5-2020

LabVIEW Improvements and Equipment Redesign of Unit Operations Distillation Column

Madalyn S Wead
University of Louisville

LABVIEW IMPROVEMENTS AND EQUIPMENT REDESIGN OF UNIT OPERATIONS DISTILLATION COLUMN

By:

Madalyn Susanne Wead

B.S., University of Louisville, 2019

A Thesis

Submitted to the Faculty of the

University of Louisville

J. B. Speed School of Engineering

As Partial Fulfillment of the Requirements

For the Professional Degree

MASTER OF ENGINEERING

Department of Chemical Engineering

May 2020

APPROVAL PAGE

LABVIEW IMPROVEMENTS AND EQUIPMENT REDESIGN OF UNIT OPERATIONS DISTILLATION COLUMN

Submitted By: Madalyn Wead
Madalyn Wead

A Thesis Approved On
April 24, 2020
(Date)

By the Following Reading and Examination Committee:

Gerald Willing
Gerald Willing, Ph.D., Thesis Director

Jim Gerstle
Jim Gerstle, Ph.D.

[illegible]
Thomas Roussel, Ph.D.

ACKNOWLEDGEMENTS

I would like to thank Dr. Willing for taking the time to be my thesis director and teaching me everything I need to know and more about the distillation column. I appreciate the multiple demonstrations of starting up the column and dealing with all of the boiler outages along the way. Thanks to Dr. Roussel as well, for all of your LabVIEW wisdom. I could not have created this final product without your expertise and willingness to teach. I would also like to thank Dr. Gerstle for first introducing this thesis opportunity to me and helping me see the potential for future distillation process control labs. Thank you to Mary Stewart for all of your hard work to troubleshoot equipment problems and for your patience with ordering the valves. Lastly, a special thanks to everyone at C&I Engineering and their work in redesigning the distillation system.

ABSTRACT

Unit Operations labs are important tools to provide hands-on learning with various chemical processes. At the University of Louisville, the concepts of distillation were taught in both a Separations Operations course (ChE 436), as well as in the Unit Operations lab (ChE 485 & 486). The distillation lab experiment allows students to connect equations and definitions from textbooks to the physical processes as they occur. LabVIEW was the software development tool used to create a program to control the distillation column from startup to shutdown. Created by National Instruments, LabVIEW is the industry standard for developing process control interfaces using graphical coding to provide a visual representation of dataflow. Improvements were made to both the distillation process equipment and the LabVIEW code to create a more interactive lab experience, enhance students' conceptual understanding of distillation, and prepare students for their future career as chemical engineers.

Equipment issues prevented the distillation column from being fully functional. Non-linear valves limited flow control, temperatures and differential pressures were unable to be measured due to malfunctioning equipment, and the overall configuration created excessive deadtime and inaccessible equipment. These deficiencies prevented the LabVIEW code from performing an automatic startup, increased risk of injury for performing maintenance tasks, and made the process difficult for students to understand.

LabVIEW code issues including broken dataflow, overcomplication, and a restrictive stacked sequence architecture further limited the use of the distillation column.

To address these deficiencies, linear valves were specified and ordered, a differential pressure cell was repaired, and the temperature instrumentation was investigated. The University of Louisville also partnered with C&I Engineering in order to develop a redesign of the distillation column system to eliminate excessive deadtime and increase equipment accessibility. Upgrades to the LabVIEW code were made as well, including converting the overall architecture to a state machine, isolating the logic operations, and removing unnecessary code.

Due to external circumstances, the valves were unable to be replaced and the temperature instrumentation issues were not resolved. In order to test the upgraded code, a simulation program to adjust and measure each control point was developed. Testing the code in a simulation mode created a low risk setting to find and correct programming and logic/sequence issues before applying the software to the distillation column. After addressing an issue with the countdown timer discovered when running the simulation, it was proven that the logic operates as designed. The code is expected to provide a more functional and safer way to operate the distillation column once the remaining equipment deficiencies are addressed.

Improvements made within the scope of this project have produced a safe and strong, yet adaptable foundation for the distillation lab and software. The proposed redesign will create a more streamlined process, better suited to enhance the student learning experience. The updated code is safer for operating the distillation column, allows for an increase in process control, and can be easily adapted for future projects to meet evolving learning objectives. Recommendations for future upgrades include the addition of error handling in the LabVIEW code, as well as developing a process control-based lab to increase active participation with the distillation process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

At the University of Louisville, the distillation column in the Unit Operations lab is a key tool in connecting the theories and mechanisms of separations operations learned in the classroom with the physical process that students may one day see in industry. The distillation column was controlled through software written in LabVIEW which allows for the user to monitor system data such as temperatures and flow rates while also manipulating valve position and pumps. Hardware issues as well as inefficient programming have prevented the distillation column from being used to its fullest potential for the student lab experience. The primary purpose of this project was to enhance the experience of using the Unit Operations Lab distillation column by upgrading the LabVIEW code used to run the column, addressing current equipment deficiencies, and collaborating with C&I Engineering on a redesign of the existing system.

A. Unit Operations Lab

Hands-on learning is an integral technique used to enhance understanding of a particular concept. There is often a disconnect in the understanding between what is read in a textbook and the hands-on experience of that concept. Specifically, in chemical engineering, students study various processes in classes such as Heat Transfer and Separation Operations where they learn equations to model different systems without experiencing the systems firsthand. This can lead to preconceived notions being developed and gaps in knowledge of how a process truly works. In an experiment conducted by Ferguson and Haegerty [3], it was shown that students who were able to learn about pulley systems through hands-on experience were better able to solve application problems related to the concept compared to students who learned about pulley systems solely from diagrams.

In chemical engineering, a unit operations lab provides hands-on experience for students with relevant pieces of equipment to bridge the gap between the equations and theories discussed in the classroom and the physical equipment and processes that occur within them. This experience builds upon a student's initial understanding from the classroom and provides a physical meaning to a once abstract variable in an equation. This type of learning can be referred to as a conceptual understanding, where students go beyond reciting a definition and instead are able to expand upon the concept to further applications [4]. By rounding out a student's conceptual understanding of a process with the unit operations lab experience, they are better prepared to one day manage or design the same process in industry.

McCabe defines the unit operations as operations used to conduct the primarily physical steps of a process such as transporting solids and fluids, transferring heat between materials, and separating components of a mixture [5]. These unit operations are used in conjunction with reaction kinetics to develop various chemical processes. By analyzing larger processes as their individual unit operations, the processes can be simplified, and similarities can be drawn between the same unit operations of vastly different chemical processes across different industries. Common equipment used in a unit operations lab includes heat exchangers, cooling towers, packed bed columns, filters, and distillation columns. These pieces of equipment are typically scaled down to a lab or benchtop size to allow for multiple operations within the same lab space, as well as to reduce equipment and utilities costs. Within the unit operations lab, students are able to perform various types of experiments with different degrees of involvement based on how the lab is designed. Some labs performed are focused more on a student's observations of the processes occurring within a specific piece of equipment while other labs require students to manipulate different variables and actively problem solve how the changes will impact the overall system. Each type of lab has its own benefits for students in providing a more wholistic understanding of the chemical processes seen in class and in the lab.

B. Distillation

One chemical engineering process that can be difficult for students to conceptualize on their own is distillation. Continuous distillation takes place typically within a vertical column and is an operation used to separate a mixture of two or more fluids based on their boiling points. In distillation, the fluid mixture is fed into the column where heat is applied at the reboiler at the bottom of the column. The heat causes the more volatile components of the mixture to vaporize while the least volatile components remain in the liquid phase.

Distillation columns can have either trays or packing as internals of the column to assist in the separation of the fluids through mass transfer. Both internal types have their own advantages and disadvantages. The decision between the two is ultimately based on the needs of the operation, such as having a specific pressure drop, or economic restrictions, such as reoccurring maintenance costs [6]. Since the distillation column in the Unit Operations lab is a trayed column, only the mechanisms of a trayed distillation column will be outlined.

For a trayed column, a vapor-liquid equilibrium is found at each tray. The equilibrium occurs as vapor bubbles rise through the tray and mix with the liquid being held on the tray before draining to the tray below. During this mixing period, mass transfer occurs between the liquid and vapor phases. The more volatile component(s) from the liquid phase transfer to the vapor phase and the least volatile component(s) from the vapor phase transfer to the liquid phase [5]. This occurs at each tray within the column. As the trays continue upward through the column, the concentration of the volatile component in the vapor phase increases until, ideally, pure vapor of the most volatile component exists at the top. To account for the realistic, non-ideal condition at the top of the column, a reflux can be added where the vapors from the top are condensed and a portion of the stream is sent back into the column to be further distilled. Reflux allows for increased top product purity, as well as provides a liquid phase for the top of the column [5].

In the Unit Operations (Unit Ops) Lab at the University of Louisville there is a continuous distillation column used to perform the separation of a water-methanol mixture. The boiling point of methanol is 65 degrees Celsius. The boiling point of water is 100 degrees Celsius. Since methanol has the lower boiling point, it is the more volatile component. Therefore, when heat is applied to the column, the methanol will vaporize at a lower temperature than the water and will be collected as the tops product while the water remains primarily in the liquid phase and is collected as the bottoms product.

FIGURE 1 shows a general schematic of the Unit Ops distillation column. The column is made of glass and has six trays with a seventh stage considered to be the reboiler. The vapor product at the top is sent to a condenser and then to an accumulator to establish enough flow for reflux as well as prevent the reflux pump from running dry. From the accumulator, the line splits so that a fraction of the tops product returns back to the column as reflux and the rest is collected as the methanol distillate in a vessel. The liquid product at the bottom of the column is also split into two streams. One stream goes to a thermosyphon reboiler where it is vaporized and sent back to the bottom of the column. The other stream is collected in a separate vessel as the bottoms product, water.

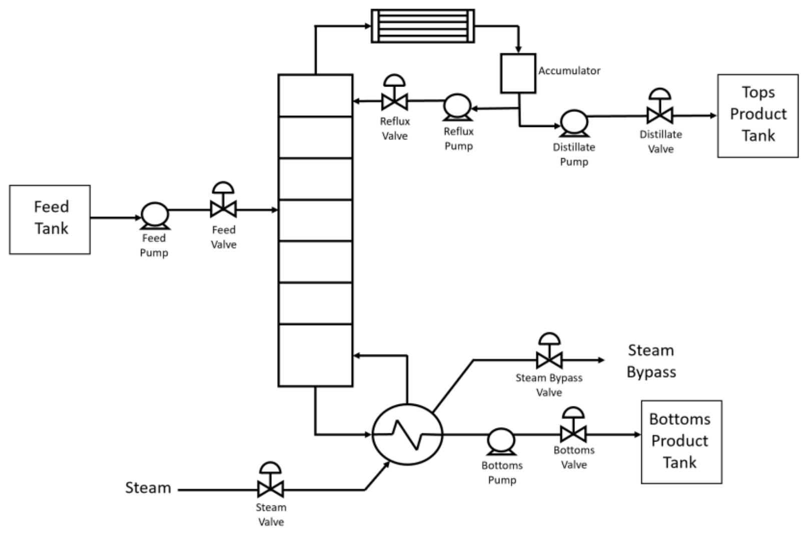


FIGURE 1: General Schematic of Unit Ops Lab Distillation Column Equipment Configuration

A McCabe-Thiele diagram can assist in visualizing the concentrations on each tray. FIGURE 2 is an example of a McCabe-Thiele diagram developed by students for the methanol-water distillation column in the Unit Ops lab [7]. The triangular steps represent the stages within the column which includes each tray and as well as the reboiler for this particular column. The vapor and liquid compositions of methanol can be read at various stages within the diagram which helps to demonstrate the increasing top product purity as the stages progress upward. McCabe-Thiele diagrams are developed under specific system assumptions that are not representative of the actual system however they still provide a rough model of the distillation operation that can assist in visually representing the separation that occurs throughout the column [2].

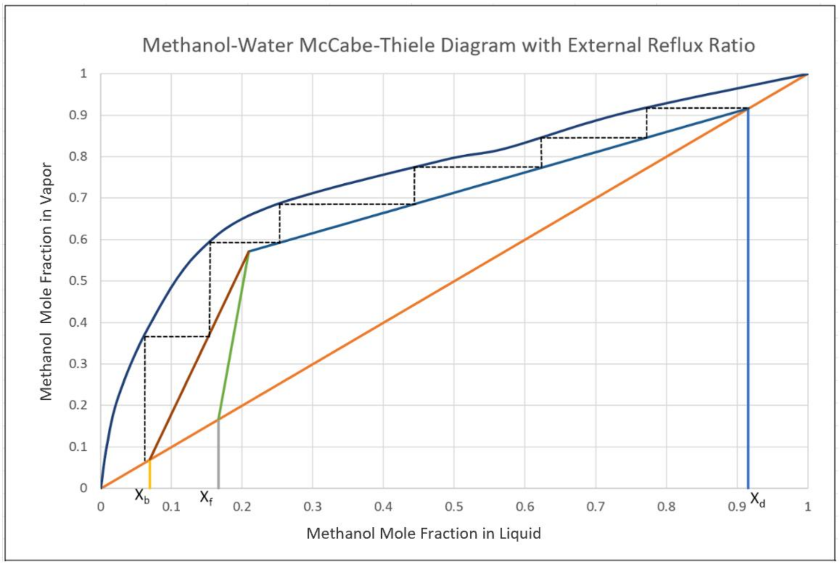


FIGURE 2: Methanol-Water McCabe-Thiele Diagram Developed for the Unit Ops Distillation Column [2]

C. LabVIEW

LabVIEW was used to create the program that controls the Unit Ops distillation column. LabVIEW is software created by National Instruments (NI) that provides a graphical programming system using G-code [8]. The use of graphics allows the user to better visualize the code for its application in comparison with traditional lines of text code that may be more difficult to interpret. Different visual cues help to distinguish between functions and their purpose as well as the sequencing of operations performed by the code.

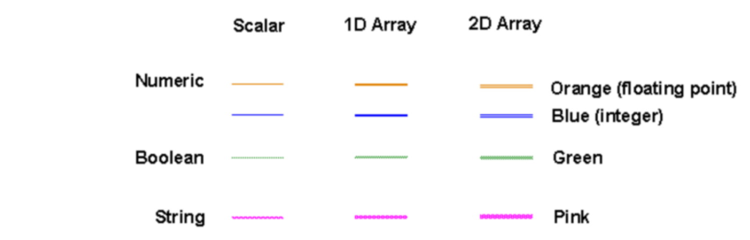


FIGURE 3: Example of LabVIEW Wire Types [1]

One of the foundational concepts of LabVIEW is dataflow. This allows functions to execute only when the necessary input variables are available to feed into the function. Dataflow also allows independent functions to execute in parallel. Wires are used to establish dataflow from inputs to functions to outputs. Different colored wires contain different types of data, as shown in FIGURE 3, created by National Instruments [1]. Blue wires are used for integers, pink wires contain strings, and green wires are for Boolean data. The thickness also represents the data contained within the wire. A thin wire indicates a scalar value, a thicker wire is for a one-dimensional array of data, and an even thicker wire represents a two-dimensional array of data. The use of wires as well as the difference in color and thickness assist in the visualization of the way specific types of data are transferred throughout the code.

A LabVIEW program is called a virtual instrument, or VI [8]. There are two primary components of the VI interface, the front panel and the block diagram. The front panel is the screen which the application user interacts with while the program is running. This is where data collected is presented to the user and where the user can control various parameters and adjust inputs based on the collected data. LabVIEW has various visual elements built into the application such as knobs, switches, charts, and buttons that enhance the user experience and can allow for a more efficient and meaningful interaction between the user and the front panel. These visuals can be arranged freely to fit the specifications necessary for the application and draw attention to the primary components the user is concerned with.

The block diagram is where the G-code is implemented, and the programmer is able to manipulate inputs to produce desired outputs. One type of input used on the block diagram is a control. This is where the user can set a specific value, whether it is an integer, string, or Boolean, that can be used for various functions on the block diagram. For each controller used in the block diagram, a corresponding visual element appears on the front panel for the user to interact with while the program is running. The control element on the front panel and block diagram are linked so that as the controller is manipulated on the front panel, the appropriate controller node on the block diagram receives the correct information from the user’s input. The same feature also happens with outputs called indicators. Indicators display output information on the front panel in the form of single values, arrays, charts, graphs, tank levels, and more. A corresponding indicator node exists on the block diagram for the programmer to wire data to in order to display the desired output on the front panel.

Similar to other coding languages, G-code utilizes structures such as for loops and while loops to build functionality in the coding. The for and while loops operate in the same manner in G-code as in other programming languages however there is a visual aspect that can assist users in understanding what code is performed within a specific loop and how data is carried to and from loops. In LabVIEW, the loops are drawn graphically in large boxes. Inside the box is the code that is performed during each loop iteration. Data can be wired into and out of loops and they can be operated in parallel, nested, or in series as desired. Another type of loop structure used in LabVIEW is the case structure (often called a case statement in traditional programming languages). In a case structure, there are specified cases that can occur either sequentially or as desired based on certain criteria being met. A case structure enclosed within a while loop becomes a powerful programming architecture called a *state machine*. The while loop allows the code inside to continue to run until a terminal condition has been met which allows the program to move between different cases as desired by the user or as programmed into the code. A diagram of a state machine is represented in FIGURE 4.

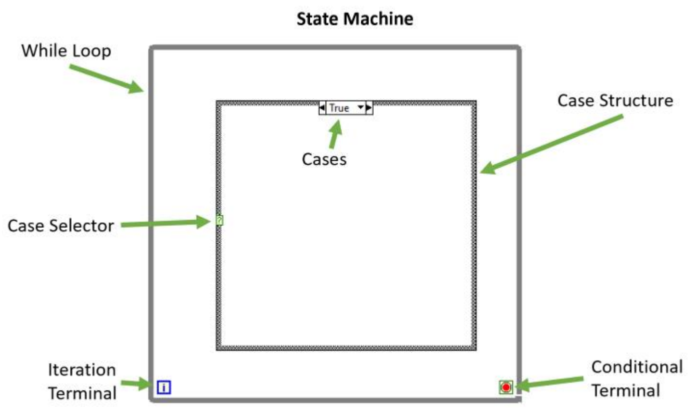


FIGURE 4: Example of LabVIEW State Machine Structure

This specific example only has two states, either true or false. However, an integer or enumerated list (simply called an enum) can also be wired into the case selector to create multiple states based on the integer or the string values of the enumerated list. Associating string values with the numerical list make the cases human readable, while retaining ease of mathematical manipulation and transitioning of the cases.

A second and key type of input used in LabVIEW is via data acquisition, also referred to as DAQ. Using NI DAQ hardware devices, various types of data can be collected from systems and used within the VI. For example, temperature can be measured through a thermocouple and the signal can be sent through an NI DAQ device to be used as input data in a VI. Running averages, standard deviations, or charts can be developed from that data which could also be used for various calculations or setpoints elsewhere within the code. LabVIEW also has functionality to allow users to control instruments. For instance, valves can be opened and closed through user input or through a programmed PID controller and pumps can be turned on and off as needed. Often, DAQ is used in conjunction with controlling instruments in order to initiate or maintain a particular process at desired specifications.

The LabVIEW software platform was designed for the primary purpose of data acquisition, analysis and process control, especially within a laboratory setting [8]. While some programs can be designed to run without much user interaction, LabVIEW specializes in providing the functionality for an interdependence between the VI and the user to successfully control a system. The user interaction is an essential function and must be built into the program and incorporated in the front panel and block diagram. For instance, the same data can be represented with various types of indicators. Consequentially, it is important for the programmer to select the most appropriate indicator for the data being presented. Not all data may be necessary for the user either. Therefore, the purpose of the application and how it will be used are important factors to consider when deciding what information to display and how to display it so that the user is not distracted by meaningless data. The same concept is applied to controllers as well. Some functions will require tighter control than others so the selection of the type of visual controller, whether it is a dial, sliding scale, or typed out integer, is important so that the user is able to efficiently control the program. FIGURE 5 displays three of the numerous types of controllers. While the sliding scale and dial may be more visually appealing to the user, they may not allow the user to have the same level of control as the numeric and thus hinder a user's ability to effectively control a system.

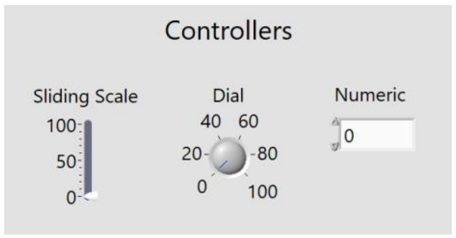


FIGURE 5: Example of Three LabVIEW Numeric Controllers

Overall, the key features of LabVIEW will be utilized to improve the existing software used to operate the distillation column. Dataflow and state machines will be incorporated to improve the code architecture. Controller and indicator selection will be used in conjunction with other techniques identified as “best practice”, such as using subVIs and minimizing wiring clutter, in order to create a more functional and maintainable program to operate the Unit Ops distillation column. Knowledge of the distillation process, process control, and the importance of hands-on experience will influence the code upgrades to create a better tool for learning. The LabVIEW interface encourages interaction between the operator and distillation column. Improving the code used to run the process will allow for students to become the operators in both manual and automatic modes. This will facilitate opportunity for more active involvement between students and the distillation experiment, thus increasing their conceptual understanding of distillation.

II. EXISTING SETUP AND DESIGN

In order to fully understand the changes made within this project, the existing equipment, current use as a lab, and programming of the Unit Ops Distillation Column must be documented. Equipment limitations and programming deficiencies must be identified to determine the necessary upgrades to improve the design and functionality of the system.

A. Equipment

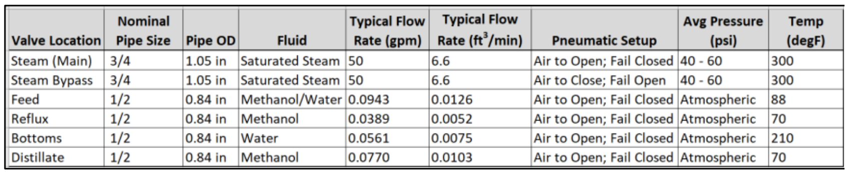
A primary source of limitations of the distillation column were due to the existing equipment used in various parts of the process. For the column itself, the material of construction inhibits the speed of the startup process as well as the ability to open the column for maintenance. Additionally, the selection of valves and malfunction of the accumulator differential pressure cell, bottom and ambient thermocouples, and top RTD have caused compounding issues that further prevent the distillation column from being operated efficiently. Finally, the organization of the piping introduces deadtime to the system which increases difficulty in controlling the process.

The current distillation column used in the Unit Operations lab was built over 40 years ago (1970’s era). The column itself is made of borosilicate glass, more commonly known as the trademarked name Pyrex®. This glass can withstand higher temperatures while allowing students to see the separation process occurring on each stage within. The thermosyphon reboiler is made of glass as well. While glass is beneficial in allowing students to gain insight on the fluid flow and separation occurring within the column, it does introduce a risk of fracture when heat is rapidly applied causing thermal shock. Therefore, it is imperative that heat is slowly introduced to the system to avoid fracturing the glass distillation column and thermosyphon reboiler. Limiting the rate at which heat is applied increases the overall startup time which is an inconvenience to the operator but is a necessary tradeoff to avoid damaging equipment that is costly to replace.

The stages within the distillation column are single bubble cap trays with weirs and downcomers on the sides. Compared to other tray types such as valve trays, a single bubble cap tray is not as efficient, and students often recommend in lab reports to replace the bubble cap trays with a more efficient tray type or packing. A primary factor preventing the trays from being switched out is the material of construction of the gaskets used within the column. The gaskets are made of Teflon® and have dried out significantly over the years due to the methanol in the system. Should the column be disassembled for maintenance, there is a high chance the gaskets would fall apart and would have to be replaced throughout the column.

Other equipment that limits the distillation column includes the control valves. Currently there are six control valves manipulated regularly within the system. TABLE 1 lists each valve as well as the current specifications.

TABLE 1: Control valve specifications for unit ops distillation column



Valve Location	Nominal Pipe Size	Pipe OD	Fluid	Typical Flow Rate (gpm)	Typical Flow Rate (ft^3/min)	Pneumatic Setup	Avg Pressure (psi)	Temp (degF)
Steam (Main)	3/4	1.05 in	Saturated Steam	50	6.6	Air to Open; Fail Closed	40 - 60	300
Steam Bypass	3/4	1.05 in	Saturated Steam	50	6.6	Air to Close; Fail Open	40 - 60	300
Feed	1/2	0.84 in	Methanol/Water	0.0943	0.0126	Air to Open; Fail Closed	Atmospheric	88
Reflux	1/2	0.84 in	Methanol	0.0389	0.0052	Air to Open; Fail Closed	Atmospheric	70
Bottoms	1/2	0.84 in	Water	0.0561	0.0075	Air to Open; Fail Closed	Atmospheric	210
Distillate	1/2	0.84 in	Methanol	0.0770	0.0103	Air to Open; Fail Closed	Atmospheric	70

Both steam valves as well as the feed valve are typically used in heating, ventilation, and air conditioning (HVAC) systems. The primary issue associated with these valves is that they lack a linear control. As the voltage signal to the actuator increases linearly by small increments, the valve opens exponentially. This makes flow control difficult for mid-range valve placements from about 30% to 70% open and renders the valve functionally useless as a control valve. It is important to note the impact this has on startup, specifically for steam control. Since the column is made of glass, it is necessary to apply heat slowly to the system which requires tighter control of steam flow. An exponential steam valve makes it increasingly difficult to regulate the steam flow and increases the risk of overheating the distillation column for an operator not familiar with the exponential valve.

A secondary issue with the steam valves is the issue of operating the valves in a steam-filled environment. The positioners for the steam valves were not sealed which resulted in steam damage that affected the precision of the positioner. For that reason, the main steam valve positioner was re-wired and moved inside the control room to prevent further damage. The positioner for the steam bypass valve remained in the steam environment. Steam damage to the bypass valve positioner has caused the valve to slip, prompting the operator to constantly adjust the positioner to maintain the desired valve position.

Feed flow is also impacted by having a non-linear control valve. Rather than impacting startup, it is normal operation that is primarily affected by the exponential flow control. When the distillation column is in normal operation and is set to run automatically, the feed flow is determined by a PID controller. The programming of the PID controller assumes the valve it is controlling is linear. Therefore, the PID controller has limited control over the feed valve which can make it difficult to maintain steady state if the controller makes a 10% valve change but in reality, the valve opens 30% and pushes the system to a different steady state than desired.

Another equipment limitation on the distillation column includes the bottoms thermocouple and the tops resistance temperature detector (RTD). Both devices are unable to report temperature measurements at their respective locations. The separation process occurring in distillation is driven by the temperature gradient within the column. It is important to have an accurate measurement of the temperature both at the top and bottom of the column to better understand the profile of the column and thus, the concentrations at each stage. Temperature is also a variable that indicates whether the process has reached a steady state. Accurate and repetitive measurements of temperature at the top of the column must be collected in order to determine if steady state has been obtained and for how long it has been maintained.

Differential pressure is another variable measured that is necessary to understand flow rates and levels within the distillation process. One important differential pressure measurement obtained is the level of the accumulator. The purpose of the accumulator is to build up distillate level so that reflux can be initiated. Additionally, having substantial reflux level will prevent the reflux pump from running dry should distillate flows decrease. Due to trapped air in the accumulator differential pressure cell tubing, an accurate measurement of the accumulator level is unable to be obtained. Not knowing the accumulator level risks the accumulator running empty causing the reflux pump to run dry or running full and disrupting the process. It also causes the operator to constantly monitor the accumulator level and make manual adjustments, taking their attention away from other parts other parts of the process.

A final limitation of the equipment is the overall layout of equipment and piping. Due to various projects performed on the distillation column throughout the years, piping has become entangled and longer than necessary for the process. This has significantly impacted the deadtime throughout the column, especially on the reflux line. Due to low volumetric flows in general throughout the system, fluid already does not travel fast. The excessive pipe lengths further hinder efficient fluid flow as a consequence of increased friction on the fluid from numerous pipe bends, as well as the impact of gravity on vertical pipe sections. Increased deadtime directly impacts the ability to control the system and must be incorporated in the PID controllers for better fluid flow control.

A secondary impact of the entangled piping is related to accessibility of equipment. Limited accessibility affects both maintenance practices as well as a student's ability to locate and identify various equipment throughout the process. From a maintenance perspective, it is essential to be able to replace a valve without risking injury due to poor ergonomics or without excessive costs to have other equipment temporarily relocated. With the current location of equipment and piping, it is very difficult for the majority of valves and pumps to be reached without contorting one's body or working in tight spaces. Repairing a valve or pump puts the individual at a greater risk for injury while performing a routine maintenance task.

From a student's perspective, the entangled piping makes it more difficult to understand how the process of distillation works. Specifically, it makes it more difficult for students to identify in person equipment, such as a condenser, that they have only seen in textbook pictures. It also complicates the ability to follow the piping and gain an understanding of the general direction of fluid flow throughout the entire distillation process. For the Unit Ops distillation column, the reflux line in particular is very difficult to follow from where it exits the accumulator to where it is sent back into the column. Reflux can be a complicated aspect of distillation for students to understand which is why it is important for them to be able to physically trace the reflux line and develop the connection between the textbook definition of reflux and what it looks like physically.

In order to eliminate the accessibility issues for maintenance purposes and enhance students' learning experience, a redesign of the distillation column is necessary.

The column itself does not need to be replaced, the primary issue is the layout of equipment. A redesign of the system will allow for modifications made from past projects that are no longer necessary to be removed. This will eliminate some of the excess piping. Reorganizing the location of equipment to more logical locations based on the flow of the process will further reduce excessive piping and thus decrease deadtime in the system. The redesign will take into consideration ease of access of the equipment for maintenance as well as for students to easily identify equipment and follow the process.

B. Unit Ops Distillation Column Lab

1. Manual Startup

While the distillation column required multiple updates, it was still able to be operated as a Unit Ops lab experiment and used as a tool for teaching distillation to students. Due to the state of existing equipment and programming, workarounds were developed in order to continue using the column in a lab setting. The primary circumvention used to operate the column was to run it in manual rather than use the existing code to startup and maintain the column at desired specifications. Running the distillation column in manual requires significantly more involvement from the operator and necessitates the operator’s attentive presence throughout the approximately two-hour long startup period. The operator must also be heavily involved in maintaining a quasi-steady state throughout normal operation of the column which can be up to three hours for the duration of the experiment.

The steps required for a manual startup of the distillation column are documented in APPENDIX I. The LabVIEW code used to run the column allows for both manual and automatic startup modes. When running the program in manual, the operator will follow the same steps that are programmed in the automatic mode; however, the operator determines the timing, makes the valve adjustments, and turns on and off pumps themselves. Running the distillation column in manual requires the operator to be knowledgeable about the order in which startup is performed. A manual startup also introduces risk to the equipment should the operator accidentally miss a step, misinterpret system data, or be away from the distillation column for an extended period of time.

