Development of a Programmable Integrated Switch Matrix (PrISM) through University-Industry Collaboration

Dr. Baha Jassemnejad, Federal Aviation Administration-CNI Airway Syatems Engineering Organization

Baha Jassemnejad was a Professor of Engineering and Physics and is a senior IEEE member as well as an ABET PEV. He is working as an Electronics Engineer VI for the FAA-Chickasaw Nation Industries, a contractor for National Airway Systems Engineering Organization.

Mr. Igor Ilikj
Jonathan Ryan Adams

Full time electrical engineering student, currently working on masters degree.

Mr. Neil Peery, CNI Aviation

Technical Manager, OK Communications Engineering Team, Chickasaw Nations Industries, Mike Monroney Aeronautical Center, Oklahoma City OK (August 2013 - present). • Leads 60+ contract personnel in providing 2nd level engineering support for Federal Aviation Administration (FAA) communications systems through directives, modifications, handbooks, technical issuances, and 24/7 field support.

Development of a Programmable Integrated Switch Matrix (PrISM) through University-Industry Collaboration

Abstract

This paper describes how student research and industry projects can benefit through joint university-industry collaboration by introducing new technology for replacing inefficient and outdated systems and software. This research project involves development of a fully customizable, user-defined hardware-software suite for automated signal routing with an open ended functionality profile. This intelligent switching system can be customized and employed in any industry where there is a need for programmable, timed, and/or simultaneous routing of analog or digital signals between devices. Potential applications of these automated switching systems include, but are not limited to: demarcation points, test floors, redundant backup systems, remote maintenance, etc. This system was designed via collaboration with the Federal Aviation Administration (FAA) Oklahoma Communications Engineering Team (OKCET) Laboratory and has found an immediate application as a large-scale switching system. The fundamental hardware unit for this system is the National Instruments (NI) PXI chassis with a NI SwitchBlock populated with matrix relay cards. The chassis can be deployed in any location, contributing to the robust nature of the design. The advantage of using an integrated NI system is modularity; the hardware can be easily tailored to the specific needs of each end user. Expansion and customization is accomplished with the addition of a wide spectrum of matrix relay cards. Matrix cards are available with a varying number of relays or switching points. The proposed system is controlled and automated by a customized virtual instrumentation (VI) that was developed using NI LabVIEW software environment and can be integrated with the PXI or function as an executable on a standalone desktop computer.

I. Introduction

The purpose of this project is to design an industry-specific product through university-industry collaboration. Such collaboration is beneficial to both parties. The students involved in the university project acquire skill sets that are valuable for their future endeavors beyond their education. On the other hand, industry representatives get to see progress being made on a project without heavy use of their resources. Such collaborations have been successfully accomplished via a wide array of university-industry interactions [1] and [2]. While these collaboration efforts do open up new opportunities, the afore-mentioned "costs" and "benefits" to these collaborations need to be taken in to consideration in order to optimize the interaction [3].

While there might be some apprehension over acquiring student help for time-sensitive and critical projects, there should be more confidence about approaching an educational institution for help on smaller, lower priority projects. Such projects might not be able to command the resources needed to get them off the ground. In those cases, using an existing university connection can gather a group of students to investigate the issue at hand. Additionally, this is being exploited by industry members who are seeking to acquire more active, hands-on methodology to train the students for their future employment [4]. This sort of collaboration would allow for the industry representatives to get a project off the ground with minimal resources, while maintaining a relationship with the student body that might be used in the future for similar endeavors.

A joint project like this can reap multiple benefits for the university component as well. Having a connection established can set up a multitude of projects that can allow the students to learn a variety of skills that can be implemented in the classroom or for their future employment after the completion of their studies [1] and [4]. A successful project could ensure that the industry representatives turn to academic resources for future projects as well. This continuing flow of knowledge was shown to be beneficial for both parties [5]. In turn, this would widen the range of accessible projects, allowing the students to learn more skills in a broader array of fields. In this case, a graduate student thesis is being tailored to involve university-industry collaboration. Such graduate student joint collaborations have been accomplished successfully in previous studies [2] and [6].

II. Objective

Large-scale switching is performed at facilities that utilize multiple communication/signal/test devices that must frequently be connected, disconnected, and rerouted. This facility could be a massive demarcation point at an airport, responsible for routing vital communications to and from in-flight aircraft; a testing facility that must perform large quantities of tests on various devices and/or signals; or any communication hub with large-scale routing needs. A system that would perform this large scale switching would need to be subjected to some requirements in order to have full functionality that would depend on the specific application. Such requirements range from power specifications on the switch relays and expandability options, to aesthetics of the graphical user interface and bulkiness of the hardware setup. By working with the industry representatives, these requirements can be addressed and incorporated into the design of the system for establishing a connection that has proven to be beneficial for both parties [1],[2] and [4].

The proposed automated switching system has an immediate use in conjunction with the Federal Aviation Administration (FAA) Oklahoma Communications Engineering Team (OKCET) Laboratory. The lab has a wide range of communication devices that need to have the capability of interfacing with each other. Having a large number of connection points requires a large number of copper wire connections. These hardwire connections need to be routed intelligently in order to achieve the functionality that is needed while occupying the minimum space and operating at a high capacity. As shown in Figure II.1, all of the devices in the lab are connected to the demarcation point. There are copper wire connections that can be routed from the demarcation point and taken to a switch matrix that should intelligently connect user defined devices together or connect to a phone line that can connect with the outside world. As shown by the example in Figure II.1, two different voice switches are connected together via the switch matrix on the bottom of the diagram. This connection is marked in red, connecting voice switch 1 radio 1 transmit to voice switch 2 radio 2 receive.

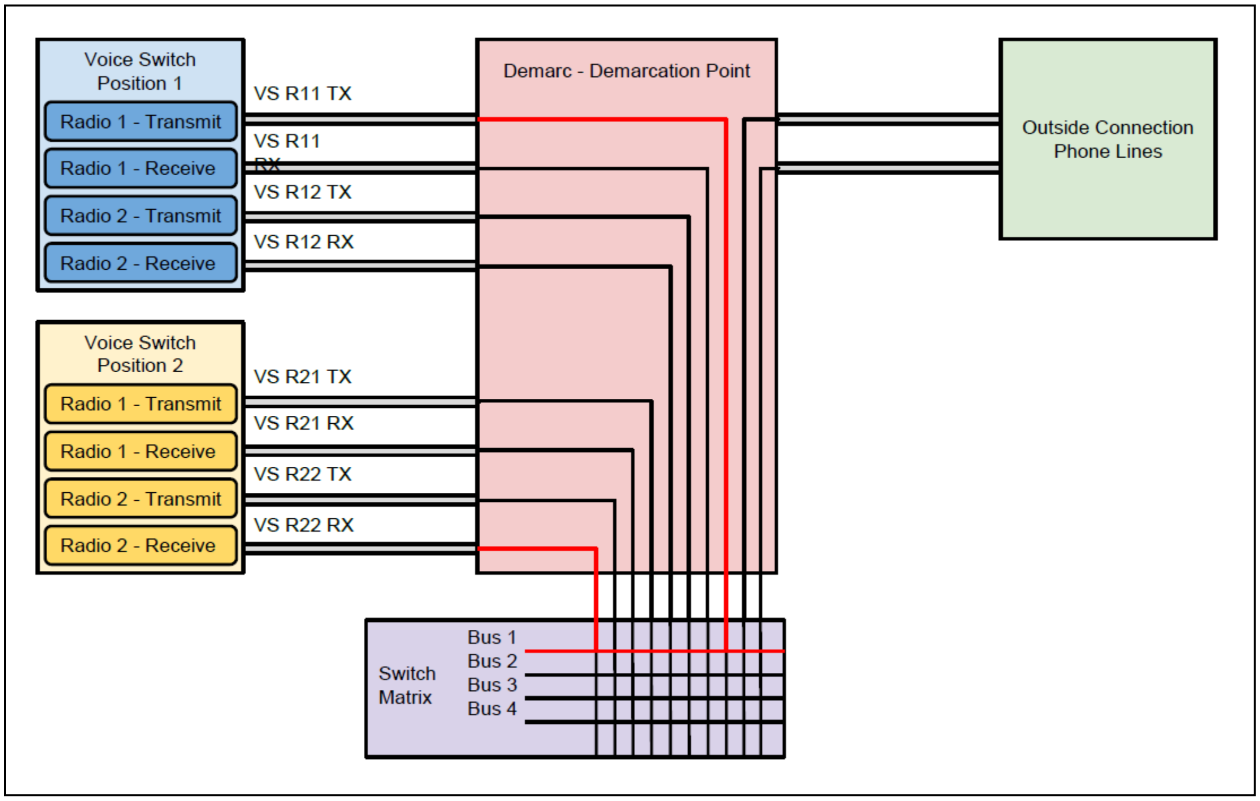


Figure II.1 Hardware Overview - two different voice switches are connected together via the switch matrix on the bottom of the diagram

The OKCET laboratory has a need for a matrix capable of routing between a large number of devices with the possibility of expansion into the thousands. Another requirement is the ability to achieve multiple simultaneous connections for testing different pieces of equipment at the same time. For these needs, in an A x B matrix, the number of rows, A, needs to be a smaller number representing the number of simultaneous connections allowed; whereas the number of columns B, needs to be a larger number representing the number of devices that would be able to interface with each other. In the case at hand, A can be a number like 8 or 16, while B needs to be able to grow into the thousands because of the number of devices the laboratory wants to have plugged in the switch matrix. Therefore, the hardware chosen to accomplish the task must have the ability to be easily expanded on in order to meet any future needs that the laboratory might have. Streamlining the future of an open-ended project has been an objective of other university-industry collaborations [5].

Having an integrated switch matrix that would allow for additional functionality to be explored is the final goal in this modular setup. But, in order to complete the requirements set by the OKCET laboratory, the setup needs to match the functionality of their current outdated switch matrix, as well as have the ability to receive any future modifications. Therefore, the hardware selected must be accepting to any other modular pieces that might need to perform other tasks for the industry end user.

The software used to incorporate the logic required to perform intelligent switching needs to be able to control the hardware that would be suited for the task as well as any additional hardware that might be added in the future. Making this match would be important, and set the tone for the entire project. Both the hardware and software need to be flexible to allow for expandability in terms of size and function, as well as be able to work in concert with each other. Due to their variety of hardware, and the ease of use of their software, National Instruments (NI) products were chosen for this project. NI hardware is intrinsically controlled by LabVIEW software, which is unique in its software development capabilities and is compatible with all of the NI products, which include a wide variety of communication and electrical engineering related equipment.

The NI hardware chosen for a large switch matrix was the PXI-2800 SwitchBlock Carrier. The SwitchBlock can carry matrix relay cards which carry the switching relays for the assembly of a large switch matrix. The one represented in Figure II.2 is shown with all six of the card slots filled with matrix relay cards.

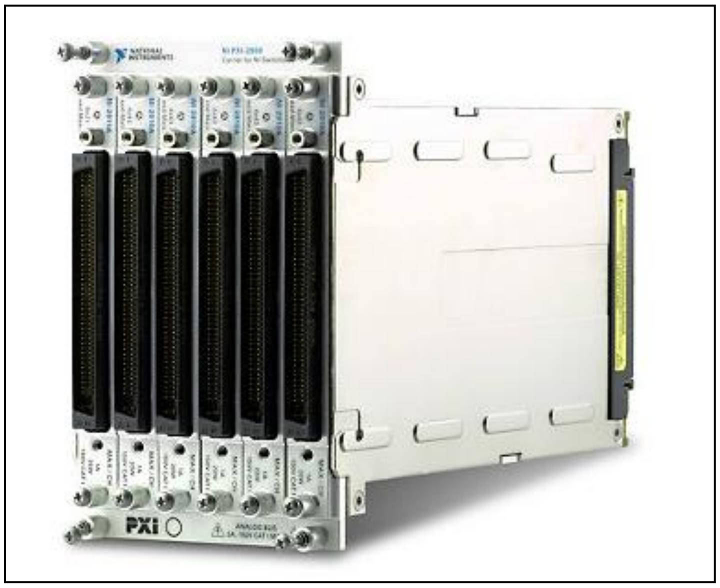


Figure II.2 PXI-2800 - all six of the card slots filled with matrix relay cards.

The base for a PXI system is a PXI chassis, which can vary in the number of slots available. For a large system, the largest possible chassis is recommended, up to 18 slots. These slots can be taken up by different modules, depending on the need of the system. The SwitchBlock carrier takes up 4 slots on a PXI chassis, and multiple switch blocks are allowed to be connected together. Each SwitchBlock carrier has 6 slots that can be occupied by matrix relay cards. The matrix relay cards also have a variety of options to choose from. There are multiple resources from NI on building large scale matrices [7]. Since the system requires a connection to multiple devices, a column expanding configuration as pictured in Figure II.3, shown in the white paper Matrix Expansion Guide by NI [8].

