MURDOCH
UNIVERSITY
PERTH, WESTERN AUSTRALIA

School of Engineering & Information Technology

ENG460 – Final Report

Commissioning and Implementing a PROFIBUS network in the Universal Water System

Benjamin Daniel Cole

1 Abstract

The Universal Water System (UWS) was built for instrumentation and control engineering students to design, implement and test different control schemes. The system has been primarily developed and designed by fourth year undergraduate and master's students with the help of on-site technicians and electricians for installation of high voltage wiring, hardware equipment and IT related tasks. As the UWS is a learning tool that provides hands on experience with an industrial grade environment and equipment, improvement and maintenance of its functionality is a vital part of the on-going thesis projects.

The main objectives of this work are split into three major segments. The first was commissioning the UWS, which consisted of updating the system's software, replacing faulty equipment and ensuring appropriate functionality of the plant's hardware. Two PROFIBUS Decentralised Peripherals (DP) flowmeters had been purchased to replace two faulty positive displacement flowmeters in the system. Hence the second objective was implement a PROFIBUS DP communication network for the new devices. The last objective was to design, implement and test more advanced control schemes through Open Platform Communication for the newly upgraded plant.

With the project now complete, the UWS is operational with a fully functioning PROFIBUS DP communication network. The server computer's operating system has been upgraded, while the Compact RIO's firmware and the programming software has been updated to the latest version. Faulty equipment has been replaced and commissioned. Namely, two replacement flowmeters and an electric flow valve. A PROFIBUS DP network has been implemented to communicate with the two replacement flowmeters. An unexpected technical difficultly led to 5 variable speed drives being added to the PROFIBUS DP network. Additionally, the compact RIO's code has been redesigned to improve efficiency, provide cyber-security, and to reduce the complexity of the client program.

Due to unforeseen circumstances and time constraints the time taken to commission the plant and implement PROFIBUS was far greater than expected; two of the three project objectives were completed, pushing the advanced control schemes and Open Platform Communication to future work. Overall, the main accomplishment of this thesis besides the project objectives, is that the system has been updated, refitted and ready for operation for the next thesis student; so they do not run into the tedious and painful issues found during this project.

2 Acknowledgements

YED graph editor for providing a free desktop application to quickly and effectively generate sick diagrams (http://www.yworks.com/en/products/yfiles/yed/). John Boulton for being a handy man, smashing out the instrument fitting/installation and high voltage wiring. Will Stirling with being an ub3r l337 h4ck3r and for all his help with IT. Slick G for being Slick G. Most importantly, Dr Linh Vu for her tireless effort, classic sense of humour and incredible patience with my terrible organisational skills. I wouldn't have been able to complete the project let alone get this far in engineering without her constant help and support.

3 Terms and Abbreviations

- UWS – Universal Water System
- NI - National Instruments
- LV - LabVIEW
- cRIO - CompactRIO
- PAC - Programmable Automation Controllers
- CPU – Central Processing Unit
- SISO - Single Input Single Output
- MIMO - Multiple Input Multiple Output
- PROFIBUS - (PROcess FIeld BUS)
- DP - Decentralised Peripherals
- API - Application Programming Interface
- I/O - Input/Output
- SPI – Serial Peripheral Interface
- VAC – Voltage Alternating Current
- VDC – Voltage Direct Current
- PDW – Process Data Word
- SBus – System Bus
- VSD – Variable Speed Drive
- DSC – Data logging & Supervisory Control
- PID – Proportional Integral Derivative
- OS – operating system
- OPC – Open Platform Communication
- FPGA – Field Programmable Gate Array
- GMC – Generic Model Controller
- RPM – Revolutions per minute
- MAX – Measurement and Automation Explorer
- VISA – Virtual Instrument Software Architecture

Table of Contents

List of figures

4 Introduction

The UWS is a process control plant located at Murdoch University's engineering building at the South Street campus. Its life began at the Rockingham campus in 2006, with many students and groups making contributions to its design and development [1]. A considerable amount of time was spent on the design, specification and purchasing of equipment before construction started with the main frame, tanks, pumps and plumbing [2]. The electrical & pneumatic systems along with the control and power panels were added after the relocation to the South Street campus in 2008. The system was finally moved to the new Murdoch engineering building in November 2010 [3]. The plant was designed as a large scale teaching tool for students studying instrumentation and control engineering. It is intended as a stepping stone between the second year instrumentation and control laboratory and the third/fourth year pilot plant.

The UWS was deliberately designed to have many different types of instruments, which vary in manufacturer, type and model, to expose students to a wide range of industrial equipment. The system is equipped with 9 pumps and 4 control valves, to control the level of 5 tanks, and 7 flows. There are three main sections to the plant, separated by hand valves, outlined in Figure 1. The hand valves can be configured to control the plant as an integrated whole, three individual systems or a two-in-one combination. Solenoid and gate valves throughout each section can be manipulated to create different process control scenarios. For example, by opening gate valve seven (GV07) to connect tanks one and two an interacting system can be created. This produces a second-order system with different dynamics compared to a single tank system [4].

Figure 2 [3] shows the actual layout of the UWS before it was moved to the engineering building at the South Street campus, however the layout remains the same. Figure 1 summarises the flow of water between the major sections. Within area A, pumps 1 and 2 drive water from tank 6 (supply) to tanks 1 and 3. Pump 9 recycles water from tank 3 to tank 1. From there, water can either flow back to supply through GV05 or flow to area B via HV10. Within area B, supply water can be drawn from either area A or from tank 06 through HV11. Pumps 5, 6 and 7 pump water to tank 05, which then flows through HV08 or SV08 to area C or back to tank 6 respectively. Area C is similar to area A, where pumps 3 and 4 pump water to tanks 2 and 4. Pump 8 recycles water from tank 4 to tank 2. From there, the water flows back to tank 06. GV07 and GV08 can be opened to connect tanks 1, 2 and 3, 4 respectively to creating interacting systems. Figure 3 shows the full piping and instrumentation diagram of the UWS.

Overall, the UWS is a dynamic teaching tool for Murdoch engineering students who major in instrumentation & control and industrial computer systems. From an instrumentation and control perspective, the software interface has been designed to have the same functionality as an industrial setting, while still including the ability to investigate, design, implement and test different control scenarios. At a fundamental level, the UWS is related to industrial computer systems engineering: the communication principles, software architecture and hardware signal wiring.

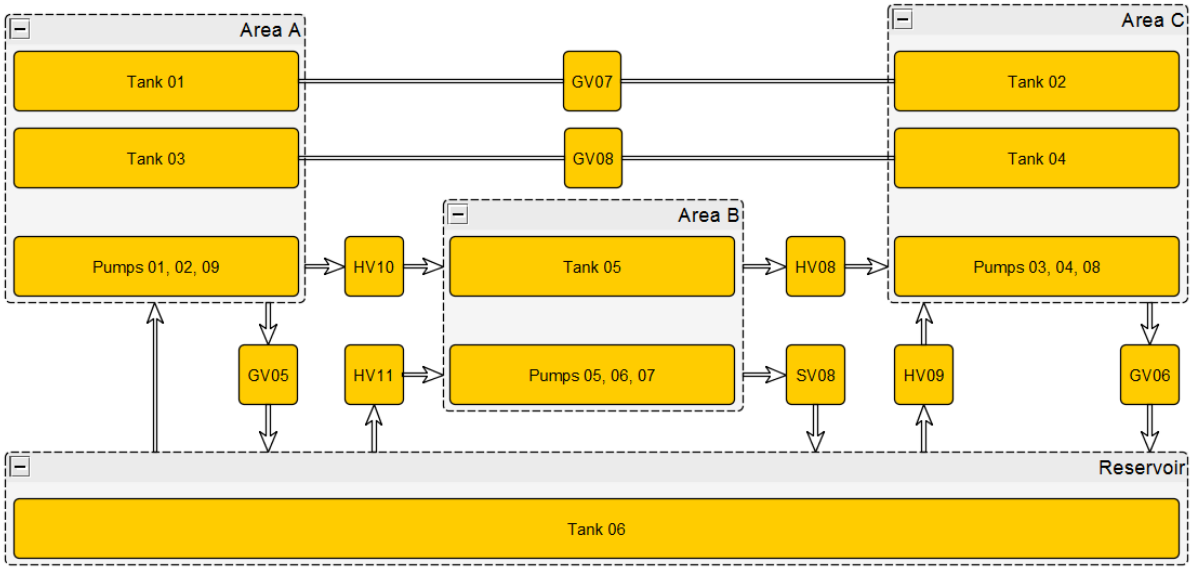

Figure 1- Flow sheet of UWS major sections, arrows indicating water flow direction.

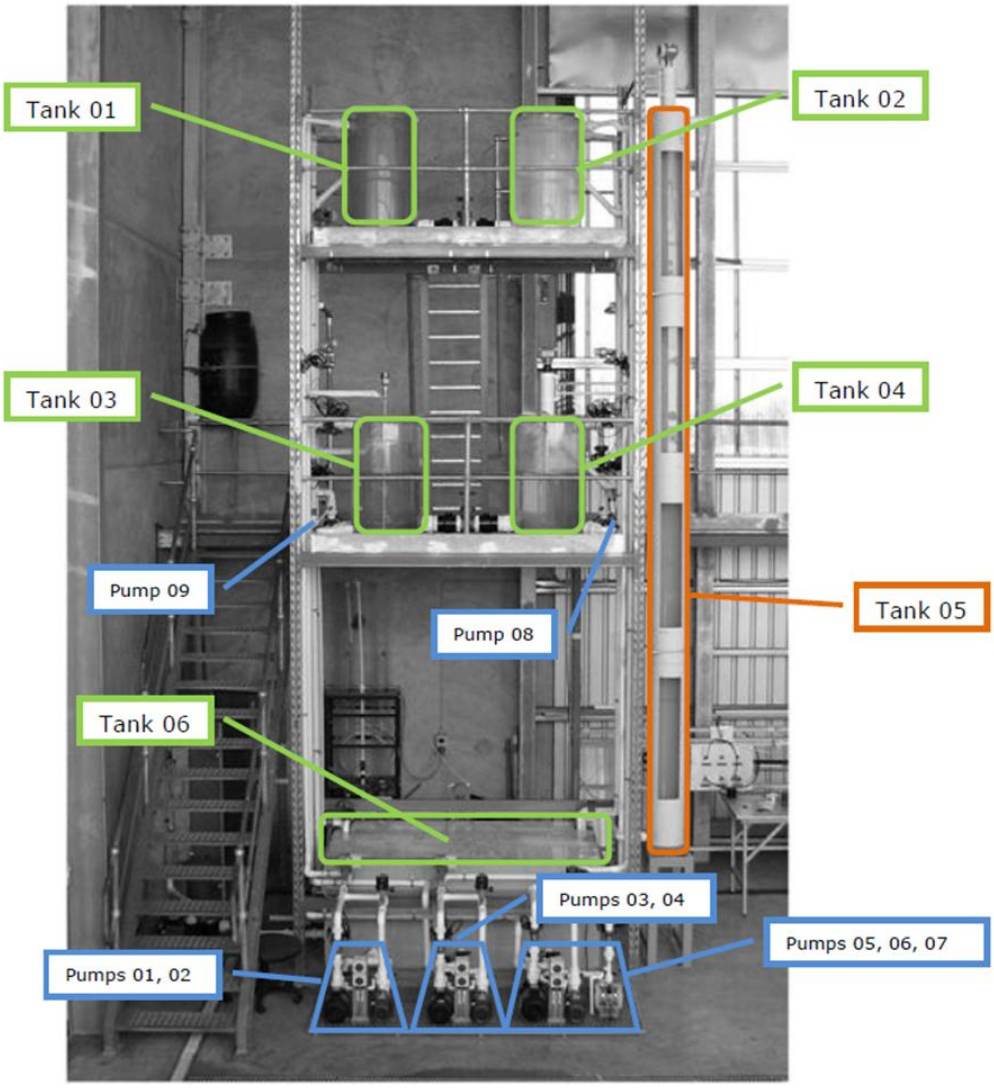

Figure 2 - Overview of UWS [3]

