Test Automation Program for 20-126

by Stacia Kwong

California Polytechnic State University
San Luis Obispo, CA

Senior Project 2015

Table of Contents

List of Figures and Tables

Abstract
Test Automation Scripting for Room 20-126 is a project to incorporate automation scripts with existing Electrical Engineering laboratory classes. Incorporation with laboratory classes will require research and defining the specific purposes. The general purpose of this project enables students to control the instruments in Room 20-126 and to obtain the data collected. The process to gain control of test instruments must be easy to use and implement because students may not have the background or knowledge for controlling instruments. Steps to solve this problem would benefit in a simple user interface for general control options and data collection.

Introduction

The purpose of this project is to create a new automation scripting suite or program for Level 200+ Cal Poly students to use in their Electrical Engineering labs. Some potential classes to use these scripting suites include: EE 241, EE 242, EE 251, EE 361, EE 346, EE 347, EE 348, and EE 449. The automation scripting suites will allow students to learn how to control instruments to measure variables needed for data collection. The automation scripting language will be discussed later on, but will be chosen based on the language that will be applicable in the “real world” as well as useful for student’s purpose for use of the program — data collection.

Background and Market Research

Cal Poly's Electrical Engineering (EE) department has numerous labs with varying equipment. Room 20-126 is unique from any other labs in the building due to the new equipment in the room. Since the equipment is new, an automation or tool does not exist to acquire measurement variables. In the older lab rooms, there are measurement tools, but these measurement tools are not compatible with the new equipment in Room 20-126. This project's objective is to solve this lack of measurement tool as well as to allow students to apply their programming knowledge.

The new equipment in Room 20-126 includes the following: Multimeter/DC Power Supply (U3606A), Triple Output DC Power Supply (E3631A and E3630A), 6 1/2 Digit Multimeter (34461A), and Mixed Signal Scope (MSO-X-2012A). All of the equipment is from Agilent, now called Keysight Technologies.

In many of the EE lab classes, students are required to acquire a large amount of data. Manually doing so would take unnecessary time because the processes is tedious and repetitive. The purpose of these labs is not to force students to do tedious things, but to learn the cause and effects of their circuits. Currently in the old labs, students are able to collect this data though a computer software to acquire graphs that vary voltage or current. The current software is not beneficial to students in the future because the software is not a common software.

There are two options for the new equipment to either create a software tool or create an automation script. If students have access to either of these tools, time to data would decrease and students would be able to learn how to control instruments from the computer.

As shown in Figure 1, Cal Poly EE students are only required to take four classes that focus and teach about programming languages. This equates to 16/194 units or 8.25% of courses. From my own experience, I have found it difficult to get into other programming classes such as CPE 102 (the following class after CPE 101) because the class is saved for certain majors such as Computer Engineering or Computer Science. The inability to take classes due to not enough space for non-majors keeps the amount of programming exposure to Electrical Engineers low.

Since Cal Poly's EE major lacks programming and automation classes, automation scripts to test circuits would be beneficial in building the student's knowledge and understanding of programming. It would also prepare students in their future careers. When Electrical Engineers begin working, most will begin at lower level jobs that may require testing of circuits or systems. If they have knowledge of automation, their jobs can be more efficient than if they were to manual operate everything. Efficiency is not only important in saving time for the engineer, but also for the company because the company is able to sell and distribute their products faster. Exposure to automation scripts early on will prepare students to be able to understand the complexity and power of programming.

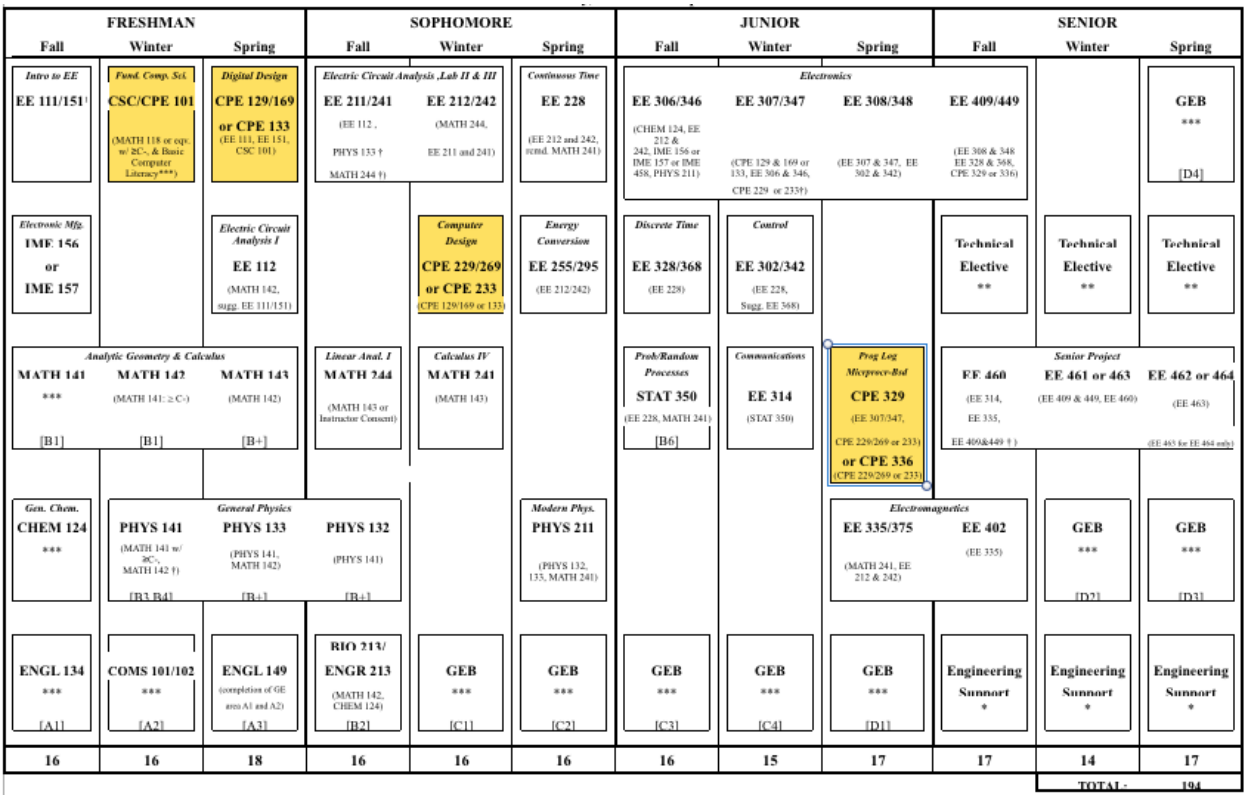

Figure 1: Cal Poly Electrical Engineering Flowchart
Classes Highlighted Expose Students to Programming [4]

The language used for automation is also very important because some programming are easier to read or easier to use or students may already have a simple understanding of certain programming languages. Using a programming language that is used in the workplace would be wise because students would be a more desirable candidate in their future careers. Programming languages that fit these criteria include Python and LabVIEW.

Creating automation in Python has many benefits, especially since the Electrical/ Computer Engineering department is changing their CPE 101 curriculum from programming in C to Python. There are more benefits to using Python other than Cal Poly's change in class curriculum, but it is a huge benefit because students will have already learned the knowledge needed before automation in Python. Also, students will be able to apply classroom knowledge to testing their physical circuits on the bench. Python is becoming a very popular programming language. According to IEEE Spectrum's most popular programming languages of 2014 in Figure 2, Python is the fifth most popular programming language. Although choosing the most popular programming may seem like the obvious choice, the first four programming languages are not as efficient as Python.

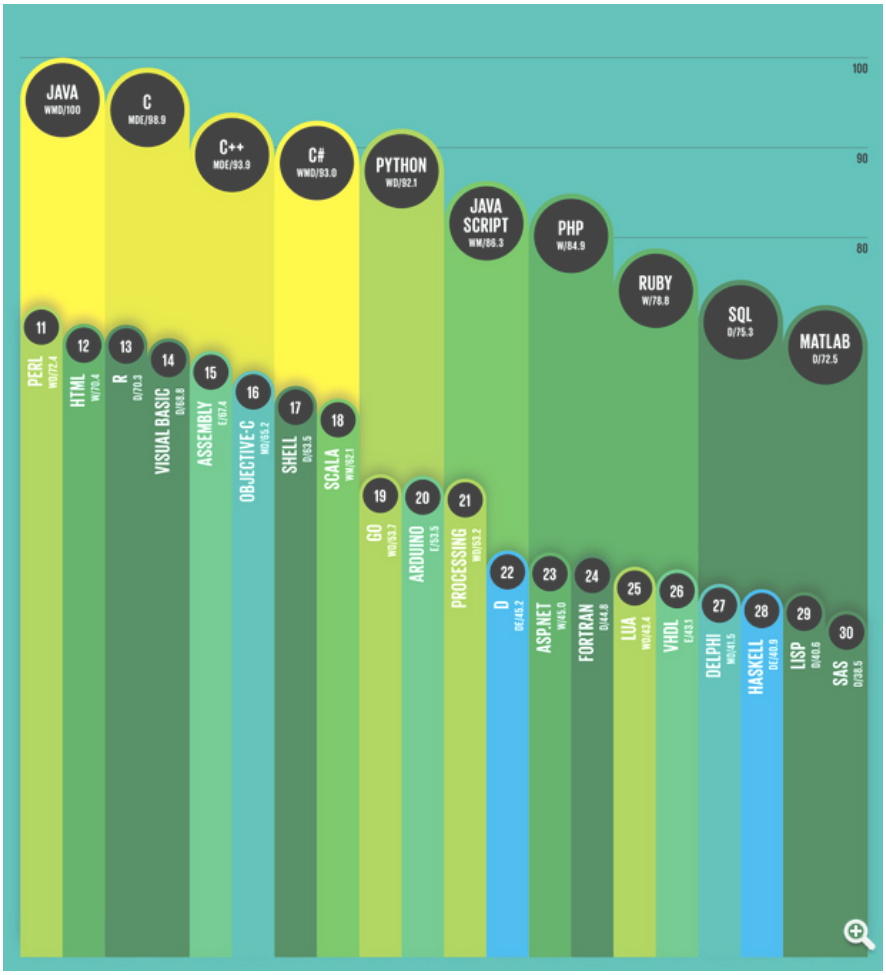

Figure 2: Top Programming Languages 2014 [1]

Java	`public class ConsoleTest {`
	`  public static void main(String[] args) {`
	`  for (int i = 0; i < 1000000; i++) {`
	`  System.out.println(i);`
	`  }`
	`  }`
	`}`

Python	`for x in xrange(1000000):` `  print x`

Figure 3: Python and Java Script Comparison [6]

Since C++ and C# are based off of C, the three are written very similarly. Figure 3 shows the difference between Java and Python. Both scripts perform the same action, but Java clearly requires more lines of code. This shows the simplicity of Python as well as the syntax. Python does not require as many words to do the same actions in C or Java. It also does not require as much syntax, such as brackets or parentheses. Since there is less syntax, Python is easier to write because there is less for users to remember to include in their code.

Python is also easier to use because the code reads like English. It is not as abstract as abstract as C or Java, and it has fewer function definition words. For example, when programming in C, the user must state what type of function output it will be receiving, whereas in Python, simply stating that a function is going to be declared is enough. Functions do not need to be told when it ends in Python.

When referring to Figure 3, Java requires many definitions prior to the conditional statement. Someone who does not know Java will not know what public, static, void, main, etc. means or why these words are in the code. However, with Python, it is easy to see that within the xrange, the program will print the x value. Instead of seven lines in Java, Python does the same function in two lines. Python does not require the user to be a well versed programmer to be able to use and understand its code. This will allow students to understand the usefulness of automation and to become appreciative of programming.

Since Python doesn't require as much syntax or words and reads like English, Python can be written faster than writing a code in C or Java. This efficiency will be beneficial when users read the script to understand what is happening as well as when scripts are written, time spent will decrease.

The downfall for Python is the spacing. The reason why there is not much syntax is due to the need of proper spacing. In Python, if the spacing is off by one space, the script will not be able to run and will return with error. For example, the spacing in Notepad++ is different than in the Python shell. A user may always use the tab button to indent and the spacing will appear to be the same in Notepad++, but when the script is opened in the Python shell, it is obvious that the spacing is different. The lines will not be aligned, causing the error.

The solution to this downfall is to consistently use the same program when editing the code. If the user is consistent in spacing and program editor, the spacing should not increase time to program. Even if there is a spacing error, spacing can be easier to correct than scripting error. Although this downfall can be frustrating when compiling, the benefits outweigh this downfall because spacing can be easily fixed by editing in the same program and performing indents in the same way at all times.
Figure 4 is a compilation of ten engineering jobs that require scripting language experience and proficiency. The jobs were chosen from big companies such as Intel, Cisco, Google, and Apple. It shows the desire for candidates with Python experience as

much as other languages. In a job opening for Cisco, the job stated “Strong Python programming skills a must, JAVA/C/C++ programming skills big plus” [7]. The desire for Python experienced candidates shows the trend toward this language.

Number of Jobs with Scripting Language Qualifications

LANGUAGE	JOB QUALIFICATIONS
C	8
C++	6
C#	1
Java	6
Python	6

Google x1
Intel x2
Cisco x2
Apple x2
Lattice Semiconductor x1
Amazon Lab126 x2

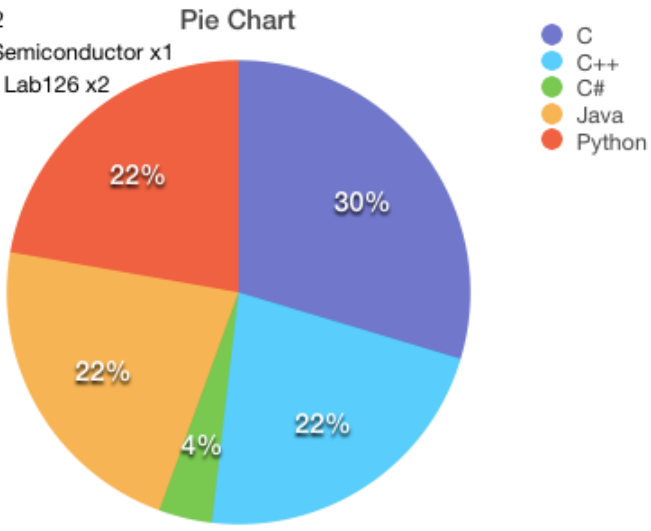

Figure 4: Scripting Language Qualification Requirements

When controlling instruments using automation scripts, users would need to have a large enough understanding to know each of the commands the instrument needs to perform. If a driver is created, users can quickly call the function and add the function to the script. Drivers also help users quickly and efficiently compile an automation script, but still have a basic understanding of what the script does.

The commands in the drivers will allow users to know the usage of the command without having to look up code. For example, if there are commands called get_current, set_current_limit, and set_current, the user will be able to determine that get_current will import the current value of the test equipment, set_current_limit will set the

equipment's limit, and set_current will set the variable current and can later change its value if necessary. The commands will use each equipment's specified SCPI code commands to control the equipment. SCPI code commands allow for the test equipment to be programmable. Without these codes, there would be no command code to program the equipment. The SCPI codes can be found for each equipment from Keysight Technologies' company website.

Drivers can be found online; however, most of the scripts are not for the measurement instruments found in Room 20-126. When looking at Keysight Technologies' (formerly Agilent) website, there are no drivers written in Python [3]. Since there are no available drivers on Keysight's website, the competition for creating drivers for the new instruments is minimal with little to no competition.

Python has many benefits including ease of reading and writing since it looks like English compared to other programming languages such as Java, it is similar to C, it is the language taught in CPE/CS 101 at Cal Poly, it is prevalent in the workplace, and it is listed as a required or desired language in job listings at major companies such as Cisco, Google, and Apple. For all of these reasons, Python was first chosen as the programming language of choice.

The downfalls for Python is the spacing, lack of the Python Shell in Cal Poly's computers, as well as the lack of an easy to implement GUI. Since Python is easy to use, the spacing makes up for the syntax. The lack of the Python Shell in the Cal Poly computers would be tedious to download for each computer in Room 20-126, but is not a make-or-break downfall. The final downfall, the lack of an easy to implement GUI, is the downfall that caused the change from Python to LabVIEW. Since this program will be used for students to easily input data, press a button, then see the output data, a GUI is extremely important. Since the need for an easy, understandable software to control the instruments for classroom applications held more value and importance, Python could not fulfill the requirements for the project.

Where Python fails, LabVIEW prevails because it is already implemented on the computers in Room 20-126 and it has an easy to use GUI. When a user opens a LabVIEW program, they will first see the "Front Panel," which is the GUI for the user to input necessary data as well as see the outputs for that program. The user will not need to see or understand how the program works. This is beneficial when students need to automate the test equipment, but have not had a lot of experience with automation.