To summarize, the manual startup can be divided into three main components. The first component is to prepare the fluids to be separated. This requires the feed tank to be prepared with a water-methanol mixture. Since the column has a finite quantity of feed mixture, it is operated as a batch system and requires that the bottom of the column be drained so that the bottoms product and top product be transferred back to the feed tank prior to startup. The final part of this component is to apply the feed to the column and fill the bottom of the column to the desired level in preparation for the separation. Each of these transfers of liquids throughout the system requires the operator to manually manipulate the desired valves and turn on the corresponding pumps to initiate fluid flow at the correct time.

The second component of startup is heating. In the heating component, there are multiple phases that occur in order to slowly build up heat in the column. The first part of heating relies on sensible heat transfer driven by the temperature gradient between the liquid in the bottom of the column and the steam being applied. The steam enters through the thermosyphon reboiler and exits through the steam bypass line; therefore, only being in contact with the heat transfer area for a short period of time and limiting the quantity of heat transfer occurring.

The last part of the heating component relies on latent heat transfer. During this phase, the steam bypass line is closed, thus allowing the steam to have a longer residence time for heat transfer. This leads to the steam condensing and the latent energy associated with condensation being transferred to the column. Heat transfer through latent energy is much more efficient compared to sensible heat transfer and allows for a maximized heat transfer to occur between the steam and the feed mixture at the bottom of the column.

There are multiple sensible and latent heat transfer steps that occur within the heating phase. Each step is determined by a specific temperature setpoint that must be met in order to progress onto the next step. During the sensible heat transfer steps, the steam valve is gradually opened to increase steam flow. As the system is transitioned from sensible to latent heat transfer, the bypass valve is incrementally closed in order to initiate a smooth conversion from one type of heat transfer to the next without shocking the system and risking fracturing the glass column. For a manual startup, these heating steps must be performed by the operator. The temperature of the bottom of the column is measured with an infrared thermometer gun rather than the failed thermocouple and the operator must constantly take measurements in order to know when the temperature is approaching the next setpoint to progress to the following step. Once a setpoint is achieved, the operator must open the steam valve and close the steam bypass valve to the next established increments and continue again to monitor the temperature and wait for the next setpoint. This occurs three times with temperature setpoints being approximately 45, 65, and 80°C in that order.

The final component to occur is the transition from startup to normal operation of the distillation column. This occurs following the completion of the heating component where the final heating setpoint has been achieved and top product vapors are starting to be collected as the distillate. At this point, distillate begins to fill up the accumulator and reflux is initiated once the accumulator reaches about 50% capacity. Once reflux is established, the temperature at the top of the column is monitored for steady state conditions where the temperature has no significant fluctuations. This signifies that the system is ready to transition to the normal operation state. Due to external impacts on the system, a steady temperature may not be achieved; therefore, a 20-minute time frame is allowed for the column to reach a steady temperature. If it is not achieved by the end of the time frame, the distillation column will be transitioned to normal operation regardless of a steady temperature for the lab to be performed in a timely manner with the limited feed supply.

From an operational standpoint, the normal operation state in manual still requires significant involvement from the operator. The primary task of the operator is to ensure the accumulator does not run full or empty. A full accumulator would upset the process and prevent distillate from being collected. An empty accumulator would cause the reflux pump to run dry which risks causing damage to the pump. Therefore, the operator must constantly monitor the accumulator level and adjust the distillate and reflux flow rates as necessary. This must be done visually due to the failure of the differential pressure cell on the accumulator, adding additional challenges as the operator must observe the accumulator in the process area while controlling the process from the control room. Constantly monitoring the accumulator level detracts from time available to monitor other important parameters such as temperatures, flow rates, and the level of the bottom of the column. Temperatures are especially important to monitor in order to understand the state of the separation as well as ensure temperatures do not get too high and risk equipment damage.

2. Lab

Once the distillation column is manually started up and has reached a normal operation state, the Unit Operations lab is able to be performed as it is currently designed. Due to the coding and equipment limitations, the lab is primarily operated as an observational lab. This lab focuses on building an understanding of the concentration gradient formed within the column for methanol and water as well as the general way in which a distillation column performs a separation of fluids. The spring 2019 lab procedure followed is documented in APPENDIX I [9]. Essentially, the procedure has students take vapor and liquid samples at each tray within the column once it is in normal operation mode and has achieved a quasi-steady state. The samples are then analyzed using digital refractometer to obtain the refractive index of each sample. From this data, students can calculate the concentrations of water and methanol at each stage within the column and develop a McCabe-Thiele diagram. Students are also tasked with developing an overall mass balance for the distillation column. This connects concepts learned in the Material and Energy Balance (MEB) class students take with a tangible process that they are likely to see again in industry or other lab settings.

There are some limitations and opportunities for improvement associated with this lab. The first is that with the current design of the lab, there are limited tasks for students to perform. Having limited tasks decreases students' involvement and can make it more difficult for the students to be actively involved in the lab operations. There are typically two lab groups of four students assigned to the distillation column lab at a time. This results in eight students in total performing the lab where space around the equipment is already limited. Student tasks in this lab include observing temperature and flow rate readings, taking vapor and liquid samples at each tray within the column, condensing the vapor samples and transporting them to the digital refractometer, and reading the digital refractometer for each sample. For some of these tasks, there is a significant amount of downtime that allows students to get distracted and stop being active participants in the lab. The downtime is further increased as there are two students performing each task where they will have to alternate each time the task is performed. The purpose of having a unit operations lab is to foster active participation in hands-on experiments to build the connection between abstract textbook concepts and tangible real-life operations. Therefore, it is essential to have enough tasks for students to perform in order to maintain active participation.

C. LabVIEW Programming

1. Block Diagram Architecture

The overall architecture used in the original LabVIEW code for running the Unit Ops distillation column was a stacked sequence structure. The stacked sequence architecture originates from the flat sequence structure which is displayed as a film strip with dataflow across the frames from left to right. An example of a flat sequence structure is pictured in FIGURE 6.

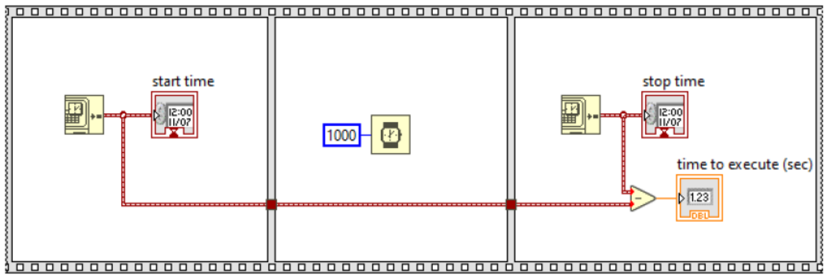


FIGURE 6: LabVIEW Flat Sequence Structure with Sample Code

In this flat sequence structure, the date and time are obtained in the first frame and displayed on an indicator. The data is then passed through the second frame where it must wait for 1000 milliseconds. After this time delay expires, the data is then allowed to move to the third and final frame where the current date and time are obtained and displayed on an indicator again and the difference is calculated between the first date and time and the second in order to determine the time it took to execute the program. This same code can be displayed as a stacked sequence structure as well. The stacked sequence is pictured in FIGURE 7.

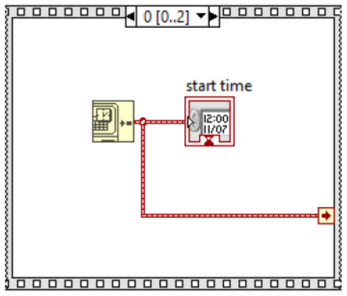
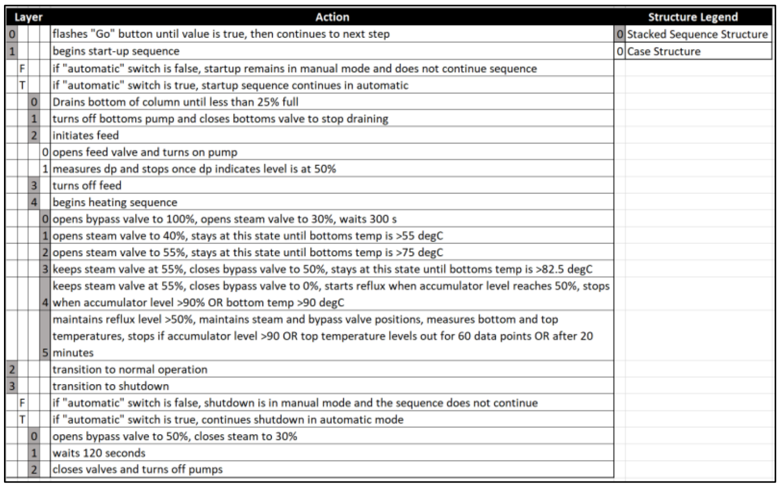


FIGURE 7: LabVIEW Stacked Sequence Structure with Sample Code

The same code exists in the stacked sequence however instead of seeing all frames at once, only one is displayed at a time. The arrow at the top center of the frame allows the user to switch between frames. When the program is running, it will maintain the consecutively numbered order of frames as it would in the flat sequence. Typically, the stacked sequence is used in place of the flat sequence structure to conserve space as they both operate the same. The only functional difference is the manner in which the sequence is displayed, whether the user sees all frames at once or just one at a time.

The distillation column software used multiple stacked sequences embedded within a master stacked sequence to perform the startup, normal operation, and shutdown of the column. The master stacked sequence had four frames: frame zero was idle, one was startup, two was normal operation, and three was shutdown. Each frame must execute completely before moving onto the next sequential frame. This structure can cause issues with displaying system data if it can only be displayed during a particular frame but is needed throughout the duration of the column operation, such as temperatures or flow rates. Case structures were also used within some of the stacked sequences, both as Boolean and integer operations. FIGURE 8 is a summary of the overall architecture of the existing distillation column software. The layers represent the architecture used, either a stacked sequence or case structure. The number of frames or cases in each architecture layer is displayed vertically down the Layer columns. This table helps to provide a visual of the subsequences that occur within each main sequence, as well as a general overview of the order in which various actions occur to startup the column.



Layer				Action	Structure Legend	
0				flashes "Go" button until value is true, then continues to next step	0	Stacked Sequence Structure
1				begins start-up sequence	0	Case Structure
	F			if "automatic" switch is false, startup remains in manual mode and does not continue sequence		
	T			if "automatic" switch is true, startup sequence continues in automatic		
		0		Drains bottom of column until less than 25% full		
		1		turns off bottoms pump and closes bottoms valve to stop draining		
		2		initiates feed		
			0	opens feed valve and turns on pump		
			1	measures dp and stops once dp indicates level is at 50%		
		3		turns off feed		
		4		begins heating sequence		
			0	opens bypass valve to 100%, opens steam valve to 30%, waits 300 s		
			1	opens steam valve to 40%, stays at this state until bottoms temp is >55 degC		
			2	opens steam valve to 55%, stays at this state until bottoms temp is >75 degC		
			3	keeps steam valve at 55%, closes bypass valve to 50%, stays at this state until bottoms temp is >82.5 degC		
			4	keeps steam valve at 55%, closes bypass valve to 0%, starts reflux when accumulator level reaches 50%, stops when accumulator level >90% OR bottom temp >90 degC		
			5	maintains reflux level >50%, maintains steam and bypass valve positions, measures bottom and top temperatures, stops if accumulator level >90 OR top temperature levels out for 60 data points OR after 20 minutes		
2				transition to normal operation		
3				transition to shutdown		
	F			if "automatic" switch is false, shutdown is in manual mode and the sequence does not continue		
	T			if "automatic" switch is true, continues shutdown in automatic mode		
		0		opens bypass valve to 50%, closes steam to 30%		
		1		waits 120 seconds		
		2		closes valves and turns off pumps		

FIGURE 8: Summary of Original Stacked Sequence LabVIEW Code Architecture

Screenshots of each frame and sub-frame are documented in APPENDIX II. All relevant code for each frame was able to be captured in a single screenshot except for the normal operation code executed in frame two of the master stacked sequence. This frame required six screenshots divided into logical groups in order to capture all the code used in the normal operation frame. Screenshots are be labeled in APPENDIX II referencing the layers documented in FIGURE 8.

2. Startup Code

In frame zero of the master stacked sequence, there are no sub-sequences that occur. This frame acts as an idle period waiting for the user to select ‘GO’ and initiate startup. The idle period allows the operator to perform any necessary tasks prior to startup and gives the operator a chance to choose whether startup will be manual or automatic. A while loop allows the program to remain in this frame until the conditional terminal reads true which is initiated by the operator clicking the ‘GO’ button. The use of a property node allows the ‘GO’ button to be blinking as the while loop is executing, calling attention to the user that they must click that button to progress. After the conditional terminal is satisfied, the ‘GO’ button stops blinking. At this point, all of the operations within the frame have been completed and the program progresses to frame one for startup.

Within frame one, there is a case structure which is used to determine if startup will occur in manual or in automatic. A switch on the front panel is manipulated by the user to select manual or automatic. If the switch reads false, the false case on the case structure is selected. This case has no subsequences. Since there are no further subsequences, all of the operations in case one have been completed and the master stacked sequence can continue to frame two. If the switch reads true, a series of subsequences are performed within the true case of the case structure to continue with an automatic startup.

FIGURE 9 shows the first frame of the first subsequence. The purpose of the overall subsequence is to drain the bottom of the column, initiate a fresh feed mixture, and fill the bottom of the column to 50% level. The actions occurring in the frame pictured in FIGURE 9 include opening the bottom valve to 100% open, turning on the bottom pump, and reading the bottoms level differential pressure. These operations are iterated until the differential pressure reads less than 25% full. Iterations occur every 4996 milliseconds, or nearly every five seconds. Due to lack of commenting in the code, it is unknown why such a specific number as 4996 milliseconds was chosen for the wait time, not to mention that this delay "blocks" the code and prevents any other code from executing.

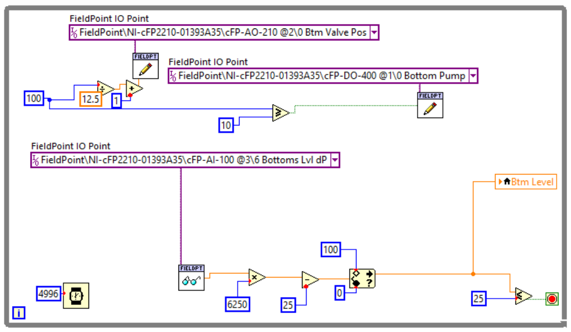

FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\0 Btm Valve Pos
FIELDPT
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\0 Bottom Pump
100
12.5
1
FIELDPT
10
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @3\6 Bottoms Lvl dP
Btm Level
100
FIELDPT
4996
6250
25
0
25
i

FIGURE 9: LabVIEW Startup Sub-Sequence Frame Draining the Column and Measuring the Level

The bottoms differential pressure is measured using a the Read FieldPoint subVI. The signal comes in as a current with a range of 4 to 20 milliamps. The math performed on the bottoms level differential pressure is a conversion from current to percent level. A similar conversion occurs for the valves. In this case, data is being written to the valve position using a Write FieldPoint subVI. The input starts as a percent open valve position and is converted to a voltage signal which is then wired through the FieldPoint subVI and sent to the valve positioner.

Iterations of the code within the while loop continue until enough fluid has drained from the bottom of the column to reach the level setpoint of less than 25% full. Once the level setpoint is achieved, the conditional terminal of the while loop is met with a true value. This terminates the while loop and is the last operation to be completed within the frame, allowing the subsequence to progress to the next frame.

The next subsequence is not as complex as it only has the bottom pump being turned off and the bottom valve being closed. Once those actions have been completed once, the program moves onto the next frame, frame 2 of the subsequence. In this frame, the code is slightly more complex. There is a while loop surrounding a case structure, effectively creating a small state machine within the stacked sequence. The state machine within the subsequence is pictured in FIGURE 10, featuring case zero. There are only two cases and they alternate starting at case zero until the conditional terminal has been met. There is no practical reason to continue alternating between the two cases. Likely, the cases only alternate due to lack of understanding of the structure of a state machine in the original coding. The function of this frame is to initiate feed by turning on the pump and opening the feed valve. This only needs to happen once and there is no need to continue turning on the pump and opening the valve as they must remain open until the bottom level has reached a setpoint of 50% full. Similar to other frames where there has been a while loop, once the conditional terminal has been satisfied on the while loop, the code is finished executing and the program progresses to the next frame.

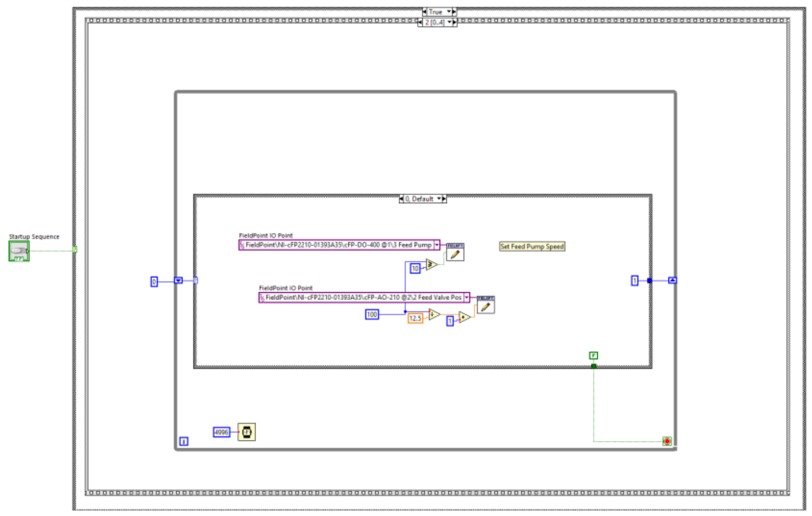

True
2 [0..4]
0, Default
Startup Sequence
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\3 Feed Pump
Set Feed Pump Speed
10
0
1
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\2 Feed Valve Pos
100
12.5
1
4996

FIGURE 10: LabVIEW Startup Sub-Sequence Closing Bottoms Valve and Turning Off Bottoms Pump

Frame three of the subsequence is similar to frame one however in this frame it is the feed pump that is turned off and the feed valve that is closed. Once those actions have occurred once, the program continues to the next frame. Frame four gets more complex as it contains another subsequence within it. Essentially, frame four is the heating phase of the distillation column startup and contains the incremental heating steps within it as an additional subsequence. The first step of six within the heating subsequence is documented in FIGURE 11. This step opens the steam bypass valve 100% and opens the steam valve 30%. The program then must wait five minutes before progressing to the next frame, allowing the distillation column time to slowly heat up with a low steam flow. It is important to note that the steam bypass line is reverse acting; therefore, by sending a signal of zero volts to the positioner, it will fully open.

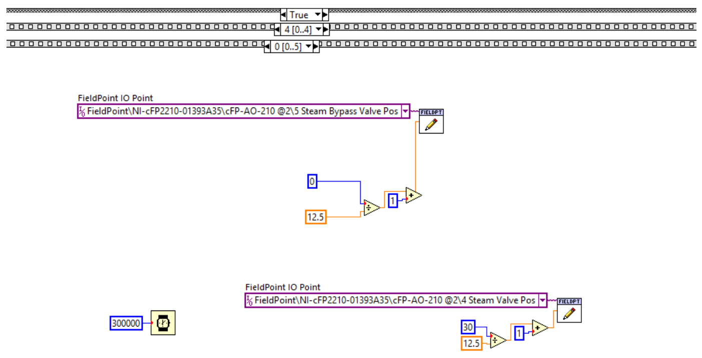

True
4 [0..4]
0 [0..5]
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\5 Steam Bypass Valve Pos
FIELDPT
0
12.5
1
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\4 Steam Valve Pos
FIELDPT
300000
30
12.5
1

FIGURE 11: LabVIEW Startup Heating Sub-Sequence Frame Zero

Frame one of the heating subsequence adds complexity as seen in FIGURE 12. In this frame, the steam valve is opened further to 40%. The temperature of the bottom of the column is also constantly measured during this frame. Each temperature measurement is collected and stored within the code and is passed back around the while loop with the use of a shift register, which is a memory location to share data between loops. An array of the most recent 60 datapoints is built, the average is then taken, and the data is converted from the instrument signal of amps to degrees Celsius. The program remains in this frame until the average bottoms temperature for the last 60 measurements reaches at least 55 degrees Celsius. Once this temperature setpoint is met, the while loop is terminated, and the next frame begins.

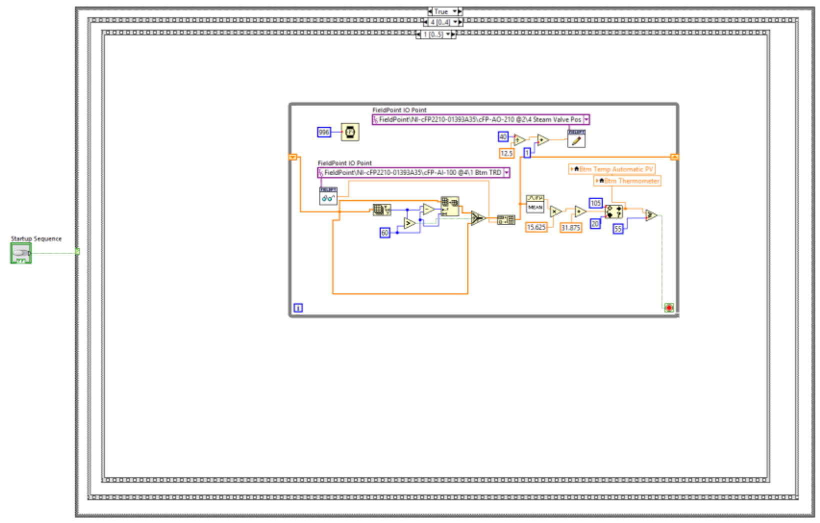


FIGURE 12: LabVIEW Startup Heating Sub-Sequence Frame One

It can be seen at this frame of the heating subsequence the necessity of having live measurements of the temperature at the bottom of the column. It is the measured temperature that determines when the program may progress to the next frame. Without that data, the automatic startup is rendered useless and startup must be performed manually, including manual temperature measurements using an infrared thermometer gun at various intervals to mimic the progression of the code.

Frames two and three are very similar to frame one. Both frames continue to measure the bottoms temperature and keep a running average of the most recent 60 measurements. In frame 2, the steam valve is opened further to 55% and the temperature setpoint necessary to progress to the next frame is 75 degrees Celsius. In frame three, the steam valve remains at 55% open, the steam bypass valve is closed to 50%, and the temperature setpoint necessary to continue to frame four is 82.5 degrees Celsius. In the heating subsequence frames one, two, and three, the bottoms temperature average is plotted on charts using a local variable of the original chart. The data is also sent to a thermometer indicator using a local variable as another way to visualize the temperature.

Similar to the previous frames, frame four also manipulates the steam bypass valve by closing it completely and continues to monitor the bottoms temperature for a new setpoint of 90 degrees Celsius. Unlike the previous three frames, frame four also has control centered around the accumulator level. The differential pressure of the accumulator is measured, and the signal is converted to percent level. When the level is greater than 50%, the reflux valve is opened to 30% and the reflux pump is turned on.

When the level drops below 50%, the reflux valve is closed, and the pump is turned off. Another difference between this frame and the previous heating subsequence frames is that one of two possible conditions can be met to satisfy the conditional terminal and stop the while loop. Either the temperature must reach 90 degrees Celsius or the level of the accumulator must reach 90% full. This frame demonstrates the necessity of having accurate differential pressure measurements around the accumulator for automatic startup. If the program is unable to measure accumulator differential pressure, the level will be unknown to the program and there becomes a risk of the accumulator running empty causing the pump to run dry or running full and upsetting the process. Additionally, the accumulator level is potentially a factor that progresses the code to the next frame, so it is important for the program to know when the level has reached 90% capacity.

Once the first of these setpoints is achieved, the system progresses to the final heating subsequence, frame five, shown in FIGURE 13. This frame builds off the code in frame four, including the same control loop for the accumulator level based on a level of 50%. The bottoms temperature continues to be measured in this frame as well but there is no longer a setpoint for the bottoms temperature to achieve. Instead, the tops temperature becomes the determining factor for the completion of the frame as it indicates when the system has reached a steady state. In frame five, the tops temperature is measured, and the signal is converted to degrees Celsius. Two operations are performed with the tops temperature. The first is that a running average of the previous thirty measurements are obtained, an average is calculated, and the data is plotted on a temperatures chart. The second operation performed is that the standard deviation of the previous 180 measurements is calculated.

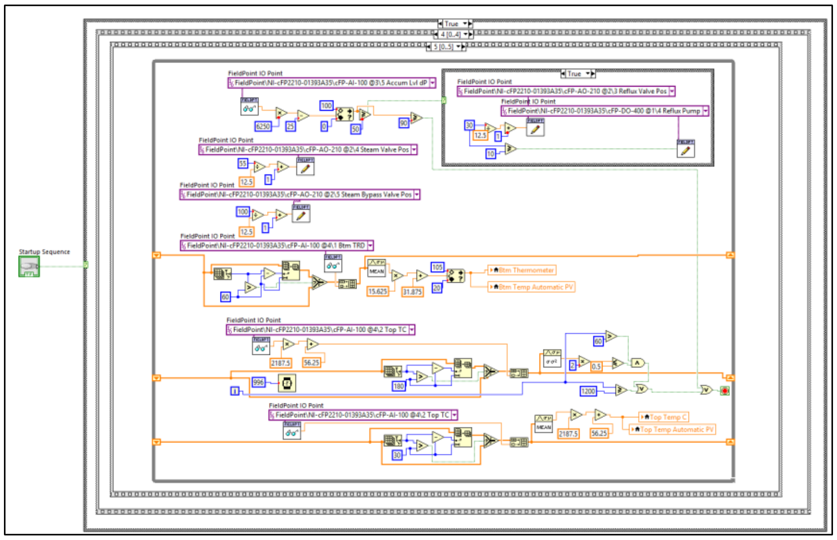


FIGURE 13: LabVIEW Startup Heating Sub-Sequence Frame Five

In order for the conditional terminal to be met with a true value, one of three possible conditions must be met. The first possible condition is that the accumulator level reaches a level greater than 90%. A second possible condition is that two times the standard deviation of at least 60 datapoints of the tops temperature is less than 0.5, indicating a steady state has been achieved. The third possible condition to satisfy the conditional terminal is the program has performed more than 1200 iterations of the while loop in frame five. This is another way to measure time since there is a wait function that only allows the while loop to iterate every 996 milliseconds, or approximately every second. 1200 iterations would result in a wait time of nearly 20 minutes. Therefore, within 20 minutes of the while loop starting, either the accumulator level must reach 90% or the tops temperature must reach a steady state or else the loop will automatically terminate at the end of 20 minutes and the heating subsequence will be finished.

3. Normal Operation

Once the heating subsequence is finished, frame four of the first subsequence will also be finished. This will cause the master stacked sequence to transition to frame two which is normal operation. There are no subsequences within the normal operation frame however there are many operations performed in parallel as well as in series within the frame. Operations within master stacked sequence frame two can be divided into six general groups: five loops and one area of miscellaneous operations. Each of the five loops can be operated in manual or in automatic, as chosen by the operator. When operating in automatic, there are five PID controllers that will adjust the corresponding valve position based on live measurements of the designated process variable in order to achieve the desired setpoint of the process variable. Normal operation in manual allows the user to manipulate each control valve and pump used by the distillation column. Therefore, if startup is chosen to be performed in manual, the master stacked sequence progresses to frame two for normal operation which allows the user to open and close valves and turn on and off pumps as desired to start up the system on their own accord.

The first loop is the bottom level loop, shown in FIGURE 14. In this loop, the bottom level differential pressure and the bottom flow rate are measured. The signals are converted to percent level and percent flow respectively. In the automatic mode, a PID controller is used to manipulate the bottom valve and bottom pump to maintain a desired level at the bottom of the column. In manual, the operator may set the valve position as desired. When the valve position is greater than 10%, the corresponding pump automatically turns on. This ensures that there will be enough fluid in the pipe when the pump turns on to prevent running the pump dry. The bottom level and bottom flow percent are displayed in their respective indicators on the front panel using local variables to provide the operator with a visual on system data.

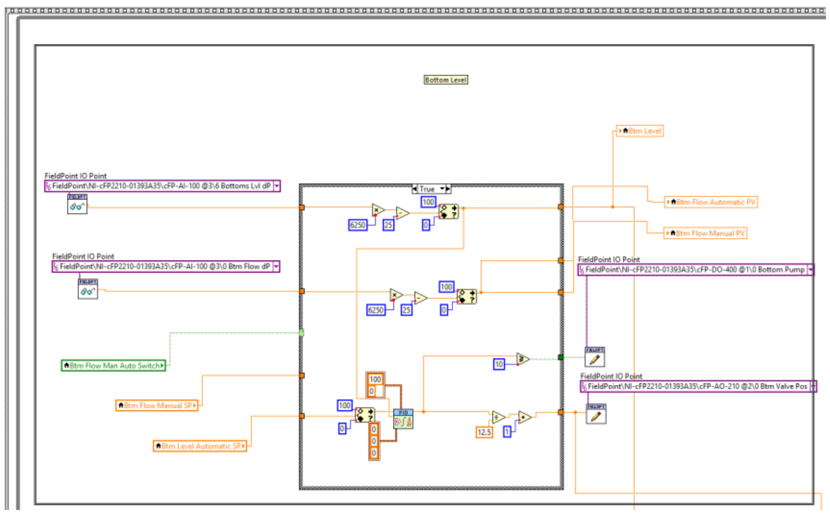


FIGURE 14: LabVIEW Normal Operation Bottom Level Loop

The second loop is the bottom temperature loop. The purpose of this loop is to monitor the temperature of the bottom of the column and the steam flow rate. The code allows the user to adjust the steam valve position as well as display the bottom temperature and steam flow rate percentage. When running in automatic, a PID controller is used to open and close the steam valve to maintain the desired value established from the sum of the measured ambient temperature and the operator's bottom delta temperature setpoint.

Another control loop in normal operation is the feed loop. This loop measures the feed flow differential pressure and plots the percent flow on the corresponding indicator. In automatic, a PID controller manipulates the feed valve and feed pump based on the measured feed flow differential pressure to achieve the operator's setpoint. The fourth control loop is for the accumulator and distillate. Distillate flow and accumulator level differential pressures are measured, converted to percentages, and displayed on the front panel as system data. In automatic, the distillate valve and pump are manipulated to maintain the setpoint for the accumulator level using a PID controller. The fifth control loop is for the temperature at the top of the column and the reflux. In this loop, a running average of the last thirty measurements of the top temperature as well as the reflux flow differential pressure are obtained and displayed on the front panel. Similar to the bottom temperature control loop in automatic, the top delta temperature setpoint is added to the measured ambient temperature as the controller setpoint while the measured top temperature is the process variable. The controller manipulates the reflux valve and reflux pump to maintain the setpoint.

The final group is the miscellaneous group. In this section of the code, various operations are executed. The ambient temperature is measured and a running average of the most recent sixty datapoints is obtained. Ambient temperature data are then used in two of the control loops elsewhere in the frame. There is also code that allows the operator to manipulate the steam bypass valve, the Bottoms to Feed pump, and the Tops to Feed pump as desired. Multiple temperatures, levels, and valve positions are wired to this area and bundled together to create respective temperatures and levels charts that are displayed on the front panel. A final control to note in frame four is the shutdown button. All six of the groups in frame four are surrounded by a single while loop with the shutdown button wired to the conditional terminal. The while loop ensures that the system remains in normal operation until the operator presses the shutdown button. Once the button is pressed, the loop is terminated and all of the code in frame four has been executed. This allows the master stacked sequence to progress to the final frame for shutdown.

