DC-DC CONVERTER CONTROL SYSTEM FOR THE ENERGY HARVESTING FROM EXERCISE MACHINES SYSTEM

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

by

Alexander Sireci

June 2017

COMMITTEE MEMBERSHIP

TITLE:	DC-DC Converter Control System for the Energy Harvesting from Exercise Machines System
AUTHOR:	Alexander Sireci
DATE SUBMITTED:	June 2017
COMMITTEE CHAIR:	David Braun, Ph. D. Professor of Electrical Engineering
COMMITTEE MEMBER:	Xiao-Hua (Helen) Yu, Ph. D. Professor of Electrical Engineering
COMMITTEE MEMBER:	Dale Dolan, Ph. D. Professor of Electrical Engineering

ABSTRACT

DC-DC Converter Control System for the Energy Harvesting from Exercise Machines System

Alexander Sireci

Current exercise machines create resistance to motion and dissipate energy as heat. Some companies create ways to harness this energy, but not cost-effectively. The Energy Harvesting from Exercise Machines (EHFEM) project reduces the cost of harnessing the renewable energy. The system architecture includes the elliptical exercise machines outputting power to DC-DC converters, which then connects to the microinverters. All microinverter outputs tie together and then connect to the grid. The control system, placed around the DC-DC converters, quickly detects changes in current, and limits the current to prevent the DC-DC converters and microinverters from entering failure states.

An artificial neural network learns to mitigate incohesive microinverter and DC-DC converter actions. The DC-DC converter outputs 36 V DC operating within its specifications, but the microinverter drops input resistance looking for the sharp decrease in power that a solar panel exhibits. Since the DC-DC converter behaves according to Ohm's Law, the inverter sees no decrease in power until the voltage drops below the microinverter's minimum input voltage. Once the microinverter turns off, the converter regulates as intended and turns the microinverter back on only to repeat this detrimental cycle. Training the neural network with the back propagation algorithm outputs a value corresponding to the feedback voltage, which increases or decreases the voltage applied from the resistive feedback in the DC-DC converter.

In order for the system to react well to changes on the order of tens of microseconds, it must read ADC values and compute the output neuron value quicker than previous control attempts. Measured voltages and currents entering and leaving the DC-DC converter constitute the neural network's input neurons. Current and voltage sensing circuit designs include low-pass filtering to reduce software noise filtering in the interest of speed. The complete solution slightly reduces the efficiency of the system under a constant load due to additional component power dissipation, while actually increasing it under the expected varying loads.

ACKNOWLEDGMENTS

I would like to dedicate this thesis to my parents, Donald and Christine Sireci, for their continued guidance and support, enabling me to follow my dreams. I would also like to thank them for teaching me that, through hard work and dedication, I can reach even my most difficult aspirations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1. Introduction

The Energy Harvesting from Exercise Machines system came to life in 2006 with a conversion process from a bicycle's physical motion to electricity [1]. The project aims to fabricate an economically viable energy harvesting machine that also reduces greenhouse gas emissions. Multiple companies produce similar systems. ReRev produces systems which harvest electricity from elliptical machines, but some price estimates put each machine's cost at $914 [2]. Due to the ReRev machines' intermittent use in a gym, the payback period exceeds the machines' lifetimes. Other companies in this sector, such as The Green Revolution and Plug Out, had seen some success, but their websites no longer operate, appearing that they no longer operate either [3]. Plug Out sells their energy capturing equipment at the same price as their standard machines, but their business model seems to have failed. Sports Art currently sells elliptical machines with microinverters integrated inside and claims the gym owner receives significant savings. However, no data support this claim, especially considering the model runs $2,200 more than their similar model without energy capturing capabilities [4]. The numerous attempts at viable solutions to capturing energy from exercise machines all fail in reaching viable return on investment periods. These companies do a great job selling their products to the green centric crowd; however, they fail to reach those gyms looking to maximize their profits. Cal Poly's work strives to bring the cost of such machines down to create a return on investment under 7 years, meaning the energy harvesting system pays for itself over its lifetime. Gym owners buy the systems to save on operating costs, as well as attracting additional, sustainability focused members.

From data obtained in 2009 and part of 2010, students and faculty use Cal Poly's elliptical machines 57% of the time, during normal gym hours [5]. In 2014 and 2015, Cal Poly admitted record-breaking numbers of students to the school, which only increases the usage and shortens the payback period [6]. Due to an absence of current estimations of the usage, the old data must suffice. Once the REC Center implements the system, another study could provide more insight into the effectiveness of the EHFEM system to harness green energy from muscle power. Calculating the return on investment, using the 57% usage figure, shows that the REC

center would save $482.11 per year across 15 elliptical machines, assuming 100% efficiency [5]. This amounts to savings of $32.14 per year per elliptical. The system realistically contains some losses, but this gives a ballpark estimate proving such a system's cost-effectiveness.

Alvin Hilario's master thesis motivates the implementation for the control system. He recommends using the control system to adjust the duty cycle of the DC-DC converter, improving its efficiency [7]. He also recommends adjusting the phase of the microinverter, but this nullifies the inverter's approval for connection to the grid and eliminates this possibility. The control system needs a maximum power point tracking (MPPT) algorithm, similar to current systems capturing wind energy [8]. The control system also limits voltage and current spikes to prevent damage to any components. Another method of controlling the system most accurately, involves controlling the feedback loop inside the DC-DC converter. Therefore the converter needs to have the right external connections. This project makes modifications to a custom DC-DC converter created from Andrew Forster's master's thesis, to allow a microcontroller access to the feedback loop [9]. Michael DeSando's thesis describes a universal battery charging circuit in this way, and the same principles he presents aid in the design of this control system [10]. The microcontroller measures the voltage and current output from the DC-DC converter and determines the voltage to present to the feedback pin. The voltage and currents are read with two ADCs, processed with the microcontroller, and then output to the feedback. The voltage could output to the feedback with a DAC, but a PWM signal and a low pass filter can also produce a DC voltage. DeSando uses a PWM signal, since the microcontroller he uses does not have an output pin connected to the internal DAC [10]. In this project, I use PWM to reduce number of hardware components.

This control system improves the DC-DC converter response time. A DC-DC converter includes a feedback path that regulates the output voltage [7]. A microcontroller outputs to this feedback pin. The advantages include temperature adjustments, as well as optimizing the feedback to stabilize the loop. A feedforward implementation could increase the speed of the microcontroller. Feedforward reduces large disturbances on the output of the system [11]. The elliptical causes large changes in output power on the order of microseconds. Instead of waiting until the output changes, the feedforward helps the DC-DC converter counteract the disturbance

as it occurs. Additionally, the microinverter changes the output voltage of the DC-DC converter requiring a smart feedback loop.

Feedforward neural networks commonly determine internal characteristics of complicated, nonlinear systems. Artificial Neural Networks (ANN) employ different learning algorithms to track voltages output in several different DC-DC converters [12]-[15]. This thesis applies ANN methodology to a specific DC-DC converter.

The system level diagram appears in Figure 1, with the area of focus on the DC-DC converter. Although the converter itself lies outside of this project's scope, the control system must interface with any DC-DC converter for optimal performance. The control system uses a microcontroller to adjust feedback loop gains.

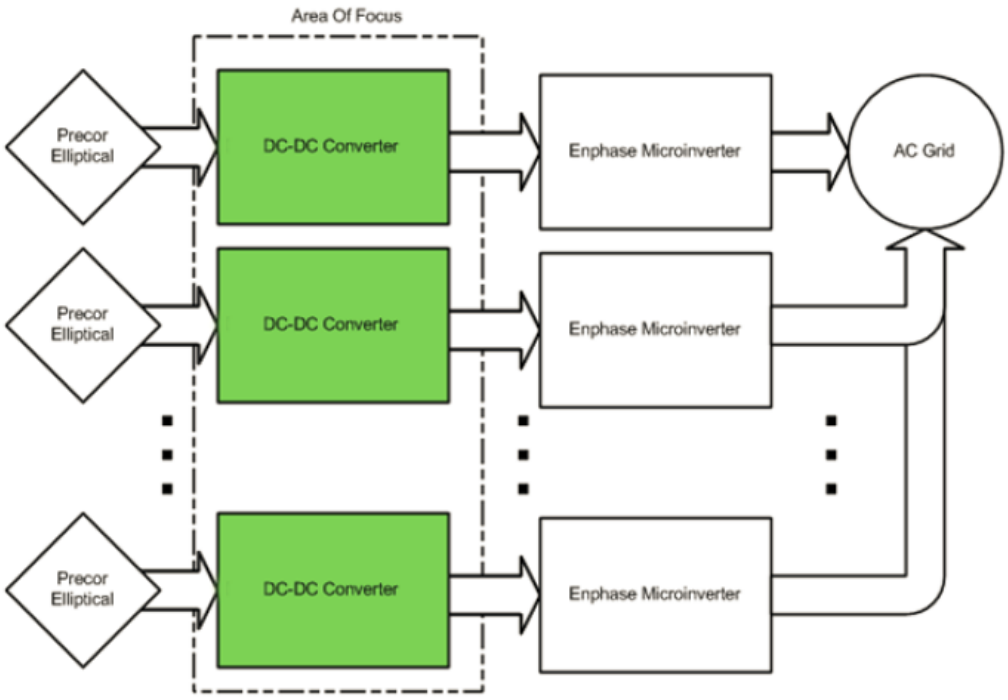

Figure 1. System Block Diagram [7]

"Maximum Power Point Tracking Based Optimal Control Wind Energy Conversion System" describes a controller method for windmills tracking maximum power efficiency through the angular position [8]. Although this approach would work, it adds unnecessary hardware and reduces compatibility with exercise machines. In addition, more inputs require more hardware to

measure values, which increases cost and space the control system occupies inside the exercise equipment. Initially, this approach seems likely to improve system performance, but the extra sensors impact the system significantly in cost, volume, and compatibility specifications.

1.1 Thesis Statement

The Energy Harvesting for Exercise Machines (EHFEM) project must respond to real-time changes with elliptical pedaling speeds. The fastest computation methods align with the most expensive ones, so exploring the limitations of microcontrollers with clocks around or above 100 MHz directs future attempts in a more narrow direction. This thesis project studies the possibility of a neural network implementation into the EHFEM system and compares different neural network types.

This project designs an artificial neural network and voltage and current sensing circuits that permit the use of the neural network in a feedback control system to regulate a four-switch buck-boost converter. This project also measures the impact on system conversion efficiency.

With the thesis defined, the next chapter itemizes detailed specifications and customer needs.