Although users may not have had previous education about LabVIEW, they can look at the wiring diagram to understand how the program works. The wiring diagram is fairly easy to understand because each block in the program is shown as an image and there are built in tools that describe the purpose of each block called Context Help. Context Help shows the inputs types and output types of the block that the computer mouse is hovering over. This is helpful when the user wants to know what type of data is allowed as well as the input or output label, so that they know what data needs to be connected at that point. When creating a program, the programmer may and should also add

comments that describes the purpose of each block in the program. This will allow users to understand the purpose for the various inputs and outputs shown on user interface that they interact with.

When looking on Keysight Technology's website, we can find drivers for each of the test instruments in Room 20-126 from LabVIEW, but not for Python. The drivers can be used for LabVIEW to talk to each of the test instruments. Since these drivers are already downloaded onto the computers in Room 20-126, no additional download is necessary.

Companies continue to use LabVIEW for automation purposes in industry. However, LabVIEW educated students may not have as a great of a need compared to Python educated students. When looking at job descriptions, Python education is commonly requested or recommended, but LabVIEW rarely appears on such lists. LabVIEW automation may not in as large of a demand, but the background and visualization of the process of the program will help the programmer to understand the process of VISA communication between a computer and test instruments.

Although LabVIEW has many of the important aspects for this project, it also has some downfalls. When creating the program in LabVIEW, there is an abundance of blocks that have various uses and purposes. If the creator of this program is not familiar with the blocks, then Context Help is helpful to an extent, but can still be an unknown block because we cannot see the specifics.

All of the equipment and computers are already setup and in use in Room 20-126, so no additional financial investments need to be made, only time to create drivers and automation scripts. However, if the equipment and computers were not already in place, the financial costs would be large. Table 1 shows the breakdown of prices for each bench. Room 20-126 has 12 benches, so the total cost for equipment calculates to $9,200 x 12 = $110,400.

A yearly purchase for the LabVIEW software program continues to be purchased for the EE department, so this would not change the EE department's spending. However, if the EE department had not already purchased LabVIEW, the yearly cost depends on the LabVIEW version. For the LabVIEW Base, yearly costs total to $1,000. Upgrades to the Full version costs $3,000 per year. Finally, the LabVIEW Professional would cost $4,000 per year.

If starting this experiment started without testing equipment and software already in place, this project would be very costly due to the prices for each of the Agilent measurement instruments and having 12 benches worth of equipment. 12 benches are necessary to hold a class size of 24 students, so each of these benches is necessary for Cal Poly's EE students to learn and acquire this automation knowledge. Depending on the EE department budget, LabVIEW could exceed budgets for the department if priorities were placed elsewhere.

Table 1: Equipment Costs per Bench

Equipment	Quantity	Approximate Cost
U3606A Multimeter/DC Power Supply	x1	$1,000
E3631A Triple Output DC Power Supply	x1	$1,000
E3630A Triple Output DC Power Supply	x1	$1,000
34461A 6 1/2 Digit Multimeter	x2	$2,000
MSO-X-2012A Mixed Signal Scope	x1	$3,000
Dell Computer, Monitor, Keyboard, & Mouse	x1	$1,200
		TOTAL COST PER BENCH $9,200

The cost to continue to operate with the test equipment is also added per year due to electricity expenses, maintenance such as calibration, and repair when needed. Agilent's website also shows an example of the breakdown of costs as shown in Figure 5. Although the initial cost is large, the cost over time will decrease [5].
To control the instruments through the computer, all instruments will need to be connected to the computer. This will allow the computer the control the measurements taken or limits set to each instrument through the automation script. Each of the instruments can be connected through GPIB

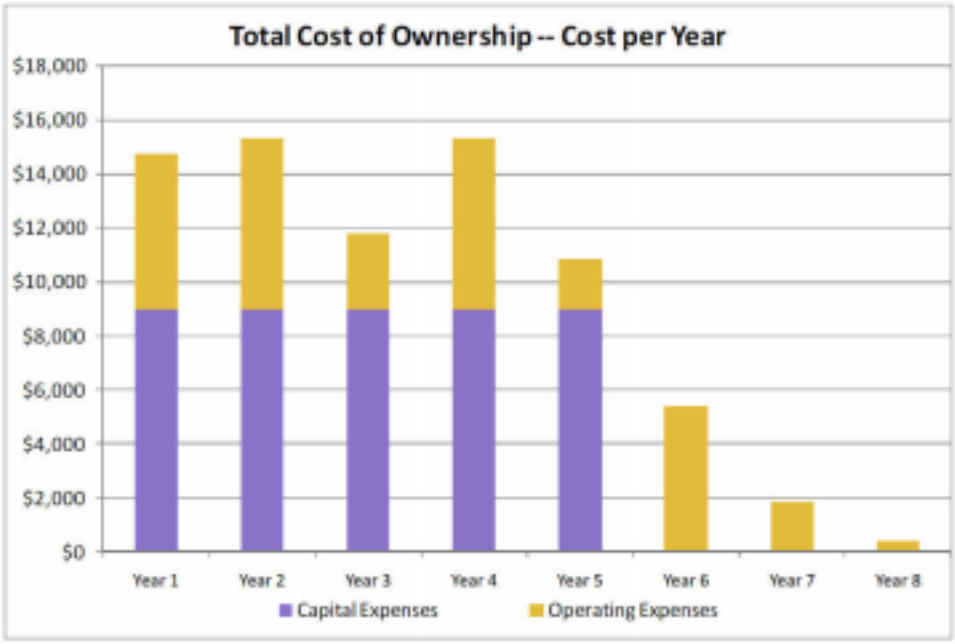

Figure 5: Total Cost of Ownership Over Eight Years [5]

Requirements and Specifications
Currently, there are no solutions for students to apply prior knowledge about programming as well as obtain data for their current lab courses. This project will come up with a solution to solve the lack of automation or test tool for students to use in Room 20-126. The following instruments will be available to control and acquire data: Multimeter/DC Power Supply (U3606A), Triple Output DC Power Supply (E3631A and E3630A), 6 1/2 Digit Multimeter (34461A), and Mixed Signal Scope (MSO-X-2012A).

The test equipment will be able to be controlled by the computer using automation scripts. All automation will be written in LabVIEW due to its ease of use, beneficial user interface, and presence in computers. Drivers will acquired or created to access needed VISA communications, such as initializing the instrument, setting and/or reading values to/from the instrument, and closing the VISA communication line. Multiple drivers will be created, so that each instrument has its own driver. Once the driver is created, the automation program can reference the driver and call the functions described in the driver. Students can view these references to the drivers if they choose to look at the wiring diagram. The wiring diagram will show all connections and which blocks are in use. If students want to observe the process of the automation program, they can click on the light bulb, then run the program. This will enable a delay throughout the program, so the user can see which block is accessed first as well as the inputs and outputs of each block.

Each instrument will be connected through GPIB or USB, so the automation scripts can control each of the instruments. The LabVIEW program will differentiate the various instruments through the instruments' addresses. Each instrument has a unique address which usually appears when the instrument first turns on and/or through its settings. Each instrument will have a different way to find the address, so manuals will need to be referenced or directions will be displayed on the user interface.

The purpose for the automation script is to be able to run one script that will set all testing equipment to necessary limits and controls, will store the results, and will compile the results into a graph. The user should not need to manually do anything from the time the script begins its run until the script is finished. The script will be self-sufficient.

If the user discovers they made an error or requires an immediate stop, the program will need a way to safely and properly stop the program. Another feature to this stop would include a way to ensure the circuit is no longer active, so that the user does not touch an active circuit and to ensure the circuit does not burn out past its limits.

This project will be able to be incorporated in many Electrical Engineering courses at Cal Poly. Specifically for the EE 346: Semiconductor Device Electronics Laboratory course, this project will be used as a curve tracer in order to be implemented. The program should be implementable in diode and transistor circuits. The program will have details such as the voltage and current limits as well as the data received by key points of observation. Data gathered may be measured in the pico- range; however, the

34461A multimeter only measures down to 100 microamps. For pico- range values, a hardware component will be added to adjust for these downfalls in the equipment.

The hardware built for this project will be added to the student's circuit such that pico- range values can be measured and obtained for the necessary graphs. Ideally this additional hardware will be added in series to the student's circuit. The physical component will be a compact box and will cost no more than $10. It is necessary for the part to be low in cost, so that student's lab kits do not increase in cost a substantial amount. The component will allow for pico- range measurements using a micro- range tool. Additional script may need to be added for accurate calculations and comparisons with all the data collected from the circuit.

To test the hardware component of this project, a known pico- range value will be measured using the component and the 34461A multimeter. The known value can be measured though a multimeter that can access values in the pico- range. The measurement difference will be tested, calculated, and improved until the average error is less than 10%.

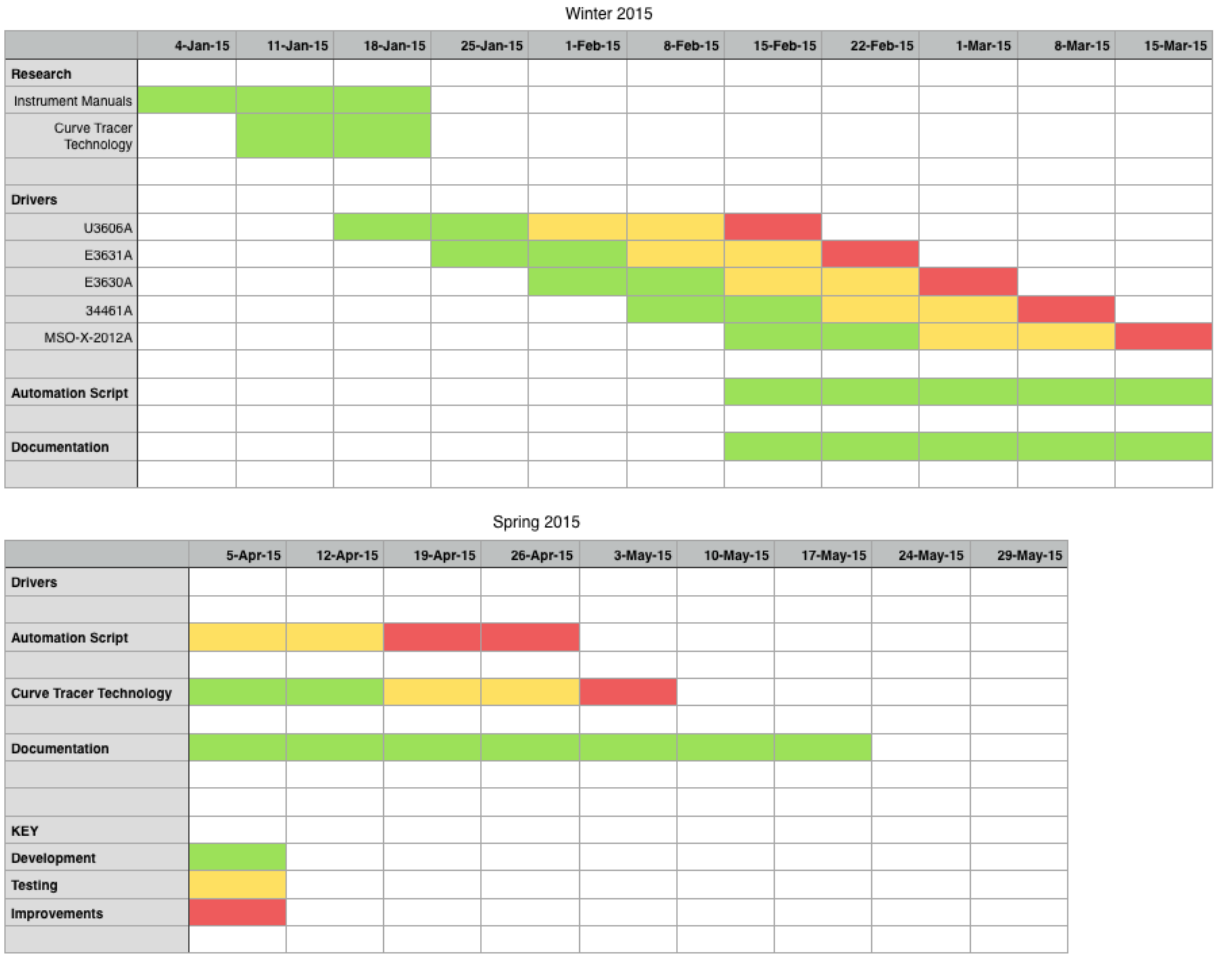

Figure 6: Gantt Chart for Winter and Spring 2015 Quarters

The Gantt Chart in Figure 6 shows that development, testing, and improvements will overlap. When developing drivers and automation scripts, it is good to test as they are created so that problems and errors can more easily be located and solved. When entire drivers or automation scripts are tested only after they are written, there can be larger errors that incorporate more than one portion of the script. Testing while development helps prevent this problem. This similarly applies to testing and improvements simultaneously occurring so that the new improvements to the scripts can be ready to be implemented.

If an instrument is lacking in any expected functions, time to finish the task will be delayed. For example, if it is necessary to check the current at any given time, but there is only a function for checking the user's set current value, a different way of determining the varying current will be needed. This new way of determining the current will not be as direct as using a function.

If there are more errors than expected or lack of functions needed to achieve laboratory class experiments, the schedule can be delayed. Research includes both learning how to use Python as well as determining the functions of each test instrument. Since the Cal Poly EE flowchart does not include LabVIEW, learning LabVIEW can affect the schedule and delay the start of the development of the drivers.

The sooner the automation for Room 20-126 is completed, Cal Poly will optimize the usage of the room. Utilization of the room increases the use of the new instruments more fully. Room 20-126 contains newer technology that benefits students to grow accustomed to various instruments as well as benefits in using more advanced test equipment compared to the other laboratory classrooms within building 20.

To test the drivers, one function will be created at a time. The functions will be tested by running each driver subVI one at a time. From this, observations of the expected commands should be seen in the instrument. Since each function is tested on its own, it will be clear which function is returning error. The automation program will be tested after connecting the subVIs together as well as outputs to view that the expected values equal the actual output values. Validation can also be seen on the test instrument and through the use of the oscilloscope on the bench. For example, if the voltage limit is set to 5V, the power supply will display a 5V limit and the oscilloscope will display a line 5V above the ground level.

Functional Decomposition (Level 0 and Level 1)

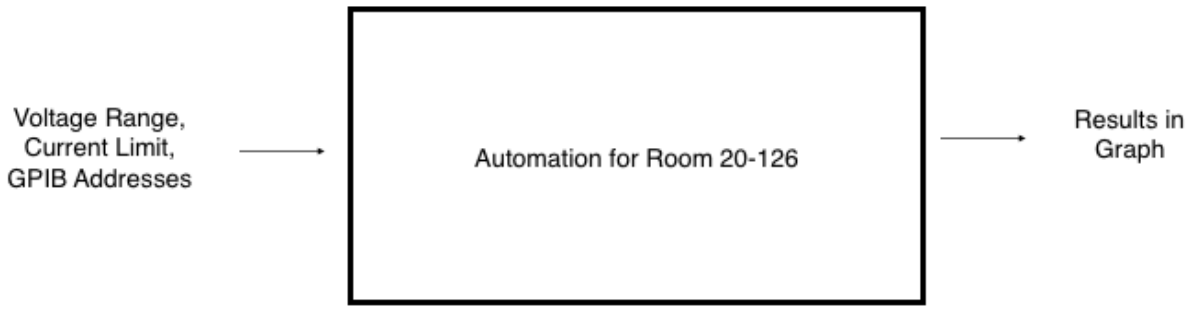

Figure 7: Level 0 Block Diagram

Figure 7 shows the overall block diagram of the automation program. The script will only need the voltage range, current limit, and GPIB addresses to compile data into a graph.

