DEPARTMENT OF ELECTRONIC & ELECTRICAL ENGINEERING

University of Strathclyde Engineering

Realisation of a Smart Grid Control Room for a Microgrid

Interim Report

Scott Clark	200905796
Connor Hughes	200916925
Jennifer Ingram	200907667
James McNiven	200943419
Gareth Mitchell	200904902

13/12/2013

Abstract

This interim report will detail the progression of the Masters project, entitled "Realisation of a Smart Grid Control Room for a Microgrid". In this project, Siemens WinCC software is being used to create a control room Supervisory Control and Data Acquisition (SCADA) environment for use in conjunction with the University of Strathclyde's Distribution Network, Automation and Protection Laboratory. A Graphical User Interface (GUI) is currently being designed for use with the available historical data, which will be adapted to incorporate and display real-time data in the future. The initial stages of the GUI design are shown, alongside the approaches to data analysis and acquisition. A plan for future project work, encompassing a Gantt chart, is included to highlight the critical paths that will ensure the timely completion of the project.

Table of Contents

Glossary of Terms

CR1	Control Room 1
CR2	Control Room 2
CSV	Comma-Separated Value
DG	Distributed Generation
DNAPL	Distributed Network Automation and Protection Laboratory
GSC	Global Scripts
GUI	Graphical User Interface
HDMI	High-Definition Multimedia Interface
HMI	Human-Machine Interface
IEC	International Electrotechnical Commission
LAN	Local Area Network
LCD	Liquid Crystal Display
LPC	Local Power Controller
LV	Low Voltage
MCB	Miniature Circuit Breaker
MCCB	Moulded Case Circuit Breaker
MTU	Master Terminal Unit
OLE	Object Linking and Embedding
OPC	OLE for Process Control
PMU	Phasor Measurement Unit
PV	Photovoltaic
RTS	Real Time Station
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
WAN	Wide Area Network

Table of Figures

1 Introduction

Room R134 in the Royal College of the University of Strathclyde is home to the Institute of Energy and Environment's microgrid facility. In terms of usability, the microgrid environment is impressive and intricate from both a hardware and software point of view. Understandably, the microgrid is attractive to the many external visitors to the department, which can vary from academics to perhaps those less technically minded, such as members of government. Currently, there is no powerful and practical existing method to exhibit the microgrid's full capabilities in an intelligent and innovative manner. Consequently, the task of this project is to produce a control room environment that will effectively display the data collected from the microgrid in a style that is beneficial to any potential visitor.

With the first 8 weeks of the project having been completed, this report will detail the work that has been undertaken thus far. This includes initial background research, analysis of historical microgrid data, and the initial stages of the control room GUI design. As well as being advantageous to any visitor's understanding of the microgrid, the control room design will consist of several exciting and novel visualisation techniques. This report will also discuss the work planned to allow for the continued progression of the project, as well as reflecting on what has so far been achieved.

1.1 Aims

The ultimate aim of this Masters group project is to create a control room environment that will prove both beneficial and informative to any visitor regardless of their background in, and understanding of, power systems or microgrids. This control room environment will be set up in CR2. CR1 is currently used to house the user interfaces and computer controls for the microgrid, however, due to limitations in the room's size and layout, it will be more desirable to display the microgrid information in CR2. CR2 contains a greater number of computers and monitors, including a large wall-mounted LCD screen, which will allow for more clarity and effectiveness in the display of microgrid information. A map of R134, where the microgrid is housed, is shown in Figure 42, Appendix A.

The main challenge will be to convert the ample data that is collected at the microgrid into valuable and succinct information, be it quantitative or qualitative. This could include inferred information that is not technically measured but could prove important for a better understanding of the microgrid's operation. The control room system will take advantage of the plethora of archived historical data which will be used in conjunction with real-time data collected from the microgrid.

Whilst still employing tried and tested visualisation techniques that have been used over the past few decades in SCADA systems, another aim of the project will be to develop novel and innovative approaches to data visualisation. This could set the benchmark for similar future projects.

1.2 Objectives

Due to the nature and size of the task at hand, it was necessary to split the project into a series of attainable project objectives. The first objective was to gain a broad understanding of both the university's microgrid and the concept of microgrids as a whole, which would provide a solid base for the progression of the project. It was felt a necessity to thoroughly analyse the vast amount of historical data, not only to aid in the decipherment of the microgrid's past operation, but also to help comprehend data collected from future, real-time running of the microgrid.

Approximately 280 different variables are measured at the microgrid, which is too great a number to efficiently monitor and display in a control room. The data collected can be divided into input and output data, and sub-divided into integer, double and Boolean data types. Therefore it was essential to conduct widespread research into previous and existing approaches to presenting information in similar, as well as more complex, control rooms. This entailed visits to the University of Strathclyde's GSE Systems Power Station Simulator Suite and Scottish Power's Operational Control Centre at their Strathkelvin House site, with both visits proving extremely educational and eye-opening.

Information in the control room will be displayed via an interactive GUI that will be built using Siemens WinCC 7 software, as well as the use of both C and Python programming languages. The objective of the initial GUI design will be to display what is deemed to be the most important information taken from the historical data, based on the results of the research into previous control room designs. The successful build of this initial GUI will pave the way for a more advanced multi-layer GUI design that combines the display of real-time data with the option of comparison with historical data trends.

Optional, but certainly attainable, objectives include the extraction and presentation of real-time data from National Grid or the university's PMU, as well as the integration of webcams into the microgrid architecture that will permit visual monitoring and allow safe off-site control.

1.3 Technical Risks

Throughout this project an up-to-date risk register is required, so as to ensure all project members abide by foreseen risks and their corresponding mitigating actions. Safety is paramount within all research facilities at the University of Strathclyde, including the Distribution Network Automation and Protection Laboratory situated in R1.34 of the Royal College Building where the majority of project work is undertaken. Therefore, it is essential for all project members to follow the institution's strict legislation. Before commencing in project work involving the microgrid facility, a safety induction was given by research staff members to ensure all safety risks were appreciated.

Technical risks can significantly affect the progression of the project. A time criticality should be associated with each risk, indicating when this risk is no longer present, as the accompanying task should be completed. Risk severity is indicated by three fields; likelihood, class and impact. These headers represent the probability, risk classification in comparison to others and the overall effect on project progress respectively. Mitigating actions are recorded to aid in overcoming any problems should they arise. The risk register is available for reference in Appendix C.

2 Background Research

2.1 Microgrids

A microgrid is a localised electrical system which is usually connected to a centralised grid, but can also operate independently after separation from the grid when a power quality event occurs. It is an example of DG, a concept which is becoming more prominent in power delivery systems, as it enables a higher efficiency of power transmission and allows for reduced energy usage in the power delivery network. Traditional networks involve the transmission of electricity over long distances, whereas DG resources generate electricity from many small sources, allowing for reduced impact on the environment and a higher security of supply [1].

When operating in grid-connected mode, the parameters of a microgrid are controlled by the main utility network, however, when islanded there are a number of elements of the microgrid system which require control techniques to be in place. These include elements such as: voltage and frequency levels, power quality, power output and a balance between supply and demand. Despite being an attractive opportunity to maintain power supply to critical loads in the event of a disturbance, intentional islanding of microgrids is a practice that is still in the early stages of development and implementation. This is due to the fact that there are a number of aspects of the dynamics prior to, during, and after microgrid separation yet to be fully understood. For this reason, islanded operation is currently disallowed by utilities in the UK, because voltage and frequency variations exceed the acceptable limits as stated by the Electricity Safety Quality and Continuity Regulation [2]. This is because there are concerns regarding power quality maintenance and the safety of personnel and equipment, both of which are intrinsic to the successful operation of a power grid network.

2.1.1 Protection

Two fundamental issues to be addressed with regards to the operation of a microgrid are determining when the microgrid should be separated from the network, and deciding how best to provide the isolated grid with sufficient coordinated fault protection whilst operating as an island [3]. The protection of a microgrid and DG-penetrated networks is an area of ongoing research and development. When the electricity market was privatised and reorganised in 1990, increased numbers of generators were installed within the different voltage levels of the network, changing the distribution system from passive to active, as the DG essentially created a multi-source power system [4]. Power flow was no longer unidirectional being transmitted only from network to customers; it also flowed from customers' additional generators into the network. The difference between passive and active network architectures is shown in Figure 1 and Figure 2 respectively. This change rendered some traditional protection methods unsuitable.

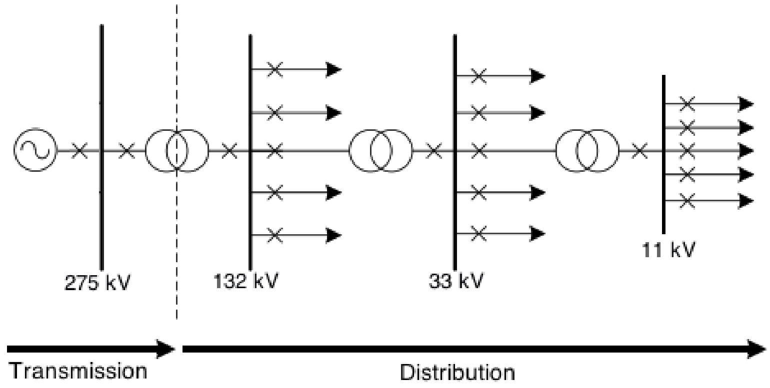


Figure 1 - Passive Network Architecture

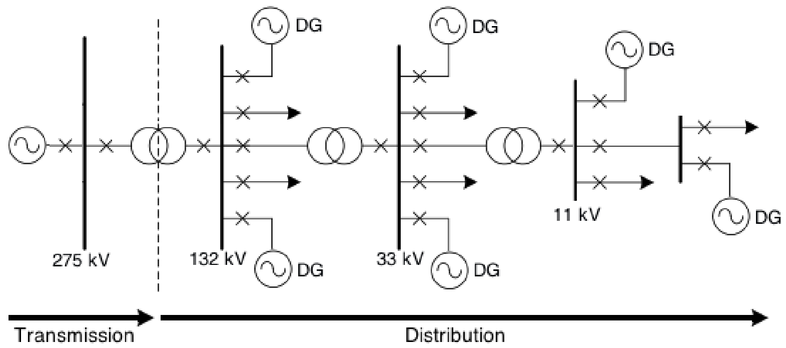


Figure 2 - Active Network Architecture

Protection of microgrids cannot be achieved with the same philosophies as traditional protection schemes, since the following additional factors must be taken into consideration: bidirectional current flow, looped feeders and reduced fault levels due to the absence of the utility grid in islanding [5].

2.1.2 Real-World Applications

The increased security of supply that is provided by a microgrid facility can have numerous benefits to a large variety of real-world applications. In 2012, a joint venture between

Alameda County and Chevron Energy Solutions saw Santa Rita jail in Dublin, California become the first self-sufficient prison, housing its own microgrid which can sustain power provision should the prison's connection to the utility grid be interrupted at any point: a hugely important factor to assure the continuous operation of security mechanisms [6]. Within the smart grid system there are five 2.3 kW wind turbines, two 1.2 MW fuel cell and a 1.2 MW solar PV array [7]. The structure ensures that there is a constant and reliable supply of electricity available to sustain the 3 MW daily load requirement of the prison, and is predicted to save Alameda County approximately $100,000 per annum in energy costs as a result of the grid's energy storage capabilities and reduced network down-time [6].

Another body that is benefitting from the installation of smart grid systems is the US Military. There are legislations in place, such as the Energy Policy Act of 2005, which require fixed army installations to meet predefined targets, and deployment of smart grid technology can help achieve these objectives. The targets include improving levels of self-sufficiency, increased use of renewable sources, and measureable energy security: the surety, survivability, supply, sufficiency and sustainability of the resources [8]. Microgrid technology can provide a means for the base to operate critical loads for extended periods of downtime when disconnected from the centralised grid.

In May 2013, Fort Bliss Army installation in El Paso, Texas began utilising a grid-connected microgrid, comprising a 120 kW solar PV array with a 300 kW energy storage battery system. Major Joe Buccino stated that in the wake of emerging threats, its ability to operate off-grid is an absolutely invaluable asset amidst any commercial utility blackout or cyberwarfare which can cause resultant loss of power [9].

The features of the microgrid also deem it an invaluable asset in the midst of an emergency situation such as a natural disaster, as it presents an effective approach to reducing the consequences of any damage seen to the electrical utility, by providing off-grid power supply to aid workers and civilians [10]. Renewable resources and mobile generators are well equipped for facing situations where conditions cannot be predicted and there is insufficient time available for comprehensive infrastructure planning [11]. The installation of distributed generators means that the immediate needs for medical equipment, lighting and power for rescue missions, and the establishment of a reliable communications network can all be addressed. This will also support the restoration efforts to the local power grid [10].

The recent Typhoon Yolanda disaster suffered by the Philippines is an example of a scene of destruction whose recovery efforts have been boosted by the installation of 160+ distributed generators, which are providing key power facilities to aid the efforts of rescue and relief workers. Meralco, the largest utility in the country, deployed 40 generator sets [12] and JCB sent 120 in order to provide electricity to community buildings such as hospitals and local authority premises, whilst vital repairs and restoration works are carried out to the damaged electrical facilities [13].

2.2 Strathclyde University Microgrid

The Distribution Network Automation and Protection Laboratory comprises a 100 kVA microgrid which has a number of advanced controllers and can be split into three power islands, and which can also be operated in either grid-connected or islanded modes. Figure 3 shows the basic overall set-up of the grid, where the coloured circles indicate the different power islands.

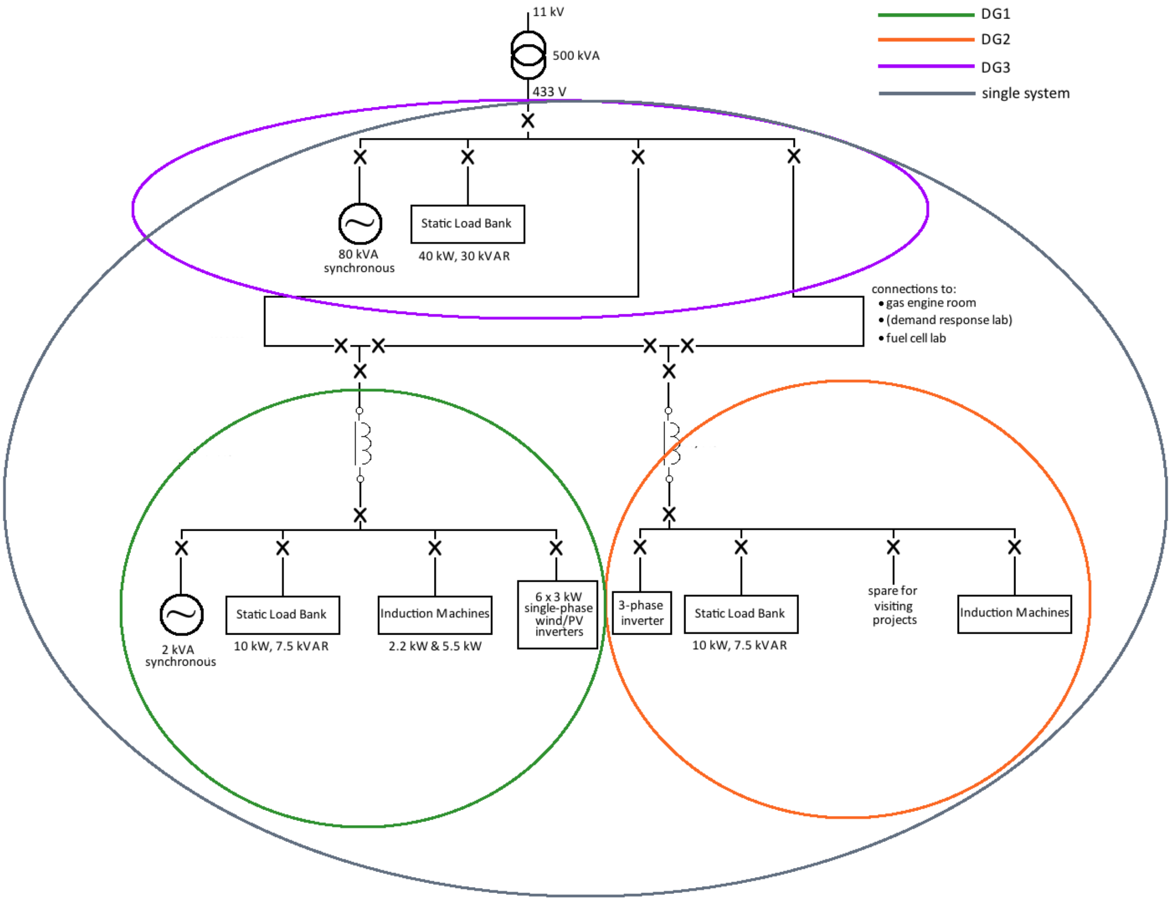


Figure 3 - Schematic of Microgrid Laboratory Set-Up

The advanced control system was custom designed and actively reads, processes and logs a large number of measurements that can then be monitored and analysed [14]. The microgrid is suitable for smart grid applications, and produces an extensive amount of data in real-time. The laboratory is integrated with a real-time digital network simulator and protection injection laboratory [15]. It is also set up to support the testing of various communication protocols, some of which include DNP-3, IEC 61850, IEC 61400, as well as associated server software such as OPC. OPC defines standard objects, methods and properties for servers of real-time information, such as smart grid devices, which are then analysed in order to ascertain that such servers contain standard OLE compliant technologies [16].

The facility offers three primary modes of operation: hardware-in-the-loop simulation; pre-set scenario playback; and direct grid-connected/islanded system operation [17].

2.2.1 Control Rooms

There are two control rooms within the University of Strathclyde's microgrid facility, CR1 and CR2. CR1 contains the user interfaces and computer controls and is the main site for control and operation of the microgrid. CR2, where the described project is focused, contains a number of servers and monitors by use of which the smart grid control room shall be realised. Another control method that facilitates the automation of certain practices is the use of the two RTS units, which are also housed within the microgrid suite, named 'Charon' and 'Larissa'. Each has its own array of analogue and digital outputs [18].

2.2.2 Equipment Overview

The key equipment that makes up the microgrid facility in the University is detailed in Table 1. Details of the important components shall be elaborated upon in the upcoming section.

Table 1 - Microgrid Equipment Overview

Microgrid Component	Location
DG3 Switchboard	Purple Island
DG2 Switchboard	Orange Island
DG1 Switchboard	Green Island
DG1 Switchbox	Green Island
DG2 Switchbox	Orange Island
DG1 Subsidiary Contactors to Induction Machines	Green Island
DG2 Subsidiary Contactors to Induction Machines	Orange Island
LV Board 1	CR1
LV Board 2	Microgrid Enclosure
80 kVA Synchronous Motor Generator	Purple Island
4 kVA Synchronous Motor Generator	Green Island
10 kVA Inverter	Orange Island
40 kW + 27 kVA Loadbank	Purple Island
2 × 10 kW + 7 kVA Loadbanks	Green / Orange Island
Siemens S120 Drives	LV Board 2
Transmission Line Simulator Inductor	Microgrid Enclosure

2.2.2.1 LV Board 1

The equipment inside the microgrid enclosure and in CR 1 is supplied by 415 V 3-phase AC via a 600A MCCB from a 500 kVA transformer. LV board 1 provides an isolation point for the equipment housed within the microgrid laboratory, and when the grid is not in use, all fuse isolators are in the open position, and locked. The equipment supplied by LV board 1 is shown in Figure 4.

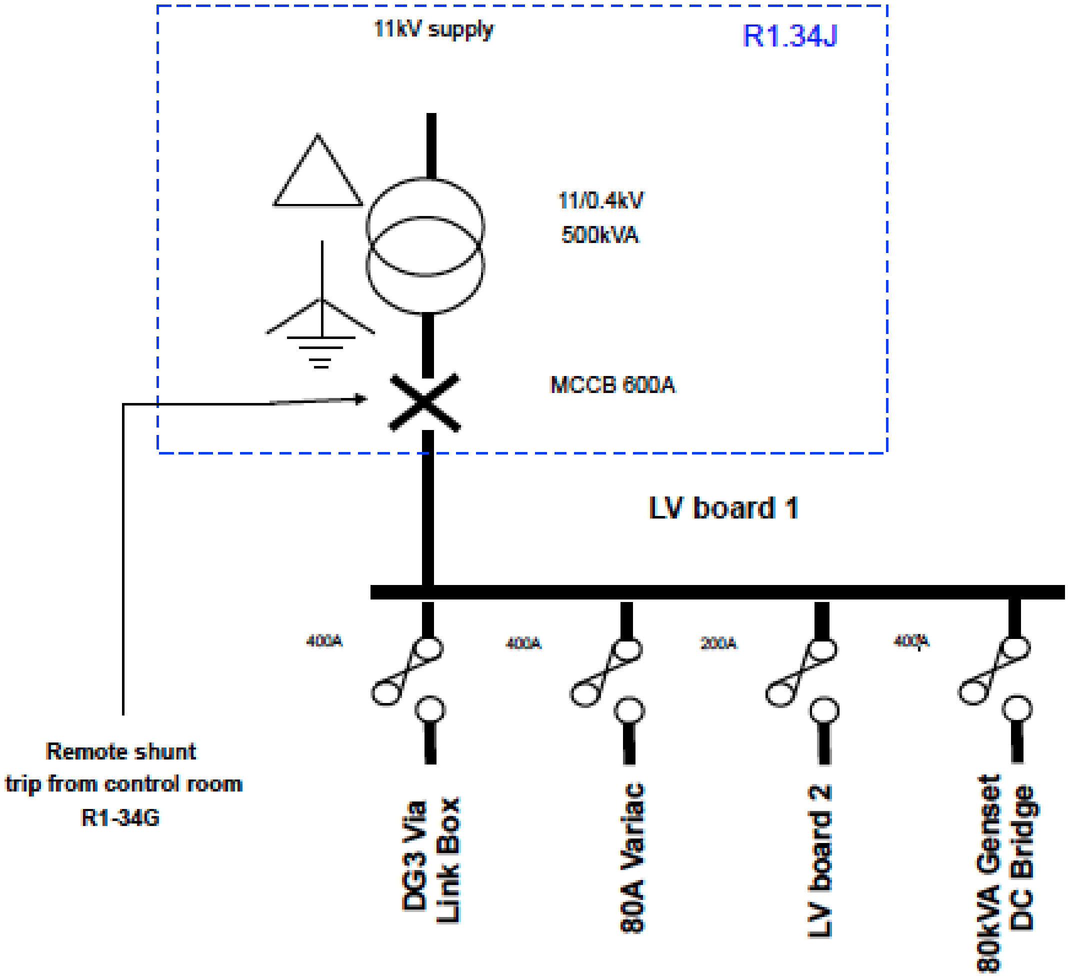


Figure 4 - Schematic of LV Board 1 [18]

2.2.2.1.1 Switchboards & Switchboxes

By closing the fused isolators, any of these components can be connected to the grid. The purpose of the switchboards is to configure the microgrid; it can be connected up in a large multitude of arrangements to simulate different scenarios, and it is the switchboards that facilitate microgrid control by providing control voltages to the generator sets. The DG1 and DG2 switchboxes are the drivers that enable the interconnection of the three power islands. Contactors facilitating interconnection can be operated manually or controlled via the RTS as previously mentioned. Schematics detailing the connections associated with each of the switchboards can be viewed in Figure 43 and Figure 44, Appendix A.

2.2.2.1.2 80 A Variac and 80 kVA Generator Set (Genset) DC Bridge

The 80 A variac is located outwith the microgrid enclosure, but its output is fed into the microgrid, acting as a variable voltage source in order to satisfy testing purposes. The power requirements also require that the 80 kVA supply for the DC Bridge and DC Motor is located on LV Board 1. The 80 kVA genset shown in Figure 5 comprises of the DC bridge circuit, which is a thyristor drive that controls the DC drive motor for the 80 kVA synchronous generator set in DG3; field control units located in CR1; a rotary exciter in the form of a dynamo and phase induction drive motor; the 80 kVA synchronous generator itself; and an isolating transformer.

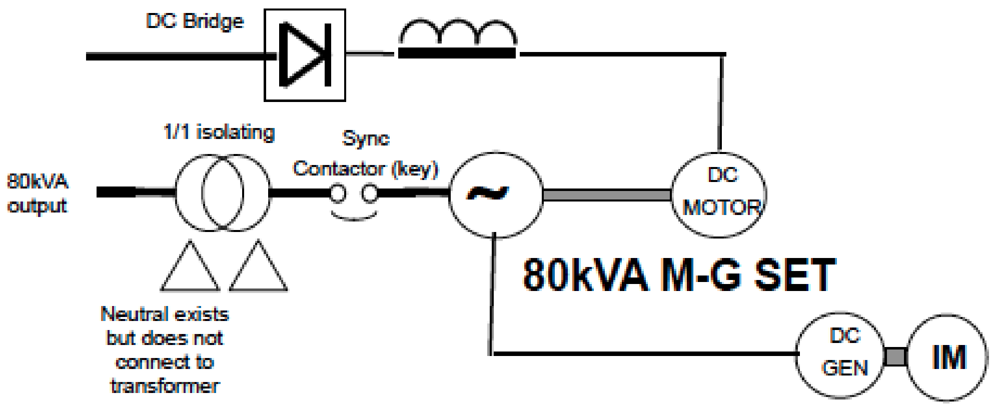


Figure 5 - Schematic of 80 kVA Generator Set [18]

2.2.2.2 LV Board 2

LV Board 2 is located inside the microgrid enclosure and serves as a power distribution board, consisting of various sockets, a 200 A MCB distribution board, Siemens Motor Drives and induction load/regenerator bus, 20 kW / 1 kV DC supply, Windy Boys (inverters to simulate wind energy power plants), on/off switch for the 80 VA synchronous generator exciter, a 2 kVA genset variac, and two spare connections, as shown in Figure 6. LV Board 2 supplies and protects the microgrid equipment located within the enclosure [19] and this will be detailed further in Section 2.2.3.

Sockets available on LV Board 2 are of sizes 13 A, 16 A and 32 A. A detailed view of the 200 MCB distribution board is shown in Figure 45, Appendix A. It determines which components are connected and disconnected from the system. The characteristics of the MCBs in use correspond to the load being supplied, and any MCB not in use is set in the 'off' position.

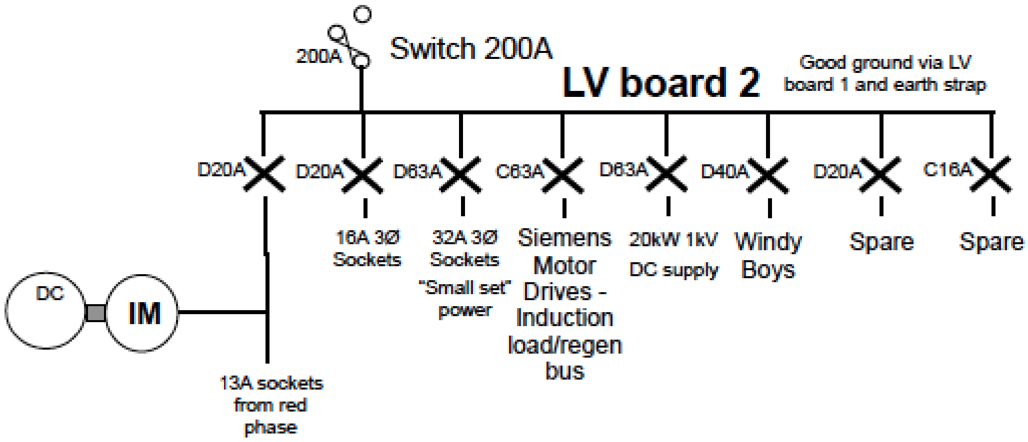


Figure 6 - Schematic of LV Board 2 [18]

2.2.3 Protection

As discussed in Section 2.1.1, the protection of a microgrid is a complex problem presenting a number of issues to be addressed. In the case of the university microgrid facility, there are a variety of protection approaches in place to ensure the security of supply and equipment. As well as the hardware protection (i.e. fuses and MCBs as already discussed) there are various software protection schemes in place. The LPC software core is host to overcurrent relays, synchronising, reverse power protection, under/over-voltage and under/over-frequency relays, all of which serve as protection approaches for the microgrid.

