ECE/COE 1896

Senior Design

Controller Board for Integrated Photonic Computing Conceptual Design

Team# 4

Prepared By: Nicholas Nobile
Wyatt Porter
Shreya Wadehra
Grace Henderson

Table of Contents

Table of Figures

Table of Tables

1. Introduction

The Controller Board for Integrated Photonic Computing is an 8 channel voltage supply with 12 bits of precision. The goal of this controller board is to develop a PCB that controls all optical matrix-matrix multiply units to achieve very high bandwidth and ultra-low energy computations. Photonic Computing is a well researched field of study, however despite its popularity, there are limited devices on the market that are low cost and compatible with this specific application.

Currently there are a few 8 channel voltage supplies on the market, however they cost upwards of thousands of dollars and do not guarantee SCPI compatibility, which is a very important feature in lab based testing. Furthermore, many of these 8 channel voltage suppliers have unnecessary functions that are not required or needed for photonic computing applications.

The controller board that will be designed will solve all of these problems. This controller board cost to manufacture will be under $200, will still have 8 channels with 12 bits of precision, and be SCPI compatible. This system will have additional features of voltage sweep functionality to help the lab technician conduct experiments easier. The hope of this design is to allow the photonic computing labs to continue their valued research without wasting money in the budget to buy an off the shelf 8-channel voltage supply with 12 bits of precision that still isn't fully catered toward their application.

The user will be able to interface the system with both a software GUI, hardware GUI, as well as python scripts that will allow the user to set the desired voltage and protection current of the system. If the user interfaces with the hardware set, the information is sent to the microcontroller. If the user interfaces with the python UI or python pyVisa SCPI commands the information will be sent to the microcontroller via the serial interface. The SCPI commands sent to the microcontroller will be processed and converted to the desired output.

The microcontroller will send out information to an FPGA that will act as a memory buffer and timer to ensure that the signals are sent out to the DAC in the correct order and time. The DAC will then convert the signals into an output voltage that has 12 bits of precision. There will also be a current sensing circuit and software loop that will monitor the current in the system and ensure that the current doesn't exceed the protection value.

The following document will describe in depth the background of the device and photonic systems, system requirements and design, and an in depth description of the conceptual design of the controller board.

2. Background

With increased bandwidth density, faster modulation capabilities, and less propagation losses than electronics, the optical domain's influence has increased dramatically across a range of applications in recent years. Integrated photonic circuits have in the last two decades begun to revolutionize many of the fiber optical networks that form the connections between servers across the internet [1] by reducing the footprint of optical MUX/modulation circuits. Other applications of this technology extend into many other fields, from the biomedical sciences [2] to computation [3-6]. While much of this progress can directly be attributed to the physics of light and waveguides, many of the performance enhancements seen in state of the art logic- and/or arithmetic-element derived devices comes more from the usage of optical phase change materials (O-PCMS) acting as fast, low-power memory units. While other technologies and methods for integrated light-based switching exist, the non-volatile nature and large optical contrasts produced by these materials has made them a hot topic in the field. The study and control of these materials in integrated photonic applications requires the ability to precisely heat these materials via joule heating. To perform this heating, many labs and companies must use precise and often custom instrumentation.

1. **Analog Devices DC2025A-A Evaluation Board:** This board utilizes a Linear Technology LTC2668 DAC to produce 16 channels of voltages from 0V to 10V with up to 16-bits of precision. The DAC itself retails for >$45 and the overall board without its additional attachments will sell for $100. At this price point, it seems quite feasible for a research lab to purchase the equipment, but the additional costs for the adapters and code to truly make it function may even double this price. Despite this, the board runs based off of a simplistic SPI based interface; this means it is tied down to a computer or other external controller to operate at all, yet it does not adhere to standard lab-grade computer interfacing schemes (such as the more readable SCPI commands used by VISA instruments which have compatibility with Python and LabView). The current ranges for the outputs do not match the requirements for the project (can only source up to 10mA/channel) and as such would also require additional stages/amplifier circuits to achieve the goal. As the design is minimalistic, there is also no physical or gui-based user interface whatsoever, and functions such as voltage sweeping are not included. Control is open circuit rather than closed circuit and as such the supply current is not a readable parameter nor regulated. The connectors are also simple headers [7-8].
2. **United Electronic Industries DNA-AO-318-020:** In another 16-bit design, this board can independently supply 8 channels with 0-20mA. As a tool meant to be used in research environments, over-voltage and over-current protection electronics are included. It even has self-testing capabilities and has support for data processing applications like Python/LabView/MATLAB. There is some support for higher functionality such as simultaneous updates for multiple channels and 1kB buffers for each channel to perform even custom arbitrary functions at speeds up to 10kHz. However, for this functionality, the price tag is >$1700 and still relies on an external computer to necessarily control the instrument. The outward connectors are also simple headers [9].
3. **Rigol DP831 Programmable DC Power Supply:** As a standalone high-power industry grade power supply, this has many benefits. It is a full VISA instrument and has a user

interface to control each of its channels. Each channel sports not just a current or voltage setting, but also programmable protection circuits for both voltage and current. However, this is the extent of its usefulness. Not all channels can supply 10V, and the price is above $500 when taxes and shipping fees are included. It typically operates at stange precisions (some of which are less than 12-bit) and offers only a total of three channels to BNC-based outputs [10].

4. **Custom Solutions:** Due to the lack of availability of higher end boards at lower price points, many researchers have opted to create custom boards within their labs. Oftentimes these boards may be designed by those with little experience in the matter and so do not focus on ensuring lab grade standards on their tools are met and/or usually only end up creating minimal functionality (i.e. these tools do not have current protections or user interfaces typically).

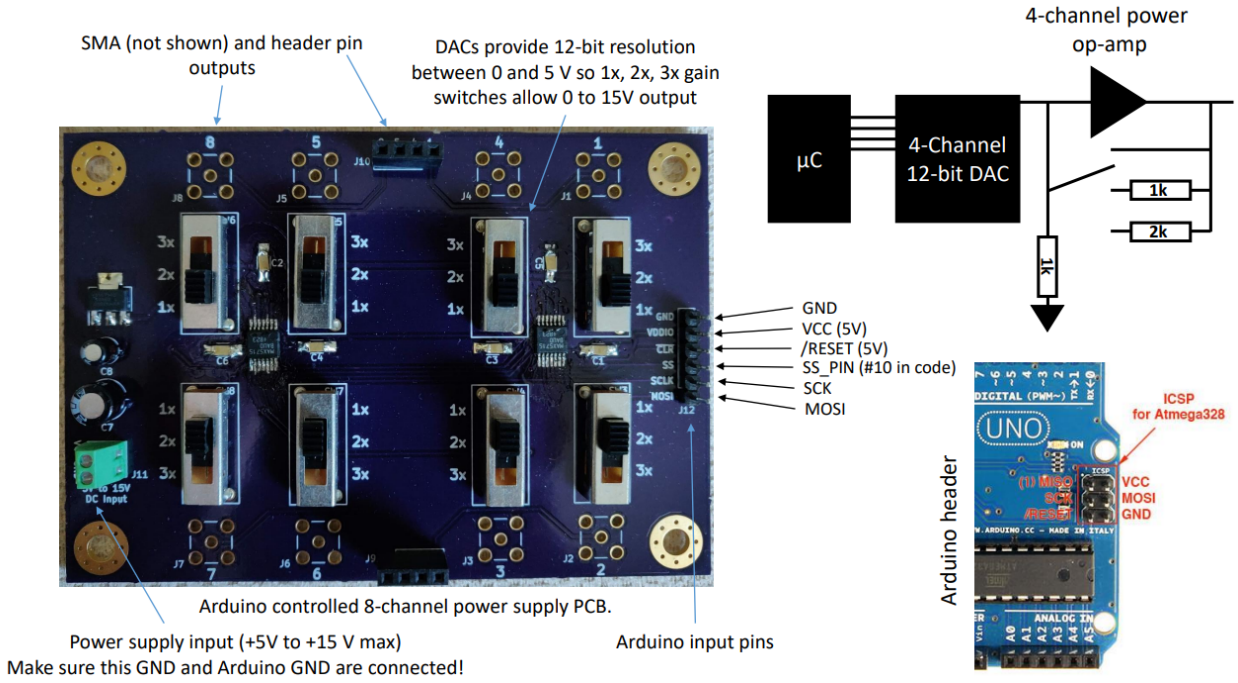

Figure 1: Self-Implemented Controller Board

Our solution to this problem will be one which takes from the strengths of the designs and minimizes the issues. We will use connectors like the SMA-based ones from the custom board to suit the types of wires used in real research labs. We will design DACs that require low bias voltages and input currents and are 12-bit to guarantee precision. We will include at least 8 channels and all of them will be able to source 0-10V and 0-15mA current. We will include interfaces for both people to use and that can communicate with standard softwares like LabView and Python using standard & readable commands. We will offer current/power protection mechanisms for the safety of both the user and Device Under Test (DUT). And we will seek to keep the price within a margin that is allowable for the course and permitted by the domain advisor.

On that note, many DACs (or Digital to Analog Converters) are resistor-ladder and DC amplifier circuits that will transform a binary number into an output scale as determined by a

binary encoded number, the gain of the amplifier, and the voltages of the references, op-amp inputs, and power rails [11-13]. Two common designs are the Binary Weighted Resistor DAC and the R-2R DACshown below. Another critical design component will be the usage of SCPI commands. These commands are unicode-8 ASCII character based commands sent over a serial connection with the following format: ":SYStem:SUBSYStem:SUB-SUBSYStem:ETC VALUE\r\n". Systems and subsystems are typically divided with colons separating them and at the front of the command. A terminating carriage return or line feed is used to signal the end of a command. Letters that are lowercase are optional, but otherwise all required characters should be capitalized. Write operations use a space and then include a value or values separated by commas. A read command will share the same formatting as a write command but will use a question mark ("?") instead of the space and value. The response from the machine will similarly encode values as unicode-8 ASCII and transmit with the same types of termination characters and commas-based separating scheme.