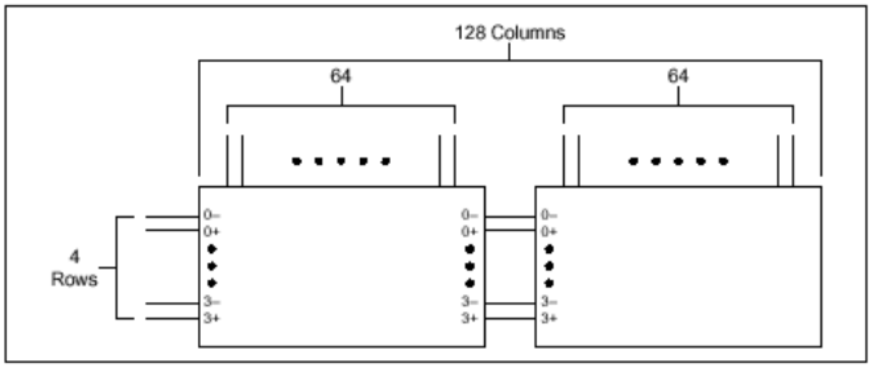


Figure II.3. Column expanding configuration

There are multiple considerations that need to be taken when choosing the type of relays and the switching logic in a switch matrix [9], [10] and [11]. NI has an array of switch matrix cards with reed relays that serve as switching points. Reed relays are commonly used as switching points for switch matrices [12] and [13]. The other available option is electromechanical relays, which have their advantages in higher current and power allowance, while occupying more space than the reed relays [14] and [15]. After considering the previously mentioned white papers from NI [7] and [8], as well as consulting with the industry end user for the planned product, a decision was made to go with the electromechanical relays over the reed relays. The OKCET laboratory specifications required to match the specifications and functionality of their switch matrix system. One of these specifications is an allowed switching current of up to 2 A. This higher current specification is only available with the bulkier electromechanical relays. There are two different electromechanical relays available that are SwitchBlock compatible, 4x71 and 8x34. The 8x34 card was chosen, because it allows for a minimum of 8 simultaneous connections at any time. The predicted load of the OKCET laboratory end users would likely require more than 4 simultaneous connections at any given time. These specifications are listed in Table II.1.

Hardware Specifications - NI 2834		
	Prototype	Projected Use
Number of Device Connections	34	1000+
Number of Bus Lines	8	8
Max Switching Current	2.0 A	2.0 A
Max Switching Power	60 W	60 W
Switch Relay Type	EM	EM

Table II.1. NI 2834 Hardware Specifications

A set of switch requirements was created in order to address the software benchmarks that needed to be met. These requirements will govern the development of an algorithm to run this system. Additionally, setting up a palette of niceties for the industry end user is also important for the success of the proposed system. These include, but are not restricted to, a login system for an administrator and users, a screen to display active connections and possible connections to be made, and report generation. These requirements are listed in Table II.2.

Algorithm Software Specifications	
	Prototype
Device Choice	• Graphical (selecting the devices from images that are grouped by device and signal type). • Text tree (selecting the devices from a list forming a text tree where each branch is a device or signal type).
Switch Relay Positions	• Graphical (pictorial representations of current configuration, showing devices and buses in use). • Text/Table (tabular representation of the switch configuration, listing the devices for each connection and the bus line they occupy).
Routing	• Intelligent routing where the algorithm chooses the first free bus for every new connection. • Options for locking down buses or latching another device on an existing connection. • All buses active notification.
Programmable Connections	• Timed connect/disconnect • Triggered connect/disconnect
Log	• Create a log of connections made • Report generation on user command
Purge	Administrator only purge all connections functionality

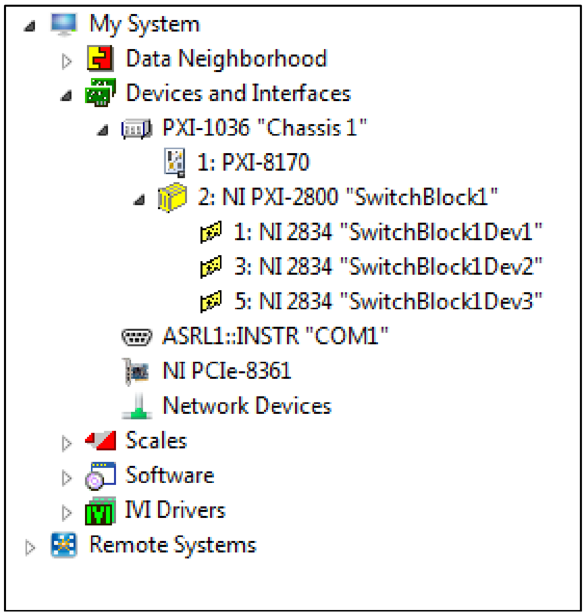
Table II.2. Software Specifications

III. Hardware Simulation

The system will perform switching at a large scale that would also have a varying number of devices connected to the matrix. This would require a flexible design that would allow for the end user to change the parameters of the system. The basic parameter is the number of devices connected, which would dictate the number of physical connections that need to be in place for every device that needs to be connected to the grid. In a matrix AxB (A is rows, B is columns), a design that would require a fixed group of devices to be connected to a different group containing a distinct set of devices would be best suited for a setup where the rows would be filled out with devices from group A, while columns would be devices from group B. In other words, the size of A and B would be directed by the amount of devices in each group. Another case is where all of the devices need to have an ability to be connected to each other at any given time. In this case, every column would be occupied by a different device, while the rows can be used to have multiple connections active at the same time. In other words, A will be the amount of devices that need to connect to the matrix, and B would be the number of simultaneous connections available [9], [10] and [11]. Communication with the industry representatives for this system is paramount for designing the specifications of the switch matrix. As specified by the NI expansion guide [7], the alignment on Figure II.2 was chosen.

As specified in the objective section, the OKCET laboratory required a switch matrix with the capability of expanding the number of columns into the thousands. Another aspect that needed to be kept in mind was the possibility of adding other functionalities, such as signal generation, measurement, power capabilities, etc. In the objective section it was mentioned that a PXI platform was chosen in order to meet these end user specifications. PXI platforms are created by NI, and provide the modular instrument platform needed for the completion of this project [16]. The PXI chosen for the prototype was a NI PXIe-1073, a 5-slot chassis that can hold a single NI PXI-2800 SwitchBlock that is enough for the proof of concept.

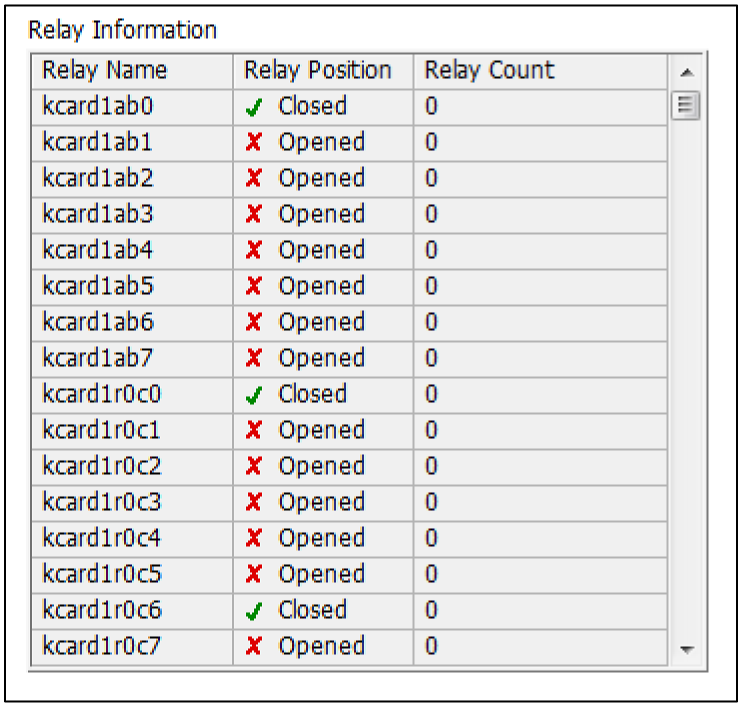
An additional functionality that NI provides is the capability to simulate the hardware before purchasing. This would ensure that the hardware is compatible and would allow for testing before any purchases are made. In an educational environment, this provides a useful tool in order to get the funds needed to purchase a prototype. Simulation with NI products is done in National Instruments Measurement Automation Explorer (NI MAX). The service provides a simple way to check the integrity of a designed system, and allows for a test of its full functionality [17], [18] and [19]. In this case, the PXI-1036 chassis was used, which is a similar chassis to the NI PXIe-1073. As seen in Figure III.1, there is a NI PXI-2800 SwitchBlock that occupies 4 of the slots available, with the added default PXI-8170 embedded controller. The embedded controller is the most common architecture available, and it serves as a feature that dictates processing speed, streaming to disk, etc. [16]. The SwitchBlock is filled with 3 NI 2834 cards which have 34 slots each. In turn, this simulated configuration has a functioning 8x102 matrix, which means that there is room for up to 8 simultaneous connections and 102 different devices.



```
My System
  Data Neighborhood
  Devices and Interfaces
    PXI-1036 "Chassis 1"
      1: PXI-8170
      2: NI PXI-2800 "SwitchBlock1"
        1: NI 2834 "SwitchBlock1Dev1"
        3: NI 2834 "SwitchBlock1Dev2"
        5: NI 2834 "SwitchBlock1Dev3"
    ASRL1::INSTR "COM1"
    NI PCIe-8361
    Network Devices
  Scales
  Software
  IVI Drivers
Remote Systems
```

Figure III.1. NI MAX PXI Simulation

Additionally, in NI MAX, the user can open the specific cards and look at what state the switch point relays are in. As seen in Figure III.2, some relays have been closed in order to establish a connection across them. Figure III.2 represents the relay positions for the card SwitchBlock1Dev1 from Figure III.l. In this case, the relay for connecting bus 0 across cards is tripped, b0, as well as the relays for devices 0 and 6, which are connected together via bus 0, to relays c0 and c6.



Relay Information

Relay Name	Relay Position	Relay Count
kcard1ab0	✓ Closed	0
kcard1ab1	✗ Opened	0
kcard1ab2	✗ Opened	0
kcard1ab3	✗ Opened	0
kcard1ab4	✗ Opened	0
kcard1ab5	✗ Opened	0
kcard1ab6	✗ Opened	0
kcard1ab7	✗ Opened	0
kcard1r0c0	✓ Closed	0
kcard1r0c1	✗ Opened	0
kcard1r0c2	✗ Opened	0
kcard1r0c3	✗ Opened	0
kcard1r0c4	✗ Opened	0
kcard1r0c5	✗ Opened	0
kcard1r0c6	✓ Closed	0
kcard1r0c7	✗ Opened	0

Figure III.2. Relay Positions in NI MAX

A more helpful tool for visualizing the grid of the switch matrix, according to which connections are made in the NI MAX simulation, can be seen in Figure III.3. This representation shows a graphical grid view of the SwitchBlock1Dev1 from Figure III.l. As it can be interpreted from the figure, there are 4 buses that are utilized: 0,1,2, and 7. Buses 0 and 7 are connected to the next card in line, SwitchBlock1Dev2, which allows for more devices to be connected on the already used bus. Some of the devices have been connected as well: c0 to c13 via bus 2 and c7 to c0 via bus 1, etc.