4. Shutdown

Similar to frame one for startup in the master stacked sequence, frame five has a case structure with two cases, one for manual shutdown and one for automatic shutdown. In order for a manual shutdown to occur, the operator must know to close the correct valves and turn off corresponding pumps while in the normal operation frame before clicking the shutdown button. Otherwise, there is no code in the manual shutdown case that allows the user to manipulate valves or pumps. This results in the final frame being complete and the program being terminated.

If the user selects automatic shutdown, the automatic case is selected which contains a shutdown subsequence with three frames. The first frame, frame zero, opens the steam bypass valve to 50% and closes the steam valve to 30%. Once these actions are completed, the program progresses to frame one which uses the wait function to wait two minutes. This allows for a gradual reduction in heat load on the column to protect the glass from a rapid cooling that could cause fracture. Once the time expires on the wait function, the program continues to the final frame. In frame two, every pump and valve are turned off and closed respectively, as shown in FIGURE 15. After these actions have been completed one time, the final frame of the shutdown subsequence and the final frame of the master stacked sequence are complete, and the distillation column software is terminated.

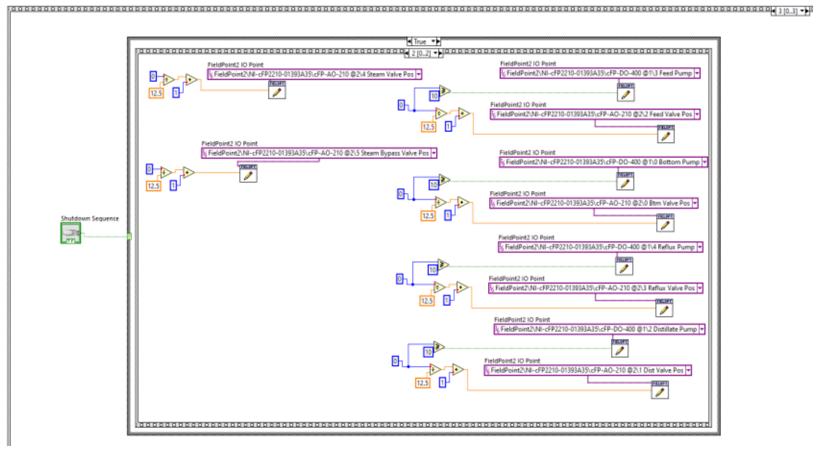

3 [0..3]
True
2 [0..2]
Shutdown Sequence
FieldPoint2 IO Point
FieldPoint2\NI-cFP2210-01393A35\cFP-AO-210 @2\4 Steam Valve Pos
FieldPoint2 IO Point
FieldPoint2\NI-cFP2210-01393A35\cFP-AO-210 @2\5 Steam Bypass Valve Pos
FieldPoint2 IO Point
FieldPoint2\NI-cFP2210-01393A35\cFP-DO-400 @1\3 Feed Pump
FieldPoint2 IO Point
FieldPoint2\NI-cFP2210-01393A35\cFP-AO-210 @2\2 Feed Valve Pos
FieldPoint2 IO Point
FieldPoint2\NI-cFP2210-01393A35\cFP-DO-400 @1\0 Bottom Pump
FieldPoint2 IO Point
FieldPoint2\NI-cFP2210-01393A35\cFP-AO-210 @2\0 Btm Valve Pos
FieldPoint2 IO Point
FieldPoint2\NI-cFP2210-01393A35\cFP-DO-400 @1\4 Reflux Pump
FieldPoint2 IO Point
FieldPoint2\NI-cFP2210-01393A35\cFP-AO-210 @2\3 Reflux Valve Pos
FieldPoint2 IO Point
FieldPoint2\NI-cFP2210-01393A35\cFP-DO-400 @1\2 Distillate Pump
FieldPoint2 IO Point
FieldPoint2\NI-cFP2210-01393A35\cFP-AO-210 @2\1 Dist Valve Pos
0
12.5
1
10

FIGURE 15: LabVIEW Shutdown Sub-Sequence Frame Two

5. <u>Code Deficiencies</u>

There are multiple deficiencies repeated throughout the original distillation column software that make it less than ideal to use to run the distillation column. These deficiencies include dataflow, lack of organization, overcomplication, and the overall architecture of the code. While some of the shortcomings simply make it more difficult for the user to interpret the code, others can have operation implications such as excessive startup times or even the potential of damaged equipment.

The first deficiency to discuss is that of dataflow. There are multiple dataflow issues that occur throughout the code. One dataflow issue occurs due to the way the code manipulates valves and corresponding pumps. FIGURE 16 demonstrates the code that is repeatedly used throughout the entire sequence to perform this task. In this specific instance, the feed valve is being closed and the feed pump is being turned off. The complication with how this code works is centered around timing. It is unknown which action will occur first, closing the valve or turning off the pump. It is also unknown how much time will pass between each action occurring, especially if there are background applications running that increase the activity of the computer's central processing unit (CPU). This can cause problems if the valve is closed first and it takes a while for the pump to be turned off, increasing risk of running the pump dry. Likewise, when the code is used to open and valve and turn a pump on, there is the risk that the pump is turned on before the valve is open, again increasing the risk of running the pump dry. This code is copied and used throughout the startup, normal operation, and shutdown of the distillation column which increases the overall risk of equipment damage due to the frequency of occurrences.

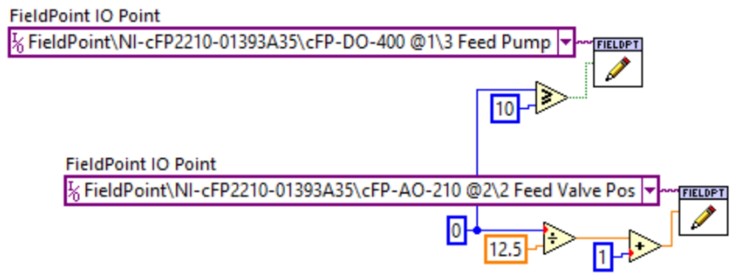

FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\3 Feed Pump
FIELDPT
10
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\2 Feed Valve Pos
FIELDPT
0
12.5
1

FIGURE 16: LabVIEW Code Used to Manipulate Valves and Corresponding Pumps

There is a second dataflow problem associated with manipulating the pumps and valves; specifically, in frames where a system condition must be met before progressing to the next frame. This can be seen during startup when draining the bottom of the column to the desired level as well as in the heating subsequence when waiting for temperature setpoints to be met, as shown in FIGURES 9 and 12. A while loop is used in these scenarios because it usually takes more than one iteration of the code before the system meets the desired setpoint. The while loop allows for as many iterations to occur as necessary which can be hundreds or thousands depending on how long each iteration takes and how fast or slow the system responds. While temperatures or levels are being measured with each iteration, the valve and pump are also being positioned with each iteration. Unlike temperature and level, however, the valve and pump configurations do not change with each iteration as they just need to be set once in the beginning and left alone until the next configuration is necessary. Constantly setting the valve and pump configurations to the same value increases traffic within the program as well as computer memory and can significantly slow down the program with unnecessary actions.

The use of local variables is a third dataflow deficiency within the program. A local variable is represented with the picture of a house next to the variable name as shown in FIGURE 17. Local variables can be created for any control or indicator on the front panel and are used to read from or write to both controls and indicators. This functionality breaks the concept of dataflow in LabVIEW because a controller can be updated with a value wired to the local variable rather than the controller itself. Or the value of a controller can be read using a local variable and used elsewhere in the code making it difficult to follow the flow of data.

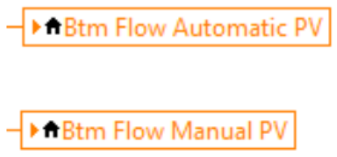


FIGURE 17: Example of LabVIEW Local Variables Used in Existing Distillation Column Code

The primary concern with using local variables throughout a program is the creation of a “race condition”. This can occur if there are two local variables of the same indicator operating in parallel. Depending on the code executed, there could be two different values being sent to each local variable. Since both local variables update the same indicator on the front panel, it becomes a race between the local variables as to which value wins and is displayed on the front panel and carried through the rest of the code. A similar race condition can happen if two local variables are reading from the same input, but due to lack of dataflow control, they read at different times which results in different data being passed through the wires for the same variable.

Local variables are used exclusively throughout the original distillation column software to update indicators and read from controls. At any given frame within the stacked sequence, there is only one local variable for each associated indicator or control. Therefore, no race conditions are apparent as the code was originally written; however, a norm of using local variables has been established in the program. This increases the risk of program updates made in the future to overuse the local variables and unintentionally create a race condition. Overall, it is best practice in LabVIEW to avoid using local variables in order to prevent any possible race conditions from occurring.

An additional impact of using local variables exclusively throughout the code is the limitations set on when indicators are updated. One example of this is the temperature chart. The local variable for this chart only exists during normal operation. This means that the temperature chart does not have live temperature data until the system has transitioned into normal operation. Since there is an entire subsequence centered around the gradual heating of the bottom of the column, it might be useful to have the temperature data appear on the chart during startup as well. The chart would provide a visual representation of the rate of change for the temperature and assist the operator in ensuring the temperature does not increase too quickly and risk damaging the glass column.

The local variable limitation applies to nearly every indicator on the front panel. Since local variables are used to update only a handful of indicators at a time during each frame of the startup sequence, only that handful of indicators are showing live data. This can make it difficult to understand the state of the system as a whole during startup when the user can only see a fraction of the system data that is potentially available to them. It is also more likely for an error to occur during startup since there are so many steps that must occur in a specific order. Live system data must be available to the operator during startup to ensure the equipment is functioning properly and the distillation column is starting up correctly. Should there be an issue, it would be very difficult to troubleshoot with only two or three indicators displaying live measurements.

While the shutdown button is not represented as a local variable within the stacked sequence, there is also an issue of accessibility of the shutdown button. Similar to the problems associated with only certain indicators being updated during specific frames, the shutdown button is only able to be used during the normal operation frame. If during startup there was an operational issue such as a malfunctioning pump or valve, the startup would need to be safely aborted and the system shutdown in a safe state. With the shutdown button only accessible during normal operation, there is no way for startup to be safely terminated and the program would have to be forced to stop, leaving the pumps and valves in an unknown and uncontrolled state. This increases the risk of equipment damage should the program need to be terminated during startup. Due to this deficiency, it would be recommended that even if the temperatures and the differential pressure on the accumulator were working properly, the code should still not run in automatic in the event of a failure during startup that requires a shutdown. A shutdown during startup would not be able to be performed safely with this code and would risk significant equipment damage.

The final issue centered around dataflow is blocking, where the program is restricted to only performing one specific task and cannot perform anything else until that task has been completed. Blocking is used both with the wait timer as well as inside of while loops throughout the code. One prime example of blocking is during the heating subsequence. The while loops are used to keep the program in the same frame until a specific temperature setpoint is achieved. Although the while loop is effective in keeping the system in the desired state until it is ready to move onto the next step of startup, it also is effective in preventing other code from being executed. The programming used in the original software essentially prevents other code from executing as the while loop is operational, which drastically limits the functionality of the program. This is likely why only handfuls of system variables are updated on the front panel indicators at a time during startup. Blocking prevents the rest of the system data from being continuously updated and cannot be compensated for without a restructure of the entire program.

A second primary example of blocking occurs during the shutdown subsequence. In frame one, the only code is a wait timer for 120,000 milliseconds, or two minutes.

Only one iteration of the code will be executed and because of the wait timer, it will take two minutes for the next iteration to occur in the next frame. This blocks the program from doing anything else during that two-minute wait period between iterations. Should there be an emergency that the operator must respond to quickly, the program is inoperable for the two minutes and no other task may be performed to mitigate the emergency. To exit this wait period, the program must be terminated, which leaves the control points in whatever state they were in at the time.

Another type of deficiency in the original programming is the lack of organization and overcomplication of code. Throughout the entire stacked sequence, the code is messy due to overlapping wires and unnecessary bends in the wires, making the dataflow difficult to follow. This especially increases with the heating subsequence with the use of the shift register to pass data through the while loop and the method in which the arrays of temperature data are created. LabVIEW has a feature that allows the programmer to clean up the wires on the block diagram. This feature makes the wires more efficient in their routing from one node to the next and simplifies the appearance of the code. There is also functionality to align nodes on the block diagram both vertically and horizontally as well as equally space them out. A cleaner alignment of nodes on the block diagram further simplifies the appearance of the code and makes it easier for someone to follow the code years after it was written. It is evident that these features were not used in the creation of the distillation column software. Wires have multiple unnecessary bends and are excessively long which adds clutter to an already busy screen.

The normal operation frame is another example of lack of organization. While there is a significantly large amount of code to execute during this frame, space was not used efficiently which resulted in code that extends well beyond the viewing area of the monitor. Since LabVIEW does not allow for the programmer to zoom out, it is increasingly difficult to interpret the code being executed during normal operation. The normal operation code has slight organization in the divisions of logical control loops; however, some data is used across multiple loops and temperature data are wired to a shift register in the while loop which causes overlap in wires throughout the frame. The overlapping wires increase the difficulty of following the dataflow and interpreting the function of the code.

Much of the mathematical operations performed within the code can be simplified. The conversions of a signal to a process variable unit and vice versa are repeated constantly throughout the stacked sequence. For example, each time a valve position is set, the valve position percent is divided by 12.5 and then the quotient is added to one in order to convert the percentage to a voltage signal. Every time a valve position is changed, the mathematical operation to convert the valve position percentage to a voltage signal is repeated, taking up space and adding clutter. Traditionally, when code is repeated at least twice within the same VI, it is logical to create a subVI (akin to a subroutine in traditional text-based programming languages) to perform the same operation. Replacing the conversion math with a subVI would clean up the code, save block diagram space, and make the program significantly easier to follow without being distracted by repetitive conversion math.

A second overcomplication of a mathematical operation is seen when the temperature arrays are built during the heating subsequence and the normal operation frame. The specific code to perform this is highlighted in FIGURE 18. The way the code is written, temperature data points are collected with each iteration and sent to an array. The size of that array is measured and the code checks if the size is greater than 60. If the size is not greater than 60, the entire array is passed through and one more temperature measurement is added to the array to create a new array. If the size is greater than 60, then only the most recent 60 datapoints in the array are passed through and added to a new temperature measurement to create a new array. The new array is then averaged and converted to degrees Celsius. This is a complex way to obtain a running average of the most recent 60 datapoints and takes up significant space on the block diagram.

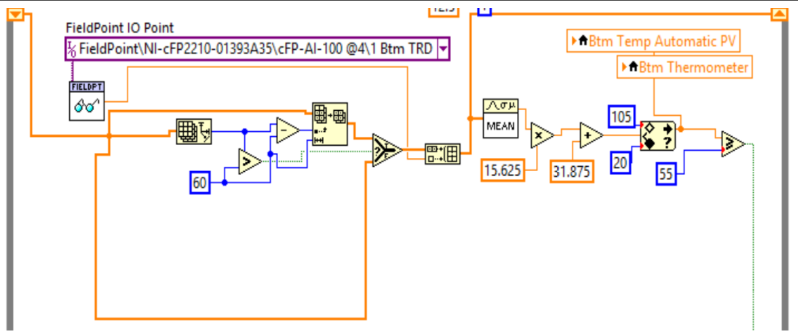


FIGURE 18: Example of LabVIEW Code Used to Generate Array of Most Recent 60 Temperature Measurements

The final deficiency to discuss is the overall architecture selected for the distillation column software. While LabVIEW offers the stacked sequence structure, it is generally known within the LabVIEW community as best practice to avoid it [10]. Stacked sequences face dataflow issues due to the lack of a shift register to pass data frame to frame. This results in the use of sequence locals to pass data which can be easy to misuse and tends to disrupt dataflow. Another issue of stacked sequences is that the frames are unable to be labeled and there is no way to revisit a frame that has already occurred. This limits the functionality of the structure and can make it more difficult to use. It is generally accepted as best practice to use a finite state machine in place of a stacked sequence structure [10]. This programming architecture eliminates dataflow issues of sequence locals and each state can be named and revisited as needed by the user.

6. Front Panel

The front panel of the distillation column LabVIEW software is shown in FIGURE 19. This is the interface the operator works with to control the distillation column and monitor system variables such as flow rates, levels, and temperatures. Switches are used to select whether certain aspects such as control loops, startup sequence, and shutdown sequence are operated in manual or in automatic. There are an additional two switches in the top right corner of the screen to turn on and off the Tops to Feed and Bottoms to Feed pumps. The screen is loosely divided into five control areas: reflux/top temperature, feed, distillate/accumulator, bottom level, and bottom temperature. These match the groupings found in the normal operation frame of the master stacked sequence. Each group has controls to set valve positions or temperature and level setpoints as well as indicators to display flow rate percentages and temperatures. The levels of the accumulator and the bottom of the column are shown graphically on the distillation diagram. As the levels increase, the tanks fill up to provide a visual representation of the level. There are three charts displayed on the front panel. On the bottom of the screen are the Levels and Temperatures charts from left to right respectively. The third chart is the Feed chart. It is partially hidden on the left side of the screen and it displays the feed valve position and flow rate percent.

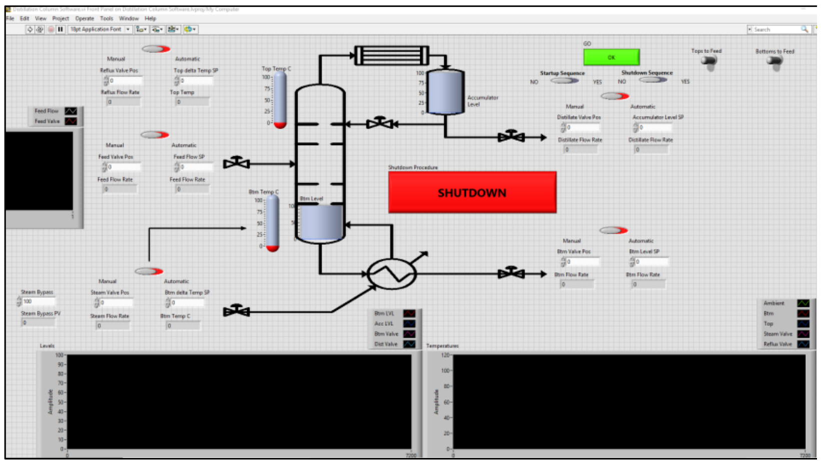


FIGURE 19: LabVIEW Front Panel of Original Distillation Column Software

One issue with the current front panel configuration is that all of the variables do not fit within the allotted screen area. The operator must scroll left to right in order to see all the controls and indicators. Overall, the front panel is not organized efficiently which contributes to the inability to fit all of the variables within the set screen area. It is also difficult to understand what controls and indicators are associated with each part of the system. Control labels (akin to variable names) add to the confusion where two indicators have the same label but represent a different set of data. The labels on the switches can be improved as well since some of the labels are located slightly too far away from the switch. In the case of the two pump switches in the top right corner, it is difficult for the operator to tell when the switch is on or off based on the orientation of the graphic alone.

Commented [R1]: There are no variables in LabVIEW Control names ("labels") are the closest thing

Not all of the data presented on the chart is relevant to the operator. Specifically, the feed chart may not be necessary to plot based on the current use of the column in the Unit Ops lab. The change in valve positions over time is not currently relevant enough to include as a trend on the charts and distracts from the more important operational data such as levels and temperatures. Valve positions are already visible on the corresponding controls which should be sufficient for the operator when controlling and monitoring the system. Overall, the front panel of the distillation column program can use some reorganization to make it more user friendly and functional for the operator. Additionally, updates will make the program more intuitive and allow for students to become operators of the process.

III. CHANGES MADE

In order to improve the current distillation column system and transition it to a more effective tool in the Unit Ops lab, many improvements were made. Equipment upgrades, including developing a proposed redesign for the entire system, assist in making the distillation column fully functional and more efficient. LabVIEW code updates dramatically improve the startup, normal operation, and shutdown sequences and allow for an increased control and safer operation of the distillation column.

A. Equipment

The equipment upgraded within the scope of this project include control valves and the accumulator differential pressure cell. The lack of measurements from the three temperature measurement points were also attempted to be addressed. In order for any new startup code to be tested on the distillation column itself, the temperature measurements and the accumulator differential pressure cell must be functioning. While the control valve upgrades are important as well, they are not considered vital to testing the startup programming performance.

The initial goal for addressing the control valves was to replace all six valves as documented in Table 1. Since there were two types of control valves used throughout the system, it would simplify documentation and bring consistency to the equipment if all of the valves were replaced with the same brand and general type of valve. This would help with maintenance efforts in the future as well as ensure a similar level of control throughout the process. A quote was obtained from Fischer Process Industries based on the specifications of each valve outlined in Table 1. A copy of the original quote for all six valves is documented in APPENDIX III. Four half-inch slot ball valves were quoted for the feed, distillate, reflux, and bottoms lines. Two three-quarters inch 15° V-Ball valves were quoted for the steam and steam bypass lines. Both types of ball valves quoted allow for a more linear control of the fluid. The slot ball was specifically selected for the half-inch line valves due to the lower flows experienced in these pipes. Both types of valves are paired with an electric actuator with a 0 - 10 voltage signal, consistent with the existing signal for the valve positioners. The electric actuator will replace the pneumatic actuators currently used and decrease clutter from the tubing associated with the pneumatics. Due to issues associated with the steam-filled environment for the steam and steam bypass valve actuators, an important feature of the electric actuator is that the body of the actuator is sealed to prevent steam from leaking in and affecting the electronic components.

External circumstances led to the decision of ordering replacements only for the steam and feed valves. These are two of the three HVAC valves with non-linear control and were identified as the most critical valves to replace out of the six total. While the steam bypass valve is also an HVAC valve, the level of control required for the valve is minimal compared to the steam and feed valves and was therefore not considered a priority to replace given the circumstances. Unfortunately, due to additional external factors, including the global pandemic of Covid-19, the valves were not able to be obtained and installed within the time frame of this project. It is planned to resume the process of ordering and installing the valves once the pandemic has ended and business are able to operate again.

The accumulator differential pressure cell was also addressed within the scope of this project and is necessary for the trial of the updated startup code. The issue associated with this piece of equipment was that there was an air bubble stuck in the tubing of the differential pressure cell that prevented an accurate pressure from being measured. To correct this, the piece of tubing with the air bubble was removed and replaced with new, clean tubing. This was only done with one piece of tubing because the comparative pressure measurement for the differential pressure cell is ambient pressure since the accumulator is not a pressurized vessel and is essentially open to atmosphere.

The final equipment to be addressed are the bottoms thermocouple, the ambient thermocouple, and the tops RTD. In order to troubleshoot the equipment, a separate LabVIEW VI was created to show the live measurements from each location. All three temperatures showed a single value and did not vary with each iteration of the program as expected. This indicated that there was an issue with the signal being transmitted to the FieldPoint devices and read in the program. The bottoms thermocouple is located partially submerged into the bottom of the column so that it is in direct contact with the fluid being heated. The thermocouple itself was replaced in hopes of achieving a live signal; however, the new thermocouple still did not provide a live measurement of the fluid. When examining the ambient temperature thermocouple, a live measurement was obtained when the thermocouple was placed in a specific orientation, indicating the issue was in the wires where there was a restriction on the signal, likely due to wear and tear. After this was discovered, it was believed that the tops and bottoms temperatures were not being read due to a fault within the wires, somewhere between the column location and the FieldPoint device. Unfortunately, the temperatures were unable to be fixed within the time frame of this project and work continues to be performed to find a solution.

B. LabVIEW Programming Upgrades

1. Block Diagram Architecture

As identified in Section II: Existing Setup and Design, there were many opportunities for improvement in the LabVIEW code used to startup, maintain, and shutdown the distillation column. One of the most impactful changes made to the distillation column software is in the overhaul of the primary architecture of the code. Rather than use stacked sequences and sub-stacked sequences in combination with some case structures, the overall architecture was converted into a state machine. The new architecture provides the baseline for further improvements to be made to address the deficiencies of the previous architecture. State machines, by design, are easy to maintain and can be reprogrammed with updated functionality simply by adding appropriate states to handle the new logic.

In the first draft of transitioning the stacked sequence into a state machine, a complete state machine replaced each existing stacked sequence structure. This resulted in multiple embedded state machines, including a while loop for each. Incorporating the while loop around each embedded case structure was problematic because it caused blocking, preventing the program from being able to process any other data until a particular state was complete. Blocking was one of the main issues within the stacked sequence code which is why it is necessary to eliminate blocking in the state machine version.

In order to eliminate blocking, the while loops were removed. The while loops served as a way to maintain the program in a particular state until a desired condition was met which would terminate the loop. If the while loop is removed, there must be another way to monitor for a specific condition to be met before progressing to the next state. One such function that can perform this is the Case Structure. The same Boolean value that was used for the conditional terminal in the while loop was instead wired to a case selector. This resulted in two possible cases, true or false. If the Boolean is true and the condition has been met, the true case will contain the logic that progresses the program to the next state. If the Boolean is false, the false case will execute instead. The false case is typically empty which allows for the program to remain in the same state until the desired condition is met. An example of a case structure used to replace a while loop is in FIGURE 20, where both cases are shown. If the bottom level is greater than or equal to the setpoint, the true case is selected which updates the startup enum with the next case to occur, thus progressing the program to the next state. If the level is not at the setpoint, the false case occurs, and the program remains in the current state.

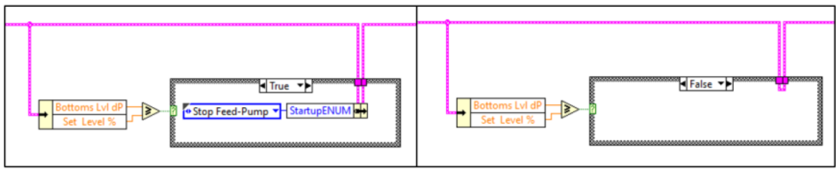
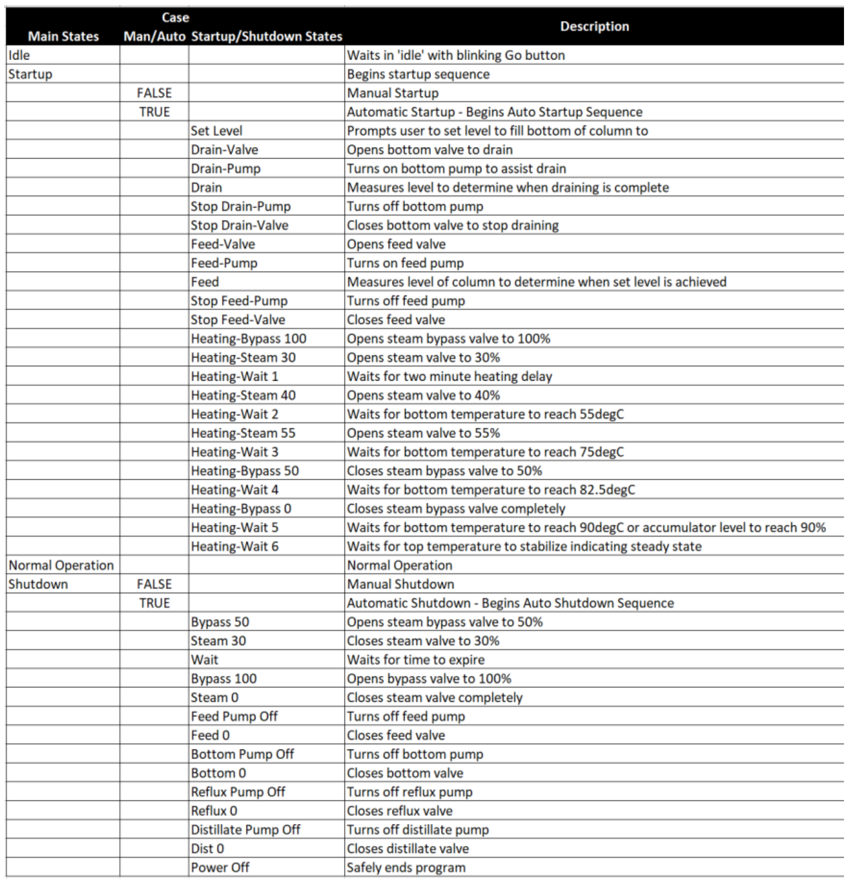


FIGURE 20: Example of True and False Cases of the LabVIEW Case Structure Used to Replace While Loops

Case structures were used to replace each while loop except for the primary while loop for the overall state machine structure. This loop was retained in order to allow for the state machine to move between states as desire so that the program will not terminate until the shutdown process has been completed. At this point in the development process, there were still multiple embedded case structures. Specifically, within the startup case, there were sub structures to handle a feed sequence and heating sequence which were not necessary to separate into sub-case structures.

To remedy this and clean up the structure, all of the startup steps were consolidated into a single startup case structure with an enum used to define each case. The enum allows for cases to have a relevant name for their purpose and provides an overview of the case structure without scrolling through each case. The purposeful names also assist in organization of the case structure and make it easier for the user to move the program between various states. Three enums were used in total. They include the main enum which moves the program between startup, normal operation, and shutdown, the startup enum, and the shutdown enum. The startup and shutdown enums move the program through each step of startup or shutdown when being performed in automatic. The overall state machine architecture can be seen in FIGURE 21 which is comparable to FIGURE 8 for the stacked sequence structure.