3. System Requirements

3.1 Functional Requirements

3.1.1 Perform Basic Voltage Supply Functions

At the core of the device there will be 8-channels that source between 0V to 10V. Each channel will also source up to 15 mA of current, which amounts to a maximum of 150mW of power per channel. These requirements were provided by the customer directly. Through the user interface described later on, the user should be able to (1) set and update the voltage for each channel individually and (2) read the sourced current and voltage from each channel. If time allows, we also have a stretch goal of incorporating the capability to perform a voltage sweep into the device.

3.1.2 Current Shut-Off

For DUT safety, the device will automatically turn off if the current exceeds the programmable threshold.

3.1.3 Channel Connections

The 8 channels will terminate in an SMA connection, which will allow the user to connect the device to the DUT. This connection was chosen because of its prevalence in Dr. Youngblodd's Photonics Lab.

3.1.4 User Interface

The device is required to be controlled via a USB interface. There will be two ways of controlling the device: (1) through Python commands and (2) through a physical button interface.

If given enough time, we have created a stretch goal of creating a graphical user interface (GUI) to allow for an alternate method of control. The GUI would allow the user to control the device without taking up any extra lab space with a physical interface.

3.2 Performance Requirements

3.2.1 Resolution

The customer requires a minimum of 12 bits of resolution for each of the 8 implemented channels.

3.2.2 Accuracy

The customer requires the device to be monatomic at room temperature with +/-0.5 LSB of accuracy. This will also allow the device to be lab grade.

3.3 Other Requirements

3.3.1 Enclosed in an Insulated Box

For user safety, the device is also required to be housed inside an insulated box.

4. Design Constraints

This section outlines the several different design constraints imposed upon the controller board for integrated photonic computing applications. These constraints include design constraints that limit the design of the semester-long project as well as design if it was being manufactured in the industry.

4.1 Schedule

Schedule is a design constraint in both this semester project as well as a constraint in the industry. However, due to the short length of this class the schedule and amount of time to complete the project will significantly affect the design and implementation of this project. The lack of time will limit the amount of functional features of the photonic controller board as well as the types of components purchased. Due to shipping lead time, only components in stock with a quick turnaround can be purchased in order to arrive in enough time to be soldered and implemented before the project deadline.

4.2 Manpower

Manpower is a common design constraint in both industries as well in this semester project. However, the structure of this class allows only 4 members in the group to design, create, and test the controller board which significantly will affect the design for this semester project. The lack of manpower will limit the extended functionality of the controller board due to the fact we only have 4 members which limits the possible amount of work to be completed, so additional

features may not be able to be implemented. If there were more members in the group the design would be able to have more functionality and capability, for instance more complex voltage sweep algorithms useful for photonic circuit testing could be implemented, but due to the limitations of people this functionality will not be implemented.

4.3 Budget

Budget will affect both industry manufacturing as well as this individual semester project. In the industry, the controller board would need to have a low enough manufacturing cost to enter the market at a competitive price point, which is currently around $2000. However, due to the structure of the class, the individual design for this class must cost less than $200. This will affect the parts purchased for the project and the design of the system to ensure that all components can be bought and assembled within that strict budget.

4.4 Safety Standards

In the case the controller board is sold in the market it needs to be compliant to the International Electrotechnical Commission (IEC) standards. This organization creates and publishes safety standards applying to electronic products. The standards IEC 60101 outline the safety of measurement, control, and laboratory equipment to help protect users and equipment against electrical shock, fire, and mechanical burn injury. These standards would restrict the design to have certain safety protocols in place to ensure the device is safe for user use. Unfortunately, the IEC standards documentation costs upwards of $300 per document. Thus due to earlier constraints of budget, knowing and implementing these specific standards are out of the scope of this specific project and will not be a constraint in the design of our specific project this semester. However, even though these specific standards will not be implemented there will be safety mechanisms implemented in our semester project that will be detailed in the following sections.

4.5 RoHS Compliant

In the case the Controller Board's design is manufactured for general consumer purposes, the device and all its components must be Restriction of Hazardous Substances (RoHS) compliant. The RoHS compliance restricts products that use levels of lead, cadmium and other harmful environmental substances. In order for this controller board to be RoHS compliant every device purchased is required to be RoHS compliant, which will limit the amount and type of devices bought for the design implementation. RoHS compliance is required for European Union countries, which is the primary location of most photonic research . Therefore this constrains our design to only choose RoHS compliant parts.

4.6 EMC Regulatory Compliance

In Industry, there are EMC regulatory compliance documentation that requires the device to pass certain electromagnetic emissions tests. These standards vary country by country. Again, similarly to the IEC 61326 safety standards these standards cost upwards of $1,000, which is out of the scope of being included in the design of this particular design project. However, if this was to be deployed in the industry the controller board would be required to be compliant with different EMC regulations to be sold in different countries.

4.7 Size

Since the controller board is mainly used in lab settings and will be placed on a lab bench top, the design should allow the final product to be small enough that it will not take up too much lab bench space. This constrains our design to ensure that the final product is around the same size as an industry standard benchtop instrument, which is roughly 1 cubic foot. This influenced the PCB design that it has to have surface mount components to ensure that the PCB remains a reasonable size.

4.8 Learning Curve

The learning curve of how to use different components is an important metric to consider when designing any project. The team chose to implement designs with components that were familiar to them, to help minimize the learning curve of how to use each component. Utilizing components that members are already familiar with frees up time to design more in depth implementations. This influenced our design decision to choose the MSP432 microcontroller due to one team member's expert knowledge. This member also had access to a development board, which simplifies design and testing of the controller board.

5. Conceptual Design

5.1 Hardware Concepts

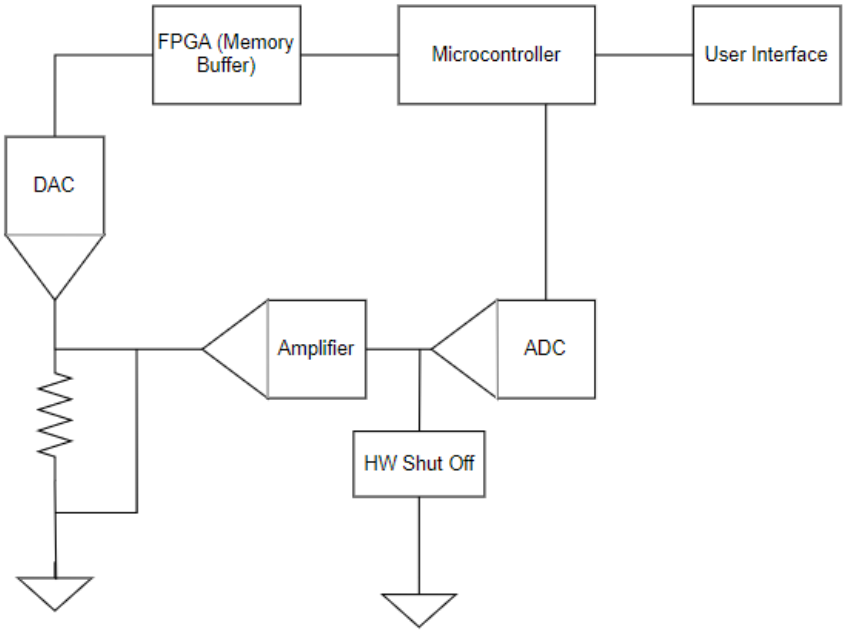

Figure 2: Design Flow Chart

Figure 2 above shows the simplistic block diagram of the overall design. It should be noted that there will be 8 DAC's, 8 Amplifiers, 8 Hardware Shut off circuits, and 8 ADC's. This design just shows one simple output to the device.

5.1.1 DAC Design

5.1.1.1 Binary Weighted Ladder

A Binary weighted ladder is one possible design implementation of a DAC. Figure 3 below shows how the hardware would be implemented in the design.

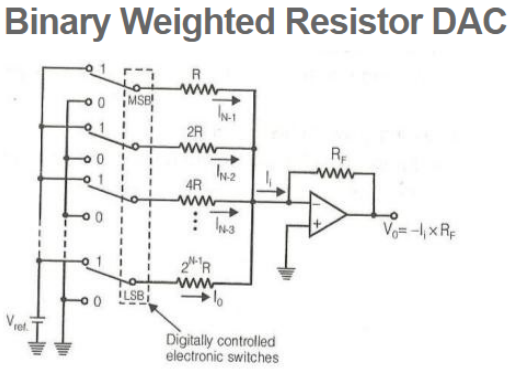

Figure 3: Binary Weighted Resistor DAC

$$V_{out} = \frac{V_{ref}}{2^N * R} * \sum_{i=0}^{N-1} 2^i V_i$$

Equation 1: Voltage out of a binary weighted DAC

The binary weighted ladder DAC relies on high precision resistors for every bit of precision needed for our design. This design would require at most 13 resistors per channel to achieve 12 bits of precision, each resistor will be 2^N-1 larger than the previous resistor which 2^11 would yield incredibly large resistors . However, it is harder to gain a higher accuracy in this design because it is a lot harder to implement trimming to increase accuracy. In order to achieve high precision with this design, this will require purchasing 12 different resistance high precision resistors. This design will be tested alongside the R-2R ladder to decide whether this design gives enough accuracy to overcome it's issues of many independent parts.

5.1.1.2 R-2R Ladder

Another implementation of a DAC is with the R-2R non-inverting ladder DAC. Figure 4 below shows the overall implementation of the design in hardware.