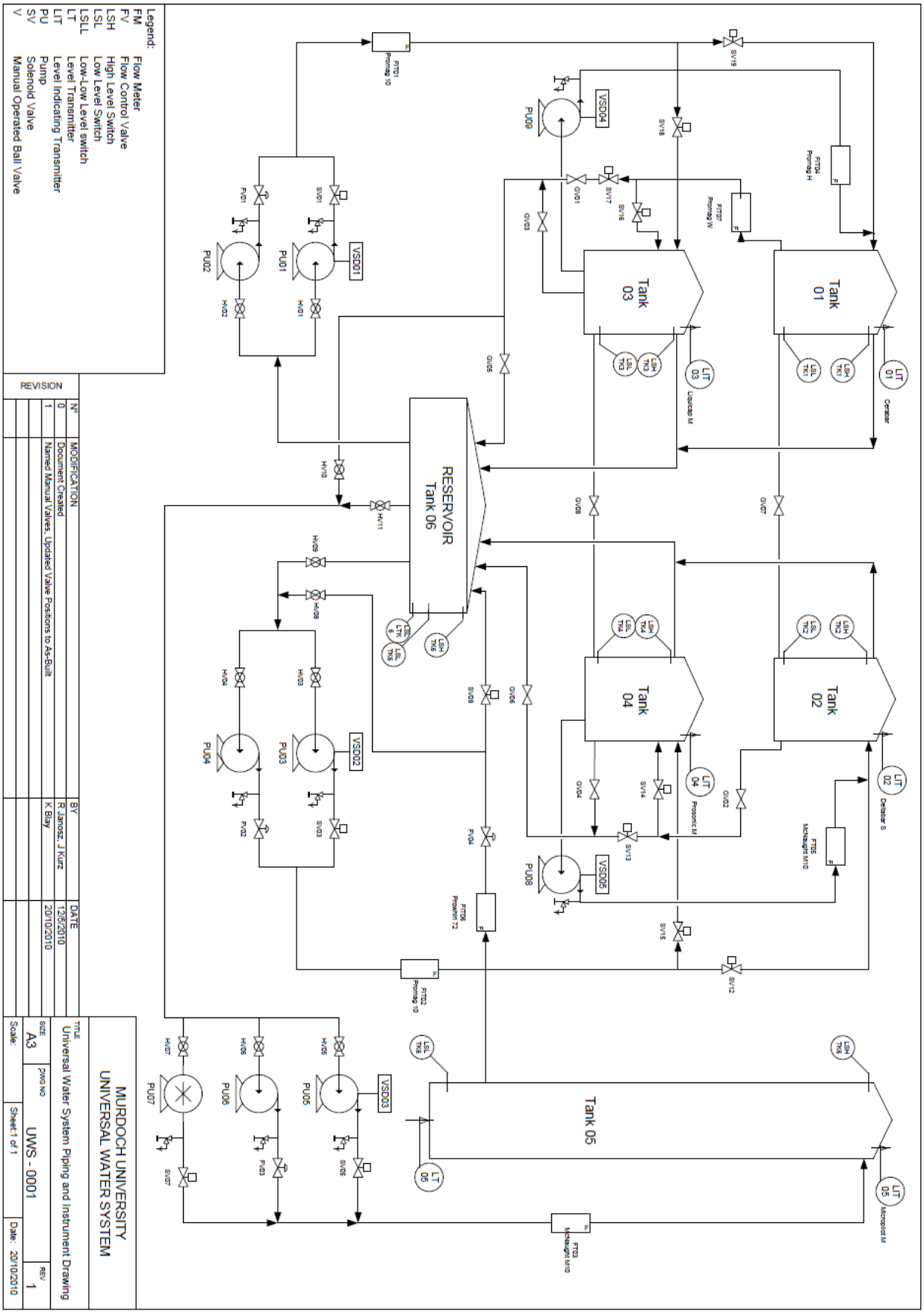

Figure 3 - UWS piping and instrumentation diagram [3]

4.1 General Overview & Previous Work

The section provides a general overview of the operation of the UWS and previous work implemented by students. Refer to previous reports for more detailed description of the instrumentation in the UWS and its operation [1] [2] [5] [3] [6].

The UWS uses a National Instruments (NI) Process Automation Controller (PAC) to communicate with the equipment. The NI PAC used for the UWS is a compactRIO (cRIO): “a reconfigurable embedded control and acquisition system, which is comprised of input/output (I/O) modules, a reconfigurable field-programmable gate array (FPGA) chassis and an embedded controller” [7] [8]. FPGAs are reprogrammable silicon chips that, when programmed, rewires the chip itself for implementation rather than run a software application like a standard CPU [9]. The software used to control the UWS utilizes a programming language called LabVIEW (LV) developed by NI. LV uses “graphical programming syntax that makes it simple to visualise, create, and code engineering systems” [10].

The cRIO embedded system transmits and receives data to/from the instruments; processing the information and monitoring the state of the system. If an undesirable condition is to occur with respect to a device, which would damage equipment or risk the safety of personnel, the cRIO will “trip” and interrupt its operation. This process called “interlocking” is a fundamental requirement of the cRIO. After the processing of data, the cRIO relays the information to a server computer that in turn hosts the data to the client machines. These computers, situated in a control room, provide a graphical user interface for alarming, trending, logging, and automatic control capabilities. Once confident in its operation, operators can create custom and advanced control schemes to operate the plant.

The multilevel control system of the plant was designed and implemented by Kane Blay, described in his thesis [3]. To summarise, he did a substantial amount of work developing the code for the entire system, creating a multilevel topology, where client machines communicate via a server computer to control the UWS (figure 4). The original design relating to variable security, the interlock system and the client program are summarised below.

Variables that are written by the cRIO, such as the run request to a pump, are hosted on the device; while variables written by client machines are hosted on the UWS server. The NI Data-logging & Supervisory Control (DSC) module [11] was used to prevent unauthorized access to the variables of the system. However the module does not allow for variable security on cRIO systems. Kane worked around this by exploiting a bug in the LV software to force security settings on the device, allowing only the server computer communication. While the sever computer only allowed access from the cRIO and client machines in the pilot plant.

The client program designed by Kane is quite extensive consisting of multiple areas with alarming, trending and built in PID control capabilities. One window is used to control and monitor the plant, extract logged data and view a trend; split into multiple tabs. The client template is Abnormal Situation Management compliant [12], meaning grey colours are used during normal operation and bright colours are used during an alarm event. The manipulated variables of the plant can be controlled manually or automatically (PID) paired with any process variable in the plant.

The interlock system works by only allowing equipment to be controlled if the respective interlock is clear. All pumps and only a few solenoid valves have interlocks. Solenoid valves are simple and only open if the tank they are discharging into is not full. Pumps are a little more elaborate, where they will only start if the stop button is false, there are no interlocks, and there is a rising edge on the start button. The interlocks are represented by a single binary unsigned integer where each bit reflects a different interlock status. For example 1 would relate to the discharge tank being full, while 101 would represent the discharge tank being full AND no flow was detected, etc.

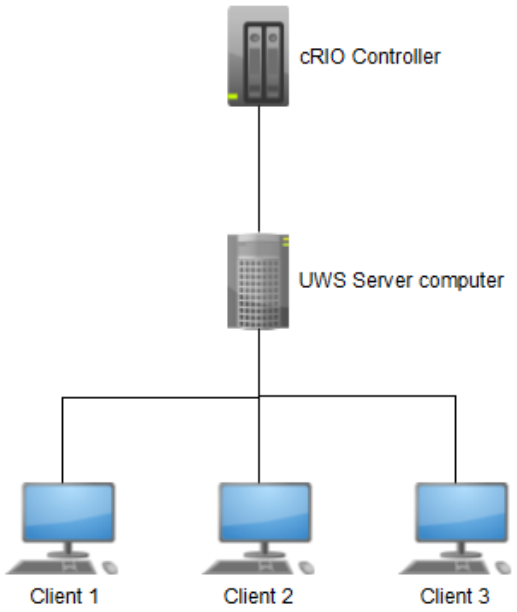

Figure 4- UWS topology

.

After Kane, a master's student, Jagadeesh Ganesan [13] implemented a standard maintenance program that ran on the client template along with more advanced control schemes. Some of which, the generic model controllers (GMC) [4] for example, are dubious and lacked a description of the calculations and a review of the performance. Following on, Arash Tokhmechi [6] did a huge amount of work calibrating all of the instruments and developing a demonstration program. During the course of building the UWS, filings and other debris accumulated in the supply tank which caused problems for the MacNaught [14] flowmeters ^. The internal mechanisms were removed to avoid blocking the flow, and hence the instruments needed to be replaced.

^ Since they rely on a gear mechanism and positive displacement, they would constantly fill with the debris and cause a blockage.

4.2 UWS initial state

The UWS is a large project and has taken many years to complete with contribution from numerous students. Keeping the system operational and updated is of utmost importance. Hence, maintenance duties and improvement to the functionality of the UWS are a primary aspect of the on-going thesis projects. Initially, the UWS was using outdated LV software and firmware from 2010, an unsupported operating system (OS) Windows XP [15], contained two broken MacNaught flowmeters and was unresponsive to the control program. Hardware needed to be replaced, software needed to be updated and an investigation into the underlying issue was needed to reactivate the plant.

As mentioned previously, there are two broken flowmeters in the plant that need replacing. Two PROMAG flowmeters [16] that use PROFIBUS (PB) [17] communication, a type of fieldbus, to transfer data had been purchased as replacement. A cRIO PB module, developed by a propriety company COMSOFT [18], had been purchased along with the replacement flowmeters to communicate with the devices. The different communication method is desirable not only because it provides variety in the system but also because of its wide use in many different industrial environments.

Although the system is highly configurable and capable of different control strategies, few students have successfully implemented and tested advanced control schemes. A past attempt has been made at GMC by Jagadeesh, however the controller had not been tested nor performance examined. Additionally, controllers have yet to be implemented through third party software via open platform communication (OPC)*. Controlling the plant through software such as EXCEL or MATLAB is desirable for students who are unfamiliar with LV. Furthermore, matrix based control strategies, such as Dynamic Matrix Control (DMC) [4], are better suited for such programming environments.

* The framework of which has been already implemented by Kane with the DSC module.

4.3 Project Objectives

This section covers the project objectives. They are split into three main categories, which relate to each main phase of the project. Commissioning the UWS was the primary goal, and once completed, could lead into the next two phases: Implementing PB communication and Advanced control schemes. The objectives are described with numbered lists, where applicable, for readability.