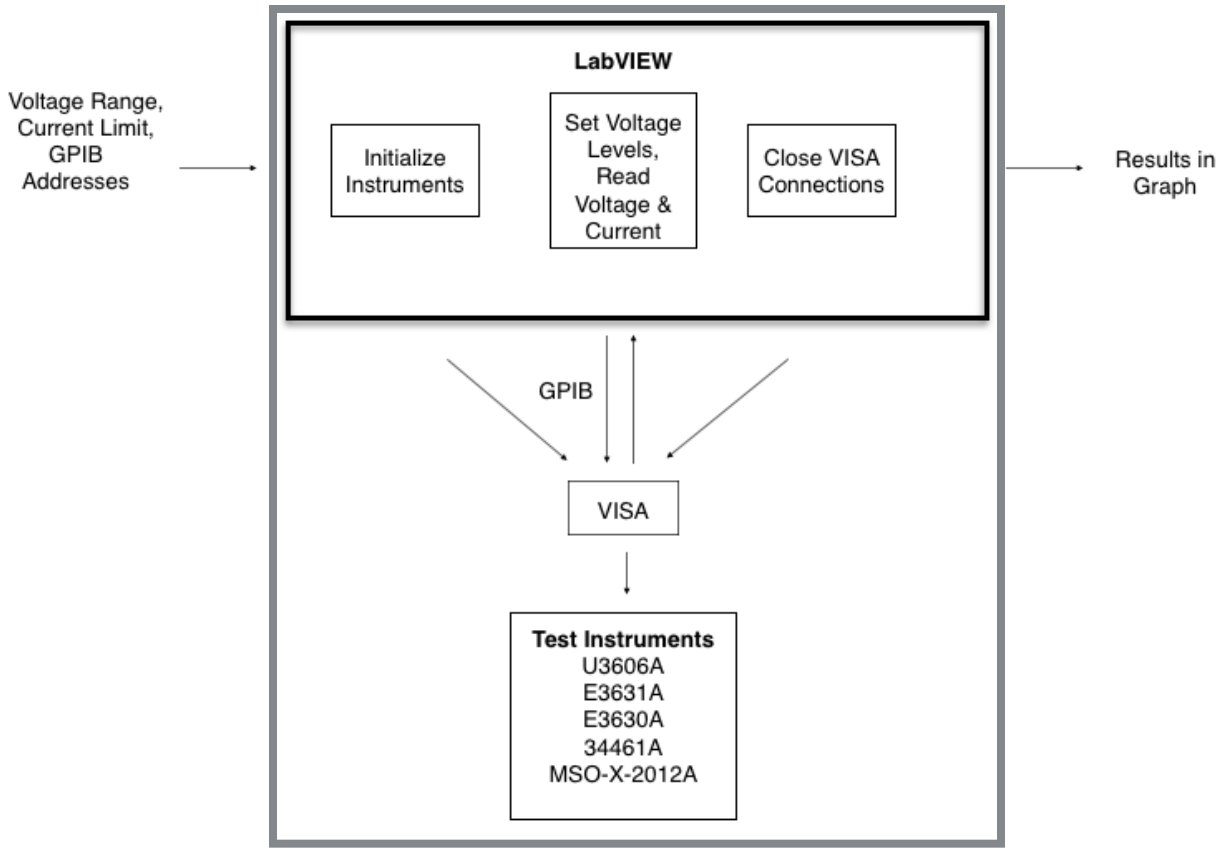

Figure 8: Level 1 Block Diagram

In Figure 8, the block diagram describes the inner flow of the automation program. The automation program will control all of the test instruments and will refer to the drivers to properly call each function.

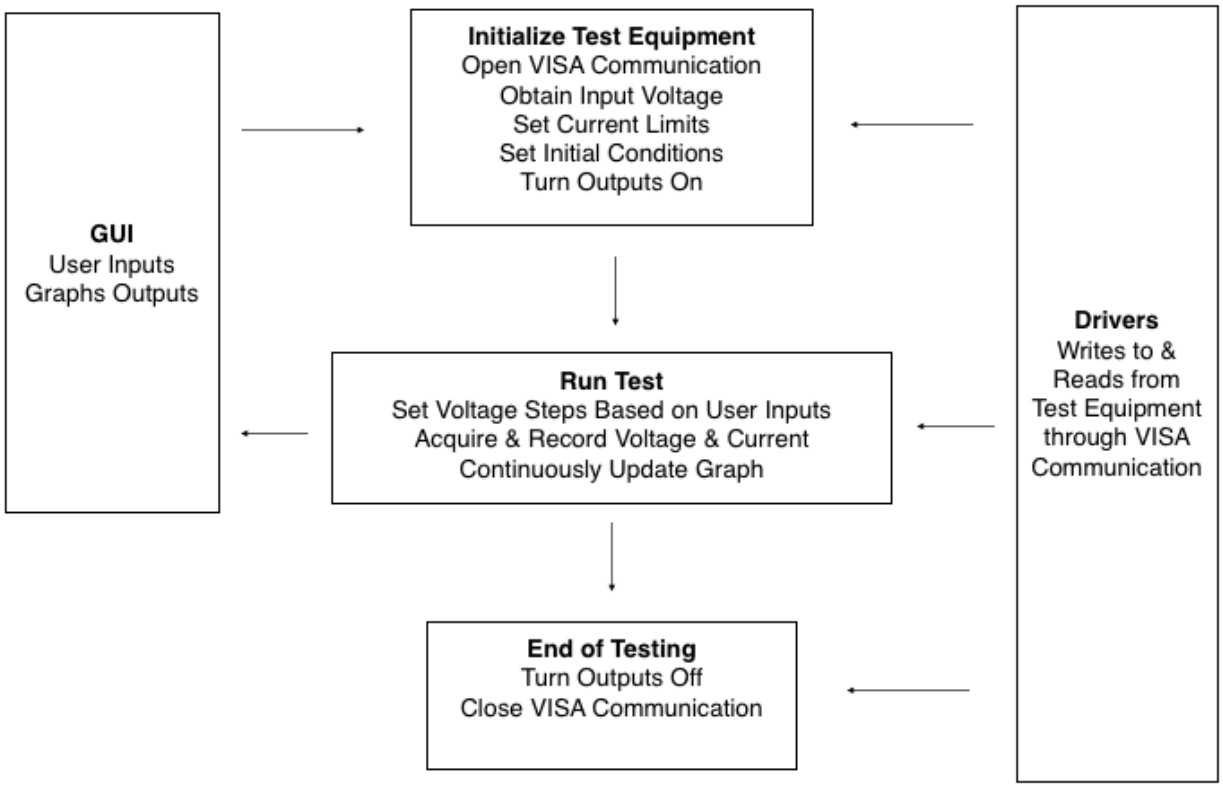

Figure 9: Software Block Diagram

Figure 9 shows the high level block diagram for the LabVIEW program. LabVIEW will utilize the commands from the drivers/subVIs to control all the test equipment through VISA communications. The drivers are accessed throughout the entire flow of the automation program for initializing VISA controls, setting test equipment, reading from the equipment, and closing the VISA control communication. Drivers are helpful in making the script easy to read, rather than using various SCPI codes and VISA blocks that do not make sense to a new user. The drivers, also called subVIs in LabVIEW, contain the necessary SCPI commands for the various function block names. For example, one block will solely initialize the Agilent 34461A 6 1/2 Digit Multimeter because the necessary commands differ from that of the Agilent E3631A Triple Output DC Power Supply. The power supply will require a command to tell the machine to set the voltage output, current limit, and then turn the outputs on; whereas the multimeter does not need any additional commands from clear and reset.

Program Design

First Iteration

The first iteration applied the basics of communicating with the test instruments via Python. Upon starting the project with Python, the necessary syntax for VISA communications was unknown due to the minimal exposure in programming in Python. To gain access to all the VISA communications for each instrument as well as designing a main automation script that will be easy to understand, multiple files that the main script can reference provides these features. Separate files for each instrument keep all commands organized. The main file would then call each of the files, so it could use the commands in all of the files without defining the commands in the main. The main file's appearance would be easy to read due to the minimalistic format.

Issues with Python became more apparent when looking for either an easily implementable or programmable user interface. The user interface holds importance because it allows the user to be unknowledgeable about the program, but still able to use the benefits of the program. Although students would have a background in Python, learning and understanding the script would take up time needed to understand the component tests. Understanding the script would benefit students, but is not a priority in this project. Hence, the change to a new language with a user interface.

Second Iteration

The second iteration changed the automation from Python to LabVIEW. LabVIEW creates a user interface as the program process is put together. The second iteration consisted of basic work due to the minimal understanding and experience with LabVIEW. In the second iteration, the diode test front panel solely consisted of three inputs: start voltage, end voltage, number of steps; and three outputs: start current, end current, and the diode current versus diode voltage graph as seen in Figure 10. The layout of the front panel was extremely basic and did not include user information details. The VISA communication capabilities took priority in the second iteration, rather than the appearance of the user interface. The instruments used for the diode test include 2 Agilent 34461A 6 1/2 Digit Multimeters and 1 Agilent E3631A Triple Output DC Power Supply.

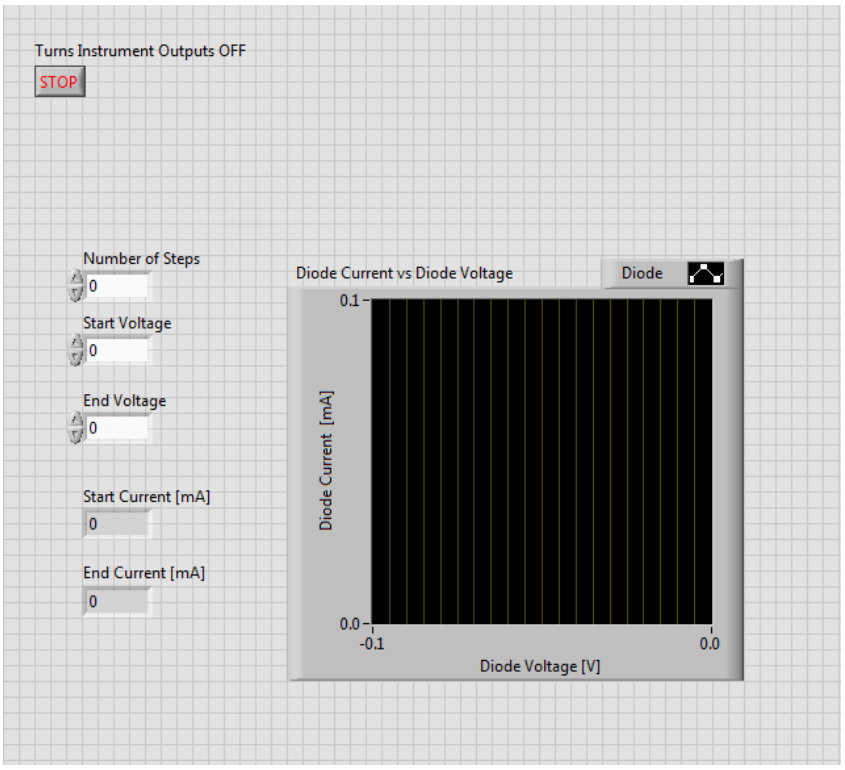

Figure 10: Diode Test Front Panel — Second Iteration

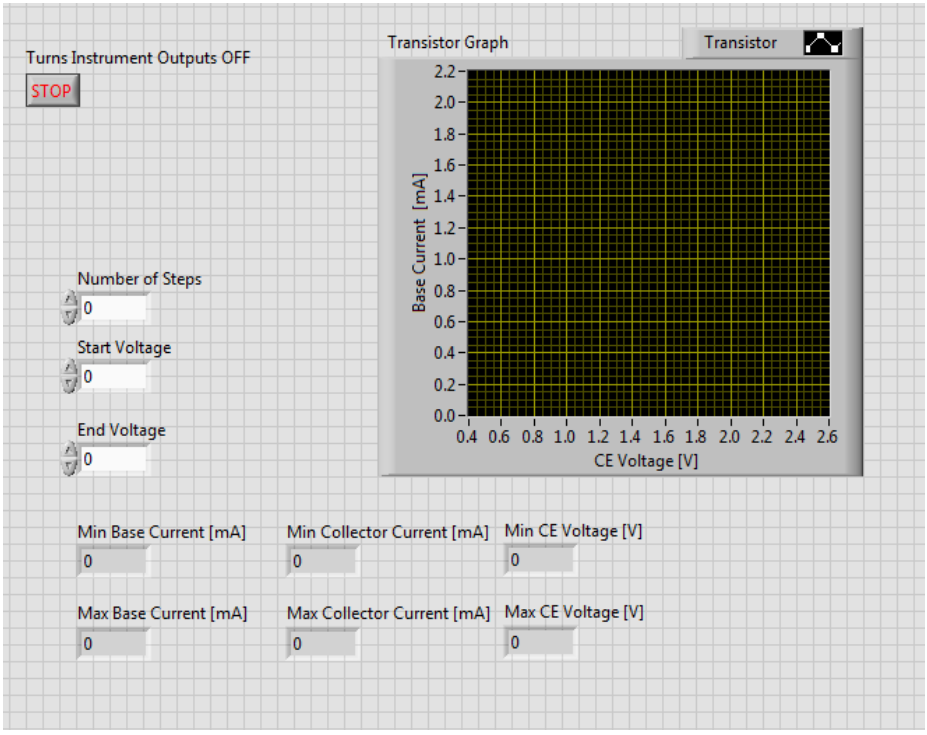

Figure 11: Transistor Test Front Panel - Second

The transistor test front panel follows the basics of the diode test, but since transistors require more inputs and outputs, which was provided by the Agilent U3606A Multimeter/ DC Power Supply because it can measure voltage or current values as well as provide power. The transistor used the same inputs as the diode test, but its outputs included the min/max base current, min/max collector current, the min/max collector-emitter voltage, and a graph of the current and voltage.

The wiring diagram was also basic and did not utilize any subVIs. VISA communication between the software and the test instruments was used through a VISA communication block that allowed writing, reading, or querying. SCPI commands could be entered one at a time, but time delays between each command was unclear. User manuals for the instruments state the required time delays between each command. This is an example of a block that the user does not know what occurs within the block.

The second iteration was a big step from the first because it consisted of a user interface as well as the ease of understanding the program process. Although the user interface appears to have very minimal information, the control of each of the test instruments was present. The necessary features to control the GPIB/USB connected instruments decreased with this change from Python to LabVIEW, which in turn decreases time to the final product.

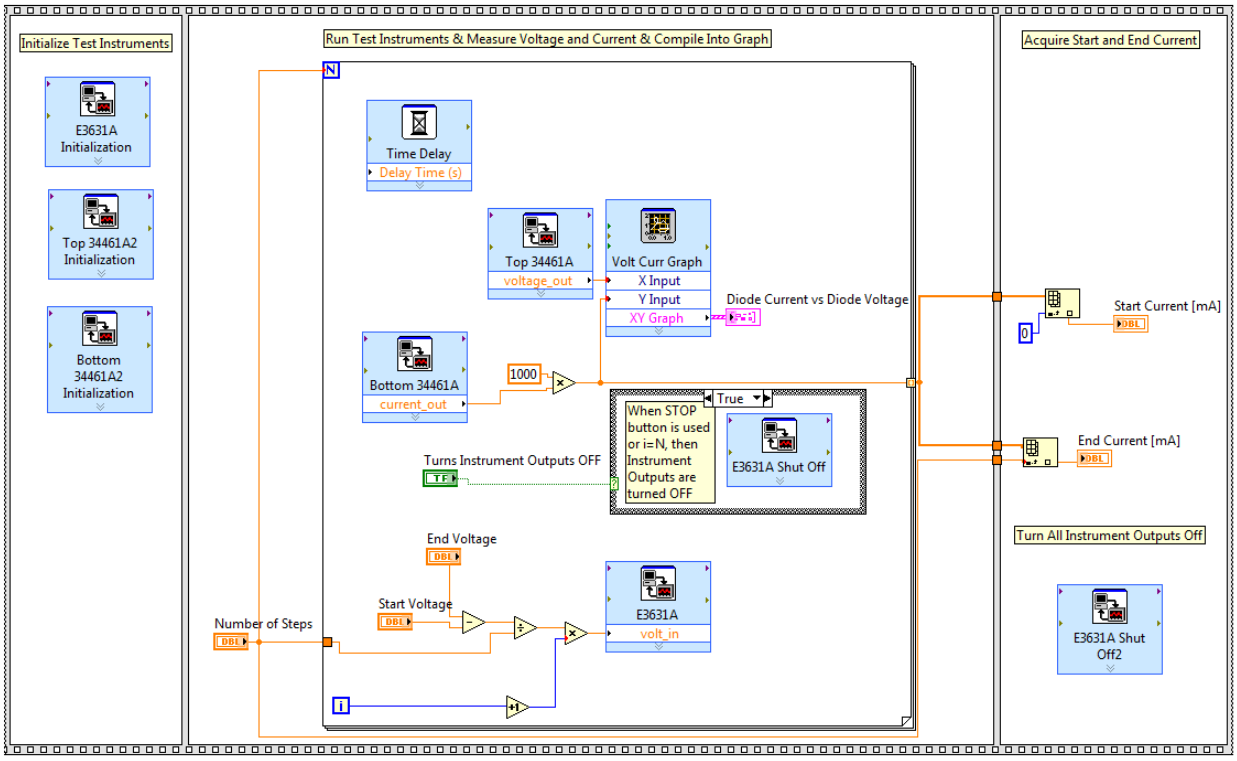

Figure 12: Diode Test Wiring Diagram - Second Iteration

The wiring diagrams were also not as developed as shown in Figure 12 and Figure 13. VISA communications accessed through Instrument Assistant blocks used in this iteration are not as reliable. The process which these blocks access VISA communications is unclear and time delays are unknown between each command. Since the data sheets for each of the instruments describes the necessary time delay between each command, future iterations should add time delays to ensure the test instruments write/read each command.