2.3 SCADA

SCADA is an acronym for Supervisory Control and Data Acquisition, and is essentially a computer system that is used for the control and monitoring of real-time data from a centralised location [20]. SCADA systems first came to fruition more than 50 years ago, and are used in a variety of different environments which include electrical power generation, transmission and distribution, as well as in manufacturing systems and even the monitoring of traffic lights [21].

SCADA systems generally consist of a series of subsystems, comprising of RTUs, SCADA master station computer system or MTU, HMI, alarm sensors and a communications network [22]. One of the main benefits associated with the use of SCADA systems is the ability to gain an instantaneous knowledge of system performance. This can reduce costs while simultaneously improving the safety of a system, as faults can be immediately detected. In electrical power systems, faults can result in loss of generation and the longer the downtime of the system, the more money the system operator will lose. The use of SCADA systems can immediately pinpoint the position and type of fault, meaning that it can be isolated quickly. This knowledge could also reduce the potential risk to whoever is working to eradicate the fault, especially in such a hazardous environment.

The acquisition of data occurs at sensors and this data is gathered by the RTUs in the system. RTUs, housed in microcontrollers, are connected to the different types of sensors, and these sensors measure different types of data depending on their circuitry. For the university's microgrid, examples of variables measured include frequency and the current state of a variety of switches in the circuit. Sensors can measure either analogue or digital values. Numerical values like frequency would be measured using an analogue sensor and then converted using an analogue-to-digital converter in the RTU for use in the control room. Conversely, digital values can only be digital 0 (off) or digital 1 (on), meaning that the state of the switches will be measured using digital sensors [23].

The data must then be sent to the central computer system, or MTU, over a communications infrastructure. This must happen in real-time, and the data must be stored in order to be accessed either in real-time or historically in order to evaluate trends or for comparison. The complexity of the communications architecture will vary depending on each individual SCADA system, but can generally include wired or wireless LAN or WAN connection. The University's microgrid communications network is shown in Figure 46**,** Appendix A**.**

Once the data has been stored in the MTU, it must be converted into information in a form that allows for the real-time monitoring, or control, of the system. This system is called the HMI, and is usually presented graphically in the form of a GUI. A range of visualisation techniques can be employed in an HMI, which are presented in Section 2.4. Alarm systems can be included in HMIs, which could be triggered by a number of different scenarios. In terms of the microgrid operation, alarms could, for example, trigger when frequency in the system drops too low or rises too high, as both of these situations could potentially be catastrophic for the system.

2.4 Visualisation of Data

Visualisation of data is primarily the most important element to consider when designing a control room, due to the fact that the data is ultimately what is required to interpret useful information from the grid. There are an innumerable amount of methods for displaying and visualising data, which can be split into two main categories: quantitative and qualitative. Quantitative data represents any value that can be numerically measured, whereas qualitative data represents any descriptive information that may be necessary to define the data. In a conventional control room, deciding on what data should be displayed is crucial, as the information displayed has to be as clear and concise as possible for the persons viewing, to ensure that any alarms or faults in the system can be identified promptly. There is no "set way" for displaying control room data, although there are preferred representations. In order for a control room HMI to be used, its benefits should exceed the workload it adds on the operator or user [24]. As the customer would like the visualisation to impress visiting guests to the university, as well as existing faculty members and students, the focus on precise colour schemes for optimal usage will be of less importance; relevant information will still be required to be concise.

To get an idea of existing control room design, and design preferences amongst control room engineers, two control room visits were organised. One of the visits was to the GSE Systems Inc. control room simulator within the University of Strathclyde. The realistic simulation is reminiscent of current existing control rooms, which gave an idea of how to display data, from colour schemes to thickness of lines, to relevant data that should be displayed on screen for the user. The second visit that was organised was to the Scottish Power Operational Control Centre, near Kirkintilloch, Glasgow. This visit was both fascinating and advantageous in developing further knowledge of control room design and visualisation methods.

Control room applications have one main purpose - to display all the important information upfront and keep the user informed about the system’s state. Additionally, SCADA systems can allow the user to control certain parts of the system. HMI for such systems should be designed to conform to the requirements of the system in question and its application [25].

The first instance reported of visual representation of a power system was in 1993, by Mahadev et al. [26], and then approximately seven years later, Weber et al. documented their own version of appropriate visualisation methods and techniques [27] [28] [29]. These reports consisted of: Visualisation of Power System Data, New Methods for the Visualisation of Electric Power System Information, and Voltage Contours for Power System Visualisation. Further effective solutions on human capability for detecting visual patterns, have since then been seldom [30].

One of the main visual displays that will be shown is a real-time schematic of the grid using Siemens WinCC software, which will be discussed in greater detail in Sections 3.2 and 5. To please visiting guests to the university, as well as students and faculty members, who may not be familiar with microgrid technology or understand the jargon associated with electrical power studies, there shall be displays that present the data in a manner that will appeal more to their interest. As previously communicated, there are various different methods and techniques that could be used for visualisation. This section will discuss a few possibilities, although Sections 2.4.2 and 2.4.3 will give a better idea of the information that will be displayed via these techniques.

Upon researching data visualisation techniques, there are a variety of intriguing, colourful diagrams and simulations of which a few are listed: animated arrows, 3D mapping, coloured lines, colour contours, line thickness, and pie charts. Figure 7 shows a selection of some of these techniques.

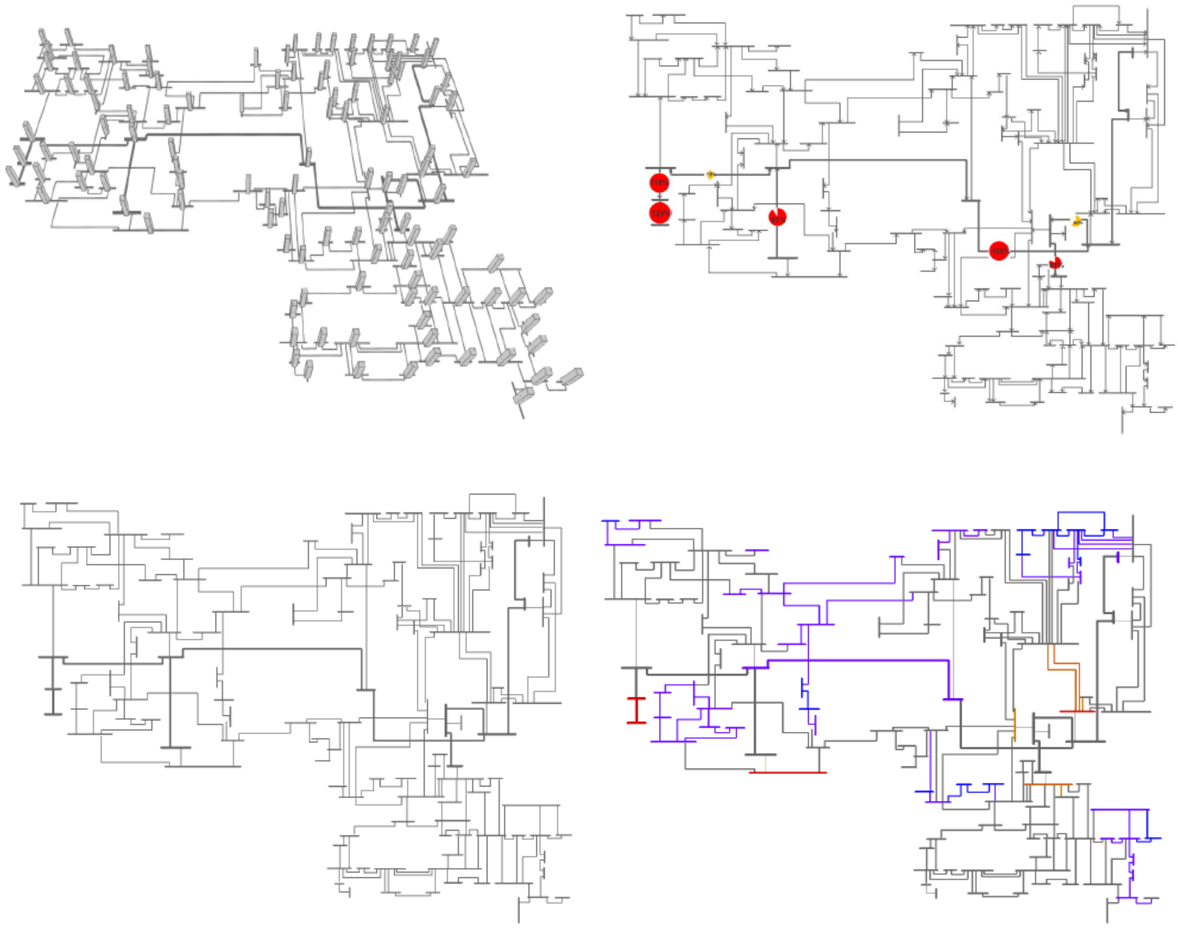


Figure 7 - 3D Visualisation (top left), Pie Charts Representing % Loading (top right), Varying Line Thickness (bottom left), Coloured Lines (bottom right) [30]

2.4.1 Existing Visualisation Techniques in the Power Industry

What follows is an introduction to some advanced visualisation techniques that can be incorporated into the final GUI design to display necessary SCADA data in an intuitive and eye-catching manner. There are alternative visualisation software programs and techniques available through online resources that can be used in tangent with WinCC to optimise visual outputs. As this is a research project, the final design should include visualisation techniques that are not commonly used, thus following a project motivation – to aid in the next-generation design of control rooms for microgrid networks.

There are companies that adopt different styles of visualising data. One of these companies, Statnett, is responsible for maintaining the balance in the Norwegian power system and has the overall supervisory responsibility and physical control with regards to the aforementioned power system [31]. The company website displays critical information on the system's performance by outputting real time data in different quantitative manners. In-depth analysis of generation, including the operational capacity of multiple generating techniques, and import/export data is represented. Historical and current frequency ratings are also available in the public domain. Some of these techniques can be incorporated into this project.

Frequency is fundamental in power system analysis; it determines the balance between generation and demand (load) for a given point in time. Frequency, over time, can build up historical trends to better the understanding of peak demand and seasonal changes in power usage. The instantaneous frequency is displayed on the Statnett company website. The graphical dial in Figure 8 displays the current Nordic power balance, which should fall between the steady frequency zone of 49.95 – 50.05 Hz. Danger areas, when the frequency is deviating by ±0.05 Hz from nominal, are highlighted in red. This design could be incorporated into the data output of frequency for the microgrid. For historical trends, Figure 9 is a worthy example of how frequency changes over time, this interactive representation allows the user to zoom in on any point of interest.

Figure 8 - Statnett Real-Time Frequency Visualisation [31]

Figure 9 - Statnett Historical Frequency Visualisation [31]

Another technique would be an interactive line graph representing the power production and consumption over a set period of time. This can be updated with new data during scenario simulations. The interactive feature of Statnett's version allows the user to physically zoom in on specific periods for the exact production and consumption MW/h value. This is illustrated in Figure 10 with the right-hand section of the figure indicating exact values of generation and demand.

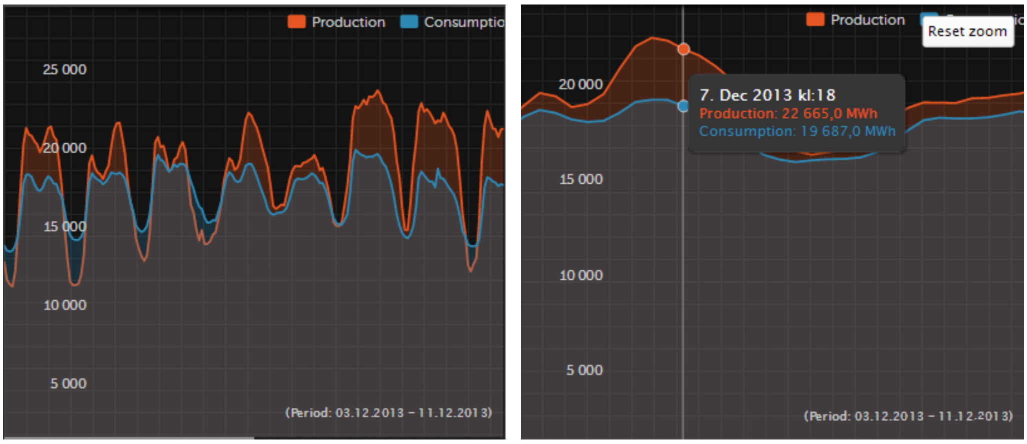


Figure 10 - Production/Consumption Visualisation [31]

2.4.2 Advanced Visualisation Concepts

Zhao Kaidi states in his report on the investigation into new data visualisation techniques that data visualisation is based upon understanding ratios and relationships amongst a set of numbers. It is a way to deliver data by shifting the focus from straight numerical reasoning to visual reasoning, where the user can study relationships between numbers by way of changing colours or sizes rather than, for example, numerical increments [32].

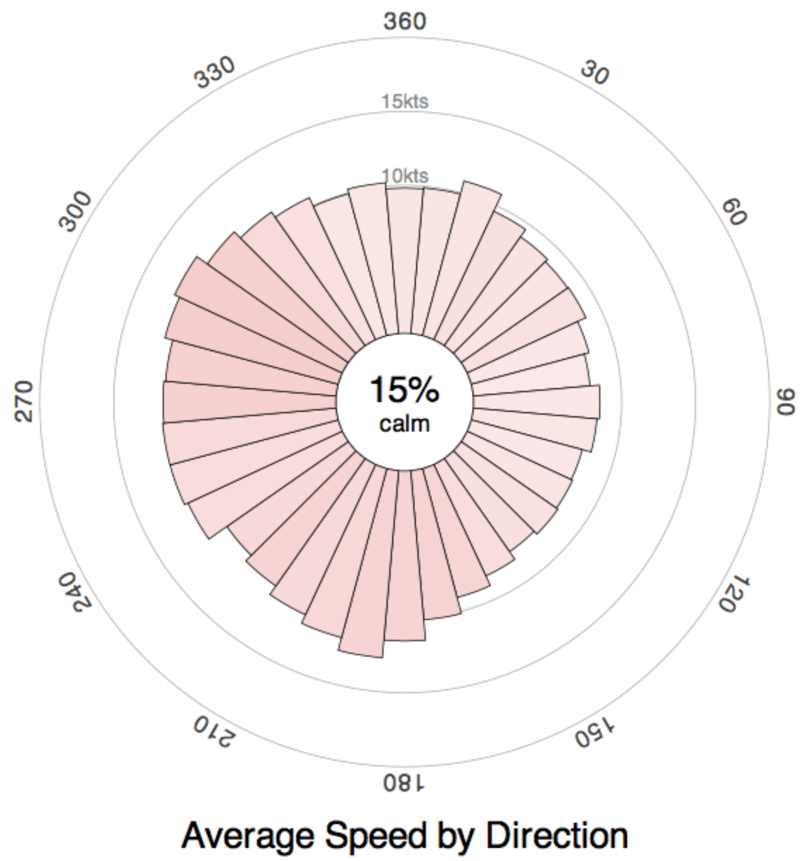


Figure 11 - Data Visualisation Technique used by Wind History [33]

The data capture method shown in Figure 11 puts this visual reasoning into practice, and is an example, used by the Wind History website, of monitoring a parameter by tracking two key measurands; in this case, speed and direction of wind [33]. In relation to the project objectives, in addition to Statnett's frequency dial, this visualisation technique could be used to present potential method of monitoring frequency, this time over a set period, i.e. a minute or an hour, where the two key measurands are time and frequency. An outer labeled contour would act as the time axis, and the inner levels would be in increments of 0.5 Hz, ranging from 49 Hz up to 51 Hz, the ultimate values that frequency should not exceed in order to avoid risk of power outages - a lesson learned during the visit to the Scottish Power Operational Control Centre. A colour code can be incorporated in order to indicate whether the system frequency is in steady state, warning state, or alarm state (below 48.5 Hz or above 51 Hz). Figure 12 is an initial draft of how the group envisages the frequency display after adapting Wind History's design to achieve the above specifications.

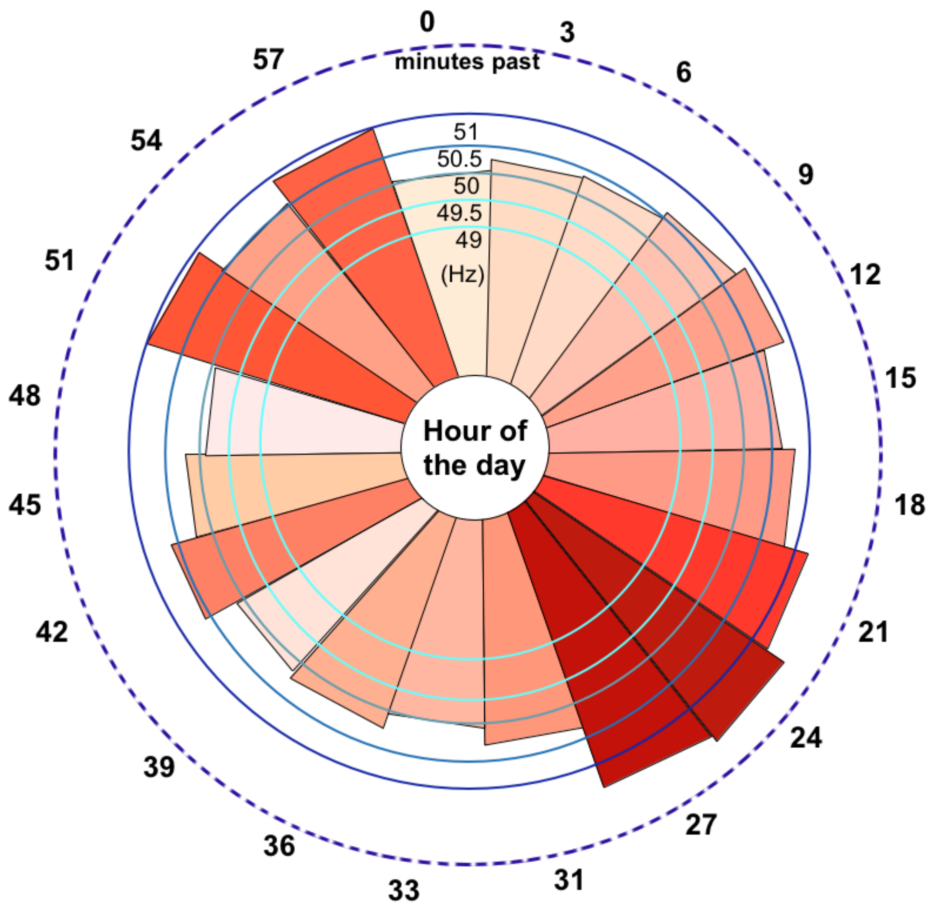


Figure 12 - Initial Concept Design of Frequency Display

Also akin to the Statnett frequency dial are the Gauge Charts which can be created using Google Developer's Visualisation Playground [34]. Examples of these are shown in Figure 13. Animations can be configured to move the dial when parameters oscillate. Along with a large range of other customisable aspects, this could deem Google Charts to be a viable and innovative way to display power system measurands such as frequencies, power flows and voltage levels.

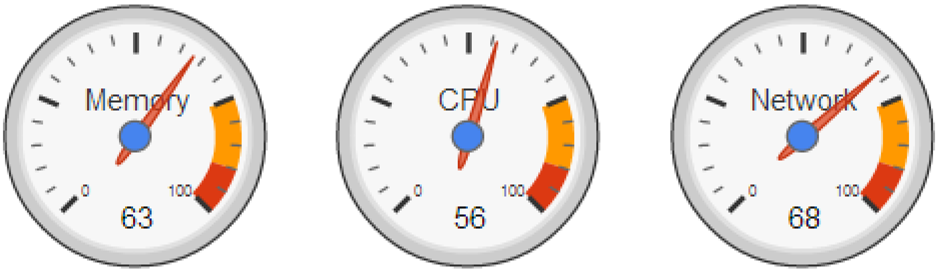


Figure 13 - Gauge Charts Created Using Google Developers Application

The Google Developers application can also facilitate an extensive range of programmable, customisable data visualisation techniques in the form of a multitude of different types of charts and maps. An innovative idea could be to create "donut" pie charts such as the one shown in Figure 14, which could appear above a component of the microgrid when the user hovers over it with the mouse. The component could remain visible through the hole in the centre of the chart, and, where in Figure 14 percentages are shown, the chart in the GUI design could display relevant parameters relating to the component in question.

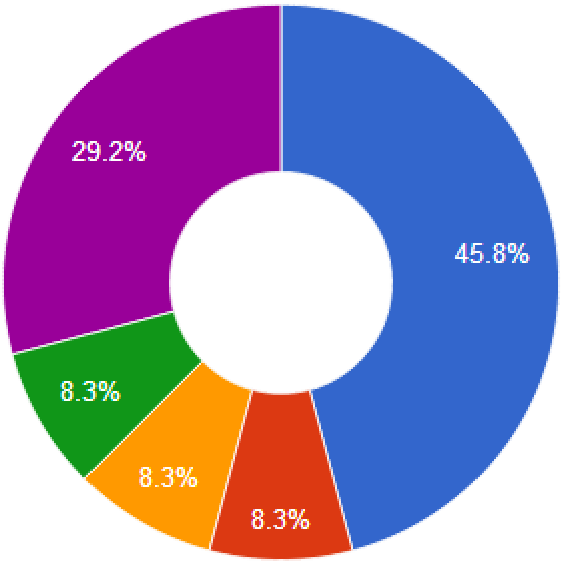


Figure 14 – "Donut" Pie Chart Visualisation Technique

2.4.3 Available Software Packages to Enable Advanced Visualisation Techniques

As aforementioned, in addition to WinCC, there are a number of available software packages which may be more suited to the advanced level of data visualisation techniques. Two packages for implementing advanced data display techniques which have become apparent during research are D3 and Bokeh.

D3 is a javascript library that allows data-drive transformations to be applied to a document. It can be used to display data ranging from simple bar graphs, to tree maps, and other more compound interactions. D3 is capable of producing very complex data visualisation outputs [35].

Bokeh is an interactive visualisation library for large datasets that uses the latest software technologies. It uses Python coding to achieve the goal of providing elegant, concise construction of graphics in the style of D3, while delivering high-performance interactivity over large data ranges [36].

3 Software & Hardware Review

3.1 MATLAB 2013a

Before the design of the control room GUI could be properly planned, it was of paramount importance to analyse the historical data available in order to gain a sound grasp of how the microgrid operates under different scenarios. The data collected is saved in CSV format and is collected every 2 milliseconds, meaning that even as little as two minutes worth of data amounts to 60,000 different measurements taken for each individual variable, which in total equates to just under 17,000,000 records of data. The most efficient way to analyse the data was to use MATLAB 2013a software. MATLAB is a high-level programming language that enables quick and easy numerical manipulation of data in comparison with other programming languages such as C/C++ and Java [37]. It can also be utilised for the creation of GUIs. The software is particularly powerful when handling large data files, and was the ideal choice for working with CSV files.

3.2 Siemens WinCC 7

The control room for the microgrid will be designed using Siemens SIMATIC WinCC software. WinCC is a SCADA and HMI system, which allows users to construct, control and monitor user interfaces for a wide range of SCADA systems [38]. The software is not industry specific and could consequently be used in many applications other than microgrid visualisation, from pharmaceutical to water treatment. This software will be the main tool used in the project and will be the environment in which the final Control Room GUI will be designed.

WinCC is made up of a number of sub-programs to handle the various different aspects of the design, whether that be Tag Management, Graphic Designer or Global Scripts. Although there are many more aspects to WinCC, these are the three sub-programs that will be the main focus for the project. The Tag Management section will be used to house the various tags defined in order to display the data taken from the microgrid, whether it is historical or real-time. The tags define each variable measured by the SCADA system as an individual variable that can then be used within WinCC. The Global Scripts will be used to write the C-Scripts that will read in the data and assign each variable to the correct tag. Finally, the Graphic Designer will be where the main control room design is carried out, with the construction of the microgrid GUI incorporating the tags that have been initialised by a C-Script. The Graphic Designer contains several elements, such as: I/O Fields, Buttons and Alarms that can be used to design the user interface. The final component of WinCC that will be used will be the 'WinCC RunTime' which is used to 'run' the control room user interface and will be the final output from the completed system.

3.3 Microsoft Visual Studio 2010

As mentioned in the previous section, WinCC will require C-Scripts to be written in order to update the tag values. Although WinCC provides a C debugger in the form of the GSC Diagnostics Applications Window, the process is slow and not well-suited to debugging code. Therefore the C-Scripts will firstly be designed and debugged using Microsoft Visual Studio 2010, which is a broad programming platform that supports a wide number of programming languages [39]. For the purpose of this project, however, the software will only be used to design and test C- Code before implementing the design in WinCC.

3.4 Communications Architecture

In order to successfully update the control room information, the communications architecture of the microgrid facility must be investigated. The laboratory contains a large number of servers and PCs, however only a small amount will be utilised by the project. Figure 15 shows a basic diagram of the communications architecture that will be used, however an extensive diagram of the entire laboratory architecture can again be found in Figure 46, Appendix A.

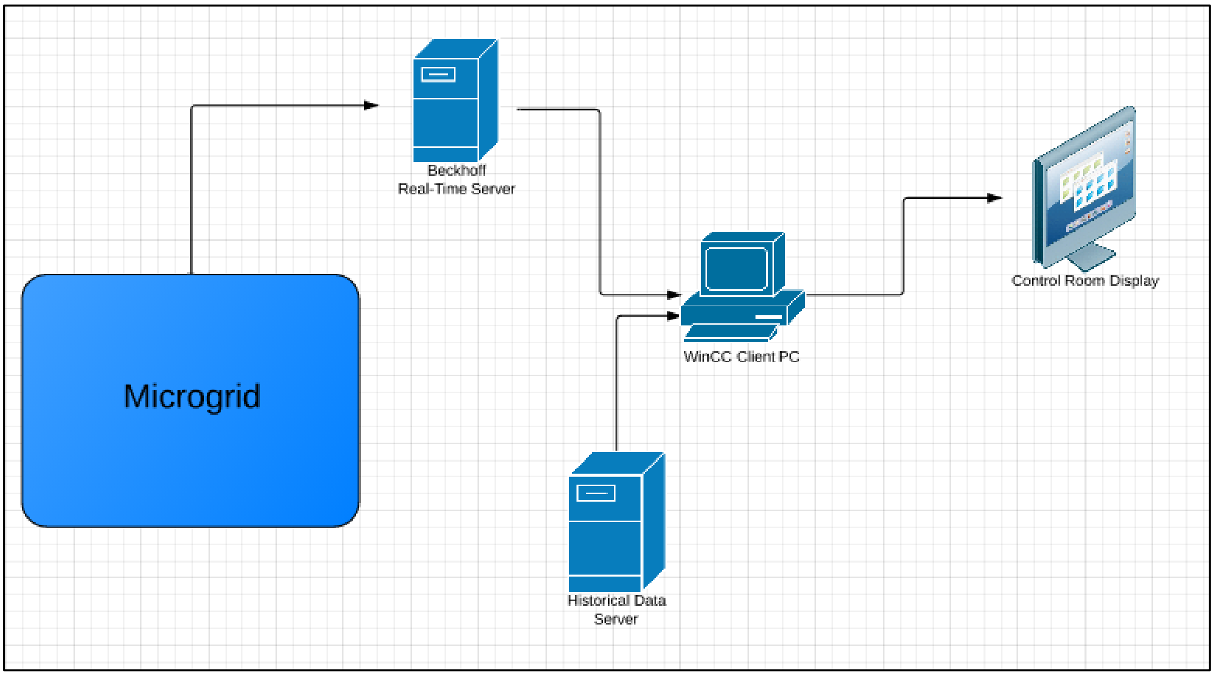


Figure 15 - Microgrid Communications Architecture

As can be seen Figure 15, the WinCC PC that will be used to run the control room programme will be connected to two servers. One will house the historical data while the 'Beckhoff' server, will read real-time data directly from the microgrid, which will then be read by the client PC. An LCD TV is connected to the client PC to display the final control room interface. Although the above architecture does not detail the entire communications set-up of the microgrid laboratory, it does show the architecture relevant to the project.