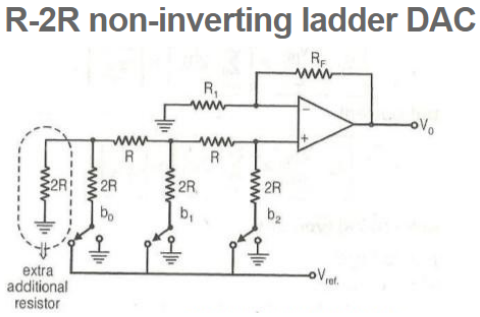

Figure 4: R-2R non-inverting ladder DAC

$$V_{out} = \frac{V_{ref}}{2^N} * \sum_{i=0}^{N-1} 2^i b_i * (1 + \frac{R_F}{R_1})$$

Equation 2: Voltage out of a R-2R non inverting ladder

The R-2R ladder will serve the purpose of setting the voltage to the load to each of the eight channels with minimum four bits of precision and best case scenario twelve bits of precision. This design can accomplish four bits of precision using nine resistors per channel totalling 72 total resistors and an operational amplifier LT1636. In order to achieve a 12 bit precision stretch goal, that would require 26 resistors per channel totalling to 208 resistors. The resistor values for R and R2 will be 10 kiloohms and 20 kiloohms. The R-2R ladder is chosen because it is the most common precision DAC architecture according to Texas Instruments. This can also use a technique called trimming to increase the accuracy of the R-2R ladder which could help us in our application of a precise DC voltage source.

5.1.2 Current Sensing Circuit

Current Detecting Circuits usually rely on the placement of a fixed resistor of known value along the output line and then measuring and amplifying the voltage dropped across it to make it into a signal recognizable by the microcontroller. This serves two functions for the customer: 1) It allows the user to measure the power at any given time and so 2) it can serve as a safety circuit to drive some kind of shut-off so the DUT is not ruined.

Per 12-bit precision with +/- 0.5LSB across a 10V range and 15mA current maximum, the output has the ability to be wrong by a factor of 1.22mV and allows for a maximum shunt resistor value of up to 81.4milliohms to detect current:

$$\frac{10V}{2^{13}} = 1.22mV;\ \frac{1.22mV}{15mA} = 81.4m\Omega = R_{shunt,\,max}$$

The ideal solution would have no or minimal input offset currents/voltages and a gain that allows for the range of values of currents (0-15mA) to take up the range of values allowed on an ADC pin of the microcontroller (0-3.3V) and perhaps offer some low pass filtering (f_{3dB} about 20kHz to still let the device operate at 10kHz speeds; higher is allowable).

$$\frac{3.3V}{15mA} = 220V/A$$

5.1.2.1 Integrated Solutions fitted to our application (ACS712; INA250)

Some integrated chip solutions do exist on the market such as the ACS712 (which is available with testboards in Benedum for no price) and the INA250. Boards like these come designed specifically for the application of current sensing. Both solutions utilize R_{shunt} that is more than a magnitude less than the maximum stated above. Both allow for low-pass filtering circuitry to be added. However, each has drawbacks. Neither supports a gain anywhere close to what is described above and so an amplification stage would need to be added to either to use them in our design. Secondly, it is not particularly clear what offset biases are present in chips such as these; with operating ranges more into the amps and tens of amps, offset currents as big as 5mA may be present, which is ill suited to our application. Figure 5 below shows how an integrated solution is implemented.

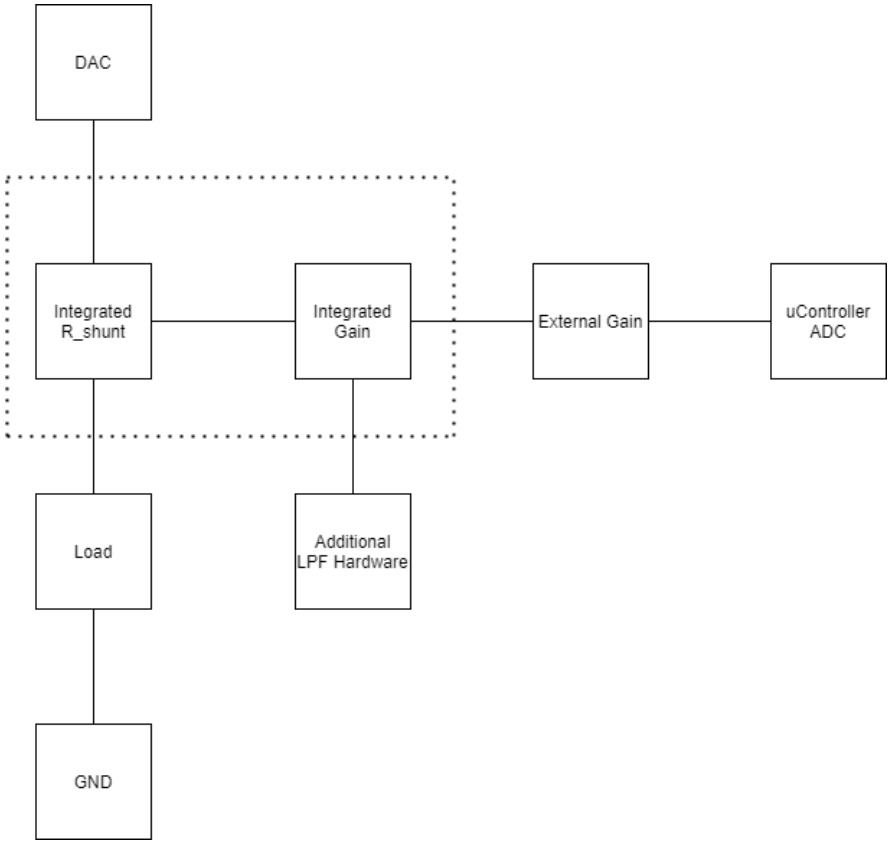

Figure 5: Current Sensing Circuit Flow Chart

5.1.2.2 Customized/Non-specific Assembly applications

A customized application where the group chooses a separate R_{shunt} and op-amp solution is also viable and would likely put to rest any issues created by integrated solutions. A great example of a circuit could be a FC4TR050DERCT-ND 50 mOhm R_{shunt} feeding into an OPA189 style op-amp. With a slightly larger resistor and very low input bias rail-to-rail op-amp able to handle our supply voltages, it can be easy to design a configuration in which low pass filtering and proper gain are all produced in the same step. Another plus of this method is the ability to purchase packages such as the OPA2189 which can also help reduce footprint size or otherwise offer multiple gain/filtering stages. Figure 6 below shows this implementation in detail.

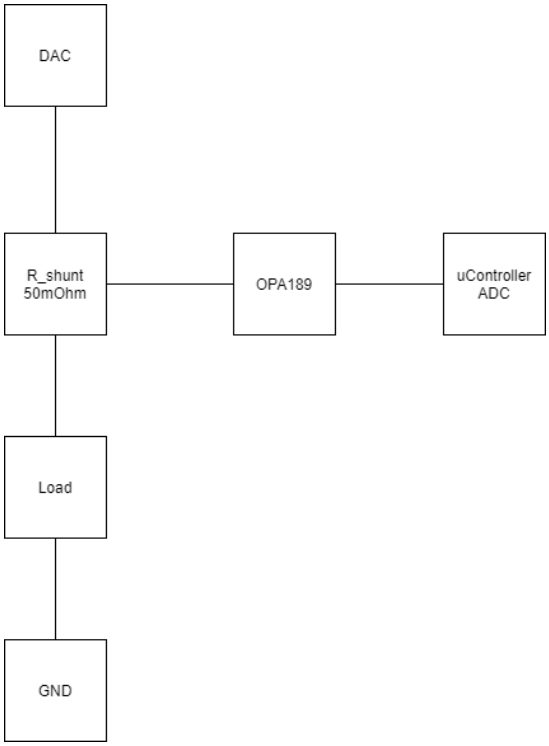

Figure 6: Current Sensing Circuit Flow Chart 2

5.1.3 Device Protection Circuitry/Automatic Shutoff

5.1.3.1 Fixed Hardware Shutoff

A fast response time fixed implementation could utilize taking the analog signal from the current detection amplifier (the same node which otherwise only goes to the ADC in the microcontroller) and comparing it via a comparator circuit to a fixed voltage representing the output at about 15mA. This digital signal could then drive a mosfet put in series with the output of the channel or simply control an integrated enable/shutdown pin on the DAC's op-amp.

An issue with this method is the memory of the design and how that memory otherwise communicates with the microcontroller. Standalone, the device will shut the circuit off, the current will stop, and then the circuit will simply turn on again since the current is within a safe threshold (=0mA) once again. An alternative could be the addition of an RS latch whereby this comparator drives the SET and the microcontroller the RESET. Upon causing an over-current situation, the user would need to either re-enable the channel or input a new voltage whereby the microcontroller could send out the reset signal as per course of a standard enabling/voltage setting procedure.

Although it can be seen that a circuit is designable, and that such a circuit may provide the best reaction time, it also involves the usage of additional hardware when none is technically required. As each channel will already require a fair sum of components, it is preferred that a smaller, more integrated solution be used.

5.1.3.2 Programmable Software Shutoff

There can be a programmable software shut off for the safety of the DUT. Each ADC (total of 8, one for each channel) will have an output that directly maps to an I/O pin of the microcontroller. While the controller board is outputting data, the microcontroller will be polling the I/O pins checking the current value against its desired current threshold as well as max allowable power. In an instance where the current feedback measures a current larger than the desired maximum value the microcontroller will send an interrupt to an I/O pin that will directly turn off the OpAmp in use. This implementation is meant to ensure safety of the DUT as well as equipment safety.

5.1.4 Power Supply

To power the system, the controller board will use a simple 12V wall outlet with a male barrel jack and a female barrel jack on the board to power the positive rail of the opamp. To power the microcontroller, an LD33V will be used to step down the voltage to send out 3.3V. The ICL7662CBA+-ND will be used to invert the input voltage to -12V to power the negative rail of the opamp.