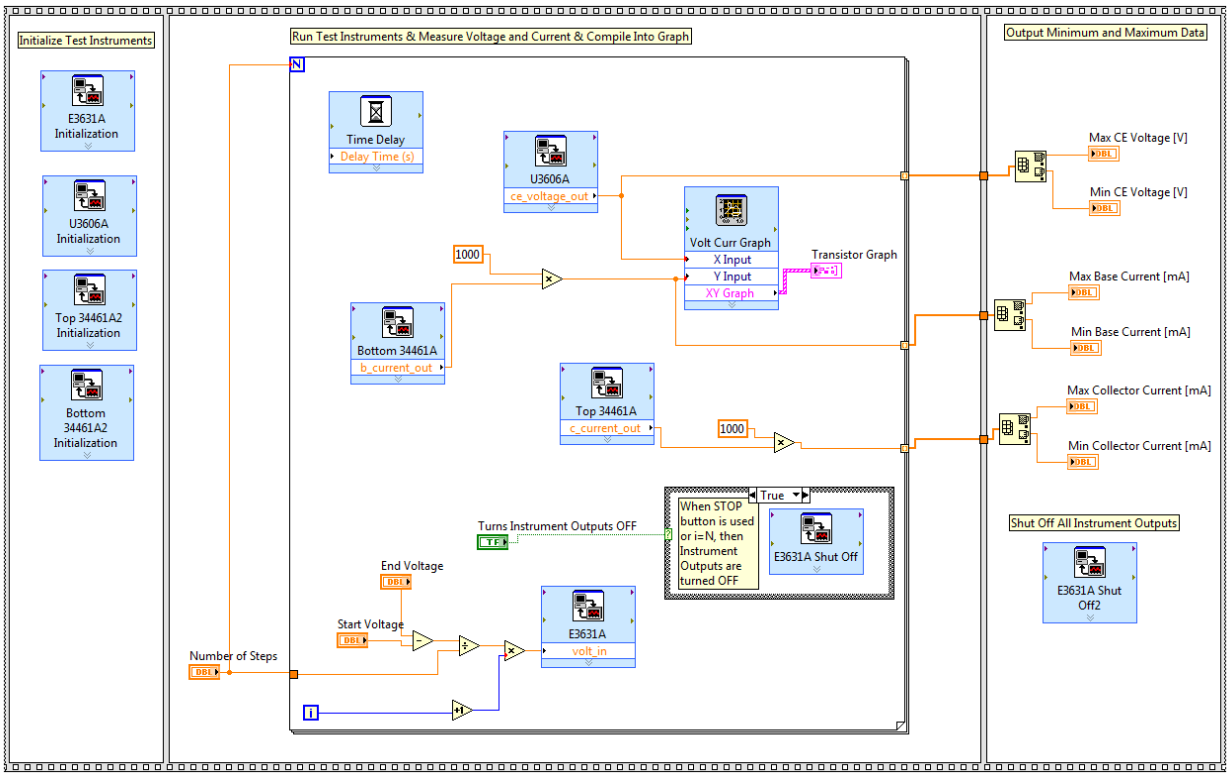

Figure 13: Transistor Test Wiring Diagram - Second Iteration

The downfalls in need of fixing included the lack of information given to the user. LabVIEW includes capabilities to add text to the front panel as well as the properties includes a documentation section that allows for information about each of the blocks or inputs/outputs. Context help will allow the user to scroll over a block/input/output and the documentation will appear. If a user is scrolling over a block in the wiring diagram, the documentation also includes the types of input/output connections.

Third Iteration

The iteration includes many improvements including a description of each of the tests, instructions/information for each of the test instruments used for the test, a circuit diagram to show the connections to perform the test, and the ability to choose the GPIB/ USB addresses for each instrument. This allows the user to use any address, rather than a set address. The previous VISA communication blocks did not allow for an address input, rather the addressed remained constant, depending on the programmer's preset address. Figure 14 shows the diode front panel for the third iteration with its improvements. Figure 15 shows the same, but for the transistor test.

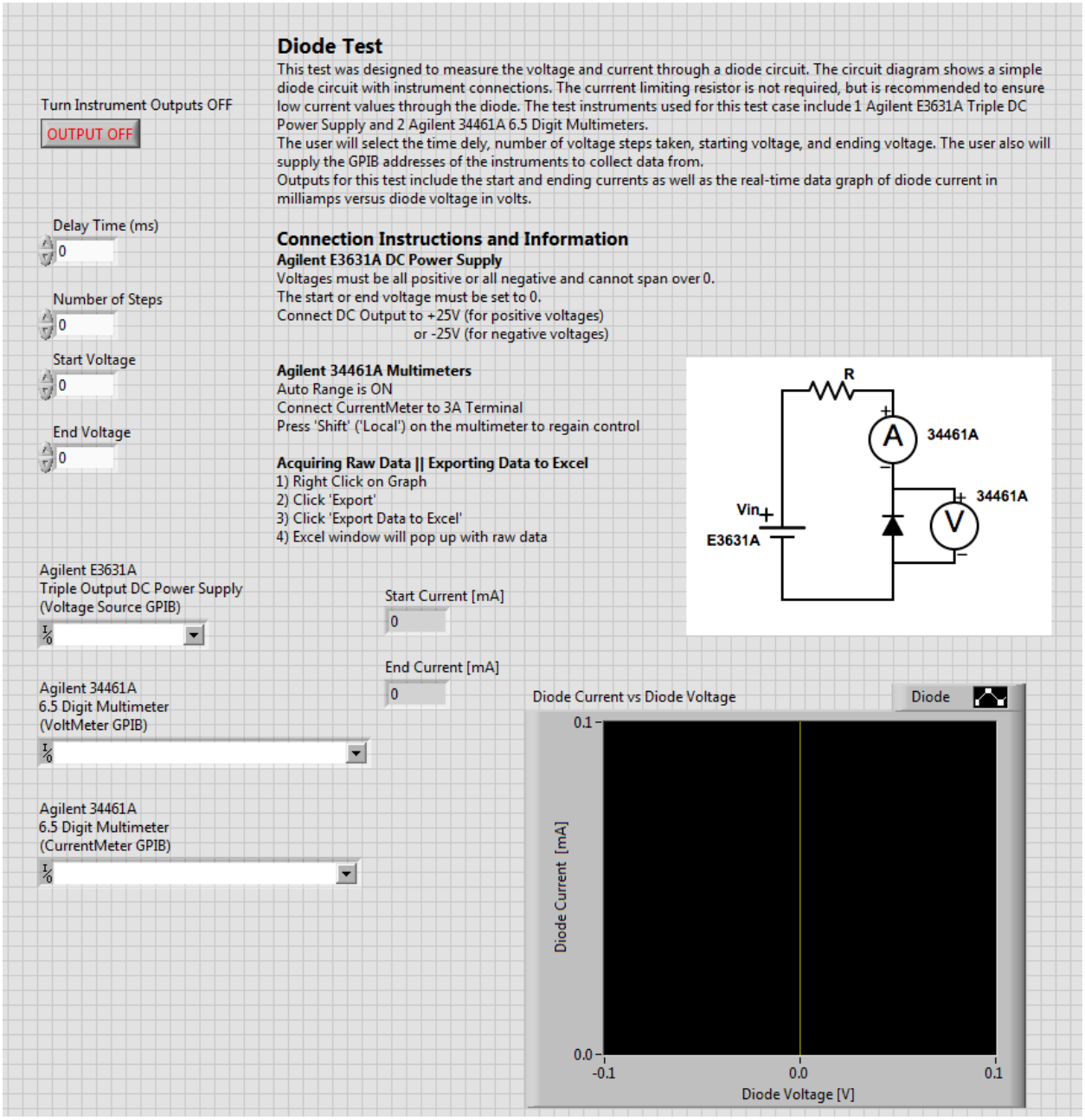

Figure 14: Diode Test Front Panel - Third Iteration

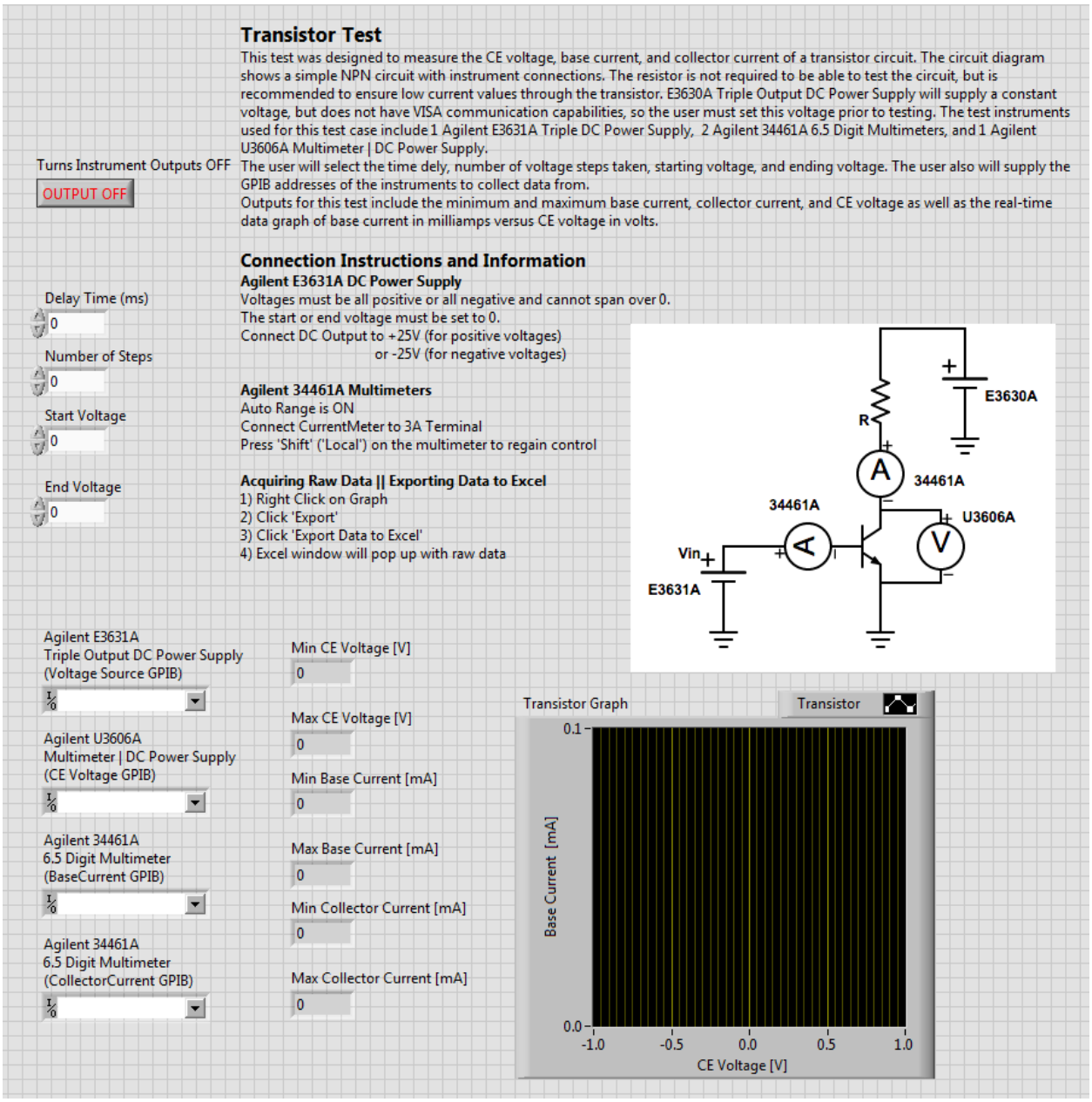

Figure 15: Transistor Test Front Panel - Third Iteration

The software for this iteration improved as well by implementing subVIs, a VI accessed by the main VI. Implementation of different VISA communication blocks made the process more clear because write, read, and close VISA blocks and time delays ensure that test instruments have enough time to read/write each command. SubVIs improve the appearance of the main VI because the new VISA blocks required more input information like GPIB/USB addresses and byte count. A main VI without subVIs would appear more messy and complicated, especially with time delays between each command. Figure 16 and Figure 17 show the new wiring diagrams for the third iteration.

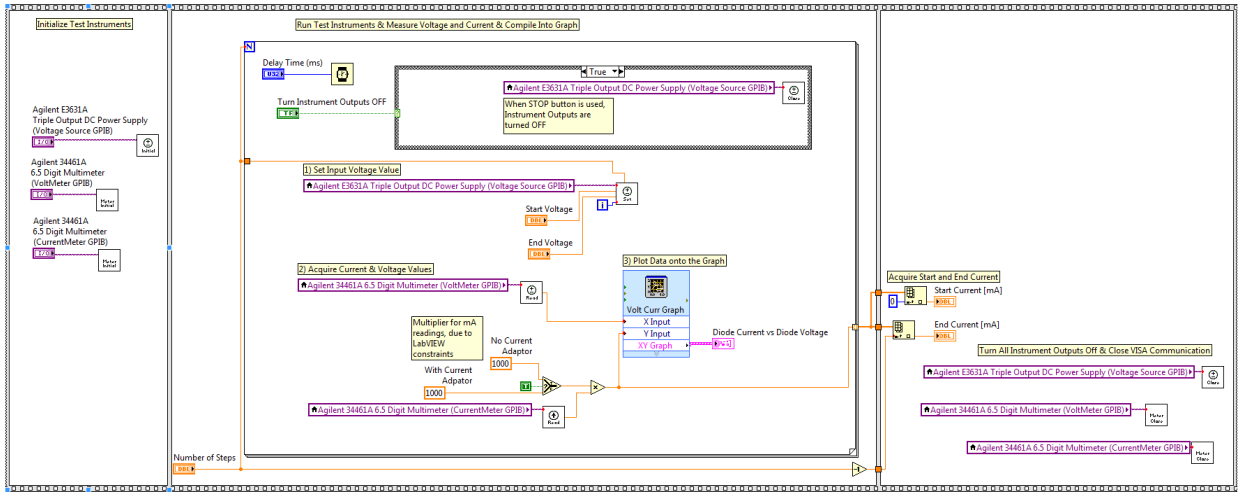

Figure 16: Diode Test Wiring Diagram - Third Iteration

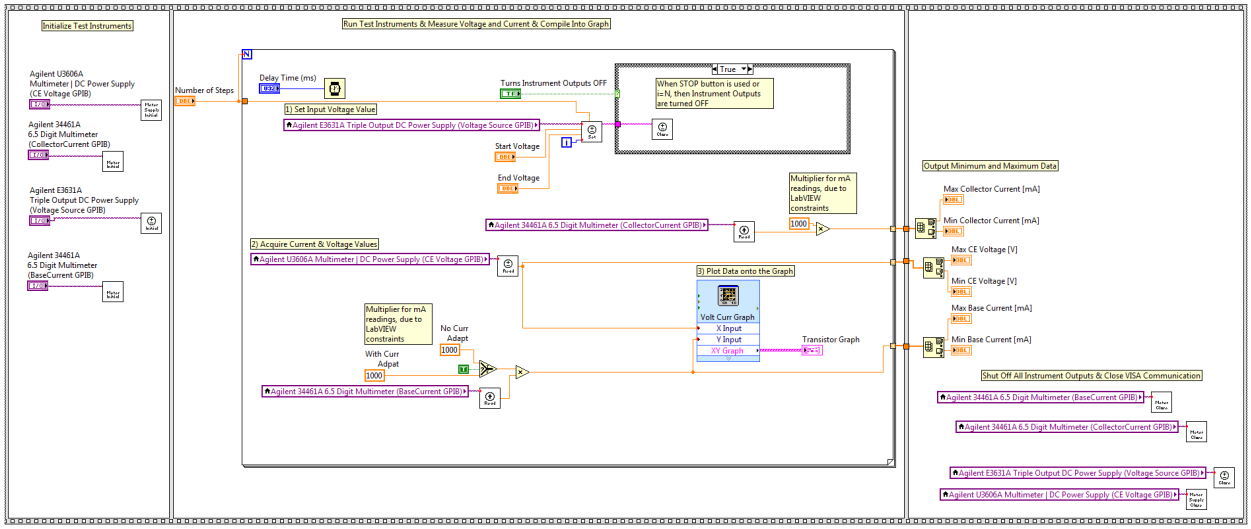

Figure 17: Transistor Test Wiring Diagram - Third Iteration

The next steps to improving the programs include more instructions and information for each of the inputs/outputs. At this point, when the user moves their mouse over any of the inputs/outputs, context help did not display any information. Also, the tests should test diodes in both forward and reverse bias and any type of transistor whether NPN, PNP, or MOSFET. Generalizations allow for universal usage.