4 Historical Data

4.1 MATLAB Analysis

All MATLAB code, including comments, that has been used to produce the results in this section can be found in Appendix B. The names of the measured variables are firstly read in from the selected CSV file using MATLAB's built-in "xlsread" function. The code then reads in the numerical data using the "csvread" function. Limitations in the "xlsread" function's ability to read in very large arrays, as well as complications arising from reading in strings using "csvread", made it necessary to perform these two functions separately. The data is then sorted into four different groups corresponding to the power island in which the data was collected. This sorted data is then divided into inputs and outputs, and then sub-divided into Boolean, integer and double data types. The clearest way to present this data is in the form of a GUI, which was designed using MATLAB's GUIDE tool. The GUI design gives the user the option to plot any variable against time, as well as the additional option of plotting a comparison of variables against time. The variables are coloured according to their power island. An example analysis performed using the GUI is shown below in Figure 16.

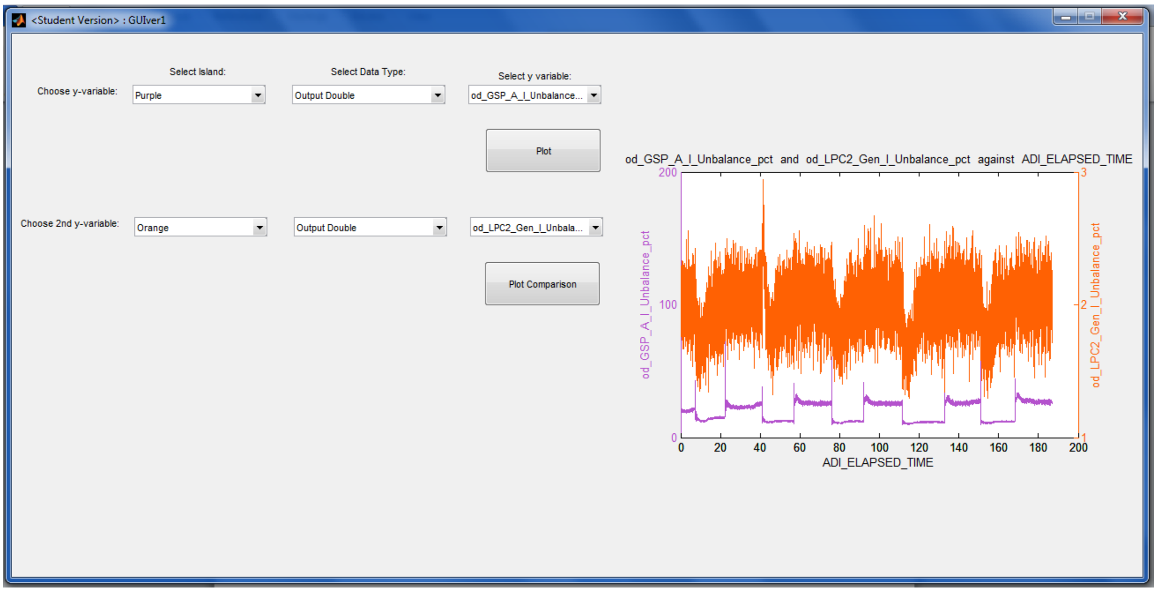


Figure 16 - Historical Data Analysis MATLAB GUI

4.1.1 Scenario One: Controlled Loadbank Power Drop at 80 kVA Synchronous Generator

This scenario is briefly explained to analyse the effect a drop in loadbank power has on the frequency levels within the microgrid system; the configuration of the switchboard at DG3/GSP is shown in Table 2. Switch 'A' connects the output of the DG3 common busbar to the input of the switchbox at DG1, thus leading to DG1 and (if required switch is 'on') DG2 power islands. It is noted that a state of 0 indicates disconnection and, likewise, a switch state 1 indicates connection.

Table 2 - Configuration of DG3 Switchboard

Switch	State	Additional Information
A	0	Disconnected from DG1 Switchbox
B	0	Disconnected from North Wall
C	1	To Loadbank (initial value of 30 kW, 22.48 kVAr)
D	1	To 80 kVA (60kW) SynGen
E	0	Disconnected from External Distribution Network
F	0	Disconnected from External Distribution Network

It is clear from this setup of switch states that this scenario investigates solely the effect of loadbank settings and resultant synchronous generator behaviour for DG3.

The loadbank, in star connecton, is initially set to 30 kW (22.48 kVAr) and is instantaneously dropped in 2 kW stages, there are 7 stages in total across the simulation time of 121.35s. Power factor throughout simulation is recorded as a constant at 0.8. A visual representation of the controllable load is illustrated in Figure 17.

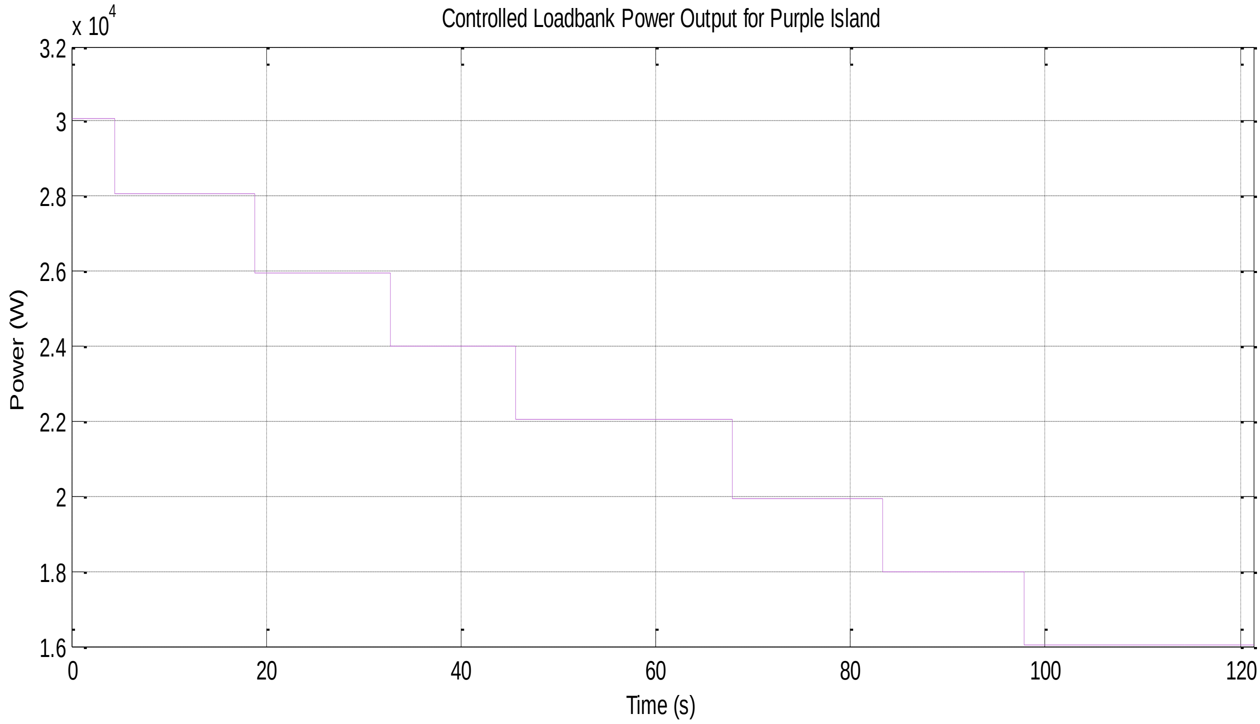


Figure 17 - Controllable Loadbank at DG3 Power Island

The effect that this loadbank has on voltage measured across the busbar at DG3 and the time taken for the system to reach stability is recorded in Table 3.

Table 3 - Voltage Response Values due to Loadbank Power Drops

Drop in Loadbank Power (kW)	ΔV at Purple Busbar	V_{peak} (V)	Time Taken to Recover (s)
30-28	+9.1	388.6	6
28-26	+8.1	395.8	2
26-24	+6.3	403.5	1
24-22	-3.6	406.7	13
22-20	+0.3	400.7	1
20-18	-0.1	404.2	2
18-16	+0.1	400.6	2

The peaks in voltage occur almost instantaneously after each 2 kW drop in loadbank power. After the 24 kW – 22 kW drop, the voltage levels no longer step up, and begin to level off (at an average value of 398.4 V). Thereafter, the voltage experiences an initial surge, but quickly recovers to this average voltage value. This is better represented in graphical form in Figure 18.

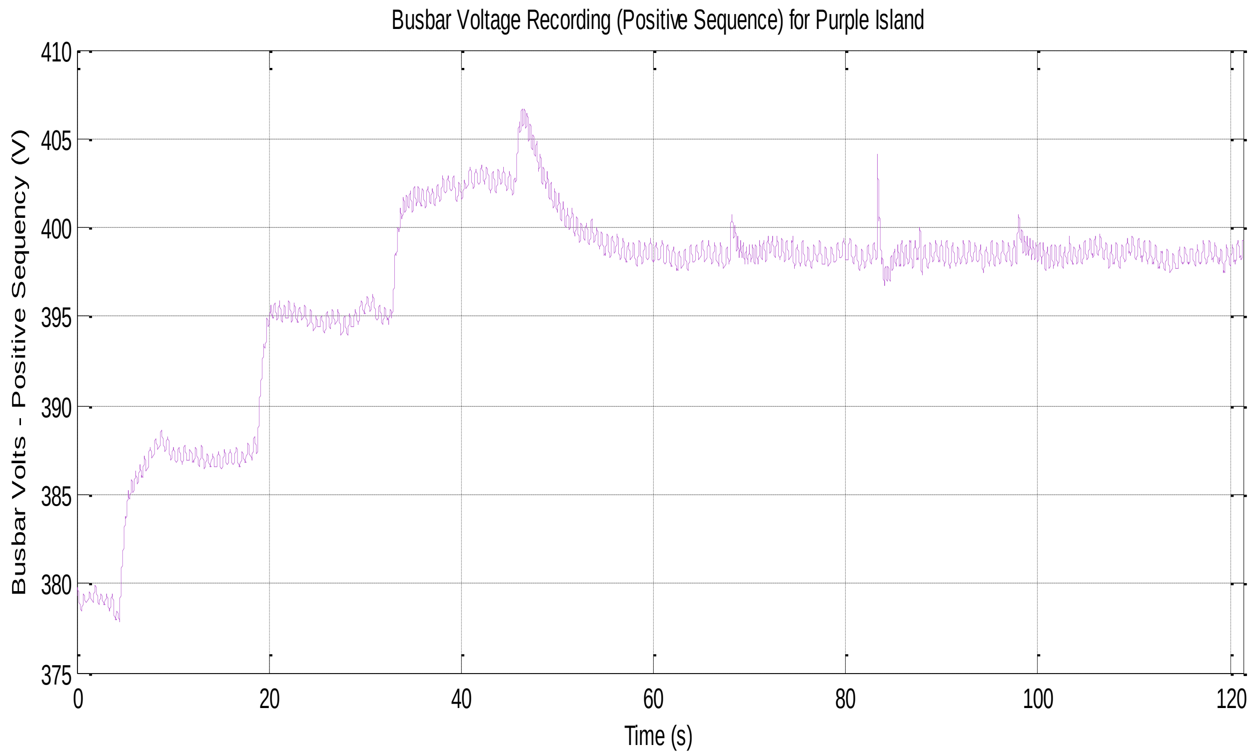


Figure 18 - Voltage Levels at DG3 Busbar

There are correlations between this voltage response and that of the system frequency: as system frequency rapidly drops, the voltage increases. For comparison, the system frequency is illustrated in Figure 19.

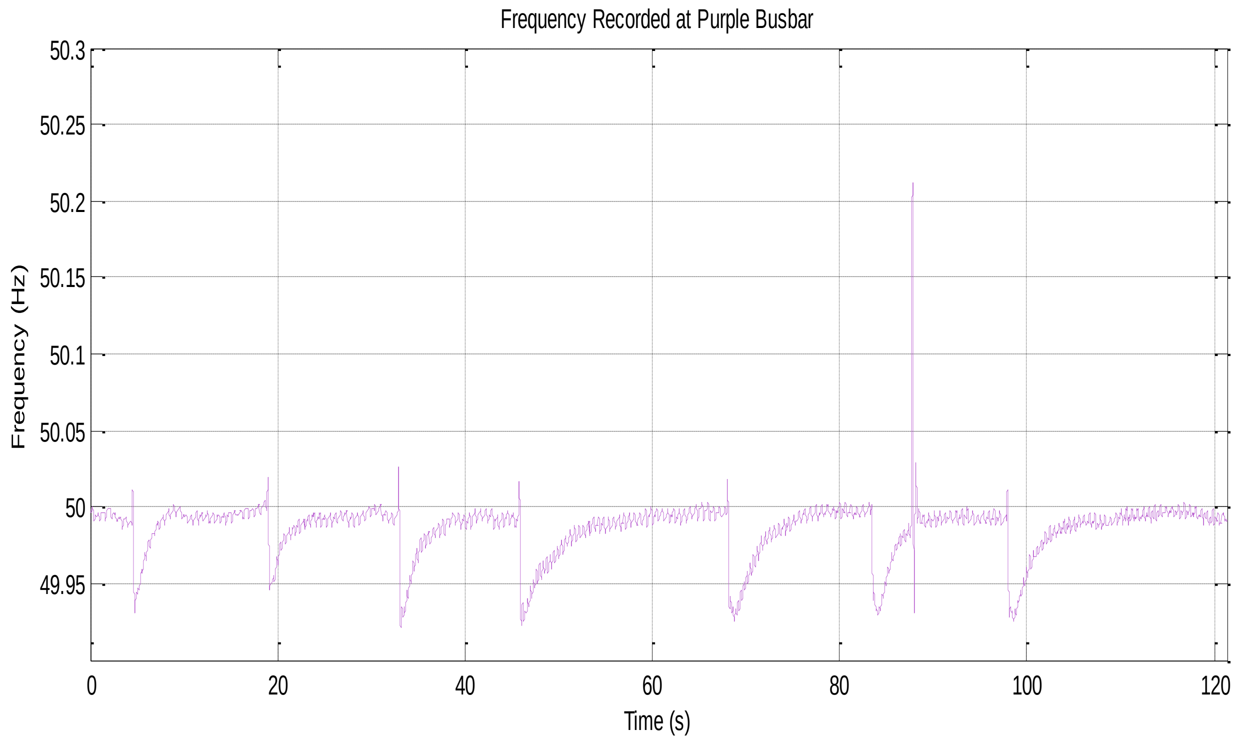


Figure 19 - System Frequency (Measured at DG3)

There is a significant surge in frequency levels at approximately 90s into simulation; this matches the surge in voltage at the same time. From principles, the frequency of power systems in the UK should oscillate typically around 50 Hz. When there is a mismatch between demand (load) and generation (synchronous generator), the system frequency deviates. It is expected that when generation exceeds demand, frequency should increase, and, alternatively, if generation does not meet demand expectations, the frequency should fall. Figure 20 displays the output power ratings of the synchronous generator for the DG3 power island.

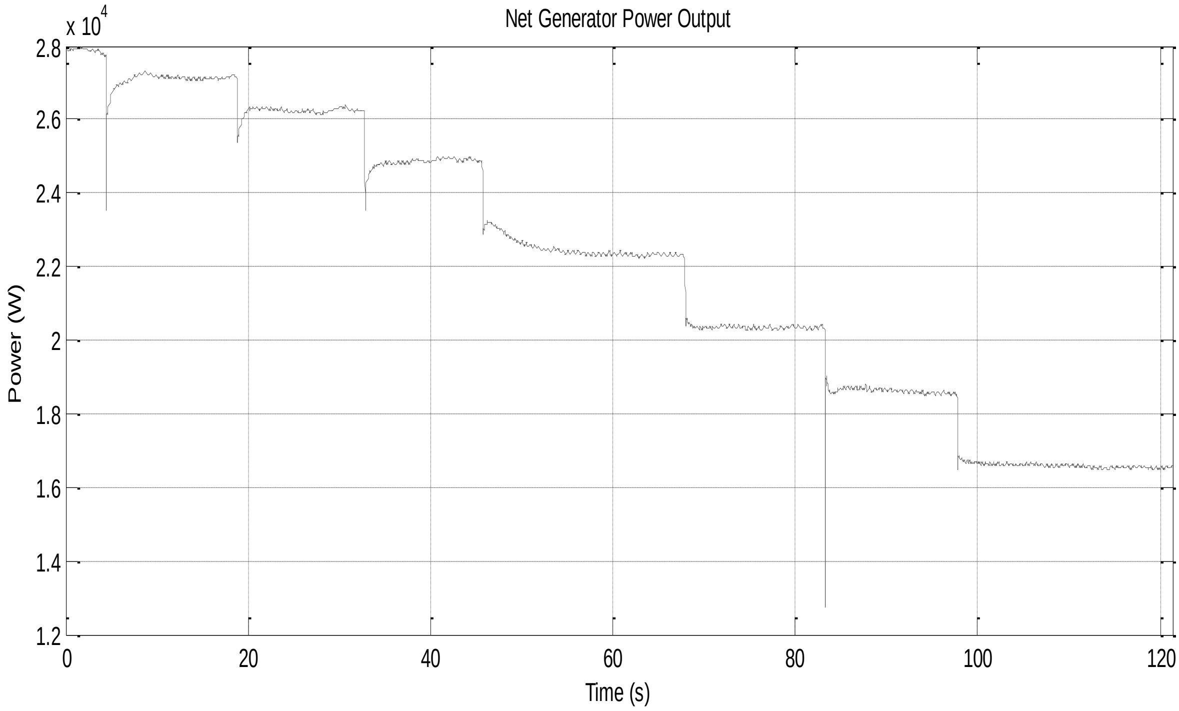


Figure 20 - Synchronous Generator Power Output

It is seen at 84s that there is an unconventional, instantaneous decrease in power generated, and the generator's response of accelerating may cause a delayed frequency surge (at 90s in Figure 19). However, an internal error in recording, by use of SCADA, may have caused this uncharacteristic result.

For each drop in loadbank value, the generator responds by slowing its rotation and producing less electrical power. The first loadbank drop in power is used in order to aid explanation. The loadbank power is reduced from 30 kW to 28 kW. The generator output power was initially 28 kW but when load is reduced, the power drops to below 24 kW, and thus demand exceeds generation. This clarifies first frequency drop from 49.98Hz to 49.93Hz. Comparing Figure 17 and Figure 20 during steady-state conditions, the demand is always higher than generation; this causes the frequency to oscillate below 50Hz throughout simulation.

4.1.2 Scenario Two: Sudden Connection of Generation and Loadbank Max/Min Changes – Frequency Analysis

From the study of archived results, it is assumed that this scenario is a study of the microgrid's performance, including frequency and power flow characteristics, due to the sudden connection of generation capacity and thereafter the changing of loadbank power rating, alternating from maximum and minimum values. The configuration of the DG3 power island is tabulated in Table 4, and shows that the DG1 power island is in operation.

Table 4 - DG3 Switchboard States

Switch	State	Additional Information
A	1	Connected to DG1 Switchbox
B	0	Disconnected from North Wall
C	1	Connected to Loadbank (initial value of 20 kW, 1.5 kVAr)
D	0	To 80kVA (60 kW) SynGen
E	1	Connected to External Distribution Network/BigGen Set
F	0	Disconnected from External Distribution Network

The value of switch states for the DG1 and DG2 power islands indicate that each power island is connected, shown in Table 5 and Table 6 respectively:

Table 5 - DG1 Switchboard States

Switch	State	Additional Information
A	0	
B	0	
C	1	Connected to "Smallset" 2 kVA Synchronous Generator
D	1	Connected to 10 kW, 7 kVAr, 12.5 kVA Loadbank
E	1	Connected from DG1 Switchbox to DG1 Switchboard
DG1 – DG2	1	Connected from DG1 Swtichbox to DG2 Switchbox
Machines	0	Disconnected from Dynamic Load Induction Machines

Table 6 - DG2 Switchboard States

Switch	State	Additional Information
A	0	Disconnected from Dynamic Load Induction Machines
B	0	Disconnected from Additional Spare Connection
C	1	Connected to 20 kW, 1 kV DC supply and 10 kVA Inverter
D	1	Connected to DG2 Loadbank (set to 8067 kW, 6441 kVAr)
E	1	Connected from DG2 Switchbox to DG2 Switchboard
DG2 – DG1	1	Connected from DG2 Switchbox to DG1 Switchbox
Machines	0	Separate Switches at Machines Open
DG2 to East Wall	0	East Wall Disconnected from DG2 Switchbox

As this scenario is more complex in its explanation of simulation and analysis of results, it can be divided into three separate power island discussions: DG3 (purple), DG2 (green) and DG1 (orange).

4.1.2.1 DG3/GSP (Purple) Power Island

Initially, there is no power generation at this bus as the generator is not connected and is not for any other generator source throughout the system. However, the synchronous generator is connected at 125s and begins rotating. Figure 21 and Figure 22, show the instantaneous generator power production due to this excitation and the frequency response upon start up at local busbar respectively. Figure 21 displays the generators' reduction of output power caused by the switching states of generating units at DG1 and DG2. As the 2 kVA synchronous generator and 10 kVA 3-phase inverter connect to the microgrid (and begin matching system frequency in the synchronous case), the 80 kVA synchronous generator at DG3 need not produce all the power required to satisfy load. Points of distributed generation connections are shown in Figure 21. Likewise, the frequency levels can be seen to increase at each inclusion of new generating units (135.2s and 159.5s) in Figure 22.

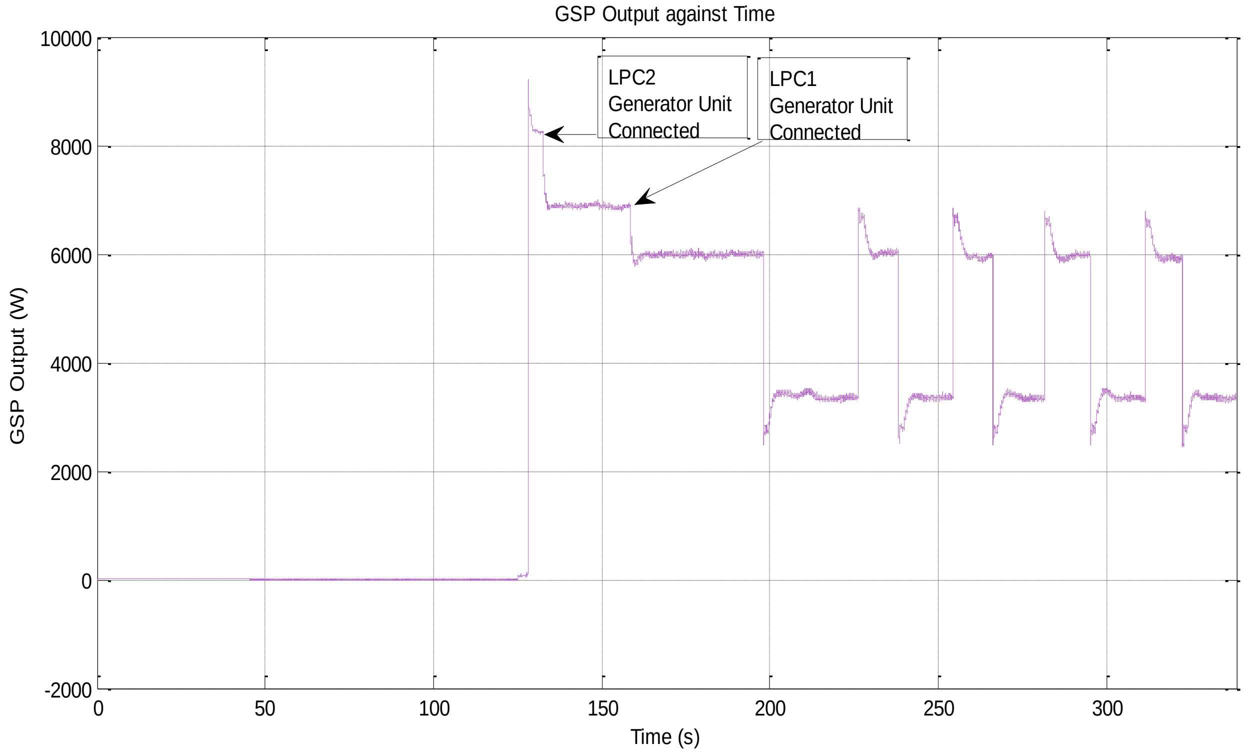


Figure 21 - Generator Power Output

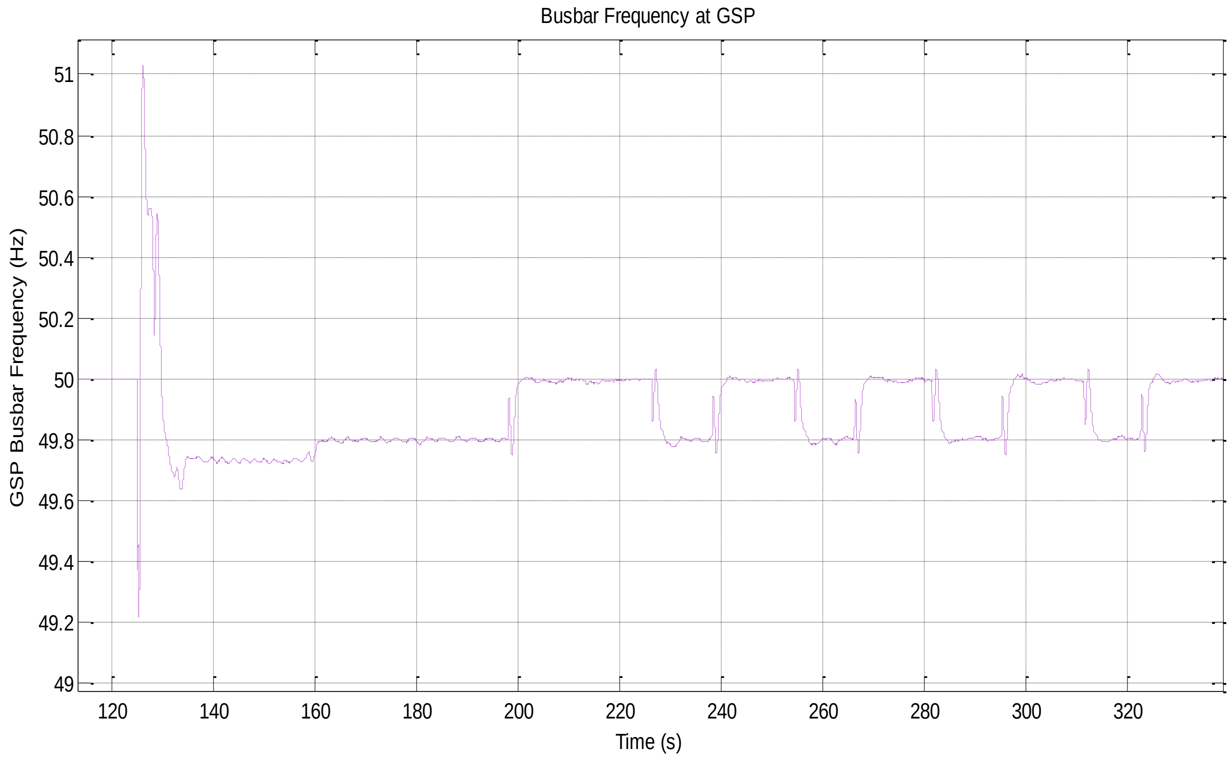


Figure 22 - Frequency Response at Busbar

The frequency experiences oscillations due to a surge of local power, before realising its steady state value at 49.79 Hz (below the reference value of 50 Hz, indicating demand of microgrid is greater than its generation capabilities).

This connection of the generator 'drives' current through the busbar, resulting in a voltage difference across it, shown in Figure 23, as there is resistive load connected, in the form of the 20 kW loadbank.