Chapter 2. Customer Needs, Requirements, and Specifications

This chapter defines the customers, lists the engineering specifications, and decides on a microcontroller that can meet the specifications.

2.1 Customer Needs Assessment

The customers involve anybody who either uses or buys the machines. In addition, since the utility company receives the captured electricity from the inverter, and Dr. Braun receives the designs, they are customers. The utility company needs to receive the electricity in a way to capture it and help generate more electricity at peak load times. Dr. Braun organizes the larger system, and needs this piece to fit seamlessly within the rest of the system. He also needs this piece, integrated within the rest of the system, to fit inside the machines and have a positive monetary return on investment. I must provide him with enough documentation for him to build copies of my project for future expansion projects.

The REC Center Management, the most direct customer, focuses on safety. Machines in the REC Center experience heavy use, which makes them ideal places to capture energy, but also exposes them to spilled water. Depending on the implementation size and placement inside of the elliptical, the REC Center may require water resistant casing. Preventing spilled water from shorting circuit components protects users from experiencing unwanted exposure to high voltage and current levels within the EHFEM circuitry. Secondarily, the REC Center needs to reduce the running costs of the gym, reduce their carbon footprint, and the system to fit inside their current exercise machines. Everyone who uses the machines, which includes me, needs an unaltered workout experience to maintain their safety and enjoyment. The resistance must still relate to the numbered resistance on the display and not change suddenly. Sudden increases or decreases in resistance could cause injury to the user, such as pulled muscles, or cause their feet to slip off the pedals. Thinking about all human interactions on a system level determines these needs and the marketing requirements.

2.2 Requirements and Specifications

Table 1. DC-DC Converter Control System Engineering Specifications

Marketing Requirements	Engineering Specifications	Justification
3	The system must improve actual efficiencies of the DC-DC converter and microinverter.	DC-DC converter and microinverter datasheets usually cite the highest efficiencies achieved. The system's practical efficiency improves as the time spent converting power increases.
2,7	The peak current cannot exceed 5A into the DC-DC converter and 7.5 A out [7].	The current cannot cause the failure modes to trip, or exceed component absolute maximum ratings.
5	The control system cannot consume more than 10 W of power during normal operation and less than 1 mW of power during standby.	The controller must make the system more efficient and, therefore, cannot consume a substantial percentage of power.
3-8	The controller must cost less than $50.	The overall system needs to pay for itself over its life, and taking the other pieces of the product into consideration, a reasonable estimate uses about 1/5 of the project total price.
3-8	The system must fit within a rectangular prism with dimensions 6 inches wide, 6 inches long, and 4 inches deep, measured inside the Precor elliptical.	If the system does not fit inside the current exercise machine casing, building a casing increases cost and crowds the gym.
4-6,8	The system must meet all specifications over a shelf life of 12 years and a 10 year active life.	It may take a long time until integration of the controller, and it still must work for a long time to ensure a reasonable payback time.
3,7	The system must meet all requirements within a temperature range from 10 ºC to 60 ºC	A gym's temperature often exceeds room temperature due to the number of people in it. The system should run efficiently regardless of varying room temperatures.
4-6,8	The system must receive power from an 11 to 13 V battery powering system present in the current elliptical trainer.	Even if voltages need stepping down, using power already available keeps the system cost low and physical size minimized.
9	The system must meet all NEC and IEEE 1547 requirements.	These codes specify certain requirements to keep systems safe and reliable.
2	Resistance levels displayed on the Precor elliptical must not change when the EHFEM system harnesses energy from when the resistor dissipates the energy as heat.	During EHFEM failure states, the resistance to motion may drop out or change suddenly. Users want to use the same resistance levels normally available to them.
1,8	The EHFEM system must reside within a NEMA 3R enclosure [16].	Users may spill water or other liquids on the EHFEM system and try to touch components. This NEMA class prevents dripping liquids from reaching electronics.

The marketing requirements determine the specifications to fulfill the project's goal. The system must respond quickly, so the specifications list several different measurements of this response speed. Specifications, like cost and size restrictions, come from profitability goals and existing machine sizing. Environmental factors the control system must endure, determine other specifications. I cannot directly test the shelf life specification, but calculations and simulation adequately evaluate aging effects on performance. To ensure compliance with all laws and general good practice guidelines, the design follows several different sets of rules. Table 1 includes a complete list of specifications and Table 2 contains the corresponding marketing requirements.

Table 2. DC-DC Converter Control System Marketing Requirements

Marketing Requirements
1. Does not harm the user
2. Maintain user experience
3. Regulate varying input currents to maximize power generation
4. Cheap parts cost
5. Low power consumption
6. Fit inside exercise machines
7. Prevent the DC-DC converters and microinverters from entering failure states.
8. The system must pay for itself over its life.
9. Follow all laws and regulations

The next section ensures the system meets these requirements through an initial design feasibility assessment.

2.3 Design Challenges

The main challenge presents itself in keeping costs sufficiently low to make the system pay for itself over its lifetime. Due to electricity's low cost, the minimal amount of energy humans exert during exercise, and the non-constant usage, the EHFEM must last many years to become economically beneficial [5]. Another challenge arises from the variable power output that can cause voltage and current spikes every microsecond. As a result, the microcontroller must respond extremely quickly.

Another difficulty lies in interfacing the controller to the rest of the system. The inputs to the ADC have magnitudes up to 51 V, which exceeds the microcontroller's 2.4 V to 3.6 V range

[17], [18]. An attenuator reduces the magnitude and adds an offset voltage to meet this specification. Attenuating the voltage increases the need for high ADC resolution.

Completing the system prototype and proving key functionality before key deadlines pose the last design challenge.

2.4 Major Project Milestones

Table 3 includes major milestones throughout the design of this project.

Table 3. DC-DC Converter Control System Deliverables

Delivery Date	Deliverable Description
2/14/2015	Design Review
1/26/2015	EE 461 demo
1/30/2015	Update cost analysis and method of cost reduction
3/14/2016	EE 461 report
11/2/2015	ABET Sr. Project Analysis
6/15/2017	EE 462 demo
6/16/2017	EE 462 Report
6/16/2017	Email Final Thesis to Committee
6/19/2017	Thesis Defense

As the first step toward thesis completion, choosing a microcontroller quickly provides insight of the project possibilities and the environment to employ the final solution.

2.5 Choosing a Microcontroller

To control the power, voltage, and current in the EHFEM system, a microcontroller provides flexibility of control with software, ability to adjust gain in various temperatures, and parallel processing of inputs. The microcontroller must provide a platform to meet all specifications and, most notably, account for speed requirements. Many microcontrollers meet the minimum requirements, but examining two common microcontrollers closely determines the best one. The MSP430 series microcontroller uses a low-power design, but its low clock frequency of 16 MHz and high ADC conversion time disqualify it [10]. Due to the performance improvement, I consider the SAM4S Xplained Pro Evaluation Kit. DeSando uses a C2000 series controller from Texas Instruments to control a DC-DC converter, so this comparison investigates this controller for this similar application [10]. Faster microcontrollers exist, but only for extremely specific applications, so the next sections evaluate the performance of the previously mentioned C2000 and SAM4S microcontrollers.

2.5.1 C2000 EVALUATION

The microcontrollers need to read analog voltages, execute a certain number of instructions on that value, and then output a change in duty cycle to the DC-DC converter. I determine the overall speed by the sum of the time it takes to complete each task. Moreover, two input values, current and voltage, need measuring, so an ideal microcontroller can process the inputs in parallel. The C2000 can process the inputs in parallel, which would slightly decrease error, since the controller samples voltage and current simultaneously and could use them to calculate power. The C2000 has optimization for power measurement algorithms, which can reduce the number of instructions needed to calculate power [19]. Table 4 shows the C2000 system clock frequency and ADC frequency. Figure 2 shows the timing characteristics for the ADC, and shows that the sampling time of the ADC equals 7 clock cycles + 13 conversion A clocks + 13 conversion B clocks + 2 clock cycles. This sampling time of 777.7 ns, measures both current and voltage, so needs halving for comparison with the Atmel microprocessor.

Table 4. C2000 Clock Frequencies [19]

		MIN	NOM	MAX	UNIT
SYSCLKOUT	$t_{c(SCO)}$, Cycle time	11.11		500	ns
	Frequency	2		90	MHz
LSPCLK(1)	$t_{c(LCO)}$, Cycle time	11.11	44.4(2)		ns
	Frequency		22.5(2)	90	MHz
ADC clock	$t_{c(ADCCLK)}$, Cycle time	22.22			ns
	Frequency			45	MHz

(1) Lower LSPCLK will reduce device power consumption.
(2) This is the default reset value if SYSCLKOUT = 90 MHz.

The overall speed of the C2000 microprocessor calculation uses equation 2.1. The datasheets specify ADC and PWM conversion times; however, the number of instructions requires estimation. The estimation determines the superior microcontroller in the microcontroller comparison section.

$$T_{total} = T_{conv\_ADC} + T_{instructions} + T_{conv\_PWM}$$

$$T_{total} = T_{conv\_ADC} + n_{instructions} \times T_{clock} + T_{conv\_PWM} \quad (2.1)$$

The following section repeats the same calculations with a different microcontroller for close examination.