Fourth Iteration || Final Design

The fourth iteration's improvements include many cosmetic changes to improve user interface and increase universal usage. Changes for both tests include descriptions for every input and output, sectioned items on the front panel, larger output value slots, finer values shown on the graph, directions for turning the test off in the cases of a needed stop, and implementation of generalizations. Descriptions help users to understand the purpose for each input/output, so they can optimize the usage of the tests. Sectioning the inputs/outputs provides the user with a clean and easy to read user interface. Larger output value slots allow the user to read the entire number, else the user can only see a partial of the output numbers. Finer values on the graph enable Excel to receive the fine tunings of the graph, since the instruments can provide up to six significant digits. A stop may be needed if the user realized they connected something incorrectly or needed to restart the program run. The stop would ensure the instruments outputs are turned off to avoid damaging their circuit.

Generalizations added for both the diode and transistor circuits enable the tests to be used for more applications, rather than just forward biased diodes or NPNs. The diode test, in Figure 18, shows two diode circuit diagrams: forward bias and reverse bias. Although only one connection changes, the usefulness for the test increases. Figure 19 shows the transistor test with a general box for the user to place any transistor in place of the box. For example, if the user wanted an NPN, I1 and V1 labels the base current and voltage, I2 and V2 label the collector current and voltage, and V23 labels the collector-emitter voltage. These labels can easily be implementable to PNPs and MOSFETs as well.

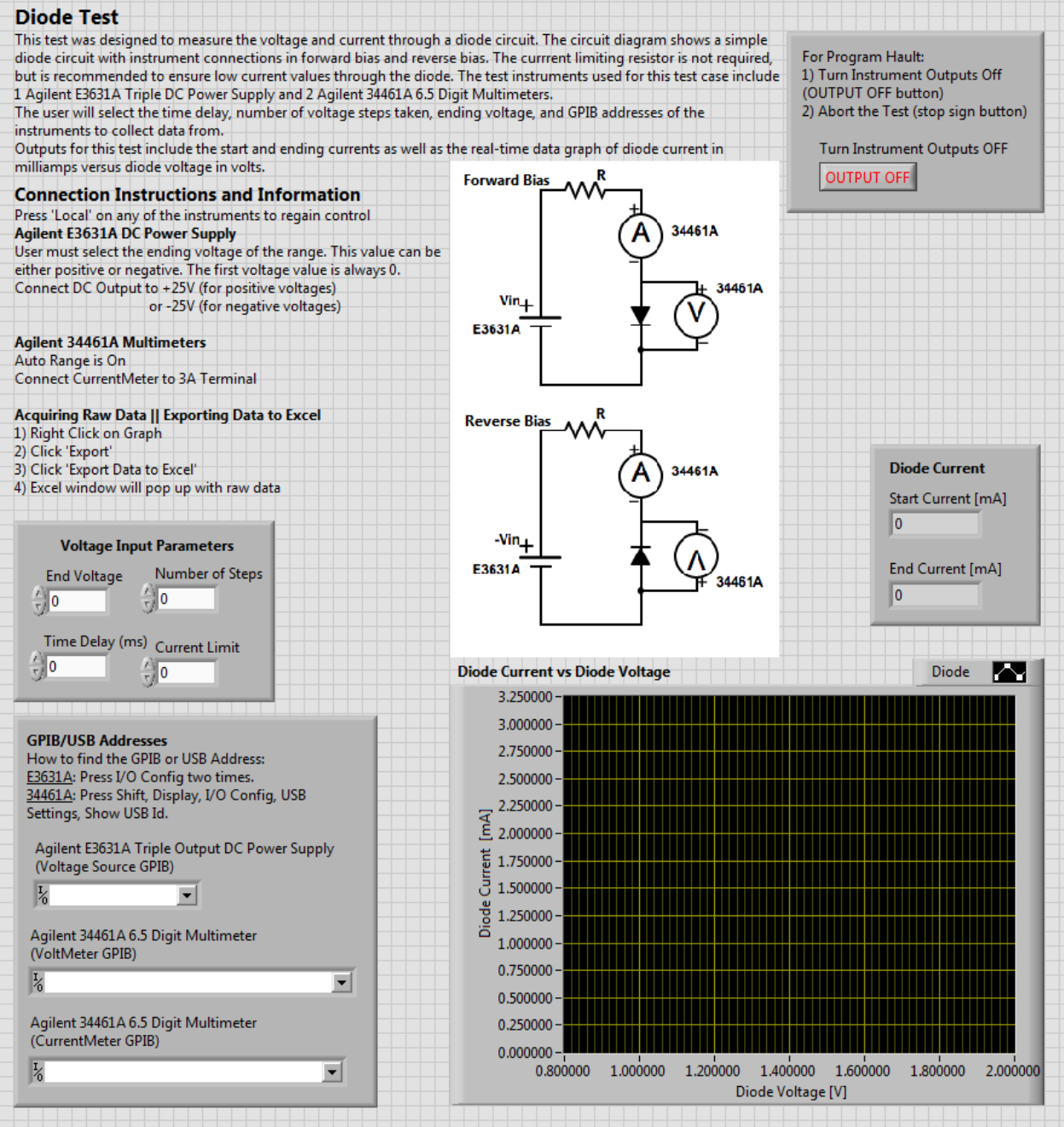

Figure 18: Diode Test Front Panel - Fourth Iteration

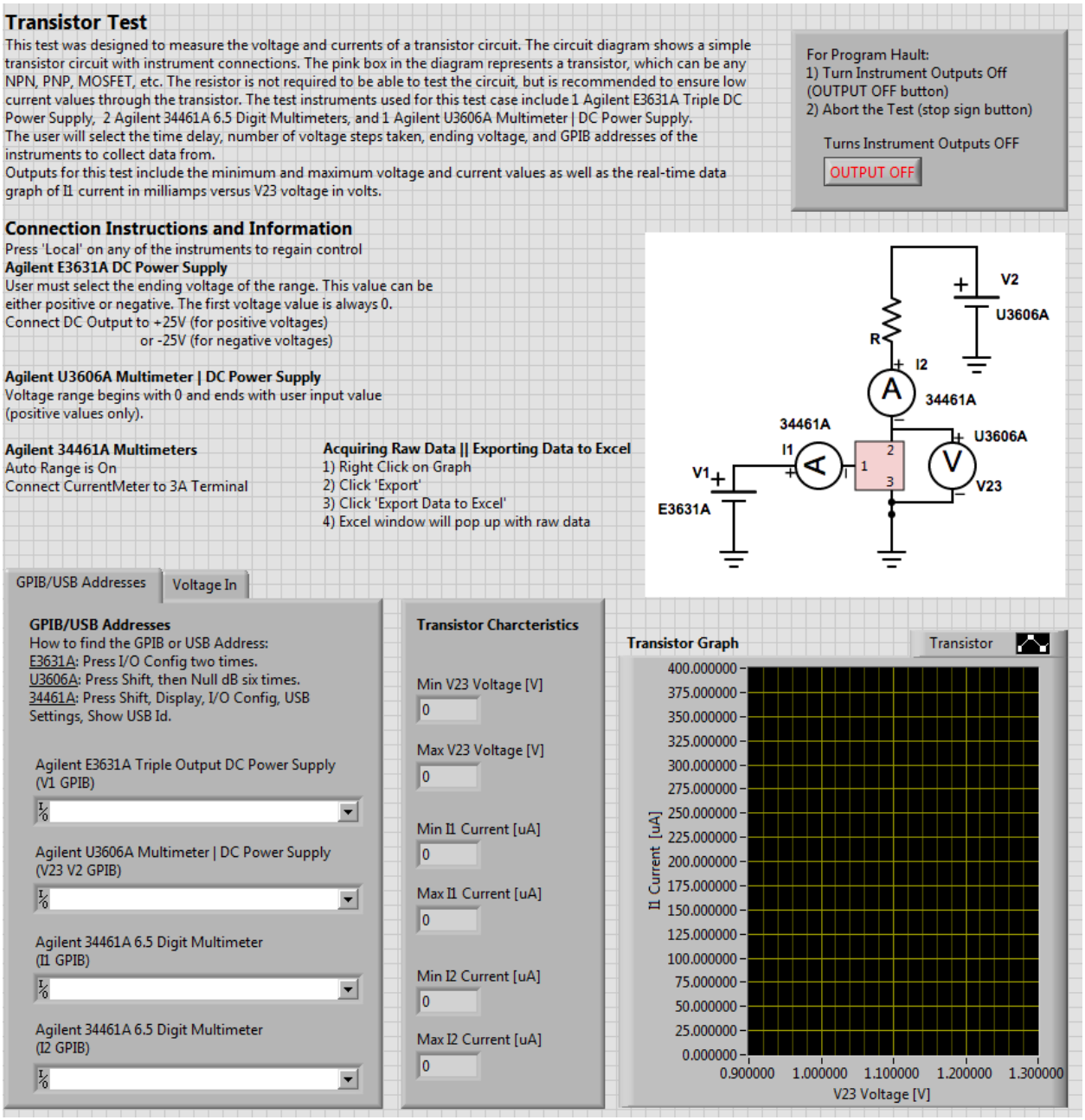

Figure 19: Transistor Test Front Panel - Fourth Iteration

In the fourth iteration, software updates were minimal and did not obviously affect the program. Error inputs and outputs were added to ensure that the program would first run one subVI, then the next, rather than running all subVIs in parallel. Running subVIs in parallel could cause error because the test instruments cannot handle more than one VISA communication at a time. Additional comments were added for future alterations and changes to the program.

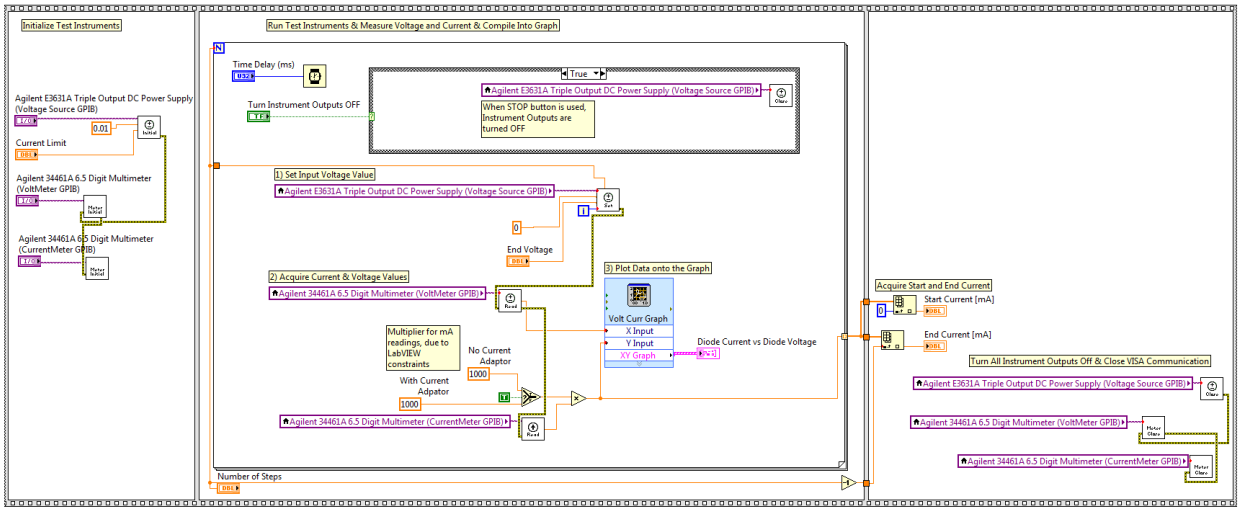

Figure 20: Diode Test Wiring Diagram - Fourth Iteration

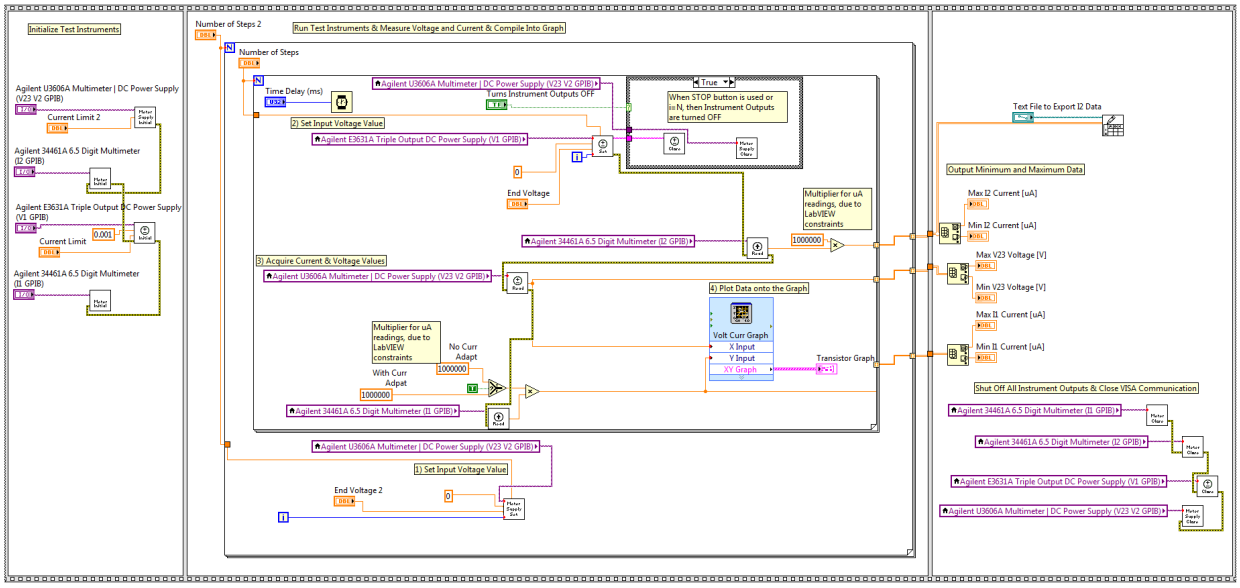

Figure 21: Transistor Test Wiring Diagram - Fourth Iteration

SubVI: Agilent 34461A 6 1/2 Digit Multimeter
Four subVIs use the functions from the Agilent 34461A multimeter. The first initializes the multimeter by clearing the reseting the machine, so there are no presets from any previous tests. VISA communication opens and the computer can communicate with the test instrument. The second reads current values the multimeter reads from the circuit. The third reads voltage values the multimeter reads. Lastly, the fourth subVI closes VISA communication. Figures 22-29 show the front panels and wiring diagrams of these subVIs. Each subVI contains differing SCPI commands with time delays between.