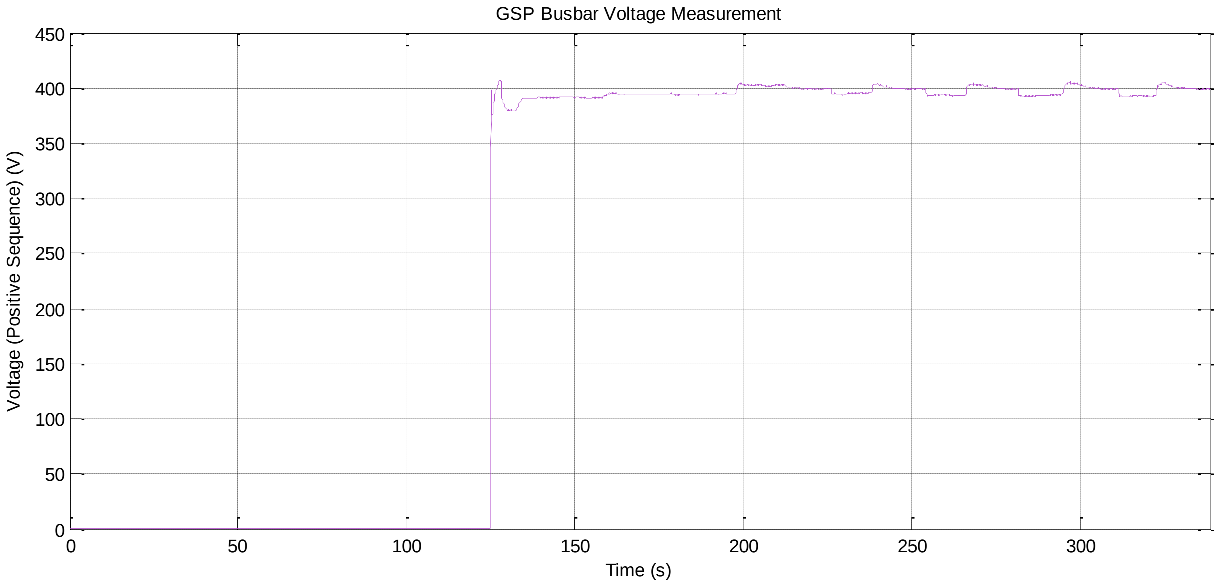


Figure 23 - Voltage Recorded across DG3 Busbar

Voltage oscillations occur due to start up, at around 125s, but after 200s, the voltage varies from 400 V (maximum grid voltage value is 433 V) to approximately 394 V. This is caused by the changing of demand in connected power islands.

4.1.2.2 DG1/LPC1 (Green) Power Island

The DG1 power island is home to the 2 kVA ("Small Set") synchronous generator which connects to the system at approximately 158.4s into the scenario and begins 'synching' to system frequency. Figure 24 and Figure 25 show the output power (both active and reactive) of the generating unit and the recorded frequency at this busbar with loadbank setting values. It is noted that the local busbar frequency matches that of the system.

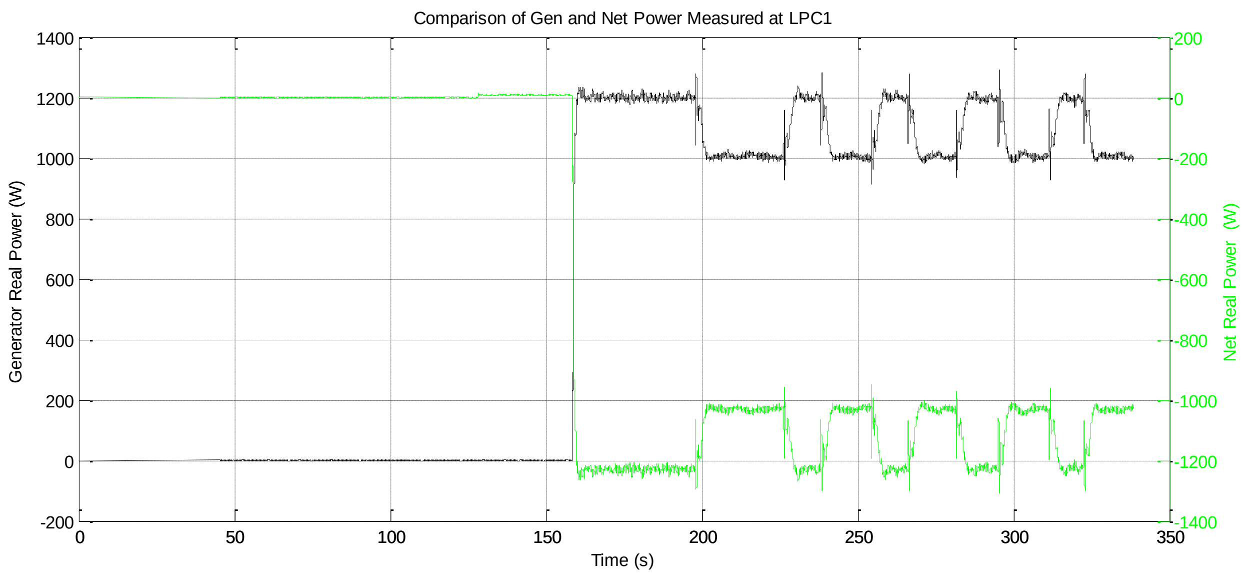


Figure 24 - LPC1 Generator Active and Reactive Power

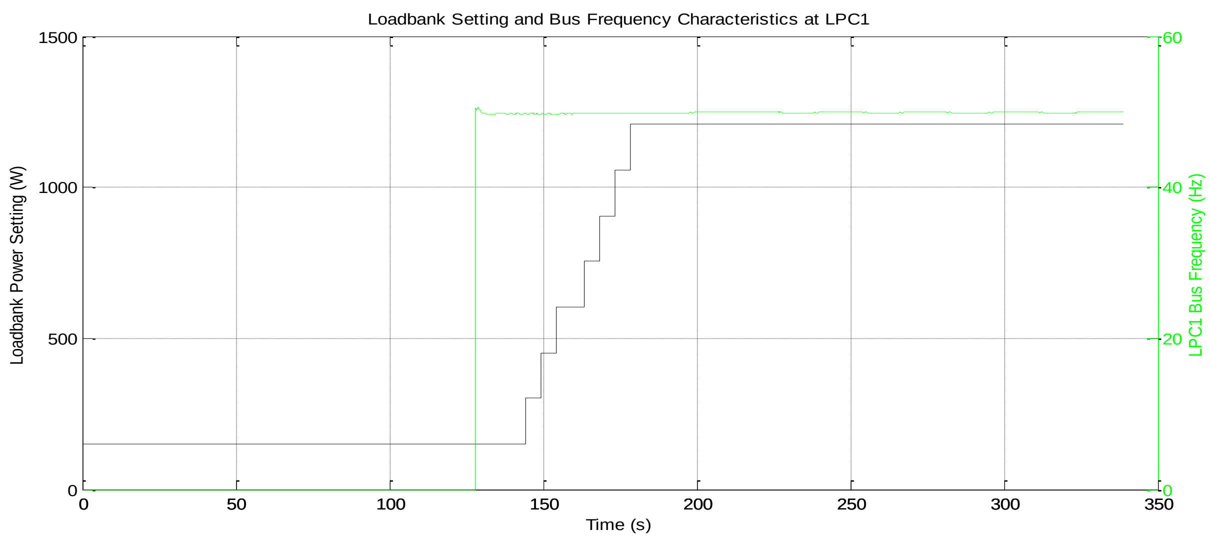


Figure 25 - LPC1 Bus Frequency and Loadbank Setting

Figure 24 again displays the generator response to alternating load conditions throughout the system, after 200s. An interesting phenomenon at the local LPC1 busbar is the increase in set loadbank power rating, shown in Figure 25**,** and this effect, or lack of, on busbar frequency. Power generation during this time is significantly greater than demand ratings for the system, thus this loadbank has minimal effect. The net real and reactive power from LPC1 is negated, indicating that this power is injected from LPC1 and absorbed elsewhere in system, specifically at LPC2.

4.1.2.3 DG2/LPC2 (Orange) Power Island

As with previous generating units, the 10 kVA inverter, connected to a DC source, is not initially connected to the microgrid. Switch 'C' at LPC2 is turned 'on' after 132.5s, and it is after this that the inverter begins to generate power. As with any generating component, the initial connection results in significant frequency variations at the local connection point (in this instance the LPC2 bus) as well as at the terminals of the inverter. Figure 26 illustrates LPC2 busbar frequency. It can be seen that there is no initial 50 Hz reference frequency, unlike Figure 21.

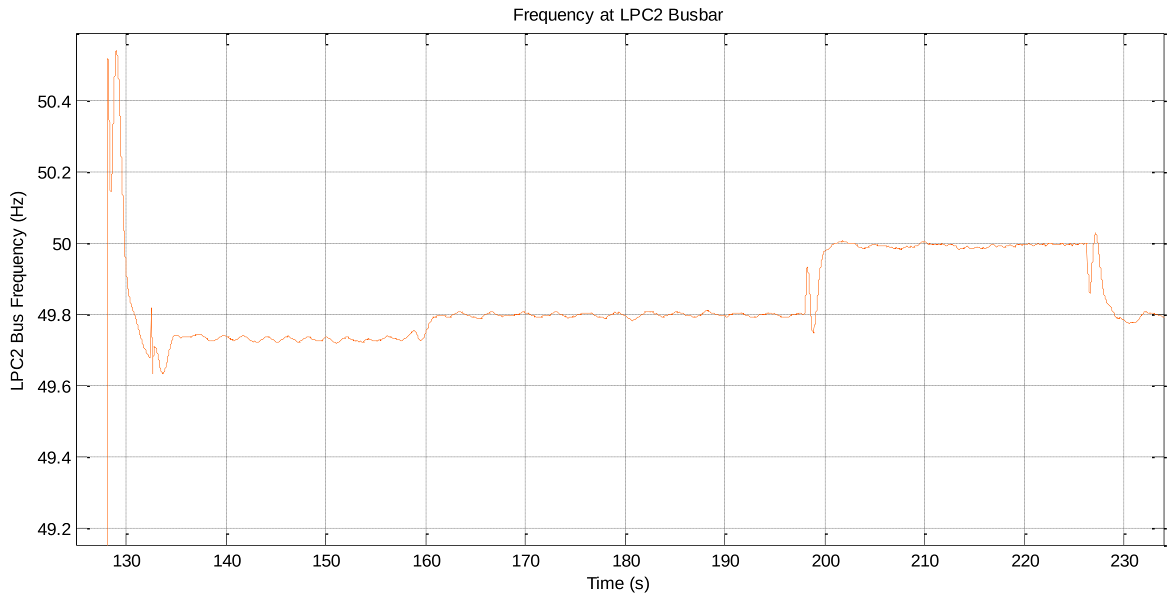


Figure 26 - LPC2 Bus Frequency

Figure 27 demonstrates the inverter start-up behaviour, in terms of active power output, due to this initial connection. The inverter behaves as expected and decreases in generation as the final synchronous generator at LPC1 is connected (159.5s).

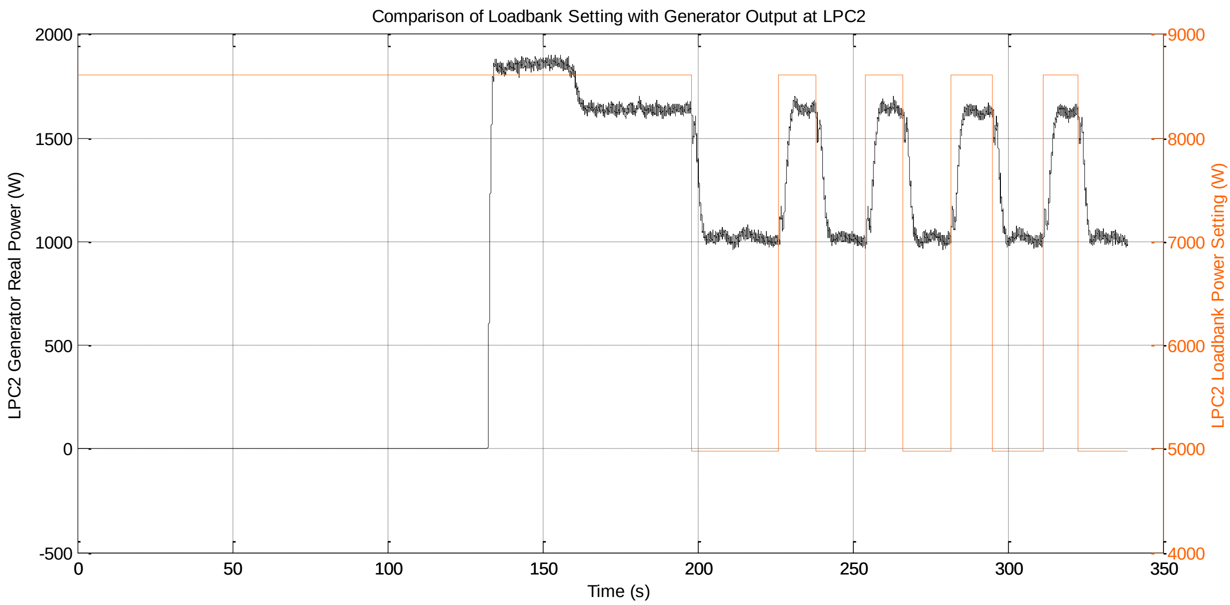


Figure 27 - Inverter at LPC2 Power Output

The resistive-inductive (or resistive-reactive) loadbank at this power island is a significant controllable factor for the analysis of power flow. The loadbank power settings, from maximum (8607 W) to minimum (4983 W) correspond to the change in output power for this bus, the inverter and for the system in its entirety. The resistive and reactive loadbank settings are illustrated in Figure 28.

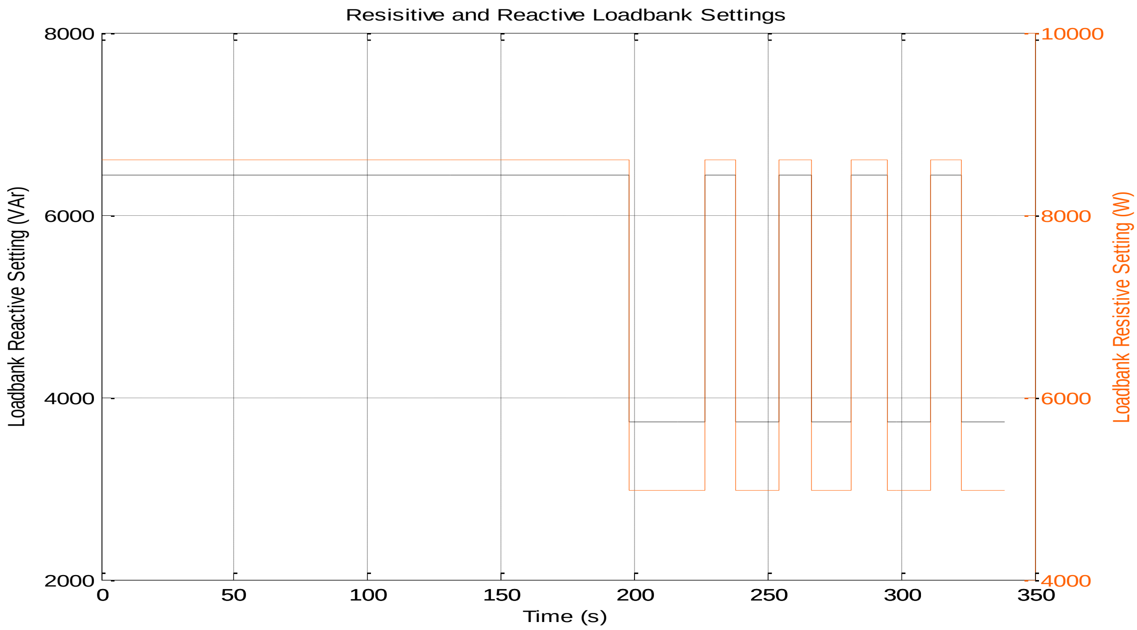


Figure 28 - Loadbank at LPC2 Settings

Figure 28 compares the loadbank resistive rating with the actual inverter output. As expected, an increase in load demand must be answered with an increase in generation for stability.

The inverter frequency is far more greatly influenced by these abrupt changes in loadbank rating. When demand exceeds generation, as is the case when loadbank is at maximum value, the frequency experiences a decline to below nominal value. This is indicated in Figure 29.

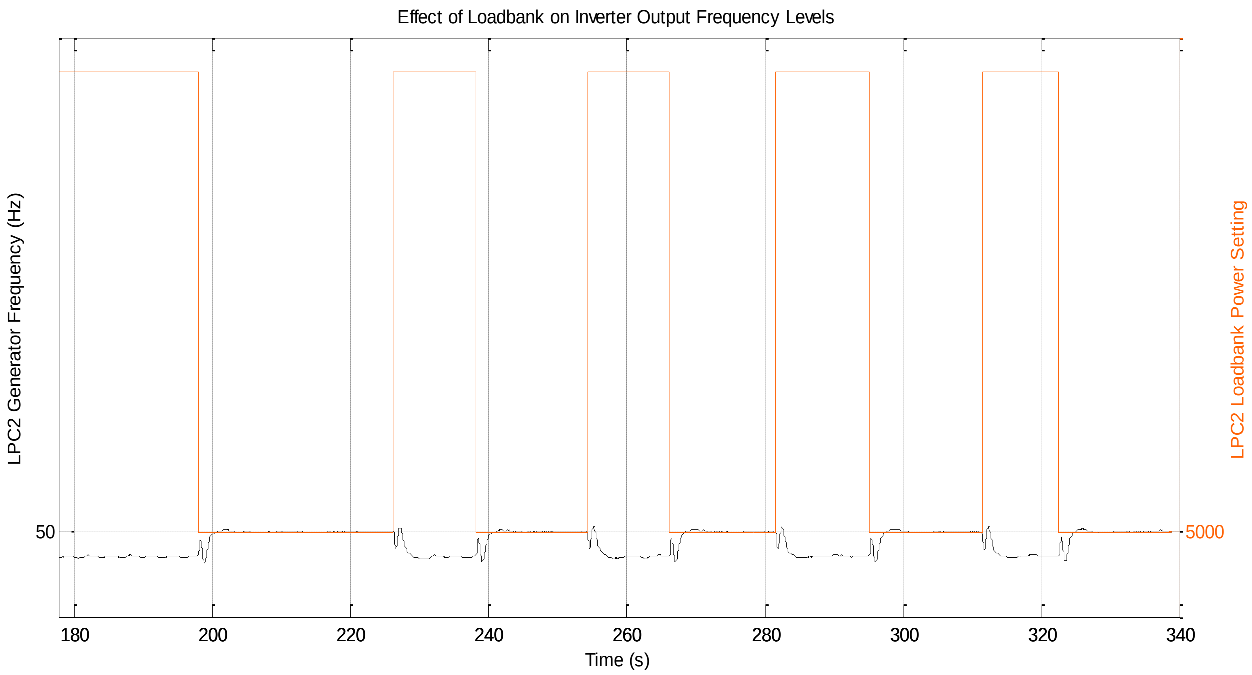


Figure 29 - Comparison of Loadbank Setting and Generator Frequency

The real and reactive components of net power at LPC2 are illustrated in Figure 30 below. As discussed throughout, the trend of each component is determined by the switching states of loadbank values and the connection of distributed generation.

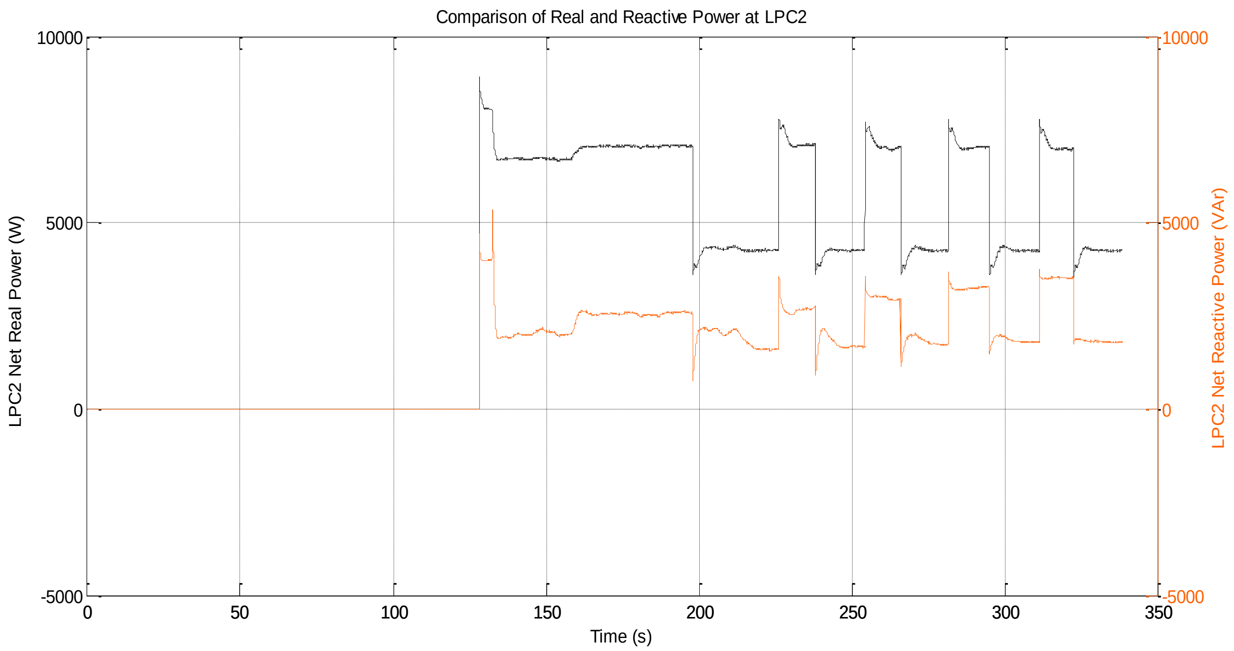


Figure 30 - LPC2 Real and Reactive Power

Reactive power is present within the system due to the presence of inductive loadbanks. Each loadbank is resistive-inductive with the latter being around 75% of the corresponding resistive load element. This arrangement allows the accurate simulation of real-life demand.

4.1.3 Reflection of MATLAB Analysis

In total, six separate scenarios were analysed to aid in the understanding of system performance and short documents for each were produced. The first scenario was selected because of its simplicity and focus on frequency analysis. The second, being more difficult, highlights microgrid performance in relation to power flows, both real and reactive, and frequency behaviour due to loadbank alterations and the connection of distributed generation when the microgrid is in operation. Understanding these past scenarios aids in the overall understanding of the microgrid. Upon completion of initial visual designs, the aforementioned past scenarios will be repeated using the WinCC design. This will support the continued improvements for GUI design based upon the effectiveness of displayed results. After upgrading the system to incorporate real-time data, this process will be repeated with newly attained results, which will be archived. This will require manually setting the microgrid's physical layout to match that of previous simulations and repeating scenarios. This initial MATLAB analysis was crucial and it will significantly benefit the future progression of the project in relation to outputting SCADA results.

4.2 Data Acquisition in WinCC

In order to correctly display the historical data in WinCC, tags must be created for each measured variable. These tags must be updated using the data stored in the CSV files. This is accomplished through the use of a C-Script that will read the CSV data in and assign each read value to the appropriate tag. Before this script is implemented in WinCC however, the software must firstly be designed and tested in Microsoft Visual Studio.

4.2.1 Visual Studio Design

The visual studio C-Script design reads in the CSV file through the use of the 'Test_Function ()' that is defined to read in a CSV file and print out a single row. Each row in the CSV file contains all 280 variables, therefore the CSV file will be read in and printed line by line to update the tag values as needed. It can be seen from Appendix B.4 that this function is held within a while loop which runs the code for each row of historical data (60677 in the case of sc1_5pc_d_1.csv), storing the variable in an array of length 280. This array is then updated after each iteration of the loop. The C-Script for the 'while' loop is shown in Figure 31 where it can be seen that the test function is called for each row of the CSV file. The 'empty for loop' will provide a delay if required, although it is more likely that the code will be too slow rather than too fast.

```
    int main() {

        long int i = 0;
//used to define which row of the csv file is printed

       // long int d = 0; //Use for delay coding
    while (i<60677){
//while i is less than the number of rows in the csv file compte Test_Fucntion
        Test_Function(i);

    // for(d=0; d<60677-i; d++);
        //Will delay the code by a set time, not needed at present

        i++; //increments the row count
    }
    return 0; //returns int 0 as required by 'non-void' function
    }
```

Figure 31 - CSV Read main ()

The main function, Figure 31, can be found to read in the CSV data correctly and print the tag values to the console, in a row-by-row fashion. The Test_Function () called by the while loop can be found in Appendix B.5. When testing the execution time of the code, it can be found that the software takes around 5 seconds to print a single row. Therefore the delay code may not be required, although this cannot be confirmed until the system is implemented and tested in WinCC. Figure 32 shows the command line console that prints the read data as required, with two rows being printed in the example shown.

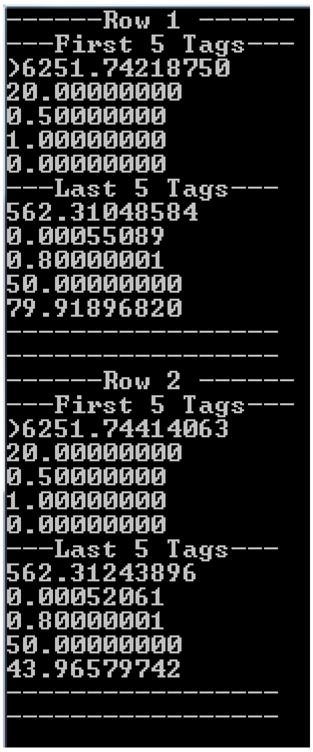


Figure 32 - CSV Read Output

Figure 32 depicts the first five and last five samples of the CSV file that have been printed, with the first and second rows shown, and so it can be concluded that the code successfully reads in the CSV data in a row-by-row basis by updating a single array. This code can then be implemented into the WinCC design in order to update the defined tags for the control room system.

4.2.2 WinCC Design

Once the C-Script has been successfully run using Visual Studio, it must be incorporated into the WinCC design in order to update the tags as required. For the purpose of historical data simulations, the code will read the CSV data row-by-row to mimic the reading of real time data. The script written in the previous section is split into two sections for WinCC implementation. The 'Test_Function' is written as a standalone .fct file that will be called upon by the C-Action associated with a defined 'Start Button' in the Graphic Designer. The button is defined to run the 'while loop' script shown in Figure 31.