	Case		
Main States	Man/Auto	Startup/Shutdown States	Description
Idle			Waits in 'idle' with blinking Go button
Startup			Begins startup sequence
	FALSE		Manual Startup
	TRUE		Automatic Startup - Begins Auto Startup Sequence
		Set Level	Prompts user to set level to fill bottom of column to
		Drain-Valve	Opens bottom valve to drain
		Drain-Pump	Turns on bottom pump to assist drain
		Drain	Measures level to determine when draining is complete
		Stop Drain-Pump	Turns off bottom pump
		Stop Drain-Valve	Closes bottom valve to stop draining
		Feed-Valve	Opens feed valve
		Feed-Pump	Turns on feed pump
		Feed	Measures level of column to determine when set level is achieved
		Stop Feed-Pump	Turns off feed pump
		Stop Feed-Valve	Closes feed valve
		Heating-Bypass 100	Opens steam bypass valve to 100%
		Heating-Steam 30	Opens steam valve to 30%
		Heating-Wait 1	Waits for two minute heating delay
		Heating-Steam 40	Opens steam valve to 40%
		Heating-Wait 2	Waits for bottom temperature to reach 55degC
		Heating-Steam 55	Opens steam valve to 55%
		Heating-Wait 3	Waits for bottom temperature to reach 75degC
		Heating-Bypass 50	Closes steam bypass valve to 50%
		Heating-Wait 4	Waits for bottom temperature to reach 82.5degC
		Heating-Bypass 0	Closes steam bypass valve completely
		Heating-Wait 5	Waits for bottom temperature to reach 90degC or accumulator level to reach 90%
		Heating-Wait 6	Waits for top temperature to stabilize indicating steady state
Normal Operation			Normal Operation
Shutdown	FALSE		Manual Shutdown
	TRUE		Automatic Shutdown - Begins Auto Shutdown Sequence
		Bypass 50	Opens steam bypass valve to 50%
		Steam 30	Closes steam valve to 30%
		Wait	Waits for time to expire
		Bypass 100	Opens bypass valve to 100%
		Steam 0	Closes steam valve completely
		Feed Pump Off	Turns off feed pump
		Feed 0	Closes feed valve
		Bottom Pump Off	Turns off bottom pump
		Bottom 0	Closes bottom valve
		Reflux Pump Off	Turns off reflux pump
		Reflux 0	Closes reflux valve
		Distillate Pump Off	Turns off distillate pump
		Dist 0	Closes distillate valve
		Power Off	Safely ends program

FIGURE 21: Summary of Updated State Machine LabVIEW Code Architecture

This structure is simplified as there are only three levels rather than four. The first level dictates which main state the program is in, either startup, normal operation, or shutdown. The second level determines if the operation will be performed in automatic or manual and this only applies to startup or shutdown. The final level is the startup and shutdown sequences that occur when in automatic mode. FIGURE 22 demonstrates how the layering of case structures appears in LabVIEW and provides a visual of the case structure architecture used. This figure specifically shows one of the steps performed within the automatic startup of the distillation column. While the cases are nested, the enum state values control the transition into and out of each, never remaining for longer than the code in each frame takes to execute (no blocking).

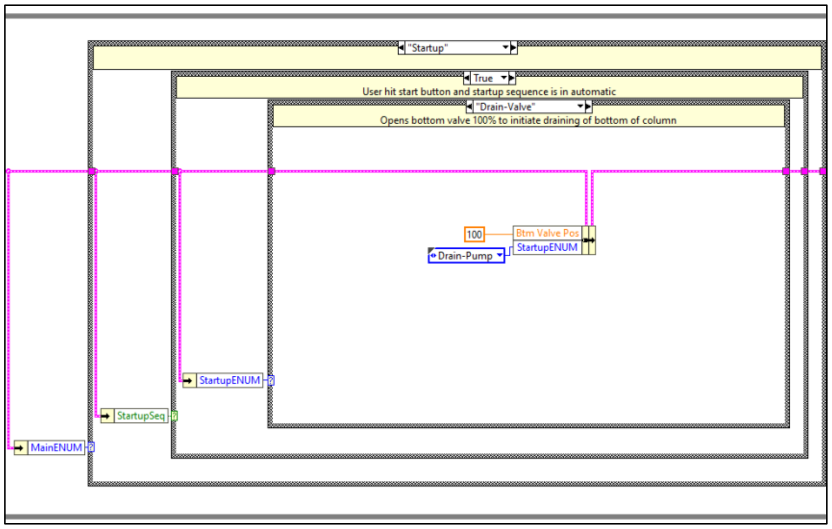

"Startup"
True
User hit start button and startup sequence is in automatic
"Drain-Valve"
Opens bottom valve 100% to initiate draining of bottom of column
100
Btm Valve Pos
StartupENUM
Drain-Pump
StartupENUM
StartupSeq
MainENUM

FIGURE 22: Example of LabVIEW Case Structure Layering for the Startup Sequence

Another major change made to the program is the use of a Master Data Cluster. In LabVIEW, a Cluster is a custom combination of different wire types all bundled into a single wire (like a wiring harness). The Master Cluster therefore provides a mechanism for a single wire passed through the state machine that contains all necessary system data, such as delays, pump settings, valve positions, and comments, all bound within the Master cluster including associated integers, Booleans (True/False), and strings,. This data is represented by a thick pink wire and is passed around a shift register on the main while loop so that it constantly updates with each iteration and contains the most recent values obtained by the system. The Master Cluster is initialized with a startup value for each individual system datum used within the code. These are represented in cluster constants sorted in logical groups. The cluster constants are then bundled together to create the Master Cluster as seen in FIGURE 23. In this FIGURE, the cluster constants are condensed to reduce clutter. They can be expanded by double clicking on the icon. The expanded version of each cluster constant can be seen in APPENDIX IV.

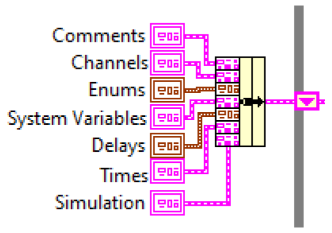


FIGURE 23: Initialization of LabVIEW Master Cluster for Distillation Column Software

Individual data can be separated from the Master Cluster using the LabVIEW "Unbundle by Name" feature so that a single datapoint can be obtained within a desired state. This is performed within the states as well as at the end of the overall while loop iteration so that corresponding indicators on the front panel can be updated with live values. FIGURE 24 shows the code that allows for indicators to be updated. The thick pink wire is the Master Cluster and each variable is separated using the Unbundle by Name function. From there, the value for each system variable that has been unbundled from the Master Cluster is then wired into the corresponding indicator, updating it with the value from the most recent iteration of the while loop. The Master Cluster data is then passed through the shift register and used for the next loop iteration. The use of the Master Cluster and the location of the indicators eliminate the need for local variables. This resolves the issue of data flow seen in the stacked sequence structure as it is easy to follow the flow of data and there are no race conditions that can occur.

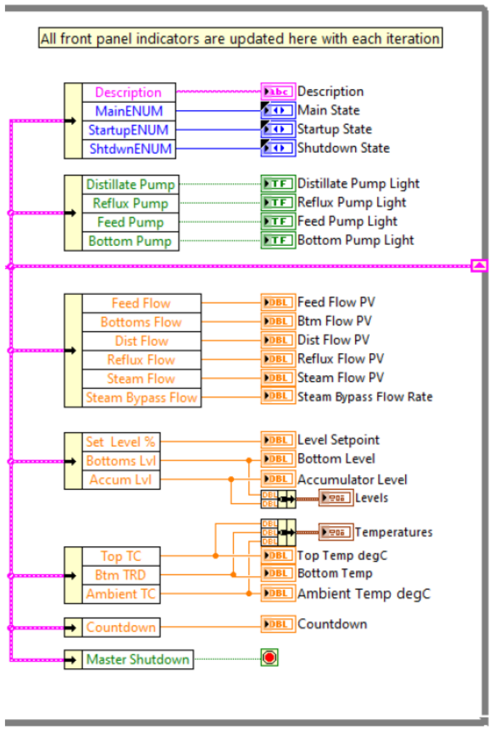


FIGURE 24: LabVIEW Master Cluster Unbundled by Name to Update Front Panel Indicators and Conditional Terminal

Likewise, a system variable from the Master Cluster can be updated to a new value within a state using the LabVIEW “Bundle by Name” feature. This allows for a single variable to be isolated from the cluster and a new value wired into the variable. The variable is then updated, and the new version of the Master Cluster is passed through the rest of the code. FIGURE 25 is an example of the Bundle by Name feature used in the distillation column code. In this example, the bottom pump system variable is being updated to a true value which will turn on the pump. The startup enum is also being updated with the next state the system will move to once the current one is complete.

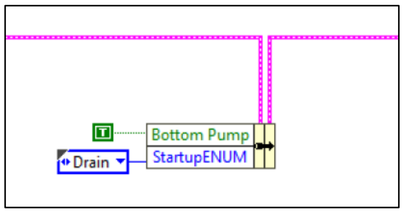
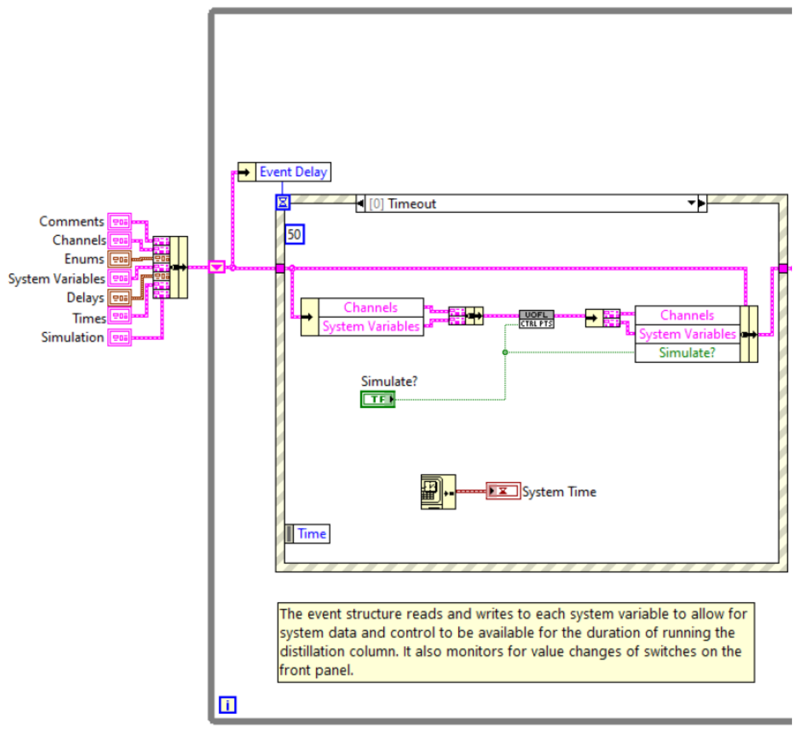


FIGURE 25: LabVIEW Example of Bundle by Name Node Used to Update the Master Cluster

The addition of a Master Cluster was necessary in order to have system data available to the operator during all states of the program. In the stacked sequence version, certain data was only available during specific frames of the sequence and the use of while loops blocked other data from being updated. The Master Cluster allows for the most-recent system data to be available to the operator during all states of the program. This layout also makes it easier for future updates to occur such as a creating a new indicator for the operator to view or adding a new variable to the Master Cluster.

Another feature that would not be possible without the use of a Master Cluster is the event structure pictured in FIGURE 26. The purpose of the event structure is to update system variables as well as check for control value changes made by the operator. The event structure has various events that it monitors for as well as a timeout event that occurs if no other event occurs within a designated time set by the Event Delay. For this structure, the Timeout event is used to read from or write to each system variable including pumps, temperatures, valve positions, and differential pressures. In order to include every pump and valve setting, and temperature and differential pressure measurement, a for loop surrounding a case structure is necessary. The iteration terminal is wired to the case selector so that at each iteration, the corresponding case is executed. This ensures that every case is executed once each time the for loop operates and once the for loop is complete, the program moves on to perform the logic written in the case structure.



The event structure reads and writes to each system variable to allow for system data and control to be available for the duration of running the distillation column. It also monitors for value changes of switches on the front panel.

FIGURE 26: LabVIEW Event Structure Used to Read and Write to System Variables and Monitor for Control Value Changes

Within the Timeout event, there is a Control Points subVI where data is either being read from or written to the FieldPoint channels. Data being read includes temperatures and differential pressures. This occurs within the subVI as pictured in FIGURE 27. A FieldPoint channel is isolated using Unbundle by Name and the data is wired to a FieldPoint Read subVI that reads the signal from the channel. From there, the data is wired to a conversion subVI in order to convert the value from a current signal to the relevant units of the variable such as degrees Celsius or level percentage. Once the value is converted to the appropriate units, the value is then used to update the corresponding system variable as part of the Master Cluster. The updated Master Cluster is then used within the rest of the code as needed for the logic performed.

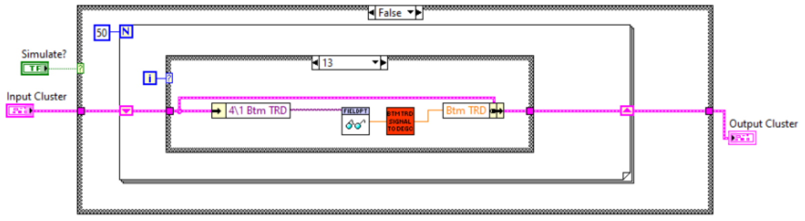


FIGURE 27: LabVIEW Control Points SubVI Used to Read and Write to FieldPoint Channels

Data that is being written within the Timeout event subVI include pump settings and valve positions. When writing to a variable, the system variable that was updated in a different part of the code is isolated from the Master Cluster using Unbundle by Name. For a valve position, the value of the variable is wired to a conversion subVI that converts it from a percentage to a voltage signal. The new voltage signal is then wired to the FieldPoint Write subVI which is wired to the corresponding FieldPoint channel to be written to. For a pump, the value is Boolean as the pump is either off or on. Therefore, no signal conversion is necessary, and the system variable can be wired directly to the FieldPoint Write subVI for Boolean values.

An additional feature included in the Control Points subVI is the ability to run the code in a simulation mode. This is especially useful to test the logic of the code and ensure that the system progresses to each state as designed. A Boolean control is used to place the program in simulation mode. If the control reads false, the false case of the Control Points subVI is executed which reads and writes to FieldPoint channels as described. If the control reads true, the program is in simulation mode and instead of reading and writing to FieldPoint channels, network variables are used. An example of the simulation case is in FIGURE 28. This can be compared to FIGURE 27 in that it deals with the same variable. The difference is that in FIGURE 28, a network variable is used instead of the FieldPoint channel in order to simulate the temperature rather than obtain a real-time measurement.

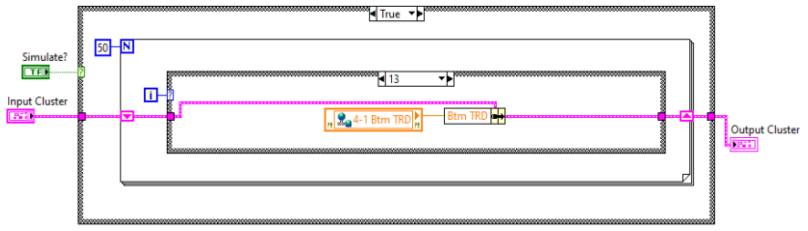


FIGURE 28: LabVIEW Control Points SubVI in Simulation Mode to Read and Write to Network Variables

The read and write operations are performed with each iteration of the Timeout event so that temperatures, levels, and flowrates are always being updated to provide the operator a live view of the process. The Timeout event also ensures that valve positions and pumps are able to be manipulated as needed for the duration of running the distillation column. This is a major improvement from the original stacked sequence code. Limitations existed within that code that prevented valves and pumps from being manipulated by the operator unless the system was specifically in the normal operation frame of the stacked sequence. The updated program now provides a live view of the system from startup to shutdown and is inherently safer as the operator is no longer running the distillation column blindly when operating in automatic mode.

Besides performing read and write operations to FieldPoint channels, the event structure also monitors for changes in controls performed by the operator when interacting with the front panel. There are controls to change valve positions and switches to turn on pumps or alternate between manual and automatic, as well as the Go and Shutdown buttons. The corresponding control nodes on the block diagram exist within the event structure and monitor for a value change on the control to occur. When the program notices a value change, the new value is wired into the system variable and the updated value is used elsewhere in the code where the resulting change is affected. An example of this event is pictured in FIGURE 29. In this figure, the program is monitoring for the Startup Sequence Boolean switch to change values, indicating that the user has specified they want an automatic startup. When a value change is captured, the new value is used to update the system variable and used later in the code to determine which case is applied for startup, manual, or automatic mode.

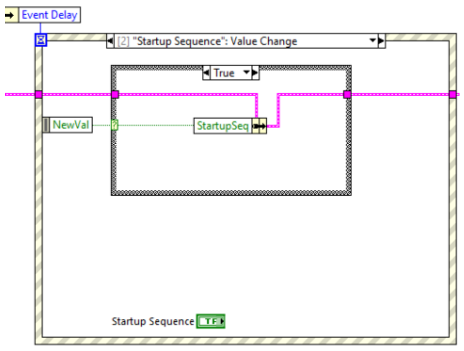


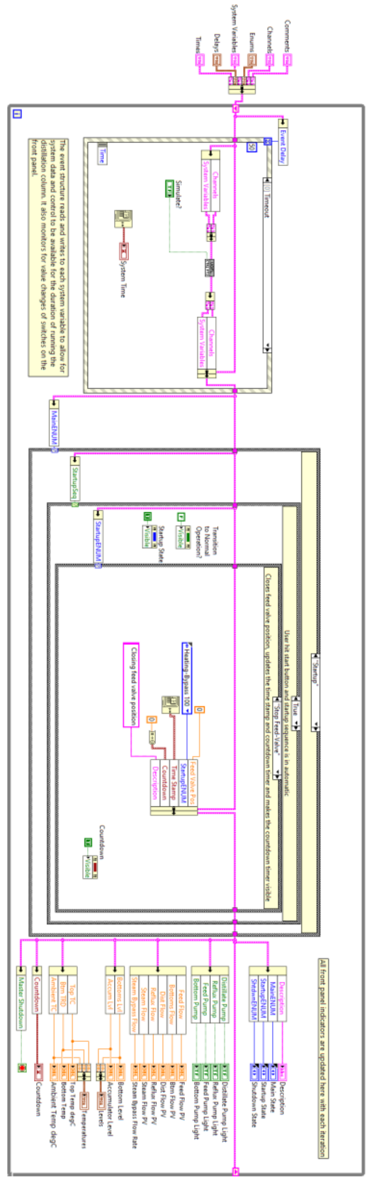
FIGURE 29: Example of LabVIEW Event Structure Monitoring for Controller Value Change

The event structure serves the purpose of updating valve positions and pump settings and reading and converting temperature and differential pressures. It monitors and interacts with the equipment on the distillation column and is performed with each iteration of the primary while loop. This allows for solely the logic of the code to exist within the case structures of the state machine which simplifies the overall architecture of the program. By using the case structures for only logic operations that dictate when each step is performed and for how long each state is maintained, the structure is simplified and condensed with only the most relevant operations. The consistency of the organization also makes the code more intuitive to a new programmer so that it would be much easier to add a new step to startup or shutdown if needed.

To summarize, the Master Cluster is first initialized with values for all necessary data including the FieldPoint channel names, corresponding system variables, time stamps, delays, state names (via enums), and comments. The data within the Master Cluster then passes through the event structure which monitors for changes in any of the controls on the front panel. If there are no control changes within a certain amount of time, the Timeout event is performed. This event updates the Master Cluster with temperature and differential pressure readings and writes the values of system variables to the corresponding FieldPoint channels or network variables. Once every case within the Timeout event is performed, the data in the updated Master Cluster then move onto the logic part of the program which takes place within the embedded case structures of the state machine. Based on what value the enum is set to in the Master Cluster, that specific case is performed within the state machine.

Within each case, logic is performed that results in updating the Master Cluster. Once all operations within the case are executed, the data in the updated Master Cluster exits the state machine. Just before the iteration of the loop is finished, all indicator values are unbundled from the Master Cluster and used to update the indicators seen by the operator on the front panel. After all indicators have been updated, the iteration of the while loop is complete and the data within the Master Cluster is passed around to the beginning of the while loop for the next iteration with updated variables from the logic performed from the previous iteration. Each iteration is performed quickly within milliseconds and in succession so that a real time view of the process variables is available to the operator throughout all states of the program.

A sample of each case, event, and subVI used within the program are documented in APPENDIX IV. Overall, there is a clear and defined flow of data through the updated program. FIGURE 30 captures the final result of the entire architecture used to control the distillation column. FIGURE 21 can be used as a guide in understanding the order and overall progression of each state.



The event structure reads and writes to each system variable to allow for system data and control to be available for the duration of running the distillation column. It also monitors for value changes of switches on the front panel.

All front panel indicators are updated here with each iteration

FIGURE 30: Overall LabVIEW State Machine Architecture to Run the Distillation Column

2. <u>SubVIs</u>

In addition to the new organization of logic, the use of subVIs further simplifies the code and overall structure. Signal conversions are repeated throughout the code and the math is repeated for each signal conversion. For example, each instance a valve position is updated within the code, the input must be converted to a voltage signal in order to communicate with the FieldPoint channel. The same math is used for each valve and is therefore repeated throughout the code, taking up space on the block diagram and adding to already tangled code. To eliminate this problem, a subVI was created for each type of conversion performed and labeled as such. The subVI will accept the input and convert it to the appropriate signal or vice versa depending on the channel, and the math will all be performed within the same node.

FIGURE 31 is an example of a conversion subVI and how it is wired. FIGURE 32 is the code executed within the subVI. In this example, the steam valve position value is wired into the subVI as seen in FIGURE 31. The value becomes the Valve Position % variable in the subVI code in FIGURE 32 and the conversion math is performed to convert the percentage to a voltage signal. The value that would show up in the Voltage Signal (1-9V) indicator is the same value that exits the subVI in FIGURE 31 and is wired into the FieldPoint subVI to communicate with the steam valve position FieldPoint channel and move the valve to a new position. This example demonstrates how subVIs can condense the code used in the main VI and makes it easier to interpret as well as clean up the block diagram. It is also important to note that with the separation of logic into the case structures of the state machine, the conversion subVIs are only located within the event structure since that is the only location in the code where the signal needs to be converted. This reduces the amount of times the subVI needs to be used. Additionally, the subVI conserves space and allows for the event structure to be minimally sized. A secondary benefit to using a subVI for repeating code is that if the conversion for a valve position signal were to change, only the subVI would need to be modified. Each use of the subVI within the main VI will automatically update with the new subVI code which saves significant time in reprogramming the system.

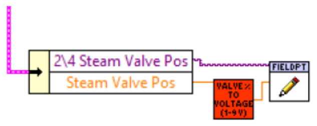


FIGURE 32: Example of LabVIEW SubVI Used to Convert Valve Position Percentage to Voltage Signal

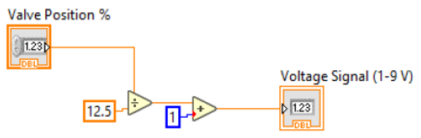


FIGURE 31: LabVIEW Code Executed Within Valve% to Voltage (1-9V) SubVI

Another location where subVIs simplified the system is during the normal operation state. In the original stacked sequence structure, it was the normal operation frame that took up the most area on the block diagram, thus forcing the entire stacked sequence structure to be just as large. For reference, it would require roughly six computer screens in a 3x2 layout to view the entirety of the code in once glance since there is no zoom function within LabVIEW. This made the block diagram difficult to work with and interpret since the code was already unorganized. It also required the programmer to trace wires while scrolling through the diagram to understand how everything was connected. The best method to simplify this state in the upgraded code was to use subVIs. The normal operation frame was already relatively divided into six subgroups related to general areas of the distillation column such as feed and distillate. This made it easy to utilize the divisions and create subVIs for each group. The sixth group of miscellaneous operations was able to be logically divided across the other five groups, resulting in five subVIs created to perform the normal operation of the distillation column.

An example of the normal operation subVI controlling the bottom level can be seen in FIGURE 33. All necessary input variables are pulled from the Master Cluster and wired to the subVI where any operations and logic are performed. FIGURE 34 is the code performed within the bottom level subVI when the bottom level is operated in automatic. The inputs and outputs of the subVI seen in FIGURE 34 correspond with the inputs and outputs pulled from the Master Cluster in FIGURE 33. The resulting output of the subVI is used to update the Master Cluster which is then passed through to update the necessary indicators and used for the next iteration of the while loop. A for loop is used to rotate once through each case which houses each of the five subVIs. This allows for the entire program to run continuously during normal operation so that system data is still constantly read or updated, and the operator can retain full control over the process.

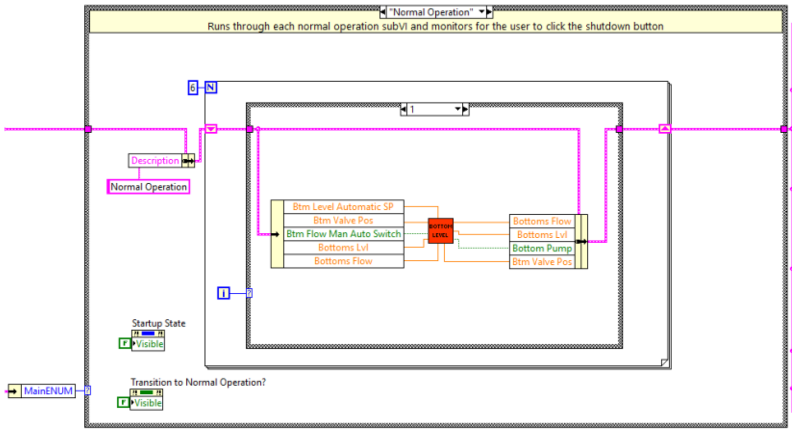


FIGURE 33: LabVIEW Normal Operation SubVI Controlling the Bottom Level

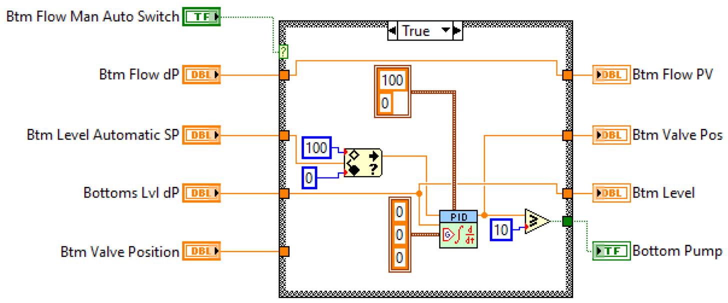


FIGURE 34: LabVIEW Code Executed Within the Bottom Level SubVI Used in Automatic Mode of Normal Operation

3. User Input Prompt

Further functionality was built into the code with the use of the Prompt User for Input subVI. In the original stacked sequence startup procedure, the bottom of the column was filled to 50% level every time. However, when starting the column up manually, it became best practice for the operator to increase the level the column is filled to in order to make up for methanol losses from the system due to evaporation from unsealed vessels. One way to incorporate this functionality into the new startup code could be to use a controller which would rely on the operator to remember to set a value within the desired range prior to pressing 'Go'. This method leaves room for error if there is a new operator who does not have the knowledge from experience to know to set the level prior to initiating startup. Therefore, it was decided that a better method would be to use a "Prompt User for Input" subVI as pictured in FIGURE 35.

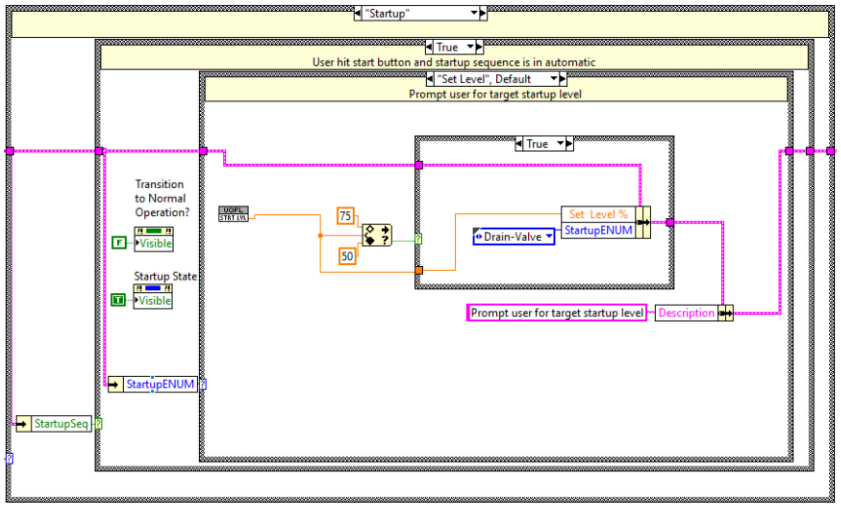
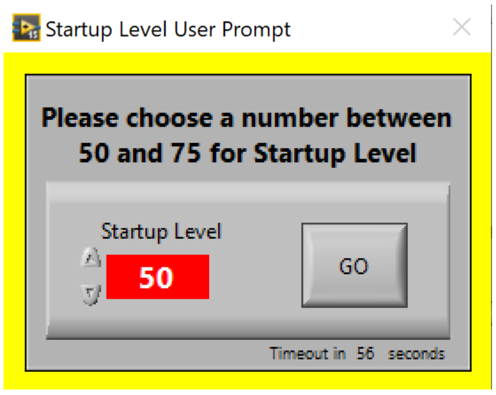

"Startup"
True
User hit start button and startup sequence is in automatic
"Set Level", Default
Prompt user for target startup level
True
Transition to Normal Operation?
Visible
Startup State
Visible
75
50
Set Level %
StartupENUM
Drain-Valve
Prompt user for target startup level
Description
StartupENUM
StartupSeq

FIGURE 35: LabVIEW Case to Call SubVI to Prompt User for Startup Level Input

When this subVI is executed in the program, a dialog box pops up for the operator and requests a value to set the level of the column to during startup. The window informs the user of the appropriate range to choose within and has a default level of 50% that will execute if the operator does not input a value within 60 seconds. The way the pop-up will appear to the operator is pictured in FIGURE 36.



Please choose a number between 50 and 75 for Startup Level

FIGURE 36: LabVIEW Prompt User for Startup Level Input Dialog Box

An additional case structure was added to handle the response should the user input a value outside of the specified range. If the input value is within the range, the program continues to the next state and accepts the level setpoint. If the input value is outside of the range, false case executes, and the program moves to an error state on the main case structure. When this occurs, another dialog box pops up for the operator that informs of an error. This can be seen in FIGURE 37. Once the operator acknowledges the error message, the program reverts to the Set Level startup state and once again prompts the user to input a level.

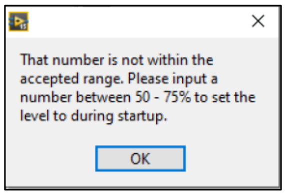

That number is not within the accepted range. Please input a number between 50 - 75% to set the level to during startup.

FIGURE 37: LabVIEW Error Dialog Box for Out of Range Level Setpoint

Prompting the user to set a level to fill the column to is the first case to occur in the automatic startup case structure. The timing of the prompt ensures that the level will be set prior to performing any other startup steps and reinforces a correct order of operation. The nature of using a pop-up dialog box forces the operator to pause and think through what an appropriate level will be based on the estimated methanol content in the feed. These features together make the Prompt User for Input subVI better suited for the scenario compared to a simple control.

4. Pump and Valve Manipulation Sequencing

One of the issues with the stacked sequence code that had risk of damaging the equipment was the way in which the logic was performed in order to open or close a valve and turn on or off the corresponding pump. The way in which the original code was written, there was no way to know which operation would occur first and how long of a delay there would be between each action. This caused risk for running a pump dry should a pump be turned on before any flow was established from opening a valve. To address this issue, the manipulation of a valve and corresponding pump were separated into two different case structures as pictured in FIGUREs 38 and 39. This establishes a set order in which the operations take place. For each instance when a valve was opened and a corresponding pump was turned on, the valve is opened first and in the following case, the pump is turned on. This ensures that a flow has been established prior to turning on a pump and will assist in preventing the pump from running dry. In FIGURE 38, the case executed is Drain-Valve. In this case, the bottom valve position is being set to 100% and the next case to be performed is set to Drain-Pump. In the Drain-Pump case in FIGURE 39, the pump system variable is set to true which will turn on the pump when the system variable is addressed in the event structure. The reverse order of operations occurs for closing a valve and turning off a pump. The pump is turned off first and then in the following case, the valve is closed. Again, this prevents a pump from operating without a liquid flow and reduces the risk of running the pump dry.