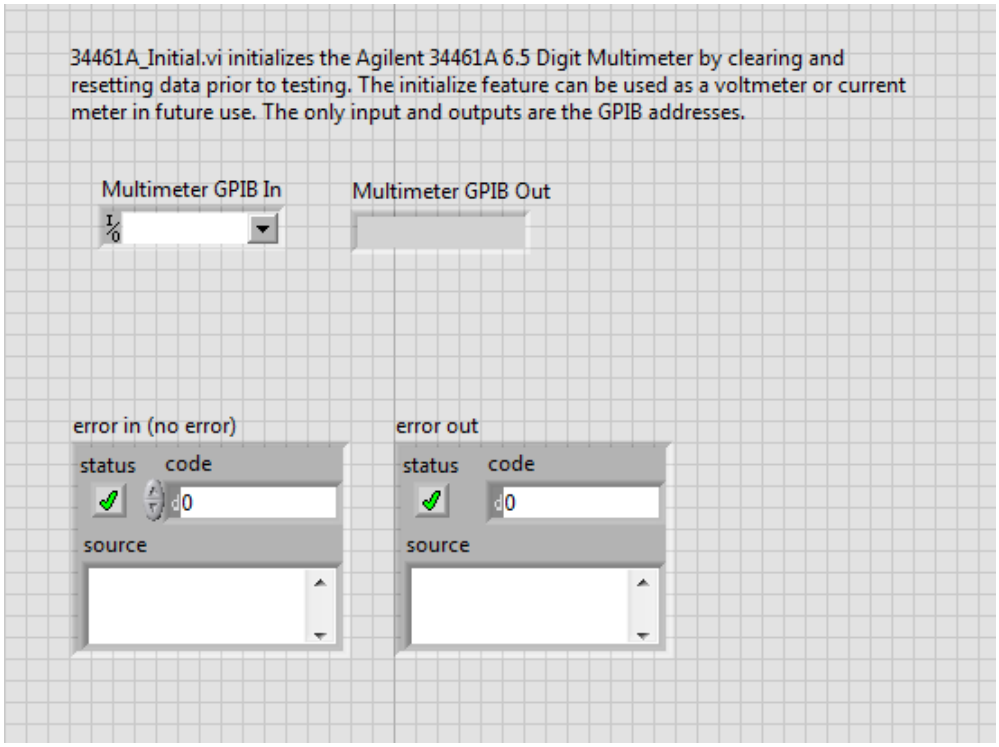

Figure 22: 34461A_Initial.vi Front Panel

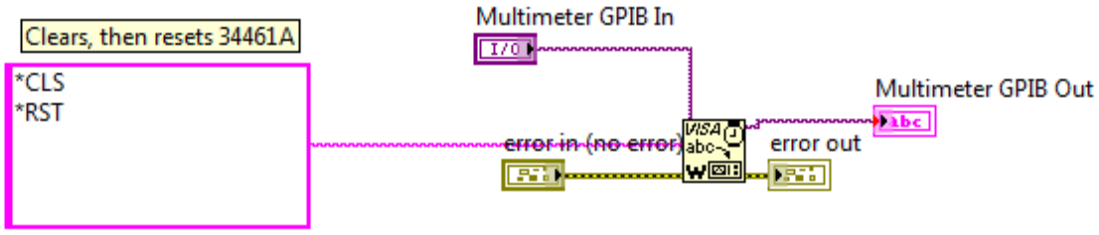

Figure 23: 34461A_Initial.vi Wiring Diagram

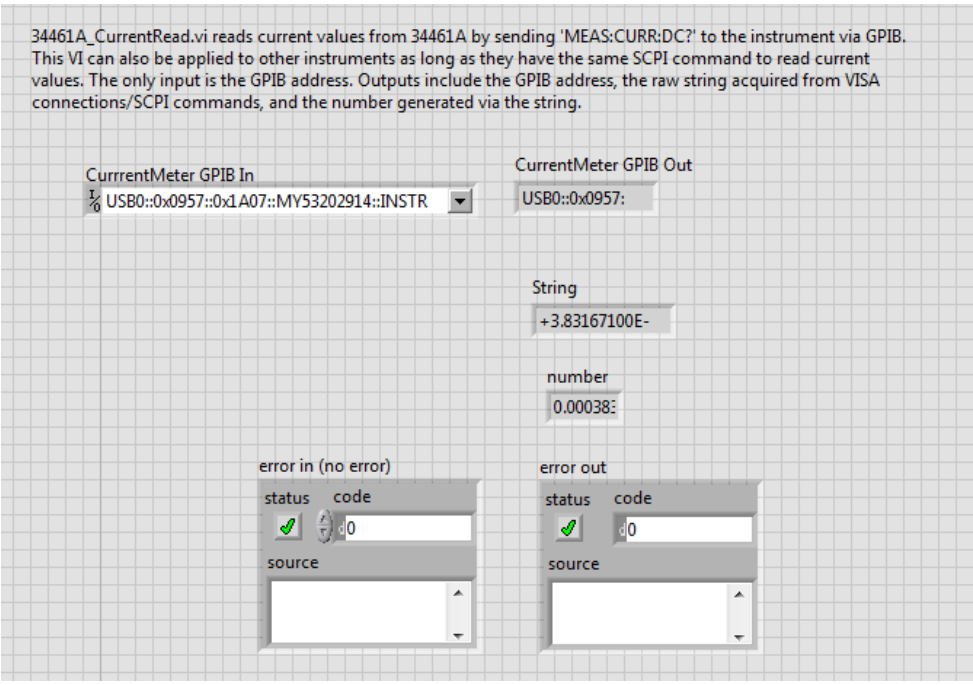

Figure 24: 34461A_CurrentRead.vi Front Panel

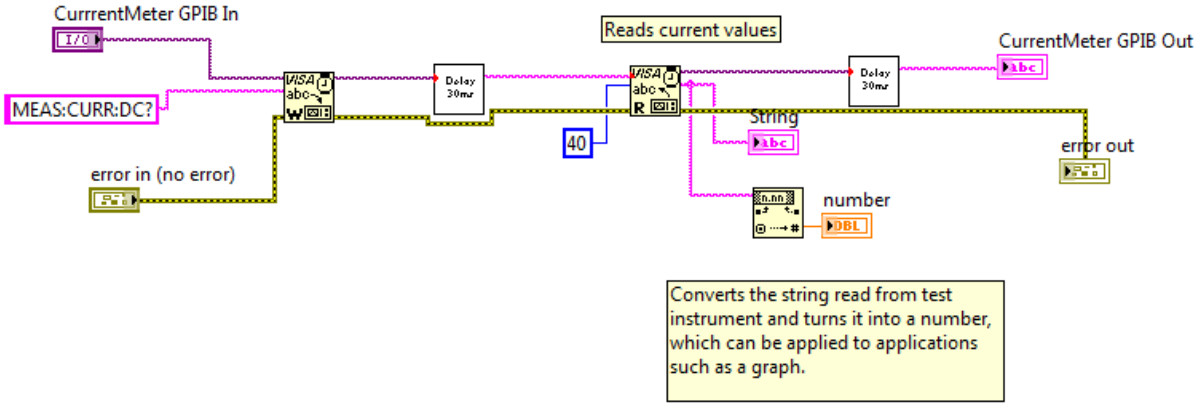

Figure 25: 34461A_CurrentRead.vi Wiring Diagram

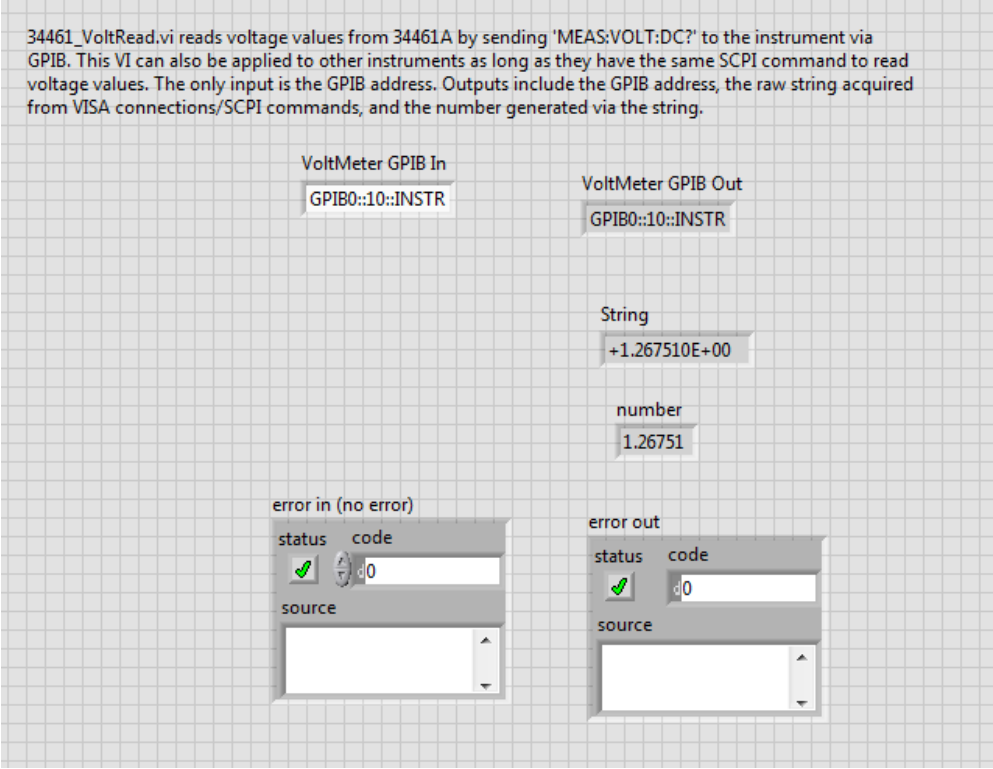

Figure 26: 34461A_VoltRead.vi Front Panel

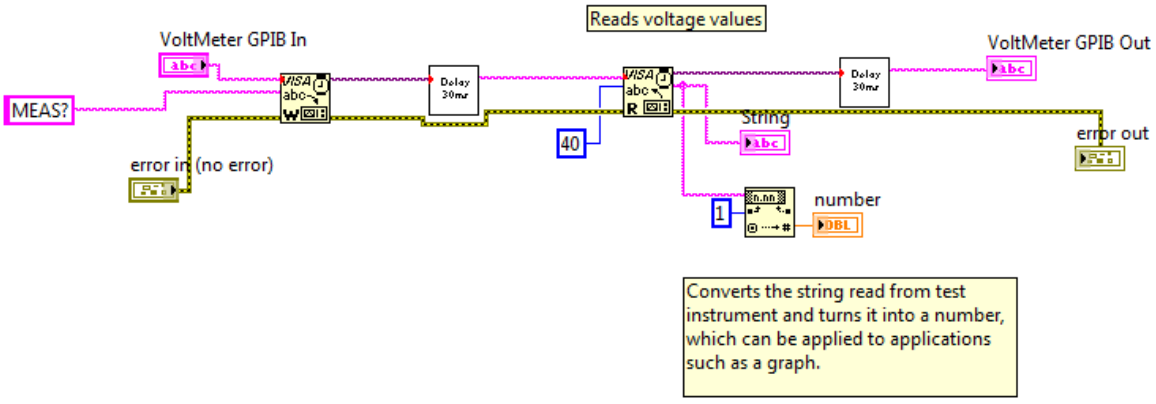

Figure 27: 34461A_VoltRead.vi Wiring Diagram

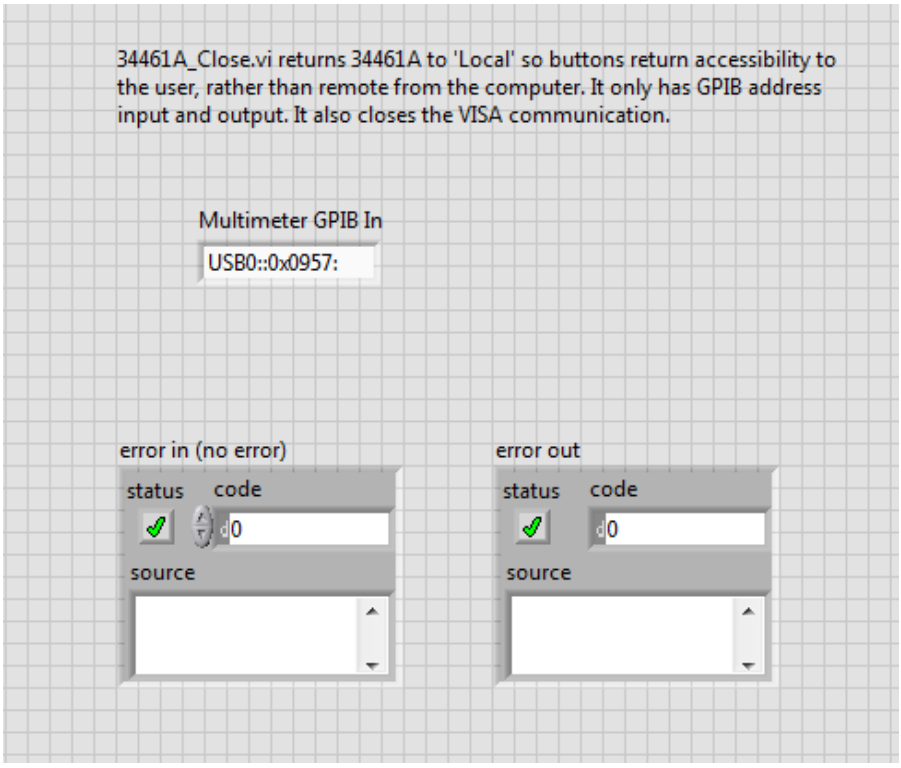

Figure 28: 34461A_Close.vi Front Panel

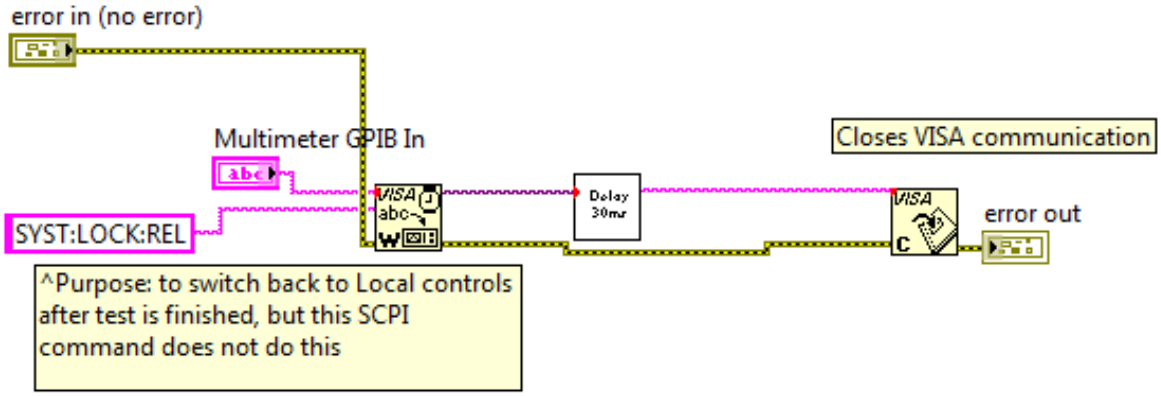

Figure 29: 34461A_Close.vi Wiring Diagram

SubVI: Agilent E3631A Triple Output DC Power Supply
Three subVIs help control the Agilent E3631A power supply. The first also initializes the power supply by clearing and reseting the machine, like the multimeter. However, the power supply also requires outputs and the current limit to be set and turned on. The initial output is set to 0V to avoid damaging the circuit. The current limit defined by the user sets in the initial subVI. The second subVI sets the voltage value based on the end voltage and number of steps. The subVI calculates each step based on the range the voltage divided by the number of steps, which provides equal steps. Lastly, the third sets voltage outputs to 0V to avoid damaging the connected circuit, turns outputs off, then closes VISA communication. Figures 30-35 show the front panels and wiring diagrams of these subVIs.