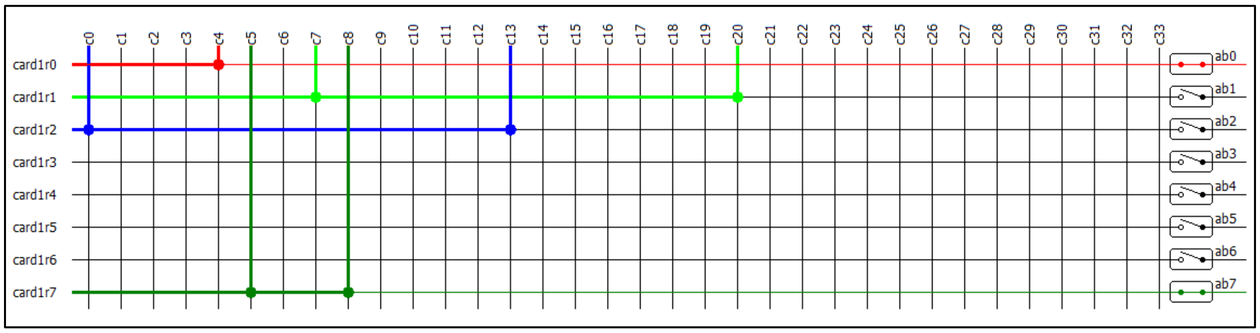
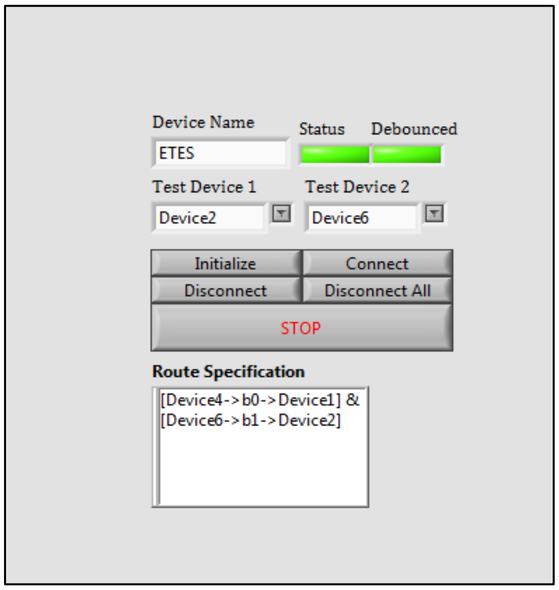

c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18 c19 c20 c21 c22 c23 c24 c25 c26 c27 c28 c29 c30 c31 c32 c33
card1r0
card1r1
card1r2
card1r3
card1r4
card1r5
card1r6
card1r7
ab0
ab1
ab2
ab3
ab4
ab5
ab6
ab7

Figure III.3. NI 2834 Card Test Panel- visualizing the grid of the switch matrix, according to which connections are made in the NI MAX simulation

IV Software Simulation

The modular integration aspect of this project also extends into the software section. The design of the PXI modular instrument was done in conjunction with LabVIEW, the NI software used for controlling NI instrumentation. Therefore, all of the programming done in LabVIEW had to follow suit and also be able to accept additional functionalities with a modular approach in the algorithm development.

The advantage of using the same developer for the software and hardware is their compatibility with other NI services. NI MAX allows for the simulation of the software in conjunction with the simulated hardware. Therefore, any programming done could be tested with NI MAX before any hardware was purchased.



[Device4->b0->Device1] &
[Device6->b1->Device2]

Figure IV.1. Simulated Front Panel

Figure IV.1 shows the front panel of the switch matrix. The user needs to put down the device name for the hardware that is used. In this example, the default name given in NI MAX is ETES. Next, the user needs to select the devices that will be connected, and click on the connect button. Any connections made show up on the bottom part of the front panel. Figure IV.2 shows device 4 has been connected to device 1 via bus 0, and device 6 has been connected to device 2 via bus 1. Any of these connections can be disconnected by clicking on the disconnect button. The Disconnect All button clears every connection that has been made.

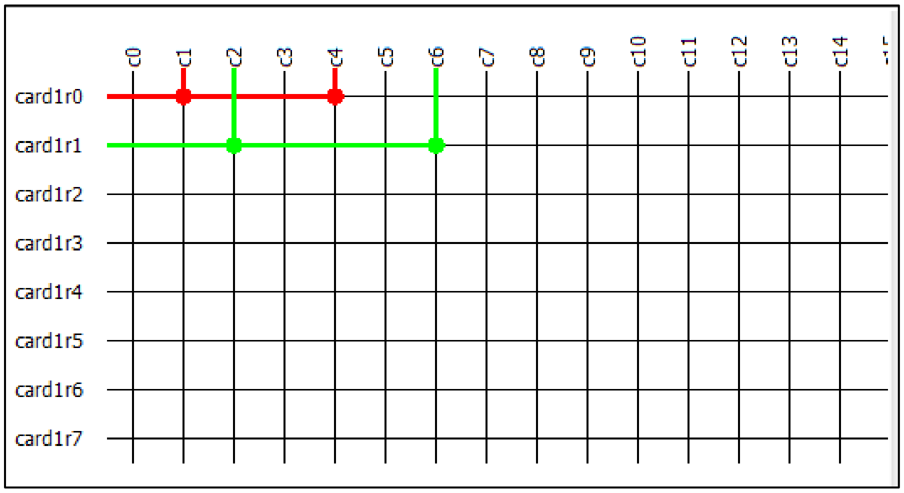


Figure IV.2- shows device 4 has been connected to device 1 via bus 0, and device 6 has been connected to device 2 via bus 1.

The diagram shown in Figure IV.2 is the display that can be seen in NI MAX in the Test Panels tab. This tab shows the state of the relays in the simulated device, in a list where every relay has a name and an open/close state, or in a diagram like Figure III.2, where every pathway is drawn in a different color depending on the bus line that is occupied. As stated previously and seen in Figure IV.2, device 4 has indeed been connected to device 1 via bus 0, and device 6 has been connected to device 2 via bus 1.

V Hardware Procurement and Setup

In order to demonstrate the functionality of a programmable integrated switch matrix, the prototype design was made to show the functionalities required at this stage of the project. The hardware requirements were presented in Table II.1.

The equipment was required via school funding. The PXI chassis purchased was a NI PXIe-1073, a 5 slot PXI model. The NI PXI-2800 SwitchBlock carrier requires 4 spots, so the 1073 PXI chassis was adequate in that regard. The card purchased was a NI 2834 8x34 card. These specifications were enough for a proof of concept prototype design, where all of the hardware requirements could be fulfilled. Figure V.1 shows the PXI chassis with the SwitchBlock with one NI 2834 card. The big cable that comes out of the card has the 34 pinouts for the copper connections that would be connected to the devices that would interact via the switch matrix.

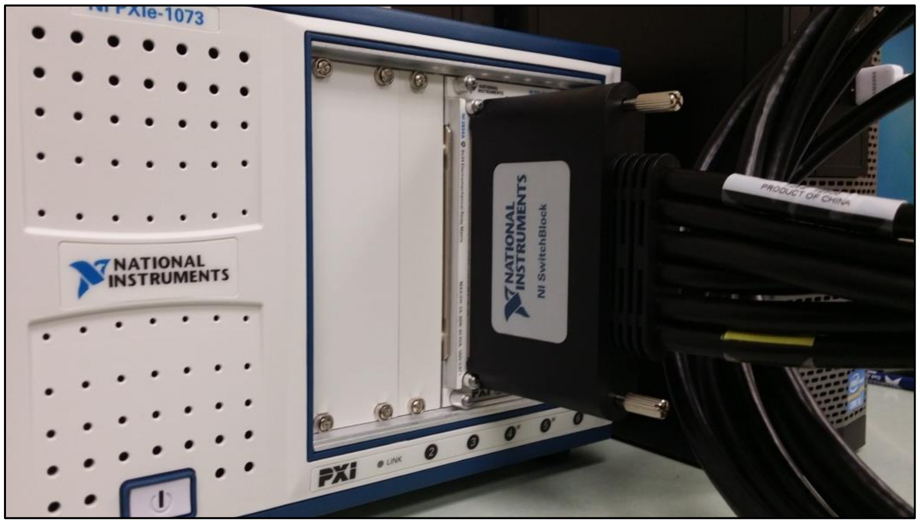


Figure V.1. PXI chassis with populated SwitchBlock

Figure V.2 shows the other end of the cable running from the NI 2834 card in Figure V.1. The copper connectors are connected to a ribbon cable that was used to perform some testing on the device in order to identify the usable ports for the prototype. The green copper connector panel is where the devices would be connected to the switch matrix, with only 34 of the possible connectors actually leading to a connection.

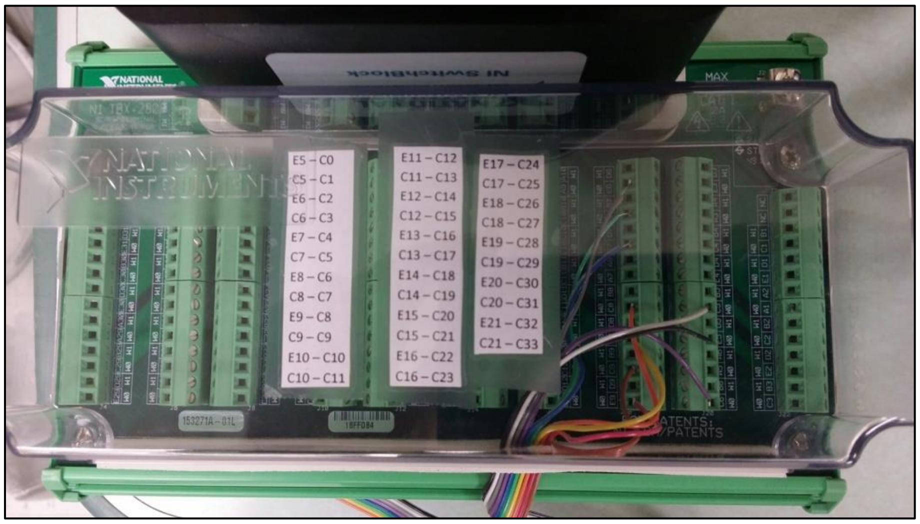


Figure V.2. Copper Connectors to Switch Matrix

VI Algorithm Development

One of the objectives of the project was to create a setup that is able to accept additional integrated functionalities, resulting in the "I" of the name PrISM. In order to leave this option, an open ended and template-oriented design had to be kept in mind when designing the algorithm. To achieve this functionality, there were some key points that needed to be accomplished for the success of the project:

- Configuration file(s) for different system parameters
- Program templates for future functionalities expansion
- Niceties for industry end user

Having a configuration file is important for the robustness of the algorithm. In designing a large switch matrix tailored for the specific needs of the end user, there are many parameters that need to be accounted for. A configuration file with these parameters would allow the end user to change only the content of the file, without having to alter the algorithm itself. These parameters would be the number of connections available, the names of the devices and their positions, the number of bus lines to allow for simultaneous connections, etc.

NI LabVIEW works by writing the logic into virtual instruments (VIs) [20]. These VIs are what controls the hardware itself. In order to be able to accommodate for the previously mentioned future development of added functionalities, all of the separate processes of this system were designed as separate subVIs. A subVI is a part of the NI hierarchical structure in programming, where a VI can be saved, and called in a different VI as a subVI [21]. This would allow for a modular design of a main VI, which can call all of the subVIs that hold different functions as they are needed by the end user.

Lastly, as a part of this joint university-industry collaboration, some niceties were provided to the end user. The first being a login system which would look at who is using the system at a specific time, allowing more options available for administrators versus a more limited menu for a regular user. This would allow for an administrator to purge connections which a user might have forgotten about in running longer tests, or adding new users into the system. Other features involve improving the design of the graphical user interface, making it more user-friendly and aesthetically pleasing.

The mathematical model of computation used to accomplish the modularity and computing needs was chosen to be a finite state machine. State machines have been successfully employed in designing virtual machines with a need to dynamically respond to a changing environment [22] and in designing faster and more responsive systems than usually available from vendors [23]. A state machine programmed in LabVIEW would have the capability to run continuously and pick and choose which functions to perform at different times, as dictated by the needs of the user. LabVIEW has intrinsically developed queue operations, which allows for the development of a queued state machine. In a queued state machine all of the tasks that are going to be accomplished are put in a queue by the producer, and as processing allows it, they are performed by a designated consumer. The diagram of a basic queued state machine in LabVIEW is presented in Figure VI.1.