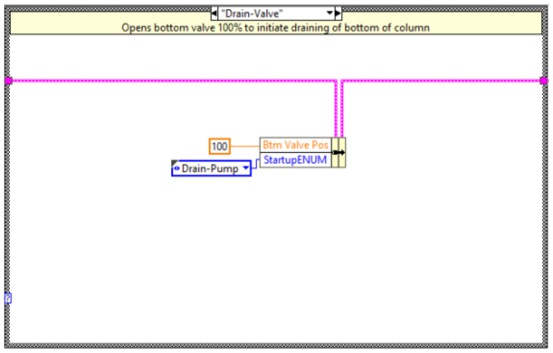

"Drain-Valve"
Opens bottom valve 100% to initiate draining of bottom of column
100
Btm Valve Pos
StartupENUM
Drain-Pump

FIGURE 38: LabVIEW Startup State to Open the Bottom Valve to 100%

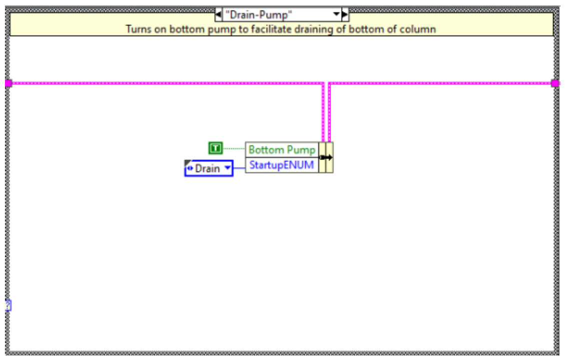

"Drain-Pump"
Turns on bottom pump to facilitate draining of bottom of column
T
Drain
Bottom Pump
StartupENUM

FIGURE 39: LabVIEW Startup State to Turn on the Bottom Pump

5. <u>Math Simplification</u>

Complicated mathematical operations were another issue seen in the stacked sequence structure. Specifically, this was present in the heating frames where running averages were being calculated as part of the logic to determine when the system met the specified heating limit. A before and after comparison is pictured in FIGURE 40 to demonstrate the same functionality performed in two different ways. The highlighted sections of the code perform the same operation but using different function nodes. In the updated version, a premade LabVIEW subVI was used. This subVI is programmed to take an average of a specific sample length and can replace all of the operations highlighted on the left side of FIGURE 40. The other math performed after the average is obtained in the original code is a signal conversion which is handled within the event structure of the updated code. This further cleans up the program which saves space and makes the logic easier to follow for future updates. The same function was used everywhere a running average of a temperature was obtained. A similar, simplified function exists for calculating standard deviation as well, which was used for measuring the variation of the top temperature in order to determine when the temperature has stabilized. Altogether, the use of this subVI saved substantial space on the block diagram and enhances the user's ability to interpret the code without being distracted or confused by complicated mathematical operations.

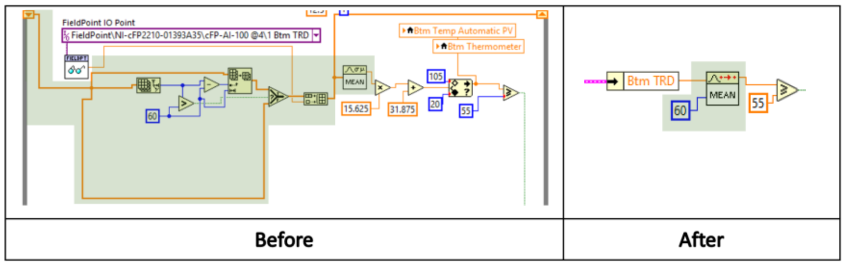


FIGURE 40: Comparison of LabVIEW Code Simplification to Perform the Same Operation

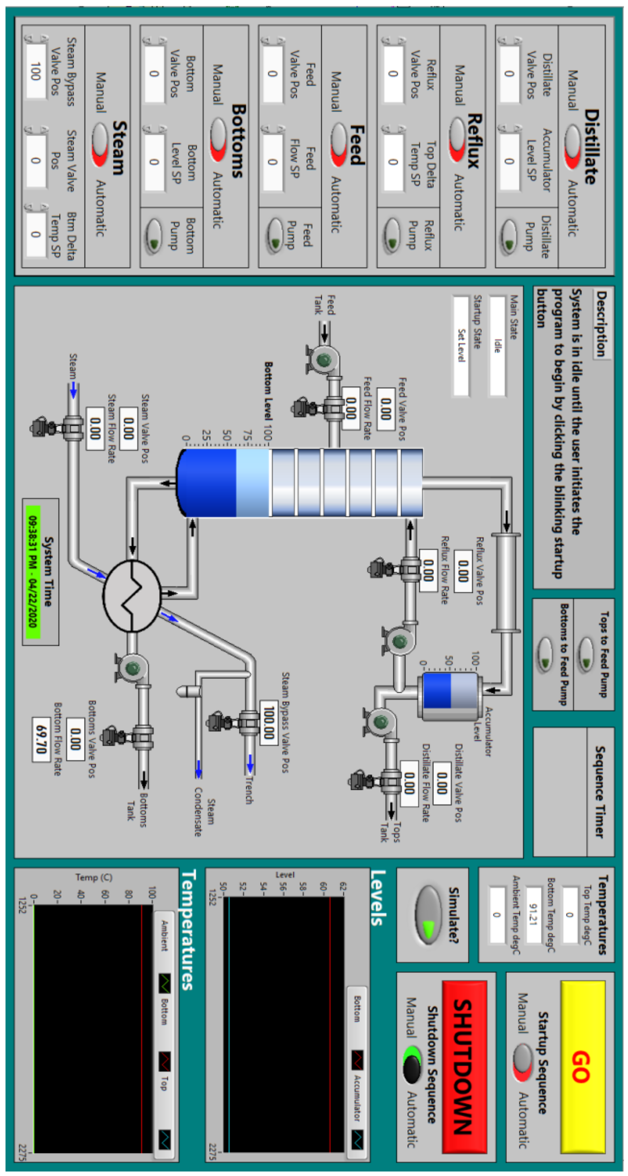
6. <u>Accessible Shutdown</u>

Another benefit to the new architecture used is the increased functionality of the shutdown button. In the stacked sequence structure, the shutdown button would only work when the system was in normal operation. Therefore, if a failure were to occur during an automatic startup, the operator had no safe way to discontinue startup and terminate the program. With the use of the event structure, the Master Cluster, and the elimination of blocking, that is no longer an issue. Since there is no blocking, the system is never stuck in a specific state preventing other operations from occurring. This allows for the event structure to constantly monitor for a value change in the Shutdown button, indicating that the user has pressed the button to move the system into the shutdown sequence. When a value change is detected, the corresponding variable is updated within the Master Cluster and carried through the program to initiate a transition to the shutdown state. Therefore, if the system experiences a failure during startup or the operator decides to terminate the startup, the program is able to perform a specific shutdown procedure and ensure that valves and pumps are placed in a safe state before terminating the program.

7. Front Panel

In addition to upgrades made to the block diagram to simplify and further improve code, the front panel was also updated. FIGURE 41 is an image of the new front panel layout. Changes were made with the overall layout, graphics, charts, available indicators, and other small details. The goal was to create a more user-friendly interface for the operator to interact with and minimize unnecessary data. For reference and comparison, the original front panel is pictured in FIGURE 19.

For the overall layout, one of the main issues was making it more apparent what control or switch is for each part of the process. To better distinguish the controls, boxes were used to group controls that regulated the same region of the distillation column. Pump controls were added to the front panel as well and placed in the appropriate groups. The groups were then given bold headings so that the operator can identify what region of the column each group is controlling within a quick glance. The control groups were organized vertically down the left side of the screen to mirror the vertical column. Each control group loosely corresponds to the relative position on the column so that it is intuitive for the operator when trying to locate a specific controller. Clean lines and negative space were utilized in order to give the program a more streamlined look and reduce clutter.



System is in idle until the user initiates the program to begin by clicking the blinking startup button

Distillate Valve Pos	0
Accumulator Level SP	0
Reflux Valve Pos	0
Top Delta Temp SP	0
Feed Valve Pos	0
Feed Flow SP	0
Bottom Valve Pos	0
Bottom Level SP	0
Steam Bypass Valve Pos	100
Steam Valve Pos	0
Btm Delta Temp SP	0
Feed Valve Pos	0.00
Feed Flow Rate	0.00
Steam Valve Pos	0.00
Steam Flow Rate	0.00
Reflux Valve Pos	0.00
Reflux Flow Rate	0.00
Distillate Valve Pos	0.00
Distillate Flow Rate	0.00
Steam Bypass Valve Pos	100.00
Bottoms Valve Pos	0.00
Bottom Flow Rate	69.70
System Time	09:38:31 PM 04/22/2020
Top Temp degC	0
Bottom Temp degC	91.21
Ambient Temp degC	0

FIGURE 41: Updated LabVIEW Front Panel for Distillation Column Software

Indicators corresponding to each controller group only displayed flow rate information calculated from a differential pressure. On the old front panel, there were two indicators for each flow rate, one would show the flow rate when the system was in manual and the other would show the flow rate when the system was in automatic. This was likely a work around to avoid local variables in the original code because the stacked sequence structure restricted the use of the same indicator across different frames. Since the updated code uses a Master Cluster which allows for system data to be available to the operator throughout the entire process, only one flow rate indicator was necessary. Therefore, the redundant indicator was removed. This significantly reduced clutter and allowed for the flow rate indicators to be moved next to the corresponding process lines on the diagram. The location of the flow rate indicators will help students visually connect variable names to their physical location within the distillation column process and enhance their understanding of the movement of fluid throughout the distillation column. Additionally, an indicator for valve position was added to the diagram to allow students to compare valve position with measured flow rate and understand the relationship between the two values.

In addition to redundant flow rate indicators being removed, the thermometer indicators for the top and bottom temperature were also removed. Instead, the top, bottom, and ambient temperatures are represented digitally and grouped together for organization purposes. While the thermometers are an interesting and unique way to represent the data, it is not the most effective way to determine the exact temperature of the system. In one of the heating sequence steps of startup, the program is monitoring for the bottoms temperature to reach 82.5 degrees Celsius. This is a relatively precise value and is difficult to identify when looking at a thermometer graphic compared to a digital output. Additionally, there was no indicator to display the ambient temperature which is important for estimating external thermal impacts on the column that could affect steady state.

The graphics were another aspect of the front panel to be updated. These modifications act as a clean-up from the previous front panel and make the graphics more visually appealing and realistic to the process. For the distillation column itself, Microsoft PowerPoint was used to design the graphic. The goal was to give the visual a glass-like shading to accurately represent the material of construction of the actual column. Six trays were added for consistency with the actual distillation column. Having six trays also allows for the feed placement to be accurate relative to the trays. The process lines and valves were replaced with realistic three-dimensional graphics using the LabVIEW Datalogging and Supervisory Control (DSC) Module which is add-on software that enhances the user interface, and particularly suited for creating P&ID (Piping and Instrumentation Diagram) visuals. Additionally, the steam bypass valve was added to the diagram since it is a relevant part of the system, specifically for the heating sequence during startup. Pump graphics were also added to their respective locations with an indicator on each pump that lights up when the pump is on. The combination of the pump and flow rate indicators will provide the operator with a better understanding of where fluid is flowing throughout the column at a quick glance.

Graphic adjustments were also made for the bottom of the column and accumulator levels. Since the bottom of the column is never drained to be completely empty, it is physically accurate for a slight level to remain even when the differential pressure indicates a level of zero. The accumulator level was given a vessel body which is a more accurate representation of the accumulator since it is a distinct cylindrical vessel. The accumulator graphics also bring more attention to the accumulator level on the screen which is beneficial since it is necessary to maintain a fluid level to prevent pump damage.

Another update made to the front panel is centered around the charts. Previously, there were three charts and not enough room for all three to be visible at the same time. After reviewing the function of each chart, it was determined that the Feed chart was not necessary to the distillation column operation. Based on the way the distillation column is currently used in the Unit Operations lab, there is no need to trend the feed flow rate and feed valve position. The data is not needed by students performing the lab or for the operator controlling the column. Therefore, the Feed chart was removed. In general, it was determined that there was no need for valve positions to be trended in any of the charts. From the operator’s perspective in how the distillation column is currently used, the operator only needs to understand the current valve position, not the history of previous valve positions. As a result, the valve positions were removed from the Levels and Temperatures charts.

The Levels and Temperatures charts were retained as they are relevant system data that the operator should have available. Temperatures are necessary because the process of distillation is driven by the temperature gradient throughout the column. The column must be slowly heated during startup to prevent shocking the glass and the stability of the temperature overtime indicates if the system is in steady state. Since both temperature changes are time dependent, it is beneficial to have a chart constantly updating with real time data. This provides the operator with a visual of the rate of change and can assist the operator in determining if the system is in a steady state or not. Levels are also another important variable to trend. It is important for the operator to understand how the levels are changing over time. Having a trend available will inform the operator whether the levels are gradually increasing, decreasing, or staying the same. By understanding the trend of the levels, the operator can predict where the levels will be after a certain period of time and can make flow rate adjustments as needed to maintain a desired level. Should levels not be maintained, there becomes a risk of equipment damage due to running the pump dry and overheating the bottom of the column.

One of the minor changes that may have a larger impact is the use of color on the Boolean toggles to switch between manual and automatic operation. When the switch is set to manual, the negative area of the switch is colored red. When the switch is set to automatic, the negative area of the switch is colored green. This small visual detail will assist in quick recognition if a system is in manual or automatic and is a clear indication when the switch changes positions. The colors also coordinate with the true/false value of the switch. Green corresponds to true and red corresponds to false. When the switch is set to true, the switch shows green and the true case for the startup case structure is executed for an automatic startup. When the switch is set to false, the switch shows red and the false case is executed for a manual startup.

Another minor change to the front panel is the addition of the current time as well as a countdown timer which is necessary for the waiting periods during the heating sequence of startup. It allows the operator to know how long until the next startup operation occurs. Through the use of property nodes, the startup timer will disappear when not in use and reappear as needed so as to not distract the operator and clutter the screen. Property nodes were also used in a few other places. When performing a manual startup, there needed to be a trigger to move the system into the normal operation state. A Boolean button was added in order to initiate this transition; however, it is not necessary for the button to always be visible. Through the use of property nodes, the button only appears in the manual startup state and disappears once the program moves to the normal operation state. Comments were also added to the front panel to provide a description of the code currently being executed. Additionally, indicators for the main state, startup state, and shutdown state are available for the operator to know which state the program is in. Property nodes were used here as well so that the shutdown state indicator is not visible during startup.

A third improved detail is replacing the toggles for the Tops to Feed and Bottoms to Feed pumps. In the original code, it was difficult for the operator to know if the toggle was switched to on or off. This increased the risk of leaving a pump on accidentally which could disrupt the process flow as well as run the risk of emptying the top and bottom product vessels and causing the pump to run dry. To address this risk, the toggles were replaced with a different control that lights up when pressed to indicate that the pump is on. The pump controls were also grouped together visually and kept isolated from the other controls since they are not typically manipulated during normal operation.

The final front panel aspects to identify are the ‘Go’ and ‘Shutdown’ buttons. They have each been grouped with their corresponding switch to determine whether the sequence will occur in manual or automatic. By placing the switch and button next to each other, it increases the likelihood of the operator considering whether to perform the task in automatic or manual. The buttons were also placed together closer in proximity and vertically aligned. This adds structure to the overall layout and maintains the cleanliness established with the controller groups on the left side of the screen. While it may appear that placing the ‘Go’ and ‘Shutdown’ buttons in closer proximity may increase the risk of clicking the wrong button, the contrasting green and red colors and large text create strong visual separation between the two and are a subconscious visual reminder of the distinction between each button.

C. Plant Simulator

Another feature added to the distillation column software is the ability to run the code in a simulation mode. This allows for changes in the logic to be evaluated without introducing risk of damaging equipment if something does not work as designed. Part of the simulator was built into the Control Points subVI in the event structure. The second part of the simulator is within another subVI called Plant Simulator. This subVI is not accessed within the main VI, instead, it is used in conjunction with the main VI to control process variables and replace the signal that would have been obtained from the FieldPoint channels. The front panel of the Plant Simulator subVI is shown in FIGURE 42.

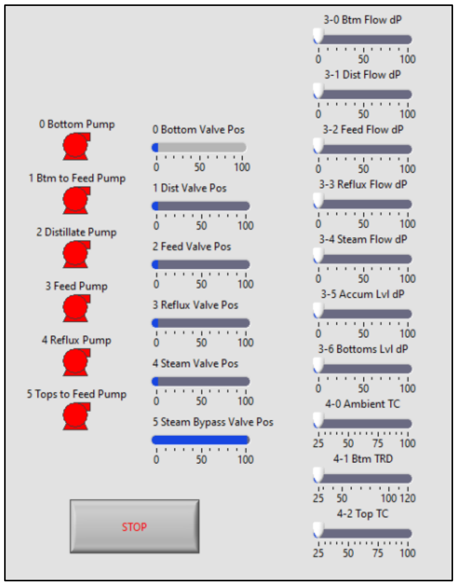


FIGURE 42: LabVIEW Plant Simulator SubVI Used to Perform Simulation of Distillation Column Program

There are indicators for the pumps and valve positions and controls for the differential pressures and temperatures. This program uses network variables to communicate with the simulation mode of the Control Points subVI so that as a temperature is changed within the Plant Simulator subVI, the corresponding temperature in the main program is updated with the same value. Another change made with the creation of the simulation mode is the use of a simulation delay. With this simulator, changes are instantaneous, and it can be hard to see each state as it happens. Therefore, a delay was added so that if the program was run in simulation mode, a delay would be applied to the startup and shutdown states so that the user can observe each state as it happens. Since the delay was repeated in nearly every startup and shutdown state, a subVI was used. The subVI, shown in FIGURE 43, used a case structure so that if the program was in simulation mode, the true case would occur, and a wait delay of five seconds would be applied. If the program is not in simulation mode, no wait delay occurs and the subVI is passed through.

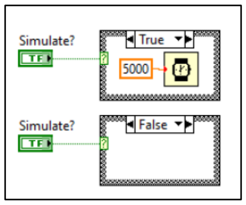


FIGURE 43: True and False Cases of LabVIEW Case Structure Used to Implement Five Second Simulation Delay

D. Distillation Column Redesign with C&I Engineering

One of the main issues with the distillation column itself is the entangled piping. The piping configuration has negative consequences on both process control and equipment accessibility, affecting the operator, maintenance tasks, and the students' learning experience. The best way to address this problem is to perform a complete redesign of the distillation column system. The University of Louisville collaborated with C&I Engineering in order to create a new equipment design addressing the current deficiencies. C&I Engineering is a consulting engineering company that specializes in process design and improvements. The collaboration between C&I and UofL to redesign the system allowed for C&I to create not only an efficient design, but also one that meets the established needs to enhance learning through the Unit Ops lab.

The primary goals of the redesign were to decrease deadtime throughout the system and to make equipment more accessible. In order to decrease deadtime, the piping configuration must be streamlined to eliminate excess pipe length and bends. Reducing these factors will reduce kinetic losses of the fluid and will allow the fluid to maintain a higher velocity. The shorter distance and the streamlined path will ultimately decrease deadtime and allow for a tighter control of the process. The second part of the goal, increased equipment accessibility, must be performed in conjunction with the first. Piping must be minimized while also maintaining an equipment configuration that allows for students and anyone performing maintenance tasks to access the equipment. Specifically, pumps, valves, and rotameters should be relocated so that they can be easily accessed to perform maintenance tasks as well as for students to easily identify the equipment and understand which part of the process they correspond to. An additional benefit to streamlining the piping will be the ease at which the students' will be able to trace the piping and follow fluid flow. Ideally, the piping bundle will run parallel with direct routes to and from each point of the process. Pipes will be labeled and color coordinated to provide visual cues for students to help them understand the fluids within each pipe and where it they are being transported.

While many pieces of equipment will be replaced within the scope of the redesign, it is not monetarily practical or feasible to replace everything. The distillation column and thermosyphon reboiler will be kept through the redesign. Since both pieces of equipment are made of glass and allow for a unique perspective of distillation that aids the learning process, they are beneficial to keep it. A replacement of a similar column and reboiler would be too costly, especially when there is no immediate need for a replacement since the original still functions well. A consequence to keeping the column, however, is that it cannot be taken apart and relocated for the redesign. The gaskets for each stage of the column have dried out due to the methanol in the system and would likely crumble if the column is taken apart. Replacing the gaskets would be tedious and too expensive to justify the end result. This restriction increases the complexity of the redesign and limits equipment orientation options. Another limitation centered around the distillation column is that students need to be able to access each tray of the column in order to take liquid and vapor samples. The sample ports are currently accessed using scaffolding around the column with a vertical ladder leading up to the platform. While the exact scaffolding does not need to be kept in the redesign, there still must be a way for students to safely access the stages of the column.

Other equipment to be kept through the redesign include the feed, top product, and bottom product vessels. These tanks are made of copper and provide adequate storage for the fluid volumes used to perform the distillation. Unlike the distillation column, these tanks can be relocated with the redesign to optimize the space. The accumulator is a final piece of equipment that will remain through the redesign. Similar to the column, it is made of glass to provide the operator with a visual of the level. This is necessary in the event that the differential pressure cell fails so that the operator can still monitor the accumulator level and run the distillation column safely. Additionally, the glass helps students to understand the purpose of the accumulator.

Throughout the design phase, multiple meetings were held in the Unit Operations lab as well as at C&I's office. Initially, the meetings were conducted in order for C&I to gain an understanding of the existing equipment and how it is used. From there, discussions began to focus on what aspects need to change and what must stay the same, such as the distillation column and tanks. Once these limitations were identified, preliminary designs began to be developed. During this stage, C&I presented the opportunity to use their laser scanning services to create a 3-D image of the distillation system. Having this image would provide C&I with accurate dimensions available for them to design around and overall enhance the precision of the designing process. This was especially helpful due to the spatial restrictions of the Unit Operations lab.

After obtaining the laser scan of the distillation process, a proposed equipment configuration was developed. One topic of discussion for the new design was focused on how students would access the stages of the distillation column to obtain samples. Three options were proposed by C&I and discussed across the team. The first option was to create a sampling station using tubing connected to the sample ports and directing all samples to a single station floor level. This would eliminate the need for scaffolding complete. Unfortunately, this option was proven to not be feasible due to the length of tubing required. A second option discussed was to use a rolling staircase and eliminate the platform scaffolding completely. Prior to the installation of the current scaffolding, this was the access method used to obtain samples. The scaffolding was installed due to safety concerns, expressed by the Accreditation Board for Engineering and Technology (ABET), associated with the rolling staircase. Therefore, this option was eliminated. The final option discussed was removing the scaffolding that wraps around the front of the column and instead, adding scaffolding that wraps around the back of the column and comes partially around the other side. In addition to the new location of the scaffolding, the ladder would also be removed and a staircase parallel to the back wall would be installed along the wall, providing a more accessible way to reach the platform. The stairway is safer and more inclusive so that all students will be able to access the sample stations on top of the column. This configuration also opens up the front of the column and allows students to have a more unobstructed view of the distillation process. Based on these benefits, the third option was selected for the final design of the process.

FIGURE 44 is the equipment design created by C&I Engineering. The FIGURE depicts the aerial view of the agreed upon scaffolding configuration, as well as the location of the rest of the equipment in the redesign. Another perspective of the layout is documented in APPENDIX V. In this equipment design, a primary change is the location of the top and bottom product tanks. Instead of being side by side, the top product tank will be located above the bottom product tank. This will conserve space as well as act as a visual reminder for which tank is the top product and which is the bottom product. The feed tank will remain in relatively the same location as it currently is. Unfortunately, piping arrangements have not been finalized and are not included in the equipment arrangement schematic. It is likely that piping will be mounted on a skid and run parallel to the front of the staircase. This would make it easier for students to trace the piping and follow the flow of the process fluids. Additionally, pumps, valves, and rotameters will be located within the general area between the distillation column and the product tanks, allowing for easier access for maintenance and learning purposes.

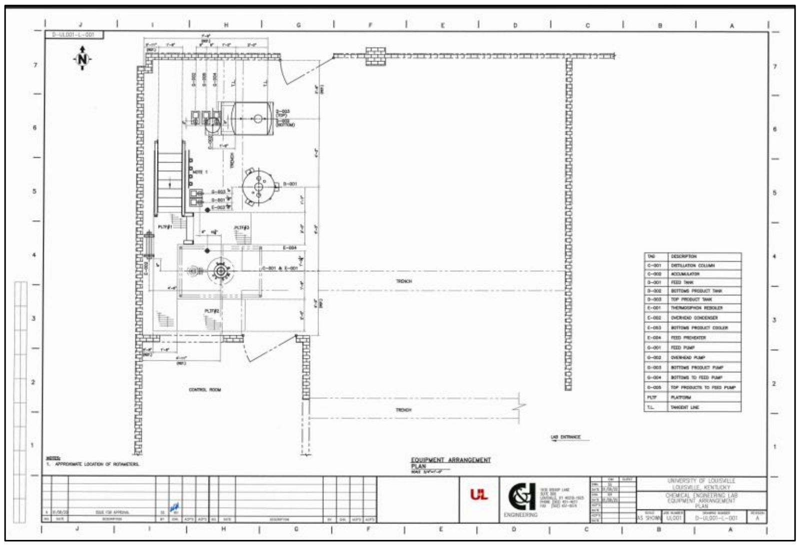


FIGURE 44: Equipment Design Developed by C&I Engineering of Proposed Unit Ops Distillation Column Redesign

IV. IMPACT

Due to the equipment unable to be fixed, the upgraded software was not able to be tested on the actual distillation column. Instead, a simulation was created and incorporated into the software in order to test the logic. The simulator was helpful for testing the updates made within the scope of this project and will also be useful to test future changes made to the program without introducing risk to the equipment.

When running the code in simulation mode to test the updated code, a major issue with the countdown timer was identified. Each state that required a countdown timer to expire before progressing to the next state would occur instantly with zero wait time. The states would pass as though there was no countdown timer at all. This prompted a closer look at the code used to create the timer. Each time a small change was made to attempt to correct the problem, the code was able to be tested in the simulation mode without the risk of damaging equipment. The simulation mode also allowed for the LabVIEW features of highlighting the execution and probing the wires to be used. Highlighting the execution slows down the program so that the user can follow the flow of data through the wires on the block diagram. Probing the wires allows for users to select specific wires to see what values are passed through them at a given time. By using the highlight execution and probe wire features while troubleshooting this issue, it was discovered that a “less than” node was used instead of a “greater than” node, resulting the Boolean that indicates if time has expired to always be true. This caused each state with a timer to execute as if the timer had already expired before any actual time had passed. Had this issue not been identified during a simulation, there would have been potential for damage to occur to the glass reboiler and column. During automatic startup, there is a state that makes the program wait for five minutes after steam is first established in order to give the column time to gradually heat up before applying more steam. Skipping the wait period would increase the risk of thermal shock and could damage the glass. Identifying the countdown timer issue within the simulation mode prevented any damages from occurring to the distillation column due to a malfunctioning timer and allowed for a low risk method of troubleshooting.

Other than the countdown timer, there were no other issues identified during the simulation. Each state occurred as designed, including states controlled by a specific setpoint being met. The simulation also confirmed the correct use of property nodes as the program successfully made indicators and controls visible or hidden as specified in the code. Overall, the simulation proved to be a useful tool to identify issues within the code and will be useful in the future to test out any other upgrades made to the software. Once the rest of the equipment deficiencies are addressed, the upgraded code is ready to be tested on the physical process in order to evaluate its performance compared to the original code.

While an actual trial of the code was not able to be obtained, the impacts of the upgraded code can still be estimated. The logic has been tested through the simulation and was proven to work as designed. Therefore, it is expected for the code to run successfully with the distillation column as well. One expected impact of the new code is the speed of the automatic startup. The automatic startup was not able to be tested but it was noted through past experiences that the code caused startup to be significantly longer than a manual startup. After dissecting the old program, it can be predicted that one of the primary causes of the extended automatic startup is a result of blocking with the wait timers. Loop iterations were slowed significantly throughout nearly all of the stacked sequence frames, preventing the code from being performed at a maximized speed. Wait timers were eliminated in the updated LabVIEW code in order to avoid blocking and allow the program to continuously measure and update variables. Therefore, it is expected that the only delays experienced in the program will be within specified states which will optimize the speed of code execution and reduce the time it takes for an automatic startup.

A known impact of the updated code is that it allows for a safer operation of the distillation column. The new architecture allows for system data to be available for the duration of running the program so that the operator is not blind to important variables such as temperatures and levels. Additionally, the new code now provides the operator with the option to shutdown the process at any point of startup or normal operation. The original code restricted the operator to only being able to safely shutdown once the program was in normal operation. Should an automatic startup require an emergency shutdown, the operator had no way to safely perform one. The updated code eliminates this issue which ultimately makes it a safer program to run the distillation column with.

V. RECOMMENDATIONS

While the upgrades completed within the scope of this project have improved the distillation column system, there is still room for opportunity to further enhance the operation of the system. The first and foremost recommendation is to address the equipment that continues to malfunction as well as pursue the planned redesign with C&I Engineering. In order to take advantage of the programming upgrades made, the equipment must be functioning properly. The updated automatic startup will be of little use if there is no way to obtain live temperatures of the column to control the heating sequence from. It will also continue to be difficult to control steam and feed flow until a more linear valve is installed. The redesign is necessary to simplify the system and make it easier for students to figure out how the separation process works, as well as make equipment more accessible for maintenance operations. Once these items are addressed, further opportunities for growth become available and the use of the distillation column can be extended to other facets of learning.

One such extension of learning includes process control. This can be applied both within the LabVIEW code itself as well as through the Unit Operations lab. Within the normal operation state, PID controllers are used when running a specific loop in automatic versus manual. After the redesign is complete, there will be new process dynamics values within the system to account for. Therefore, the PID controllers will need to be tuned in order to better control the new system configuration. The tuning of the controllers can be performed as another upgrade project to further improve the code. Alternatively, it could be incorporated within the process control course as a tangible example of how to tune a controller and the impact it has on fluid control.

Once tuning parameters are obtained for each controller, a known basis can be established. From this, a process control lab can be developed where students adjust the parameters of the PID controller and see how the process responds. Specifically, students could plan and perform various types of bump tests with the controllers to study the relationship between the process variable and the controller output to develop their tuning parameters. There is functionality within LabVIEW to have controls on the front panel so that students can adjust parameters of the PID controllers in normal operation. This will allow students to observe the impact different tuning parameters have on flow rates, temperatures, and levels. It would also provide students a firsthand experience in tuning controllers and require more active participation to help reinforce the concept of process control. New charts can be added to track the necessary variables for tuning controllers. Through the use of property nodes, the charts can be hidden so that the operator can select which charts they want to see. This will allow for the necessary charts to be displayed with limited space on the front panel.