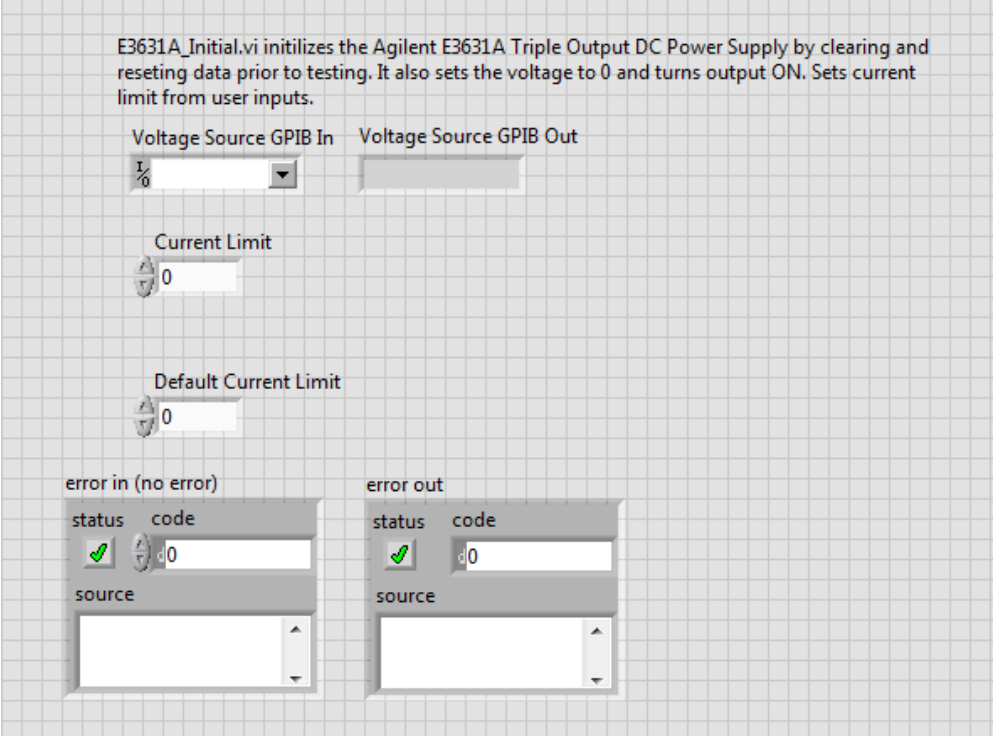

Figure 30: E3631A_Initial.vi Front Panel

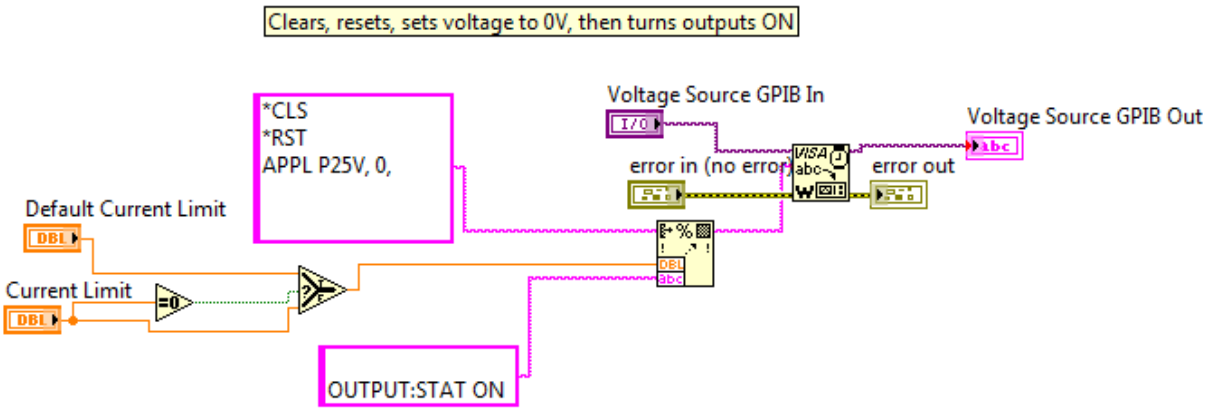

Figure 31: E3631A_Initial.vi Wiring Diagram

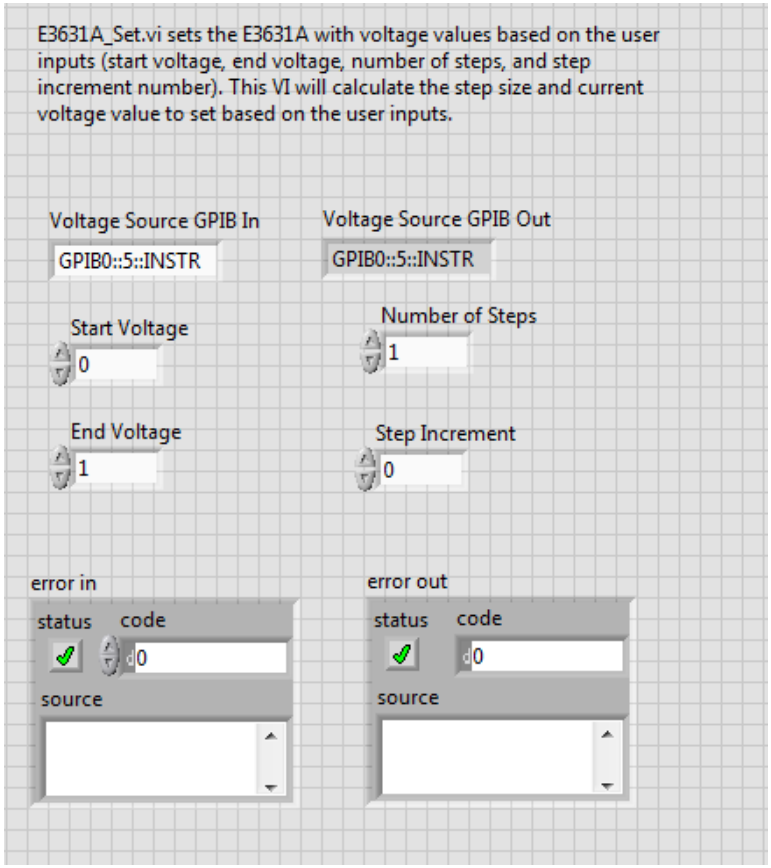

Figure 32: E3631A_Set.vi Front Panel

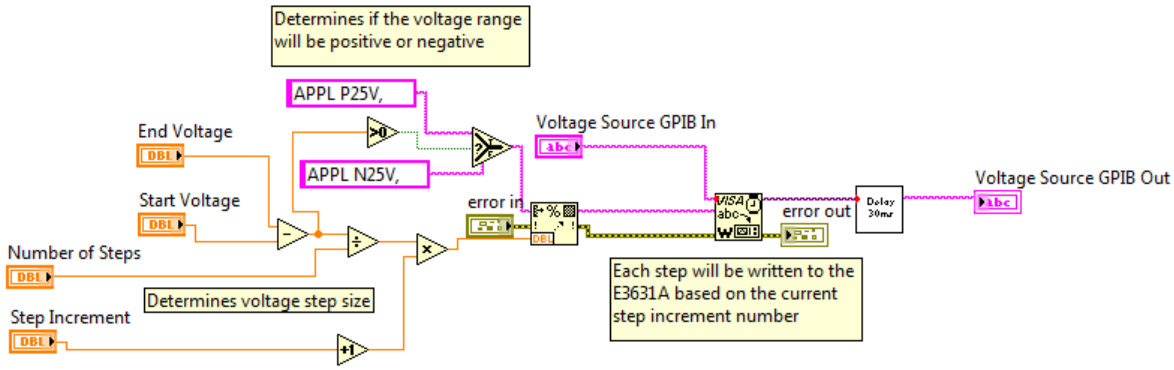

Figure 33: E3631A_Set.vi Wiring Diagram

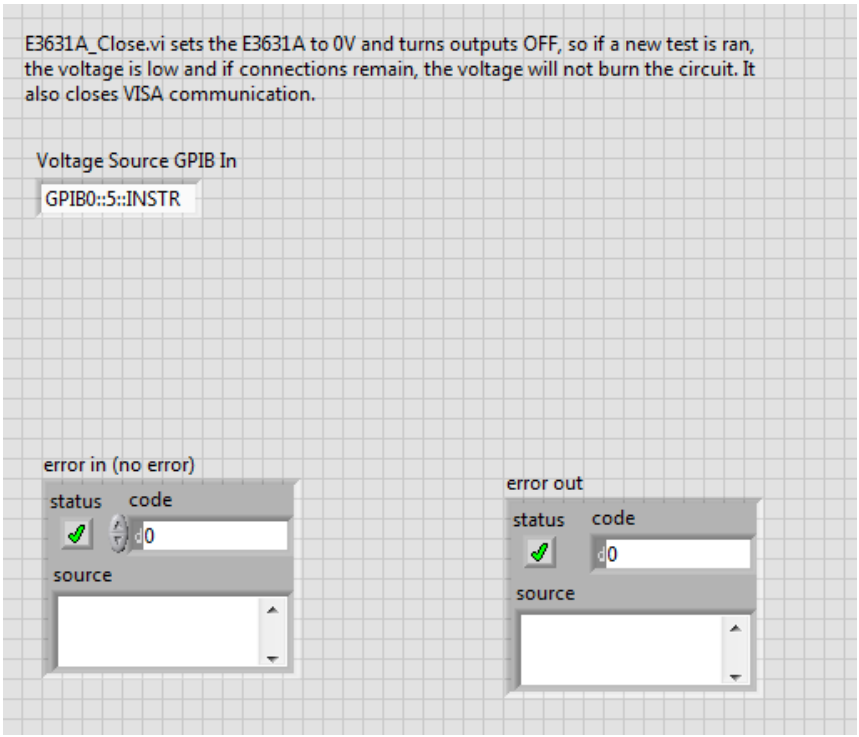

Figure 34: E3631A_Close.vi Front Panel

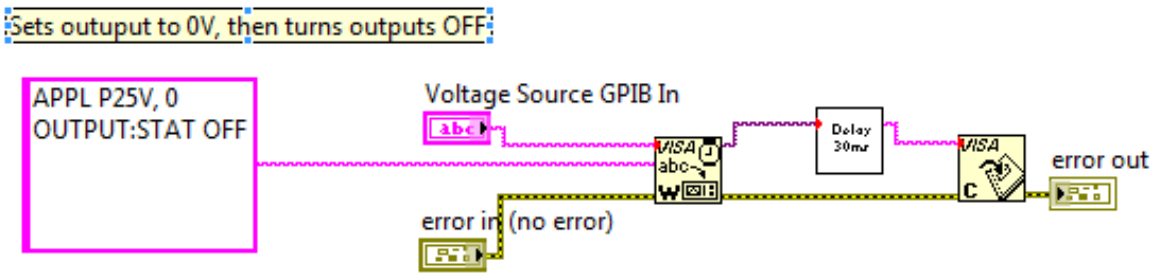

Figure 35: E3631_Close.vi Wiring Diagram

SubVI: Agilent U3606A Multimeter/DC Power Supply
Four subVIs control the Agilent U3606A, but one re-uses the Agilent 34461A Multimeter's read voltage subVI because the SCPI command is the same. The first, like the others, initializes the instrument through resetting, clearing, setting the current limit, and turning outputs on. The second, sets voltage values based on the user's inputs: End Voltage and Number of Steps. Lastly, the third subVI changes the voltage output to 0V to protect the connected circuit, turns outputs off, then closes VISA communication. Figures 36-41 show the front panels and wiring diagrams for Agilent U3606A subVIs.