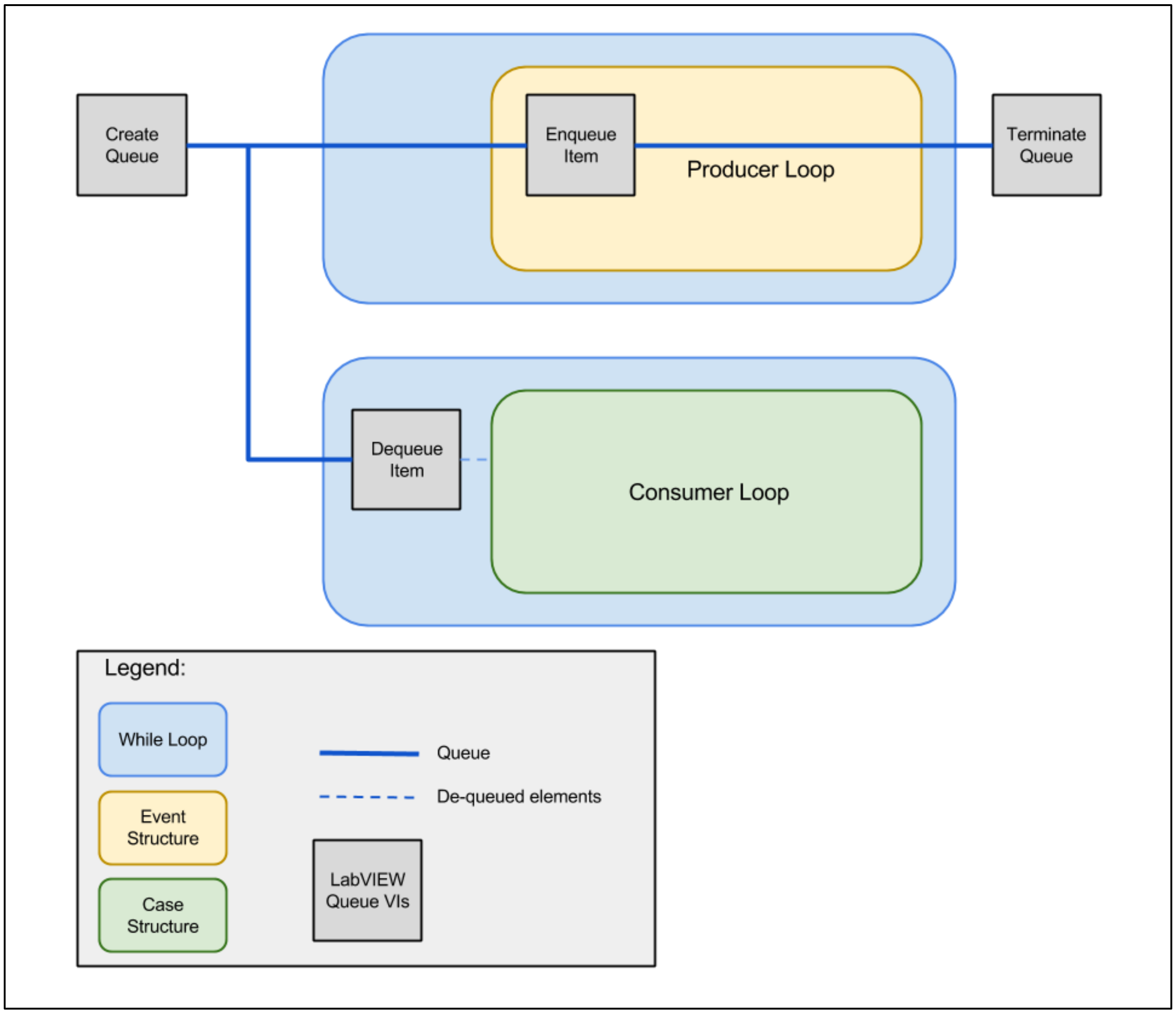


Figure VI.1. Basic Queued State Machine in LabVIEW

LabVIEW has developed intrinsic queue functionality, which allows for the usage of available VIs to perform the queue operations. The producer loop, as depicted in Figure VI.1, is an event structure inside of a while loop. The while loop is active for the entire time the application is up, or until there is an unexpected error which will terminate the queue, display a message and promptly close the application. The producer loop is triggered by different events, each event adding a queue item via the enqueue VI. These events can be user button clicks, errors, and timers. All of these elements are de-queued in the consumer loop. The consumer loop is a case structure nested in a while loop. The task that was queued by the producer is de-queued in the while loop of the consumer and executed. The case structure of the consumer has a number of different cases that depend on the tasks given by the producer. This ensures that every task given to the algorithm will be completed on a first come first serve basis. While this is the functionality of a basic queued state machine, a modified version needed to be designed to suit the needs of this project.

To accommodate the possible future applications that the end user might need, the state machine was altered to accept any number of VIs that can accomplish different functions. As previously mentioned, these functions can be a part of the current setup (connecting different devices, login logic), or future developments that might be a part of the project (signal generation, signal measurement, etc.). These VIs were placed in a sequence structure at the bottom of the algorithm, as pictured in Figure VI.2.

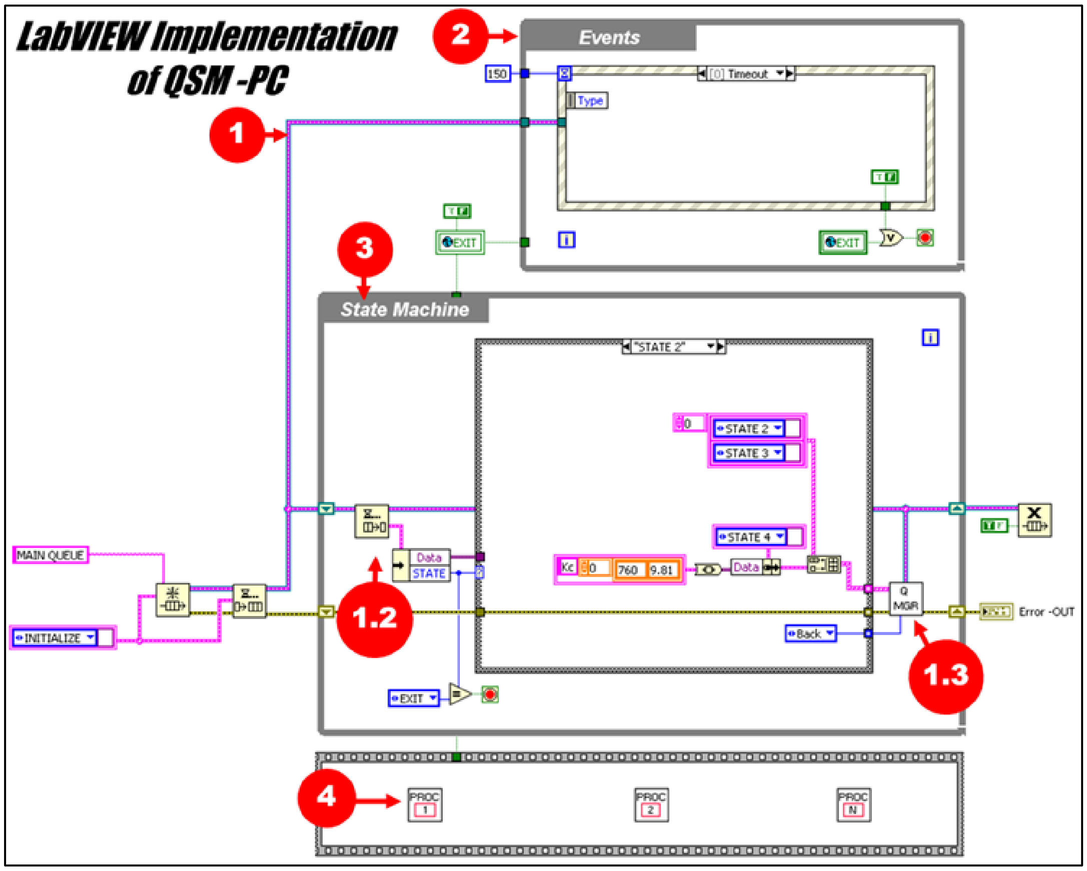
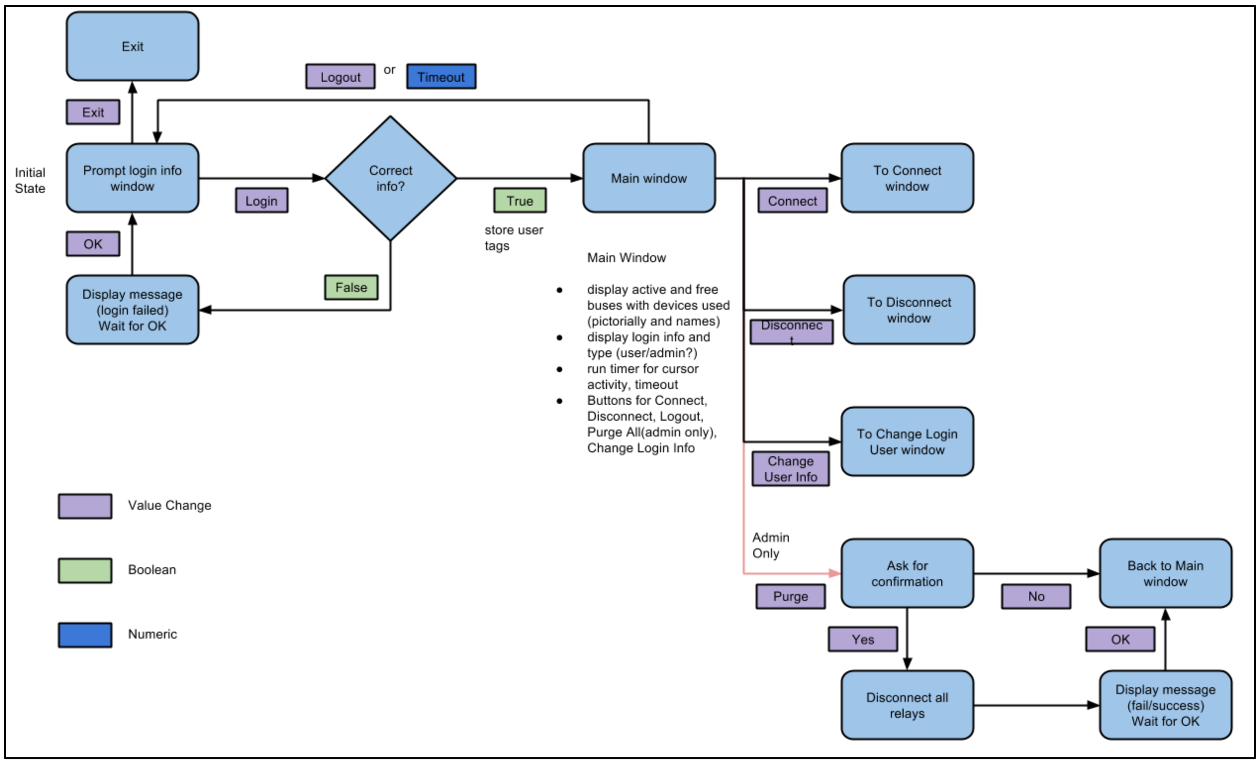


Figure VI.2. Queued State Machine example [24]

The differences in the more sophisticated structure used in the project are marked 1.3 and 4 in Figure VI.2. 1.3 is a queue manager subVI that has the function of changing the order of tasks performed. Some tasks might require the ability to be put in ahead of the pending tasks. An example for these emergencies might be an error or hitting the stop button. When either one of those is selected, the preference is for the program to stop. In those cases, a ring control was preselected with 'Front', which will mean that the case is automatically added to the front of the queue. The non-emergency cases, which would contain most of the normal functions required by the program, were preselected with the ring setting 'Back'. This means that these tasks will be performed on a first come, first serve basis, as most of those are in line with the normal function of the program, and don't need priorities. The structure marked as 4 is a flat sequence structure. A flat sequence structure in LabVIEW executes whatever is placed inside of it left to right, pixel by pixel. In this case any VIs that can perform multiple functions run in parallel, starting from left to right. The design of these VIs requires them to be running on idle the entire time until being called for to perform some function. In that case, any user input will distribute different states to the parallel processes needed, and these VIs dumped in the sequence structure will move away from the idle state. This can work for as many parallel processes as the final product might need; therefore, it can accommodate any future developments in the project.

As previously mentioned, for the scope of this stage of the project, there were two main functions that needed to be accomplished: connecting and disconnecting devices using a switch matrix, and providing a set of niceties for the industry end user. The following section will describe the logic in flow charts that were programmed in LabVIEW in order to operate the hardware.



Exit
Logout
or
Timeout
Exit
Initial State
Prompt login info window
Login
Correct info?
True
store user tags
Main window
Connect
To Connect window
OK
False
Display message (login failed) Wait for OK
Main Window
display active and free buses with devices used (pictorially and names)
display login info and type (user/admin?)
run timer for cursor activity, timeout
Buttons for Connect, Disconnect, Logout, Purge All(admin only), Change Login Info
Disconnect
To Disconnect window
Change User Info
To Change Login User window
Value Change
Boolean
Numeric
Admin Only
Purge
Ask for confirmation
No
Back to Main window
Yes
OK
Disconnect all relays
Display message (fail/success) Wait for OK

Figure VI.3. Main program flow chart

The main program has the modified queued state machine setup pictured in Figure VI.3. There are some guidelines for appropriately reading the flow charts pictured VI.3-VI.6. Every rectangular bubble represents an action that is performed by the program. Every diamond shape is a query that the algorithm has to go through to advance to the next stage. As specified, the different colors for the parameters represent different variable types, purple is a value change on a button (like clicking the OK or Cancel button in a simple dialog, or a choice between Connect, Disconnect, Change User Info, or Purge), green is a true or false boolean, and blue is a numeric value. Prior to the main window being displayed, there is a login menu that the user has to navigate in order to access the main window. According the credentials of the user, there are different options available on the main window. The main window itself is the gateway for all of the possible parallel processes that can be called by the user at any time. These four parallel processes are Connect, Disconnect, Change User Info, and Purge. As noted in Figure VI.3, the Purge is only available if the user has logged in as an administrator. The idle state that the main program is on, that is: when none of the buttons are being pressed, is displaying the main window which has the four previously mentioned parallel processes ready to be ran. In addition to these four parallel processes that need to be triggered by the user clicking on their respective buttons, there are other events that can trigger parallel processes not available to the user. These events are safeties like error shutdowns, stopping the application through an emergency button (the close button on the window), or other functions available to every other user, such as a logout button or an administrator-generated report**.**

On one hand, the error and exit application functions that are possible events have a high priority level; the queue manager 1.3 pictured in Figure VI.2 assigns them to front of the main function queue, as well as to the subVI queues which will cause them to shut down promptly. On the other hand, triggering the events that are in the normal scope of the program (such as connecting, disconnecting, user info, logins, etc.), causes the queue manager to put these tasks on the back of the queues on their respective subVIs. For example, clicking on the Connect button would put an Add Connection task to the back of the queue for the subVI Connect. After this is accomplished, the main program goes back to its idle state, while all of the work is performed by the called subVI. The subVIs and their logic are explained in the following paragraphs.