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName)
{ //above code is produce by WinCC and cannot be editted

#define ROWS 60677 // Set the number rows of data to be read in during the simulation

long int i = 0;  //defines long int i, long int required to handle values as large as ROWS

while(i<ROWS){  //reads in data for each row of the CSV file

    Test_Function(i);   //Executes Test_Function to update tags

     printf ("If No Errors Exist The Tags Have Been Successfully Updated\n");
     //Prints statement to GSC Diagnostics
    //if an error has occuredi  it will be printed in GSC Diagnositcs

     i = i++; //increments i as required
}

}
```

Figure 33 - 'Start' Function WinCC

Comparing Figure 33 with the Visual Studio version (in Figure 31) it can be seen that the two scripts are very similar, although it should be noted that the '#include' files, situated at the beginning of the Visual Studio code, are not required in WinCC. The 'Test_Function.fct' script can be found in Appendix B.5 where the 'SetTagFloat ()' function is used to update the tag values as required. It is possible to confirm that the historical data is being correctly read into the system by defining a number output fields set to display several tag values. It should be noted that, in order to correctly display the tag value, the 'Output Format' must be set to the correct number format. For example, to display the 'Elapsed Time' value of 6251.742, the format must be set to 9999.999, as shown in Figure 34.

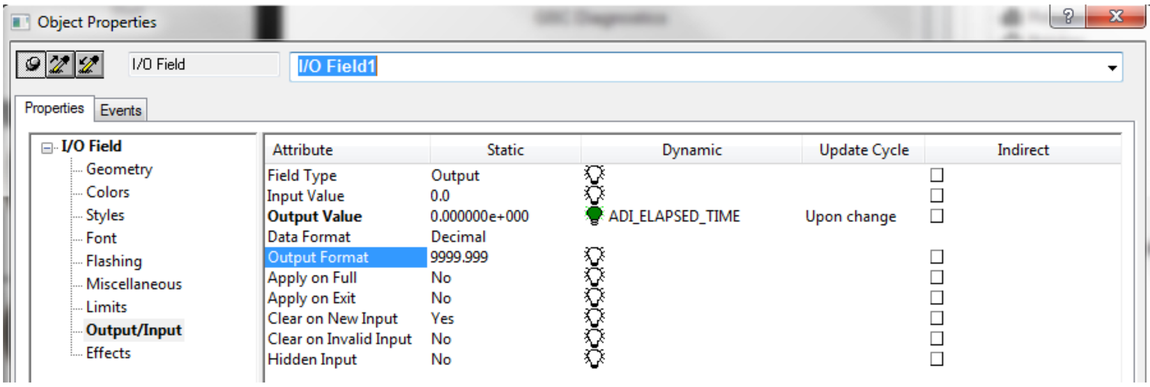


Figure 34 - I/O Field Number Formatting

With five I/O Fields defined appropriately, the C-Script can be tested using WinCC RunTime to run the graphic WinCC design. Once the RunTime has begun, the C-Action can be run by clicking on the 'Start' Button. After the code has executed the 'Test_Function' script, the output fields are updated accordingly, as shown in Figure 35.

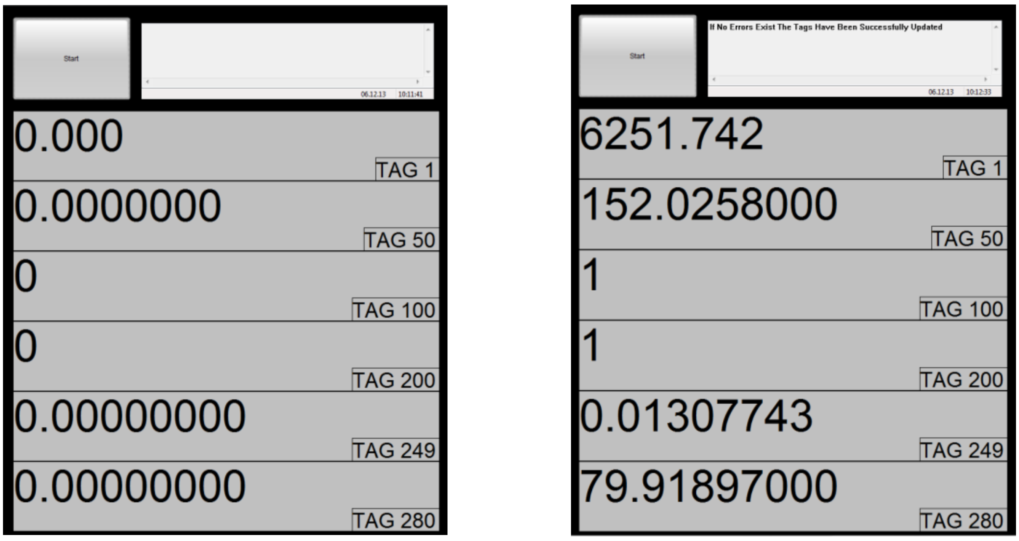


Figure 35 - WinCC RunTime Screen before (left) and after (right) Start Button is Pressed

It can be seen that the tags are correctly updated using the C-Script and historical simulations will be able to be run using the control room design. One drawback from the current design is the fact that the C-Script takes around 14.5 seconds to update the tag values. This is insufficient for a realistic simulation due to the fact that each row update is an increment of 2ms. For this reason, alternative methods for reading in the CSV file were investigated.

One alternative method could be the use of a Direct Link Library (.dll). This involves creating a .dll file in Visual Studio that contains the ‘CSV Read’ code, which can then be opened by the WinCC C-Script. The proposed idea would input the tag[] array to the .dll file and fill it with the specific CSV row. This array would then be output back to the WinCC C-Script and used to update the tags. However, when implementing this method, it can be found that RunTime takes around 25 seconds to update the tags, 10.5 seconds slower than the ‘Test_Function.fct’ method. It can be concluded that the originally proposed method will be used, unless another solution can be found that will improve the execution time of the software.

In terms of the final real-time data design, the issue of timing should not affect the system. This is due to the fact that the delay in the software is caused by reading in the entire .csv file on each iteration of the ‘while’ loop. When reading the data in real-time, only one row of data will be read in meaning that code will run more efficiently. Reading in the real-time data to the WinCC design will be one of the main targets for the remainder of the project, in order to produce a real-time system as required by the project specification.

5 WinCC GUI Design

As clarity is an important aspect to take into consideration, the initial subject of focus was the background display of the system. It is crucial to note that, in a conventional control room, colours should only be used for information that requires the attention of the user as to alert them to any significant changes within the system; thus it was decided to follow this to an extent. From the aforementioned visits in Section 2.4, it became clear that the appropriate choice of colour for the background was black. This was for the reason that most colours can be easily identified upon a darker background, especially bright colours, as they contrast with the darkness of the black. The next decision to be made was the colour that would be used to identify the individual islands. In view of original work upon the microgrid [40], the three islands - DG1, DG2 and DG3 - are identified as being green, orange, and purple respectively, and the overall microgrid is illustrated in grey. This has been incorporated into the group logo, and will be evident from the welcome screen and the menu screen, both of which can be seen in Figure 36 and Figure 37 respectively:

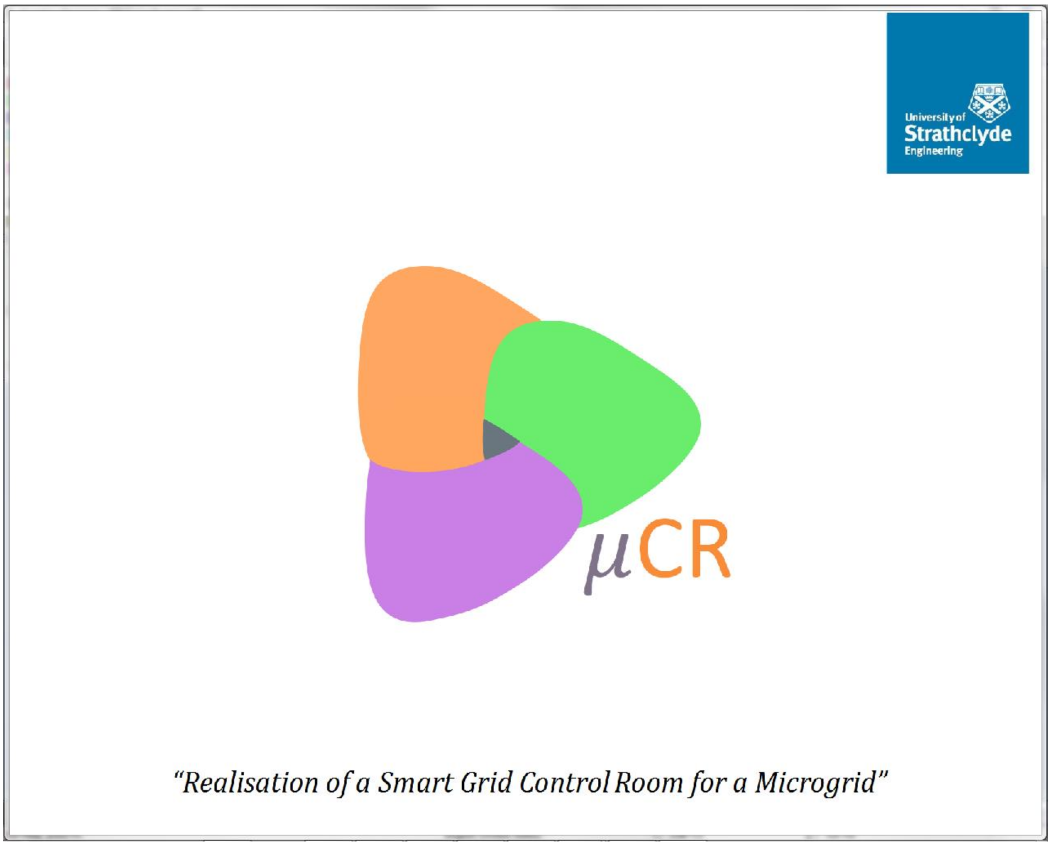


Figure 36 - WinCC GUI Welcome Screen

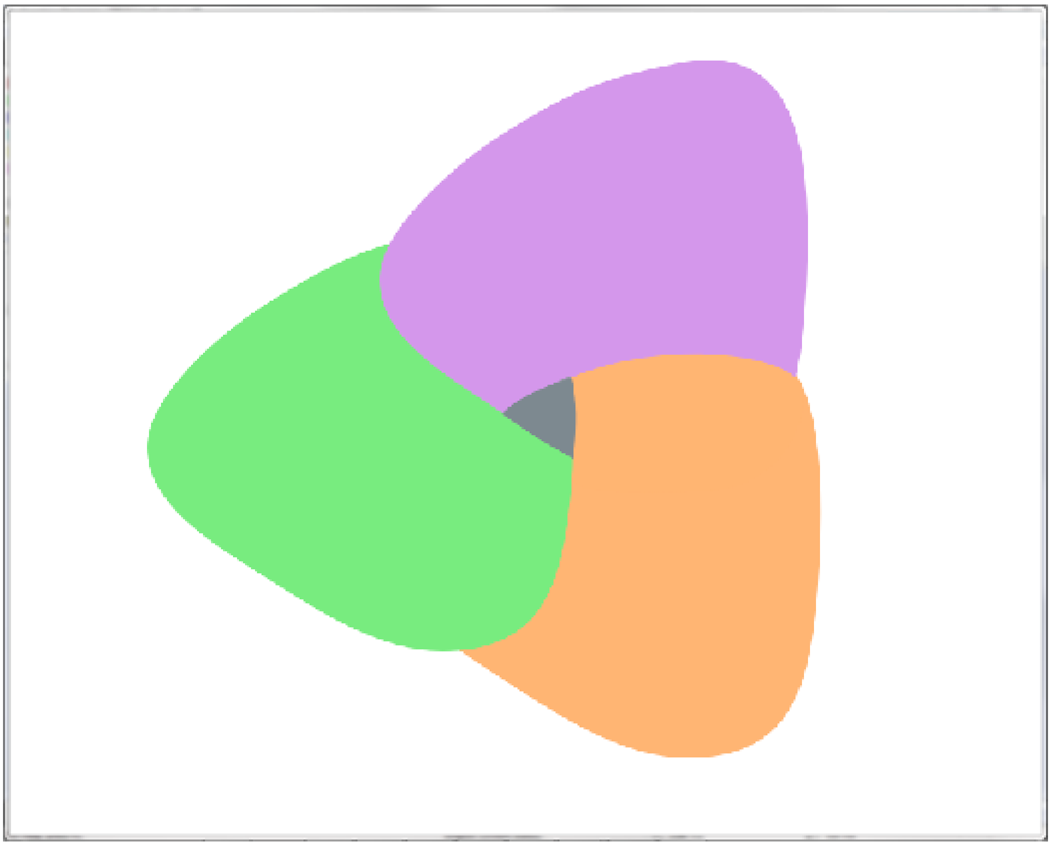


Figure 37 - WinCC GUI Menu Screen

As can be seen from Figure 36 and Figure 37, the three coloured shapes each represent the individual islands, and the grey centre represents the overall microgrid schematic, which comprises all islands interconnected.

Figure 36 will welcome the user to the system and will lead to the menu screen upon clicking the left mouse button. Figure 37 will be an interactive interface, where the user will have the choice to pick one of the three islands, or the overall schematic, by left clicking the desired colour. This will then lead to its respective circuitry.

To indicate that the connections are live, the circuitry for each island will be coloured cyan when active, as cyan is bright enough for the user to view upon a change; and when an island is disconnected, i.e. not active, the circuitry will depict the de-energised lines as grey. Figure 38 shows the overall schematic when the microgrid is de-energised.

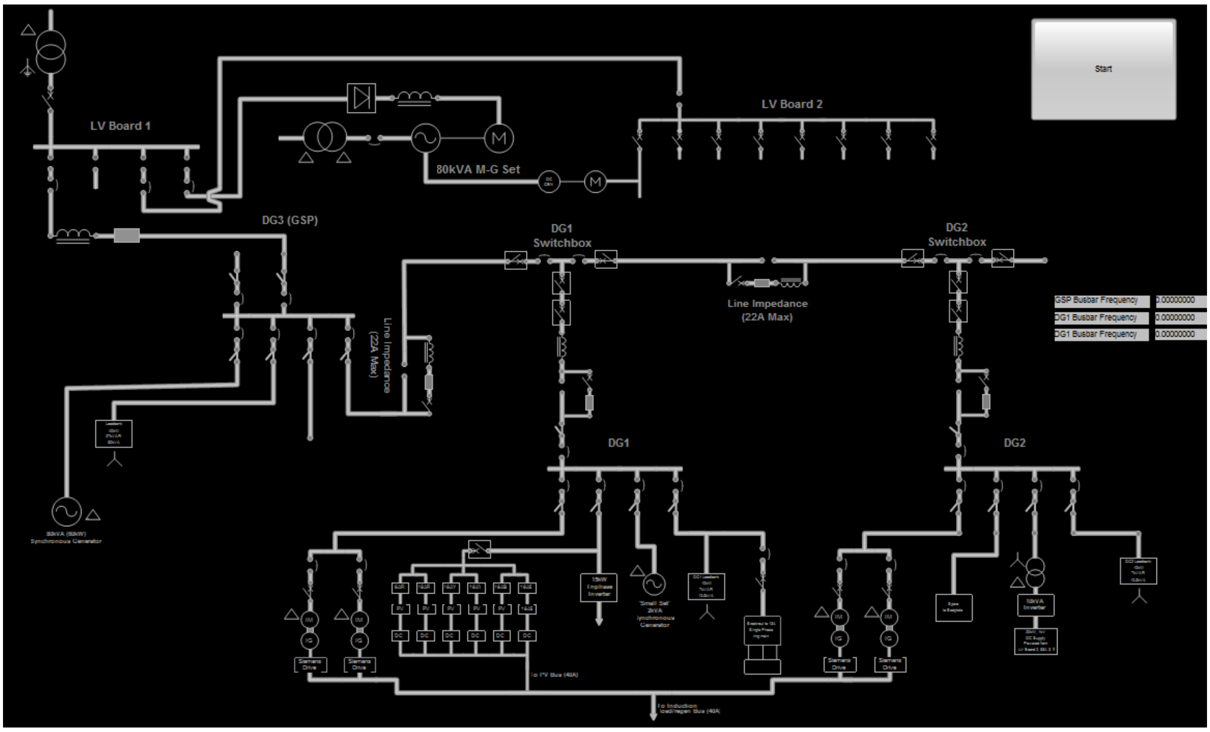


Figure 38 - WinCC Microgrid Schematic Diagram

The specific components of the microgrid schematic have been discussed within Section 2.2

The reason for grey being the chosen colour when a line is de-energised is due to the fact that de-energised lines are very important to the user, as this may identify a fault within the system, and grey is a clear contrast to black. A method that was discussed was using dashed lines for de-energised sections, although this was found to be less recognisable in comparison to a colour change.

On the right hand side of Figure 38, it can be seen that each island's individual frequency is shown. This shall be updated automatically when the project progresses to working with real-time and historical data. The large start button on the top right hand corner allows the simulation to begin reading in data.

Figure 39 is the schematic for island DG3 indicating where parts of the circuit are live, and the lines are energised.

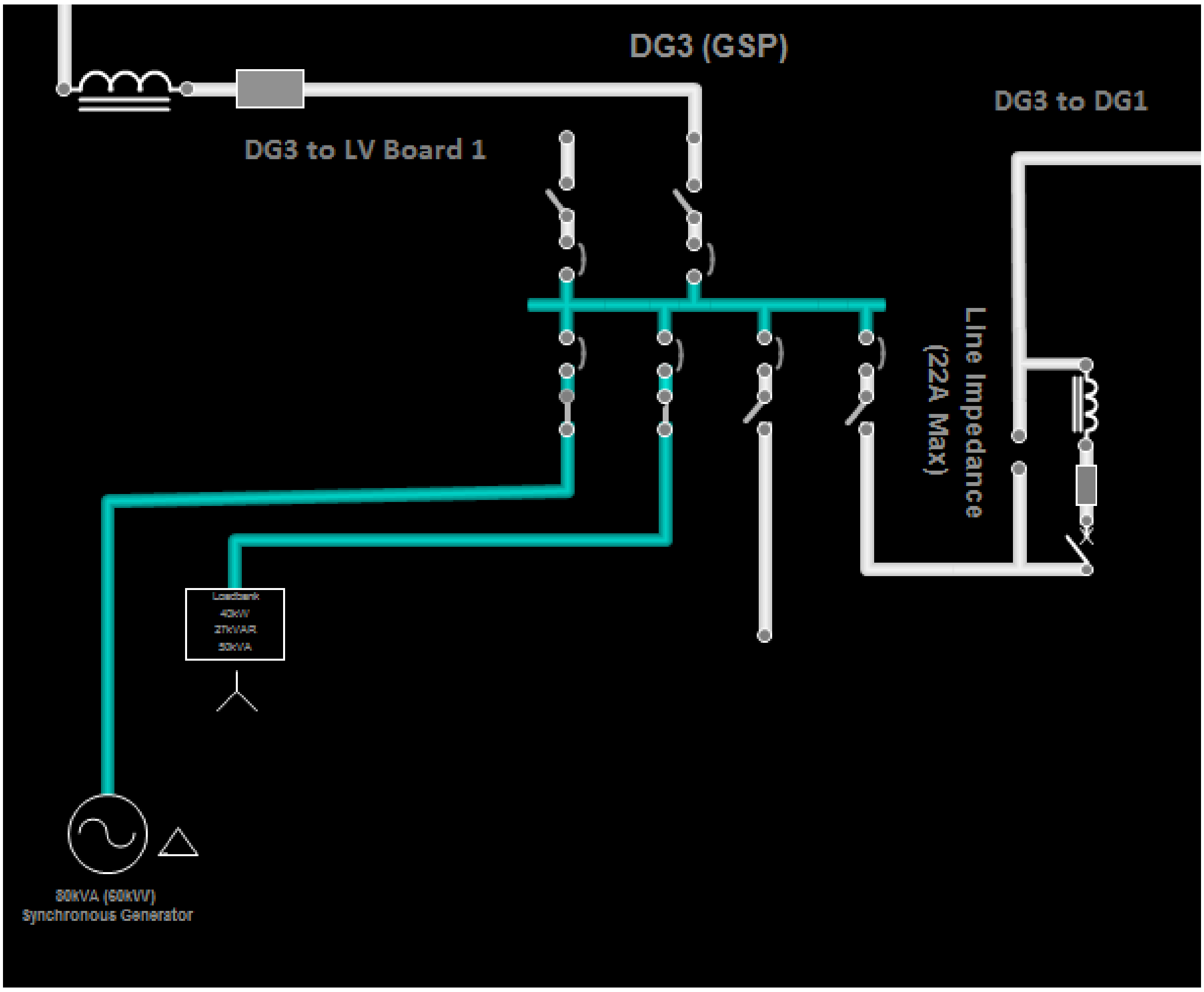


Figure 39 - WinCC Depiction of Energised (cyan) and De-Energised (white) Lines

As transmission line status has been discussed previously in Section 2.4, the status of breakers and switches is also of importance, as is the magnitude and direction of power flow. The direction of power flow can be indicated with arrows, and the magnitude of percentage loading can be directly related to the size of the arrow. The only issue with the dynamic arrows is that it may prove to be too distracting to the user as the screen would possibly become overly cluttered. An alternative method to displaying power flows could be by pie charts rather than arrows, although this may also result in visual clutter on the display. A similar idea proposed by Mikkelsen et al. [30] was to use 2D bars as opposed to pie charts.

Another important piece of information that should be displayed is the present generated power of the microgrid. This information can be displayed at the side of the schematic overview in the form of a graph or chart, or even a quantitative value for the user to view easily.

The status of breakers and switches are of importance to the operator, therefore when a breaker is closed, the icon representing the breaker on the schematic will be filled with the same colour as the live circuit. When a breaker is open, there will be no fill. If a fault arrives, an alarm will be appear in red and produce an audible noise.

In a control room, it is common that there is a “video wall”, also referred to as a “power wall” [30]. In CR2, there is an LCD television on the wall which will act as the “video wall”, connected via an HDMI cable to the main PC, on which the project is being built. This TV will display the WinCC system. Figure 40 shows a picture of the control room, where the video wall can be clearly seen in the middle of the photo. The two black PC monitors, which are situated on the right-hand side, are used to display information from other software. The glass window, in front of these two monitors, looks directly onto the microgrid.

Figure 40 - Control Room Video Wall

The complete system will comprise a welcome screen followed by hierarchical levels of the microgrid. The hierarchy can be broken down to four separate levels. The initial top-level will be an overview of the microgrid, as a single-line diagram, displaying basic information including the power flows. Upon the user clicking on a specific island within the grid, the second level will display a larger, more detailed version of that island, along with its relevant quantitative and qualitative information. The third level will be accessed by the user clicking on certain areas within that island, which again will entail a more detailed level of relevant information with respect to the components chosen, which may include graphs or charts. This was discussed in Section 2.4.

Alarms will be added to the system in the second semester. There are various methods that could be chosen to depict an alarm. It is necessary to make separate alarms distinguishable, especially alarms that indicate a potential safety hazard. The following methods could be deemed effective: flashing lights of certain colour for individual alarm type; auditory alarms with different sound corresponding to each alarm type; a pop-up display that alerts user of a proble;, or a combination of any of these alert systems.

6 Future Work

The main focus of the project work thus far has been to develop and refine many ideas relating to the visualisation of the data collected at the microgrid. It could be said that a solid base of the project is now firmly in place, with a functioning GUI being developed for use with all historical data that is currently available. Teething problems involving the various types of software detailed in this report have been overcome. The GUI is currently in a standard control room format, comprising a schematic diagram of the microgrid with a black background and lines that are either cyan if energised or grey if de-energised. This format is very beneficial to a control room operator, particularly if they were tasked to use the SCADA system for an extended period of time.

The current design operates for the historical data available, but the main objective henceforth is to adapt the design to work seamlessly with the communications architecture already in place at the microgrid, such that real-time data may be analysed. The capability to compare real-time data with historical data will also be incorporated into the system.

The main platform on which to exhibit what has been achieved during the undertaking of this project to a wide and varied audience will come on the day of the Tradeshow. Consequently, a lot of the project work during the second semester will be geared towards this event. As this is a software-based project, the best way to showcase the SCADA system when the microgrid is running will be either to show faculty members CR2 or remotely access the control room computers via a laptop in the Tradeshow room.