4.3.1 Commissioning the UWS

1) Get the UWS operating and functional.
    a) Upgrade the OS and server computer to Windows 7.
        i) Update the LV software to the latest versions, from 2010 to 2014.
        ii) Update the cRIO firmware to latest version
    b) Software developed by previous students can be reinstalled and system can be tested for any additional faulty equipment.
    c) Replacement equipment can be installed with the help of university technicians, specifically 2 PB flowmeters [16].
        i) Signal wiring can be updated.

4.3.2 PB Implementation

The replacement flowmeters communicate via PB compared to the original "pulse count" from the MacNaughts [14]. Thus the second objective consisted of researching, developing and implementing PB communications network in the UWS. Since PB is quite an elaborate technology with lots of components, it could be researched extensively in parallel to commissioning the UWS. Although it was a significant task, the cRIO PB module came with configuration tools, example programs, and LV application programmatic interfaces (APIs) to aid in the implementation of the network [19].

1) Implementing a PB Network
    a) Investigation:
        i) How the cRIO communicates with the PB instruments and vice versa.
        ii) How to configure a PB network.
        iii) Programing PB communication in LV and changes to cRIO code.
    b) Implementation:
        i) Slowly understand and implement PB communication itself.
            (1) Communicate with devices through configuration software.
            (2) Communicate with devices through LV APIs.
            (3) Integrate PB code into existing program.

4.3.3 Open Platform Communication and Advanced Control Schemes

Once the UWS has been commissioned and PB network has been successfully implemented, the UWS can be used to develop and test control schemes. Although the framework for OPC communication has been implemented by Kane, no work has been done into actually controlling the plant through third party software. OPC provides a standardised way of communication between third party software. Creating basic templates for using Excel or MATLAB to control the UWS would be a huge benefit to future students.

4.4 Scope of Report

The remainder of the report is split into chapters relating to major aspects of the project. Overviews of PB communication and newly introduced technologies are provided to the reader in Section 5. This relates to necessary background information for the reader to understand the following chapters. The next three sections describe the major work undertaken during the project: Section 6 Commissioning the UWS relates to the first objective, Section 7 PB Implementation relates to the second objective and Section 8 Software Redesign relates to an additional objective that wasn't previously foreseen. The Discussion section, Section 9, describes the major issues encountered which led to not all of the project objectives being satisfied. Section 10 describes the future work recommended for future students. Finally is the conclusion which summarises the report.

5 System Overview

This section contains all necessary information on the technologies introduced to the UWS. It covers an overview of the PB, newly introduced hardware and software packages. The PB overview does not go into a huge amount of detail but rather serves as an introduction to readers who are new to the concept. Similarly, the hardware overview only discusses newly introduced features rather than repeat the work discussed in previous reports.

5.1 PROFIBUS

This section shows some fundamental knowledge on PB communication and the PB network which is used/implemented in the UWS. PB is a standard fieldbus communications protocol for the automation industry, openly published in IEC61158 [20]. Due to the standardised nature of the protocol, PB is currently the most popular fieldbus with an estimated 40 million nodes installed across the globe. PB provides fast data exchange, improved diagnostics capabilities and a simpler plant design compared to conventional methods such as 4-20mA signalling. “PROFIBUS was created in 1986 by the German government in cooperation with several manufacturers of automation equipment. It is a messaging format specifically designed for high-speed serial I/O in factory and building automation applications.” [21]

There are two major variants of PB: Decentralised Peripherals (DP) and Process Automation (PA). PB DP, simply put, is the communication between distributed I/O devices and the embedded system. Where the I/O is decentralised from the main PAC. In contrast, the cRIO’s analog I/O channels, for example, are connected directly to the device, hence they are centralised I/O [21]. PB PA is built upon PB DP, it was designed specifically for hazardous environments and carries the power and data over the same wire [22]. The UWS is by no means a hazardous area in danger of explosions, hence PB DP was the appropriate choice for communication.

PB is defined in internationally recognised standards. The architecture of the protocol is based upon the Open System Interconnection (OSI) reference model. The OSI model is an international standard, ISO7498, which precisely defines seven transmission layers for telecommunication regardless of the underlying technology and internal structure [23]. PB DP defines layers 1, 2 and 7, the physical layer, data link layer and application layer. Layers 3 through to 6 are not used [24]. For the application later, different levels of the PB DP protocol are defined. DP-V0 is the most basic, for cyclic (continuous) data exchange. DP-V1 and DP-V2 are more advanced functions that are not implemented in the UWS. The second layer, called the security layer or the fieldbus data link works via the master slave method. Where the PACs are the masters and the actuators and sensors are the slaves. The slaves will only respond once the master requests data. RS-485 is the most common technology used by PB for the physical layer, and is used in the UWS PB network. Data is transmitted differentially through a shielded two core cable, typically with a violet sheath, via a bus topology [25].

5.1.1 PROFIBUS DP Communication

As mentioned previously, DP is used to communicate with decentralised instruments through a centralised controller. The functionality of PB DP has been extended since its creation; the foundation (DPV0) provides cyclic data exchange and diagnostic reporting between devices. The next level, (DPV1) allows for acyclic data exchange between devices [26]. For example, cyclic (repeated) data exchange in the UWS would be the flow rate reading from the flow meter. Acyclic data exchange in the UWS would be used to configure the flowmeter's settings [17].

PB DP is a network of devices connected to a "bus" that are considered either "masters" or "slaves". Typically, a sensor or actuator is a slave while the master is a typically a PAC. The "bus" is a bi-directional network where messages/data are received by all devices. Hence, a unique address is assigned to each device to ensure messages are sent to the appropriate destination. The master initiates the request for data and the slave immediately responds, meaning that only one device has control of the bus at any given time. The address of a device is most commonly set through a number of hardware switches located on the unit. Termination resistors are used on each end of the bus, to terminate signals sent by devices. If the bus is not correctly terminated, data can get reflected back down the bus and disrupt later signals [17]. A diagram of the PB topology is shown in Figure 5

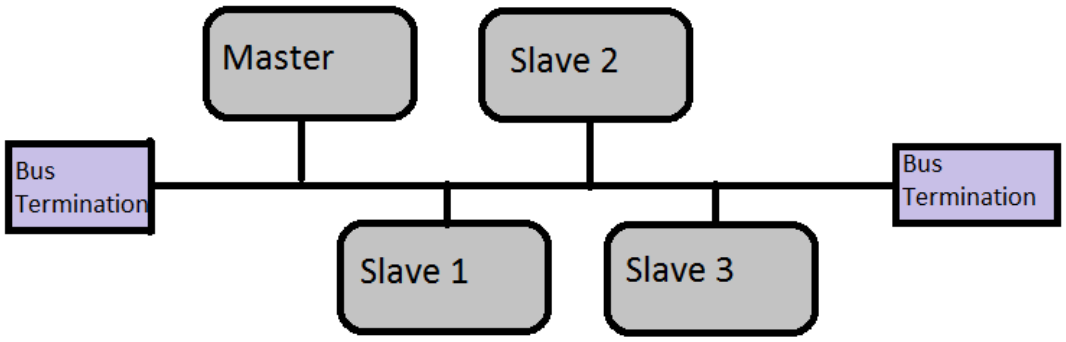

Figure 5 - Bus topology

5.1.1.1 Configuration, Initialisation & Operation

The following quotation from NI regarding PB initialisation conveys the concept well: "You specify which slave devices the master should find on the bus as well as which information should be transferred from the master to each slave during a start-up phase. All of the information that the master must know to start up the bus comes from a configuration database file that is generated by a PB configuration tool. Each vendor of PB master devices offers a configuration tool for generating the database file for their masters" [17]. When a PB slave device is produced by a company, a device description file or a "GSD file" must also be made [27]. These files are created for the master's configuration software to define functionality of the slave. The master's configuration tool is also used to:

- Specify PB addresses
- Specify I/O cyclic data
- Specify system baud rate
- Specify start-up parameters for different modes of operation
- Create database file for master

The PB Implementation section goes into more detail about configuring a PB network, specifically the one implemented on the UWS. Once the master has the appropriate configuration database file it is able to start-up each of the slave instruments at their defined addresses. The master attempts to initialise the devices to determine whether the physical field instrument matches the functionality described in the database file [23]. If successful the slave begins data exchange or operational mode. The database file is stored in retentive memory, meaning if the master loses power the database file will remain intact and ready to initialise and operate the bus once power has been restored. Similarly, if a slave loses power or is removed from the bus, the master can reinitialise the device (if consistent with the configuration file) once it returns [17].

Once the bus has been initialised and the slaves have been set into operational mode, the master will exchange cyclic I/O data with the slaves. The master sends output data to an addressed slave, and the slave responds immediately with its input data [24]. "Cyclic" simply means the process is repeated periodically, and since the bus operates at very high speeds (1.5Mbits/s being the average) the exchange of data typically occurs many times per scan cycle of the control logic [17].

Along with the cyclic I/O data exchange, the PB allows for broad diagnostic reporting capabilities that vendors can customise for their products. The PB slave will indicate a diagnostic condition and the master will fetch the data on the next scan cycle. A device's diagnostic data is split into four different categories: standard, device related, module related and channel related diagnostics. Standard diagnostics, which are typically related to start-up issues, are common to all slaves and must be reported to the master. The most common diagnostic reports from slaves are configuration, parameterization and address faults. A configuration fault relates to a discrepancy between the I/O data in the configuration file and the I/O data that the slave is expecting. A parameterization fault means that the slave device specified at the address in the configuration file is different from the one found on the bus. Lastly, an address fault simply means no slave was found at the specified address. The remaining diagnostic categories are not used in this project [17].

5.1.2 Overview of UWS PB Network

The cRIO is the master of the PB network and handles cyclic I/O data exchange and diagnostic reporting to three slave devices: two flowmeters and one gateway (UOH11B). The COMSOFT Configurator III program [19] was used to define the addresses, network baud rate and cyclic I/O data of the slaves. The flowmeters have been configured to send an analog reading of the flow in litres/minute [16] while the gateway has been set up to control the five VSDs via the PB network.

The VSDs are digitally controlled via the System-Bus (SBus). “The SBus is a CAN bus according to the CAN specification 2.0, parts A and B” [28]. Process data telegrams are used to enable the VSDs, monitor their status (ie ready for operation and output enabled) and control their speed. The VSD gateway converts PB signals to/from the cRIO to SBus process data telegrams for the VSDs. Figure 6 depicts the topology of the PB network. Since PB was a major aspect of the project, the following chapter will be dedicated to implementation of the communication network. Table 1 provides a summary of the cyclic IO data of the PB master.