5.1.5 Microcontroller-DAC Interface design

Whether it is an R-2R Ladder or Binary Weighted Resistor Chain implementation, each bit of precision for each channel's DAC requires a dedicated wire offering a digital signal. With 8 channels operating at 12 bits of precision, a total of 96 wires must be in the finalized design just to drive the DACs. And with many of the microcontrollers on this document having fewer than 96 GPIO pins (i.e. even the most advanced MSP432 having only 84), a method of increasing the number of available pins is essential for the project's success. Depending on the implementation of this section, additional system functionality such as maximum speed or processor strain will be decided.

5.1.5.1 Serial to Parallel Shift Registers

The most simple and economical solution to the problem is the usage of shift registers. Given a simple clock and data line, many bits can be represented while maintaining pins on the microcontroller. An element such as the HEF4894B-Q100 offers the ability to produce 12 parallel bits in such a way and performs this task under $2, making it economically feasible to have one for each channel. Driving these circuits would be relatively simplistic for a microcontroller - they would be the only required element between the DAC and microcontroller. With setting and shifting times all under the microsecond range and the ability to operate with clocks at several MHz at a chip level, running the circuit in even the hundreds of kilohertz range is entirely feasible. However, this configuration places a toll on the processor for every update to the output, which makes its strain continuous and makes higher end designs (i.e. voltage sweeping) a truly CPU intensive task to perform.

5.1.5.2 Dedicated Memory and Counter Chips

On Digi-key alone, there are over 50,000 types of memory chips available and over 3,000 counters. Many of these chips easily sell for under a dollar a piece making them seem like an economical solution. But, many of the memory chips are traditional in the sense that they hold 8-bit words and operate almost entirely using serialized communications for the exchange of information and addressing, and in either case this makes them ill-suited to our application which requires at least 12-bit parallel output to drive the DACs. However, memory chips such as the CY7C1041BN-15VXC do exist and can, with the help of a dedicated counter chip like 74VHC4020FT, perform at least as fast as the shift register implementation. With a combined price under $3, this is also an option which may be able to be implemented per channel. With a dedicated counter module and memory unit, it is possible to drive not just DC signals, but also voltage sweeps and even arbitrary function generation without the need of continuous processor strain. In theory, the microcontroller could program each memory point and then simply allocate a timer submodule to act as the clock driving the counter which then plays though each memory point.

However, this implementation still has numerous drawbacks. The complexity to program the DAC is now vastly increased, and may even not address the original issue: as the memory operates with parallelized I/O, the processor would need at least as many GPIO pins per channel as it would connect to the DACs directly. Furthermore, due to the rarity of higher bit-word parallel-output capable memory, it is clear that most timer modules do not line up to provide full usefulness to the chip (i.e. even the 74VHC4020FT is a 14 bit timer which would not utilize the full functionality of a rare 18-bit addressable memory such as the CY7C1041BN-15VXC). In the most realistic implementation, the only way the microcontroller could control the addressing of the memory is through the counter to reduce pin usage, which would be a rather troublesome setup indeed. In addition, with so many addressing wires, I/O wires, clock wires, and two chips; this circuit would produce an incredible footprint

5.1.5.3 FPGA/mPGA

An FPGA/mPGA could offer the best solution if implementation of higher functionality is required. With a hard to solder footprint and prices for MAC 10 series FPGAs starting at almost $4 each (10M02DCV36C8G sells on Digi-Key for $3.78), it is possible this solution is outside the expertise/timing and budgetary constraints imposed upon a senior design project. However, it otherwise offers benefits over the other two design options. Unlike the shift register, it would be easy to create memories and counters to provide the higher functionality, and unlike the discrete solution posed above, these elements can be custom fit so that serialized access to the memory from the primary microcontroller can be achieved, and the counter and memory are well aligned functionally. It also has the added benefit of being a single chip and imposing custom pin interfacing which otherwise reduces the required nets/wires that need to be on the final PCB.

5.2 Software Design

5.2.1 Compiler Design

The function of the compiler is to accept user input or SCPI commands from a python script using PyVisa, and be able to interpret and send the commands to the rest of the system in the order they were received.

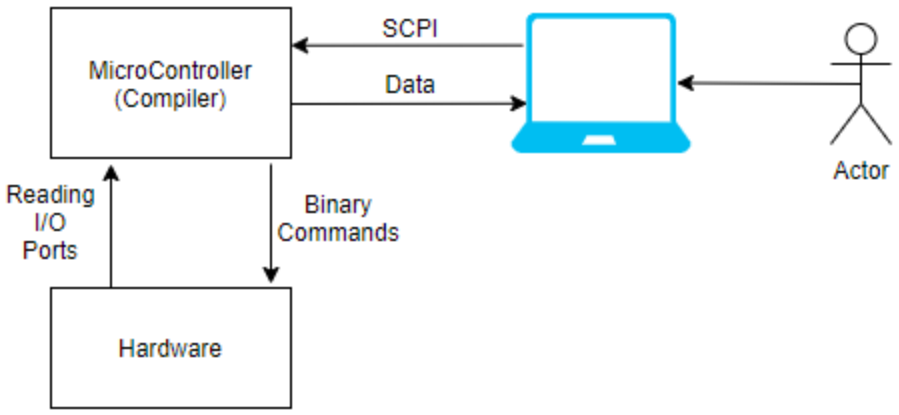

Figure 7: Compiler Flow Chart

Below is the standard SCPI Commands that the compiler will accept. It should be noted that it accepts the commands on the left (short version) and the commands on the right (long version).This design is under the assumption pyVisa will send out an ASCII string per each SCPI instruction. This includes the start of all SCPI commands with a ':' as the start character and '/r/n' as the termination character.

1. Set Voltage:
    a. :CHANX:VOLT XXX / CHANnelX:VOLTage XXX
2. Set Cur Protection
    a. :CHANX:CUR:PROT XXX / CHANnelX:CURrent:PROTection XXX
3. Set Voltage Sweep:
    a. :CHANX:SWE XXX,XXX /CHANnelX:SWEep XXX,XXX
4. Measure:
    a. :CHANX:VOLT? / CHANnelX:VOLTage?
    b. :CHANX:CUR? / CHANnelX:CURrent?
5. Output:
    a. CHANX:Out:START / CHANnelX:OUTPut:START
    b. CHANX:Out:STOP / CHANnelX:OUTPut:STOP
6. Reset / Clear:
    a. *RST - will set the system to the factory reset.
    b. *CLS - will clear status registers

To physically implement the compiler there are two separate design concepts that will be explored in detail in the next two sections, each having their own unique advantages and disadvantages.

Microcontroller: MSP432P401M

This design is based on using the MSP432P401M microcontroller that runs on an Arm 32-bit CPU with floating point capability. This microcontroller has great flexibility with I/O support with 48 interrupt pins and 24 IO port mapping pins. Furthermore the MSP432 has more than adequate amount of memory.

5.2.1.1 Compiler Design Concept 1

The overall interaction diagram of the first compiler design is shown below in Figure 8.

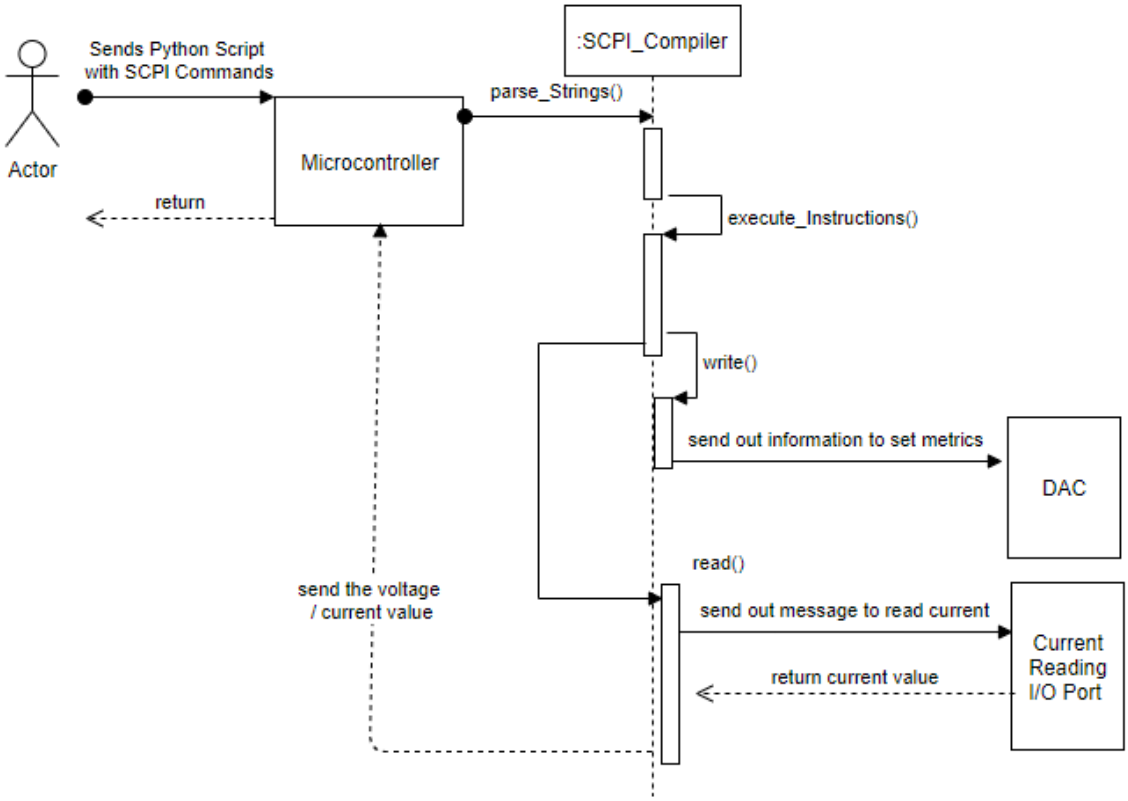

Figure 8: Compiler Design Concept 1 Interaction Diagram

The first step in the interaction diagram is the user runs a python script that sends SCPI commands to the microcontroller over a serial interface. Once the Microcontroller receives the ascii strings it starts running the SCPI Compiler class. Here the first step in the process is to run parse_String(). This function takes the SCPI ascii string and parses through it, and creates an instruction node that correlates to a specific function of that instruction. Each Instruction node is then inserted into the syntax tree. For this specific implementation the syntax tree will be a linked list. Once all instruction nodes are inserted into the linked list, the SCPI_Compiler will call execute_Instructions(). This function will traverse through the linked list of instruction nodes. At each instruction node it will execute said function. If the instruction node is a set/reset/output function it will call the write command and write to the correct I/O port the 12 bits of data and 10 bits of address bits to the FPGA. If the instruction node is a read operation it

will send out a message and access the read I/O port and then send the information all the way back to the user. Figure 8 below is the class diagram of the implementation above.