Another important step in the logic of the algorithm is the existence of a connection name, which is saved as a number, and a tag which saves some data about each connection made. Each tag has four components: user, lock, bus, and device. The number that stands for connection name saves each connection made as a number, the first connection gets labeled as "1", the second "2", etc. This is done for logging and report generation purposes. If an administrator wants to look at the history of the connections made, each report will have a report number, as well as connection numbers saved for each session the application is on. The application is designed to be run continuously, and only people with the proper authority can shut down the processes that are ongoing in order to reboot the system or clear the connections in case complications arise. The other data fragment that is carried by every connection is the four tags: user, lock, bus, and device. The user tag logs which user has made a certain connection. Only people with the proper credentials can make connections, and only they can terminate their own connections. Therefore, the system needs to know the identity of the person who made each active connection at every time. An exception to this rule is an administrator, who can terminate any connection at any time. The lock tag checks if the user has locked down a connection. If the connection is locked, no one can latch another device onto the same bus line. If the state is unlocked, then additional connections can be made if the users desire to do so. The bus tag checks which bus is occupied by a connection, a tag mostly used internally in the logic of the connection forming, so that the optimal path is found through the switch matrix. The device tag checks which devices have been connected to the grid, making them unavailable for connections if they have been locked out, similarly only used by the algorithm, to ensure that there are no conflicting cases that can exist. All of these tags are used by the connect and disconnect parallel processes.

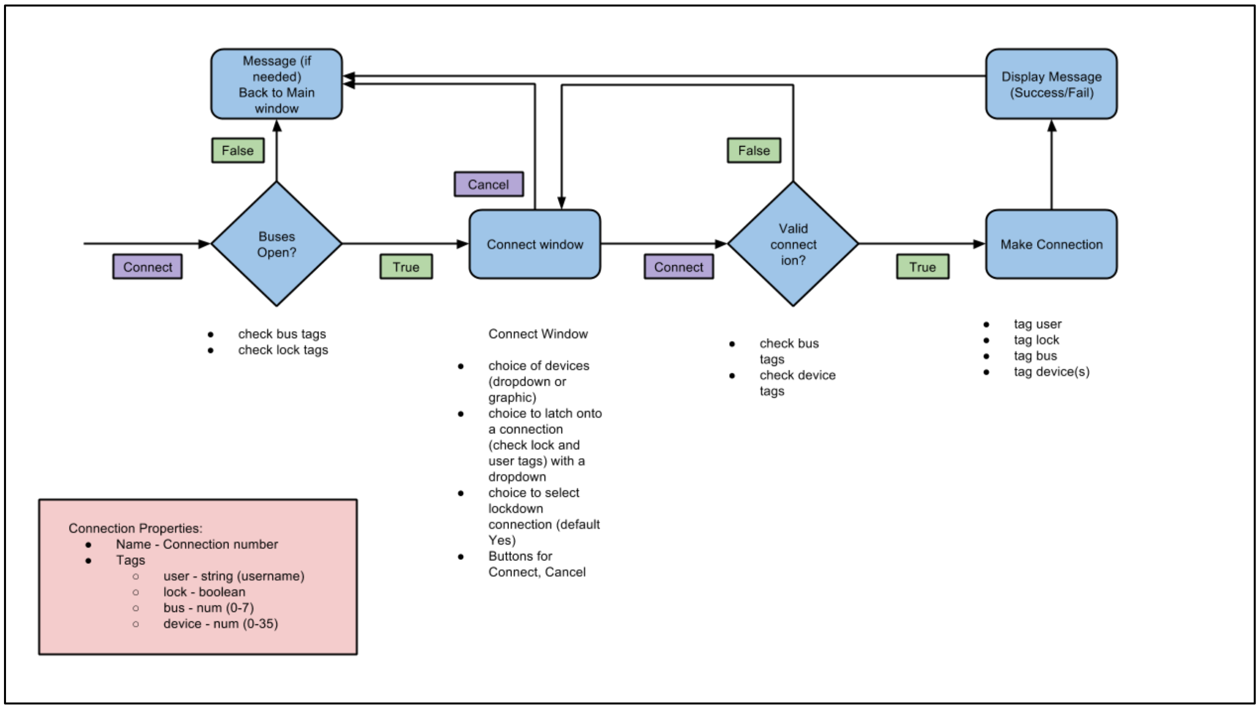

Message (if needed) Back to Main window
Display Message (Success/Fail)
False
False
Cancel
Connect
Buses Open?
True
Connect window
Connect
Valid connect ion?
True
Make Connection
• check bus tags
• check lock tags
Connect Window
• choice of devices (dropdown or graphic)
• choice to latch onto a connection (check lock and user tags) with a dropdown
• choice to select lockdown connection (default Yes)
• Buttons for Connect, Cancel
• check bus tags
• check device tags
• tag user
• tag lock
• tag bus
• tag device(s)
Connection Properties:
• Name - Connection number
• Tags
○ user - string (username)
○ lock - boolean
○ bus - num (0-7)
○ device - num (0-35)

Figure VI.4. Connect flow chart for Connect subVI

The flow chart for the Connect subVI is shown in Figure VI.4. The same guidelines apply- the rectangular bubbles are different states and the diamonds are queries for the system. The same data types as Figure VI.3 apply as well. Whenever the user clicks on the Connect button, the system goes into a query where a hardware check is performed. If there are not any buses available for a connection to be made, then the system automatically notifies the user that there are no possibilities for a new connection to be made. In order to connect a new set of devices together in the matrix, there has to be a bus line available for this new connection to be achieved. In the current hardware connection, up to eight simultaneous connections are allowed. If all eight bus lines are busy at the same time, a message notifies the user that is attempting to make a connection that the switch matrix is unable to do so. After this message the user is brought back to the main window.

In case there are available buses, the system goes to the next step which is the connect window. The connect window has some options for the user to choose. There is a choice of devices to select, a choice to select a lockdown on the channel, and an option to cancel the action and return to the main window. The connect window itself is a dialog pop up window, and can be removed by selecting connection settings and clicking the connect button, or clicking the cancel button at any time. Both of these pathways lead the user back to the main menu. When the user selects the devices and clicks on the connect button, the system checks if those devices are available. Once again, there are two possible outcomes: the devices are available and the hardware makes the connection or, one or both of the devices are not available and the user sees a message and is brought back to the main window.

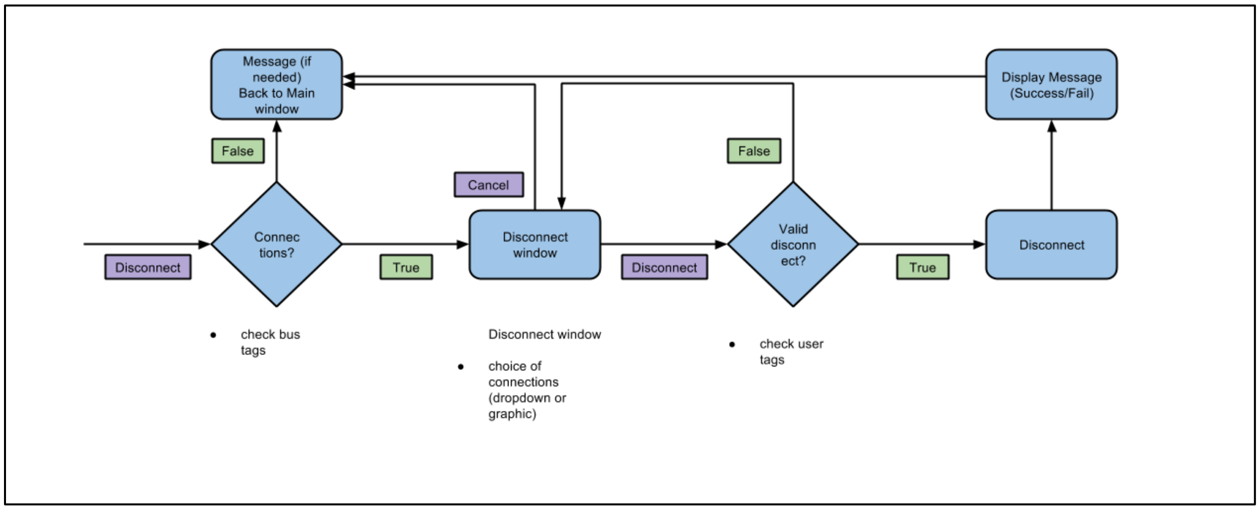
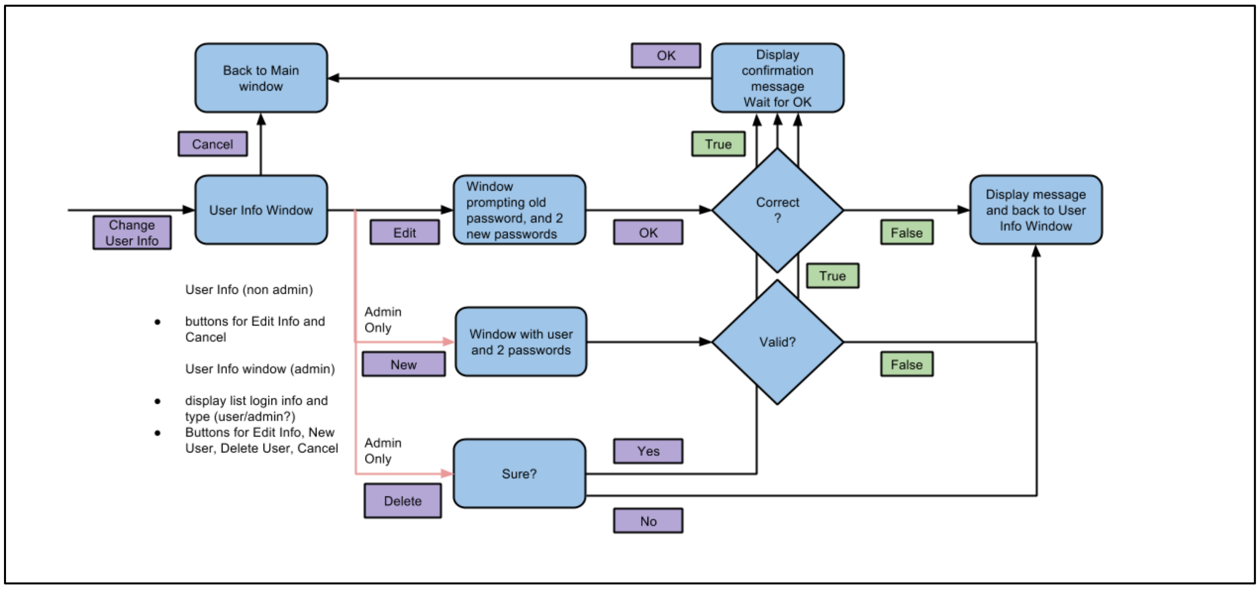


Figure VI.5. Disconnect flow chart

The disconnect parallel process, pictured in Figure VI.5, follows a logic that is similar to the connect logic, but has some differences. When the user clicks on the disconnect button from the main menu, the system checks if there are any connections that have been made by the same user. If there aren't any connections available that the user has the authority to shut down, the system sends a message saying that there are no connections available to disconnect. On the other hand, if there are connections available, the user is taken to a different pop-up window. In this window -unlike the connect pop-up- there is a list of available connections that the user can disconnect. These available connections have buttons that are highlighted, and available to press on. The connections that are unavailable to be disconnected are grayed out, and the user cannot press down on those buttons. From this point there are two options: the user can either select any of the available buttons and the hardware will disconnect the two devices connected on that bus line, or, press cancel in order to return to the main menu without causing any changes to the switch matrix.