As mentioned previously, the VSDs are controlled via process data telegrams, or process data words (PDWs). These are 2 bytes or 16 bits in size. To summarise the operation of the VSDs, they each employ three PDWs as inputs and three PDWs outputs. The VSD gateway gets one entire block of data from the cRIO for all the VSDs, which the gateway then distributes to each drive. There are five VSDs, each with 3 PDWs in and out, meaning the cRIO cyclically exchanges 30 bytes of data to/from the gateway. The VSDs’ PDWs are configured and defined as follows [28]:

- Output PDWs to VSDs:
  - PDW1: VSD Control Word
    - Binary 10 = STOP
    - Binary 11 = ENABLE
  - PDW2: VSD Speed; revolutions per minute (RPM)
    - Conversion: 5 * RPM = PumpSpeed% * 5 * 15 = DataOut
  - PDW3: No Function
- Input PDWs from VSDs
  - PDW1: VSD Status Word
    - Most Significant Byte: Error Number/Unit State (unused)
    - Least Significant Byte: Basic Status Block (fixed definition)
      - Bit0: Output enabled
      - Bit1: Invertor Ready
      - Bit2: PO data enabled (unused)
      - Bit5: Fault/Warning (unused, changes definition of most significant byte)
      - Bits 3, 4, 6 & 7: Reserved (unused)
  - PDW2: No Function
  - PDW3: No Function

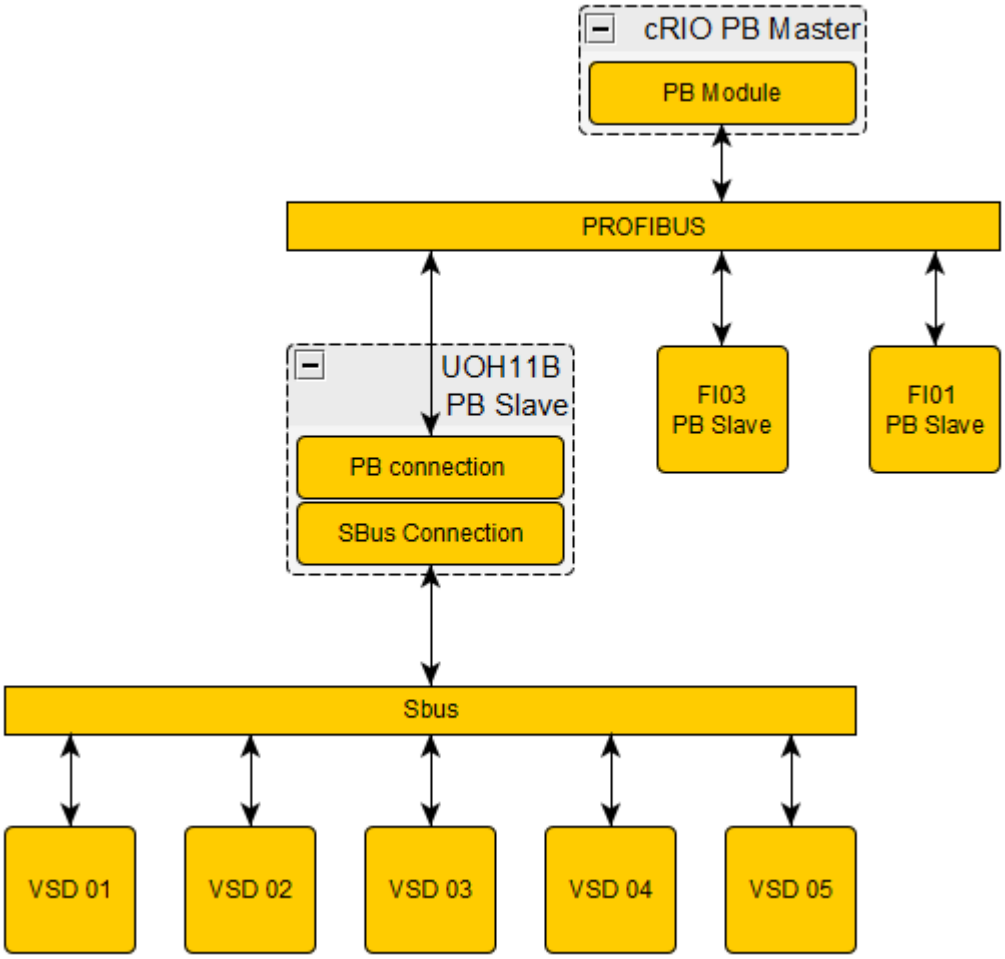

Figure 6 - UWS PB Topology

Table 1 - PB cyclic IO summary

Slave	PB Address	Input data from slaves			Output data to slaves		
		Description	# of bytes	Byte index	Description	# of bytes	Byte index
FI01	1	Volumetric flowrate	4	0 – 3	N/A		
		Measurement Status	1	4			
FI03	3	Volumetric flowrate	4	0 - 3			
		Measurement Status	1	4			
VSD Gateway	5	VSD01 Status word	2	0 – 1	VSD01 Control word	2	0 – 1
		No function	2	2 - 3	VSD01 Speed RPM	2	2 – 3
			2	4 - 5	No function	2	4 – 5
		VSD02 Status word	2	6 – 7	VSD02 Control word	2	6 – 7
		No function	2	8 – 9	VSD02 Speed RPM	2	8 – 9
			2	10 – 11	No function	2	10 – 11
		VSD03 Status word	2	12 – 13	VSD03 Control word	2	12 – 13
		No function	2	14 – 15	VSD03 Speed RPM	2	14 – 15
			2	16 – 17	No function	2	16 – 17
		VSD04 Status word	2	18 – 19	VSD04 Control word	2	18 – 19
		No function	2	20 – 21	VSD04 Speed RPM	2	20 – 21
			2	22 – 23	No function	2	22 – 23
		VSD05 Status word	2	24 – 25	VSD05 Control word	2	24 – 25
		No function	2	26 – 27	VSD05 Speed RPM	2	26 – 27
			2	28 - 29	No function	2	28 - 29

5.2 Hardware Overview

As mentioned previously, the UWS consists of many different types of pumps, flowmeters and sensors, for students to experience a wide variety of industrial instruments. This section will present the equipment that has been installed for this project. A detailed description of the hardware in the UWS can be found in Tokhmechi [6].

5.2.1 cRIO PB Module

The cRIO PB master/slave module is an interface that connects PB DP networks to NI-PACs. The device is made by a proprietary company (Comsoft [18]) who provides drivers, graphical configuration software for the PB network and LV API SubVIs. LV example programs are also provided to help users to understand the module's functionality. The module is capable of adding the cRIO as a master (DPV0 & DPV1) or a slave node in the PB network, with the baud rates from 9.6 Kbit/s to 12 Mbit/s [29]. Figure 7 shows the installed cRIO PB module for the UWS network.

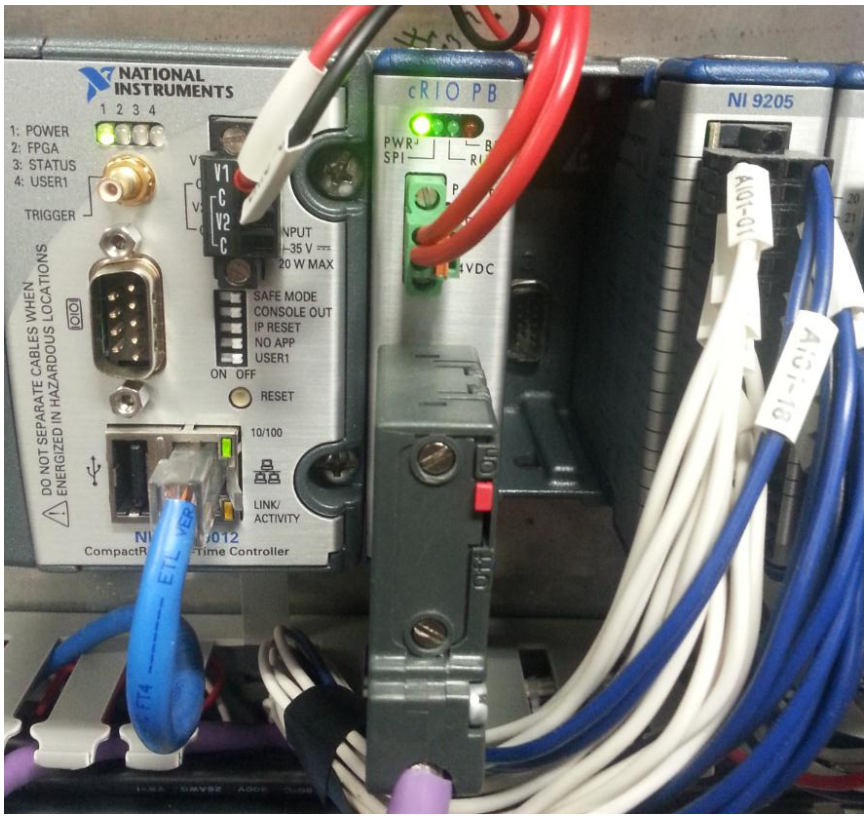

Figure 7 - cRIO PB Module installed in chassis

The module was purchased to communicate with the flowmeters replacing the MacNaught positive displacement flow instruments. The cRIO PB module replaced the NI 9401 Bidirectional digital I/O module that communicated with the faulty MacNaught flowmeters. The PB module is supported only in a cRIO reconfigurable chassis: NI cRIO-911x or NI single board RIO devices. As can be seen in Figure 7 the module has 24VDC power terminals, 9-pin serial port for the PB DP network and four indication lights. This module requires an additional 24V, separate from the cRIO power supply, to power the device. It consumes 2.5W of power and must be placed in slot 1 of the chassis with slot 2 empty to appropriately dissipate heat [18]. A summary of the indication lights is described below in Table 2. The cRIO PB master/slave module is capable of DPV1 acyclic communication, but that was beyond the scope of this project.

Table 2 - cRIO PB Module LED description [18]

LED	State	Function
Power - PWR (Green)	On/Off	Indication of power supply connection
Serial Peripheral Interface - SPI (Green)	On/Off	Indication of data being transmitted.
RUN (Green)	As DP Master	
	On/Off	Scanning Activated/not activated
	As DP Slave	
	On/Off	DP Slave initialised/not active
	Fast Flash	DP Slave (AutoSlaveMode) waits for a DP Master to communicate
	Slow Flash	DP Slave (AutoSlaveMode) baud rate detection active
Bus Fault - BF (Red)	As DP master	
	On/Off	PB OK/PB Bus Fail (Configuration error or non responding DP slave)
	As DP Slave	
	On/Off	Data exchange OK/No Data exchange with master (Configuration error or DP master not active)
	Fast Flash	DP Slave (AutoSlaveMode) waits for a DP Master to communicate
	Slow Flash	DP Slave (AutoSlaveMode) baud rate detection active

5.2.2 Proline Promag 53W Flowmeters – Retake photo

There are two Promag 53W Flow meters [16] in the UWS, FI01 & FI03, which were purchased to replace the faulty MacNaught flowmeters. They are manufactured by Endress+Hauser for PB DP networks. Figure 8 show one of the PB flowmeters. These flowmeters exploit Faraday's law of magnetic induction [30] to perform measurement, outlined in the following quotation extracted the technical information manual.

Figure 8 - PB Flowmeter

"Following Faraday's law of magnetic induction, a voltage is induced in a conductor moving through a magnetic field. In the electromagnetic measuring principle, the flowing medium is the moving conductor. The voltage induced is proportional to the flow velocity and is supplied to the amplifier by means of two measuring electrodes. The flow volume is calculated by means of the pipe cross-sectional area. The DC magnetic field is created through a switched direct current of alternating polarity." [16]

These PB flowmeters cause no restrictions within the pipe, compared to the previous gear mechanism from the positive displacement MacNaught flowmeters, hence there is no pressure drop across the device. The local display of the device shows the measured flowrate and the status of the PB communication. It is also possible to set parameters through the display (or through DPV1 communication) however that was beyond the scope of the project.

Lastly, the instruments are powered with 240VAC and circuit breakers are installed in the wiring cabinet for device protection in case that a fault occurs. Hence, it is imperative that the instruments are turned off if the internals of the instrument need to be accessed! A brief description of the terminals is shown in Table 3, and a photo of the terminals in Figure 9.