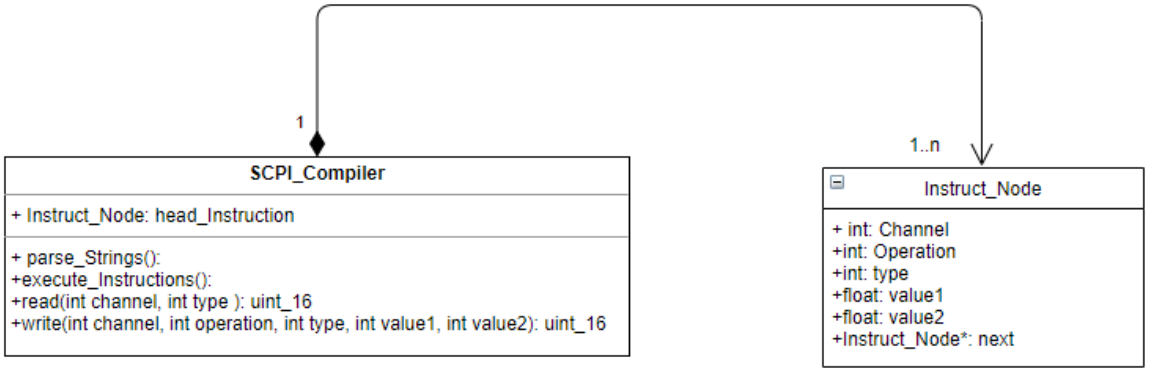

Figure 9: Class Diagram

The SCPI_Compiler has a head_node that holds the head of the instruction node linked list. The instruction node has 6 data types that are explained in more detail below in table 1.

Name	**Type**	**Values**	**Explanation**
Channel	int	1-8	(each number corresponding to each output channel)
Operation	int	0-4	0 - this signifies this instruction is a * Command (reset / clear) 1 - this signifies this instruction is a measure? query command 2 - this signifies this instruction is a Set command of either setting the voltage value or the current protection limit 3 - Set Voltage Sweep 4 - Out (turning channel on and off)
type	int	0-1	0 - This signifies this is a voltage (in case of operation =0, 0 means rst) 1 - this signifies if it is a current (in case of operation =0, 1 means clear) 2 - Stop Flag 3 - Start Flag
value 1	Float	0-max	This is the setting value of voltage 1 or the protected current limit
value 2	Float	0-max	This is setting the value of the voltage sweep max value.

Table 1: SCPI_Compiler Table

5.2.2 Compiler Design Concept 2

The second compiler design concept will utilize the same SCPI format described in section 5.2, however this design will use a more conventional compiler design and utilize a different syntax tree data structure. Instead of design concept 1 that only had 2 stages (parse_string() and execute_Instructions()), this compiler design will utilize 3 distinctive stages of compiling the SCPI commands. Figure 10 below details the functional diagram of this implementation.

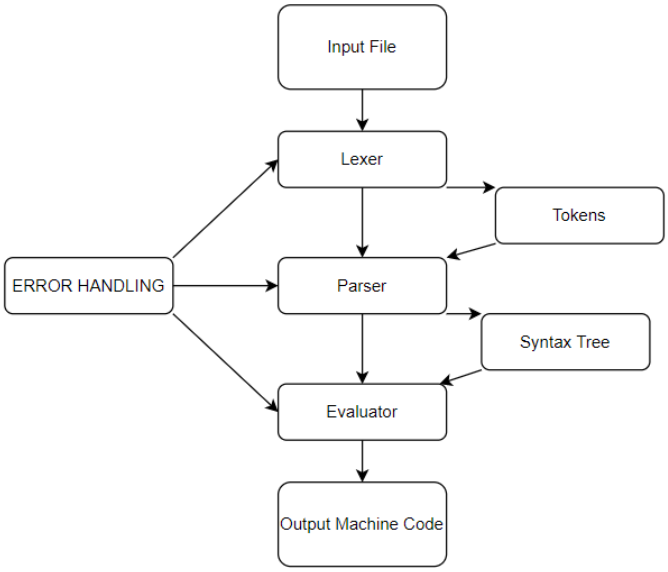

Figure 10: Second Design Flow Chart

The ascii SCPI command will enter the Lexer which will simply take every instruction and parse through the instruction and convert each individual command into tokens and put them into a linked list. The addition of a linked list that holds token values from the lexer stage will add slight memory and operational overhead to this implementation. Once lexical analysis is done the parser will take the tokens and put them into a syntax tree shown in Figure 11 below.

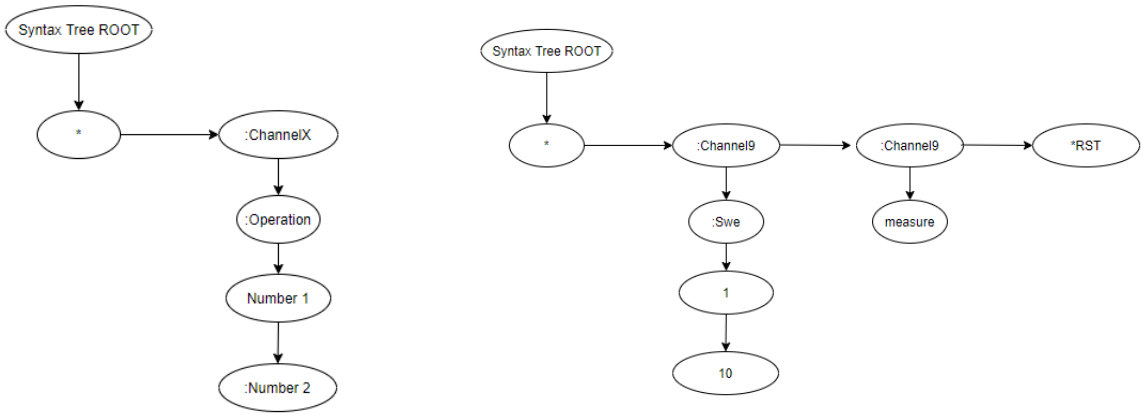

Figure 11: Syntax Tree

In this design, the syntax tree is a linked list of linked lists, which is different from a simple linked list of nodes. The advantages of having a tree-like data structure is that it is much more flexible in adding more SCPI commands in the future. Furthermore these tokens have less memory overhead due to less metadata stored in each node.

After the syntax tree is created it will go through the execution process that will simply traverse the syntax-tree and execute the commands. The Interaction and class diagrams of how the entire compilation process is shown below, which are very similar to design concept 1. The differences in the interaction diagram are shown in red in Figure