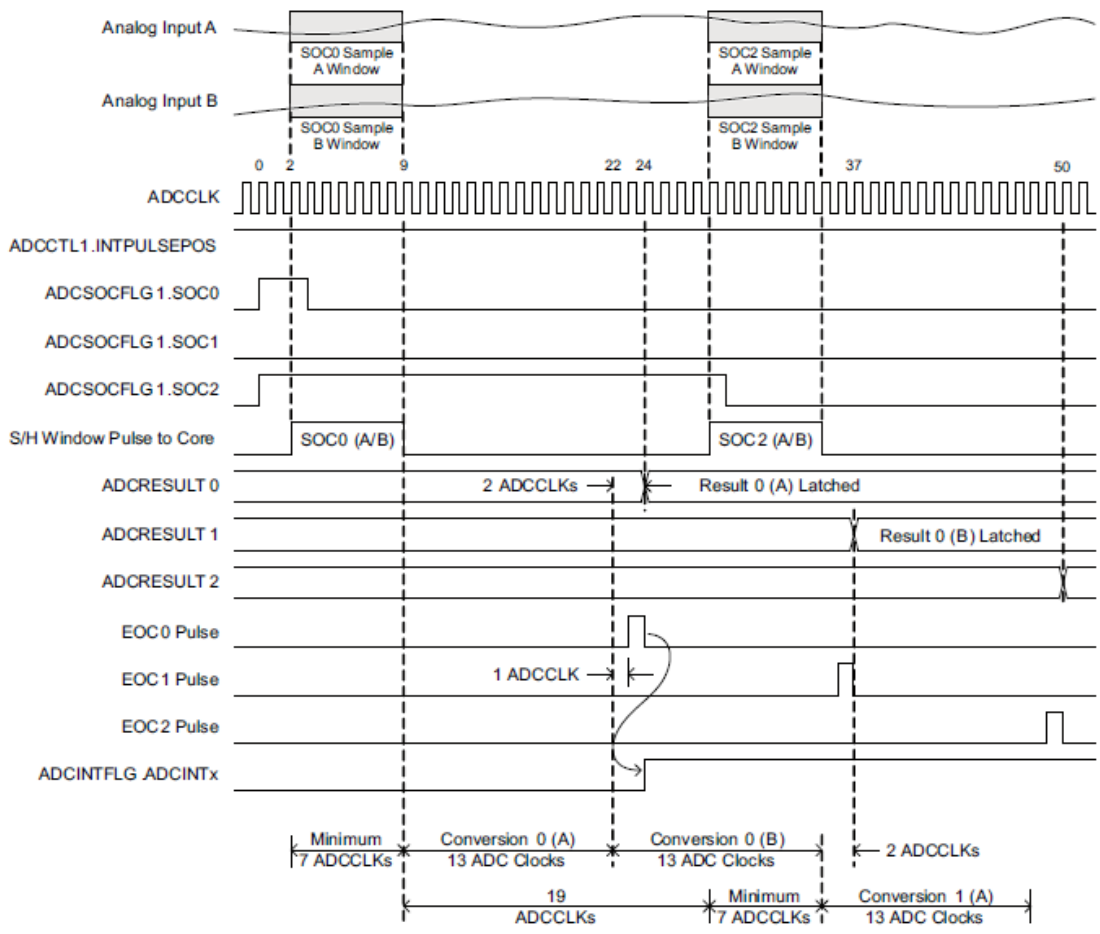

Figure 2. C2000 ADC Timing Diagram [19]

2.5.2 ATMEL SAM4S EVALUATION

The Atmel SAM4S does not have parallel sampling, but samples 1 µs apart likely contain minimal error. This controller catches some attention due to its high clock frequency. The impressive speed allows 33% more calculations per second than the C2000. This microcontroller runs faster relative to the C2000, as the number of instructions in the control routine increases. Atmel does not give much insight into the timing of the ADC, but includes the times to have an overall sampling frequency of 1.1 MHz as shown in Table 5 [20]. Therefore, the ADC conversion time equals 909.1 ns. Figure 3 gives insight into the ADC timing, but the hold time lacks specification. One can assume the max sampling frequency includes the hold time.

Table 5. SAM4S ADC Timing Characteristics [20]

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{ADC}	Clock Frequency		1	20	22	MHz
t_{CP_ADC}	Clock Period		45	50	1000	ns
f_S	Sampling Frequency		0.05	1	1.1	MHz
t_{START}	ADC Startup time	Sleep mode to Normal mode Fast Wake-up mode to Normal mode	20 4	30 8	40 12	µs
$t_{TRACKTIM}$	Tracking Time	Refer to notes 1 and 2.	15(1)	–	–(2)	t_{CP_ADC}
t_{CONV}	Conversion Time (3)	Number of ADC clock pulses to perform a conversion. TRACKTIM < 15	–	20	–	t_{CP_ADC}
t_{CAL}	Calibration time	Calibration time given for one channel and one gain/one offset.	–	306	–	t_{CP_ADC}

Notes:
1. If ADC_MR.TRACKTIM is programmed with a value < 15, then the min. value is applied by default.
2. Refer to Figure 44-21 "Simplified Acquisition Path" for the max. tracking time computation.
3. Sampling frequency $f_s=1/t_{CONV}$ in FREERUN mode. Oherwise, f_s is defined by the trigger timing. If TRACKTIM >14: $t_{CONV} = t_{HOLD} +(TRACKTIM+1) \times t_{CP\_ADC}$ with hold time $t_{HOLD}=5\ t_{CP\_ADC}$.

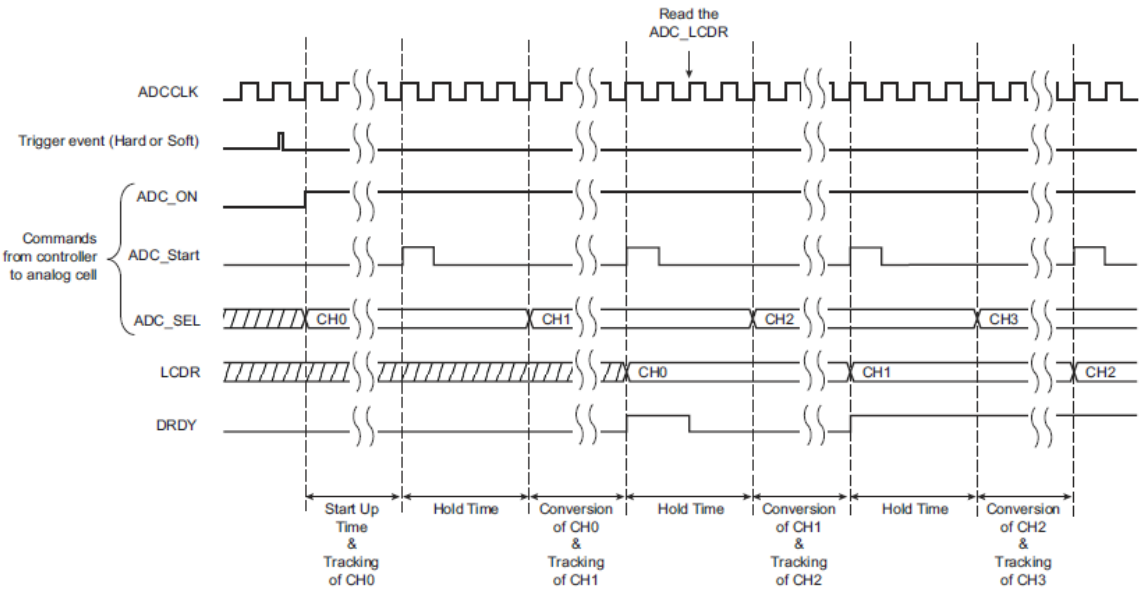

Figure 3. SAM4S ADC Timing Diagram [20]

A direct comparison completes the picture relating both microcontrollers.

2.5.3 MICROCONTROLLER COMPARISON

First, I evaluate the speed of each microcontroller. To do this, equation 2.1 produces a number for each microcontroller, with n as the only unknown. The time to output a PWM wave through a low pass filter to take the average DC value follows the assumption that each controller produces an output in about the same time. The Atmel likely works slightly faster since it can

complete more instructions per second with the faster clock speed. In considering the extreme case of evaluating no instructions, the C2000 samples faster due to its shorter ADC sampling time. With a large number of instructions, the Atmel SAM4S's speed becomes faster due to its fast clock speed. To find the number of instructions where the C2000 becomes slower, I equate the two equations set equal to each other, and solve for the number of instructions. This method determines the crossover point at 187 instructions. Basically if the interrupt subroutine completes in fewer instructions, then the C2000 becomes a faster solution. Since the lower bound likely sits around 500 instructions, the Atmel SAM4S produces quicker response times.

Table 6 depicts a decision matrix to provide a clear picture of the microcontroller best suited for the project's needs.

Table 6. Microcontroller Decision Matrix

	C2000 LaunchXL-F28069M	Atmel ATSAM4SD32C-XPRO	Rank (1-3)		Weight (%)
Operating frequency	90 MHz	120 MHz	2	3	43
ADC	7 channel, 12-bit	16 channel, 10 or 12-bit	3	3	5
fADC	45 MHz	22 MHz	3	2	10
Parallel Sampling?	Yes	No	3	1	5
tconv	388.9 ns	909.1 ns	3	2	10
Data rate	3.46 MSPS	1.1 MSPS	3	1	10
PWM	90 MHz, 0 to 3.3 V	120 MHz, 0 to 3.3 V	2	3	15
Cost	$24.99	$42.65	3	1	2
Total Score	**2.42**	**2.46**			100

The Atmel SAM4S ends up coming slightly ahead, which confirms the calculations that it performs faster. Since the scores come so close together, reasoning may promote choosing the C2000. For instance, parallel sampling minimizes power reading error. The Atmel can only sample sequentially, giving a power reading with voltage and current measured 1 µs apart. Assuming this does not cause any major issues or intolerable error, this consideration does not lend reason to choose the C2000 over the Atmel SAM4S.

With the heart of the control system chosen, defining the interactions of the controls with the surrounding components becomes the next step.

Chapter 3. Functional Decomposition

This chapter defines the interfaces between the new hardware and existing hardware using a top-down design approach.

3.1 Level 0 Block Diagram

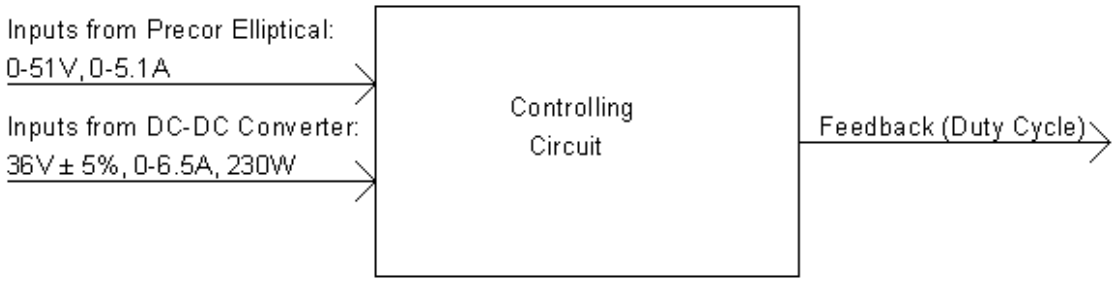

Figure 4. Level 0 Block Diagram

The block diagram in Figure 4 depicts the inputs and outputs to the controller. The controller measures output current and voltage of the Precor elliptical with a resistive divider and a current sense IC. The measurements then enable the system to control the duty cycle of the DC-DC converter described by Andrew Forster in his thesis [9]. The advantages to controlling his system are that since he had not made the final revisions during the start of my thesis, I had the ability to ask him to bring nodes out of the converter for me to control.

Table 7. Functional Requirements of the Level 0 Block Diagram

Module	DC-DC Converter Controller
Inputs	-Voltage from Precor elliptical: 0 to 65 Vpeak -Current from Precor elliptical: 5 A maximum -Voltage from DC-DC converter: 36 V nominal -Current from DC-DC converter: 7 A maximum
Outputs	-Duty cycle feedback to DC-DC converter: 4% to 94%
Functionality	Regulate the duty cycle of the DC-DC converter with the feedback pin to control its output levels to match the microinverter's desired levels.