Table 3 - PB flowmeter terminal summary

Terminal Number	Function
1 (L1)	Power supply, 85 – 260VAC
2 (N)	Power supply, neutral
26 (RxD/TxD-N)	PB DP (B)
27 (RxD/TxD-P)	PB DP (A)

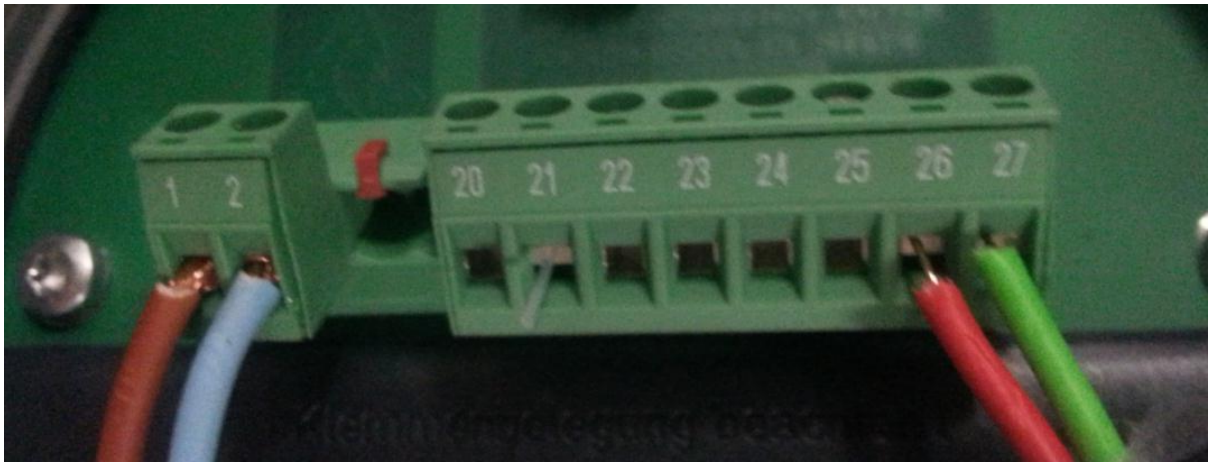

Figure 9 - PB flowmeter terminals

5.2.3 Fieldbus Gateway DFP21B/UOH11B & MOVITRAC 07A Frequency Inverters

The MOVITRAC 07A frequency inverters (or simply VSD) are not new to the existing system, but the method of communication has changed. Originally, the VSDs communicated via 3 sets of wires: a 24V digital output to enable the inverter, a 24V digital input to read the status of the inverter and a 4-20mA signal to set the speed of the VSD. In this project, the VSDs are to be completely controlled via the PB network. The shift to a PB network allows for a higher level of functionality with minimal wiring.

The fieldbus gateway [31] is manufactured by the same company as the inverter, SEW Eurodrive, and is designed to integrate up to 8 MOVITRAC 07 inverters into a PB DP network. The DFP21B is a PB DP interface card installed into a gateway housing UOH11B (See Figure 10). The gateway communicates with the inverters serially via the SBus [28]; converting PB DP data into a format understood by the inverters and vice versa.

The SBus is a CAN bus [28] network where the gateway acts as the SBus Master and the inverters act as the SBus Slaves. Each device must contain a unique address, which in this case is the VSD number. One block of the cyclic output data for all VSDs is sent from the cRIO PB module to the gateway, which then distributes the information to the appropriate invertor. The reverse process occurs for the cyclic input data where a block of data relating to all of the VSDs is sent from the gateway to the master. A number of parameters had to be set on each inverter to enable the communication via the SBus, to be described in Section 7.1.2.

It should be noted that the gateway is designed for the MOVITRAC B frequency inverters, not for the MOVITRAC A in the UWS; however both inverters have a similar functionality. The MOVITRAC A inverters have been phased out, and the MOVITRAC B are the new models. Hence, the related documentation can be ambiguous and difficult to find. The process of implementing the gateway into the UWS PB network is outlined in Section 7: PB implementation. Figure 11 shows a summary of the gateways I/O and indication lights with Table 4 summarising the function.

Name	Description	Function
Run	PB operation indication LED (Green)	Indication of correct PB operation
Bus Fault	PB bus fault indication LED (Red)	Indication of PB error
X30	9-pin serial port	PB connection port
2^0, 2^1 … 2^6	DIP Switches for PB address	For setting PB station address
AS	Autosetup switch	Autosetup for gateway operation
H1	System Error Indication (Red)	Indicates System Error on the SBus
H2	LED (Green)	Reserved
X24	RS485 Port	RS485 interface for PC diagnostics and parameter setting via MOVITOOLS
X26:1	Screw terminals	SC11 SBus high
X26:2	Screw terminals	SC12 SBus low
X26:3	Screw terminals	GND
X26:6	Screw terminals	GND
X26:7	Screw terminals	24VDC

Table 4 - Gateway I/O summary

Figure 10 – VSD Gateway

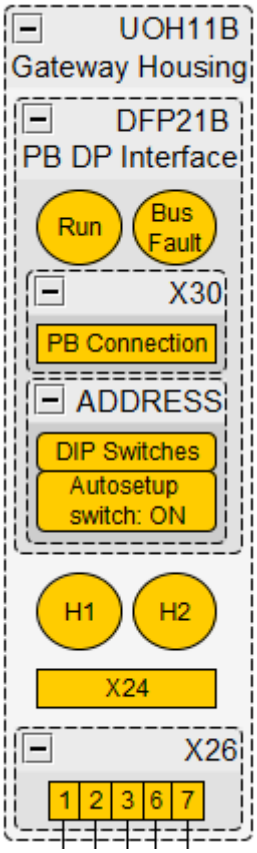

Figure 11 - VSD Gateway Summary of front panel

5.2.4 ELECTRIC FLOW VALVE – Determine which terminals are sinking/sourcing etc

In this project an electrically activated control valve is used to replace the pneumatic valve FV03 to increase the flow area and as a result to reduce the pressure drop across the device. Furthermore, the electric control valve adds a different form of actuation in the existing system. Figure 12 shows the electric flow valve installed in the UWS while Table 5 summarises the wires.

Table 5 - Electric Flow valve terminals/wires

Terminal	Function
1	24VDC negative
2	24VDC positive
3	4-20mA signal (source)
5	4-20mA signal (sink)

Figure 12- electric flow valve installed in the UWS

6 Commissioning the UWS

This section will describe the process of commissioning the UWS. The problems that were encountered, and how they were overcome.

6.1 Operating System and Software Issues

As of April 8th 2014, Windows XP OS was no longer supported by Microsoft [15]. This meant that updates were no longer available for the OS and the PC is unprotected. From an IT perspective, a computer with XP is considered a security threat to the network. It was a major objective to port the UWS server computer to a supported OS. Additionally, the server computer and cRIO for the UWS was running outdated versions of LV. Consequently upgrading the software to the latest versions was a focus of this objective.

Originally the system was not operational and instruments were unresponsive with the client program. With the replacement server machine running Windows 7, the first step undertaken was attempting to remotely run the UWS cRIO code. However, Kane Blay's exploitation of a LV bug to force variable security on the cRIO variable library created errors when attempting to run the program. The errors stated that the shared variables hosted on the cRIO were broken. LV prevents the cRIO code being deployed until the errors in the code have been fixed. This naturally led to a very simple program being created that consisted of 1 while loop and 1 variable. The attempted deployment of the program was halted by an error stating "The current module settings require NI Scan Engine support on the controller…", however the required module *was* already installed on the device.

Following a tutorial by NI, software was reinstalled onto the embedded system, however the same error prevented deployment [32]. The cRIO was completely reformatted in an attempt to resolve the issue, and a project was created from scratch following an extensive guide by NI on cRIO controllers [33]. However there were issues detecting the chassis and using Distributed System Manager. NI support was contacted several times in an attempt to shed light on the issue; they suggested checking LabVIEW compatibility with the NI RIO software [34]. NI measurement and automation explorer (MAX) can be used to configure NI hardware and software, view devices connected to your system and update NI software [35]. After investigating with MAX, the versions were compatible and there was no logical solution to the cause of the error using the Windows 7 UWS server replacement computer. At this point, a different computer was used to test to see if it was the computer itself that was faulty. The old UWS server running XP and LabVIEW 2011 was used following the same tutorials described previously, and it worked seamlessly. Distributed System Manger showed all the modules, channels and CPU; LabVIEW programs could be deployed without any "scan engine" issues whatsoever.

It was assumed that LabVIEW 2014 was the reason of the error so, with the help of the University IT technician, a server platform running Windows 8 and LabVIEW 2011 was used to develop and port code. However, it was discovered later that the cRIO worked perfectly fine with the latest version of LabVIEW running on a different Windows 7 computer. Fundamentally, it was the original replacement server computer itself that was faulty, for unknown reasons. In conclusion, Windows 8 with LV 2014 can be used for the cRIO and the UWS.

6.2 IP address issue

There was an issue with the cRIO controller denying access from the host computer. The error message stated that the server computer "wasn't on the target's allowable access list" A google search provided a solution from a NI forum:

> "1. Open My Documents.
> 2. In the address bar at the top, type ftp://[Your CRIO's IP]
> 3. You should now see the files inside the FTP, as you would a normal Windows folder.
> 4. Right-Click the ni-rt.ini file, and select Copy.
> 5. Paste the ni-rt.ini file on your Windows desktop.
> 6. On your Windows desktop, double-click the ni-rt.ini file to open it. You may be prompted to select an application to open the file. Please select Notepad.
> 7. Edit the server.tcp.access and RTTarget.IPAccess tokens as shown below.
>
> server.tcp.access="+*"
> RTTarget.IPAccess="+*"
>
> 8. Save the ni-rt.ini file in Notepad.
> 9. Drag the ni-rt.ini file from your windows desktop back to the FTP folder, when prompted to overwrite the file select 'Yes'
> 10. Reboot your Compact RIO by pulling out the power, and plugging it back in." [36]

Following the above procedure fixed the issue; which stems from the properties of the project file used to program the cRIO.

6.3 Replacing faulty equipment

This section describes the process that was undertaken to replace the faulty equipment in the UWS, the changes to the wiring and to the original system; and the hardware that was replaced or moved during the commissioning phase. Two MacNaught flowmeters were replaced by PB flowmeters, an electric control valve and a PB VSD gateway were installed into the system. All instrument installation at designated locations was undertaken by the school technicians.

As discussed earlier, small plastic debris collected in the supply tank during the construction of the UWS and caused the positive displacement flowmeters to jam, get stuck and stop the flow. Consequently, the internals of the instruments were removed by the technicians to allow water flow. Thus replacing the faulty flowmeters with working instruments was a key objective; consisting of physical installation of the equipment, and the wiring of the instruments. It was decided that the replacement PB flow meters would be situated at FT01 and FT03 because of their large cross-sectional diameter and to ease the laying of cable; FT03 and FT05 were the locations of the MacNaught Meters. Thus FT01 was moved to FT05 meaning that FT01 would be free for the replacement Flowmeter. A pneumatic control valve FV03 was replaced by the electric control valve which provided a larger cross-sectional diameter, hence higher flowrates, and a different form of actuation.