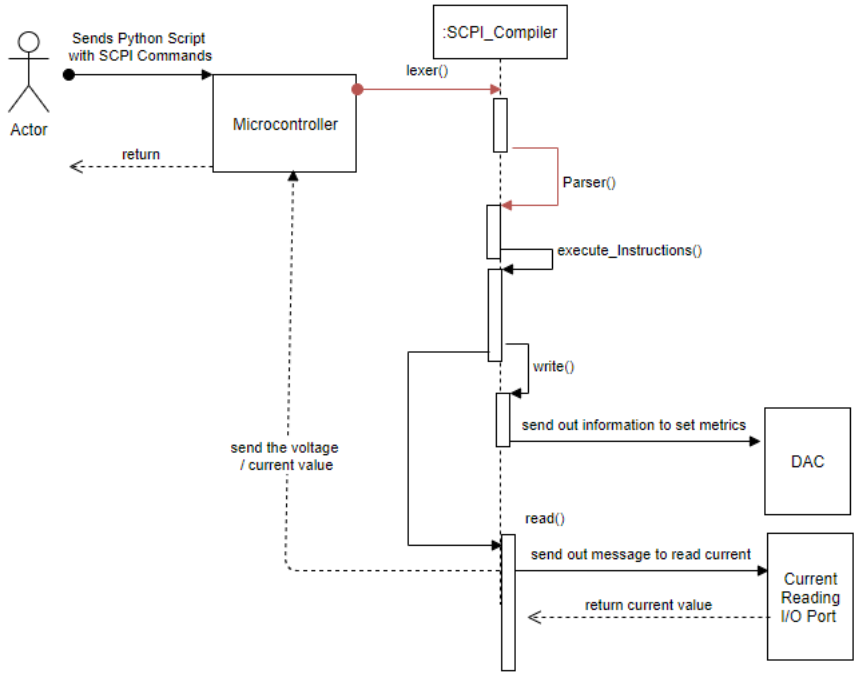

Figure 12: Interaction Diagram of Compiler Concept 2

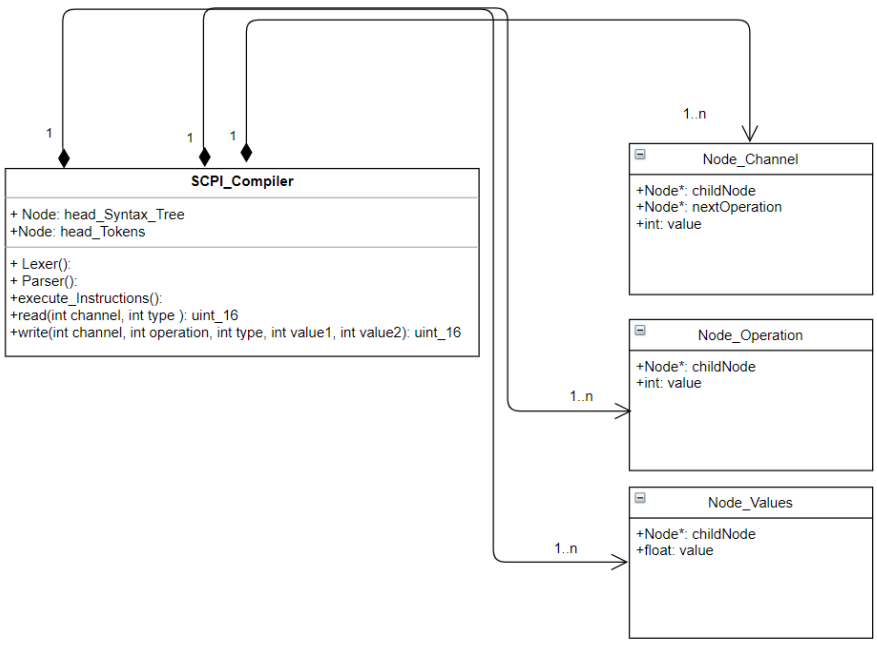

Figure 13: Class Diagram for Compiler Design 2

Again, design 2 has slightly different lexical and parsing techniques. Design 2 adds more flexibility with a larger SCPI command library, however will add more run time overhead and overhead of holding onto a linked list of tokens before inputting them into a syntax tree.

5.3 UI Design

5.3.1 3D Printed Box

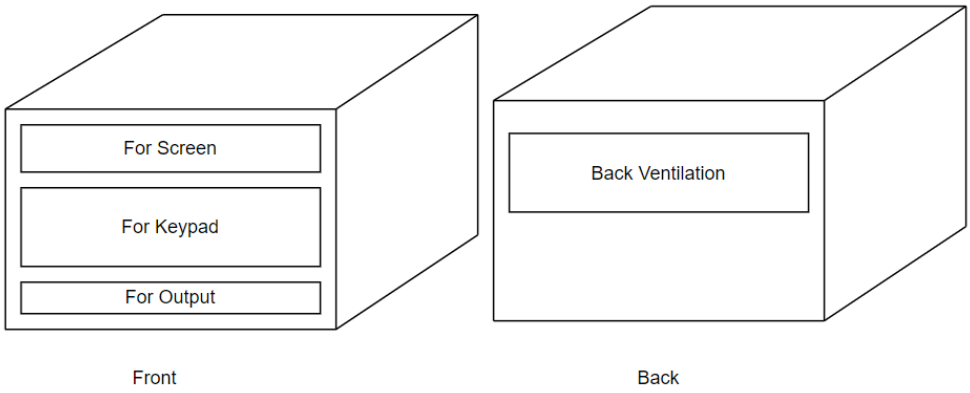

Figure 14: 3D Printed Box

The box that will be used as the enclosure will have a front area that allows the inputs and outputs and UI Display, and will have a back hole for ventilation. Note that this box will be compatible if the UI is chosen to be hardware, if it is software there will only be a space for the outputs to be listed and a switch to turn it on or off. The exact dimensions of

the 3D box will not be known until the PCB size is determined, which will be done later on in the semester.

5.3.2 User Interface Design Concept - Hardware

For the Design of a Hardware Interface, only 3 major components would need to be used: A group of 6 directional/selecting/backing out keys, a number pad, and a simple display at least capable of producing text. The selecting keys can use generic buttons, standard GPIO pins on the microcontroller, and pull-up/pull-down network resistors as needed. The number keys can use a readily available keypad with reading software similar to that developed for the corresponding lab in ECE0202. The display would likely utilize something like an Adafruit 1743 or even the simplisticNHD-0216K1Z-FL-YBW. The interface would allow a user to move through a set of menus that are organized in the same format as the SCPI commands, and ultimately would operate much like them, but allowing a user to use the machine without an external computer. Depending on the type of screen employed, a graphical representation of the data may be shown versus time.

5.3.3 User Interface Design Concept - Software

Below is the design of the Python UI, which will have the same function as a hardware UI.

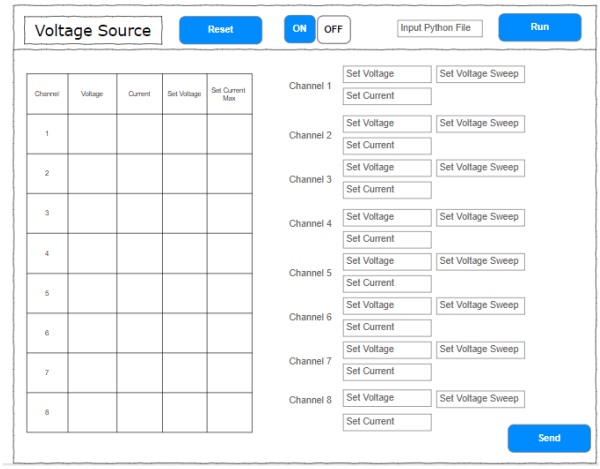

Figure 15: User Interface Software

5.4 Selected Design Concept

Current Sensing Circuit → Custom & Discrete Current Sensing Circuit:

As per the descriptions above, it seems clear that, while there are integrated solutions that exist, most of them are geared for power electronics and current ranges far greater than those expected in this application (1A ranges minimum) such that, to adapt them to our circuit, it would take about the same amount of electronics as a custom solution would. That said, integrated solutions essentially fall short in this regard, taking up more space and creating complexity where none is

really required. Furthermore, it is more the task of this class to design non-trivial circuits ourselves.

Device Protection → Both:

As both solutions can work together, attempts will be made to incorporate both into the final design, but the software solution will ultimately be the fallback if the hardware solution cannot be crafted due to time constraints. This decision is largely based on the availability of the OPA2189 which, as it houses two op amps within its circuit, can act as both the amplification stage for the current sensing circuit and the comparator needed for this circuit without increasing footprint and complexity substantially. This decision is also based on the price/availability of RS latch circuits, which are considerably cheap or may even be able to be utilized from the FPGA.

DAC Design → R2R Ladder:

The DAC design chosen for the project is the R2R ladder configuration. This DAC is industry standard when designing a system that requires precision. Our 12 bit precision stretch goal would require 26 resistors per channel totalling 208 resistors for 8 channels. The resistor values for R and R2 will be 10 kiloohms and 20 kiloohms. Also with a technique called trimming we can adjust the accuracy even further. It is mainly this reason why we chose an R2R DAC as opposed to the binary weighted DAC.

Microcontroller_DAC Interface → FPGA / MPGA Design:

The FPGA design is chosen as the primary design decision due to its flexibility and ability to interface with serial input and produce parallel output. The main issue with the other designs is the lack of GPIO pins to control the device. Furthermore the FPGA can incorporate both a memory buffer as well as a timer module. This is important when it comes to implementing voltage sweep functionality. Due to the constraints from our design this is the only viable solution that can be implemented.

User Interface → UI Design Software:

The main UI design will be the software interface. Since there is a high budget already, implementing a software UI will cost less money. Furthermore the quality of a hardware UI will be less user friendly due to lack of purchasing high end parts. A software UI will also allow much more flexibility in adding more features given the time constraint of the project.

Compiler Design→ Design 1

Compiler design 1 is the optimal choice. This design merges a typical compiler design of Lexer, Parser, and executor and combines the lexing and parsing in one function and creates a syntax tree using a simple linked list of nodes. Due to the linearity and sequential nature of SCPI commands a linked list is functional and allows for a slightly more simplistic approach with some advantages:

- Lower memory footprint with less code storage due to one less function to write
- Lower memory footprint without having to store a linked list of tokens in memory (design concept 2 had both a linked list of tokens and a syntax tree)
- Faster run time for traversing a linked list over a tree.

However, even though this is the chosen implementation it is important to discuss the drawbacks to this design:

- This design has less flexibility when it comes to adding SCPI commands This design can add different commands; however each new operation may increase the metadata of the instruction node. If too many SCPI commands are implemented, the metadata overhead of each instruction node may outweigh the benefits of removing a linked list of tokens. However that being said, since this specific application is so specific to voltage and current there are not a lot of SCPI commands that are required to be implemented so for the scope of this class this design is sufficient.
- This design does have slightly more memory overhead in the Instruction node vs. the various nodes implemented in design concept 2, however this is not large enough to overcome the memory footprint design concept 2 added with a linked list of tokens.

6. System Test and Verification

The following section details the key measurable performance criteria for the controller board as well as the test cases used to test each individual component as well as entire system testing.

6.1 Performance Criteria

6.1.1 Precision of the output voltage for each channel based on specified output voltage.

We expect the output voltage per channel to have a resolution of minimum 12 bits.

6.1.2 Current Shut-Off

The current in the device should never exceed the protection threshold (a programmable threshold). If the current does exceed this threshold, the device will completely shut off for the safety of the user. This should be initiated by both a hardware shut-off and a software shut-off.

6.1.3 Voltage noise and DC offset is lower than +/- 0.5 LSB

To be lab grade, the device must have an accuracy of +/- 0.5 LSB.

6.2 Software Systems

6.2.1 Unit Testing of Microcontroller Compiler

- Unit Testing of parse_Strings():
    - Ensures all allowed grammar is accepted by the parser and parsed correctly. This unit tests all possible SCPI commands entered without throwing any errors.
    - Ensures any grammar that is not defined in the system throws an error that is handled correctly
    - Test the order of the input string is handled correctly. If the order of SCPI commands is in the wrong order will throw proper error
- Unit Testing of execute_Instructions():

    - Ensures that all instructions that are input are executed in the correct order and execute the correct order.
- Unit Testing of write():
    - Ensures that all the write commands send the correct signal output to the FPGA.
- Unit Testing of Read():
    - Ensures that the value that is read from the device is precise to the value read by voltmeter.