The input voltage from the Precor elliptical ranges from 0-65 V, protection limited to 51 Vpeak [18]. The input current cannot exceed 5 A due to the Buck-Boost Converter design [9]. Assuming we use the Enphase M190 microinverter, the maximum power from the DC-DC converter cannot exceed 230W [20]. If the system interfaces with the Enphase M215 microinverter, then the power limit increases to 270 W, which does not place a further constraint

on the voltage and current [20]. The duty cycle range from Alvin Hilario's DC-DC converter project describes a range from 4% to 94% [7].

After further decomposition of this block diagram, the thesis becomes well defined.

3.2 Level 1 Block Diagram

Figure 5 shows the level 1 block diagram and Table 8 contains the functional requirements. The input power comes from the Precor elliptical and measurements come from voltage and current sensing circuits. The level 1 block diagram ignores conversions between analog and digital and vice versa.

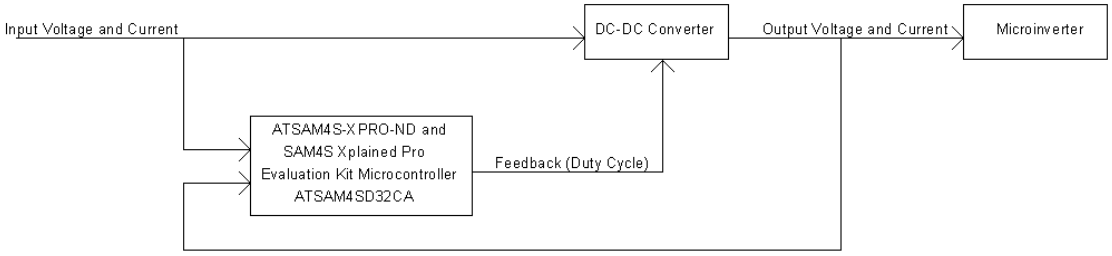

Figure 5. Level 1 Block Diagram

Table 8. Functional Requirements of the Level 1 Block Diagram

Module	**Microcontroller**
Inputs	-Digitized output voltage and current -Input voltage and current
Outputs	-Duty cycle adjust voltage
Functionality	Use the difference in outputs and inputs to adjust the DC-DC converter to the desired power output.
Module	**Microinverter**
Inputs	-DC-DC converter output voltage and current
Outputs	-AC power
Functionality	The microinverter converts the regulated DC voltage from the converter to AC power transferrable to the grid.
Module	**DC-DC Converter**
Inputs	-Input voltage and current -Duty cycle adjust voltage
Outputs	-Output voltage and current
Functionality	The DC-DC converter, designed separately, has a way to adjust the feedback and duty cycle as appropriate. The control system aims to adjust the feedback and duty cycle to provide a tuned response unachievable with static feedback.

Upon defining the thesis's technical requirements, the next chapter schedules tasks.

Chapter 4. Project Planning

In every project, deadlines and key achievements dictate the completion pace. This thesis keeps a strict schedule as a useful progress gauge and to determine time from completion. This chapter also explores project cost, another key planning point.

4.1 Projected Project Timeline

To outline the project's schedule, a Gantt chart portrays the project timeline as shown in Figure 6 and 7. Figure 6 shows the timeline for the 2015-2016 school year and its associated tasks, while Figure 7 shows the timeline for the 2016-2017 school year. The chart allows for the exploration of three different control implementations, as well as test and optimization phases for three different designs. The project plan and thesis defense documents experience continual updates to follow any changes made during the project.

The actual Gantt chart progress differed from the ideal case due to unforeseen obstacles. Figures 8 and 9 show updated versions for each school year. Important research in neural networks, a field I only understood at an introductory level, delayed the design phases until the 2016-2017 school year.

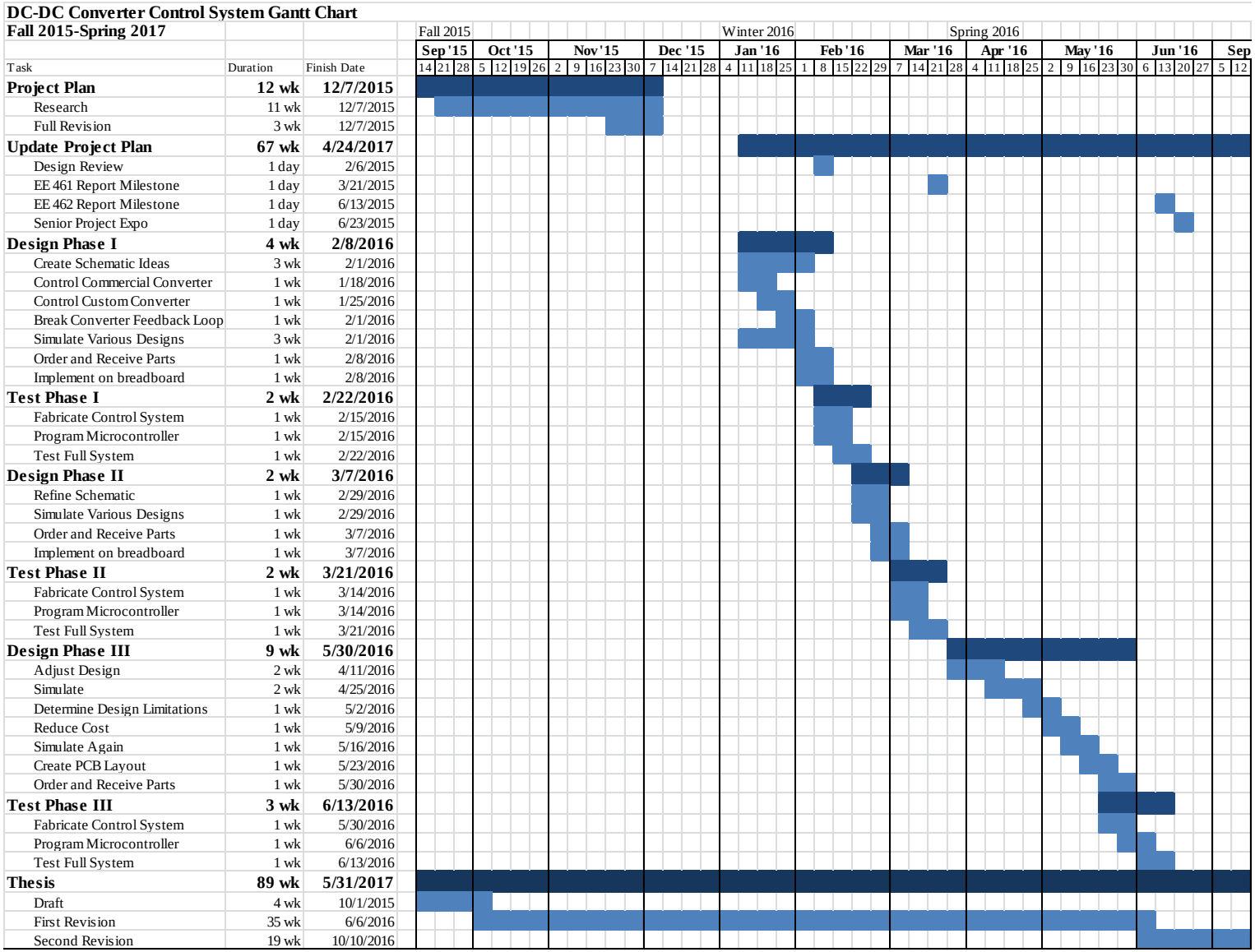

Figure 6. DC-DC Converter Control System Gantt Chart – 2015-16 School Year Initial Plan

DC-DC Converter Control System Gantt Chart
Fall 2015-Spring 2017

Task	Duration	Finish Date
Update Project Plan	**67 wk**	**4/24/2017**
Design Phase IV	**9 wk**	**11/21/2016**
Adjust Design	2 wk	10/3/2016
Simulate	2 wk	10/17/2016
Determine Design Limitations	1 wk	10/24/2017
Reduce Cost	1 wk	10/31/2016
Simulate Again	1 wk	11/7/2016
Adjust PCB Layout	1 wk	11/14/2016
Order and Receive Parts	1 wk	11/21/2016
Test Phase IV	**3 wk**	**12/12/2016**
Fabricate Control System	1 wk	11/28/2016
Program Microcontroller	1 wk	12/5/2016
Test Full System	1 wk	12/12/2016
Design Phase V	**9 wk**	**3/13/2017**
Adjust Design	2 wk	1/23/2017
Simulate	2 wk	2/6/2017
Determine Design Limitations	1 wk	2/13/2017
Reduce Cost	1 wk	2/20/2017
Simulate Again	1 wk	2/27/2017
Adjust PCB Layout	1 wk	3/6/2017
Order and Receive Parts	1 wk	3/13/2017
Test Phase V	**6 wk**	**4/24/2017**
Fabricate Control System	1 wk	3/20/2017
Program Microcontroller	1 wk	3/27/2017
Test Full System	1 wk	4/3/2017
Final System Integration	3 wk	4/24/2017
Analyze Improved Efficiency	6 wk	4/24/2017
Thesis	**89 wk**	**5/31/2017**
Draft	4 wk	10/1/2015
First Revision	35 wk	6/6/2016
Second Revision	19 wk	10/10/2016
Third Revision	16 wk	1/30/2017
Fall Report to Advisor	1 day	12/19/2016
Fourth Revision	9 wk	4/3/2017
Winter Report to Advisor	1 day	3/20/2017
Format Complete Report	4 wk	5/1/2017
Final Advisor Revision	1 wk	5/8/2017
Revise Report	1 wk	5/15/2017
Advisor Approval	1 wk	5/22/2017
Committee Approval	1 wk	5/30/2017
Thesis Defense	1 day	5/31/2017