The FT01 cable was disconnected because the new PB flowmeters required a different cable, however it is still in the cable tray in case needed in the future. Minimal amounts of wiring had to be done for the FT01 movement to FT05; simply the FT01 cable was disconnected and the FT05 cable was used for the new connection. This meant the registered channel stayed the same programming-wise, only it now corresponds to FT05 instead of FT01.FV03 required 4-20 mA, the same as the instrument it replaced, however it also required 24VDC. This was supplied using the previous FT03 cable that runs in close proximity to the valve, since it is no longer used for the PB flow meters, which required different wiring.

The PB flowmeters required 240VAC power, which were wired by the school electrician, with an individual circuit breaker for each flowmeter. It is important to note that 240VAC can easily cause a fatal accident, therefore circuit breakers are required in case one needs to access the internals of the device. A PB gateway was installed in the marshalling cabinet for the VSDs of the system. This together with the signal wiring of the PB will be further discussed in the PB section.

6.4 Equipment testing

Once the communication with the cRIO had been established, equipment testing was carried out to check the instrument operation. Since at this stage the cRIO had been freshly reformatted and by default the C-series modules I/O are network published, the tests were performed through the NI Distributed System Manager. The tests were carried out to test the operation, but not the calibration, of the equipment. The result confirmed that most of the equipment in the system appropriately worked. Faulty equipment found included:

- The digital input module channel 9: “TK06 Lo-Level float switch” which was not functioning.
    - This device needs to be further investigated to determine whether it needs to be replaced.
- Bellows pump (PU07) didn’t seem to pump water, but made sounds as if it was operational.
    - Further investigation needs to be carried out.

7 PB implementation

A broad overview of PB has been provided in Section 5.2. This section will focus on the implementation of PB communications in the UWS specifically. For convenience, it is split into two major categories, the hardware and software aspects. Overall, the PB instruments in the UWS are very sophisticated with a ridiculous amount of functionality. Thus, the settings are kept to their respective default where possible.

Originally, the plan was to have just the flowmeters on the PB network, however an unforeseen technicality led to the VSDs being integrated into the PB network through a PB DP Gateway interface. As mentioned previously, the cRIO PB module requires 2.5W of power and "must be placed in slot 1 with slot 2 empty to appropriately dissipate heat" [18]. However at the time, there was only one slot free for the PB module. Thus one of the modules had to be eliminated to create space for the PB network. Initially, a PB digital output or analog output module was to be purchased to eliminate one of the modules. However due to time constraints this was not a viable option.

It was discovered that the VSDs are capable of serial communication to control the drives, via "SBus" [28]. Since one of the analog output modules only used one of the four channels to control the speed of the VSD, controlling the speed through the serial port of the cRIO could eliminate an entire module. Ultimately, this solution would be temporary and an ugly band aid type fix in code and hardware. Fortunately, a PB gateway was sourced from a previous project that was no longer being used. This gateway model was not designed for these VSDs but it was compatible with the inverters.

A number of VSD parameters must be set to accept the communications from the SBus, which was achieved through a configuration tool called MOVITOOLS [37]. The inverters are controlled via process input/output data words (PDWs), where each drive has 3 IN and 3 OUT. These PDWs are sent as an entire block by the cRIO module to the gateway, which distributes the information to each drive, and vice versa.

7.1 Hardware Implementation

This section discusses the process of implementing the PB from a hardware perspective, corresponding to each devices. As for the wiring of the PB DP network, instruments are daisy chained with the instrument at each end of the bus using resistors to terminate the signal. The ends of the bus are the cRIO and FT03 respectively. cRIO uses external termination having the 9-pin PB connectors with build-in termination. FT03 uses internal termination described in 7.1.1. . Once the appropriate length of the cable is cut and stripped, one of the lines will go into terminal A and the other to terminal B. The plug is simply connected to the cRIO with termination switched ON, plugged into the PB gateway with termination switched OFF then the cable runs through the cable tray to the flowmeters. The flowmeters have 2 physical terminals inside the device rather than the 9-pin connectors, but wiring is just as easy. One cable goes into terminal 27 - A and the other to terminal 26 – B.

7.1.1 PB Flowmeters

The PB flowmeters have to be set up for the hardware addressing via the miniature dip switches located inside the device. This is achieved by unscrewing the faceplate, removing the display module, and carefully sliding out the right most card. There are 12 switches in the top left corner of the card. The top eight are used for DP, while the bottom 4 remain switched in the OFF position. The eighth switch from the top is switched to the ON position to allow/enable the hardware addressing via the miniature switches. Each switch represents a bit location of the address. Thus FT01 has switches 1 and 8 switched ON to enable the hardware addressing with an address of ONE; while FT03 has switches 1, 2 and 8 switched ON to enable the hardware addressing with an address of THREE. [38]

FT03 is the last unit on the bus, thus it needs to terminate the signal. For PB baud rates less than 1.5Mbaud termination can be achieved internally though the miniature switches. On the same board as the hardware address miniature switches, the terminating resistors are activated via SW1, which is located on the top right of the board. There are four switches, which must all be turned ON to enable the terminating resistors.

7.1.2 PB Gateway & Inverter

The wiring for the gateway and invertors is very simple. The PB DP cable is a 9 pin connector that is inserted into the PB connection port on the front of the device with termination OFF. 24V is supplied to the appropriate terminals, and the SBus connection is daisy chained to each VSD. Figure 13 only depicts two VSDs connected to the gateway but the underlying principle is the same. Similar to PB networks, units at the end of the bus have to be terminated appropriately. The gateway is always assumed to be at the start/end of the bus and terminates signals internally. For VSDs however, S12 has to be switched ON for devices at the end of the bus. With respect to the UWS, VSD05 is at the end of the bus, so S12 for that invertor is switched ON. Lastly, terminal 1 of the VSDs is 24V and has to be wired to terminal 2 “enable clockwise” for safety reasons [28].

The dip switches on the front end of the gateway device have to be configured to set the desired address. In the UWS the configured address is 5, thus the three switch positions are binary 101. The AS switch, or auto-setup switch has to be set to the ON position to enable the gateway to automatically find the invertors below its hierarchical level.

Once the devices have been appropriately wired, the internal settings have to be changed to tell the device that its control commands are coming from the SBus. The parameters were changed using MOVITOOL, the configuration software for the VSDs; and are outlined in Table 6. The settings are consistent across all 5 inverters, except for 813 SBus address, where each address corresponds to the VSD number.

Table 6 - VSD Parameters for SBus Control

Parameter	Name	Setting	Description
100	Set point source	SBus	Gets set point from the SBus
101	Control Signal Source	SBus	Gets control commands from the SBus
813	SBus Address	1-5	SBus address (VSD #)
815	SBus timeout delay	0.2s	Timeout delay
816	SBus Baud Rate	1000	SBus transmission speed
870 **871** **872**	PO1 Setpoint description PO2 Setpoint description PO3 Setpoint description	Control Word 1 Speed No function	Content of the process output data words
873 **874** **875** **876**	PI1 Actual Value PI2 Actual Value PI3 Actual Value	Status word 1 No function No function	Content of the process input data words

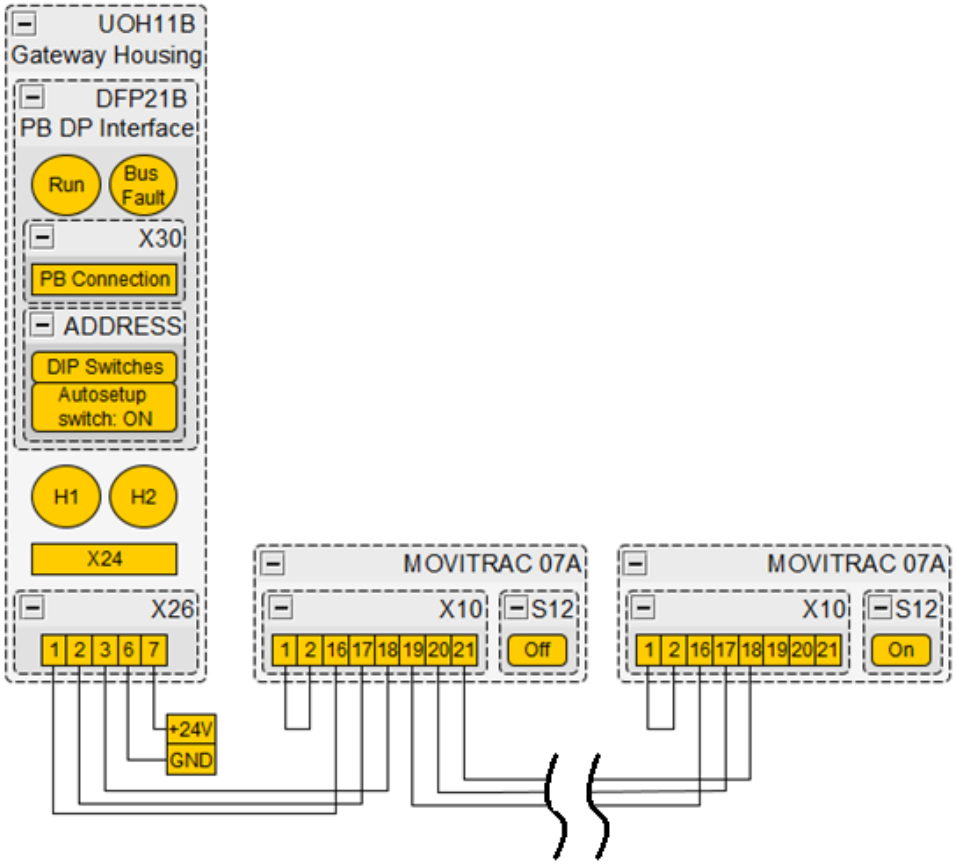

Figure 13 - Gateway/VSD wiring summary

7.2 Software Implementation

The software implementation of the PB DP network consists of configuration of the network, downloading the database file and implementing LV code.

7.2.1 Configuring the PB Network

Before the database file can be downloaded to the cRIO, the NI virtual instrument software architecture (VISA) [39] security settings through MAX have to be changed to allow all targets access. The corresponding remote IP address of the cRIO should be ticked under the “General settings>Remote”. [19]

Once this has been achieved, a network map of the PB DP system can be created with the CONFIGURATOR III software. GSD Files of the devices are available online from the respective manufacturer or alternatively from the PB website. Once the GSD file has been added to the software, devices on the network are dragged and dropped into the configuration and defined. For example, for the UWS, 1 master cRIO PB module is added as address 0, 2 PB flowmeters are added to the network, addresses 1 and 3 respectively, and 1 gateway slave device is added at address 5.

Once the baud rate has been defined and the addresses set, the last step is to define the cyclic IO data to exchange between master and slave. Both the flowmeters and gateway have “modules” which can be dropped into “slots” used for cyclic data transfer. The flowmeters have 9 available slots. Slot 1 is the analog input module, which for a factory setting is volumetric flowrate. The remaining 8 slots are unused for this project. Thus “Analog Input” is added to slot 1 and “empty module” is added to the remaining slots. The Analog input module has 0 output bytes, and 5 input bytes (4 bytes for flow and 1 byte for status) [16], the remaining slots are empty.

The gateway has three slots. However it uses ONE module for all drives located in slot 3 [31], and since each drive has 3 PDW in and 3 PDW out per drive, the UWS gateway has 30 output bytes (6 bytes/3words per drive) and 30 input bytes. The remaining two slots are empty.