Shown in Figure 41 is the current Gantt Chart to track the project's progress, as well as lining out the work that will be undertaken during the second semester. As can be seen, a number of deliverables have already been achieved, with the next few months consisting of the completion of the basic control room system before the addition of innovative and intelligent visualisation techniques. The incorporation of real-time data from the microgrid has also been included and it can be seen that the project is still on schedule to be completed on time and ready to present at the Tradeshow.

As can be expected, several changes have been made to the project schedule throughout the initial stages. The original Gantt Chart can be found in Appendix D.

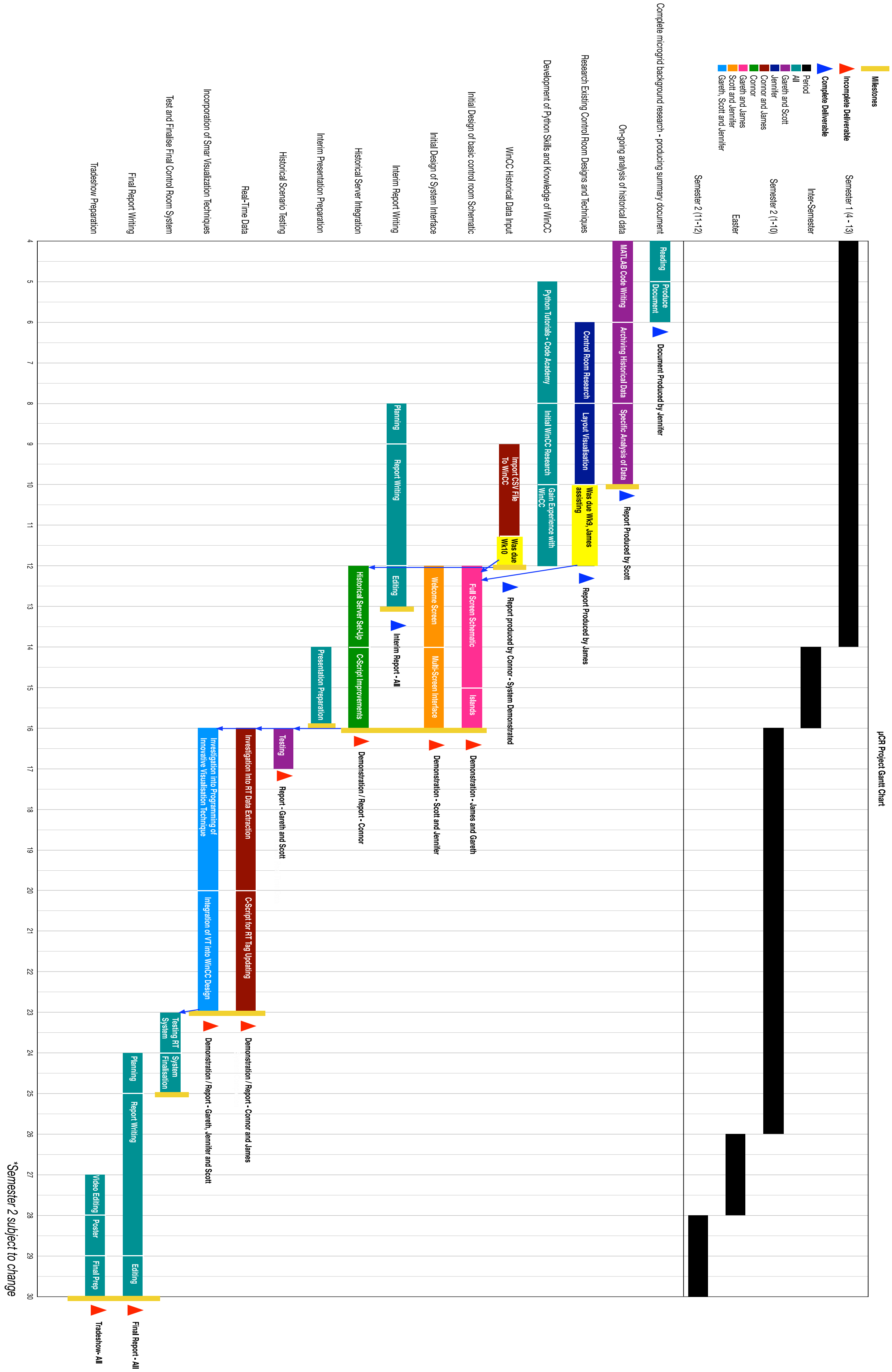


Figure 41 - Current Gantt Chart

7 Conclusions

7.1 Reflection on Technical Risks

During the initial proposal of this project, some potential risks were identified. The initial group proposal document can be viewed in Appendix D. A brief reflection of these risks will be discussed and their impedance, if any, on project progress is expanded upon.

An initial risk discussed was technical incompetency, referring to potential problems faced with new software. It was suggested to ask members of staff who specialise in specific software to aid in the steep learning curves faced. During the investigation into Siemens WinCC software, it was apparent that there would be an extended learning period as this software was previously unknown to all group members. All changes to project progress were updated in the project Gantt chart. There were no members of staff present at the university who had expert knowledge in WinCC, thus additional members of the project group began learning the software early on. Another technical risk associated with WinCC was the initial installation on a PC in CR2. This stage took longer than anticipated due to problems with obtaining the licence and installing. Help from research staff and Siemens technical advisors aided in the successful installation of the program.

There always exists the risk of not meeting set deadlines, and producing deliverables for each objective, and so this was detailed on the initial risk register. This occurred for some deadlines, such as for preliminary documentation demonstrating the understanding of essential background research and initial setting up of WinCC software. This, again, was updated in the Gantt chart with all semester 1 objectives being completed.

A risk involving the lack of access to the control room and microgrid laboratory was realised early into the project plan, however, this was mitigated through communication with the facility's research staff to find times when access is unavailable. Planning ahead of schedule will ensure that this does not significantly impede progression.

Non-technical risks include illness to project members and loss of historical grid data or project documentation. Non-technical risks can delay project progress as a result of having to repeat previously completed objectives or by the straining of resources as an adaptation to cover for ill group members. All data is backed up extensively using different online and offline storage technologies, including USB devices, online sharing folders and back up files on personal laptops. No project members have succumbed to illness and so, thus far, no mitigating actions for this purpose have taken place.

7.2 Final Conclusions

The initial investigations and research carried out by way of literature review and familiarisation with software tools have highlighted key aspects of both microgrid operation and control room design. Examples of these are important measurands such as frequency and power flows in the microgrid and use of colours and advanced visual aids in the control room. The project aims to achieve innovative solutions to data visualisation within a control room environment.

As stated in the aims and objectives section of this report, the main challenge of the project is to translate the extracted microgrid data from its raw form into intelligible and valuable information such that it can be displayed in an aesthetically pleasing format, both for persons with previous exposure to electrical power systems and for those who lack prior knowledge in this field.

The first phase of the project was to understand the microgrid concept and its applications to real-world power facilities. This has been achieved through reviewing existing literature (i.e. journal articles, the DNAPL manual and reports regarding the Santa Rita Jail facility and Fort Bliss Army Installation) as documented in Section 1. Analysing historical data from the R1.34 microgrid facility, as discussed in Section 4, was achieved using a MATLAB GUI. From this, the behaviour of the research facility under varying simulation scenarios was understood. This has provided a sound basis from which to begin designing an optimal platform for the presentation of microgrid data, as well as realising which data fields are most important to include following customer requirements.

The Siemens WinCC design began after historical data, in the form of .csv files, was imported into the software. From here, an initial design concept was established for the control room GUI. This initial schematic design incorporates on-field research of current power system procedures which can be advanced visualisation techniques. There is also the possibility of utilising additional software packages to achieve more advanced data visualisation techniques. These advanced techniques will provide an innovative method for presenting valuable information in an unconventional manner, in comparison to existing functional control rooms.

Overall, the project has progressed sufficiently thus far such that the final system will be completed in time for demonstration at the Tradeshow.

8 Works Cited

[1] A. G. Anastasiasdis, J. Vasiljevska, and A. G. Tsikalakis N. D. Hatziargyriou, "Quantification of Economic, Environmental and Operational Benefits of Microgrid," in *Power Tech Conference, 2009 IEEE*.

[2] S. Chowdhury, P. A. Crossley, and C. F. Ten S. P. Chowdhury, "UK Scenario of Islanded Operation of Active Distribution Networks - A Survey," in *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*.

[3] S. P. Chowdhury, and P. A. Crossley S. Chowdhury, "Protection Issues for Microgrids," *IET Microgrids and Active Distribution Networks*, pp. 77 - 95, 2009.

[4] P Lewis B. Cory, "The Reorganisation of the Electricity Supply Industry - A Critical Review," *Power Engineering Journal*, vol. 11, pp. 42 - 46, 1997.

[5] W. E Feero, D. C Dawson, and J Stevens, "Protection Issues of the Microgrid Concept," *Transmission Reliability Program, Office of Power Technologies*, pp. 1-24, 2002.

[6] Alameda County Government. (2012) First-of-its-Kind Smart Grid at Santa Rita Jail Completed by Alameda County and Chevron Energy Solutions. [Online]. http://www.acgov.org/pdf/2012-03-22SmartGridpr.pdf

[7] Alameda County Government. (2012) Santa Rita Jail Smart Grid - Project Overview. [Online]. http://www.acgov.org/pdf/SRJSmartGridOverview.pdf

[8] R. A. Ducey M. D. Johnson, "Overview of U.S. Army Microgrid Efforts at Fixed Installations," in *Power and Energy Society General Meeting, 2011 IEEE*, 2011.

[9] Jeff St. John. (2013, May) The Energy Collective. [Online]. http://theenergycollective.com/jeffstjohn/226081/military-microgrid-smart-grid-asset

[10] Lei Wu, Weiwei Song, Zhihui Jiang Yuo Yuan, "Collaborative Control of Microgrid for Emergency Response and Disaster Relief," in *International Conference on Sustainable Power Generation and Supply, 2009, SUPERGEN '09*.

[11] Amos Brocco, "Ad-hoc Self-Organized Microgrid for Rural Electrification and Post-Disaster Response," in *IEEE Green Technologies Conference*, 2013.

[12] Bong Lozada. (2013) Inquirer. [Online]. http://newsinfo.inquirer.net/531249/meralco-sends-40-power-generators-to-yolanda-hit-areas

[13] Margo Cole. (2013) New Civil Engineer. [Online]. http://www.nce.co.uk/news/construction-equipment/jcb-sends-generators-and-diggers-to-help-typhoon-clear-up-work/8655714.article

[14] Distributed Energy Resources Research Infrastructure. DERR-RI. [Online]. http://www.der-ri.net

[15] Distributed Energy Resources Research Infrastructure. DER-RI. [Online]. http://www.der-ri.net

[16] The OPC Foundation. [Online]. http://www.opcfoundation.org/Default.aspx/01_about/01_whatis.asp?MID=AboutOPC

[17] P. Crolla, G. Burt A. Roscoe, Strathclyde Power Systems Lab Poster.

[18] Gordon Jackson Andrew Roscoe, "Microgrid Laboratory Manual," EEE, Institute for Energy and Environment, University of Strathclyde,.

[19] Andrew Roscoe and Gordon Jackson, "Microgrid Laboratory Manual," EEE, Institute for Energy and Environment, University of Strathclyde,.

[20] Omega Communications Ltd. (2011) SCADA Remote Monitoring. [Online]. http://omegacom.ca/wireless/scada-remote-monitoring/838/

[21] DPStele. Where is SCADA used? [Online]. http://www.dpstele.com/white-papers/scada/page4_1.php

[22] DPStele. An Advanced SCADA Monitoring System Will Effectively Monitor Your Network. [Online]. http://www.dpstele.com/dpsnews/techinfo/scada/scada_monitoring.php

[23] Engineers Garage. (2012) SCADA Systems. [Online]. http://www.engineersgarage.com/articles/scada-systems?page=2

[24] J Y Fiset, *Human-Machine Interface Design for Process Control Applications*.: ISA, 2008.

[25] Muhammad Saas Bin Azhar and Ammad Aslam, "Multiple Coordinated Information Visualization Techniques In Control Room Environment," Ronneby, Sweden, May, 2009.

[26] P M Mahadev and R D Christie, "Envisioning Power System Data: Concepts and a Prototype System State Representation," *IEEE Transactions on Power Systems*, vol. 8, no. 3, pp. 1084-1090, 1993.

[27] J Webber and T Overbye, "Visualization of Power System Data," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, 2000, p. 7.

[28] J Webber and T Overbye, "New Methods for the Visualization of Electric Power System Information," *IEEE Symposium on Information Visualization*, pp. 131-, 2000.

[29] J Webber and T Overbye, "Voltage Contours for Power System Visualization," *IEEE Transactions on Power Systems*, vol. 15, no. 1, pp. 404-409, 2000.

[30] C Mikkelsen, J Johansson, and M Cooper, "Visualization of Power System Data on Situation Overview Displays," *Information Visualisation (IV), 2012 16th International Conference on*, pp. 188-197, 2012.

[31] Statnett. (2013) The power system right now. [Online]. http://www.statnett.no/en/Market-and-operations/

[32] Zhao Kaidi, "Data Visualization," School of Computing, National University of Singapore,.

[33] (2013, Dec) Wind History. [Online]. www.windhistory.com

[34] Google. Google Developer's Application - Chart Gallery. [Online]. https://google-developers.appspot.com/chart/interactive/docs/gallery

[35] D3. D3 Data-Driven Documents. [Online]. http://d3js.org/

[36] Bokeh. Bokeh 0.3. [Online]. http://bokeh.pydata.org/

[37] Mathworks. (2013) MATLAB. [Online]. http://www.mathworks.co.uk/products/matlab/

[38] Siemens. (2013) SCADA System SIMATIC WinCC. [Online]. http://www.automation.siemens.com/mcms/human-machine-interface/en/visualization-software/scada/Pages/Default.aspx

[39] Microsoft. (2013) Visual Studio. [Online]. http://www.visualstudio.com

[40] Andrew Roscoe and Gordon Jackson, "Microgrid Laboratory Report," Glasgow, 2013.

9 Appendix

A. Figures

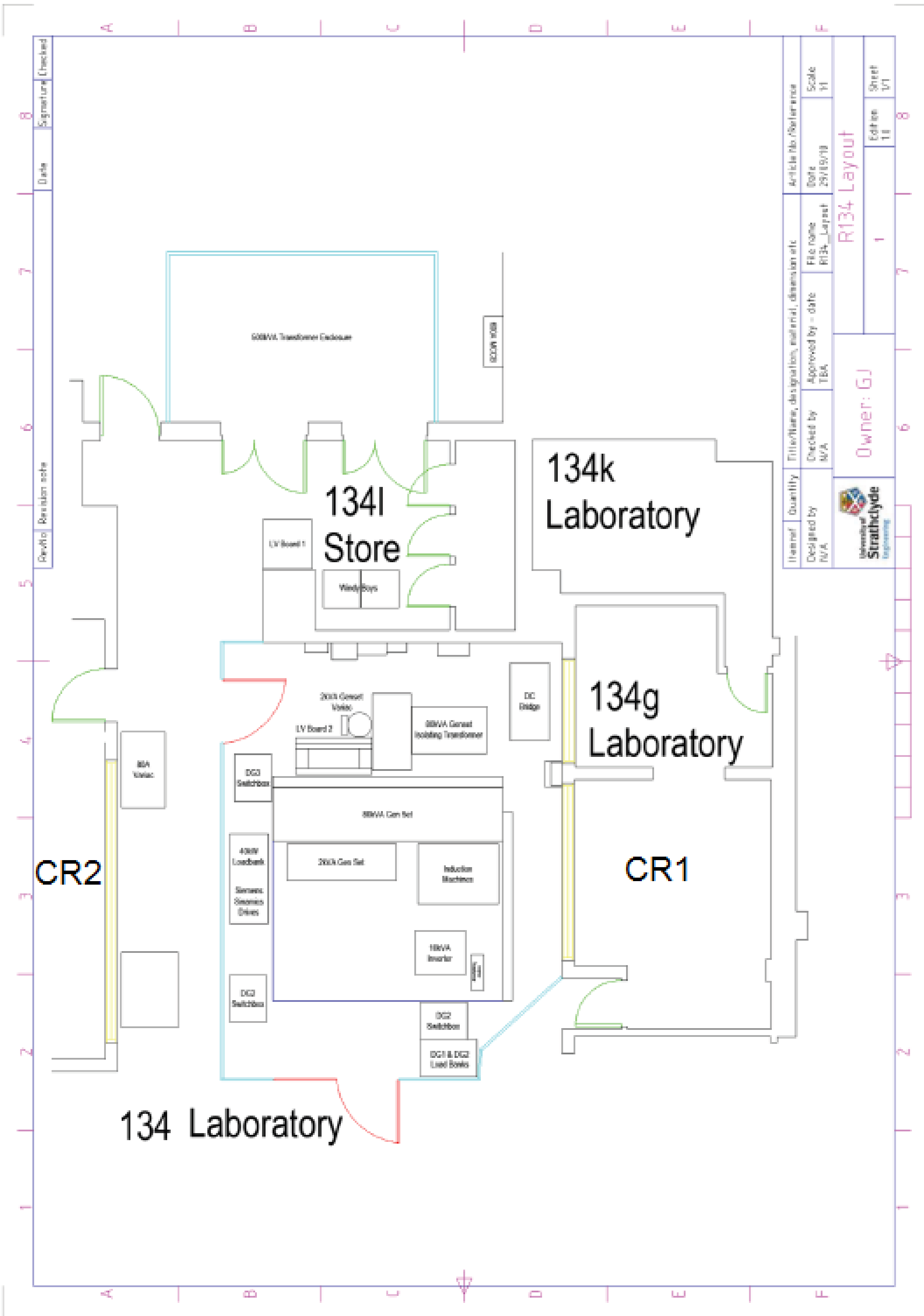


Figure 42 - Map of R134

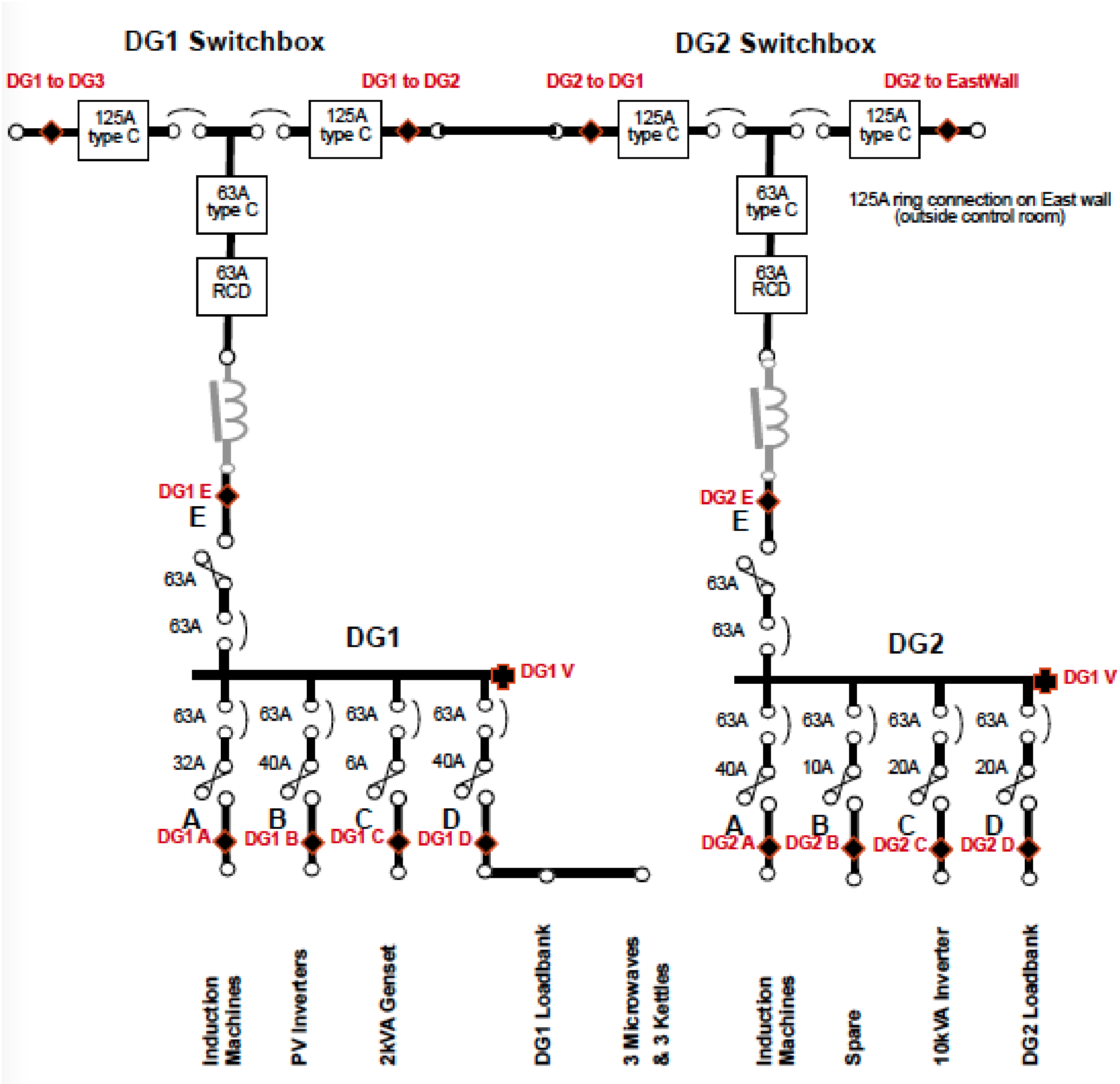


Figure 43 - Schematics of DG1 and DG2 Switchboards [18]

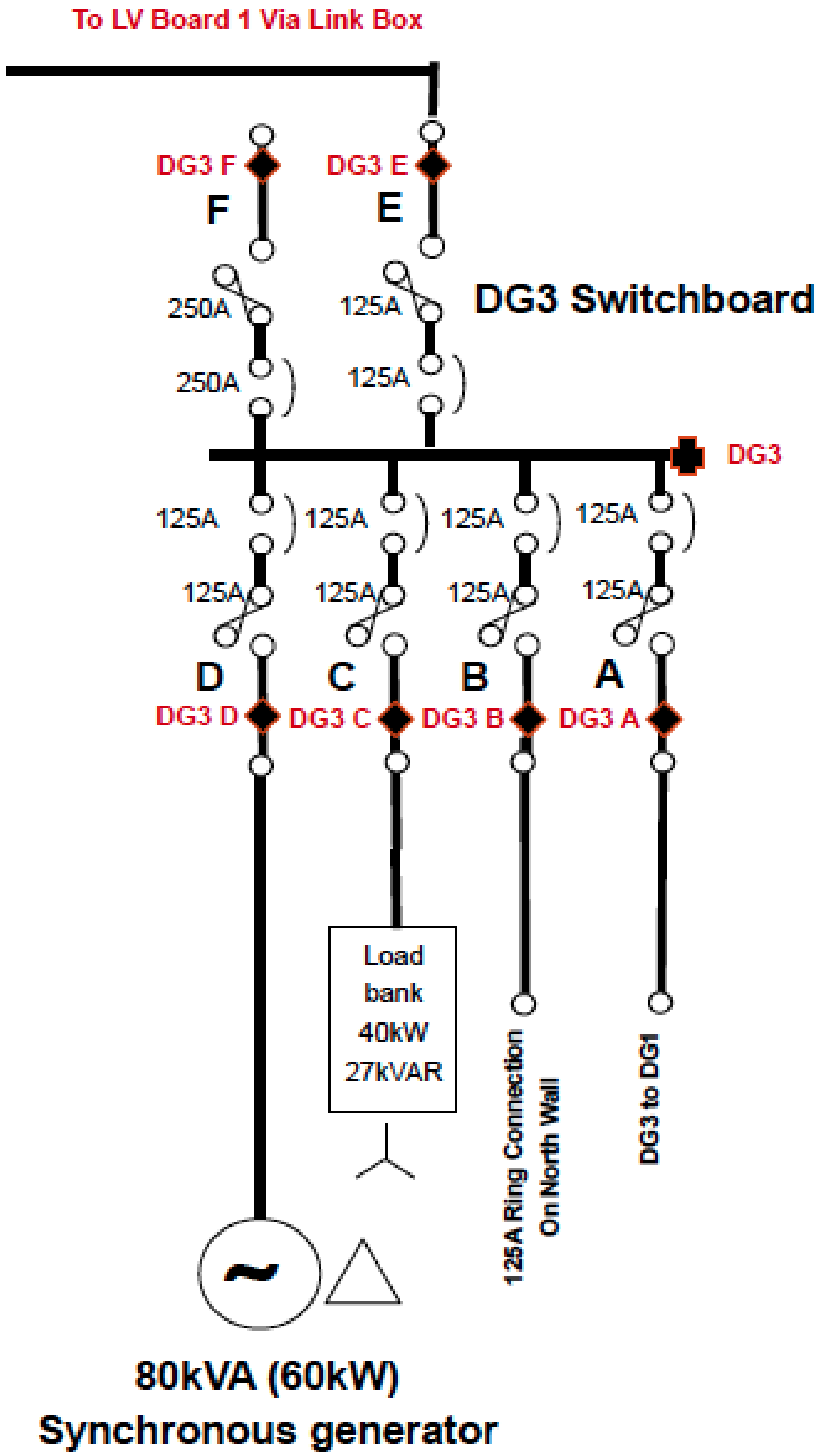


Figure 44 - Schematic of DG3 Switchboard [18]

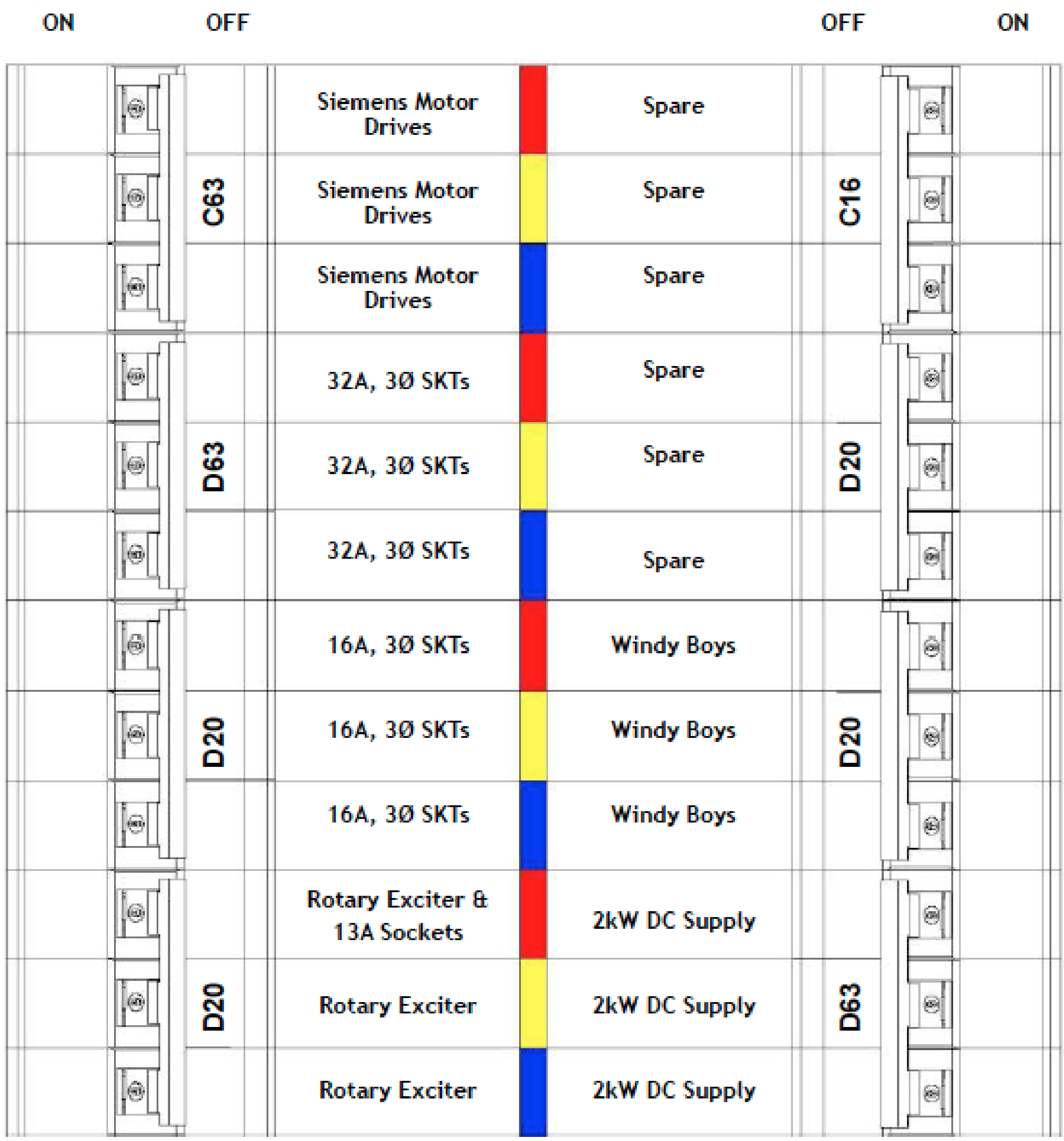


Figure 45 - Usage of MCBs in 200A Distribution Board of LV Board 2 [18]

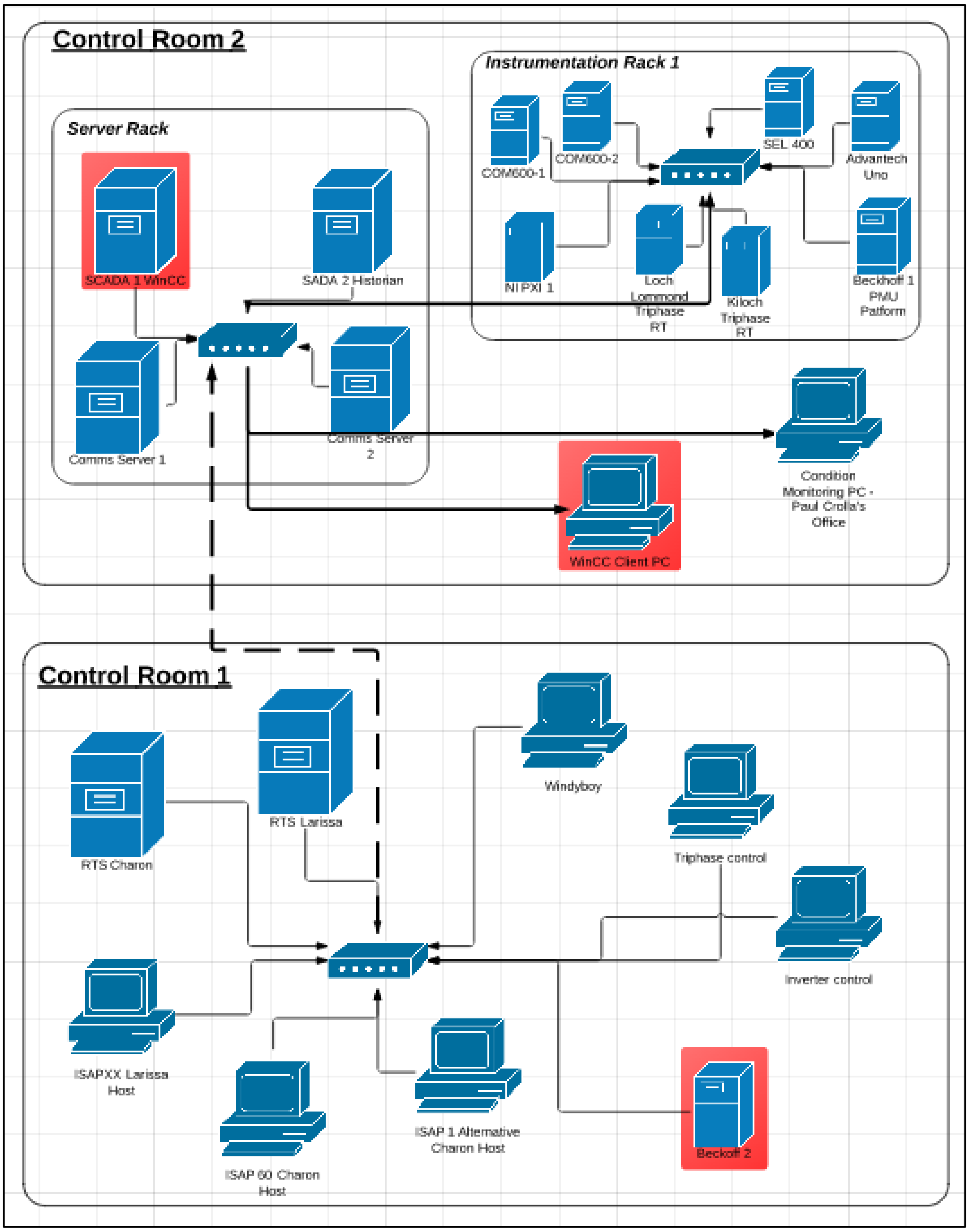


Figure 46 - Microgrid Communications Architecture

B. Code

B.1 GUIver1.m