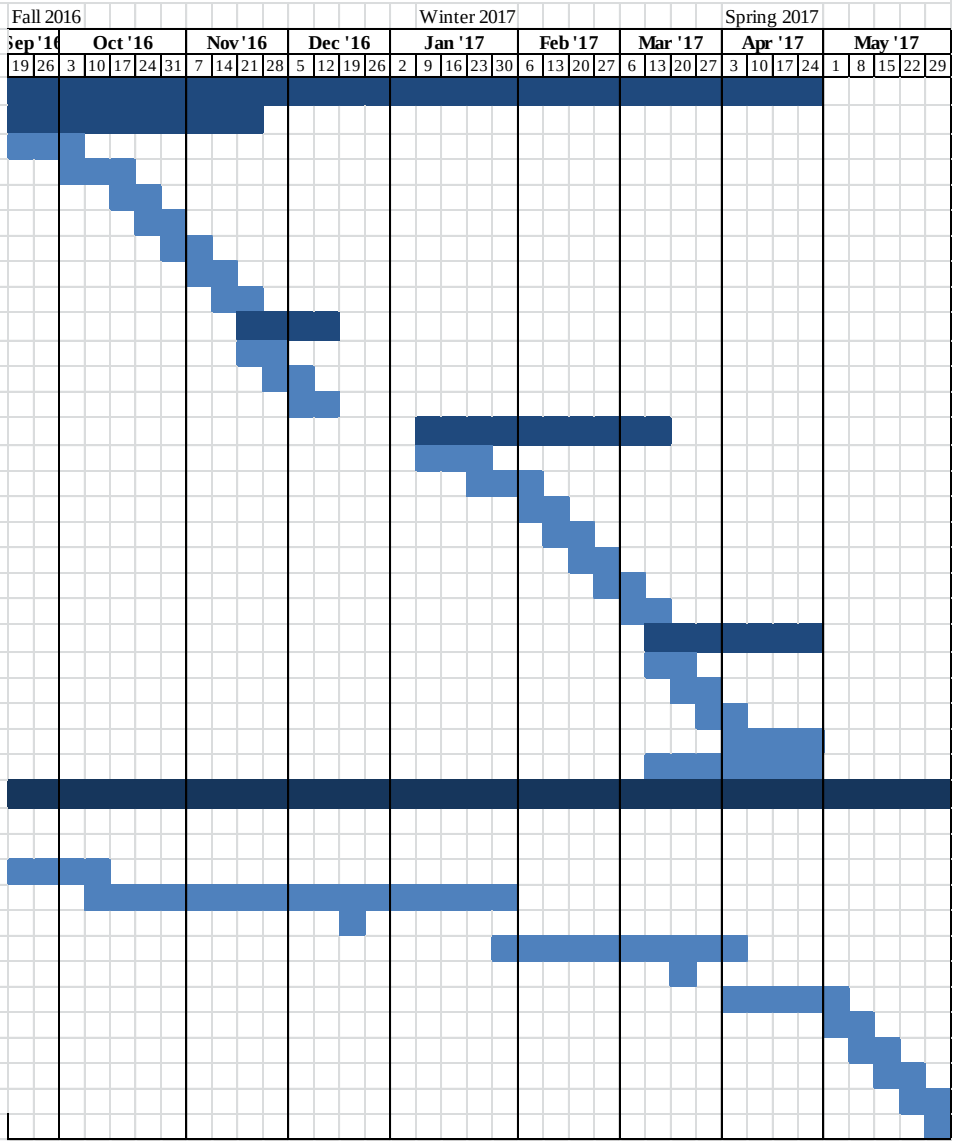

Figure 7. DC-DC Converter Control System Gantt Chart – 2016-17 School Year Initial Plan

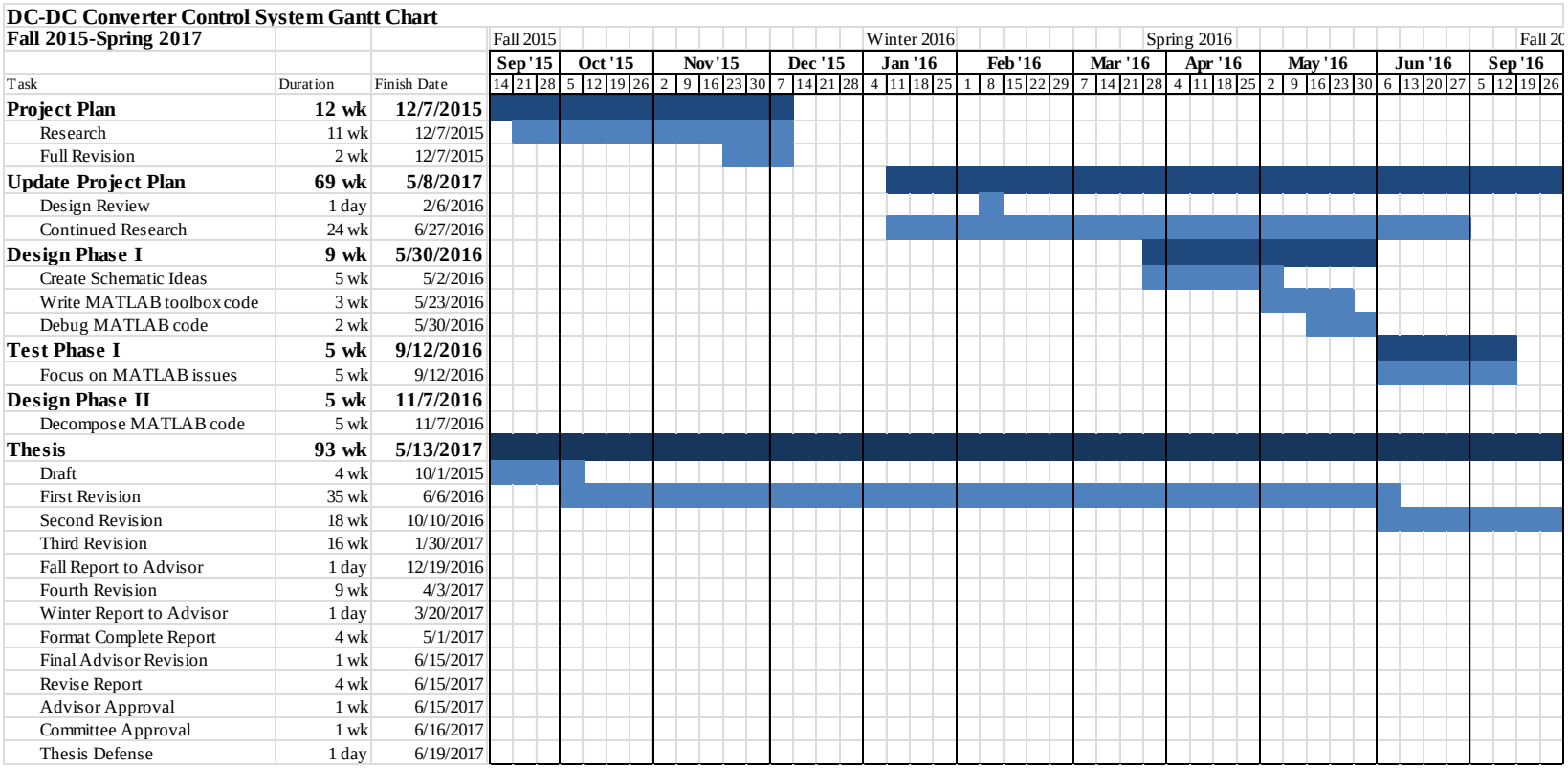

Figure 8. DC-DC Converter Control System Gantt Chart – 2015-16 School Year Actual Progress

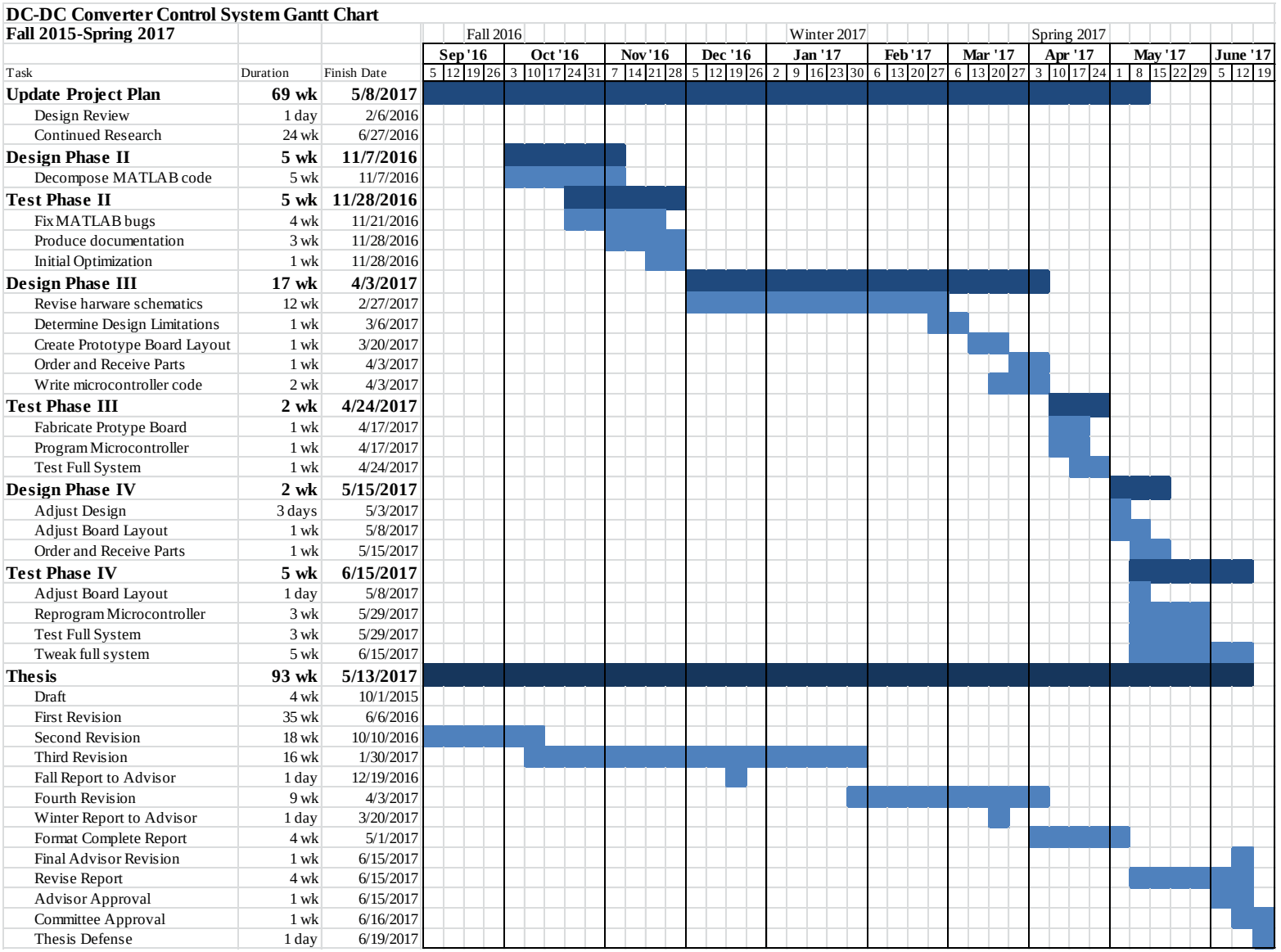

Figure 9. DC-DC Converter Control System Gantt Chart – 2016-17 School Year Actual Progress

Upon examining the project timeline, the project also estimates the costs.

