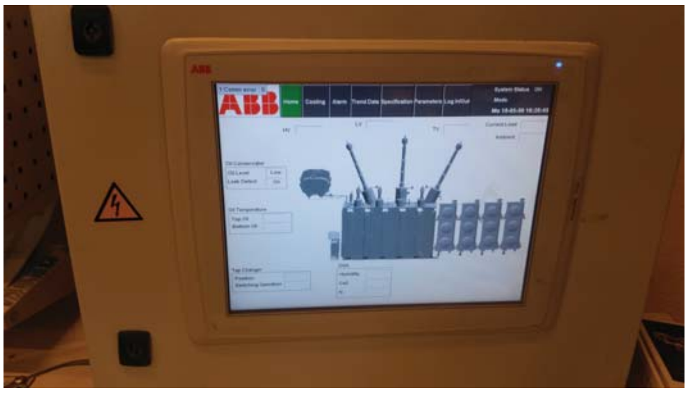


Figure B.2

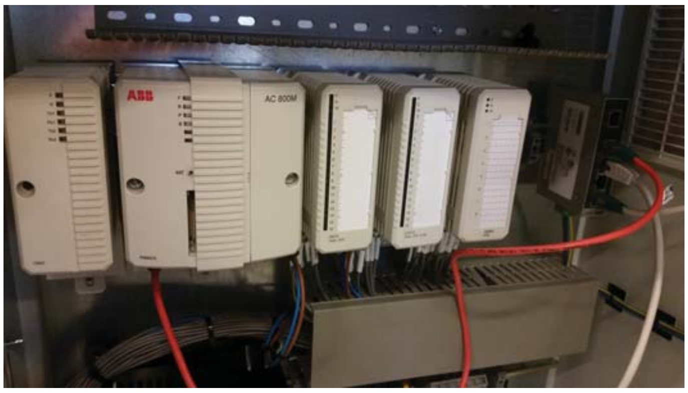


Figure B.3

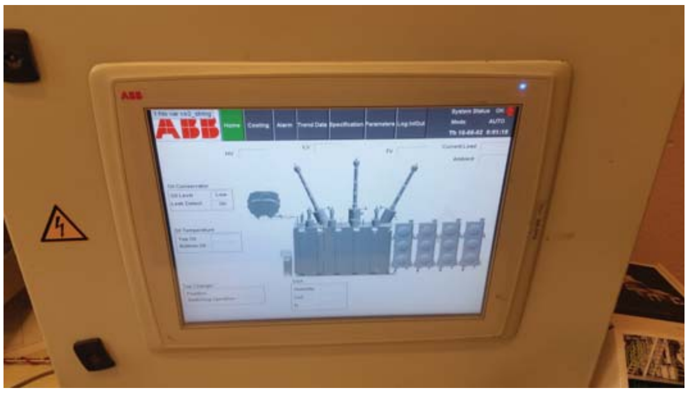


Figure B.4