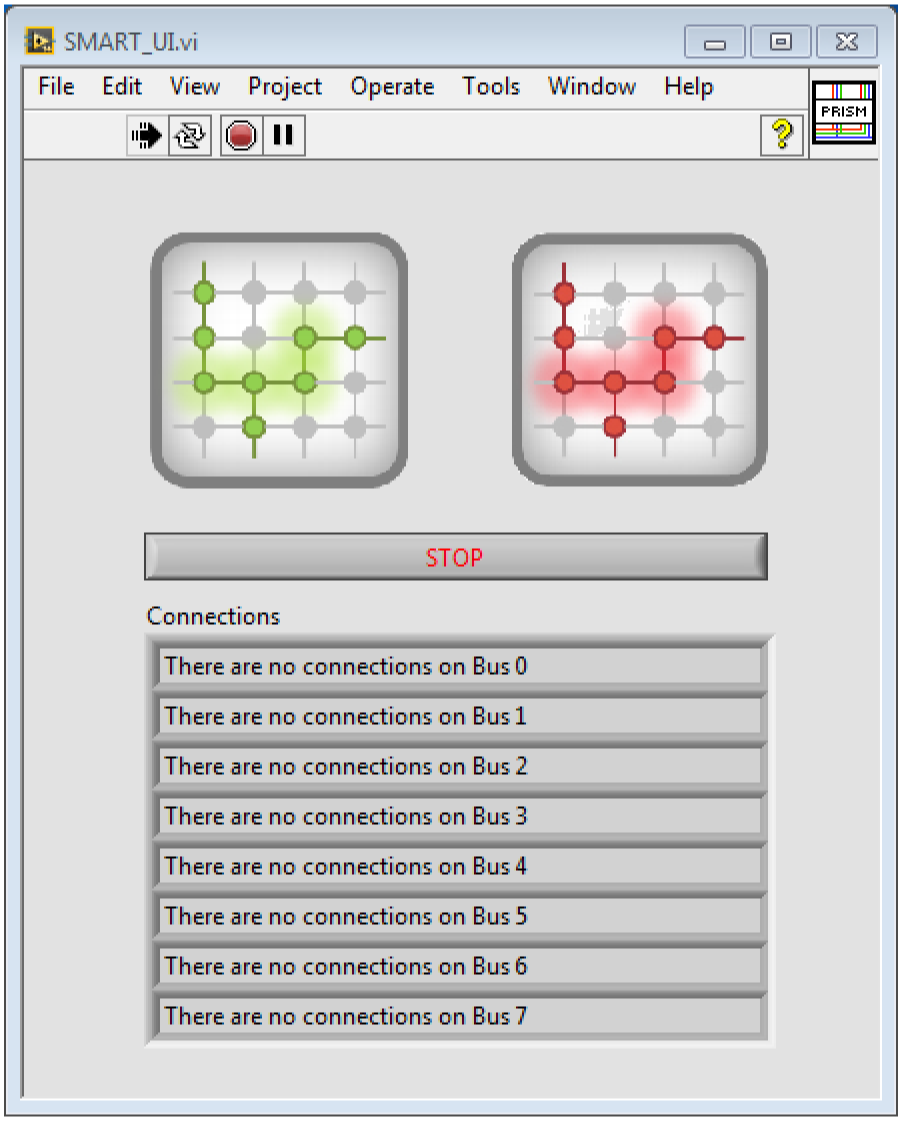
Back to Main window
OK
Display confirmation message Wait for OK
Cancel
True
Change User Info
User Info Window
Edit
Window prompting old password, and 2 new passwords
OK
Correct ?
False
Display message and back to User Info Window
True
User Info (non admin)
buttons for Edit Info and Cancel
User Info window (admin)
display list login info and type (user/admin?)
Buttons for Edit Info, New User, Delete User, Cancel
Admin Only
New
Window with user and 2 passwords
Valid?
False
Admin Only
Sure?
Yes
Delete
No

Figure VI.6. Change user info flow chart

The change user info is using a different subVI from the connect and disconnect parallel processes. In this subVI, the user can change their login info, add new users, as well as delete old users. If the user has an administrator login, all of these options are available for selection. If the user does not have an administrator login, only the option to edit his/her own info is available. This was done as a nicety to the industry end user, to allow for control over who can change the information of the users. This is important because in order to use the program, every new user needs to log in, which requires clearance from the administrator.

VII Results

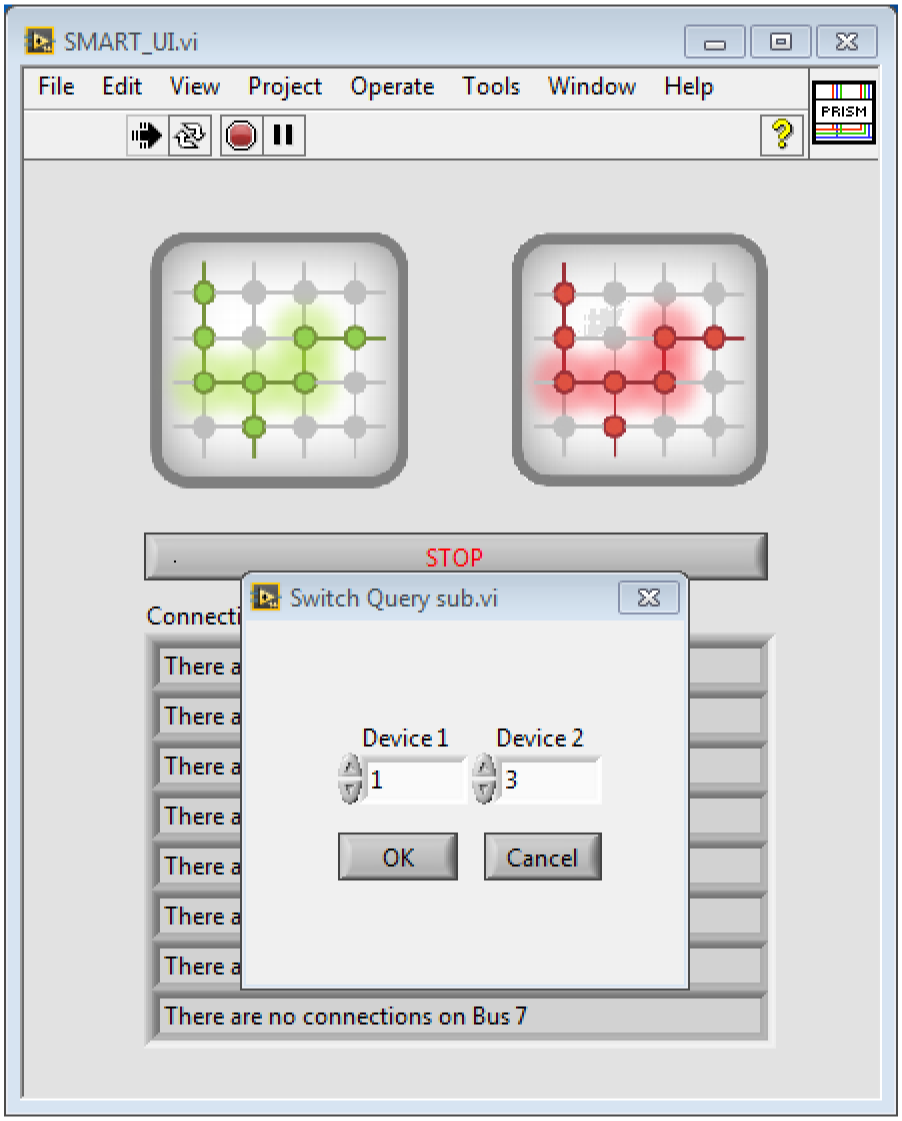
The algorithm mentioned in section VI was programmed in LabVIEW, and the resulting project ran in conjunction with the purchased hardware. The function of the program was tested by presenting a series of scenarios in order to test for bugs and ensure that the logic is being executed properly. In order to make sure that the switch matrix was executing properly, the relays were checked in NI MAX after being tripped by the program. In addition to looking at simulated hardware, NI MAX can ensure that the connected hardware is running properly as well. All of the logic involved in creating new connections and disconnecting the older ones was accomplished. The graphical user interface (GUI) that is presented in this paper is fully functional; all of the buttons accomplish the tasks they are designed to perform.



There are no connections on Bus 0
There are no connections on Bus 1
There are no connections on Bus 2
There are no connections on Bus 3
There are no connections on Bus 4
There are no connections on Bus 5
There are no connections on Bus 6
There are no connections on Bus 7

Figure VII.1 for the algorithm mentioned in section VI

The algorithm mentioned in section VI was programmed in LabVIEW, and the resulting project ran in Figure VII.1 shows the default window that appears when the application is pulled up. There are two main buttons that were designed in order to simplify the interface between the user and the program. The green button is a make a connection button, which will prompt a switch query to occur. The red button is a disconnect button, which will prompt a disconnect switch query to occur. Both of these only show up when the logic allows them to. For example, if a user wants to create a connection when all of the buses are taken, a message will display that no new simultaneous connections can be made. If there are no connections made, and the user clicks on the disconnect button, there will be a message displayed that there are no connections to be altered.



Connecti

There a

There a

There a

There a

There a

There a

There a

There are no connections on Bus 7

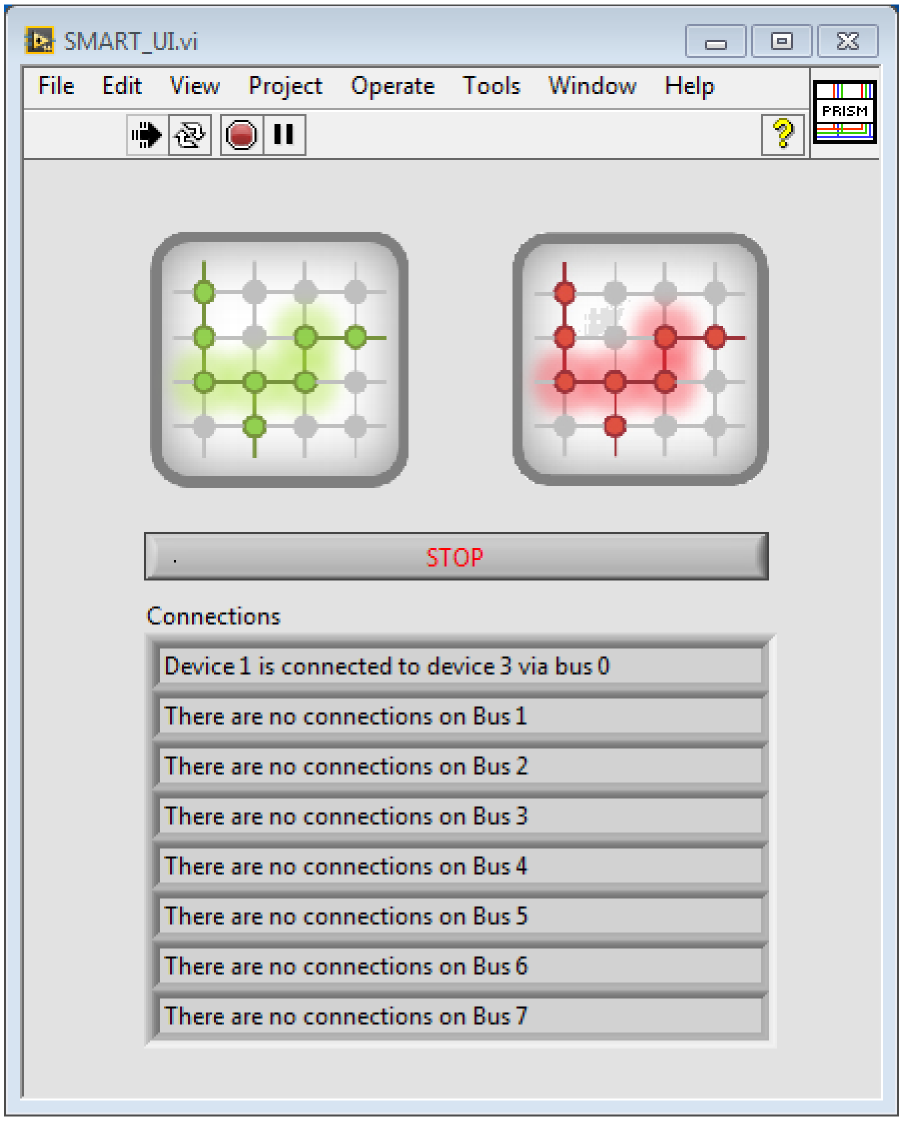
Device 1 1

Device 2 3

Figure VII.2

After pressing the green make a connection button, there is the switch query that appears as a pop-up window. While the pop-up switch query window is up, the user is unable to go back and click any other buttons on the main window. The switch query allows for a few selections to be made. If the user wants to go back to the main window without making a change, he can click on the cancel button at any time and the switch matrix will remain unaltered. The user has to name valid devices in order for the connection to be made. The switch query performs a variety of checks to make sure that the connection can be made:

- Validity of device name
- Two different devices need to be selected
- Ensure that the device selected is not in use already



Connections

Device 1 is connected to device 3 via bus 0
There are no connections on Bus 1
There are no connections on Bus 2
There are no connections on Bus 3
There are no connections on Bus 4
There are no connections on Bus 5
There are no connections on Bus 6
There are no connections on Bus 7

Figure VII.3- Depicting selection of device 1 and device 3 from the switch query presented in Figure VII.2,

Following selection of device 1 and device 3 from the switch query presented in Figure VII.2, the system will then make the connection via the easiest route it can find. As seen in Figure VII.3, the connection from device 1 to device 3 was made via bus 0, which was the first bus that was not in use. The main window has a large section where the user can see what is being occupied by each of the bus lines. As seen in Figure VII.3, the only connection made was the one that was presented in this paragraph, and all of the other bus lines are still free.