6.2.2 Unit Testing of the Python GUI

- Ensure that all buttons send proper signals to microcontroller
- Ensure that all currents read are the correct values
- Ensure proper communication between the computer and the controller board

6.2.3 Unit Testing of the software current controller

- Ensure that if current is over current protection value it turns off
- Ensure that when current is below protection threshold the channel is turned on
- Ensure that the time it takes to receive current over protection valur and shuts off channel in an adequate amount of time (i.e. not 5 seconds to turn off channel)
- Ensure that if no current protection is set it automatically defaults to the maximum allowable current 15 mA.

6.2.4 FPGA Unit Testing

- Testing the functionality accepting serial input of data, storing the data, and outputting the data in parallel
- Testing proper functionality of the counter controlling memory input and out

6.3 Hardware Systems

6.3.1 Current Detection Unit Tests

- Check if the system stays on for multiple increments of an amount of current lower than the threshold current
- Check if the system stays on for one increment of the current resolution below the threshold current
- Check if the system turns off with a current above one increment of the resolution of the threshold current
- Check if the system turns off at multiple increments of an amount of current higher than the threshold current
- While checking if the system turns off, check to make sure no current is flowing through the device

6.3.2 DAC Unit Tests

- Set at least 20 digital values and use an oscilloscope to ensure the proper analog voltage is created (use values that set at least each of the bits once)

- Time permitting, automate either random testing or complete testing to generate all possible input values and compare them to the output analog value

6.3.3 Hardware UI Unit Tests

- Check to ensure each button is registered when pressed
- Set 20 digital values and use an oscilloscope to ensure the proper voltage is outputted on the respective channel
- Repeat for each channel

6.4 Entire System Tests

After all the unit tests are completed the overall system requires testing to ensure that the entire system works and is able to meet all requirements.

- The first test will be to measure the output voltage of the system with a voltmeter with at least 12 bit precision and compare that to the controller board's desired output voltage. This will be conducted on the range of possible voltages.
- This test will be done by creating voltage sweeps and setting different current protection limits and monitoring that the current never exceeds the protected current value.
- Voltage noise will be looked at through an oscilloscope to ensure that the signal integrity is intact and lower than +/- 0.5 LSB.
- Finally, current shut off will be tested, using a test similar to the one listed in the current shut off unit test.

7. Team

7.1 Wyatt Porter

Wyatt is an Electrical Engineering major pursuing an autonomous systems concentration. Wyatt has experience in creating control circuits as well as some experience in machine learning. Wyatt will be responsible for designing the precision DAC for the project as well as the power supply. He has experience in soldering and AutoCad softwares such as Eagle and Solidworks.

7.1.1 Skills learned in ECE coursework

Wyatt had taken ECE 1212 that goes over the design and testing of DACs. Also he learned basic circuit knowledge in ECE 101. He also learned the properties of operational amplifiers in ECE 102. This will give him an in-depth grasp on a precision DAC design. Furthermore, classes such as ECE 1673 may prove useful for controlling aspects of the project such as the hardware shutoff.

7.1.2 Skills learned outside ECE coursework

Although ECE 1212 teaches about DACs, it is mainly focused on the ideal DAC which won't be completely useful in the project. Wyatt has found a video series from Texas Instruments that explains how to tune DACs in the real world. This knowledge will be

paramount to creating the precision needed for the project. Wyatt will also have to learn how to create a power supply for the circuit, this will be done by pooling knowledge with Nic who has some experience in this area.

7.2 Grace Henderson

Grace is a Computer Engineering major and has a high level of comfort in writing and designing code and has two summers of industry experience in soldering and modifying CEVB's and running test scripts on microcontrollers. Grace will be responsible for designing, implementing, and testing the SCPI Compiler in Microcontroller which includes communication with python pyVisa library and the microcontroller and converting ASCII strings into signals that can be read by the DAC.

7.2.1 Skills learned in ECE / COE Coursework

Grace has taken classes in ECE 1175 Embedded Systems Design and CS1550 Operating Systems that go over concepts on how to handle I/O devices and handle interrupts. Courses in ECE 1140 Systems and Software Engineering and CS 449 Introduction to Systems Software has given Grace a high level of understanding of Software design and C language skills. Furthermore, classes in CS1501 Algorithms and CS 445 Data Structures give a good foundation and understanding of how to organize code and implement data structures like syntax trees and linked lists.

7.2.2 Skills learned outside ECE/ COE Coursework

The physical implementation and compiler designs is not something Grace has prior experience with, and will require more information and study on different compiler designs and more importantly different ways to implement: Lexer, Parser, and Executor. Furthermore, Grace has little experience coding on microcontrollers so learning how the specific microcontroller that was picked will also take time to research and understand how it operates.

7.3 Nic Nobile

Nic will be responsible for the current detecting circuit design and testing, the current shutoff mechanism/code, whatever hardware user interface gets implemented, and purchasing. Nic is an Electrical Engineering Major with a background in integrated photonics research, ULSI Rocket Payload Electronics design and fabrication, and microcontroller-centered hardware and software designs. Nic also has about a year's experience in industrial documentation, purchasing, CAD design, data analytics, and experimentation design from Powerex, Inc. He has also recently learned to perform stainless steel TIG welding.

7.3.1 Skills learned in ECE coursework

Nic has taken ECE0202 - Embedded Processors and Interfacing as well as ECE1188 - Cyber-Physical Systems and so is well versed in microcontroller GPIO/memory mapped I/O, C++, hardware timers, and discrete time PID control systems. Having completed ECE0101 - Linear Circuits and Systems, ECE0102 - Microelectronic Circuits, and ECE1212 - Electronic Circuit Design Lab, he is capable of developing and testing amplifier and control circuitry. Nic

has also learned PCB fabrication and testing techniques through ECE1895 - Junior Design and ECE1232 - Intro to Lasers & Optical Electronics.

7.3.2 Skills learned outside ECE coursework

Nic has spent time in an integrated photonics research lab which has given him a background using lab-based equipment, SCPI command-oriented programming, and O-PCM joule heating. As the ultimate primary user of the design project and the one closest to the domain advisor, he will be able to provide much of the domain-specific knowledge needed to understand the task and will act as the primary intermediary with the domain advisor. Through his work with Pitt SOAR, he has also developed 3D printing experience and further refined his PCB design, fabrication, and testing techniques.

Nic has little experience with designing hardware-based feedback mechanisms and unobtrusive current detectors. To further refine these skills, Nic will seek help in online tutorials, consult prior textbooks (such as Sedra/Smith Microelectronic Circuits Seventh Edition), and consult professors here at Pitt that are the current faculty for the most applicable classes. Consulting the faculty in charge of Senior Design and his advisor, Dr. Kerestes, Nic will be able to identify who those people are. And while Nic has had training in SERC for surface mount soldering, 3D printing, and in-house PCB printing, he will consult Bill Mcgahey and Jim Lyle for particulars in these fields as needed.

7.4 Shreya Wadehra

Shreya is an Electrical Engineering major with minors in Computer Science and Physics. The EE side will help with hardware, the CS side with software, and the Physics side with understanding the photonic application of the project. Shreya also has project and lab experience with PCBs and layout, internship and lab experience in digital design and soldering, and mechanical design experience through the makerspaces. Shreya will be responsible for interfacing between the hardware and software components of the project, the FPGA design and implementation, and helping with microcontroller work and analog design of the DAC.

7.4.1 Skills learned in ECE coursework

On the embedded systems side, Shreya has taken and TAed for ECE201 and ECE202, and is currently in ECE1175 (Embedded Design). On the digital design side, Shreya has taken ECE101, ECE102, and ECE1212. She has also been exposed to PCB design and layout through ECE1895 (Junior Design). Shreya has also taken CS445 (Data Structures), CS447 (Computer Organization), CS449 (C programming), and CS1501 (Algorithms), which will all help with the embedded and software side of the project.

7.4.2 Skills learned outside ECE coursework

Shreya has built Python interfaces before and worked with parsing String data (similar to the compiler design). She also has experience using 3D printing and laser cutting to build enclosures through her work as a makerspace mentor. Shreya will spend some time learning how to

interface between a microcontroller and an FPGA. While she has used both of them independently, she is new to using multiple FPGAs and a microcontroller together. Shreya has also never programmed an FPGA without a development board, so she will take some time learning how to set that up. Shreya will also learn about current sensing and shut off circuitry to aid with the design for those parts of the project.