4.2 Preliminary Cost Estimates

The parts cost and labor cost estimation in Table 9 assesses the feasibility of the project. The microcontroller constitutes one major cost, which needs to run fast enough to respond within one microsecond. Funsten and Kiddoo demonstrate that this microcontroller works, and my assessment confirms their findings [18].The labor hourly rate from my most recent internship offer estimates the corresponding field and number of hours equals about 1 hour per day. I budget $10 for a custom voltage sense circuit, after a few unsuccessful attempts at finding an affordable and accurate one. Since the plan uses a breadboard to test my early designs, the budget allots only one PCB due to their high costs.

Equation 4.1 determines cost estimates, as Ford and Coulston suggest in their design book [22]. I employ his equation for every item in Table 9.

$$Cost = \frac{cost_{optimistic} + 4 \cdot cost_{most\ likely} + cost_{pessimistic}}{6} \tag{4.1}$$

Table 9. DC-DC Converter Control System Initial Cost Estimate

	Price per unit					
Item	**Optimistic**	**Most Likely**	**Pessimitic**	**Quantity**	**Total Cost**	**Justification**
Labor (per hour rate)	$24.00	$30.89	$35.00	581	$17,677.89	Internship hourly rate and 1 hour per week
ATSAM4S-XPRO-ND and SAM4S Xplained Pro Evaluation Kit	$42.65	$42.65	$50.00	1	$43.88	From Funsten and Kiddoo [11]
Shipping Costs	$0.00	$4.99	$10.00	3	$14.98	Estimated receiving parts once per design cycle
Resistors	$0.08	$0.10	$0.15	20	$2.10	
Ceramic Capacitors	$0.20	$0.25	$0.50	12	$3.40	
Electrolytic Capacitors	$0.40	$0.50	$0.75	4	$2.10	Likely will need a few since my circuit needs to respond quickly
Voltage Sense Circuit (Custom?)	$2.00	$10.00	$20.00	2	$20.67	Have not found a suitable voltage sense yet
Current Sense IC (LMP8481?)	$0.60	$1.11	$5.00	2	$3.35	Digikey price
PCB	$30.00	$40.00	$55.00	1	$40.83	Estimated from past projects
ADC (ADC122S706?)	$2.00	$3.86	$5.00	2	$7.48	Digikey price
DAC (MCP47A1T-A0E/LT?)	$0.40	$0.57	$1.00	2	$1.23	Digikey price
Parts Totals				**49**	**$140.01**	
Total Project Cost					**$17,817.90**	

The parts cost of $140 exceeds the maximum cost specification, but this cost should drop after multiple design revisions and integrated circuit prices continue to drop. Moreover, bulk discounts on a batch would slash component prices drastically.

4.3 Actual Project Costs

The actual costs of this project come under the expected costs due to not implementing the board design on a PCB. A PCB might provide more accurate results and higher frequency applications, but the simplicity of the hardware and limited time allow the forgoing of this step. Table 10 lists the manufacturer part numbers for easy ordering of identical parts. The prices per unit of some values contain partial pennies when ordered in bulk. Labor costs come above expected values from working on this project more hours than originally budgeted.

Table 10. DC-DC Converter Control System Actual Project Costs

DC-DC Converter Control System Cost Estimate				
Item	**Manufacturer Part Number**	**Price Per Unit**	**Quantity**	**Total Cost**
Labor (per hour rate)		$30.89	723	$22,333.47
ATSAM4S-XPRO-ND and SAM4S Xplained Pro Evaluation Kit	ATSAM4SD32C	$42.65	1	$42.65
Shipping Costs		$7.14	1	$7.14
Capacitor, ceramic, 8200 pF, 50 V, 5%	FK18C0G1H822J	$0.22	2	$0.45
Capacitor, ceramic, 1800 pF, 50 V, 5%	FG18C0G1H182JNT06	$0.23	2	$0.46
Capacitor, ceramic, 0.012 uF, 50 V, 10%	C322C123K5R5TA	$0.22	2	$0.45
Capacitor, ceramic, 0.1 uF, 50 V, 10%	K104K10X7RF5UH5	$0.13	2	$0.26
Capacitor, ceramic, 2.2 uF, 16 V, 10%	FK18X5R1C225K	$0.22	2	$0.45
Capacitor, ceramic, 0.1 uF, 100 V, 10%	SR201C104KAR	$0.17	1	$0.17
Capacitor, electrolytic, 100 uF, 100 V , 20%	UVR2A101MPD	$0.37	1	$0.37
Capacitor, electrolytic, 1 mF, 100 V , 20%	UVR2A102MRD6	$1.49	1	$1.49
Capacitor, ceramic SMD, 10 nF, 1 kV, 10%	C4532X7R3A103K200KA	$0.58	2	$1.15
SOT23 to DIP Adapter	BOB-00717	$0.95	6	$5.70
Male header, 40 pin vertical, 0.1"	HDR100IMP40M-G-V-TH	$0.69	2	$1.38
IC OPAMP, rail-to-rail amplifier	MAX4322EUK+T	$1.65	4	$6.60
IC OPAMP, current sense amplifier	LTC6101HVCCS5#TRMPBF	$3.15	2	$6.30
Prototyping board, 5 cm x 7 cm and jumper wires	CF PCB 001a-1	$1.58	1	$1.58
Resistor SMD, 0.001 Ohm, 1%, 1 W	CSR1206-0R001F1	$0.48	2	$0.95
Resistor, 100 Ohm, 1%, 1/4 W, through hole	RNMF14FTC100R	$0.07	2	$0.14
Resistor, 10 kOhm, 1%, 1/4 W, through hole	RNF14FTD10K0	$0.07	4	$0.29
Resistor, 15 kOhm, 1%, 1/4 W, through hole	RNF14FTD15K0	$0.07	2	$0.14
Resistor, 1 MOhm, 1%, 1/4 W, through hole	RNMF14FTC1M00	$0.07	2	$0.14
Resistor, 5.6 kOhm, 1%, 1/4W, through hole	RNMF14FTC5K60	$0.07	1	$0.07
Resistor, 100 kOhm, 1%, 1/4W, through hole	RNF14FTD100K	$0.07	1	$0.07
Tax				$5.40
Parts Totals			**46**	**$83.81**
Total Project Cost				**$22,411.73**

After completing the project planning and cost assessments, the focus shifts to the technical details.

Chapter 5. Hardware Design and Initial Testing

This chapter explores the design approach and fully explains design tradeoffs. The DC-DC converter throughout this section refers to the design by Andrew Forster using the LT8705 buck/boost converter. Figure 10 shows the DC-DC converter used in testing [9].

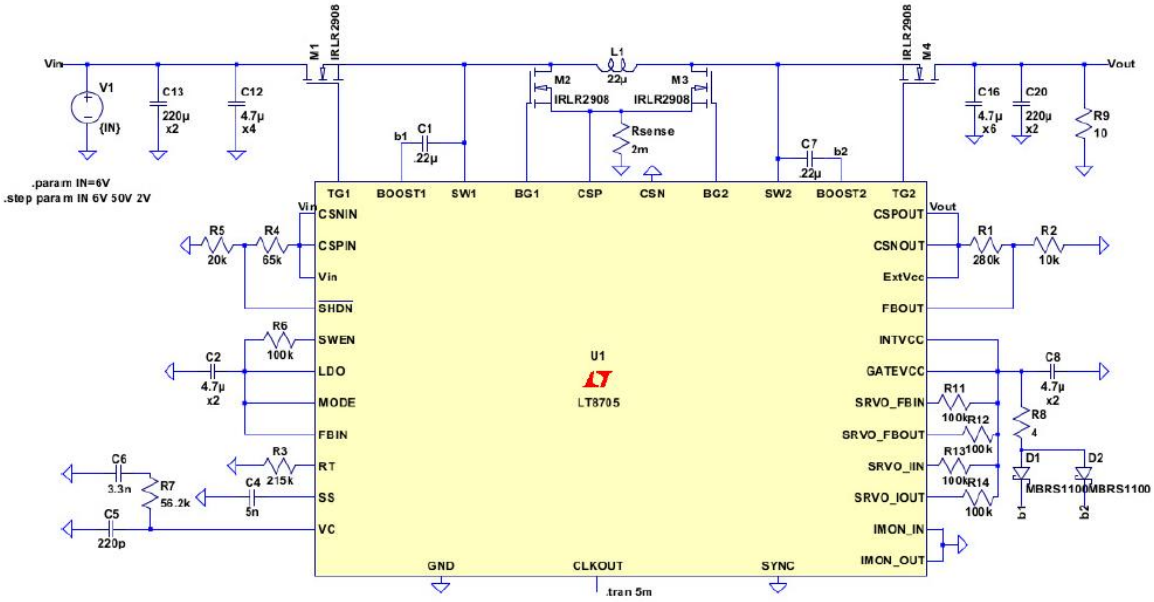

Figure 10. DC-DC Converter Schematic [9]

5.1 DC-DC Converter Theory of Operation

The DC-DC converter used in this system changes operation regions depending on the difference in input and output voltages. Table 11 summarizes the operation regions.

Table 11. DC-DC Converter Operation Regions [23]

Voltage Range	Vout >> Vin	Vout ≈ Vin	Vin >> Vout
Region	Boost	Buck-boost	Buck
Transistor States	M1 ON, M2 OFF PWM M3, M4	4-switch PWM	M4 ON, M3 OFF PWM M1, M2
Relevant Equations	$DC_{(M3,boost)} = \left(1 - \frac{V_{in}}{V_{out}}\right) \times 100\%$ $DC_{(Abs_{min},M3,boost)} = t_{on(M3,min)} \times f \times 100\%$	See Operation section of datasheet	$DC_{(M2,buck)} = \left(1 - \frac{V_{out}}{V_{in}}\right) \times 100\%$ $DC_{(Abs_{min},M2,buck)} = t_{on(M2,min)} \times f \times 100\%$

In Table 11, the typical value of t_{on} in both cases is 260 ns, which provides a ballpark estimate for the limit of control. The microcontroller code needs to prevent the converter from attempting to surpass this limit, since its reaction becomes unpredictable in this region.