With better process control through optimized equipment design and tuned controllers, another potential use of the distillation column can be simulated upsets. This can be performed as a class demonstration or as an extension of a process control-based lab. By working through a process upset, students will learn how upsets occur, how they affect equilibrium throughout the column, and how to adjust other parameters to recover from an upset. The updated front panel will provide an organized and clear interface for students to interact with. All necessary system data, including temperatures, flow rates, and levels, are available for the students to use to monitor the impact of an upset. Controls are also available for students to react to the upset and attempt to actively recover the process. Having the data and control capability available will force students to actively problem solve based on the available data. This will allow students to develop an understanding of the relationship between the different parameters and guide students to analyze the system as a whole. If performed as a lab, there would be an increase in engagement between the students and the distillation column as they decide on changes to make and monitor how the system responds. It provides an opportunity for teamwork to recover the process as well as an opportunity for active learning as they apply their knowledge of distillation and material balances to a tangible process.

Another process control improvement that can be completed after the redesign is the implementation of updated control loops. One loop would control the top of the distillation column and a second loop would control the bottom. For the top of the column, the distillate and reflux valves would be manipulated based on the temperature of the top of the column as well as level of the accumulator. This loop would provide tighter temperature control while maintaining a reflux and accumulator level. For the bottom of the column, a control loop would be used to control the temperature and level of the bottom of the column by adjusting the steam and bottoms valves. These loops would provide better top and bottom product purities and ultimately provide better control over the temperatures and levels of the column. While it is possible for the loops to have been incorporated within the scope of this project, it was not practical to do so with the upcoming redesign. The redesign is expected to occur as soon as possible and once it is complete, the control loops would have to be retuned for the new deadtime values within the system. Creating the updated control loops before the redesign would result in obsolete tuning parameters and no use of any work performed to obtain the parameters. Therefore, it is recommended to obtain the new tuning parameters and create the control loops after the redesign is complete to provide better product purity control.

Building in error handling is another recommendation for future code improvements. Even with the updated code, there are limited safeguards in the event an error occurs. For example, if the program or operator tells a valve to open 100% but the valve never physically opens, there is nothing built into the code to address the issue. Eventually the operator would become aware of the issue once impacts on flow rates, temperatures, and levels are observed; however, if it is a pump malfunctioning, this could be too late to prevent equipment damage from occurring. Therefore, additional functionality in the code must be built to check that valves and pumps actually move to the position they are set to. Deadband would have to be accounted for to ensure that a valve set to 25% does not cause an error if it is physically at 26% instead. Another example of error handling would be monitoring temperatures and levels and sending the operator an alert if the temperatures or levels exceed safe operating limits. These alerts would increase the safety of the system and reduce the risk of overheating the column or cavitating the reflux, distillate, and bottoms pumps. An important factor to consider when implementing error handling is that there would have to be a way for the operator to decide between continuing with the current state of the program, moving to a new state to troubleshoot the problem, or performing a safe shutdown of the process. Additionally, when the program identifies an error and moves to an error state, it would need to ensure that the equipment not experiencing an error remain in a safe position and can still be controlled if needed. While these upgrades are not exactly necessary in order to run the distillation column, they are important and will be beneficial for handling equipment errors when they inevitably occur.

One more programming update that would build off of the changes made within this project is improving the plant simulator. The plant simulator currently allows for the user to control temperatures and levels to progress the program to the next state. A more advanced way to simulate the distillation column would be to develop temperature and differential pressure signals that mimics how the actual signals would respond to a change in valve position or pump configuration. This would provide a more realistic simulation that would give the user a more accurate idea of how the actual system will respond to the code.

Due to the impacts of Covid-19, universities, including the University of Louisville, have switched to online courses as a temporary safety precaution. While many classes can be converted to online courses, the Unit Operations lab is much more difficult to transition. The purpose of the Unit Operations lab is to provide students with an opportunity for active learning through hands-on lab experiments with chemical engineering processes and equipment. Transitioning this course to online reduces the opportunity for hands-on learning and decreases the effectiveness of the course. However, having an accurate simulation of a distillation column through LabVIEW would allow students to still get hands-on process control experience in an online format. LabVIEW is an accessible software for students to install on their personal computers. A simulation with realistic process data for temperatures and flow rates would allow for students to make changes as a system operator and monitor how other parameters of the system are impacted. A fully functioning, realistic distillation simulator would be an essential tool for delivering the Unit Operations course online in a creative and effective manner. With the lasting impacts of Covid-19, as well as in preparation for future needs to move learning online, a simulation is an important tool to be developed to allow for active learning within a digital environment.

A final recommendation that would further improve the use of the distillation column is to develop a web application. The web application would enable the operator to have live system data available on a portable device such as a smart phone or tablet through a single page web application. The web application would update as new data is available to provide the operator with real-time process data. With the inclusion of error handling within the code, the web application would also be able to display system alerts for various errors. This allows the operator to step away from the system but still have full access to system data to monitor the process, make sure it remains in a safe state, and respond to any alerts. Additionally, the web application can be used to execute the process upset as part of a process control lab. Students would be monitoring and controlling the system through the LabVIEW code while the instructor could force an upset from the web application on a mobile device that students will have to identify and recover from. This type of lab activity would provide students with an opportunity to practice teamwork and problem-solving skills in conjunction with their knowledge of process control and distillation.

An Application Programming Interface, or API, could be developed using a variety of hosts, including Microsoft, Google, or Amazon to communicate with the single page web application. This aligns with the concepts of Industry 4.0 such as using the cloud for data storage and increased data accessibility to enhance flexibility within the manufacturing environment [11]. By incorporating concepts of Industry 4.0, it will be more convenient for the operator to monitor the distillation column from afar, while also providing students a preview of the possibilities of integrated technology in manufacturing.

VI. CONCLUSIONS

Prior to the improvements completed over the course of this project, use of the Operations Lab distillation column was limited. Startup could only be performed in manual with significant operator involvement throughout the entire process. There was little control over the system with essential equipment either insufficient for process control or not functioning at all. The control program use of a stacked sequence structure limited when data was available to the operator and prevented the possibility of an emergency shutdown during startup. LabVIEW code on the block diagram had multiple deficiencies outside of the chosen programming architecture, including data flow issues and overcomplicated code. Controls and indicators on the front panel were unorganized, redundant and excessive data were displayed, and the graphics were outdated. Due to the restrictions associated with the equipment and LabVIEW code, the functionality of the distillation column was minimal. While this makes it more difficult to operate the distillation column, it also reduces its use within the Unit Operations lab as a tool for learning.

In order to optimize the use of the distillation column, significant changes were made. Faulty equipment was addressed, and an overall redesign was developed with C&I Engineering. These changes will allow for better control over the system and improved accessibility of equipment for maintenance and learning purposes. In addition to the equipment and layout upgrades, the LabVIEW code to run the column was completely overhauled and updated. The overall architecture was converted to a state machine and code was simplified by separating the logic from read/write operations and provided a cleaner and safer mechanism to run the distillation column. Real-time system data is now available during all states of the code and shutdown can occur as needed, even during startup. The upgraded code is also more flexible and can be readily adapted for future updates performed on the distillation column.

The changes made to the LabVIEW code were necessary in order to develop a strong foundation. This new foundation in the form of a state machine can easily handle future upgrades for whatever functionality is determined necessary to operate the distillation column. Additionally, the updated program is more intuitive which will allow for students to become the operators of the distillation column. Allowing students to directly control the distillation column and make decisions on manipulating pumps and valves will increase their active participation within the lab and overall enhance their distillation learning experience. Another benefit of the code upgrades includes the plant simulator. Operating the code in simulation mode provides a method for testing future updates and allows the code to be executed in a safe state for troubleshooting without any risk to the equipment. The foundation of the simulation mode also paves a path for a more fully developed simulation to be created to allow for continued active learning through a digital classroom.

It is the combination, however, of the equipment and code updates that will improve the use of the distillation column as a learning tool. The optimized system will allow for more involved experiments to be performed and will help transition the distillation column from being a passive lab to a more active experiment. A wider variety of experiments can be conducted around the distillation column with hands-on tasks for students to perform such as tuning controllers or managing process upsets. This unique perspective will help students understand the process as a whole and how each system variable is related to the other. Additionally, the concepts of process control and separation operations can be taught in conjunction, highlighting the intersectionality of the two courses and how they are commonly used together in a manufacturing environment.

Ultimately, this is not the final form of the code or distillation column configuration. The work performed within the scope of this project paves the way for future updates to be made as the function of the distillation column shifts to meet new and evolving learning goals. The purpose of the distillation column is to provide a lab scale example of a common separation operation process and provide students with hands on experience to supplement their learning. An optimized equipment layout and flexible LabVIEW architecture will be able to handle future upgrades and increased functionality to better enhance student learning and prepare students for their careers.

APPENDIX I: PROCEDURES

Proposed Manual Start-up Procedure of Unit Ops Distillation Column

1. Ensure enough MeOH and H20 in system, add more if necessary
    a. We added 5 gallons of MeOH and filled rest of feed tank with H20 (not all the way full)
    b. Usually there is a switch to transfer bottoms product to feed tank but switch is broken
    c. Typically only need to add more methanol after column has been run numerous times and MeOH has evaporated
2. Manually turn on steam and city water
    a. Steam valve at pressure gauge on wall
    b. Two valves to open for city water, one downstairs by steam valve and one upstairs behind table
3. Bleed the steam line by manually opening two valves to allow condensate to drain
    a. Make sure valve to column is closed
4. Start the LabVIEW program running in manual mode
    a. Startup sequence flipped to 'no'
    b. Shutdown sequence flipped to 'yes'
    c. Ensure all buttons are set to manual
5. Open the bottom valve position to 100%
    a. This turns on the pump and drains the bottom of the column to the bottoms product tank
    b. Code in LabVIEW is setup so that if there is any flow established by opening the valve, the pump will turn on
    c. May need to check pneumatics and if compressed air is open to ensure valve is working
6. Close the manual steam bypass valves to direct the steam through the control valve
7. Close the bottoms to feed valve when level reaches top of stem which is set as zero level based on dp calculations in program (program reads level based on dp and is accurate)
8. Turn on switch in program to drain bottoms tank to feed tank
    a. Open valve at rotameter
    b. Fill feed tank all the way using site glass, then close the manual valve and turn the switch off
9. Turn on feed and add level to column
    a. Manually add feed by setting feed valve to 100% which will turn on the feed pump using similar logic to the bottoms drain.
    b. Use the feed bypass to fill up quicker

    c. Fill until pot reaches 60-70% level (aim for higher than 50% because it helps column to reach steady state faster because level is lost to evaporation and composition of MeOH and H20 varies)

10. Once level is reached, close the feed valve by setting it to 0% and closing the manual bypass
    a. Close bypass first because it flows faster
11. Open the steam bypass control valve using controller at column
    a. Valve is pressure to open so use increment button to add pneumatic pressure to open the bypass valve
12. Open steam valve to 45% in LabVIEW then slowly open manual valve to let steam flow into thermosyphon
    a. Maintain until temp of bottoms is between 40-50 degC
13. Open steam value to 55% in LabVIEW
    a. Maintain until temp of bottoms is between 60-70 degC
14. Back off bypass
    a. Remove pressure to close valve a bit
    b. Note: the valve slips so you must keep adjusting the pressure to maintain the valve position
    c. Watch for steam at stirred tank and close bypass more if steam is visible
15. Open steam valve to 65% in LabVIEW
16. Back of bypass and close it with the pneumatic controller and using manual valves to isolate the line in case the valve slips
17. Maintain steam valve at 65% for duration of run
18. Check tops temperature and accumulator level
    a. Accumulator maintains level for pump so no running pump dry when flow fluctuates
19. Look for 60% level in accumulator then start reflux by opening reflux valve to 100%
    a. A lot of lag here because the reflux line is very long
20. Timer starts where either top temp equilibrates at a constant value for 2 minutes or waits 15 minutes before continuing
21. Transition to continuous operation mode
22. Continue to check accumulator level and manually drain some to product tank to prevent overflow by cracking open the valve

Experimental Procedure for Unit Ops Distillation Lab

1. The column was started approximately 5 hours prior to data acquisition to allow it to reach steady state. The column was judged to be in steady state when the temperatures of the feed, bottoms product, and tops product did not change for approximately 1 hour. The data could then be collected.
2. The temperature for the top and bottom trays was measured in degrees Celsius and read from control panel.
3. The flow rates of the bottoms product rate, the tops product rate, the overhead rate, and the feed rate were collected from rotameters beside the top and bottom product tanks. The rotameter was read from just below the widest point of the float. The temperature was also measured in degrees Celsius from a thermometer on top of the rotameter.
4. A sample was drawn by the instructor from the feed, bottom, and top product tanks. The refractive index values of the samples were then measured by the use of refractometer. The refractive index values were recorded.
5. The individual tray samples were drawn by placing a syringe in a port on each bubble tray. The gas sample from the individual tray below was collected from the port on the farthest right of the column. The liquid sample from the individual tray was collected from the port on the farthest left on the column. This was done for all 6 trays.
6. To condense the gas samples into a liquid, an ice-cold rag was wrapped around the syringe as the sample was drawn into it, thus condensing it.
7. Once a sample was drawn, the refractometer was used to determine the refractive index values.
8. The bottom, top, overhead, and feed rates were again measured on the rotameter and then averaged with the first set of flow rate data.

APPENDIX II: STACKED SEQUENCE LABVIEW CODE

The following and screenshots of the original LabVIEW stacked sequence code to run the distillation column. Screenshots are labeled with corresponding frame numbers to represent the frame of each sequence and subsequence.

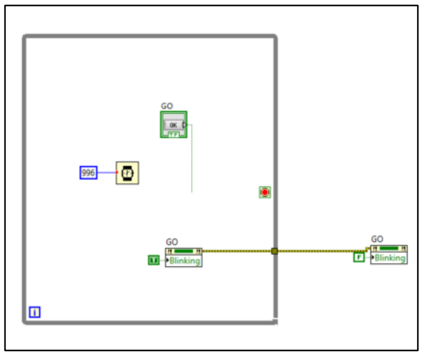


Figure 45: Frame 0 - Idle

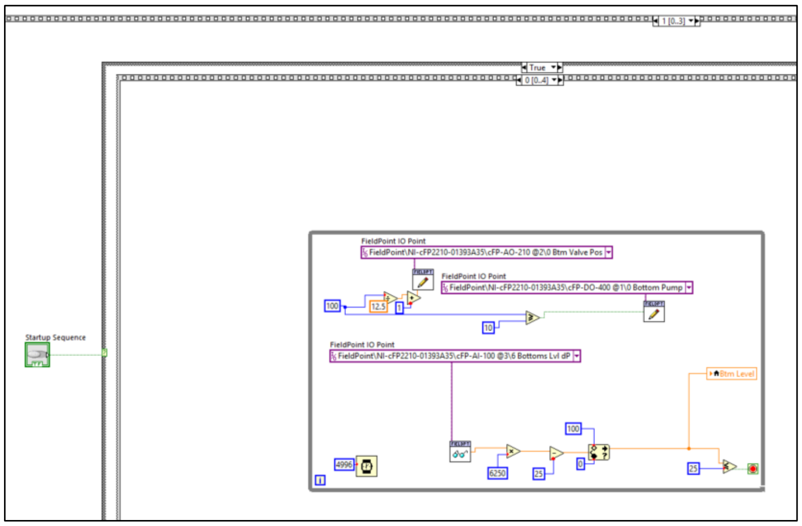

1 [0..3]
True
0 [0..4]
Startup Sequence
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\0 Btm Valve Pos
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\0 Bottom Pump
100
12.5
1
10
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @3\6 Bottoms Lvl dP
Btm Level
100
4996
6250
25
0
25

Figure 46: Frame 1.0 - Drain Column

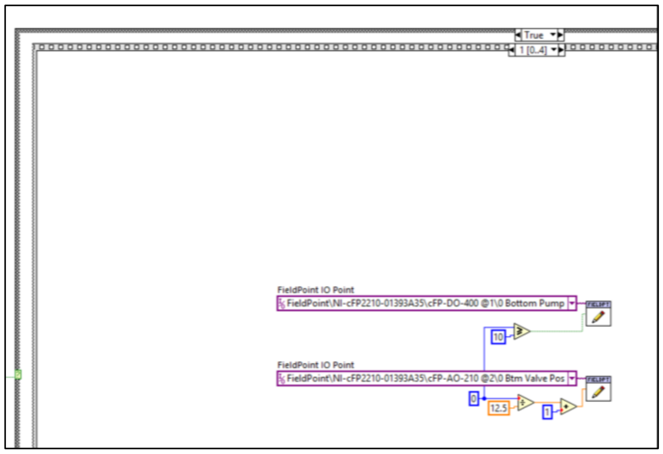

True
1 [0..4]
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\0 Bottom Pump
10
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\0 Btm Valve Pos
0
12.5
1

Figure 47: Frame 1.1- Stop Drain

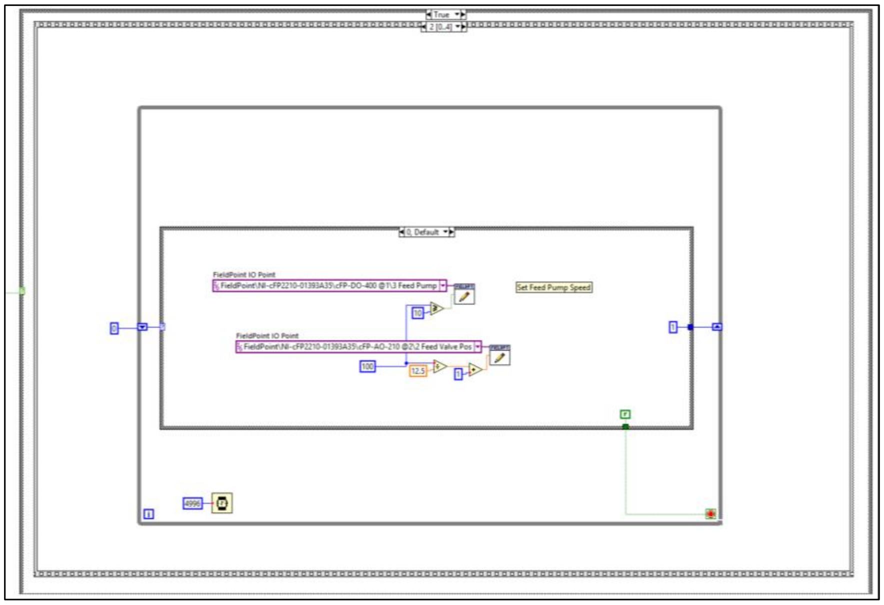

True
2 [0..4]
0, Default
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\3 Feed Pump
Set Feed Pump Speed
10
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\2 Feed Valve Pos
100
12.5
1
0
1
F
4996

Figure 48: Frame 1.2.0- Feed Sub-Sequence

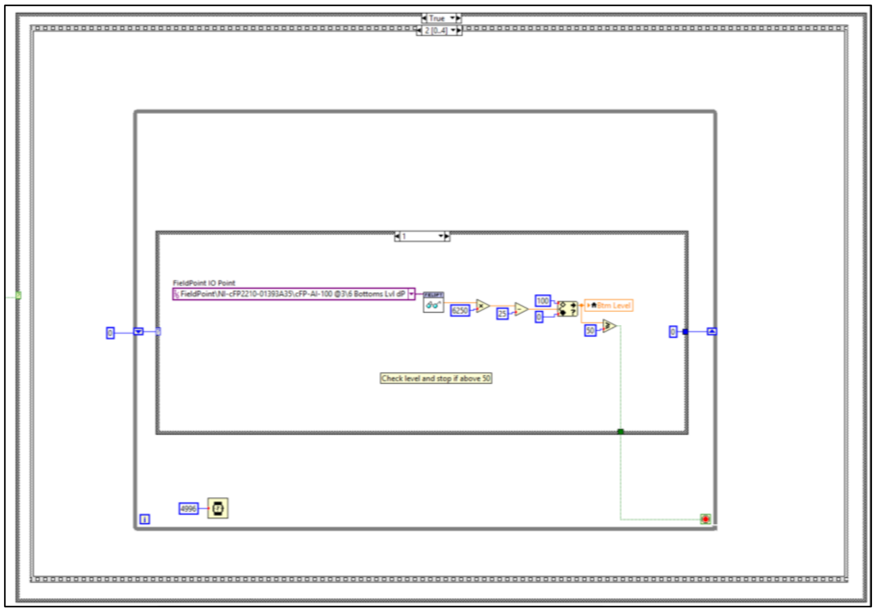

True
2 [0..4]
1
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @3\6 Bottoms Lvl dP
6250
25
100
0
Btm Level
50
0
0
Check level and stop if above 50
4996
i

Figure 49: 1.2.1- Feed Sub-Sequence

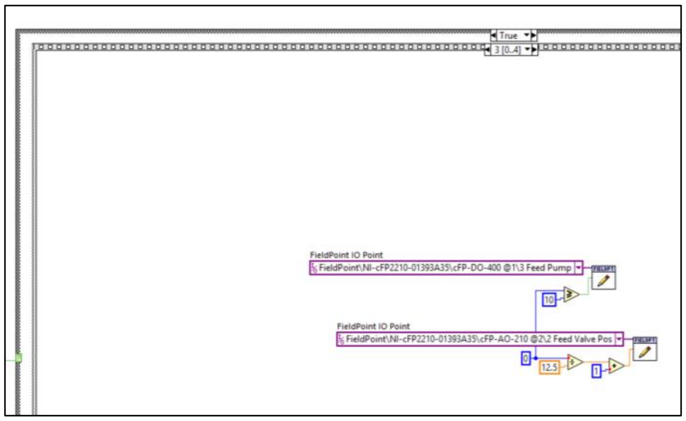

True
3 [0..4]
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\3 Feed Pump
10
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\2 Feed Valve Pos
0
12.5
1

Figure 50: 1.3- Stop Feed

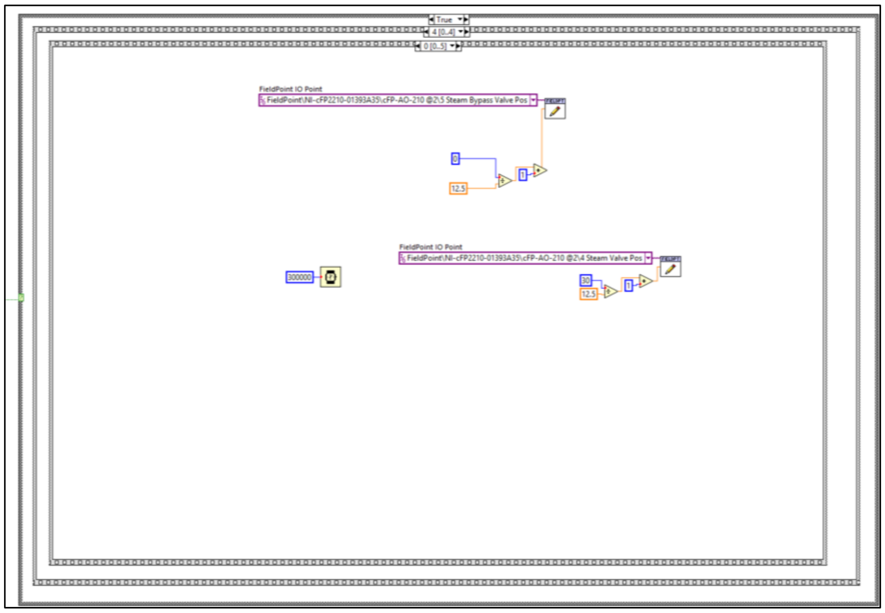

True
4 [0..4]
0 [0..5]
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\5 Steam Bypass Valve Pos
0
12.5
1
300000
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\4 Steam Valve Pos
30
12.5
1

Figure 51: 1.4.0 - Heating Sub-Sequence

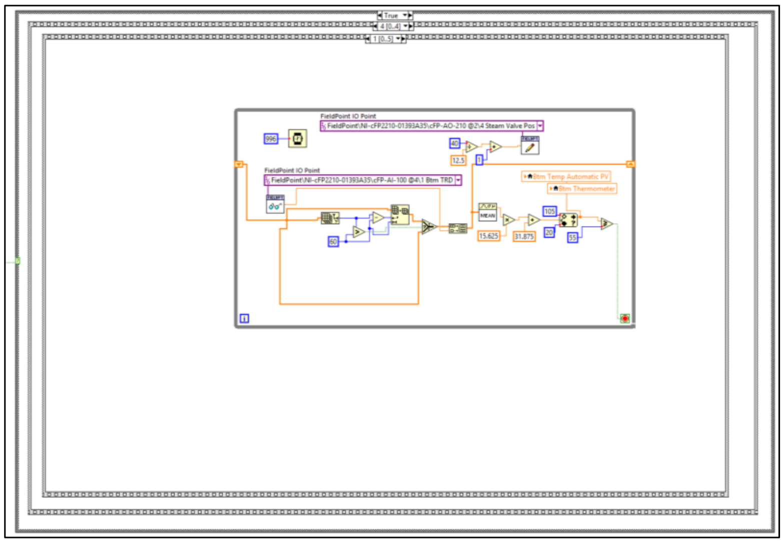


Figure 52: 1.4.1- Heating Sub-Sequence

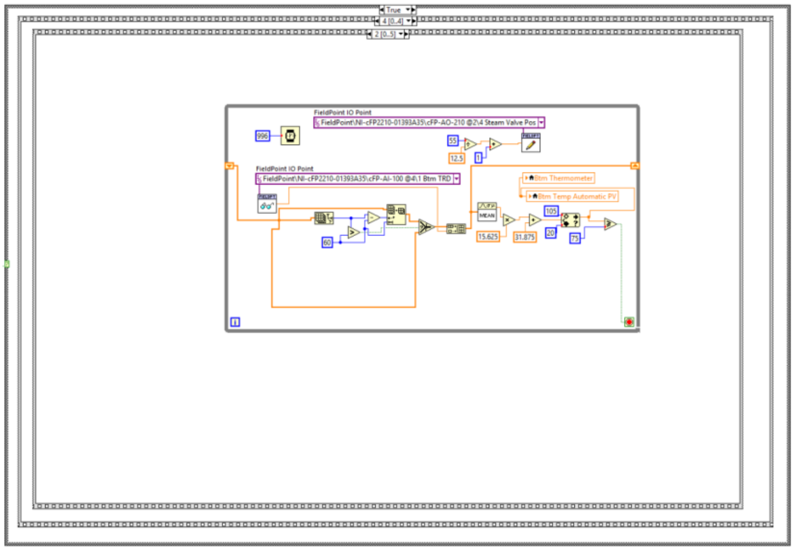


Figure 53: 1.4.2 - Heating Sub-Sequence

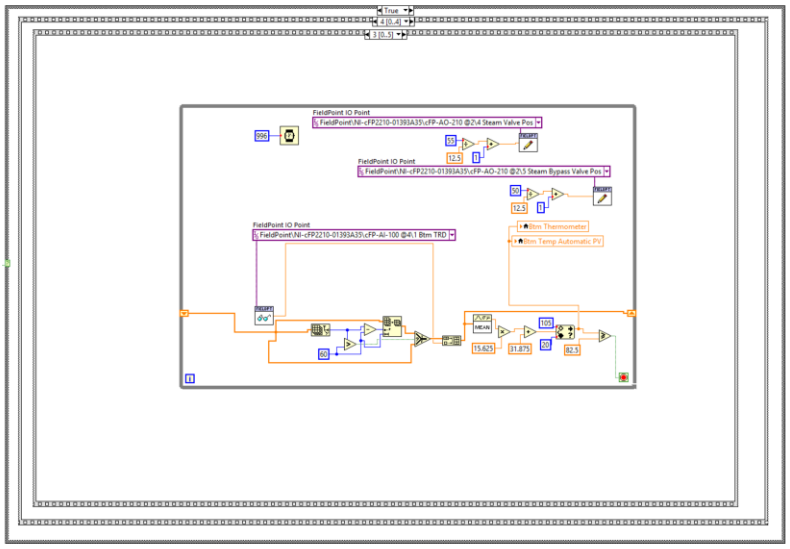

True
4 [0..4]
3 [0..5]
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\4 Steam Valve Pos
996
55
12.5
1
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\5 Steam Bypass Valve Pos
50
12.5
1
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @4\1 Btm TRD
Btm Thermometer
Btm Temp Automatic PV
MEAN
60
15.625
31.875
105
20
82.5

Figure 54: 1.4.3 - Heating Sub-Sequence

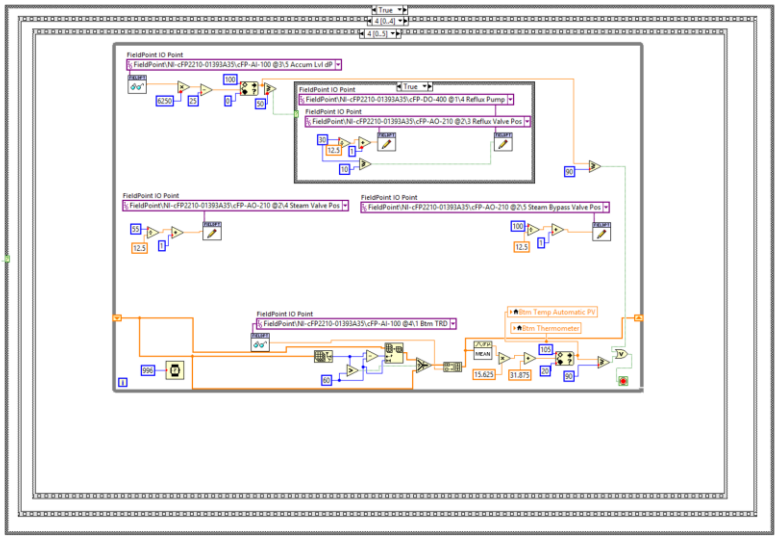


Figure 55: 1.4.4 - Heating Sub-Sequence

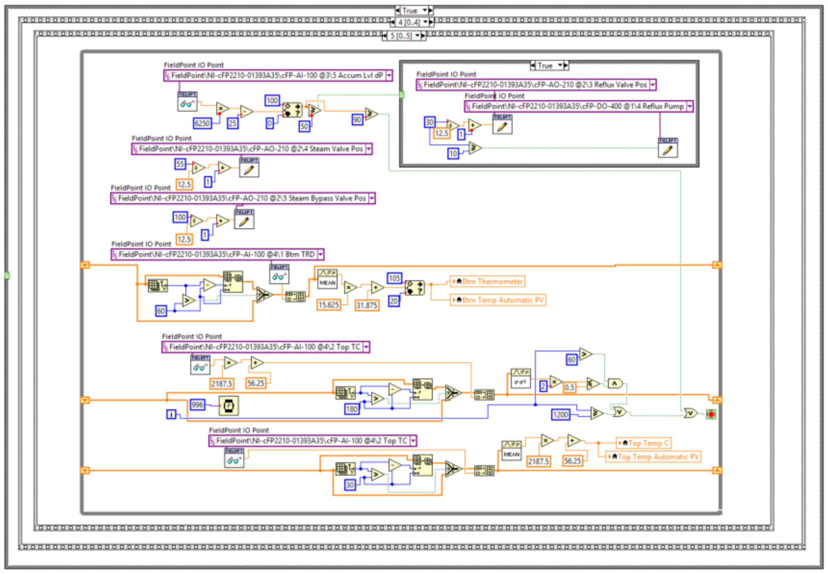


Figure 56: 1.4.5 - Heating Sub-Sequence

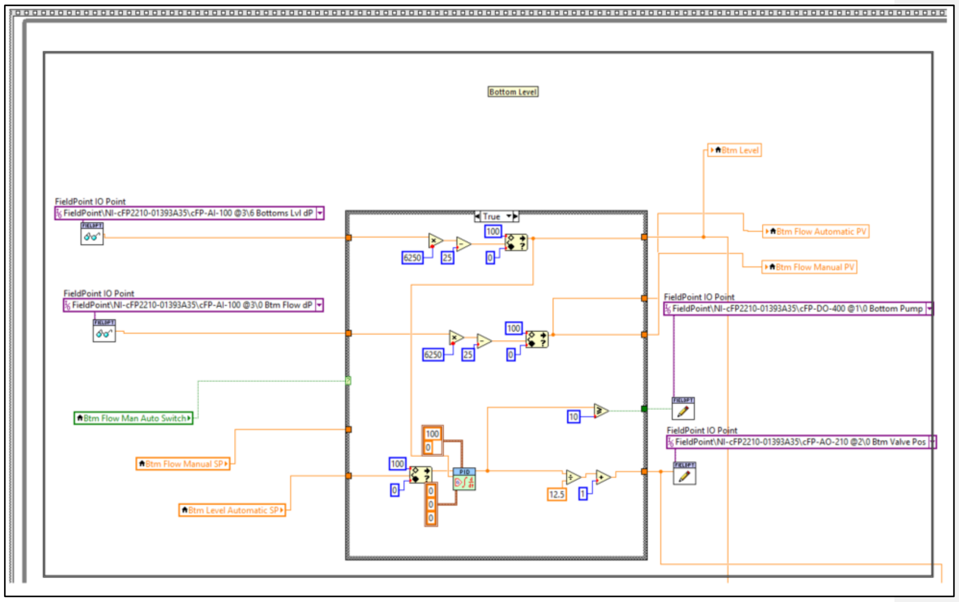

Bottom Level
Btm Level
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @3\6 Bottoms Lvl dP
True
Btm Flow Automatic PV
Btm Flow Manual PV
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @3\0 Btm Flow dP
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\0 Bottom Pump
Btm Flow Man Auto Switch
Btm Flow Manual SP
Btm Level Automatic SP
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\0 Btm Valve Pos
PID
6250
25
100
0
10
12.5
1