8. Schedule and Budget Plan

8.1 Project Schedule

Week 9/27:

- Grace: Have basic SCPI commands compiled and all tokens being created.
- Nic: Finalize Current Detecting Circuitry Schematic and build a breadboard model for it and power supply circuits. Order 1st round of parts.
- Wyatt: Finalize DAC schematics of an R2R ladder as well as a Binary Weighted ladder.
- Shreya: Create FPGA design

Week 10/4:

- Grace: Have all SCPI commands listed implemented and properly parsed into syntax tree and have implemented proper error handling.
- Nic: Finalize Building Breadboard Model and Troubleshoot its functionality.
- Wyatt: Start the schematics of the power supply for the board.
- Shreya: Finalize FPGA design and make schematic

CHECK OFF 1 EXPECTATIONS:

1. Have a software algorithm that can take SCPI commands and compile them. Show all unit tests proving grammar and syntax errors are handled properly.
2. Have a design done with DAC with either a Breadboard model and or on Spice.
3. Have pseudocode for FPGA design and begin implementation

Week 10/11: CHECK OFF 1 (10/11 or 10/13)

- Grace: Demonstrate the SCPI compiler accepting all SCPI commands and proper error handling and inputting it into a syntax tree and executing commands in proper order.
- Nic: Begin programming Current Detection/Shutoff mechanisms and creating PCB layout for power supply and current detection circuits
- Wyatt: Finalize the DAC design and test on breadboard/Spice program.
- Shreya: Have pseudocode for FPGA design and begin implementation

Week 10/18:

- Grace: Create proper communication with microcontroller and python and begin testing sending inputs and having the microcontroller accept the SCPI ascii string.
- Nic: Finalize PCB layout for power supply & current detection/shutoff by integrating with other pcb components
- Wyatt: Work with Grace to have the DAC handle inputs from the microcontroller

- Shreya: Finish FPGA implementation for one channel

Week 10/25: Midterm Presentation (10/25 or 10/27)

- Grace: Have 1 SCPI command accepted by microcontroller and show it is processing proper output signals to FPGA
- Nic: Demonstrate current Sensing circuitry and power supply
- Wyatt: Demonstrate the DAC working properly on using microcontroller outputs. As well as a working power supply.
- Shreya: Demonstrate FPGA design working for one channel

Week 11/1:

- Grace: Implement rest of SCPI commands to send out proper signals, have I/O communication for all commands.
- Nic: PCB Assembly and aid in enclosure & UI design; Help verify code for reading currents.
- Wyatt: Create an electrically insulated enclosure for the board. Troubleshoot issues with the DAC design. And Power supply.
- Shreya: Have FPGA working for two channels

CHECK OFF 2 EXPECTATIONS:

- Have the python ability to interface with microcontrollers to send out proper signals when SCPI commands are sent. Have proper reading and writing communication.
- PCB implementation of one channel and enclosure design more flushed out.
- FPGA working for 4 channels

Week 11/8: CHECK OFF 2 (11/8 or 11/10)

- Grace: Ensure and debug I/O communication and SCPI compilation. Start working on python interface
- Nic: Troubleshoot PCB implemented current detecting circuitry; Help finalize and fabricate enclosure
- Wyatt: Show the working DAC and power supply assist in fabricating the enclosure.
- Shreya: FPGA working for 4 channels

Week 11/15:

- Grace: Finalize having I/O properly working with the microcontroller and finalize python interface.
- Nic: Continue troubleshooting as necessary and aid other members to make parts functional. Work on Hardware User Interface if possible and as needed.
- Wyatt: Troubleshoot DAC and work closely with Grace to ensure precision.
- Shreya: Finish FPGA implementation for all 8 channels

Week 11/22: Finishing All Design Construction in order to analyze data

- Grace: Begin testing the implementation of design.
- Nic: Perform Tests of functionality on Current Sensing modules and User Interface Elements

- Wyatt: Continue testing the board on its functionality on the DAC, ensuring proper precision.
- Shreya: Test FPGA design

Week 11/29:

- Grace: Gathering and analyzing data of the project. Focus on testing creating tests and having python data sweep and graphs of output data.
- Nic: Numerically analyze data from tests, create graphs, and aid with final presentation preparation (i.e. making posters, video device driving optical circuit). Begin writeup of Final Report.
- Wyatt: Help analyze the data and assist in preparation for the Final Report.
- Shreya: Help analyze data, work on final report

Week 12/6: Final Presentation (12/6 or 12/8)

- Grace: Write up final report, ensure that software on microcontroller functions properly.
- Nic: Write up similar sections on the final report as a preliminary one. Pray that the device does not fail during presentation.
- Wyatt: Write up sections on the final report that I worked closely with.
- Shreya: Work on final report

Week 12/13: Final Report and Peer evaluation due (12/17)

- Grace: Finalize report and testing final parts of project, and aid others in whatever sections need to be completed. Complete peer evaluation.
- Nic: Finish writing my sections of the report and aid others to write their sections. Complete the peer evaluation.
- Wyatt: Finalize the report and complete peer evaluation.
- Shreya: Work on final report

8.2 Project Budget

	Name	Vendor	Part Number	Quantity	Unit Price	Total Price
DAC	Op amp	Digikey	LT1636CS8#PBF-ND	8	4.61	36.88
	10k resistor	Digikey	RNCP0805FTD10K0CT-ND	112	0.0297	3.33
	20 k resistor	Digikey	RNCP0805FTD20K0CT-ND	112	0.0297	3.33
Current	50mOhm	Digikey	FC4TR050	10	0.62	6.2

Detection and Surge Protection	resistor		DERCT-ND			
	RS Latch	Digikey	296-CD4044BDW-ND	10	0.46	4.6
	Low input bias op amp	Digikey	OPA2189ID	9	5.45	49.05
	750 resistor	Digikey	ERJ-PB3D7500V/P20269CT-ND	10	0.135	1.35
	3.3M resister	Digikey	RK73H2ATTD3304F/2019-RK73H2ATTD3304FCT-ND	10	0.033	0.33
Mictrocontroller		Digikey	MSP432P401M	2	24.59	49.18
Power Supply	5V regulator	Digikey	L7805CV	1	0	0
	3V regulator	Digikey	LD33V	1	0	0
	charge pump	Digikey	ICL7662CBA+-ND	1	5.41	5.41
	12V wall plug	Digikey	SWI12-12-N-P5/102-3430-ND	1	12.67	12.67
	Barrel Jack	Digkey	PJ-102AH/CP-102AH-ND	1	0.76	0.76
PCB	100mmx125mm 4 layer board 5pcs	AllPCB	n/a	1	44	44
	SMD Stencil	AllPCB	n/a	1	20	20
	Shipping	AllPCB	n/a	1	58.61	58.61

FPGA	FPGA	Digikey	10M02DCV36C8G	8	3.78	30.24
				total:		325.94

8.3 Minimum Standard for Project Completion

The minimum requirements for the project completion is as follows:

It will contain 1 output voltage with 4 bits of precision between 0V to 5V with a maximum of 15mA with no more than 75 mW of power per channel. The accuracy of the device should be within +/- 2 LSB. It will still be in an insulated box with current shut off. The basic SCPI commands that will be accepted will be to set voltage and to read the voltage and current values of the part.

The minimum requirement of the user interface will allow the user to set the voltage and current threshold and be able to read the current of the channel. This has to be done without the use of SCPI commands.

However, even though this is the minimum standard this will not be the goal and is the worst case scenario if other parts of the project fail.

8.4 Final Demonstration

The final demonstration of the working prototype will be as follows:

1. User Interface that can set and read the expected voltage and actual current of each channel on the board and display the information to the user.
2. Show the functionality of writing a python script and sending the information to the microcontroller that will then set and control the board's output channels. To verify it is functioning properly there will be a voltmeter measuring the output and or will power a circuit of some sort.
3. Due to the photonic circuit's inability to be moved easily, part of the demonstration will include showing a video of the controller board working on a physical photonic circuit.

The tests to be shown:

1. The output of our device will be attached to an oscilloscope to show the signal integrity of the output of our device to prove that we have a clean output voltage, with noise

matching the required precisions (causing < +/- 0.5LSB per channel). This test will be conducted with only one channel outputting voltage, all channels outputting the same voltage, and when all channels output various voltages. These various tests aim to show how crosstalk between signals affects the noise level of the output.
2. There will be a test on another output channel connected to a voltmeter that can read voltages to precision of at least 500uV to show both the accuracy of the output of the voltage our device is to the desired voltage programmed by the user..

Furthermore our presentation will include graphs and data analysis that discusses the tests and their results that were mentioned earlier in section 6.

References

1. C. Doerr *et al.*, "Single-chip silicon photonics 100-Gb/s coherent transceiver," *OFC 2014*, 2014, pp. 1-3, doi: 10.1364/OFC.2014.Th5C.1.
2. Kirill Zinoviev, Laura G. Carrascosa, José Sánchez del Río, Borja Sepúlveda, Carlos Domínguez, Laura M. Lechuga, "Silicon Photonic Biosensors for Lab-on-a-Chip Applications", *Advances in Optical Technologies*, vol. 2008, Article ID 383927, 6 pages, 2008. https://doi.org/10.1155/2008/383927
3. Feldmann, J., Stegmaier, M., Gruhler, N. *et al.* Calculating with light using a chip-scale all-optical abacus. *Nat Commun* 8, 1256 (2017). https://doi.org/10.1038/s41467-017-01506-3
4. Feldmann, J., Youngblood, N., Karpov, M. *et al.* Parallel convolutional processing using an integrated photonic tensor core. *Nature* 589, 52–58 (2021). https://doi.org/10.1038/s41586-020-03070-1
5. S. G.-C. Carrillo, E. Gemo, X. Li, N. Youngblood, A. Katumba, P. Bienstman, W. Pernice, H. Bhaskaran, and C. D. Wright, “Behavioral modeling of integrated phase-change photonic devices for neuromorphic computing applications,” APL Mater. **7**, 091113 (2019). https://doi.org/10.1063/1.5111840
6. Arrazola, J.M., Bergholm, V., Brádler, K. *et al.* Quantum circuits with many photons on a programmable nanophotonic chip. *Nature* 591, 54–60 (2021). https://doi.org/10.1038/s41586-021-03202-1
7. https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/dc2025a-a.html#eb-overview
8. https://www.analog.com/media/en/technical-documentation/data-sheets/2668fa.pdf
9. https://www.ueidaq.com/products/8-channel-isolated-0-20-ma-d-a-board-with-current-and-voltage-readback
10. https://www.rigolna.com/products/dc-power-loads/dp800/
11. https://www.electronics-tutorials.ws/opamp/opamp_3.html
12. https://www.analog.com/media/en/training-seminars/tutorials/MT-015.pdf
13. https://www.rfwireless-world.com/Terminology/Difference-between-DAC-types.html
14. https://e2e.ti.com/blogs_/archives/b/precisionhub/posts/need-a-higher-accuracy-from-r2r-ladder-based-architecture-try-trimming
15. https://e2e.ti.com/blogs_/b/analogwire/posts/dac-essentials-the-resistor-ladder