Once the configuration of the network is complete, and reflects the physical system and addresses, the configuration can be downloaded to the cRIO device. Once downloaded, the configuration software can be used to initialise the cRIO master and test the PB network via an “online” mode. If there is a configuration error, a blue ring is shown around the device, while if the device is not found at the specified address a red ring is shown around the device. Figure 14 shows a configuration error (blue) and an address error (red). If everything is configured correctly, a green ring is shown around the modules as in Figure 15.

The software can be used to send and observe cyclic I/O data to the configured devices. The flowmeter’s cyclic input data was observed, which had a measurement status of 80 (byte 4) and an alternating analog measurement (bytes 0-3). When the pipe is empty the flowmeter provides a simulated alternating value with a status of 80 [40], confirming correct installation. The VSD communication was tested by setting the speed of each inverter (bytes 2 & 3, 8 & 9 … 26 & 27) to unique numbers with the configuration software to ensure that the SBus addresses were configured correctly. Parameters P094 through P099 relate to the Process Output and Process Input values and can be used for SBus diagnostics [41]. Once the data had been set, each inverter’s parameter for speed (PO2 setpoint, P095) was checked with MOVITOOLs. The correct value was received by each device, confirming proper installation.

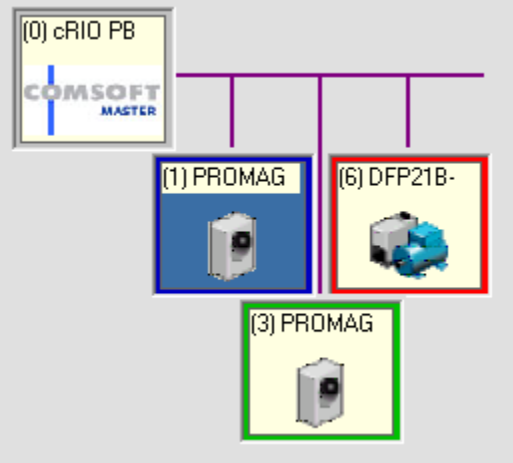

Figure 14 – Diagram depicting PROFIBUS communication error: VSD gateway ((6) DFP21B) not found, flowmeter 1 ((1) PROMAG) configuration error

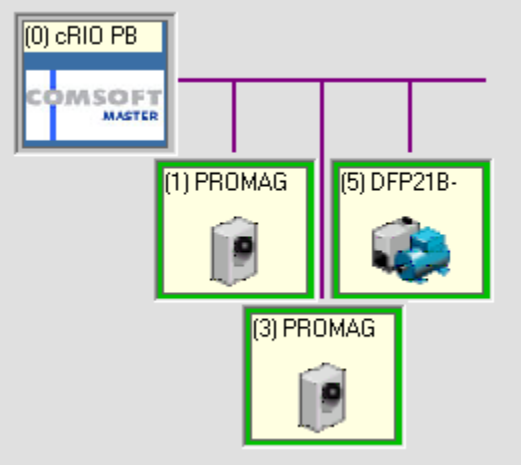

Figure 15 - Successful communications

7.2.2 LabVIEW Architecture

PB communication on the cRIO can be programmed in either FPGA [9] mode through property nodes or through simple API subVIs in scan interface mode. The scan time of the UWS is slow compared to the speeds of FPGA and is not needed for the project, however the APIs provided by COMSOFT still require an FPGA target and VI to communicate between the PB network and scan interface software. Figure 16 shows the basic architecture of the LV software, where the main IO is the scan interface software that communicates through the FPGA target to the PB module. The company provides an example program and describe how to set up the LV project for PB communication. It is possible to use the APIs to send and receive the entire block of cyclic IO data at once or to communicate with each slave individually. For this project each slave is communicates individually for simplicity. The manual for cRIO PB master communication describes in detail how to set up the LV software and communicate with each device [19]. Additionally, the UWS cRIO code is extensively documented within the VI to help users understand the code.

Figure 17 shows the basic topology of the program, the CS_cRIO-PB_DP-MasterExample(Host).vi contains the easy to use SubVIs (APIs) which convey the information to CS_cRIO-PB_DP-MasterExample(FPGA).vi which then communicates directly with the module.

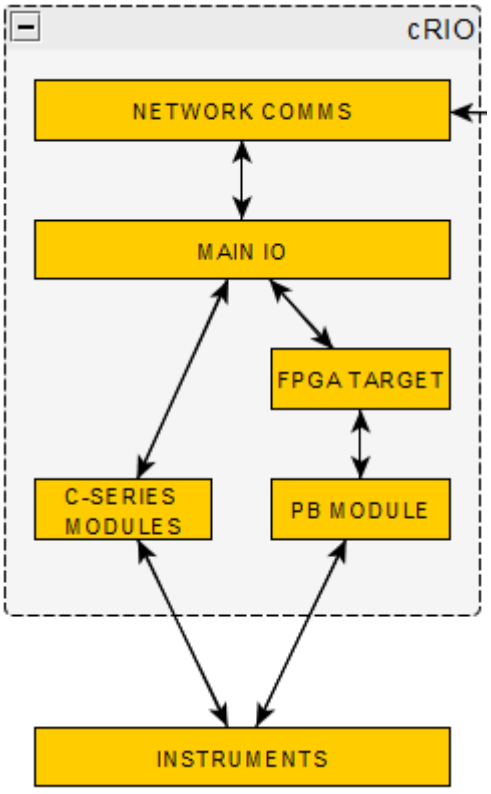

Figure 16 - Summary of LabVIEW architecture

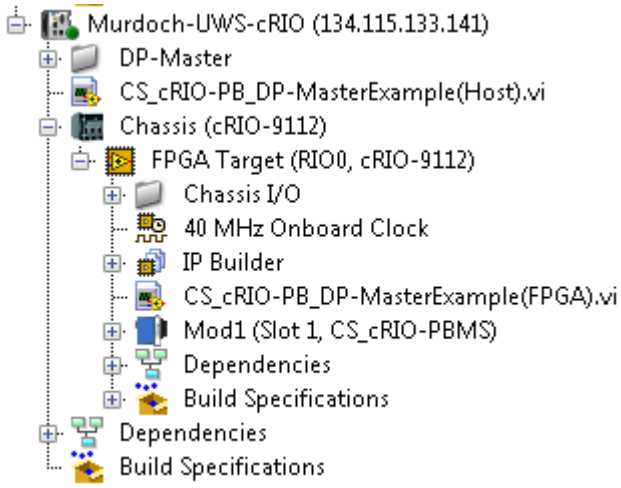

Figure 17- LabVIEW architecture

8 Software redesign

During the course of the commissioning phase, it was discovered that the cRIO code needed to be redesigned. There were a number of reasons that warranted this decision: the hosting of variables on the cRIO, containing the server and cRIO code in one project, the implementation of PB code, to reduce the amount of redundant data and to shift previous students work from the client program to the cRIO device. The majority of the code documentation resides within notes and comments within the Vis.

8.1 Variable Hosting

The LabVIEW cRIO developer's guide [42] was a valuable tool to help determine an appropriate design structure and data passing. Blay originally designed two separate projects, one for the cRIO and one for the server computer, each containing their own variable library. Hosted variables on the cRIO cannot provide variable security. Blay worked around this by exploiting a LV bug to force security on the variables. The fundamental flaw with the original design was the separation of the two libraries. The main variable library for clients must reside on the server computer to allow for shared variable security. This problem was resolved by having one LV project that contained the server computer and the cRIO target. The server computer had the entire variable library split into sub libraries which contain similar variables. The cRIO uses a "networking communication" loop to read and send the network shared variables in and out of the local program. Single process shared variables (local variables) are then used to pass the information around the different loops. Figure 18 depicts the overall architecture of the UWS control system.

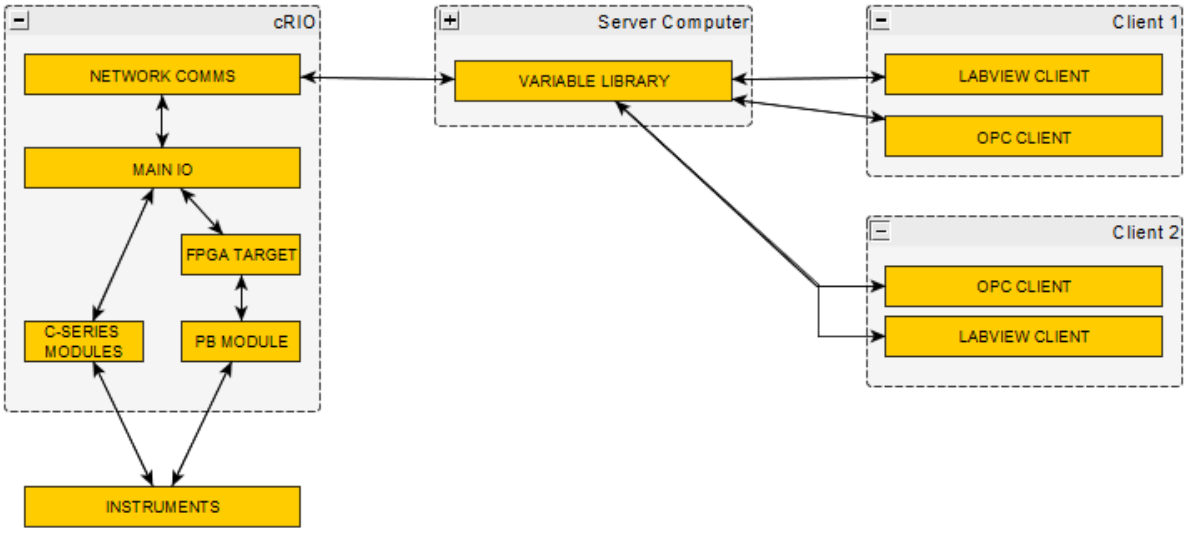

Figure 18 - Overall architecture

8.2 Code modularisation & added functionality

As described in the cRIO developer's guide [42], it was necessary to modularise common code within subVIs. For this reason, the PB communications is entirely handled by a separate loop that passes the information to/from the Main IO loop for processing via local variables. This is desirable should any future students implement additional PB instruments into the UWS.

During the course of the redesign, it was necessary to push more functionality down onto the cRIO system rather than have the client machines full with redundant code. For example, local control can be fully controlled by the cRIO instead of having the loops run on the local machine. This creates less clutter on the client machine but makes the cRIO program a little more complex. The added benefit of having local control on the cRIO, is that a demonstration program can be run quite easily on the machine by enabling the controllers, and disabling network communication temporarily. Once the time limit has expired for the demonstration program, the cRIO returns to normal operation.

The PV of the PID controllers can be selected from any PV in the UWS through a selector integer. The number of the integer reflects which PV is used to control the pump or valve, while a 0 disables the controller. The parameters of the PID are sent via an array: set point, proportional gain, integral time and derivative time.

8.3 Data Redesign

As mentioned previously, the variable libraries of the UWS have been split into sub-libraries to improve the organisation of variables. Additionally, data has been condensed to improve efficiency and elegance. For example, instead of having multiple Boolean values for each pump, they are condensed into a “control byte” for simplicity. The variable libraries are summarized in the following items:

- Manipulated Variables
  - Valve positions
  - Pump speeds
- Process Variables
  - Levels
  - Flows
- Pump and Solenoid valve control
  - Pump enable
  - Solenoid valve enable
- Pump and Solenoid Valve Status
  - Pump Status
  - Solenoid Valve Status
- Alarms
  - Float switches
  - Pump tripped
  - PB Error
- UWS Status
  - Loop times
- PID controllers
  - Controller variables + set-point array
  - PV selector integer

9 Discussion

This section discusses the major problems encountered and how the resulting work deviated from the original plan before addressing the future directions for the UWS.