Figure 57: 2 - Normal Operation Bottom Level Loop

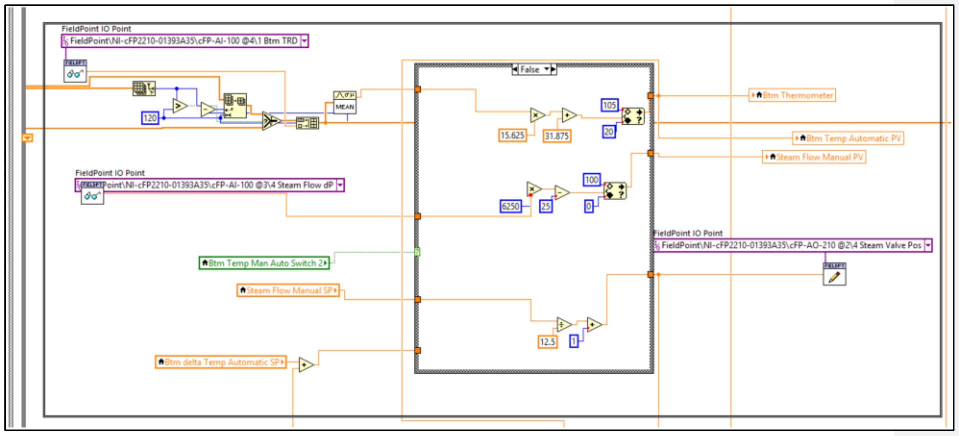
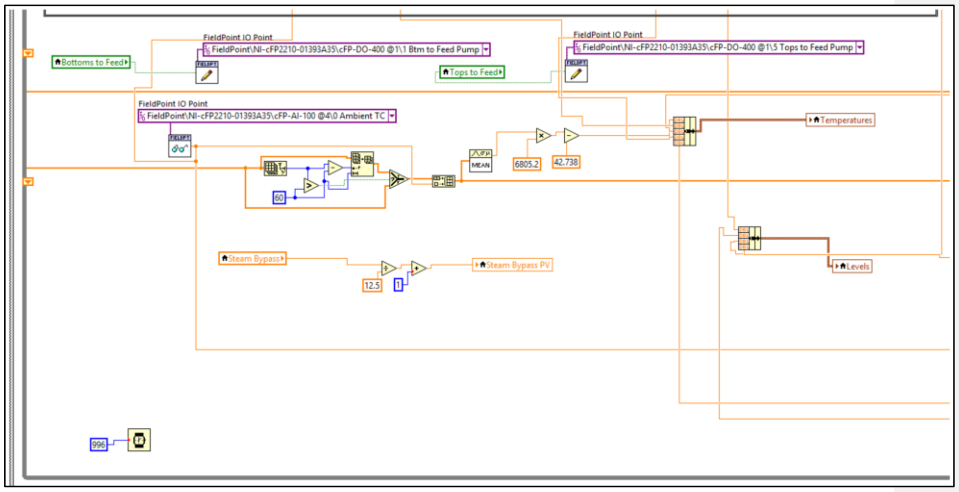
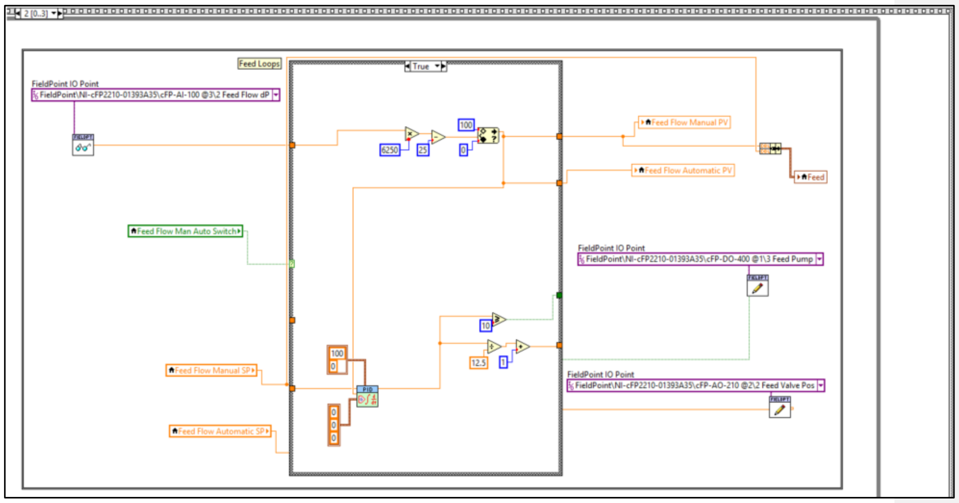

FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @4\1 Btm TRD
False
MEAN
120
Btm Thermometer
Btm Temp Automatic PV
Steam Flow Manual PV
15.625
31.875
105
20
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @3\4 Steam Flow dP
6250
25
100
0
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\4 Steam Valve Pos
Btm Temp Man Auto Switch 2
Steam Flow Manual SP
12.5
1
Btm delta Temp Automatic SP

Figure 58: 2 - Normal Operation Bottom Temperature Loop



FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\1 Btm to Feed Pump
Bottoms to Feed
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\5 Tops to Feed Pump
Tops to Feed
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @4\0 Ambient TC
Temperatures
60
MEAN
6805.2
42.738
Levels
Steam Bypass
12.5
1
Steam Bypass PV
996

Figure 59: 2 - Normal Operation Miscellaneous



2 [0..3]
Feed Loops
True
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @3\2 Feed Flow dP
6250
25
100
0
Feed Flow Manual PV
Feed Flow Automatic PV
Feed
Feed Flow Man Auto Switch
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\3 Feed Pump
10
100
0
12.5
1
PID
Feed Flow Manual SP
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\2 Feed Valve Pos
0
0
0
Feed Flow Automatic SP

Figure 60: 2 - Normal Operation Feed Loop

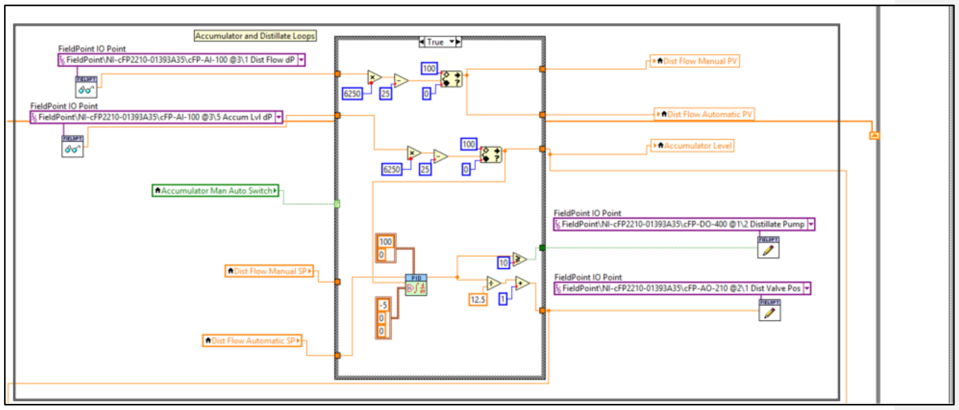

Accumulator and Distillate Loops
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @3\1 Dist Flow dP
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @3\5 Accum Lvl dP
Accumulator Man Auto Switch
Dist Flow Manual SP
Dist Flow Automatic SP
True
6250
25
100
0
100
0
-5
0
0
PID
10
12.5
1
Dist Flow Manual PV
Dist Flow Automatic PV
Accumulator Level
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\2 Distillate Pump
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\1 Dist Valve Pos

Figure 61: 2 - Normal Operation Accumulator and Distillate Loop

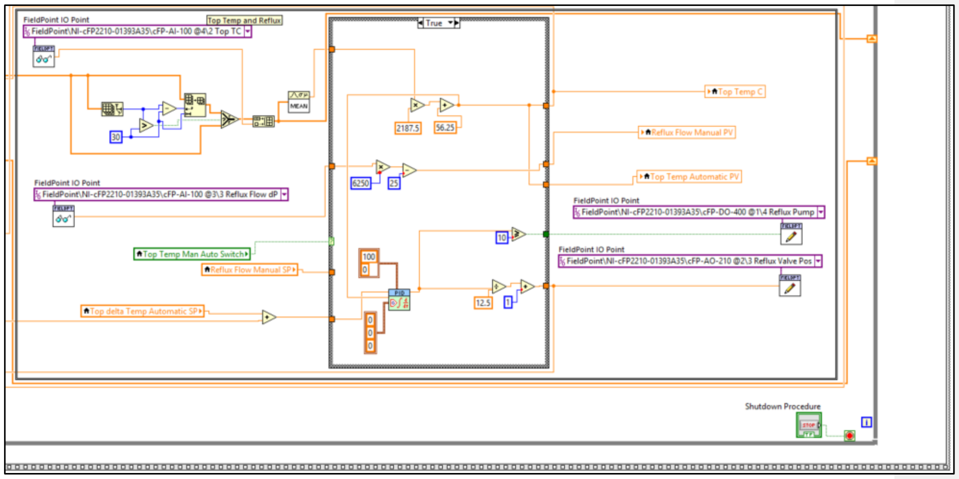

FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @4\2 Top TC
Top Temp and Reflux
30
MEAN
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @3\3 Reflux Flow dP
Top Temp Man Auto Switch
Reflux Flow Manual SP
Top delta Temp Automatic SP
True
2187.5
56.25
6250
25
100
0
0
0
0
PID
10
12.5
1
Top Temp C
Reflux Flow Manual PV
Top Temp Automatic PV
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\4 Reflux Pump
FieldPoint IO Point
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\3 Reflux Valve Pos
Shutdown Procedure
STOP

Figure 62: 2 - Normal Operation Top Temperature and Reflux Loop

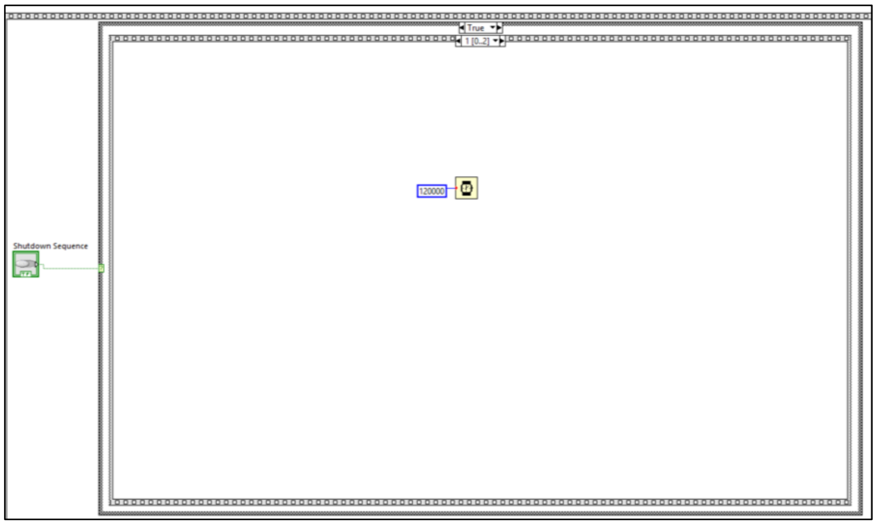


Figure 63: 3.1 - Shutdown Wait

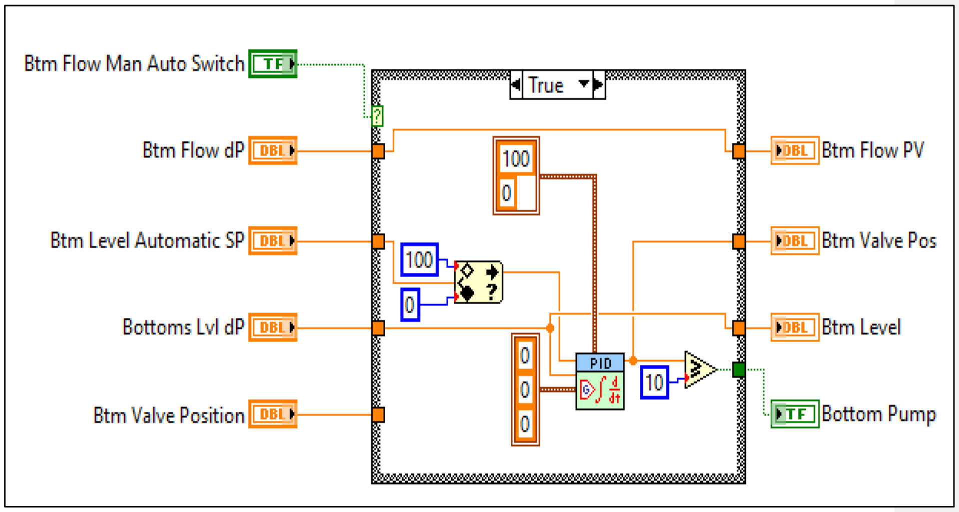
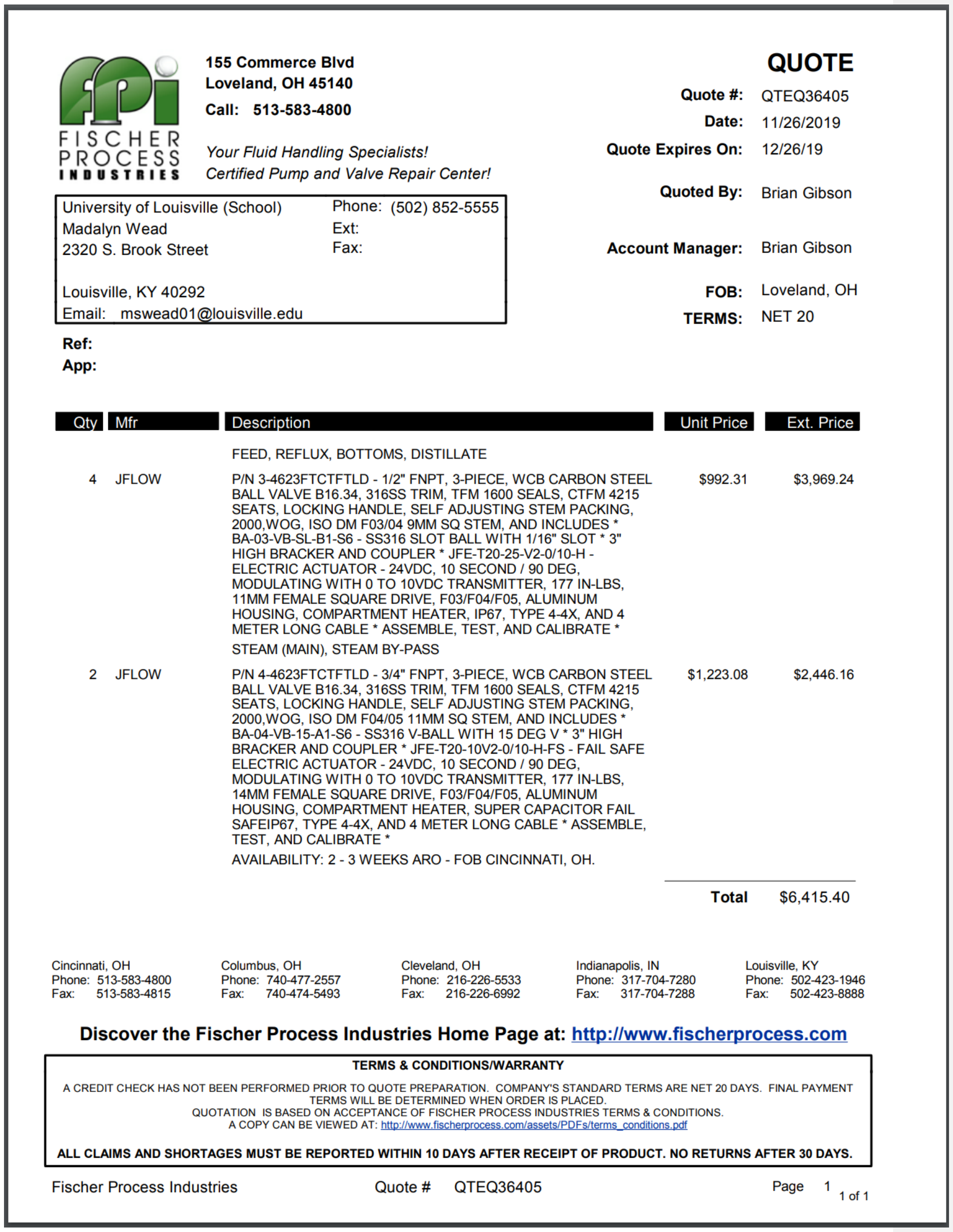


Figure 64: 3.2 - Shutdown Sequence

APPENDIX III: FISCHER VALVE QUOTE



FISCHER PROCESS INDUSTRIES

155 Commerce Blvd
Loveland, OH 45140
Call: 513-583-4800

Your Fluid Handling Specialists!
Certified Pump and Valve Repair Center!

QUOTE

Quote #:	QTEQ36405
Date:	11/26/2019
Quote Expires On:	12/26/19
Quoted By:	Brian Gibson
Account Manager:	Brian Gibson
FOB:	Loveland, OH
TERMS:	NET 20

University of Louisville (School)
Madalyn Wead
2320 S. Brook Street

Louisville, KY 40292
Email: mswead01@louisville.edu

Phone: (502) 852-5555
Ext:
Fax:

Ref:
App:

Qty	Mfr	Description	Unit Price	Ext. Price
		FEED, REFLUX, BOTTOMS, DISTILLATE		
4	JFLOW	P/N 3-4623FTCTFTLD - 1/2" FNPT, 3-PIECE, WCB CARBON STEEL BALL VALVE B16.34, 316SS TRIM, TFM 1600 SEALS, CTFM 4215 SEATS, LOCKING HANDLE, SELF ADJUSTING STEM PACKING, 2000,WOG, ISO DM F03/04 9MM SQ STEM, AND INCLUDES * BA-03-VB-SL-B1-S6 - SS316 SLOT BALL WITH 1/16" SLOT * 3" HIGH BRACKER AND COUPLER * JFE-T20-25-V2-0/10-H - ELECTRIC ACTUATOR - 24VDC, 10 SECOND / 90 DEG, MODULATING WITH 0 TO 10VDC TRANSMITTER, 177 IN-LBS, 11MM FEMALE SQUARE DRIVE, F03/F04/F05, ALUMINUM HOUSING, COMPARTMENT HEATER, IP67, TYPE 4-4X, AND 4 METER LONG CABLE * ASSEMBLE, TEST, AND CALIBRATE *	$992.31	$3,969.24
		STEAM (MAIN), STEAM BY-PASS		
2	JFLOW	P/N 4-4623FTCTFTLD - 3/4" FNPT, 3-PIECE, WCB CARBON STEEL BALL VALVE B16.34, 316SS TRIM, TFM 1600 SEALS, CTFM 4215 SEATS, LOCKING HANDLE, SELF ADJUSTING STEM PACKING, 2000,WOG, ISO DM F04/05 11MM SQ STEM, AND INCLUDES * BA-04-VB-15-A1-S6 - SS316 V-BALL WITH 15 DEG V * 3" HIGH BRACKER AND COUPLER * JFE-T20-10V2-0/10-H-FS - FAIL SAFE ELECTRIC ACTUATOR - 24VDC, 10 SECOND / 90 DEG, MODULATING WITH 0 TO 10VDC TRANSMITTER, 177 IN-LBS, 14MM FEMALE SQUARE DRIVE, F03/F04/F05, ALUMINUM HOUSING, COMPARTMENT HEATER, SUPER CAPACITOR FAIL SAFEIP67, TYPE 4-4X, AND 4 METER LONG CABLE * ASSEMBLE, TEST, AND CALIBRATE *	$1,223.08	$2,446.16
		AVAILABILITY: 2 - 3 WEEKS ARO - FOB CINCINNATI, OH.		
			Total	$6,415.40

Cincinnati, OH	Columbus, OH	Cleveland, OH	Indianapolis, IN	Louisville, KY
Phone: 513-583-4800	Phone: 740-477-2557	Phone: 216-226-5533	Phone: 317-704-7280	Phone: 502-423-1946
Fax: 513-583-4815	Fax: 740-474-5493	Fax: 216-226-6992	Fax: 317-704-7288	Fax: 502-423-8888

Discover the Fischer Process Industries Home Page at: http://www.fischerprocess.com

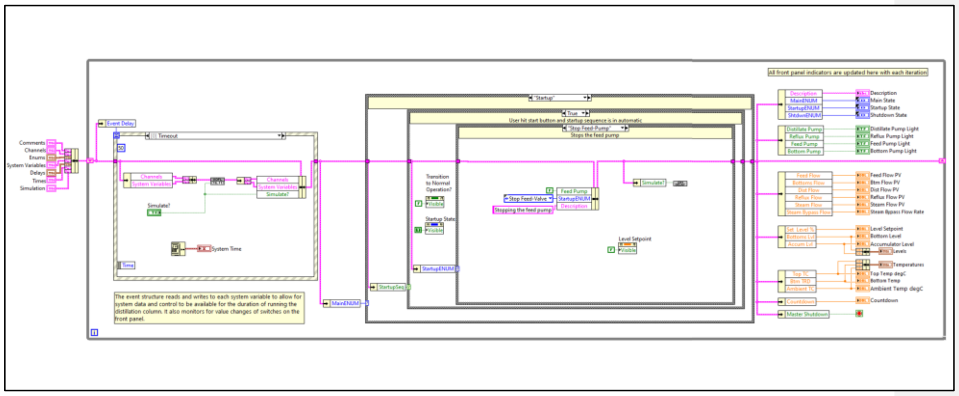
TERMS & CONDITIONS/WARRANTY

A CREDIT CHECK HAS NOT BEEN PERFORMED PRIOR TO QUOTE PREPARATION. COMPANY'S STANDARD TERMS ARE NET 20 DAYS. FINAL PAYMENT TERMS WILL BE DETERMINED WHEN ORDER IS PLACED.
QUOTATION IS BASED ON ACCEPTANCE OF FISCHER PROCESS INDUSTRIES TERMS & CONDITIONS.
A COPY CAN BE VIEWED AT: http://www.fischerprocess.com/assets/PDFs/terms_conditions.pdf

ALL CLAIMS AND SHORTAGES MUST BE REPORTED WITHIN 10 DAYS AFTER RECEIPT OF PRODUCT. NO RETURNS AFTER 30 DAYS.

Fischer Process Industries   Quote # QTEQ36405   Page 1   1 of 1

APPENDIX IV: UPDATED LABVIEW CODE

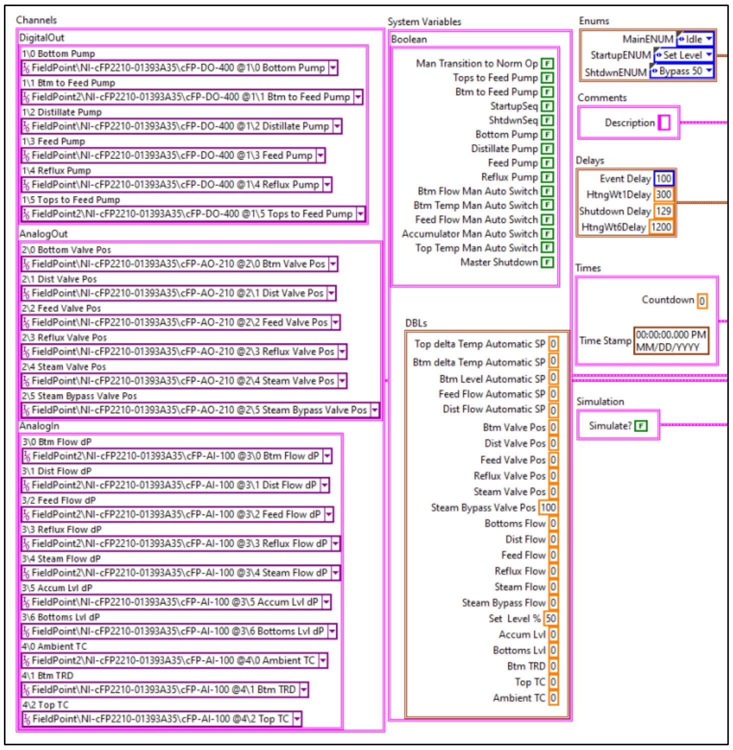


The event structure reads and writes to each system variable to allow for system data and control to be available for the duration of running the distillation column. It also monitors for value changes of switches on the front panel.