```
function varargout = GUIver1(varargin)
% Code for the GUI that allows user to plot the CSV data
% 13/11/13 - Created by Gareth Mitchell - Ver 3.

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
   'gui_Singleton',  gui_Singleton, ...
   'gui_OpeningFcn', @GUIver1_OpeningFcn, ...
   'gui_OutputFcn',  @GUIver1_OutputFcn, ...
   'gui_LayoutFcn',  [] , ...
   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end


function GUIver1_OpeningFcn(hObject, eventdata, handles,  varargin)
% handles    structure with handles and user data (see GUIDATA)

csvFileName='TC_7_1.csv'; %choose input CSV file
csvFileSheet=csvFileName(1:(end-4)); %removes .csv extension
[~,dateTimeName,~]=xlsread(csvFileName,csvFileSheet,'B7'); %finds the date of the file
correctedDateTimeName=regexprep(dateTimeName,'[^\w'']','_'); %removes colons and spaces and replaces with underscore
islandFile=strcat(correctedDateTimeName{1},'.mat');


if(exist(islandFile))==2; %checks if .mat file exists
   load(islandFile);

else
   [IslandData]=ReadingInWholeFileVer3(csvFileName);
   SplittingInputOutputVer3(IslandData,correctedDateTimeName);
   load(islandFile);
end

handles.islandFile = islandFile; %saves islandFile to handles structure
handles.output = hObject;
```

```
handles.IslandIndexY1 = 1;
handles.IslandIndexY2 = 1;

handles.DataTypeIndexY1 = 1;
handles.DataTypeIndexY2 = 1;

handles.VariableIndexY1=1;
handles.VariableIndexY2=1;

handles.VariableNumX=0;

handles.VariableNumY1=0;
handles.lineColourY2=0;

handles.VariableNumY2=0;
handles.lineColourY2=0;

guidata(hObject, handles); %updates handle structure


% --- Outputs from this function are returned to the command line.
function varargout = GUIver1_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;


% --- Executes on selection change in popupIslandY1.
function popupIslandY1_Callback(hObject, eventdata, handles)

island_name= {'Green','Orange','Purple','Grey (NET)'}; %island names
set(handles.popupIslandY1,'String',island_name); %populates pop up menu
valY1 = get(hObject,'Value'); %determines which island is selected in menu
handles.IslandIndexY1 = valY1;
guidata(hObject,handles); %updates handle structure

% --- Executes during object creation, after setting all properties.
function popupIslandY1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupDataTypeY1.
function popupDataTypeY1_Callback(hObject, eventdata, handles)
```

```
datatype_name= {'Output Double','Output Integer','Output Boolean','Input Double','Input
Integer','Input Boolean'};
set(handles.popupDataTypeY1,'String',datatype_name); %populates pop up menu

valY1 = get(hObject,'Value'); %determines selected data type
handles.DataTypeIndexY1 = valY1;
guidata(hObject,handles);


% --- Executes during object creation, after setting all properties.
function popupDataTypeY1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupVariableY1.
function popupVariableY1_Callback(hObject, eventdata, handles)

islandFile = handles.islandFile;
load(islandFile);

IslandIndexY1 = handles.IslandIndexY1;
DataTypeIndex = handles.DataTypeIndexY1;


xvariable_name=AllHeaders{IslandIndexY1}{DataTypeIndex};
set(handles.popupVariableY1,'String',xvariable_name); %populates pop up menu
valY1 = get(hObject,'Value');
handles.VariableIndexY1 = valY1;
guidata(hObject,handles);


% --- Executes during object creation, after setting all properties.
function popupVariableY1_CreateFcn(hObject, eventdata, handles)


if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in plotButtonY.
```

```
function plotButtonY_Callback(hObject, eventdata, handles)


islandFile = handles.islandFile;
load(islandFile);
graphColours={[0 1 0],[255/255 97/255 3/255],[180/255 82/255 205/255],[84/255 84/255
84/255]}; %RGB values of Island colours

VariableIndexY1=handles.VariableIndexY1;
IslandIndexY1=handles.IslandIndexY1;
DataTypeIndexY1=handles.DataTypeIndexY1;

handles.lineColourY1=graphColours{IslandIndexY1};
handles.VariableNumX=AllNums{IslandIndexY1}{DataTypeIndexY1}(:,1); %selects time x-axis
handles.VariableNumY1=AllNums{IslandIndexY1}{DataTypeIndexY1}(:,VariableIndexY1);
%user selected y-axis


guidata(hObject,handles);

lineColourY1=handles.lineColourY1;
xvariableNum=handles.VariableNumX;
yvariableNum=handles.VariableNumY1;

xvariableNumCorrected=xvariableNum-xvariableNum(1);


xvariableName=strrep(AllHeaders{IslandIndexY1}{DataTypeIndexY1}(:,1),'_','\_'); %x-axis
name
yvariableName=strrep(AllHeaders{IslandIndexY1}{DataTypeIndexY1}(:,VariableIndexY1),'_','\
_'); %y-axis name

plot(xvariableNumCorrected,yvariableNum,'Color',lineColourY1);
xlabel(xvariableName);
ylabel(yvariableName);
title(strcat(yvariableName,{'  against  '},xvariableName));
xlim([min(xvariableNumCorrected) max(xvariableNumCorrected)]);
legend(yvariableName);



%________________The following is for plotting comparisons___________________


% --- Executes on selection change in popupIslandY2.
function popupIslandY2_Callback(hObject, eventdata, handles)
island_name= {'Green','Orange','Purple','Grey (NET)'};
```

```
set(handles.popupIslandY2,'String',island_name); %populates pop up menu
valY2 = get(hObject,'Value');
handles.IslandIndexY2 = valY2;
guidata(hObject,handles);


% --- Executes during object creation, after setting all properties.
function popupIslandY2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupDataTypeY2.
function popupDataTypeY2_Callback(hObject, eventdata, handles)
datatype_name= {'Output Double','Output Integer','Output Boolean','Input Double','Input
Integer','Input Boolean'};
set(handles.popupDataTypeY2,'String',datatype_name); %populates pop up menu

valY2 = get(hObject,'Value');
handles.DataTypeIndexY2 = valY2;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function popupDataTypeY2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupVariableY2.
function popupVariableY2_Callback(hObject, eventdata, handles)
islandFile = handles.islandFile;
load(islandFile);

IslandIndexY2 = handles.IslandIndexY2; %FINALLY WORKS
DataTypeIndexY2 = handles.DataTypeIndexY2;


xvariable_name=AllHeaders{IslandIndexY2}{DataTypeIndexY2};
set(handles.popupVariableY2,'String',xvariable_name); %populates pop up menu
valY2 = get(hObject,'Value');
handles.VariableIndexY2 = valY2;
```

```
guidata(hObject,handles);


% --- Executes during object creation, after setting all properties.
function popupVariableY2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in plotButtonY2.
function plotButtonY2_Callback(hObject, eventdata, handles)

islandFile = handles.islandFile;
load(islandFile);
graphColours={[0 1 0],[255/255 97/255 3/255],[180/255 82/255 205/255],[84/255 84/255
84/255]}; %

VariableIndexY1=handles.VariableIndexY1; %1st y variable = Y1
IslandIndexY1=handles.IslandIndexY1;
DataTypeIndexY1=handles.DataTypeIndexY1;

handles.lineColourY1=graphColours{IslandIndexY1};
handles.VariableNumY1=AllNums{IslandIndexY1}{DataTypeIndexY1}(:,VariableIndexY1);

%-----------------------------------------------------------------------

VariableIndexY2=handles.VariableIndexY2; %second y variable = Y2
IslandIndexY2=handles.IslandIndexY2;
DataTypeIndexY2=handles.DataTypeIndexY2;

handles.lineColourY2=graphColours{IslandIndexY2};
handles.VariableNumX=AllNums{IslandIndexY2}{DataTypeIndexY2}(:,1);
handles.VariableNumY2=AllNums{IslandIndexY2}{DataTypeIndexY2}(:,VariableIndexY2);


guidata(hObject,handles);

lineColourY1=handles.lineColourY1;
lineColourY2=handles.lineColourY2;
xvariableNum=handles.VariableNumX;
yvariableNum1=handles.VariableNumY1;
yvariableNum2=handles.VariableNumY2;

xvariableNumCorrected=xvariableNum-xvariableNum(1);
```

```
xvariableName=strrep(AllHeaders{IslandIndexY2}{DataTypeIndexY2}(:,1),'_','\_');
yvariableName1=strrep(AllHeaders{IslandIndexY1}{DataTypeIndexY1}(:,VariableIndexY1),'_',
'\_');
yvariableName2=strrep(AllHeaders{IslandIndexY2}{DataTypeIndexY2}(:,VariableIndexY2),'_',
'\_'); %right hand y-axis name

if lineColourY1==lineColourY2; %if two y-axis variables are from the same island...
  if lineColourY1==[84/255 84/255 84/255]; %changes colour of one axis to black

     lineColourY1=[153/255 0 0];
  else


     lineColourY1=[0, 0, 0];

  end

end

[AX,H1,H2]=plotyy(xvariableNumCorrected,yvariableNum1,xvariableNumCorrected,yvariabl
eNum2,'plot'); %plots two y-axes
set(H1,'color',lineColourY1);
set(H2,'color',lineColourY2);
set(AX,{'ycolor'},{lineColourY1;lineColourY2});
set(get(AX(1),'Ylabel'),'String',yvariableName1);
set(get(AX(2),'Ylabel'),'String',yvariableName2);
xlabel(xvariableName);
title(strcat(yvariableName1,{'  and  '},yvariableName2,{'  against  '},xvariableName));
```

B.2 ReadingInWholeFileVer3.m

```
% Code that reads in CSV files and splits data up accordingly
% 30/10/13 - Created by Gareth Mitchell - Ver 3.

function [IslandData]=ReadingInWholeFileVer3(csvFileName)

csvSheet=csvFileName(1:(end-4)); %removes the .csv extension

[~,HeaderText,~]=xlsread(csvFileName,csvSheet,'A11:JT11'); %column headers
numFile=csvread(csvFileName,12,0); %where the variable numerical values start, 12th row

for i=1:length(HeaderText);

   newString(i)={HeaderText{i}(36:end)}; %removes Larissa part

end

expressionTime={'ELAPSED_TIME'};
expression1Files={'DG1' 'DG2' 'DG3' 'NET'}; %used for searching through the data
expression2Files={'LPC1' 'LPC2' 'GSP' 'NET'};
% islandNumFiles={'GreenNum','OrangeNum','PurpleNum'};
% islandHeaderFiles={'GreenHeader','OrangeHeader','PurpleHeader'};
islandNames={'Green','Orange','Purple','NET'};

for i=1:4;

   str=newString;

   expression1=expression1Files{i};
   expression2=expression2Files{i};
   island=islandNames{i};

   matchTime=regexp(str,expressionTime,'match'); %searches data for island
   matchStr1=regexp(str,expression1,'match');
   matchStr2=regexp(str,expression2,'match');

   if i<4;

      matchStr=strcat(matchTime,matchStr1,matchStr2); %for green, orange, purple islands

   else
      matchStr=strcat(matchTime,matchStr1); %for the grey (NET) island

   end

   emptycells=1-cellfun(@isempty,matchStr);
   value=find(emptycells == 1);  %indicates which columns contain the searched for term
```

```
    islandNames{i}=newString(:,value); %finds values from corresponding column
    islandNum{i}=numFile(:,value);
end

greenHeader=islandNames{1};
orangeHeader=islandNames{2};
purpleHeader=islandNames{3};
NETHeader=islandNames{4};

ColumnHeaders={greenHeader,orangeHeader,purpleHeader,NETHeader}; %nested Cell with
sorted island variable names

greenNum=islandNum{1};
orangeNum=islandNum{2};
purpleNum=islandNum{3};
NETNum=islandNum{4};

ColumnNumbers={greenNum,orangeNum,purpleNum,NETNum}; %nested cell with sorted
island variable values

IslandData={ColumnHeaders,ColumnNumbers}; %nested cell that negates the need to save
a .mat file
end
```

B.3 SplittingInputOutputVer3

```
% Code that saves island data as a .mat file using nested cells
% od=output double, ob=output boolean, oi=output integer
% id=input double, ib=input boolean, ii=nput integer
% 28/10/13 - Created by Gareth Mitchell - Ver 3.

function SplittingInputOutputVer3(IslandData,correctedDateTimeName)

expression={'od_','oi_','ob_','id_','ii_','ib_'}; %data types


for i=1:length(IslandData{1}); %counts the amount of power islands

  string = IslandData{1}{i};
  headerFile=IslandData{1}{i};
  numberFile=IslandData{2}{i};

  for n=1:length(expression);
    expressionComp = expression{n};
    matchString = regexp(string,expressionComp,'match'); %matches cells to criteria in
'expression'
    Cells = 1 - cellfun(@isempty,matchString); %finds the cells that match
    matchingCellsWOutTime=find(Cells == 1);
    matchingCells=[1 matchingCellsWOutTime]; %time included
    headerNames{n}=headerFile(:,matchingCells); %saves sorted data into headerNames
    headerNums{n}=numberFile(:,matchingCells);%saves sorted data into headerNums


  end

  if i == 1;


    GreenHeader={headerNames{1},headerNames{2},headerNames{3}...
      headerNames{4},headerNames{5},headerNames{6}};
    GreenNum={headerNums{1},headerNums{2},headerNums{3}...
      headerNums{4},headerNums{5},headerNums{6}};


  else  if i == 2;

      OrangeHeader={headerNames{1},headerNames{2},headerNames{3}...
        headerNames{4},headerNames{5},headerNames{6}};
      OrangeNum={headerNums{1},headerNums{2},headerNums{3}...
        headerNums{4},headerNums{5},headerNums{6}};
```

```
        else  if i == 3;


            PurpleHeader={headerNames{1},headerNames{2},headerNames{3}...
              headerNames{4},headerNames{5},headerNames{6}};
            PurpleNum={headerNums{1},headerNums{2},headerNums{3}...
              headerNums{4},headerNums{5},headerNums{6}};




          else  if i == 4;

              GreyHeader={headerNames{1},headerNames{2},headerNames{3}...
                headerNames{4},headerNames{5},headerNames{6}};
              GreyNum={headerNums{1},headerNums{2},headerNums{3}...
                headerNums{4},headerNums{5},headerNums{6}};


            end

          end
        end
      end
    end

AllHeaders={GreenHeader,OrangeHeader,PurpleHeader,GreyHeader};
AllNums={GreenNum,OrangeNum,PurpleNum,GreyNum};

save(correctedDateTimeName{1},'AllHeaders','AllNums'); %saves to a .mat file
```

B.4 main.c

```c
//  main.c
//  test
//
//  Created by Connor Hughes on 18/11/2013.

#include <stdio.h>  //includes standard input output
#include <stdlib.h> //includes standard library - for fopen and fclose functions

        void Test_Function(long int i){  //defines Tes_Funtion with input i

        #define COLS 280   //defines the number of columns in the csv file
   #define ROWS 60677 //defines the number of rows in the csv file

   long int j,k,l;    //used for reading through the csv file
   float read[COLS];  //sets up array to temporalary store column values
   float tag[COLS];   //sets up array to update tag values - for printing


   FILE *csv=fopen("E:/sc1_5pc_d.csv", "r");  //opens .csv file for reading
        //Note the memory stick with path E:/ is being used to mimic the server
        //In practice this path will be changed to access the correct server for either
historical data or real time functionality

  if(!csv) {  //if csv file does not exist or hsa not been opened
      printf("Error opening file"); //Error opening file
   }

    for(j=0;j<ROWS;j++){  //counts through the rows of the csv file
   for(k=0;k<COLS;k++) {   //counts through the columns of the csv file

      fscanf(csv,"%f%*c",&read[k]);  //scans each comma seperated value
      if(j==i){
                        //if the row count is equal to  the value for i, the row to be used for
tag updating
          tag[k] = read [k]; //update tag array with read values
          }
      }
        //note the above code is configured to read the entire csv file before selecting the
row of data needed
        //this is to ensure that the code takes the same amount of time to execute on each
iteration


        //printf statements print the first 5 and last 5 values of the given csv row

         printf("------Row %d ------\n",i+1);
```

```
     printf("---Fisrt 5 Tags---\n)");
    for(l=0;l<5;l++){    //for first 5 columns

            printf("%.8f\n",tag[l]); //prints tag float with 8 decimal places
    }
            printf("---Last 5 Tags---\n");
    for(l=COLS-5;l<COLS;l++){  //for last 5 columns

            printf("%.8f\n",tag[l]); //prints tag float with 8 decimal places
    }
    }
            printf("-----------------\n");
            printf("-----------------\n");


  fclose(csv);  //closes the csv file
    }

    int main() {

   long int i = 0;
//used to define which row of the csv file is printed

   // long int d = 0; //Use for delay coding
    while (i<60677){
//while i is less than the number of rows in the csv file compte Test_Fucntion
            Test_Function(i);

  // for(d=0; d<60677-i; d++);
            //Will delay the code by a set time, not needed at present

   i++; //increments the row count
    }
    return 0; //returns int 0 as required by 'non-void' function
    }
```

B.5 Test_Function.fct

```
void Test_Function(long int i)
{
  #define COLS 280 //defines the number of columns in the CSV file
  #define ROWS 60677  //defines the number of rows in the CSV file

   long int j,k,l;  //ints used to counts through the CSV file
   float tag[COLS];  //tag array to store all variable values for tag assignment
   float read[COLS]; //read array to read all variable values from the CSV file
   FILE *csv=fopen("E:\sc1_5pc_d.csv", "r"); //opens CSV file
       //Path will connect to relevant server for final system
   if(!csv) {   //if CSV file does not open correctly or cannot be found
     printf("Error Opening CSV File\n"); //prints error statemen to GSC Diagnostics
   }
   for (j=0;j<ROWS;j++){  //reads through all rows of the CSV file

   for(k=0;k<COLS;k++) {   //reads through all columns of the CSV file
      fscanf(csv,"%f%*c",&read[k]); //scans CSV file placing values in read array
       if(j==i){
         tag[k] = read[k]; //if the row is equal to the desired row (i) then update tag array
       }
         }
       }
//----------UPDATES THE TAGS--------------------------------
SetTagFloat("ADI_ELAPSED_TIME",tag[0]);
SetTagFloat("id_LOAD_DGIx_State2_Holdtime",tag[1]);
SetTagFloat("id_Switch_DG1_C_OAC",tag[2]);
SetTagFloat("ii_LOAD_DG2_State",tag[3]);
SetTagFloat("od_LOAD_DG1_SequenceTimeToEnd",tag[4]);
SetTagFloat("od_NET_Gen_Pcontrol_FKi",tag[5]);
SetTagFloat("od_LPC2_Gen_RMS_Volts_PosSeq",tag[6]);
SetTagFloat("ob_NET_Gen_PhaseLock",tag[7]);
SetTagFloat("id_NET_Gen_PControl_Kff",tag[8]);
SetTagFloat("ib_LOAD_DGIx_SequenceStart",tag[9]);
SetTagFloat("od_NET_Gen_Pcontrol_FKt",tag[10]);
SetTagFloat("ib_NET_Gen_P_Move_LPFs_to_Kdt",tag[11]);
SetTagFloat("id_LOAD_DG1_Sheddable_8P_State1",tag[12]);
SetTagFloat("od_NET_Gen_F_ref_Hz",tag[13]);
SetTagFloat("id_LOAD_DG2_Fixed_P_State2",tag[14]);
SetTagFloat("od_GSP_A_Output_VARs",tag[15]);
SetTagFloat("od_NET_Gen_QControl_D_pu",tag[16]);
SetTagFloat("od_NET_Gen_I_Unbalance",tag[17]);
SetTagFloat("ii_NET_Gen_PhaseLock_Type",tag[18]);
SetTagFloat("ii_NET_Gen_Scenario",tag[19]);
SetTagFloat("od_LPC2_Bus_Load_RealPower",tag[20]);
SetTagFloat("ib_IncoherentCount_ResetMaxHold",tag[21]);
SetTagFloat("od_NET_Gen_Output_VARs",tag[22]);
```

```
SetTagFloat("od_LPC2_Net_RMS_Volts_PosSeq",tag[23]);
SetTagFloat("id_Loadbank_DG1_Q_Pf_Manual",tag[24]);
SetTagFloat("od_NET_Gen_QControl_FF_pu",tag[25]);
SetTagFloat("ib_NET_Gen_SequenceStart_1",tag[26]);
SetTagFloat("od_LPC2_Gen_I_Unbalance_pct",tag[27]);
SetTagFloat("od_NET_Gen_HIL_Qe_pu",tag[28]);
SetTagFloat("id_Switch_DG2_to_EastWall_OAC",tag[29]);
SetTagFloat("od_NET_Gen_RPMRaw",tag[30]);
SetTagFloat("od_NET_Gen_Pcontrol_FKat",tag[31]);
SetTagFloat("id_LOAD_DG1_LOAD_Pf_State2",tag[32]);
SetTagFloat("id_NET_Gen_V_Droop_Pct",tag[33]);
SetTagFloat("od_LPC1_Bus_RMS_Volts_PosSeq",tag[34]);
SetTagFloat("ob_Switch_DG3_C",tag[35]);
SetTagFloat("ob_NET_Gen_Softstart_active",tag[36]);
SetTagFloat("id_LOAD_DG1_QMultiplier_State2",tag[37]);
SetTagFloat("od_LPC2_Net_Frequency",tag[38]);
SetTagFloat("ob_Clock_StopLED",tag[39]);
SetTagFloat("od_NET_Gen_F_ref_ROCOF_Hzs",tag[40]);
SetTagFloat("ob_Switch_DG1_DomesticLoads",tag[41]);
SetTagFloat("ob_Switch_DG2_IndMachine_7p5kW_A",tag[42]);
SetTagFloat("id_NET_Gen_Manual_Current_Limit_pu",tag[43]);
SetTagFloat("od_LPC2_Gen_RealPower",tag[44]);
SetTagFloat("id_Loadbank_GSP_P_Watts_Manual",tag[45]);
SetTagFloat("ob_Switch_DG1_IndMachine_5p5kW",tag[46]);
SetTagFloat("od_Loadbank_DG1_P_Watts_Set",tag[47]);
SetTagFloat("id_Switch_DG1_to_DG3_A_OAC",tag[48]);
SetTagFloat("od_NET_Gen_Efficiency_pct",tag[49]);
SetTagFloat("od_LPC1_Gen_RealPower",tag[50]);
SetTagFloat("od_NET_Gen_QControl_I_pu",tag[51]);
SetTagFloat("id_Switch_DG1_IndMachine_2p2kW_OAC",tag[52]);
SetTagFloat("od_GSP_Phase_lead_deg",tag[53]);
SetTagFloat("id_NET_Gen_SpeedControl_pu",tag[54]);
SetTagFloat("id_LOAD_DG1_Sheddable_2P_State1",tag[55]);
SetTagFloat("id_LOAD_DG2_LOAD_Pf_State1",tag[56]);
SetTagFloat("od_Loadbank_DG2_Q_Vars_Set",tag[57]);
SetTagFloat("od_NET_Gen_Phase_Error_SquareLaw",tag[58]);
SetTagFloat("id_LOAD_DG1_Sheddable_5P_State2",tag[59]);
SetTagFloat("ib_NET_Gen_Enable",tag[60]);
SetTagFloat("id_Switch_DG3_F_OAC",tag[61]);
SetTagFloat("id_Loadbank_DG2_P_Watts_Manual",tag[62]);
SetTagFloat("ob_NET_Gen_TachoFrequencyMismatch_Alarm",tag[63]);
SetTagFloat("od_NET_Gen_Pcontrol_FKdt",tag[64]);
SetTagFloat("id_NET_Gen_State2_Holdtime",tag[65]);
SetTagFloat("od_GSP_A_Output_Watts",tag[66]);
SetTagFloat("id_LOAD_DG2_PMultiplier_State2",tag[67]);
SetTagFloat("id_LOAD_DG2_Sheddable_4P_State1",tag[68]);
SetTagFloat("od_LPC1_Gen_I_Unbalance_pct",tag[69]);
```

```
SetTagFloat("id_NET_Throttle_LPF",tag[70]);
SetTagFloat("id_LOAD_DG2_Sheddable_3P_State1",tag[71]);
SetTagFloat("id_NET_Gen_P_set_for_no_F_droop_pu",tag[72]);
SetTagFloat("ob_Switch_DG2_to_EastWall",tag[73]);
SetTagFloat("od_GSP_V_Unbalance_pct",tag[74]);
SetTagFloat("od_NET_Gen_Pcontrol_PKt",tag[75]);
SetTagFloat("od_NET_Gen_Pcontrol_PKd",tag[76]);
SetTagFloat("id_Switch_DG2_A_OAC",tag[77]);
SetTagFloat("id_Switch_DG2_IndMachine_7p5kW_A_OAC",tag[78]);
SetTagFloat("od_LPC1_Net_ReactivePower",tag[79]);
SetTagFloat("id_LOAD_DG2_Sheddable_8P_State1",tag[80]);
SetTagFloat("ib_NET_Gen_SequenceAbort_1",tag[81]);
SetTagFloat("od_Loadbank_GSP_P_Watts_Set",tag[82]);
SetTagFloat("ob_Switch_DG2_A",tag[83]);
SetTagFloat("ob_Switch_DG1_D",tag[84]);
SetTagFloat("ob_Switch_DG1_B",tag[85]);
SetTagFloat("ob_STOP",tag[86]);
SetTagFloat("id_LOAD_DG1_Sheddable_7P_State1",tag[87]);
SetTagFloat("id_LOAD_DG1_Sheddable_7P_State2",tag[88]);
SetTagFloat("id_Switch_DG1_DomesticLoads_OAC",tag[89]);
SetTagFloat("id_NET_Gen_Volt_State1_pu",tag[90]);
SetTagFloat("od_LPC2_Bus_Load_ReactivePower",tag[91]);
SetTagFloat("ib_Loadbank_GSP_NoWatts",tag[92]);
SetTagFloat("od_GSP_Busbar_Frequency",tag[93]);
SetTagFloat("od_LPC2_Bus_RMS_Volts_PosSeq",tag[94]);
SetTagFloat("ob_Larissa_MustStop",tag[95]);
SetTagFloat("ob_Switch_DG1_to_DG2",tag[96]);
SetTagFloat("od_NET_Gen_PControl_FF_pu",tag[97]);
SetTagFloat("od_NET_Gen_Drooped_F_target_Hz",tag[98]);
SetTagFloat("ob_Switch_DG3_D",tag[99]);
SetTagFloat("id_LOAD_DG1_Sheddable_1P_State2",tag[100]);
SetTagFloat("id_Switch_DG2_to_DG1_OAC",tag[101]);
SetTagFloat("ob_Switch_DG1_IndMachine_2p2kW",tag[102]);
SetTagFloat("ob_Switch_DG1_A",tag[103]);
SetTagFloat("id_Loadbank_GSP_Manual_Cycle_Freq",tag[104]);
SetTagFloat("id_LOAD_DG2_Sheddable_5P_State1",tag[105]);
SetTagFloat("id_Switch_DG3_E_OAC",tag[106]);
SetTagFloat("od_NET_Gen_PControl_P_pu",tag[107]);
SetTagFloat("od_NET_Gen_Pcontrol_PKdt",tag[108]);
SetTagFloat("id_LOAD_DG1_Sheddable_3P_State1",tag[109]);
SetTagFloat("id_NET_Gen_State2to1_Ramptime",tag[110]);
SetTagFloat("od_GSP_Busbar_Volts_PosSeq",tag[111]);
SetTagFloat("id_Switch_DG3_A_OAC",tag[112]);
SetTagFloat("id_NET_Gen_Xs_pu",tag[113]);
SetTagFloat("id_LOAD_DG2_Sheddable_7P_State2",tag[114]);
SetTagFloat("od_NET_Gen_Field_V",tag[115]);
SetTagFloat("od_NET_Gen_Pcontrol_FKa",tag[116]);
```