9.1 Major Problems encountered

9.1.1 Windows OS Port

This was by far the biggest mistake made during the whole thesis project that could have been easily avoided. It led to significant delays in the progress and overall greater delays than originally predicted/intended. The problem could have been resolved a lot quicker if a different computer had been used to test the cRIO from the start instead of making assumptions.

9.1.2 Variable security issue

A lot of time was spent attempting to avoid the method used by Blay [3] to force shared variable security settings on the cRIO controller. The variable security was of utmost importance to prevent anyone on the university network from directly writing to instruments. The process of identifying the problem and a solution led to the redesign of the cRIO code and architecture. Originally, the some of the variables were hosted on the cRIO device and security was forced to only allow the server computer access, however this led to shared variables being broken. The internet was scoured for a solution, but cRIO are just not designed to have security settings, they are embedded systems. The variables should be on a PC that had the DSC module installed to provide variable security.

9.1.3 PB addressing

A simple mistake in hindsight, however a lot of time was wasted attempting to configure the PB network. The addresses in the COMSOFT Configurator program reflected the hardware addresses on the dip switches for the PB instruments; however all the instruments were showing a “station not existent” error when using the online mode. This is characterised by a red box around the device in the PB network and is due to the master not being able to find the specified instrument. After reviewing the appropriate documentation it was found that an important step had been overlooked for the PB flowmeters; a miniature switch within the device had to be turned on to allow for hardware addressing rather than the default software addressing. This eliminated the error for the flowmeters. As for the PB VSD gateway, MOVITOOLS was used to check the gateway’s address where a discrepancy was found. Restarting the device refreshed the PB address from the hardware DIP switches and eliminated the error.

10 Future Work

This section discusses possible directions that can be undertaken by future students working on the UWS.

10.1 OPC Implementation & Advanced control Schemes

Implementing OPC communication would couple perfectly with developing advanced control schemes. With the DSC module installed, network Shared variables are automatically deployed as an OPC server. It is just a matter of setting up and configuring the variables through the OPC server manager. NI has many tutorials online. Despite Jagadeesh, very few students have attempted advanced control schemes in the UWS despite the systems high configurability. None have done a proper case study into the UWS, testing the performance of various control schemes. If done appropriately, this would be valuable resources for future students.

10.2 Updating Diagrams & Documentation

The wiring diagrams and pneumatic diagrams of the UWS need to be updated to reflect the current state of the system. Many of the original documents created by Blay are accurate, however they need some minor updates. Modules have been swapped and wires have been removed during the implementation of the PB network. It is a small but valuable task that would significantly aid future students.

10.3 Demonstration program

Creating a full demonstration program that runs automatically for maintenance purposes would increase the longevity of the plant's equipment. The foundation for its operation has been implemented with local control on the cRIO. Some simple tests need to be carried out to determine the appropriate tuning for the PID controllers before implementing an automatic demonstration program.

11 Conclusion

In Conclusion, the UWS has been successfully commissioned and PB DP network implemented. The UWS server computer is running on an updated operating system Windows 8 with LV software version 2014. Faulty equipment has been replaced, and existing equipment has been tested. A fully functioning PB DP network has been implemented in the UWS. Five VSDs are completely controlled through a gateway and two flowmeters installed.

The cRIO code has been restructured with appropriate variable passing and security, data has been condensed, modularised and simplified. Unfortunately due to time constraints not all of the original project objectives were achievable. Too much time was spent on commissioning the UWS and implementing a PB network, that Advanced control schemes and OPC implementation was shifted to future work.

12 References

[1] J. Sheppard, “Universal Water System (Design Review, Hardware Installation and Testing),” Murdoch, Perth, 2009.

[2] J. Kurz, “National Instruments CompactRIO Control for the Ultimate Water System,” Murdoch University, Perth, 2010.

[3] K. Blay, “Design and construction of a multi-level control system using the Compact RIO controller and LabVIEW,” Murdoch University, Perth, 2010.

[4] O. &. Ray, Process Dynamics Modelling & Control, Oxford: Oxford University Press, 1994.

[5] R. Janosz, “Ultimate Water System, Installation & Commissioning,” Murdoch University, Perth, 2010.

[6] A. Tokhmechi, “Universal Water System,” Murdoch University, Perth, 2014.

[7] National Instruments, “NI CompactRIO,” 2014. [Online]. Available: http://www.ni.com/compactrio/. [Accessed 2014].

[8] National Instruments, “What Is NI CompactRIO,” National Instruments, 2015. [Online]. Available: www.ni.com/compactrio/whatis/. [Accessed 2015].

[9] National Instruments, “NI FPGA,” National Instruments, 2015. [Online]. Available: www.ni.com/fgpa. [Accessed 2015].

[10] National Instruments, “LabVIEW System Design Software,” 2014. [Online]. Available: http://www.ni.com/labview/. [Accessed 2014].

[11] National Instruments, “LabVIEW Datalogging and Supervisory Control (DSC) Module,” National Instruments, 2015. [Online]. Available: www.ni.com/labview/labviewdsc/. [Accessed 2015].

[12] ASM Consortium, “Definition - Abnormal Situation,” ASM Consortium, 2015. [Online]. Available: www.asmconsortium.net/defined/definition/Pages/default.aspx. [Accessed 2015].

[13] J. Ganesan, “Design of Standard Maintenance Program and Case Study on UWS,” Murdoch University, Perth, 2012.

[14] MacNaught Pty Ltd, “Positive Displacement Flow Meters: WM10 - 1" pulse meters, M10 - 1" pulse and LC display meters data sheet,” MacNaught, N/A, 2008.

[15] Microsoft, “Support for Windows XP has ended,” Microsoft, 8 April 2014. [Online]. Available: https://www.microsoft.com/en-us/WindowsForBusiness/end-of-xp-support. [Accessed 2014].

[16] Endress+Hauser, “Technical Information - Proline Promag 50W, 53W - Electromagnetic Flow Measuring System,” Endress+Hauser, Reinach.

[17] National Instruments, “PROFIBUS Overview,” 22 Jun 2010. [Online]. Available: http://www.ni.com/white-paper/6958/en/#toc4. [Accessed Feb 2015].

[18] COMSOFT GmbH, “CompactRIO PROFIBUS DP - Installation Instruction,” COMSOFT GmbH, Karlsruhe, 2010.

[19] COMSOFT, “CompactRIO PROFIBUS DP - DP Master - Getting started,” COMSOFT GmbH, 2010.

[20] International Electrotechnical Commission, “Industrial communication networks - Fieldbus specifications,” 2014.

[21] ACROMAG, “Introduction to PROFIBUS DP,” ACROMAG INCORPORATED, Wixom, 2002.

[22] A. Verwer, “Introduction to PROFIBUS and PROFINET,” VTC, Aberdeen, 2012.

[23] PROFIBUS, “PROFIBUS - Technical Description,” PROFIBUS, September 1999. [Online]. Available: www.itk.ntnu.no/fag/TTK4175/Lab/Profibus/Profibus-Technical%20Description.pdf. [Accessed 2015].

[24] Wikipedia, “Profibus,” 2015. [Online]. Available: https://en.wikipedia.org/wiki/Profibus. [Accessed 2015].

[25] Wikipedia, “RS-485,” Wikipedia, 2015. [Online]. Available: https://en.wikipedia.org/wiki/RS-485. [Accessed 2015].

[26] PROFIBUS Australia, “PROFIBUS Association of Australia (PAA),” 2015. [Online]. Available: http://www.profibusaustralia.com.au/. [Accessed 2015].

[27] M. Felser, “PROFIBUS Manual,” PROFIBUS, 2013. [Online]. Available: profibus.felser.ch/en/index.html?gsd_dateien.htm. [Accessed 2015].

[28] SEW Eurodrive, “MOVITRAC 07 Communication,” SEW Eurodrive, Bruchsal, 2003.

[29] National Instruments, “NI CompactRIO PROFIBUS Master/Slave Module,” 2014. [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/en/nid/208383. [Accessed 2015].

[30] R. Nave, “Faraday's Law,” HyperPhysics, 2012. [Online]. Available: hyperphysics.phy-astr.gsu.edu/hbase/electric/farlaw.html. [Accessed 2015].

[31] SEW Eurodrive, “Fieldbus Interface DFP21B PROFIBUS DP,” SEW Eurodrive, Bruchsal, 2006.

[32] National Instruments, “Install Software on Your CompactRIO Controller,” National Instruments, 2014. [Online]. Available: http://www.ni.com/getting-started/set-up-hardware/compactrio/controller-software. [Accessed 2014].

[33] National Instruments, “Getting Started with CompactRIO - Remotely Monitoring I/O,” 02 May 2012. [Online]. Available: http://www.ni.com/tutorial/11199/en/. [Accessed 2014].

[34] National Instruments, “NI-RIO & LabVIEW Version Compatibility,” National Instruments, 2015. [Online]. Available: http://digital.ni.com/public.nsf/allkb/577CC9A7DCFC73DF8625738400116CC3.

[35] National Instruments, “What Is Measurement & Automation Explorere (MAX)?,” National Instruments, 25 03 2005. [Online]. Available: digital.ni.com/public.nsf/allkb/71544521BDE34FFB86256FCF005F4FB6. [Accessed 2015].

[36] GoCougs, “NI Disscussion Forums,” National Instruments, 20 05 2010. [Online]. Available: http://forums.ni.com/t5/LabVIEW/IP-address-is-not-on-the-target-s-allowable-access-list/td-p/1133613. [Accessed 2014].

[37] SEW Eurodrive, “MOVITOOLS Motion Studio,” SEW Eurodrive, 2015. [Online]. Available: www.sew-eurodrive.com.au/produkt/movitools-motionstudio.htm. [Accessed 2015].

[38] Endress + Hauser, “Operating manual Proline Promag 53 PROFIBUS DP/PA,” Reinach, 2010.

[39] National Instruments, “National Instruments VISA,” 2014. [Online]. Available: https://www.ni.com/visa/. [Accessed 2015].

[40] Endress + Hauser, “Description of Device Functions Proline Promag 53 PROFIBUS DP/PA,” Reinach, 2010.

[41] SEW Eurodrive, “MOVITRAC 07 Operating Instructions,” SEW Eurodrive, Bruchsal, 2004.

[42] National Instruments, “NI LabVIEW for CompactRIO Developer's Guide”.

[43] University of New South Wales , “Thesis Structure,” 30 July 2014. [Online]. Available: https://student.unsw.edu.au/thesis-structure.

[44] B. Hua, “Case Study on UWS and Design of a Weekly Cleaning Program,” Murdoch University, Perth, 2014.

[45] National Instruments, “Setting Up the CompactRIO PROFIBUS Module,” 03 Feb 2014. [Online]. Available: http://www.ni.com/tutorial/11078/en/. [Accessed 2015].

[46] National Instruments, “Labview System Design Software,” National Instruments, 2015. [Online]. Available: www.ni.com/labview/. [Accessed 2015].