In boost region and in buck region, calculations can produce open-loop transfer functions of the DC-DC converter, but in the buck/boost region, the open-loop transfer function becomes difficult to determine. A larger issue arises when determining the closed loop transfer functions. The DC-DC converter's proprietary control scheme causes great difficulty in determining the feedback's effects, except experimentally. Moreover, the control scheme could change slightly, if circuit component values change with aging effects. Therefore, an ideal controller design adaptively determines performance of the DC-DC converter and adjusts the feedback pin appropriately. Neural networks realize this possibility. Neural networks can regulate DC-DC converters more closely than with conventional control schemes [14]. A MATLAB model trains the neural network off-line to specific test data, and then also on-line, with the controller connected in the system to account for parasitic losses [13].

State averaging equations provide control in each mode due to the transistors' pulse width modulation. State averaging would be useful if controlling transistor switching directly, but this project controls transistor gates indirectly through the LT8705 controller.

5.2 Methods of Control of the DC-DC Converter

Initially the controller used a state-space variable representation, since state-space allows controller pole placement anywhere, instead of only of the left hand plane of the s-plane. After realizing the specific controller contains inherent, difficult to characterize, nonlinearities of the system, the following sections examine a neural network approach.

5.2.1 CURRENT CONTROL

The DC-DC converter does not utilize the IMON_IN and IMON_OUT pins. To employ current control, these pins connect to the microcontroller, since when their voltages rise near 1.208V typical, the Vcc pin voltage lowers, and the current decreases. This pin connects to an external resistor and the voltage is proportional to the resistance, but this produces constant

losses, decreasing efficiency. A large current appears to have the same effect as a large voltage to the converter controller, so the next section also examines voltage feedback.

5.2.2 VOLTAGE FEEDBACK PIN

The DC-DC converter integrated circuit regulates the output voltage with hardware. In hardware, comparators flip states to regulate the voltage, which regulates the voltage more loosely. The microcontroller advantages include the ability to react even with a minimal error signal and stronger regulation as the input and output voltages become further apart.

As the voltage on the FBOUT pin becomes larger than its threshold of 1.207 V, the DC-DC converter allows less current into the output, thereby reducing the voltage, and vice versa. Controlling the voltage on this pin with a microcontroller allows faster reaction times to rapidly changing input power. The hardware setup determines if the control occurs in a manner linearly proportional to the voltage above or below the threshold, or if the pin connects to a simple hysteretic comparator for bang-bang control.

5.2.3 CHANGING CONVERTER MODES

Once the controller properly regulates voltage, efficiency improves. One way to improve the efficiency of the DC-DC converter changes the mode between continuous and discontinuous. When the MODE pin falls below 0.4 V, continuous mode activates, while discontinuous mode activates when the voltage exceeds 2.3 V. Burst mode activates between 1 and 1.7 V [23]. While the pin connects to ground, the converter remains in continuous mode, but when the microcontroller outputs to that pin, the mode can change. Burst mode can increase efficiency as well, since it prevents unnecessary switching, which produces power losses [23]. Depending on performance, both discontinuous and burst mode can improve efficiency, so changing the mode from continuous could prove more efficient, but possibly more difficult to control from larger output voltage and current swings.

5.2.4 SENSING CONVERTER OPERATION REGION

As previously discussed and summarized in Table 11, the DC-DC converter changes transfer functions depending on output voltage. The luxury of neural networks enables the network to conclude the relationship at all points.

Controlling the DC-DC converter requires accurate voltage and current sensing.

5.3 Sensing Voltage and Current Before and After the DC-DC Converter

Initially, the neural network harnesses identical information as the DC-DC converter to determine its algorithms. The final neural network uses back propagation instead, correcting any undesired outputs as they appear. The R_{sense} resistor connects on the low-side of the converter switches, so the voltage node between the resistor and switches determines the current through the resistor, since the other side connects to ground. These sense resistors limit the maximum current flowing through the DC-DC converter to prevent overloading, but the current provides insight into the power and helps predict the inverter's performance. The CSPIN, CSNIN, CSPOUT, and CSNOUT pins all help determine the current through the DC-DC converter. Since initially the converter does not use them, their traces need disconnecting and a sense resistor of the same 1 mΩ value needs connecting. However, measuring the current through the inductor would offer an alternative, which might provide sufficient information. If it does, then the input and output current pins could not need any modifications.

To minimize modifications to the DC-DC converter, I designed separate current and voltage sensing circuits. The measurements must match the input neurons to the neural network. Therefore, the microcontroller measures the voltage and the current both before and after the DC-DC converter. A fifth input neuron would require current measurement inside the DC-DC converter and cutting traces, so excluding it avoids this need. Theoretically, the voltage and current ranges before and after the DC-DC converter differ. However, I designed all voltage and current measurement circuits identically, except for the filter capacitors.

5.3.1 VOLTAGE SENSING CIRCUIT DESIGN AND INITIAL TESTING

Simple voltage dividers drop the large voltage to a range the Atmel ADC can handle. I estimate the maximum voltage at 65 V in case the input protection circuit changes or starts passing through larger than the nominal 51 V limit. The 2.6 V reference voltage inside the Atmel microcontroller, set through the potentiometer, determines the maximum voltage the ADC could read. Desiring a maximum voltage of about half of this value, software sets the gain of the Atmel ADC to 2, which halves the reference voltage. The reference voltage cannot drop below 2.4 V for

proper operation of the ADC [17]. Standard resistor values dictate using a 15 kΩ resistor on bottom (R1 in Figure 11) and a 1 MΩ resistor on top to use 80% of the ADC range. The ADC reads the voltage as a 10 bit number to increase readout speed over 12 bit conversions. A capacitor in parallel across the bottom resistor limits the rate the voltage readings change. The bandwidth attenuates noise from the 200 kHz frequency of the DC-DC converter while allowing quick changes in measurement readings with minimal latency. The capacitor value of 8.2 nF sets the bandwidth at 1.31 kHz. Figure 11 shows the voltage sensing circuit, where Vin measures the voltage and Vout_V passes into the ADC in the Atmel Microcontroller.

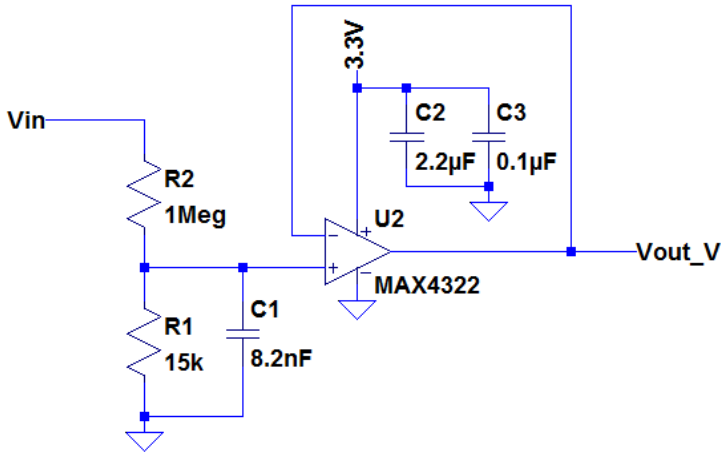

Figure 11. Voltage Sensing Circuit – Vin Ranges from 0 to 65 V Producing Vout Ranging from 0 to 0.96 V

One design decision involved the need of the voltage follower to buffer the output voltage. Despite many arguments against needing the voltage follower, early testing with voltage supplies proved its need. The ADC from the Atmel SAM4S Xplained Pro superimposes a voltage on the node only during operation, which calibration could have accounted for, but the ADC would sometimes go haywire reading all four values as almost identical. Installation of the voltage buffers increases voltage reading consistency. This design chooses the MAX4322 to reuse the design from Dr. Braun. Heritage designs save development time since he used the voltage buffer on an Atmel microcontroller in the EHFEM system [24]. The input resistance to the MAX4322 of 500 kΩ causes a voltage decrease of about 3%, which calibration also accounts for. The power rail connects to the 3.3 V power header pin from the microcontroller evaluation board. Since the

bottom rail connects to ground and voltages close to 0 V need accurate readings as well, the operational amplifier chosen must drive voltages rail-to-rail.

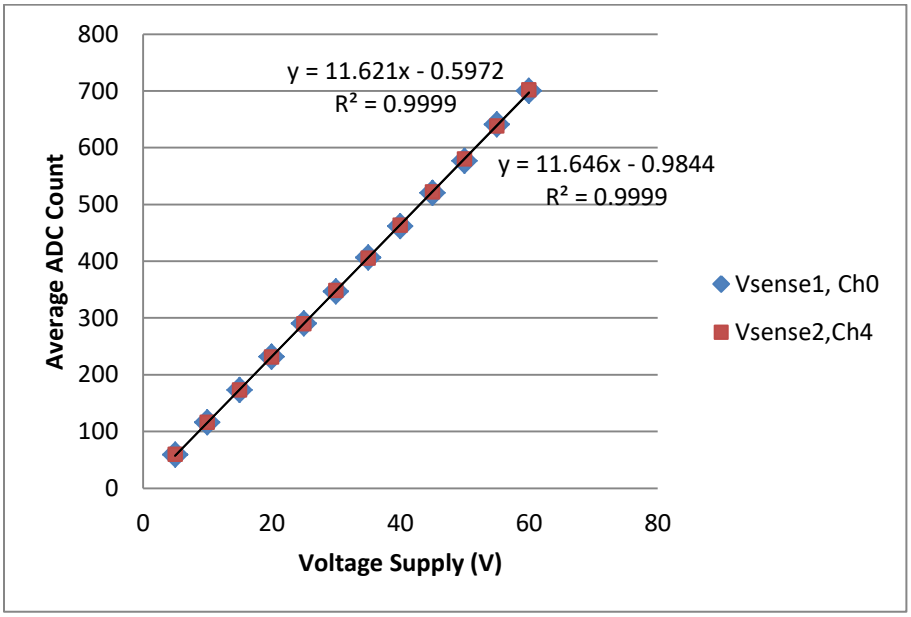

Figure 12. Average ADC Count versus Voltage Supply for Voltage Sensing Circuits

The voltage sensing circuit relates voltage on the input of the voltage divider to an ADC count reading. Figure 12 shows the relationships for both ADC circuits, input to the ADC pin they connect to in the final design. The top trend line equation corresponds to Vsense 1 and the bottom trend line equation corresponds to Vsense 2. The average count from three readings taken about a half second apart tests for consistency and mitigates any random variations. A change in 80 mV on the input to the divider causes an increase in the count by 1, determining the resolution. An ability to differentiate between values just 0.1 V apart proves this experimentally. The equations leave the data points at 0 V out, since the ADC count hits a minimum at 10 or 11, unable to reach 0. Ultimately, this minimally impacts the overall system, since the voltage should never reach below 5 V, and, if it does, the circuit still operates the same.