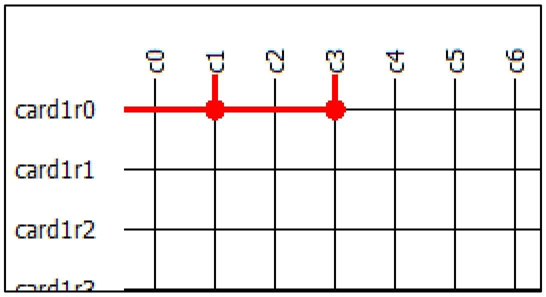
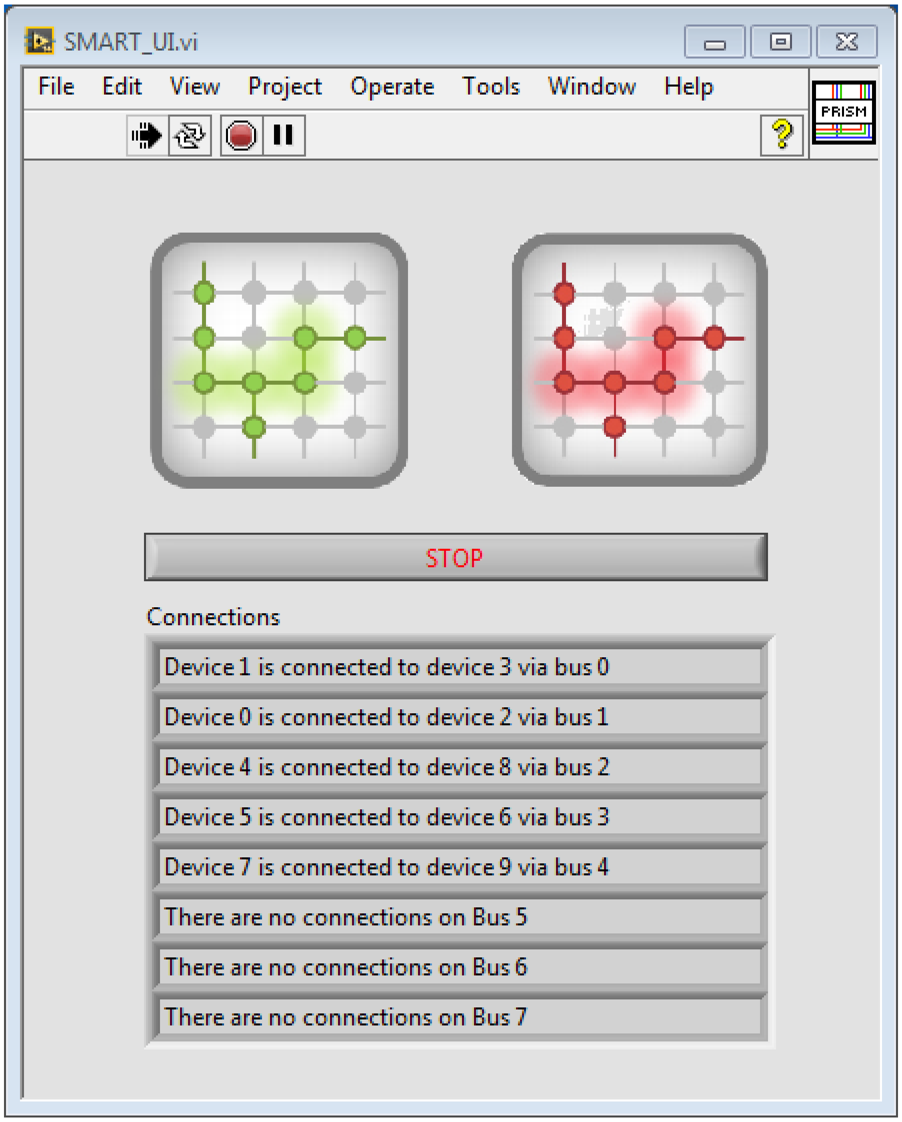


Figure VII.4- connection made on the physical switch matrix

As seen in Figure VII.4, the connection was made on the physical switch matrix as well. The representation is as seen in NI MAX when viewing the NI 2834 card that is installed in the PXI SwitchBlock.



Device 1 is connected to device 3 via bus 0
Device 0 is connected to device 2 via bus 1
Device 4 is connected to device 8 via bus 2
Device 5 is connected to device 6 via bus 3
Device 7 is connected to device 9 via bus 4
There are no connections on Bus 5
There are no connections on Bus 6
There are no connections on Bus 7

Figure VII.5- the result of making more connections on the switch matrix

Figure VII.5 shows the result of making more connections on the switch matrix. The program finds the first available bus for each one, and after checking that there are no extraneous connections made, proceeds to make the physical connection. As seen from Figure VII.5, there are five different connections that have been made. Each one of the devices connected is different, since the system doesn't allow for the same device to be connected across bus lines.

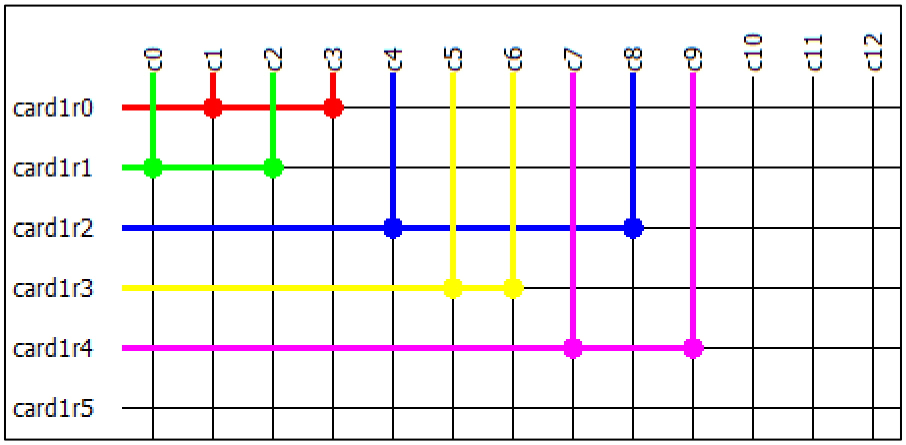


Figure VII.6- representation on these five simultaneous connections is exactly as seen in the main window in Figure VII.5.

Looking at the connections that were made in the main window in Figure VII.5, we can recognize that there are five simultaneous connections that are active. These are supposed to occupy the first five available buses: bus 0 through 4. As seen in Figure VII.6, the NI MAX representation on these five simultaneous connections is exactly as seen in the main window in Figure VII.5. There are five pairs of devices, each connected on a separate bus line from each other, all in the correct spots. Figure VII.6 shows that the algorithm can correctly sort through the information requested by the user, and assign the proper switch matrix relays to be closed.

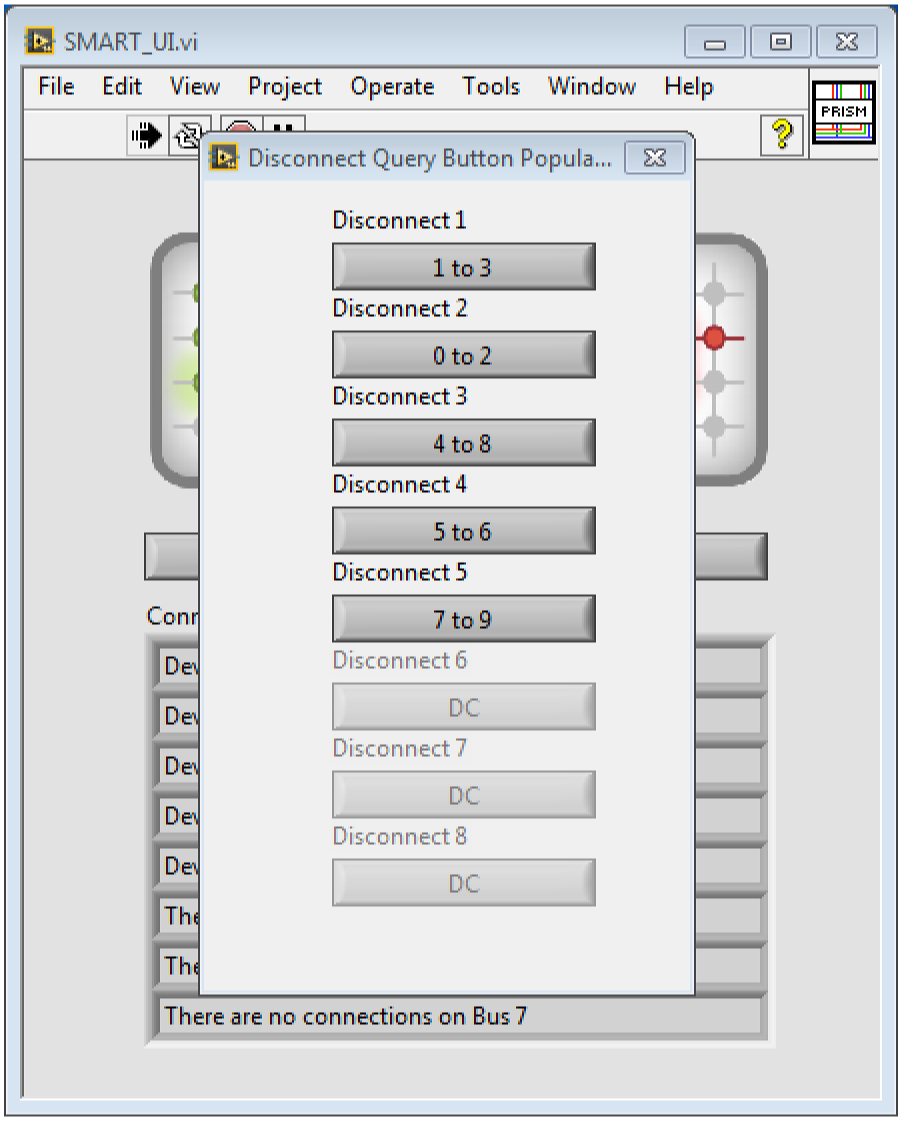
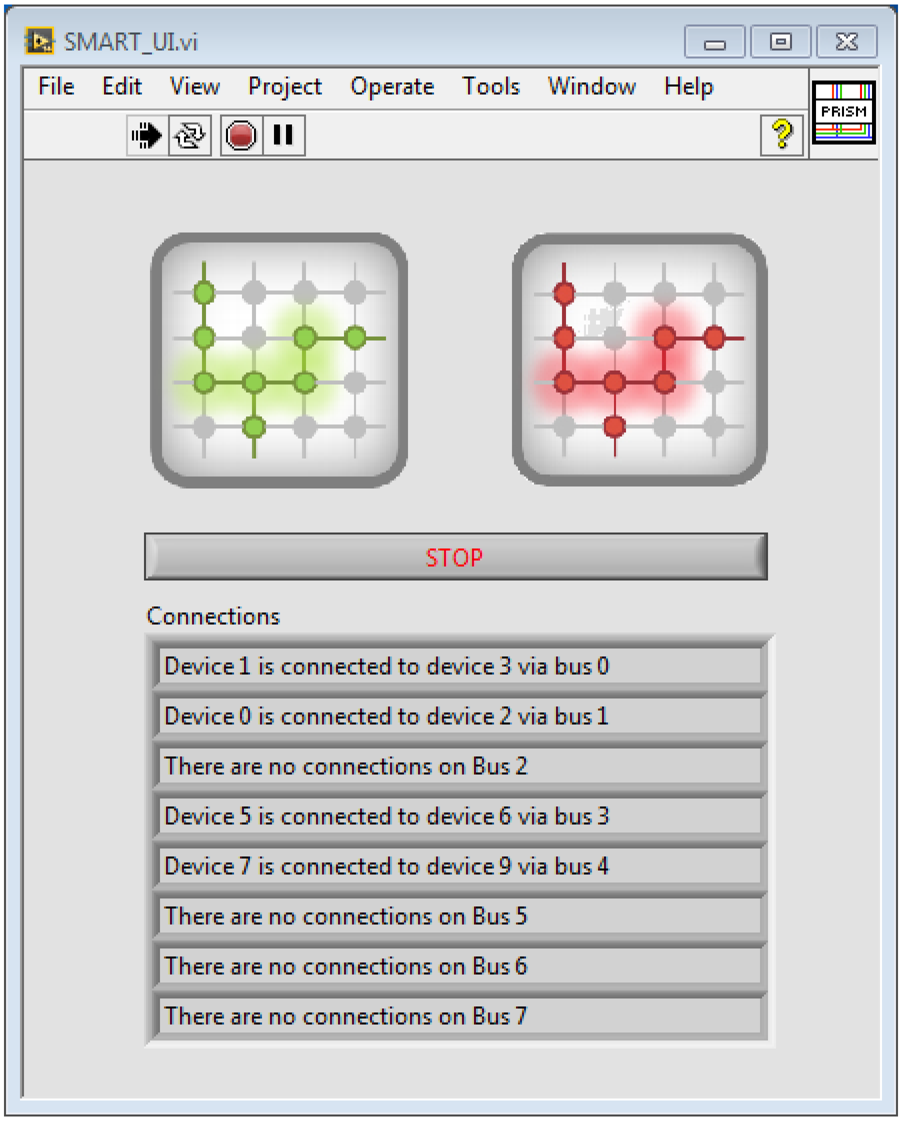