User hit start button and startup sequence is in automatic

All front panel indicators are updated here with each iteration

Figure 65: Block diagram of Updated LabVIEW Code with State Machine Architecture



Channels

DigitalOut

1\0 Bottom Pump
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\0 Bottom Pump
1\1 Btm to Feed Pump
FieldPoint2\NI-cFP2210-01393A35\cFP-DO-400 @1\1 Btm to Feed Pump
1\2 Distillate Pump
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\2 Distillate Pump
1\3 Feed Pump
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\3 Feed Pump
1\4 Reflux Pump
FieldPoint\NI-cFP2210-01393A35\cFP-DO-400 @1\4 Reflux Pump
1\5 Tops to Feed Pump
FieldPoint2\NI-cFP2210-01393A35\cFP-DO-400 @1\5 Tops to Feed Pump

AnalogOut

2\0 Bottom Valve Pos
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\0 Btm Valve Pos
2\1 Dist Valve Pos
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\1 Dist Valve Pos
2\2 Feed Valve Pos
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\2 Feed Valve Pos
2\3 Reflux Valve Pos
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\3 Reflux Valve Pos
2\4 Steam Valve Pos
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\4 Steam Valve Pos
2\5 Steam Bypass Valve Pos
FieldPoint\NI-cFP2210-01393A35\cFP-AO-210 @2\5 Steam Bypass Valve Pos

AnalogIn

3\0 Btm Flow dP
FieldPoint2\NI-cFP2210-01393A35\cFP-AI-100 @3\0 Btm Flow dP
3\1 Dist Flow dP
FieldPoint2\NI-cFP2210-01393A35\cFP-AI-100 @3\1 Dist Flow dP
3/2 Feed Flow dP
FieldPoint2\NI-cFP2210-01393A35\cFP-AI-100 @3\2 Feed Flow dP
3\3 Reflux Flow dP
FieldPoint2\NI-cFP2210-01393A35\cFP-AI-100 @3\3 Reflux Flow dP
3\4 Steam Flow dP
FieldPoint2\NI-cFP2210-01393A35\cFP-AI-100 @3\4 Steam Flow dP
3\5 Accum Lvl dP
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @3\5 Accum Lvl dP
3\6 Bottoms Lvl dP
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @3\6 Bottoms Lvl dP
4\0 Ambient TC
FieldPoint2\NI-cFP2210-01393A35\cFP-AI-100 @4\0 Ambient TC
4\1 Btm TRD
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @4\1 Btm TRD
4\2 Top TC
FieldPoint\NI-cFP2210-01393A35\cFP-AI-100 @4\2 Top TC

System Variables

Boolean

Man Transition to Norm Op	F
Tops to Feed Pump	F
Btm to Feed Pump	F
StartupSeq	F
ShtdwnSeq	F
Bottom Pump	F
Distillate Pump	F
Feed Pump	F
Reflux Pump	F
Btm Flow Man Auto Switch	F
Btm Temp Man Auto Switch	F
Feed Flow Man Auto Switch	F
Accumulator Man Auto Switch	F
Top Temp Man Auto Switch	F
Master Shutdown	F

DBLs

Top delta Temp Automatic SP	0
Btm delta Temp Automatic SP	0
Btm Level Automatic SP	0
Feed Flow Automatic SP	0
Dist Flow Automatic SP	0
Btm Valve Pos	0
Dist Valve Pos	0
Feed Valve Pos	0
Reflux Valve Pos	0
Steam Valve Pos	0
Steam Bypass Valve Pos	100
Bottoms Flow	0
Dist Flow	0
Feed Flow	0
Reflux Flow	0
Steam Flow	0
Steam Bypass Flow	0
Set Level %	50
Accum Lvl	0
Bottoms Lvl	0
Btm TRD	0
Top TC	0
Ambient TC	0

Enums

MainENUM	Idle
StartupENUM	Set Level
ShtdwnENUM	Bypass 50

Comments

Description

Delays

Event Delay	100
HtngWt1Delay	300
Shutdown Delay	129
HtngWt6Delay	1200

Times

Countdown 0

Time Stamp 00:00:00.000 PM MM/DD/YYYY

Simulation

Simulate? F

Figure 66: Expanded Master Cluster Initialization Variables

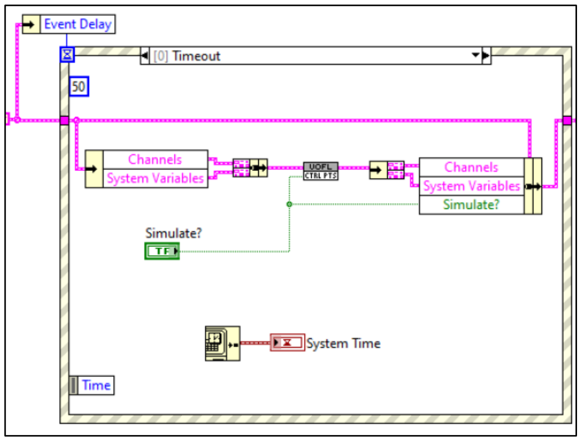


Figure 67: Timeout Event to Update FieldPoint Channels or Simulated Variables with Control Points SubVI

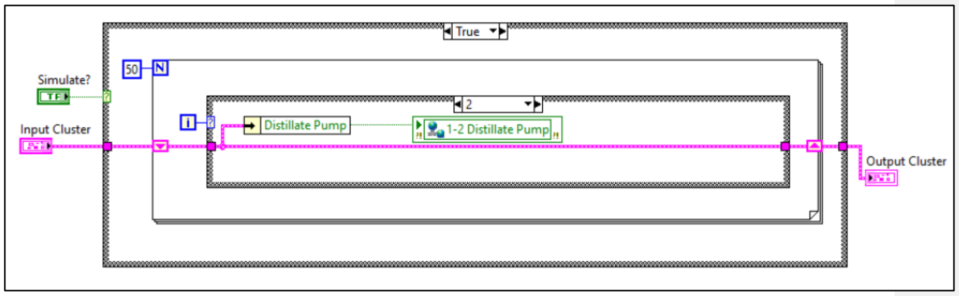


Figure 68: Control Points SubVI for Simulated Signal

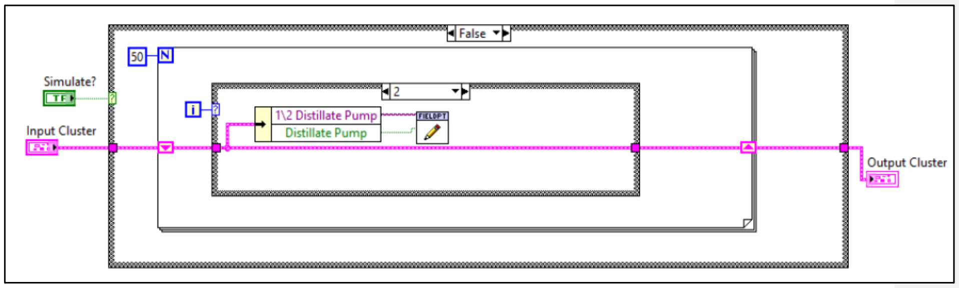


Figure 69: Control Points SubVI for Actual Signal from FieldPoint Channel

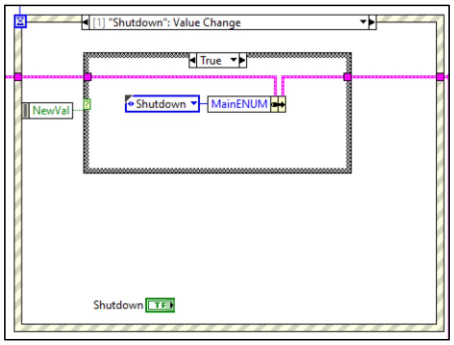


Figure 70: Example of Event Structure Monitoring for Control Value Change to Move the System to Shutdown State

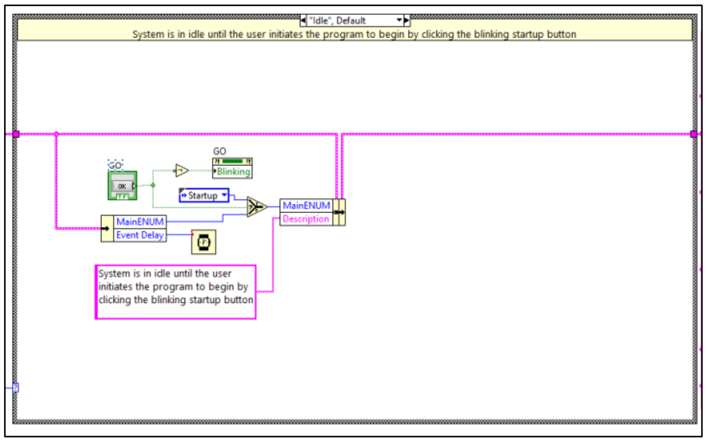

"Idle", Default
System is in idle until the user initiates the program to begin by clicking the blinking startup button
GO
GO
Blinking
Startup
MainENUM
Event Delay
MainENUM
Description
System is in idle until the user initiates the program to begin by clicking the blinking startup button

Figure 71: Idle State

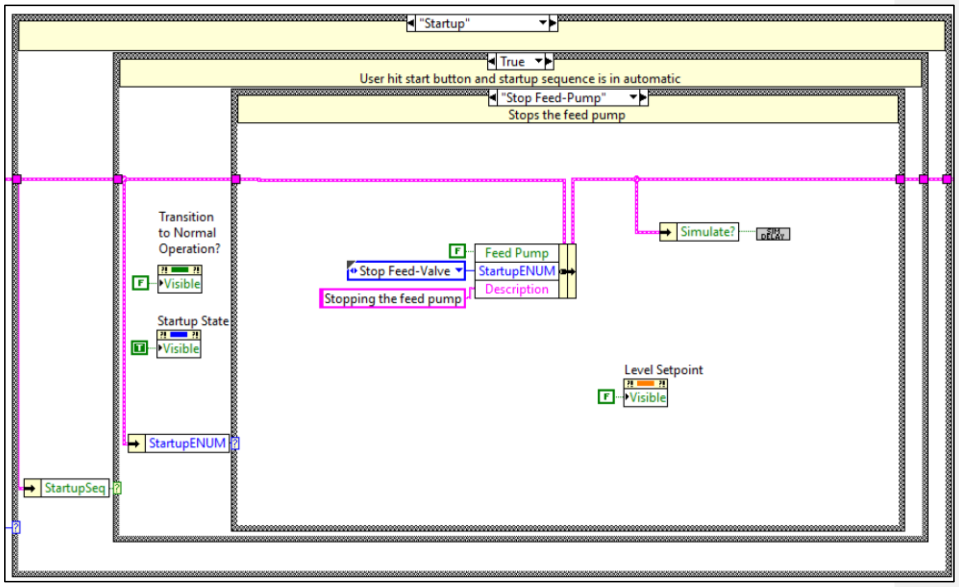


Figure 72: Example of Automatic Startup Case

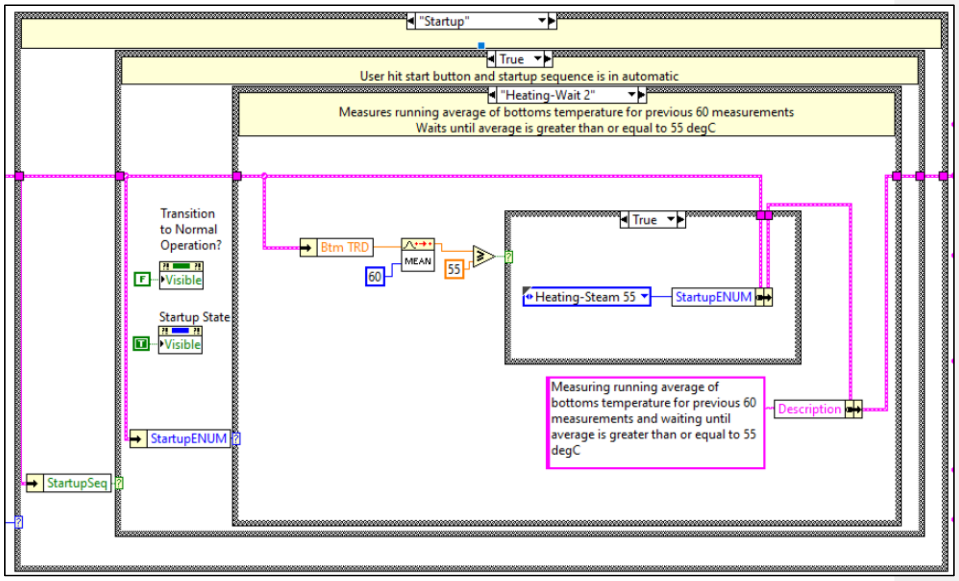

"Startup"
True
User hit start button and startup sequence is in automatic
"Heating-Wait 2"
Measures running average of bottoms temperature for previous 60 measurements
Waits until average is greater than or equal to 55 degC
Transition to Normal Operation?
Visible
Startup State
Visible
StartupENUM
StartupSeq
Btm TRD
MEAN
60
55
True
Heating-Steam 55
StartupENUM
Measuring running average of bottoms temperature for previous 60 measurements and waiting until average is greater than or equal to 55 degC
Description

Figure 73: Example of Automatic Startup Case Waiting to Progress Until a Temperature Setpoint is Met

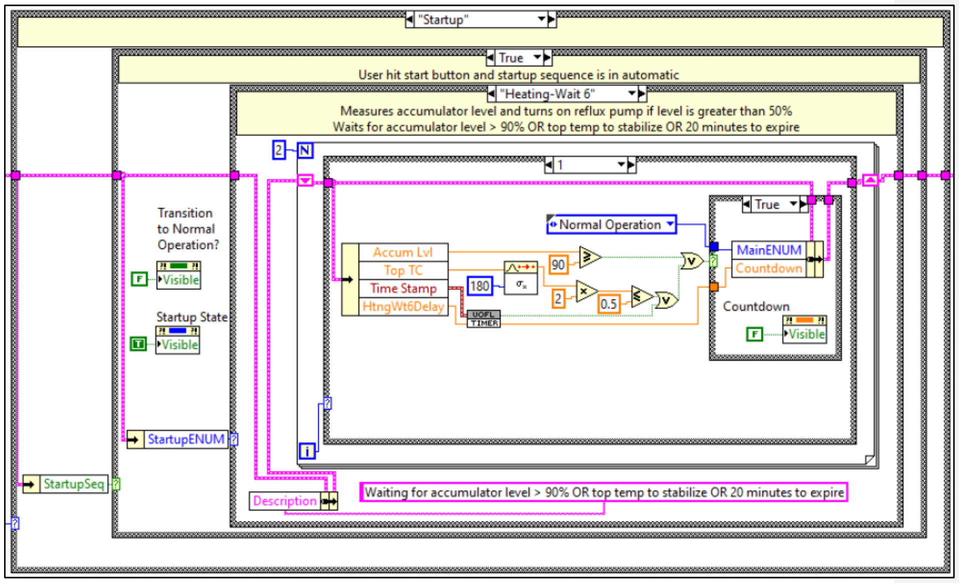

"Startup"
True
User hit start button and startup sequence is in automatic
"Heating-Wait 6"
Measures accumulator level and turns on reflux pump if level is greater than 50%
Waits for accumulator level > 90% OR top temp to stabilize OR 20 minutes to expire
Transition to Normal Operation?
Visible
Startup State
Visible
StartupENUM
StartupSeq
Normal Operation
Accum Lvl
Top TC
Time Stamp
HtngWt6Delay
180
90
2
0.5
MainENUM
Countdown
Visible
Description
Waiting for accumulator level > 90% OR top temp to stabilize OR 20 minutes to expire

Figure 74: Example of Final Automatic Startup Case Monitoring for One of Three Events to Occur to Transition to Normal Operation

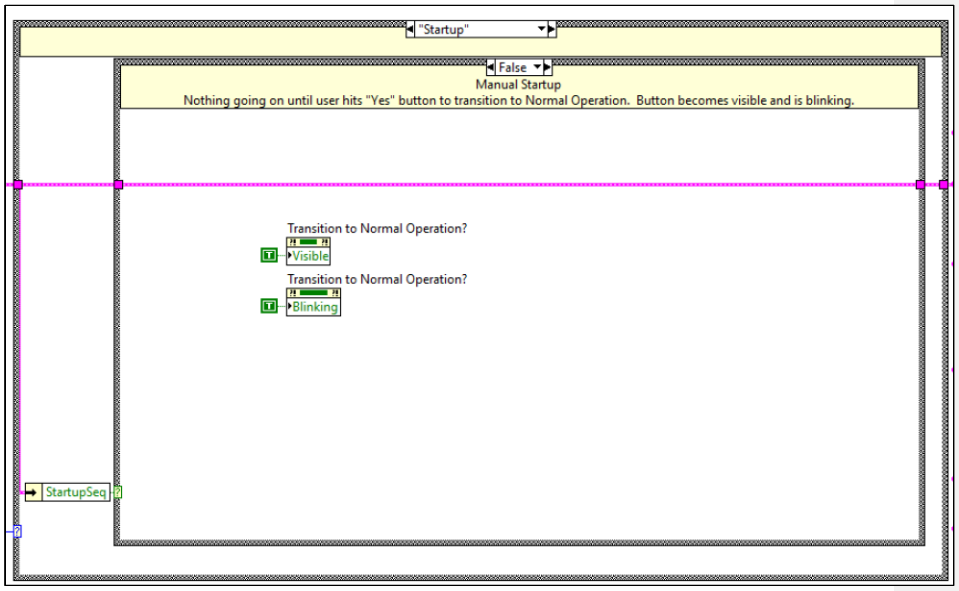

"Startup"
False
Manual Startup
Nothing going on until user hits "Yes" button to transition to Normal Operation. Button becomes visible and is blinking.
Transition to Normal Operation?
Visible
Transition to Normal Operation?
Blinking
StartupSeq

Figure 75: Manual Startup Case

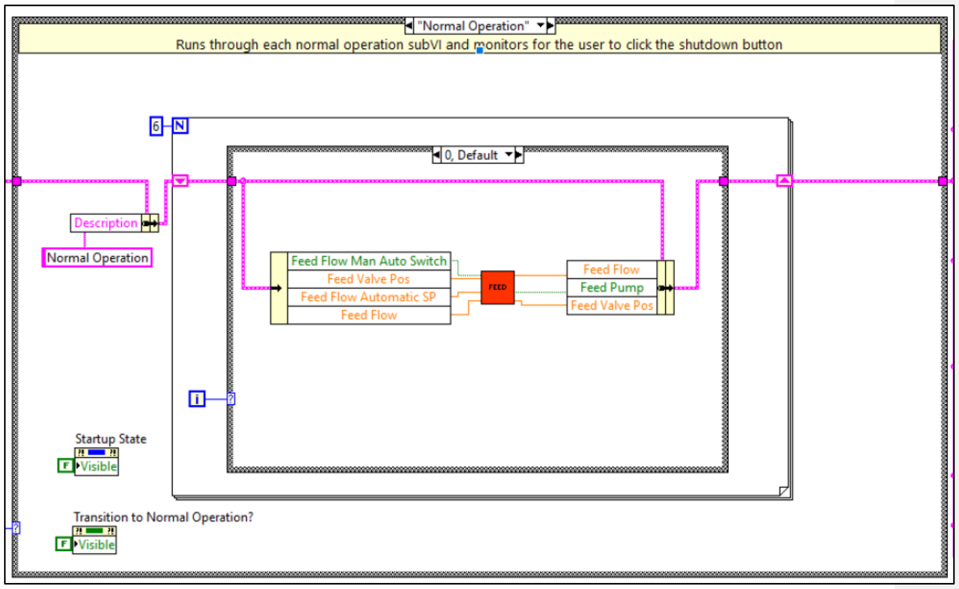


Figure 76: Normal Operation State

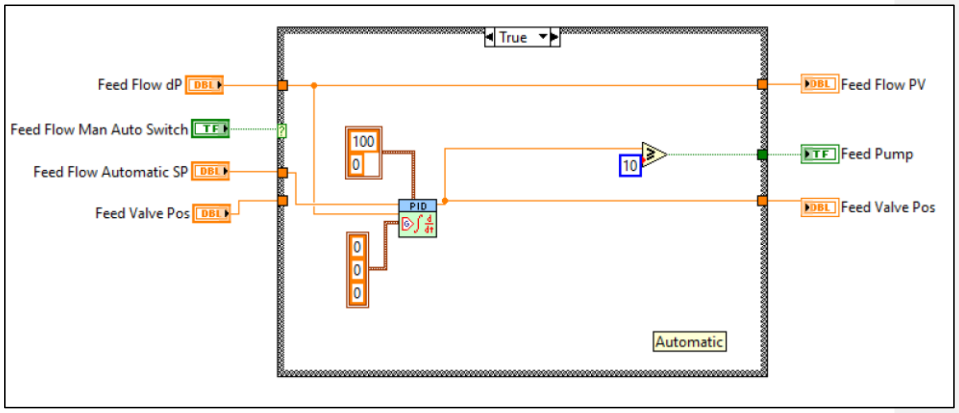


Figure 77: Example of Feed SubVI Executed in Normal Operation State

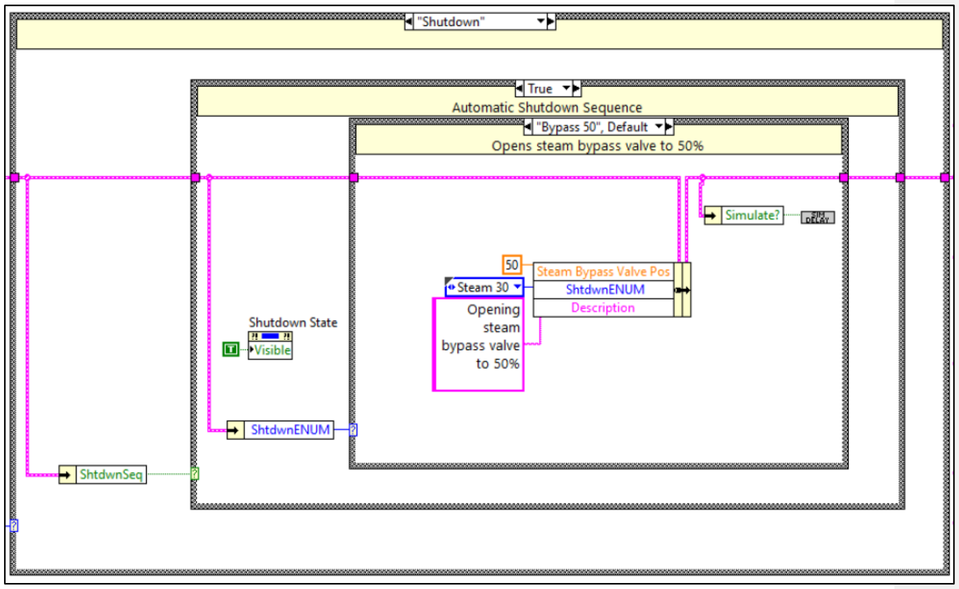


Figure 78: Example of a Case in the Automatic Shutdown State

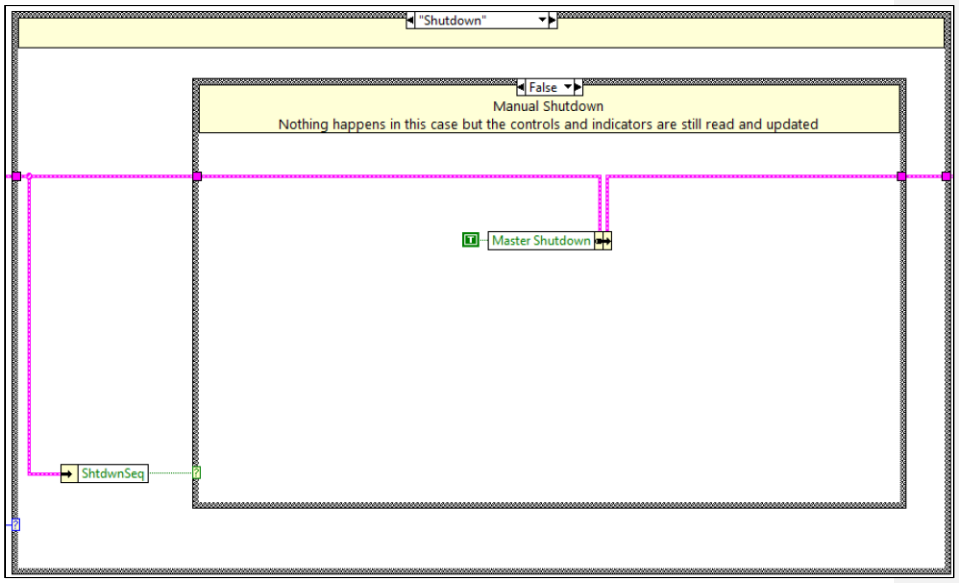

"Shutdown"
False
Manual Shutdown
Nothing happens in this case but the controls and indicators are still read and updated
Master Shutdown
ShtdwnSeq

Figure 79: Manual Shutdown State

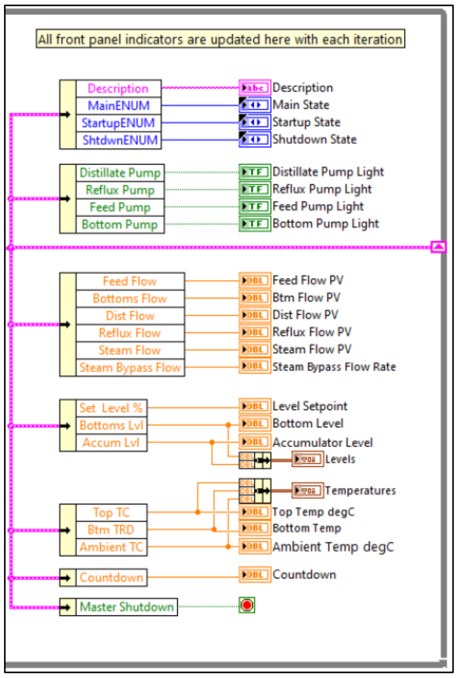


Figure 80: Indicators and Conditional Terminal Updated at the End of Each Loop Iteration

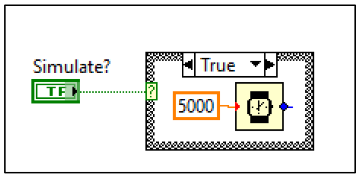


Figure 81: Simulation Delay SubVI Code

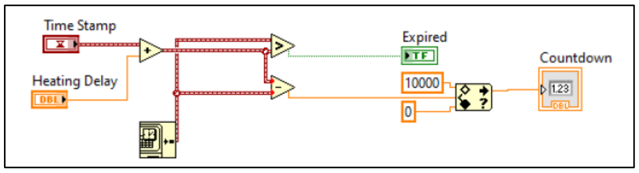


Figure 82: Timer SubVI Code

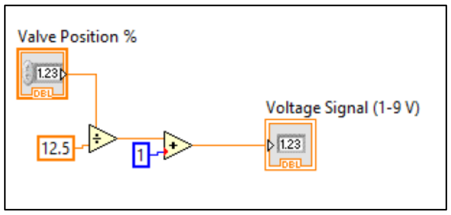


Figure 83: Example of Conversion SubVI Code

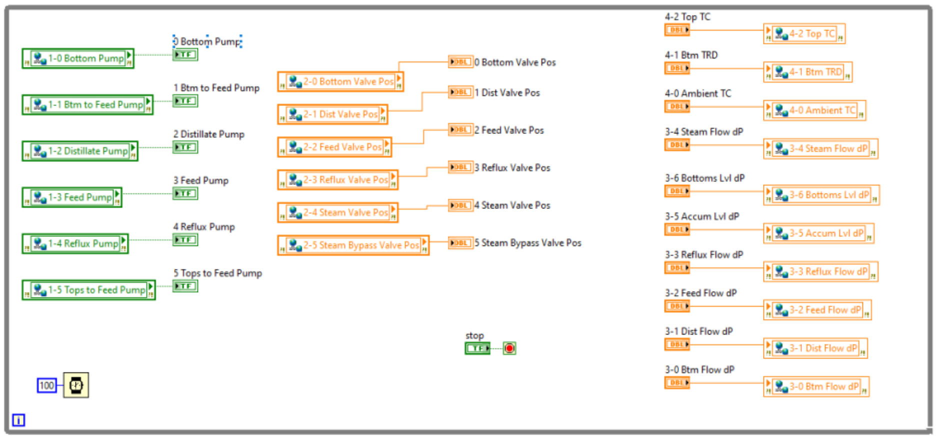


Figure 84: Simulator Block diagram Updating Network Variables

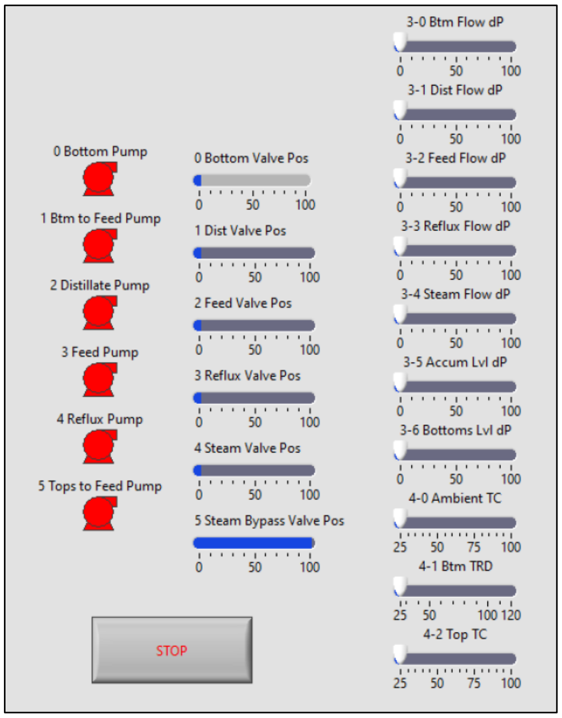


Figure 85: Simulator Front Panel Interface

APPENDIX V: C&I ENGINEERING REDESIGN DIAGRAMS

Equipment arrangement diagrams created by C&I Engineering for the redesign of the distillation process.

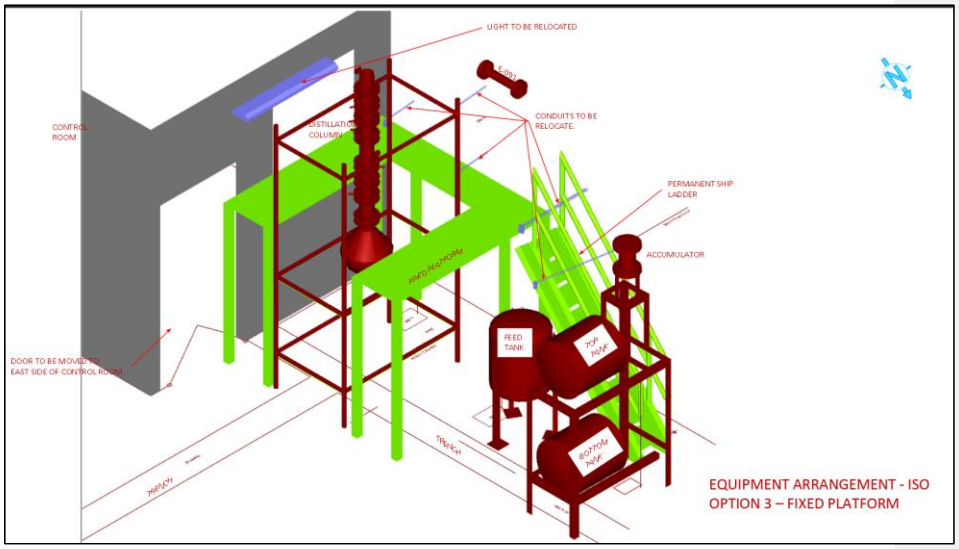


Figure 86: Isometric Diagram of Redesign Equipment Arrangement Developed by C&I Engineering

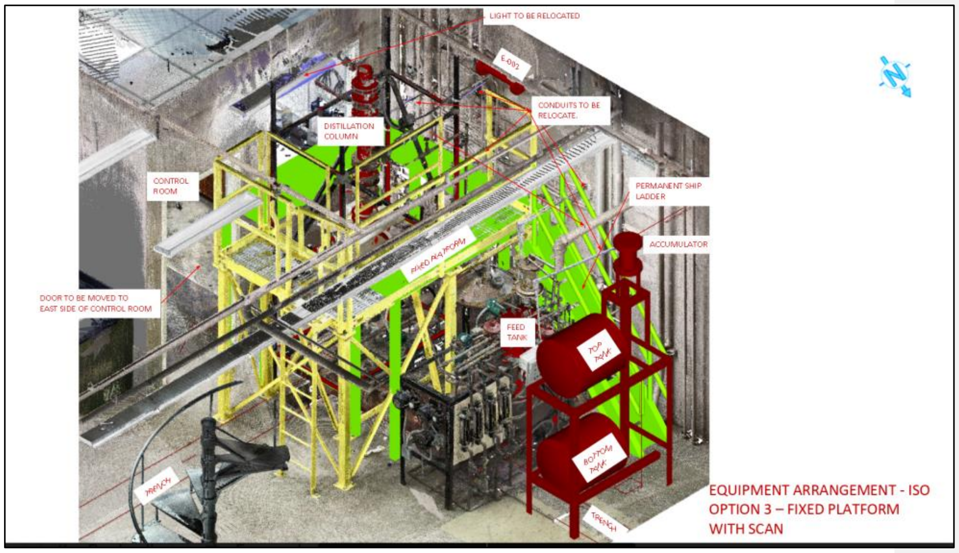


Figure 87: Isometric Diagram Layered Overtop of Equipment Scan Image Developed by C&I Engineering

References

1. Instruments, N., *Meaning of Different Wire Colors in LabVIEW*. 2019.
2. Seader, J.D., E.J. Henley, and D.K. Roper, *Separation Process Principles: Chemical and Biochemical Operations*. 3rd ed. 1998, New Jersey: John Wiley & Sons, Inc.
3. Ferguson, E.L. and M. Hegarty, *Learning With Real Machines or Diagrams: Application of Knowledge to Real-World Problems.* Cognition and Instruction, 1995. **13**(1): p. 129-160.
4. Konicek-Morgan, R. and P. Keeley, *Teaching Conceptual Understanding in Science*. 2015, National Science Teachers Association Press: Arlington, VA.
5. McCabe, W., J.C. Smith, and P. Harriot, *Unit Operations of Chemical Engineering*. 7th ed. 1956, New York: McGraw-Hill.
6. Turton, R., et al., *Analysis, Synthesis, and Design of Chemical Processes*. 5th ed. 2018: Pearson Education, Inc.
7. Hahnert, E., et al., *Unit Operations Lab II: Distillation Experiment*. 2019, University of Louisville.
8. Essick, J., *Hands-On Introduction to LabVIEW for Scientists and Engineers*. 3rd ed. 2009, New York: Oxford University Press.
9. Gerstle, J., *Unit Operations Lab II Lab Procedures*. 2019, January.
10. Norbert and Ray.R. *State Machine vs Sequence Structure for DAQ Read/Write and File I/O*. 2008 Web Forum [cited 4/12/2020]; Available from: https://forums.ni.com/t5/LabVIEW/State-Machine-vs-Sequence-Structure-for-DAQ-read-write-and-file/td-p/775424?profile.language=en.
11. Rüßmann, M., et al., *Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries*. 2015.