```
SetTagFloat("od_LPC1_Bus_Load_ReactivePower",tag[117]);
SetTagFloat("od_NET_Gen_QControl_P_pu",tag[118]);
SetTagFloat("id_Loadbank_DG2_Q_Pf_Manual",tag[119]);
SetTagFloat("id_NET_Gen_Q_Kp",tag[120]);
SetTagFloat("id_LOAD_DG1_Fixed_P_State2",tag[121]);
SetTagFloat("id_NET_Gen_F_Droop_Pct",tag[122]);
SetTagFloat("id_NET_Gen_Rs_pu",tag[123]);
SetTagFloat("id_Switch_DG3_B_OAC",tag[124]);
SetTagFloat("od_NET_Gen_RPM",tag[125]);
SetTagFloat("od_NET_Gen_DC_Motor_Armature_I",tag[126]);
SetTagFloat("od_NET_Gen_Output_Watts",tag[127]);
SetTagFloat("od_LPC2_Bus_Frequency",tag[128]);
SetTagFloat("id_Switch_DG3_D_OAC",tag[129]);
SetTagFloat("od_LPC2_Gen_Frequency",tag[130]);
SetTagFloat("od_NET_Gen_SequenceTimeFromStart",tag[131]);
SetTagFloat("od_NET_Gen_Dynamo_V",tag[132]);
SetTagFloat("id_NET_Gen_Freq_State1",tag[133]);
SetTagFloat("ii_Microgrid_Phase_Reference",tag[134]);
SetTagFloat("od_LPC2_Gen_V_Unbalance_pct",tag[135]);
SetTagFloat("od_GSP_C_Output_Watts",tag[136]);
SetTagFloat("id_Switch_DG1_B_OAC",tag[137]);
SetTagFloat("od_NET_Gen_Field_I",tag[138]);
SetTagFloat("id_LOAD_DG2_Sheddable_1P_State1",tag[139]);
SetTagFloat("ii_Deck_Selector",tag[140]);
SetTagFloat("od_NET_Gen_Field_I_pu",tag[141]);
SetTagFloat("od_DG3_A_I_Lead_V_deg",tag[142]);
SetTagFloat("id_NET_Gen_Ext_Ref_PhaseAdvance_us",tag[143]);
SetTagFloat("id_Switch_DG1_to_DG2_OAC",tag[144]);
SetTagFloat("id_Switch_DG2_C_OAC",tag[145]);
SetTagFloat("ib_ZeroLoadBanksDG1DG2",tag[146]);
SetTagFloat("id_NET_Gen_P_MaxPhaseErrorDeg",tag[147]);
SetTagFloat("id_LOAD_DG2_Sheddable_6P_State2",tag[148]);
SetTagFloat("ob_NET_Gen_PhaseLock_Freewheel",tag[149]);
SetTagFloat("od_NET_Gen_DC_Motor_Armature_V",tag[150]);
SetTagFloat("od_LPC1_Net_Frequency",tag[151]);
SetTagFloat("id_Switch_DG2_IndMachine_7p5kW_B_OAC",tag[152]);
SetTagFloat("id_Switch_DG2_B_OAC",tag[153]);
SetTagFloat("ii_Loadbank_DG2_Control_Method",tag[154]);
SetTagFloat("id_Switch_DG1_E_OAC",tag[155]);
SetTagFloat("id_NET_Gen_PKp_boost",tag[156]);
SetTagFloat("ii_LOAD_DG1_State",tag[157]);
SetTagFloat("ob_Switch_DG2_C",tag[158]);
SetTagFloat("id_NET_Gen_PowerRating_FF",tag[159]);
SetTagFloat("od_LPC1_Gen_V_Unbalance_pct",tag[160]);
SetTagFloat("od_LPC2_Net_RealPower",tag[161]);
SetTagFloat("od_LPC1_Bus_Frequency",tag[162]);
SetTagFloat("id_LOAD_DG2_Sheddable_2P_State1",tag[163]);
```

```
SetTagFloat("ob_Switch_DG1_to_DG3_A",tag[164]);
SetTagFloat("od_NET_Gen_V_Unbalance",tag[165]);
SetTagFloat("od_NET_Gen_PControl_D_pu",tag[166]);
SetTagFloat("od_NET_Gen_Pcontrol_PKa",tag[167]);
SetTagFloat("ob_Switch_DG3_F",tag[168]);
SetTagFloat("id_NET_Gen_FHzMod_SwitchToSquareDeg",tag[169]);
SetTagFloat("id_NET_Gen_Droop_LPF_Fc",tag[170]);
SetTagFloat("id_LOAD_DG1_Sheddable_6P_State2",tag[171]);
SetTagFloat("oi_Deck_Selector",tag[172]);
SetTagFloat("id_NET_Gen_Manual_Field_Control_pu",tag[173]);
SetTagFloat("id_LOAD_DG2_Sheddable_4P_State2",tag[174]);
SetTagFloat("od_LPC1_Net_RMS_Volts_PosSeq",tag[175]);
SetTagFloat("od_LPC1_Gen_RMS_Volts_PosSeq",tag[176]);
SetTagFloat("od_DG1_Phase_lead_deg",tag[177]);
SetTagFloat("ob_NotSTOP",tag[178]);
SetTagFloat("ib_NET_Gen_HIL_Include_DG3_in_load",tag[179]);
SetTagFloat("id_NET_Gen_P_FreqResponse_t_a_secs",tag[180]);
SetTagFloat("id_NET_Gen_State1to2_Ramptime",tag[181]);
SetTagFloat("id_NET_Gen_FKi_boost",tag[182]);
SetTagFloat("id_LOAD_DG1_Sheddable_5P_State1",tag[183]);
SetTagFloat("ob_Switch_DG2_B",tag[184]);
SetTagFloat("ii_Loadbank_DG1_Control_Method",tag[185]);
SetTagFloat("id_LOAD_DG1_LOAD_Pf_State1",tag[186]);
SetTagFloat("id_Loadbank_DG1_P_Watts_Manual",tag[187]);
SetTagFloat("od_Larissa_turnaround_delay_frames",tag[188]);
SetTagFloat("id_Switch_DG1_IndMachine_5p5kW_OAC",tag[189]);
SetTagFloat("id_NET_Gen_Q_Kd_LPF_Fc",tag[190]);
SetTagFloat("ob_Switch_DG1_E",tag[191]);
SetTagFloat("ob_Switch_DG1_C",tag[192]);
SetTagFloat("od_NET_Gen_RMS_Volts_PosSeq_pu",tag[193]);
SetTagFloat("od_LPC1_Gen_ReactivePower",tag[194]);
SetTagFloat("id_LOAD_DG2_Sheddable_5P_State2",tag[195]);
SetTagFloat("od_NET_Gen_Pcontrol_FKd",tag[196]);
SetTagFloat("id_LOAD_DG1_Sheddable_4P_State1",tag[197]);
SetTagFloat("oi_Incoherent_Scramnet_MaxCount",tag[198]);
SetTagFloat("id_LOAD_DG2_QMultiplier_State2",tag[199]);
SetTagFloat("od_GSP_C_Output_VARs",tag[200]);
SetTagFloat("id_NET_Gen_FKa_factor_0to1",tag[201]);
SetTagFloat("od_NET_Gen_FreqTarget_Hz",tag[202]);
SetTagFloat("od_NET_Gen_V_ref_pu",tag[203]);
SetTagFloat("od_NET_Gen_DC_Motor_Armature_Watts",tag[204]);
SetTagFloat("ob_Switch_DG3_E",tag[205]);
SetTagFloat("id_LOAD_DG1_Sheddable_8P_State2",tag[206]);
SetTagFloat("id_LOAD_DG1_PMultiplier_State2",tag[207]);
SetTagFloat("ob_Incoherent_Scramnet_StopLED",tag[208]);
SetTagFloat("od_LPC2_Net_ReactivePower",tag[209]);
SetTagFloat("od_Loadbank_GSP_Q_Vars_Set",tag[210]);
```

```
SetTagFloat("ib_NET_Gen_SequenceAbort_2",tag[211]);
SetTagFloat("od_LPC1_Bus_Load_RealPower",tag[212]);
SetTagFloat("id_NET_Gen_Q_Kd",tag[213]);
SetTagFloat("od_NET_Gen_Drooped_V_target_pu",tag[214]);
SetTagFloat("ib_NET_Gen_PhaseLock_Activate",tag[215]);
SetTagFloat("od_LPC1_Net_RealPower",tag[216]);
SetTagFloat("ob_Switch_DG3_B",tag[217]);
SetTagFloat("id_NET_Gen_HIL_1pu_Scale_Power",tag[218]);
SetTagFloat("od_Loadbank_DG2_P_Watts_Set",tag[219]);
SetTagFloat("od_NET_Gen_RMS_Volts_PosSeq",tag[220]);
SetTagFloat("id_LOAD_DG2_Fixed_P_State1",tag[221]);
SetTagFloat("id_LOAD_DG1_Sheddable_4P_State2",tag[222]);
SetTagFloat("ib_Loadbank_GSP_Manual",tag[223]);
SetTagFloat("od_NET_Gen_Frequency",tag[224]);
SetTagFloat("od_NET_Gen_Throttle_Control_pu",tag[225]);
SetTagFloat("ob_Switch_DG2_to_DG1",tag[226]);
SetTagFloat("id_NET_Gen_P_Kd_factor_0to1",tag[227]);
SetTagFloat("od_NET_Gen_PControl_A_pu",tag[228]);
SetTagFloat("id_LOAD_DG2_Sheddable_7P_State1",tag[229]);
SetTagFloat("od_NET_Gen_Pcontrol_PKat",tag[230]);
SetTagFloat("id_NET_Gen_P_PhaseResponse_t_b_secs",tag[231]);
SetTagFloat("ob_Switch_DG2_E",tag[232]);
SetTagFloat("id_NET_Gen_PKa_factor_0to1",tag[233]);
SetTagFloat("id_LOAD_DG2_Sheddable_1P_State2",tag[234]);
SetTagFloat("id_Switch_DG3_C_OAC",tag[235]);
SetTagFloat("id_NET_Gen_Q_Ki",tag[236]);
SetTagFloat("id_Switch_DG2_D_OAC",tag[237]);
SetTagFloat("id_LOAD_DG1_Sheddable_2P_State2",tag[238]);
SetTagFloat("oi_Incoherent_Scramnet_MaxCountMax",tag[239]);
SetTagFloat("id_Switch_DG2_E_OAC",tag[240]);
SetTagFloat("od_NET_Gen_PControl_I_pu",tag[241]);
SetTagFloat("od_NET_Gen_PhaseLockErrorDegs",tag[242]);
SetTagFloat("ib_LOAD_DGIx_SequenceAbort",tag[243]);
SetTagFloat("ob_Switch_DG2_D",tag[244]);
SetTagFloat("id_LOAD_DG1_Sheddable_6P_State1",tag[245]);
SetTagFloat("id_LOAD_DGIx_State1to2_Ramptime",tag[246]);
SetTagFloat("id_LOAD_DG2_Sheddable_2P_State2",tag[247]);
SetTagFloat("od_LPC1_Gen_Frequency",tag[248]);
SetTagFloat("ob_Switch_DG3_A",tag[249]);
SetTagFloat("id_LOAD_DG2_Sheddable_3P_State2",tag[250]);
SetTagFloat("ib_STOP_LarissaGUIButton",tag[251]);
SetTagFloat("id_NET_Gen_Q_Kff",tag[252]);
SetTagFloat("id_LOAD_DGIx_State2to1_Ramptime",tag[253]);
SetTagFloat("id_LOAD_DG1_Fixed_P_State1",tag[254]);
SetTagFloat("id_Switch_DG1_A_OAC",tag[255]);
SetTagFloat("id_LOAD_DG2_Sheddable_8P_State2",tag[256]);
SetTagFloat("od_NET_Gen_Field_Watts",tag[257]);
```

```
SetTagFloat("id_LOAD_DG2_Sheddable_6P_State1",tag[258]);
SetTagFloat("ib_NET_Gen_P_AddKaLPFs",tag[259]);
SetTagFloat("id_LOAD_DG1_Sheddable_3P_State2",tag[260]);
SetTagFloat("id_Switch_DG1_D_OAC",tag[261]);
SetTagFloat("od_NET_Gen_Pcontrol_PKp",tag[262]);
SetTagFloat("od_NET_Gen_HIL_Pe_pu",tag[263]);
SetTagFloat("ob_Switch_DG2_IndMachine_7p5kW_B",tag[264]);
SetTagFloat("od_GSP_A_I_Unbalance_pct",tag[265]);
SetTagFloat("od_NET_Gen_QControl_pu",tag[266]);
SetTagFloat("ib_NET_Gen_SequenceStart_2",tag[267]);
SetTagFloat("id_LOAD_DG1_Sheddable_1P_State1",tag[268]);
SetTagFloat("od_NET_Gen_SequenceTimeToEnd",tag[269]);
SetTagFloat("ob_Charon_StopLED",tag[270]);
SetTagFloat("id_NET_Gen_Volt_State2_pu",tag[271]);
SetTagFloat("od_NET_Gen_Pcontrol_FKp",tag[272]);
SetTagFloat("od_Loadbank_DG1_Q_Vars_Set",tag[273]);
SetTagFloat("id_LOAD_DG2_LOAD_Pf_State2",tag[274]);
SetTagFloat("od_Master_clock",tag[275]);
SetTagFloat("od_LPC2_Gen_ReactivePower",tag[276]);
SetTagFloat("id_Loadbank_GSP_Q_Pf_Manual",tag[277]);
SetTagFloat("id_NET_Gen_Freq_State2",tag[278]);
SetTagFloat("od_DG2_Phase_lead_deg",tag[279]);

  fclose(csv);  //closes CSV file
}
```

C. Risk Register

Risk ID	Risk Title	Date Identified	Time Criticality	Project Members	Likelihood	Impact	Class	Mitigations
#1	Unable to set up server	08/11/2013	Week 15 (S2)	Connor, James	M	H	H	Gain administrative access and understand server operation
#2	Lack of access to historical data	04/11/2013	Week 10 (S1)	Gareth, Scott	M	M	H	Ask Paul Crolla for access to additional past scenarios
#3	Lack of access to control room	30/10/2013	Week 15 - 24 (S2) & Trade Show Booking	All	M	H	H	Invited to microgrid timetable
#4	Server and WinCC compatability issues	08/11/2013	Week 19 (S1)	All	H	H	H	Downgrade/update operating systems on Control Room 2 computers
#5	Downtime of microgrid	11/11/2013	Week 8 - 12 (S1), Week 15 - 24 (S2) & Trade Show Booking	All	L	H	M	Work from home/alternative labs. Deviate to alternative tasks to make best use of time
#6	Loss of data and results	30/10/2013	Week 12 (S1), Week 16 (S2)	Gareth, Scott	L	L	M	Continued back up thorughout project progress
#7	WinCC and real time data compatability issues	30/10/2013	Week 17 (S2)	Connor, James	H	H	H	Research, assign more group members, approach appropriate university employees for additional help
#8	Steep learning curve of WinCC and it's features	30/10/2013	Week 11 (S1)	Connor, James	H	H	H	Assign more group members or divide tasks for easier learning
#9	Illness of project members	30/10/2013	Throughout Project	All	L	L	L	Spread task of ill member between other group members (if it takes priority)
#10	Tripping hazards in control room	25/11/2013	Throughout Project	All	L	L	L	Clear control room to improve safety
#11	Loose live connections in control room	25/11/2013	Throughout Project	All	L	L	L	Identify problem to Richard/other technicians
#12	Webcam installation issues	11/11/2013	Week 26 (S2)	Undecided	L	L	M	Approach Richard for assistance
#13	Incorporating customer requirements into GUI design	23/10/2013	Week 26 (S2)	All	L	H	M	Assign more group members to task as customer is priority
#14	Falling behind in project plan	11/11/2013	Throughout Project	All	M	H	H	Update Gantt chart and project flow. Assess project progression and undertake mitigation actions if required
#15	C code issues with WinCC to read in historical data	18/11/2013	Week 11 (S1) complete before Interim Report	Connor, James	M	H	H	Assign additional group members, find SQL server solutions
#16	Understanding historical data in relation to microgrid's behaviour	20/11/2013	Week 11 (S1) complete before Interim Report	Scott	L	M	M	Background research ahead of interim report hand in
#17	Unable to integrate fault recorder software/hardware	15/11/2013	Week 17 (S2)	Scott	M	L	L	Assign additional group members, do not use for realisation of Control Room (adapt Wincc Visualisation)
#18	No project member assigned	18/11/2013	Week 11 (S1) before Interim Report	All	L	H	H	Contact Head of Department until mentor assigned (Mitigated)

*Group members are invited to update Risk Register upon realising new risks.

L	Low
M	Medium
H	High

D. Initial Project Proposal

Department of Electronic & Electrical Engineering

MEng/BEng in EEE/CES
19.520 MEng Project

This project registration form is organised in four parts:

PART 1: STATEMENT OF INTENT
PART 2: PROJECT WORK PLAN
PART 3: RESOURCE REQUIREMENT
PART 4: RISK ASSESSMENT
PART 5: SAFETY DECLARATION

0. All parts of the form must be completed jointly by group members and lodged with in R2.06 (EEE Resource Centre) by 13.30 on 31st October 2013.
1. Copies of the completed form will be sent to the Project Mentor
2. The group is advised to retain a copy of the completed form for future reference.

Group Names

1) Scott Clark	2) Connor Hughes	3) Jennifer Ingram
4) James McNiven	5) Gareth Mitchell	
Project Title: Realisation of a Smart Grid Control Room for a Microgrid		

PART 1: STATEMENT OF INTENT

The purpose of this section is: (i) to provide a concise description of the project, and (ii) to state a set of objectives that will provide the guide for assessing the project. Students should note the importance of item (ii), which should be discussed in detail with their project sponsor.

A. Project Description:

With the development and deployment of smart grid schemes to power networks (often incorporating significant penetrations of renewables) it becomes necessary to rapidly increase the volumes of monitoring data captured. At the same time a number of advanced IED's are becoming available on the market that, together with the adoption of open standards for digital communications (e.g. IEC61850), provide a convenient means of acquiring data and sharing it with other platforms.

The University's Distribution Network Automation and Protection laboratory houses a 100kW microgrid with a number of advanced controllers and measurements suitable for smart grid applications. These all produce copious amounts of data in real-time and it is now necessary to be able to display the valuable information in a meaningful manner.

This project will develop the applications required to capture data on the microgrid's substation computers, distribute measurements and alarms over a digital communications network, and visualise pertinent information within a control room SCADA environment. In this way, this project will establish an environment for the prototyping of smart grid applications from measurement to control room display.

B. Project Objectives:

Project objectives must be stated in such a way that they can be translated into achievable goals during the conduct of the project. For this reason, the stated objectives must be specific and realistic to be attained within the time provided. It is important to note that the achievements of the project work will be measured against the objectives stated here. Copies of this section will be made available to persons involved with the assessment of this project.

1. *Under the "Importance" column below, enter one of the following as appropriate: "Major", "Minor", or "Optional".*

Project Objectives (add more if required)	Importance
Understand the existing microgrid, its protection, measurement, control and control room.	Major
Understand and analyse historical data obtained from the microgrid. Archive historical data onto easily accessible file server.	Major
Research and enhance previous solutions to presenting data, including articles, papers, marketing material from various companies.	Major
Produce pseudo-code including basic functionality and HMI layout for initial GUI design.	Major
Build initial software design for GUI.	Major
Run simulated scenarios using archived data from the microgrid to test software system.	Major
Extract real-time data, using the software system to perform real-time analysis.	Major
Integrate visual monitoring of grid to allow safe off site control.	Optional
Extract and display real-time data from National Grid or PMU for comparison.	Optional

PART 2: PROJECT WORK PLAN

Identify project milestones and summarise your work plans in the table below in the order you do them. (Example: preliminary design, prototyping, simulation modelling, results validation, write-up, etc).

	Project Milestones/Work Phases	**Expected Week Time** *Enter start and end week Ex.: Week 6 to week 8*
1	Complete microgrid background research, producing summary document highlighting our findings.	Weeks 4 – 6
2	On-going analysis and archiving of historical data.	Weeks 4 – 10
3	Research into existing control room designs and techniques.	Weeks 6 – 8
4	Development of Python skills and knowledge of Siemens WinCC.	Weeks 6 – 12
5	Initial design of basic control room GUI.	Weeks 8 – 12
6	Interim report writing	Weeks 9 – 12

7	Preparation for interim presentation	Week 12 – January
8	Building and testing of initial GUI design, using historical data (including pre-determined scenarios).	January break
9	Investigation into, and extraction of, real-time data from the microgrid.	Semester 2: Weeks 1 – 6
10	Adapt GUI to incorporate real-time data.	Weeks 4 – 10
11	Test and finalise updated GUI for real-time data.	Weeks 9 – 10
12	Final report writing	Weeks 9 – 12
13	Final poster/tradeshow preparation	Weeks 11 – 12

PART 3: RESOURCE REQUIREMENT

A. Software:

List the software required for the project. This includes programming languages, application packages, CAD tools, etc.

Software *(indicate version no. if applicable)*	Software Administrator *(CS Dept, EEE Dept, Comp. Centre)*	Target Use *(teaching, research, etc)*	Platform *(PC, workstation)*	Expected Usage *(hours/week)*
Matlab R2013a	EEE Dept	Data analysis	PC	10
Microsoft Excel 2010	EEE Dept	Data extraction	PC	2
Siemens WinCC	Siemens	Creation of GUI	PC	10
Python	Codeacademy.com	Learning	PC	12

B. Hardware:

List major hardware components such as circuit boards, LSI/VLSI integrated circuits, and special purpose components.

R134 Microgrid
Desktop computers
Additional monitors
Control room servers
Webcams (optional)

C. Laboratory/Work Area:

Indicate the laboratory room(s) and/or project work area for the project.

R134 Control Room

D. Logbook:

Confirmation that each student has A4, hardback, bound logbook. YES

PART 4: TECHNICAL RISK

Management of project work requires that technical risk be assessed in advance, during initial planning and as an ongoing process. As the first stage to this process, identify any aspects of risk associated with your project proposal. Risk in this context is taken to mean any event or action (or inaction) that would jeopardise any project outcomes or significantly impede project progress. Furthermore having identified such potential risks, indicate what actions you would take to mitigate the effects of this risk. (Consult your sponsor for advice but examples of such risks include non-delivery of a key component, illness or absence from University, non-completion by student or other of key deliverable, equipment malfunction, extended learning curves- new techniques or software, etc).

	Possible Risk:	**Mitigating Action:**
1	Loss of grid data	All data backed up extensively
2	Group member illness	Re-allocate group member tasks and keep ill member up to date
3	Objectives & milestones not being met	Adjust objectives and milestones accordingly
4	Technical Incompetency	Assistance from other group members or academic staff.
5	Lack of control room access	Possible remote access, advanced scheduling.

Students will be asked to reflect upon parts 1, 2 and 4 in the interim reports and also in the final report.

PART 5: SAFETY DECLARATION

All project students must be aware of the need for safe working during the conduct of their project. The Area Safety Regulations for the Department of Electronic and Electrical Engineering, which appear in the Undergraduate Handbook, provide general guidance. Project students should consult with their Supervisor to obtain specific instructions or written Risk Assessment relating to their own project. By signing at the end of this form, the project student is declaring that:

1. 1. he/she has attended the EEE UG safety seminar.
2. 2. he/she has completed the online safety assessment quiz
3. 3. he/she has read and understood the Area Safety Regulations and will abide by these regulations during the conduct of the project, and
4. 4. he/she has consulted with the project supervisor who, if applicable, has specified the additional Risk Assessment which is as stated below:

<u>Additional Scheme of Work Specified By Project Supervisor</u> *(enter NONE if not applicable)*

Signature of Student: ……………………………………………………………………

Signature of Student: ……………………………………………………………………

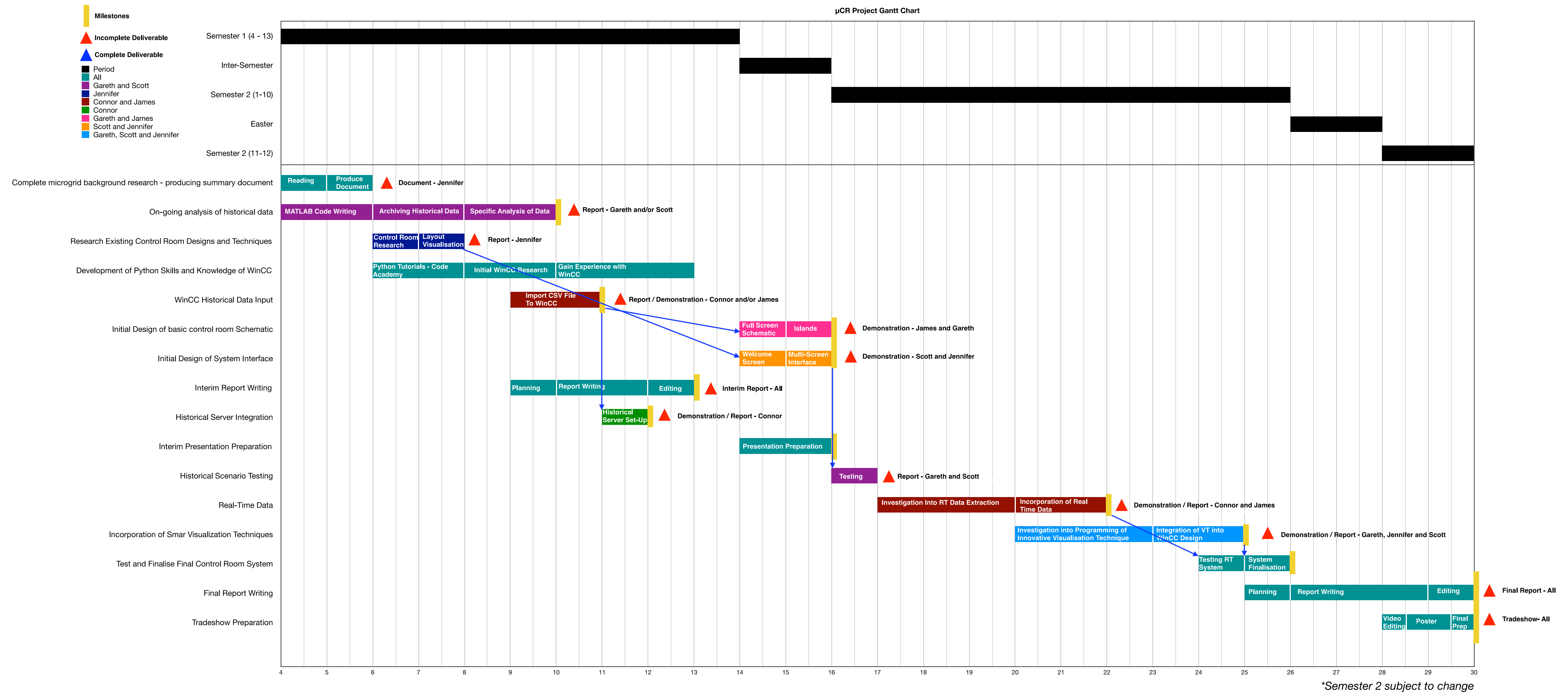
Signature of Student: ……………………………………………………………………

Signature of Student: ……………………………………………………………………

Signature of Student: ……………………………………………………………………

Signature of Student: ……………………………………………………………………

Date ……………………………………….


µCR Project Gantt Chart
Milestones
Incomplete Deliverable
Complete Deliverable
Period
All
Gareth and Scott
Jennifer
Connor and James
Connor
Gareth and James
Scott and Jennifer
Gareth, Scott and Jennifer
Semester 1 (4 - 13)
Inter-Semester
Semester 2 (1-10)
Easter
Semester 2 (11-12)
Complete microgrid background research - producing summary document
On-going analysis of historical data
Research Existing Control Room Designs and Techniques
Development of Python Skills and Knowledge of WinCC
WinCC Historical Data Input
Initial Design of basic control room Schematic
Initial Design of System Interface
Interim Report Writing
Historical Server Integration
Interim Presentation Preparation
Historical Scenario Testing
Real-Time Data
Incorporation of Smar Visualization Techniques
Test and Finalise Final Control Room System
Final Report Writing
Tradeshow Preparation
Reading
Produce Document
Document - Jennifer
MATLAB Code Writing
Archiving Historical Data
Specific Analysis of Data
Report - Gareth and/or Scott
Control Room Research
Layout Visualisation
Report - Jennifer
Python Tutorials - Code Academy
Initial WinCC Research
Gain Experience with WinCC
Import CSV File To WinCC
Report / Demonstration - Connor and/or James
Full Screen Schematic
Islands
Demonstration - James and Gareth
Welcome Screen
Multi-Screen Interface
Demonstration - Scott and Jennifer
Planning
Report Writing
Editing
Interim Report - All
Historical Server Set-Up
Demonstration / Report - Connor
Presentation Preparation
Testing
Report - Gareth and Scott
Investigation Into RT Data Extraction
Incorporation of Real Time Data
Demonstration / Report - Connor and James
Investigation into Programming of Innovative Visualisation Technique
Integration of VT into WinCC Design
Demonstration / Report - Gareth, Jennifer and Scott
Testing RT System
System Finalisation
Planning
Report Writing
Editing
Final Report - All
Video Editing
Poster
Final Prep
Tradeshow- All
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
*Semester 2 subject to change