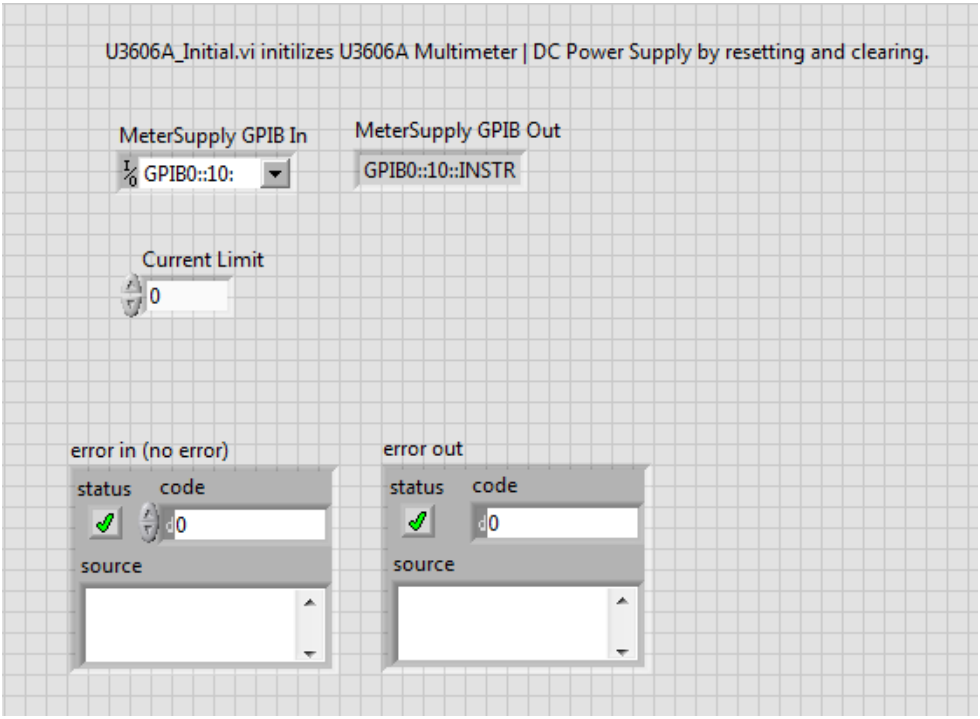

Figure 36: U3606A_Initial.vi Front Panel

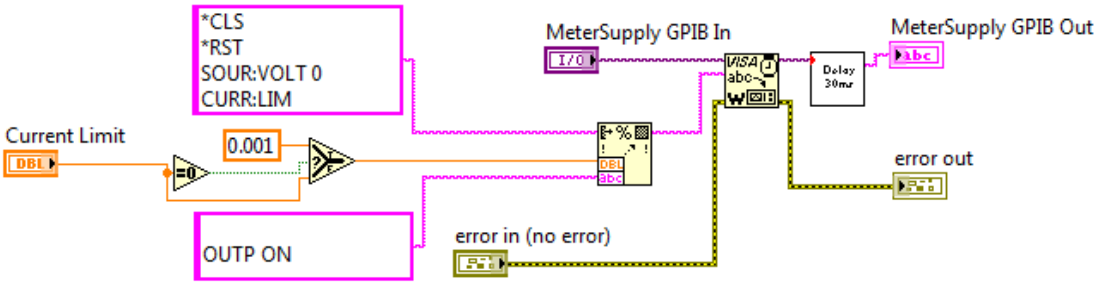

Figure 37: U3606A_Initial.vi Wiring Diagram

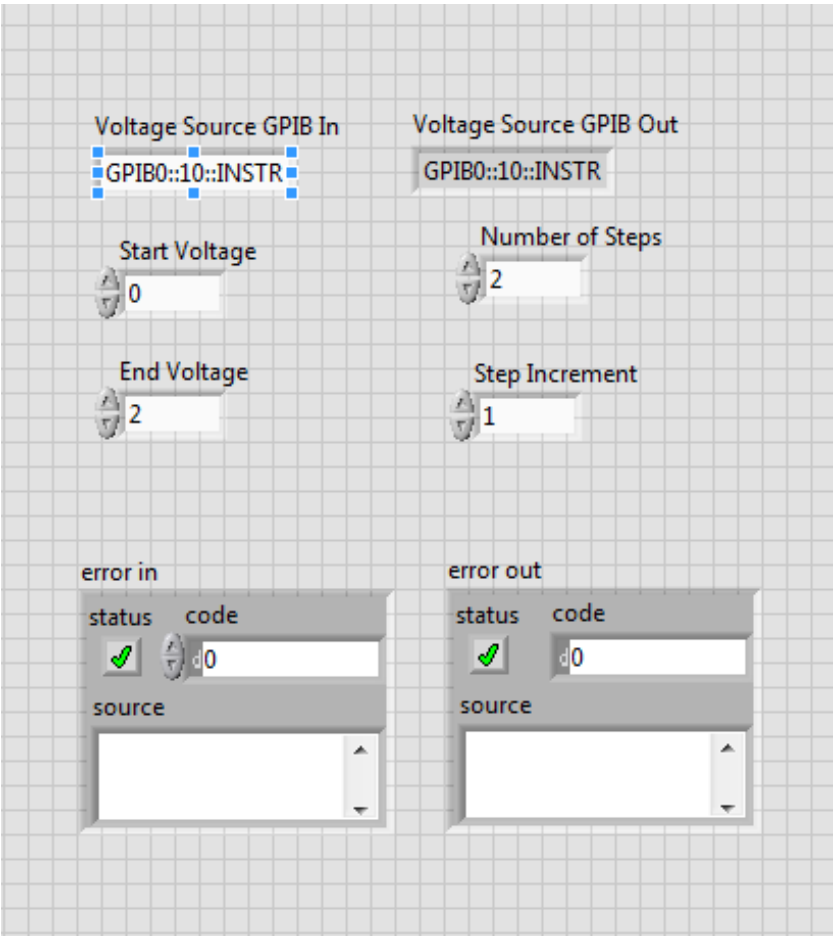

Figure 38: U3606A_Set.vi Front Panel

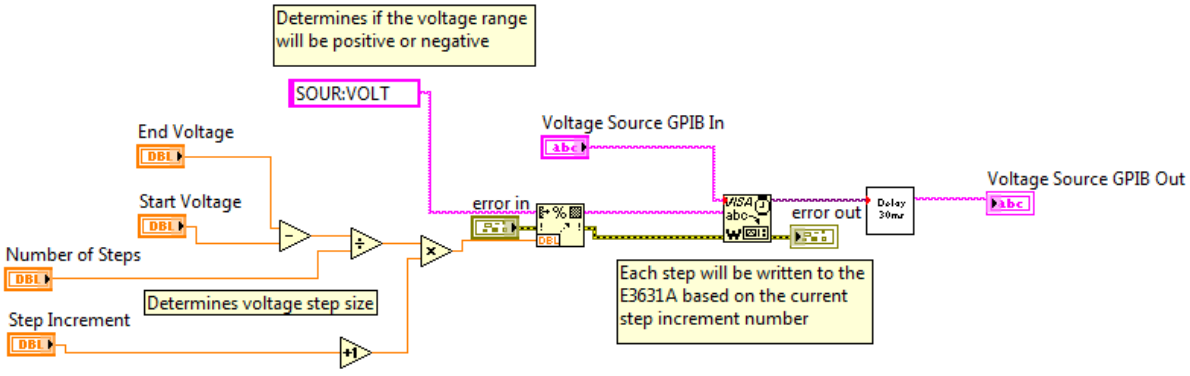

Figure 39: U3606A_Set.vi Wiring Diagram

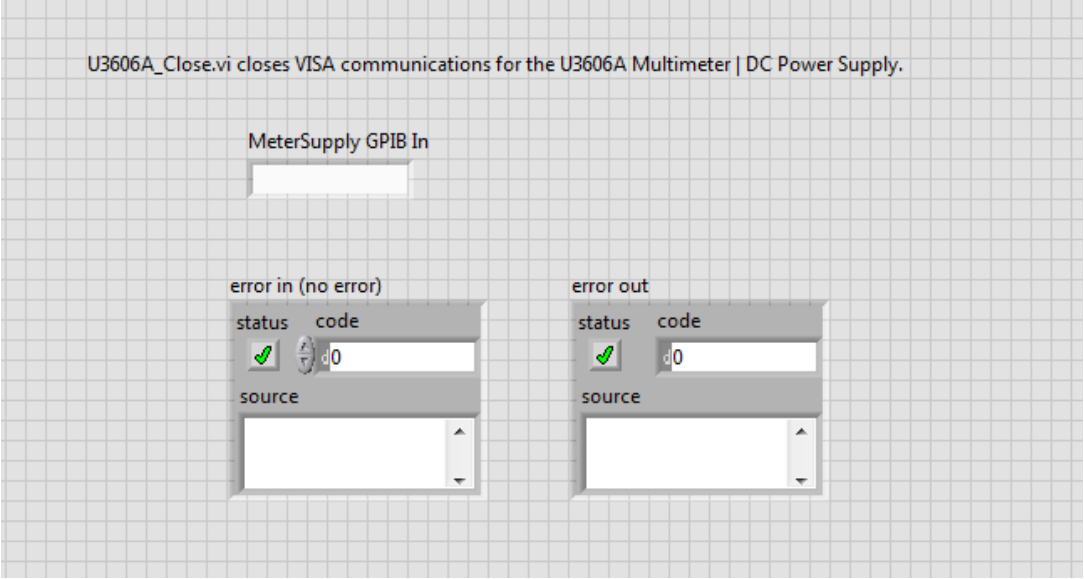

Figure 40: U3606A_Close.vi Front Panel

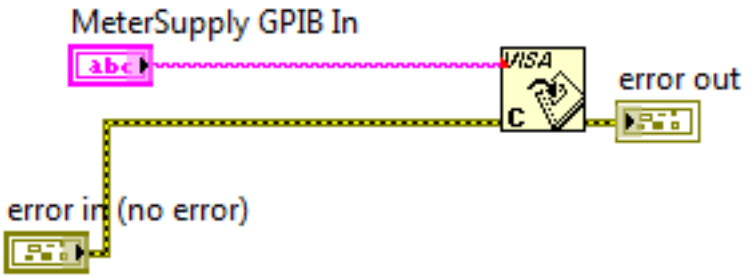

Figure 41: U3606A_Close.vi Wiring Diagram

Current Adapter
Instruments in Room 20-126 do not record measurements below the 100uA scale, so the current adapter should make the instrument into think the circuit has a larger current. The software program can also calculate the real current. The software can calculate these adjustments to output proper values to the user not the front panel, but since the current adapter has not been finalized, the calculation to determine the real current value is unknown.

Figure 42 shows the circuit diagram of the current adapter. Ideally the current adapter will consist of a two stage gain using two op-amps with a total of 10^6 gain. The two stage gain allows for a larger gain, but with smaller resistor values. This will enable the instruments to read currents in the range of 100pA instead of the 100uA range.

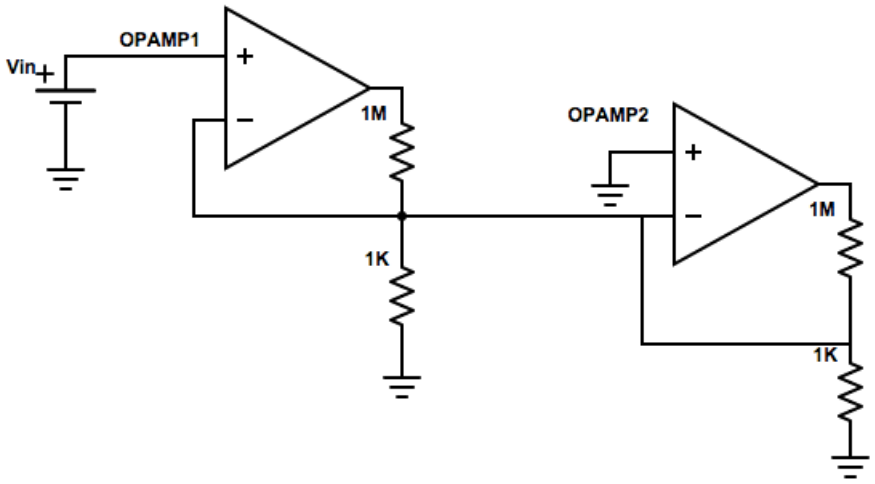

Figure 42: Current Adapter Circuit Diagram

MC33078P op-amps were optimal due to low noise, since small current values could be incorrectly read if the signal contained a large noise spike. While testing this circuit with MC33078P op-amps, a 4V output limit was observed. Although the output voltage should have been larger, the output voltage would clip at 4V regardless if the signal was DC or AC. This could be problematic because the voltage is not guaranteed to stay below 4V. Incorrect data readings would occur.

Conclusion

Although the beginnings of this project appeared to be slow and unhopeful, the final project accomplishes many of the goals for this project. Creating a automation program that can control the existing test equipment through VISA communication allows for a greater usage as well as provides students with the ability to gain experience with various test equipment. This new experience will allow students to learn how to adapt to new challenges.

LabVIEW allows for diode and transistor tests to take place in Room 20-126 due to the automation of the test equipment already installed in the room. Also, LabVIEW has already been installed in all of the computers in the room, so implementation of the diode and transistor tests should be conducted with ease. To apply these diode and transistor tests, the programs will need to be downloaded onto the computer, but no other additional connections or downloads are necessary.

Each new iteration in the software program improves compared to the previous version. Continuous changes has helped this project to move forward and achieve its goal. This project shows the progression from a potential Python automation program to a user friendly LabVIEW software program. Even within the LabVIEW iterations, progress increases with more in-depth software solutions and easier to use user interfaces.

The total cost for the current adapter thus far totals to less than $10. The breadboard costs $6, MC33078P op-amps $0.88 each, and resistors $0.10 each. The total for 1 breadboard, 2 op-amps, and 4 resistors calculates to $8.16. Although the current adapter has not reached the desired goal, the multiple stage gain circuit is applicable to future iterations. Due to time constraints, the software program took priority over the current adapter. For future iterations, changing the op-amp used could benefit the circuit and allow for larger output values.

References

[1] For Top Programming Languages Picture
http://spectrum.ieee.org/computing/software/top-10-programming-languages

IEEE is a well known and respected organization for Electrical Engineers. This webpage is of significance because it shows how Python compares with other popular programming languages used today.

[2] Comparison of Programming Languages
https://www.udemy.com/blog/best-programming-language/

The differences between various programming languages is important when choosing which language to use for the automation scripts.

[3] Keysight Technologies Drivers
http://www.keysight.com/main/editorial.jspx?cc=US&lc=eng&ckey=1441328&id=1441328&cmpid=zzfinddrivers

Keysight Technologies (previously Agilent) is the maker of the test instruments in Room 20-126. The drivers available on its website is trusted to be reliable and compatible with their products. Python scripts are not listed on the webpage.

[4] Cal Poly EE flowchart
http://www.ee.calpoly.edu/academics/

This project is based on Cal Poly's EE curriculum. This allows the observance of the flowchart for Electrical Engineers.

[5] Agilent Cost of Ownership
http://literature.agilent.com/litweb/pdf/5990-6642EN.pdf

Agilent (now Keysight Technologies) is the maker of the test instruments in Room 20-126. It provided an equation to determine the cost for purchasing and maintaining its products.

[6] Java vs Python
http://twistedmatrix.com/users/glyph/rant/python-vs-java.html

This webpage gives multiple examples of the differences between the Java and Python languages. The differences are shown through examples of each.

[7] Cisco Job Opening
http://jobs.cisco.com/job/San-Jose-Software-Development-Engineer-Dynamic-DevOps-Engineering-Team-CA-95101/222274900/

Cisco is a very large and well known company in the engineering field. Observing its valued programming languages is key to learning about the current market for new hire candidates.

[8] LabVIEW Program Prices
http://www.ni.com/labview/buy/

LabVIEW provides various levels of its automation program. Each level of the program provides more features than the previous, which in turn increases the cost. The three levels include Base, Full, and Professional priced at $1,000, $3,000, and $5,000 per year respectively.

Senior Project Analysis

Project Title: Automation for Room 20-126

Student: Stacia Kwong

Advisor's Name: Dr. Dennis Derickson **Initials:** **Date:**

1. *Summary of Functional Requirements*
   This project will incorporate testing equipment in Room 20-126 with Electrical Engineering laboratory classes. Drivers and automation programs will be created to automate the equipment through running a program on the computer. All drivers and automation programs will be written in LabVIEW due to its easy to implement user interface and benefits for future EE students. The automation programs will allow users to control the equipment by setting voltage and current limits. The data gathered can then be compiled into a graph displayed on the front panel.

2. *Primary Constraints*
   The main difficulty with this project will be learning how to effectively program in LabVIEW, since there is no required LabVIEW class in the EE flowchart. However, LabVIEW appears like pictures and has help features like Context Help, which helps the creator and user of the program. The creator can understand the functions of certain blocks within the program and the user can understand the purposes

   Another difficulty may occur when finding the proper commands to call the functions for each of the instruments. If a manual is difficult to locate, time to begin programming will be delayed.

3. *Economic*
   If this project were to arise without existing equipment, the grand total for only equipment would be $110,400. This price includes equipment for 12 benches as is in Room 20-126. The cost for this project also includes time, which is shown in the Gantt Chart in Figure A.

   The cost for the current adapter should be less than $10 to keep student lab kit prices low. Since lab kits already cost an extra $30-50 above tuition prices, ideally keeping the current adapter cost low would benefit students.

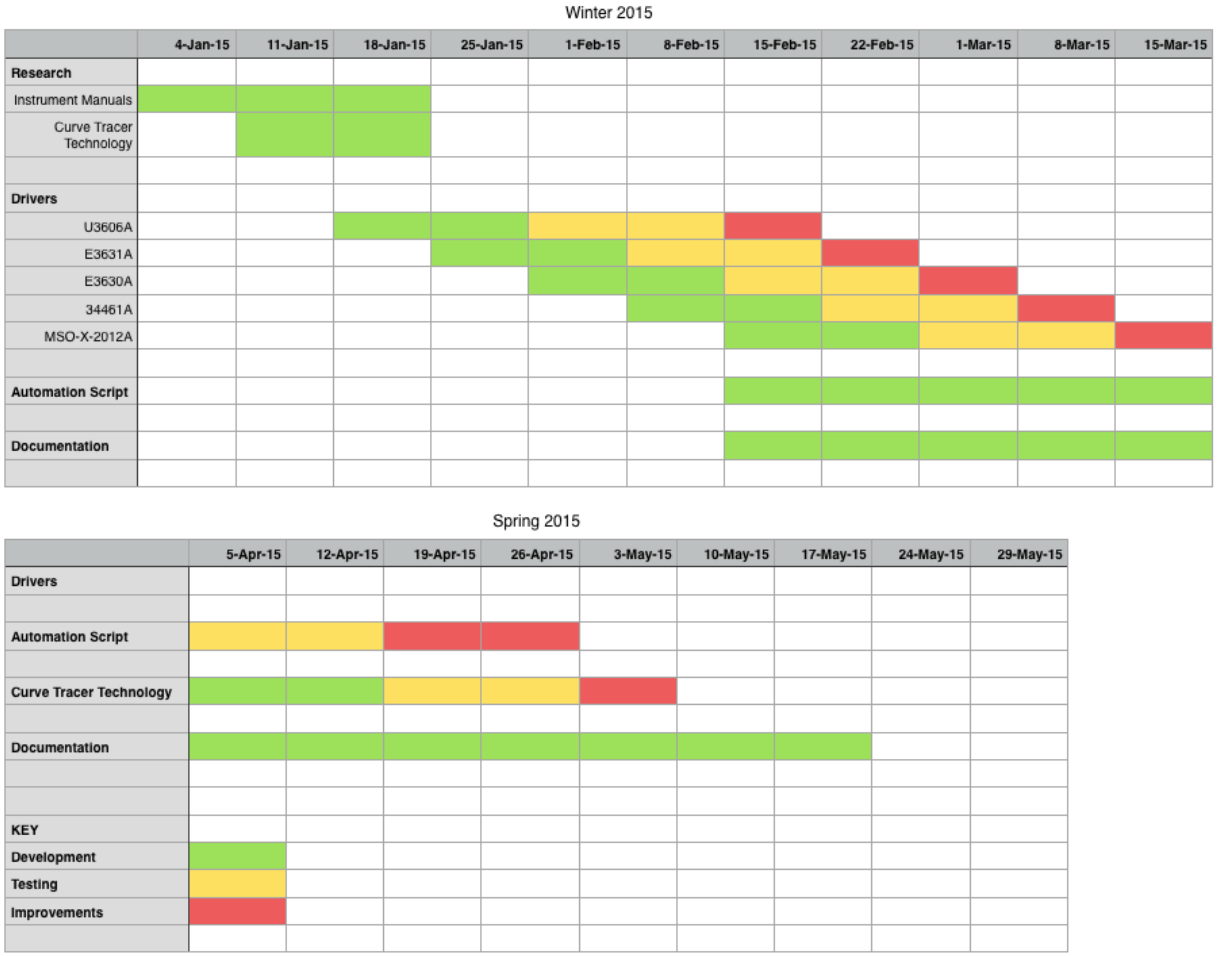

Figure A: Gantt Chart for Winter and Spring 2015 Quarters

4. *If Manufactured on a Commercial Basis*
   Once the automation programs are written, programs can be reused and limits can be changed easily. The bulk of time is prior to use of automation programs and will be easy to alter to specific tasks. Since the GPIB/USB addresses are user inputs, the program controls any instrument with the same capabilities and SCPI commands. However, the program will advise using the specified test instruments to avoid any errors that may occur.

5. *Environmental*
   Since this project is solely with testing equipment and computers, the environmental impact is from electricity to power the equipment.

6. *Manufacturability*
   When the automation programs are created, no manufacturing is needed because the programs can be transferred through USB device or another electronic device. Changes to address may be needed because the equipment on one bench may

have a different address on another, but are defined by the user prior to running the program.

7. *Sustainability*
   The automation programs will be able to be reused, improved, and additional tasks can be added. The programs will only be changed if additional features are added. The drivers will not need to be changed because SCPI commands do not change. Since Cal Poly will be changing from the quarter system to the semester system, labs may be changed, so programs may be changed depending whether the curriculum of the labs are changed.

8. *Ethical*
   IEEE Code of Ethics lists out ten bullets for engineers to follow in the workplace [1]. The following two statements fit this project: (1) to improve the understanding of technology; its appropriate application, and potential consequences (2) to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others.

   The first is important because understanding of how the testing equipment and LabVIEW perform is critical to create an effective driver and program to decrease unnecessary programming blocks. If there are extra programming blocks, the program will take longer to perform, effecting testing time to get data and results.

   The second is important because it is important to have other engineers to give input because they may see potential problems within the program. It is still important for these engineers to follow the first statement, so they can give valuable feedback.

9. *Health and Safety*
   There is very little health and safety risks due to the main portion of this project created on the computer. If voltage or current is too high for the circuit, the circuit can burn out and smoke can occur. However, with voltage and current limits, this problem should be kept at a minimum.

10. *Social and Political*
    Controlling test instruments through automation programs decreases the need for a person to manual test circuits, but increases the need for people to create the automation program. Rather than a tedious job, this job creates the need for more knowledge about LabVIEW as well as problem solving when the program do not perform as expected.

    The main stakeholder for this project is the EE department and staff because this project would affect the teaching curriculum for various EE laboratory classes, starting in the 200-level classes and above. If this project is not completed before Fall 2015, Room 20-126 would not be of use to these students and the students

would not be able to develop their automation/programming and validation skills.

11. *Development*
    The development process will include as much testing time as development of programs. While testing I will be determining the most effective ways to test the programs and to find the worst case for each of the functions. This will better ensure that the drivers will correctly call and perform each function.

Additional References

[1] IEEE Code of Ethics
http://www.ieee.org/about/corporate/governance/p7-8.html

The IEEE Code of Ethics is beneficial to reference due to IEEE's credibility. The Code of Ethics is good for companies and engineers to follow because describes how to professionally conduct themselves in the workplace.