Figure VII.7- Clicking on the red disconnect button shows the pop-up disconnect query window

Clicking on the red disconnect button shows the pop-up disconnect query window, as shown in Figure VII.7. Similar to the connect pop-up, this window doesn't allow for the user to click out until he has selected an option available to him. The algorithm looks at the connections that have been made and populates the window for bus lines that can be disconnected. The buttons themselves are programmatically generated, and display which devices have been connected on each bus. The buttons that are not greyed out are available for the user to click on, and they will promptly close down that connection.

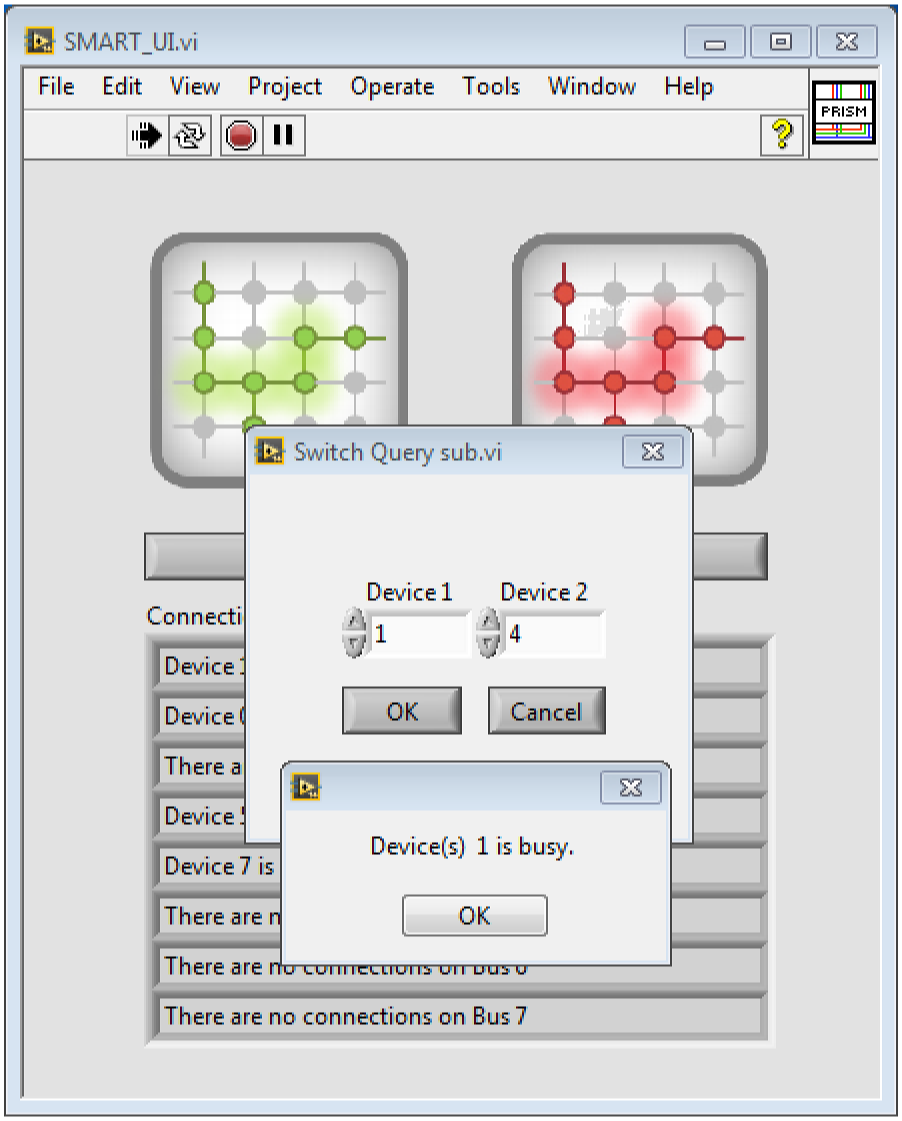


Connections

Device 1 is connected to device 3 via bus 0
Device 0 is connected to device 2 via bus 1
There are no connections on Bus 2
Device 5 is connected to device 6 via bus 3
Device 7 is connected to device 9 via bus 4
There are no connections on Bus 5
There are no connections on Bus 6
There are no connections on Bus 7

Figure VII.8- The main window after disconnecting device 4 to device 8 that were connected on bus 2.

Figure VII.8 shows the main window after disconnecting device 4 to device 8 that were connected on bus 2. The program recognized which bus needed to be disconnected, and left the other connections in place in case there were other tests running on those connections.



Device(s) 1 is busy.

OK
Connecti
Device 1
Device 0
There a
Device 5
Device 7 is
There are n
There are no connections on Bus 6
There are no connections on Bus 7

Figure VII.9- Dialog window notifies the user that device 1 is busy, already being connected to device 3 on bus 0.

There are multiple routes that show how the algorithm recognizes that there are mistakes in the logic of the user, and sends quick dialog windows in order to ensure the user knows why the connection cannot be established. An example is shown in Figure VII.9, in which a dialog notifies the user that device 1 is busy, already being connected to device 3 on bus 0.

VIII Conclusions

The university-industry collaboration that drove this project to its completion has proven to be successful in merging these two environments, as seen in other literature [1],[2],[4]. The hardware/software pairing is fully functional and ready to be tested and used by the industry representatives. The success of this project was achieved via thorough and informative communication by the university counterpart, seeking the expertise from the industry representatives.

In order to achieve the previously mentioned streamlined project development between the two entities [5], there is communication underway in order to develop more projects that can use the developed PXI platform. One such project is the use of a PXI controlled virtual transmission impairment measurement set in order to perform line integrity testing. The project is being developed using compatible programming techniques, with VIs that can be incorporated with PrISM VIs.

Further developments on this project will conclude with the completion of the graduate student thesis that the project is centered around. These developments will include, but are not limited to:

- Noise and signal loss testing over multiple relays of the PXI switch matrix
- Testing using real time live communication devices at the industry end user facilities
- Continued development of a basic login system
- Report generation on user request

Acknowledgement

This work was accomplished through the authors' collaboration with Mr. Clint Quisenberry, Mr. Barry Foister, Mr. Jack Rouse and Mr. Joe Hoffert in the Oklahoma Communications Engineering Team at Mike Monroney Aeronautical Center-Federal Aviation and Administration (MMAC FAA).
This work was supported in part by Chickasaw Nation Industries (CNI), a contractor for National Airway Systems Engineering (NASE) organization and Mr. Neil Peery.

References

1. de Jongh, P. J., & Erasmus, C. M. (2014). Industry-directed training and research programmes: The BMI experience. South African Journal Of Science, 110(11/12), 17-24.

2. Waters, C. E., Alvine, S., & Eble-Hankins, M. (2012). Industry-Experienced Graduate Student Program: Innovative Collaboration in Architectural Engineering at the University of Nebraska, Lincoln. Journal Of Architectural Engineering,18(1), 61-63.

3. Behrens, T., Gray, D. (2001). Unintended Consequences of Cooperative Research: Impact of Industry Sponsorship on Climate for Academic Freedom and Other Graduate Student Outcome. Research Policy 30.2. 179-99.

4. Lucia, Ó., Burdio, J. M., Acero, J., Barragán, L. A., & Garcia, J. R. (2012). Educational opportunities based on the university-industry synergies in an open innovation framework. European Journal Of Engineering Education, 37(1), 15-28.

5. Nielsen, C., & Cappelen, K. (2014). Exploring the Mechanisms of Knowledge Transfer in University-Industry Collaborations: A Study of Companies, Students and Researchers. Higher Education Quarterly, 68(4), 375-393.

6. Wolcott, M., Brown, S., King, M., Ascher-Barnstone, D., Beyreuther, T., and Olsen, K. (2011) Model for Faculty, Student, and Practitioner Development in Sustainability Engineering through an Integrated Design Experience. Journal of Professional Issues in Engineering Education and Practice 137.2. 94.

7. National Instruments (2009). *Creating a Large Switch Matrix* [White paper] Retrieved Dec 21, 2014 from National Instruments: http://www.ni.com/white-paper/3176/en/

8. National Instruments (2006). *Matrix Switch Expansion Guide* [White paper] Retrieved Dec 21, 2014 from National Instruments: http://www.ni.com/white-paper/3628/en/

9. Dimitrakopoulos, G. (2010). Designing Network On-Chip Architectures in the Nanoscale Era. Chapman and Hall. 67-88.

10. Chan, K. Y. (2008). Monolithic crossbar MEMS switch matrix. IEEE MTT-S International

11. Hoare, R. D. (2006). A Near-optimal Real-time Hardware Scheduler for Large Cardinality Crossbar Switches. SC Conference, (p. 8).Microwave Symposium Digest , 129-132.

12. Campbell, B. (1984) Automatic Systems Facilitate Efficient Reed Relay Characterization. Source: *Electronics industry*, v 10, n 10, p 37, 39, Oct 1984.

13. Xu, L. Zhang, J., Miedzinski, B. New design of multi-contact reed relay for improving switching load capacity. Electrical Contacts, Proceedings of the Annual Holm Conference on Electrical Contacts, p 214-219, 1998.

14. Malone, E., Lipson, H. (2007) Freeform fabrication of a complete electromechanical relay. 18th Solid Freeform Fabrication Symposium, SFF 2007, p 513-526, 2007, 18th Solid Freeform Fabrication Symposium, SFF.

15. Malone, E., Lipson, H. Freeform fabrication of a complete electromechanical relay. *18th Solid Freeform Fabrication Symposium, SFF 2007*, p 513-526, 2007, *18th Solid Freeform Fabrication Symposium, SFF 2007*

16. National Instruments (2013). *Choosing the Right PXI System Architecture* [White paper] Retrieved Dec 22, 2014 from National Instruments: http://www.ni.com/white-paper/2722/en/

17. National Instruments (2013). *NI-DAQmx Simulated Devices* [White paper] Retrieved Dec 22, 2014 from National Instruments: http://www.ni.com/white-paper/3698/en/

18. National Instruments (2010). *Using Test Panels in Measurement & Automation Explorer for Devices Supported by NI-DAQmx* [White paper] Retrieved Dec 22, 2014 from National Instruments: http://www.ni.com/white-paper/4638/en/

19. National Instruments (2011). *NI-DAQmx Device Pinouts* [White paper] Retrieved Dec 22, 2014 from National Instruments: http://www.ni.com/white-paper/4053/en/

20. National Instruments (2013). *Virtual Devices* [White paper] Retrieved Dec 27, 2014 from National Instruments: http://www.ni.com/white-paper/4752/en/

21. National Instruments (2008). *Tutorial: Sub-VIs* [White paper] Retrieved Dec 27, 2014 from National Instruments: http://www.ni.com/white-paper/7593/en/

22. Warriach, E. U., Tei, K. (2013) Fault Detection in Wireless Sensor Networks: A Machine Learning Approach. 2013 IEEE 16th International Conference on Computational Science and Engineering, pp. 758-765.

23. Hao, L., Stitt, G. (2013) Virtual finite-state-machine architectures for fast compilation and portability. 2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors, pp. 91-94.

24. Anthony L. (2013, Nov 18) LabVIEW Queued State Machine Architecture. Retrieved from: https://decibel.ni.com/content/docs/DOC-32964