5.3.2 CURRENT SENSING CIRCUIT DESIGN AND INITIAL TESTING

The current sensing circuit, shown in Figure 13, must not dissipate large amounts of power through the sense resistor, so a 1 mΩ resistance offers this advantage. However, the voltage across this resistor even at the max current of about 5 A only reaches 5 mV. Due to this

small sensing voltage, a preamplifier or gain stage must increase this voltage. Considering the maximum voltage of 65 V, the LT6101HV current sensing chip meets the requirements. The input voltage range of 5 to 100 V exceeds the requirement. This integrated circuit allows for a gain of 100 with a Rin value of 100 Ω and Rout of 10 kΩ. The current readings have shown spikes up to 80 A, despite no part that can handle or source 80 A, suggesting the discrepancy arose from measurement error [9]. The capacitor from Vout to ground produces a cutoff frequency of 1.32 kHz to match the bandwidth of the voltage sensing circuit. The circuit shown in Figure 13 only utilizes about 60% of the ADC range at a maximum of 7 A, but the extra room ensures accurate current spike measurements for future mitigation with the neural network and other circuitry. Additionally, soldering the sense resistor, although nominally 1 mΩ, adds contact resistance and increases this value. Vin comes from the DC-DC converter or input protection circuit, while maintaining R_LOAD of 10 Ω to the elliptical preserves identical workout conditions to the user. Vout_I connects to the ADC in the Atmel microcontroller. The voltage followers attempted to fix problems during testing, but after failing to fix the specific problem, desoldering the circuit seemed a waste of time. Future projects can skip those components if desired.

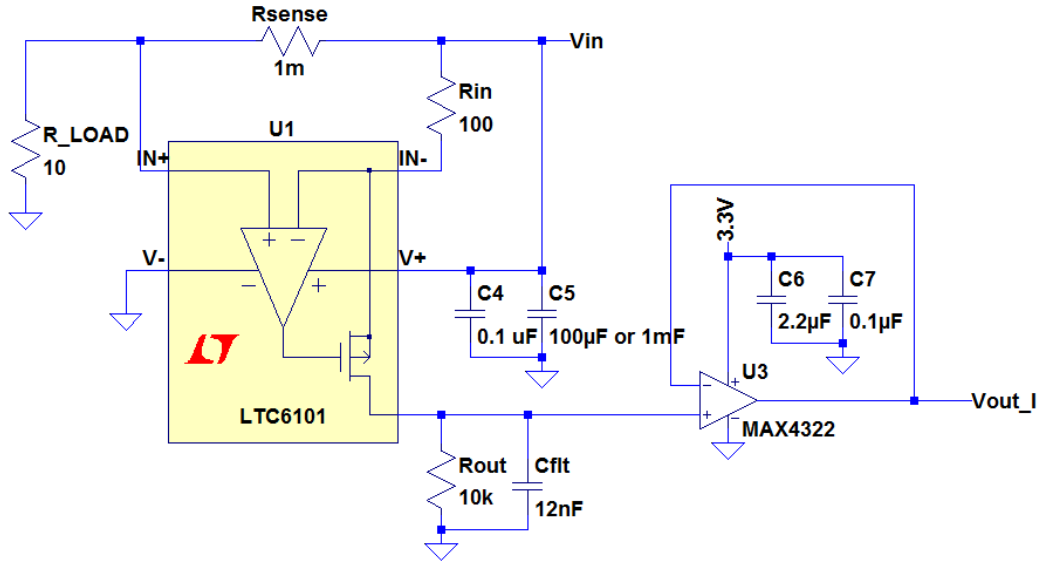

Figure 13. Current Sensing Circuit

The first current sensing circuit tests use a power supply to provide the voltage on the input and a 10 Ω resistor rated at 300 W. This resistor can replace the two 20 Ω resistors in

parallel in the EHFEM system, while testing the current sensing circuit to 5.1 A. This correctly produces a linear relationship between the current and voltage output from the sensing circuit as shown in Figure 14. As the graph shows, the resistance of each circuit appears slightly different, likely due to soldering the sense resistor on the second circuit upside down. This exposes the pads instead of connecting through the solder, decreasing the resistance since the current effectively travels a shorter distance between the ends of the resistor.

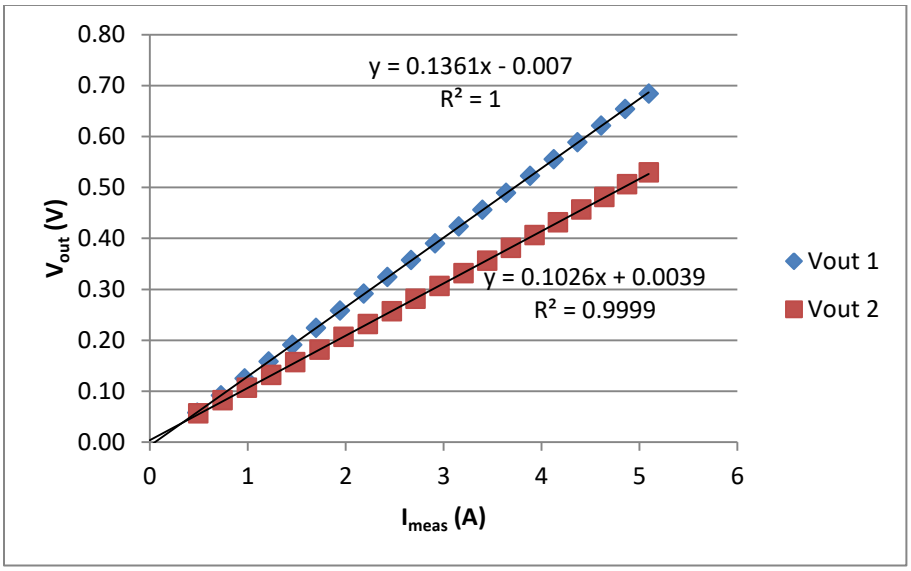

Figure 14. Output Voltage from LT6101 versus Measured Current

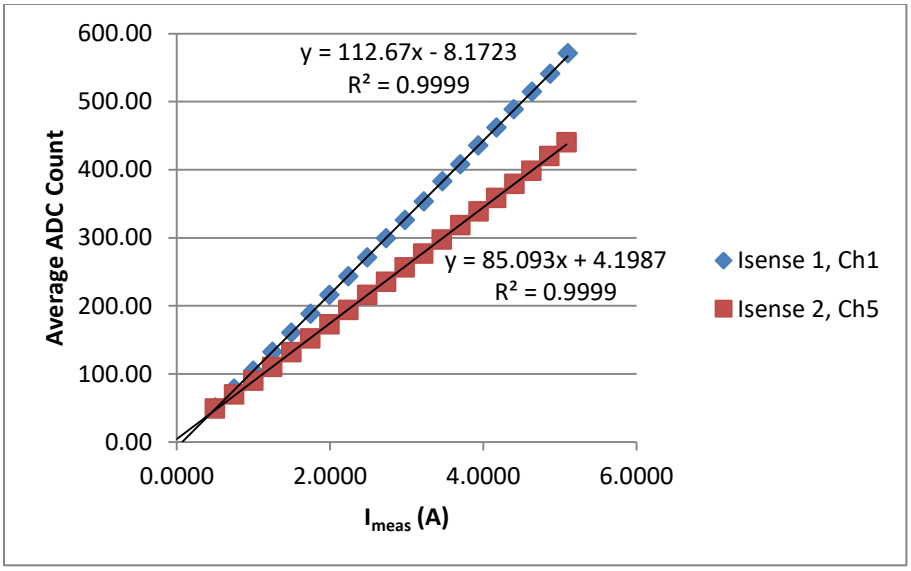

Figure 15. ADC Count Read versus Measured Current

Figure 15 then relates the current to the ADC count. The trend shows almost identical differences in slope for each line, proving consistent test data across two different test sessions. This test omits a resolution test of the current sense due to heat rapidly changing current levels with the 10 Ω test resistor. The data holds true, since the microcontroller takes ADC counts close together, and the measured current is recorded simultaneously to the ADC counts.

5.4 Design to Prevent Current Overloading

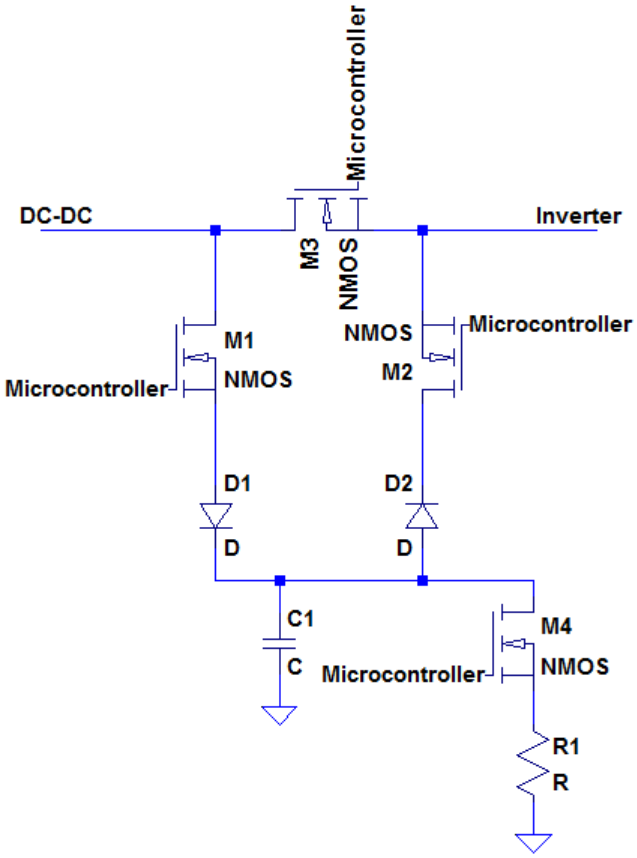

Figure 16. Harness Extra Current Schematic

The EHFEM system experiences currents above accepted limits by the DC-DC converter. Instead of dissipating excess power as heat in standard current limiting circuits, energy from excess currents could get harnessed and output when the input power reduces. Figure 16 gives one suggestion for what a current harnessing system might look like. Admittedly, overloading the capacitor presents some risk. To prevent this, another system, very similar to the capacitor system, connects in cascade to allow the microcontroller to bleed off some voltage from

the capacitor. This secondary system uses a resistor to dissipate excess energy as heat. Ideally, the cascaded system never engages, but provides a safety net.

Upon this initial brainstorm, this project leaves this circuit design for future work, leading to Alec Robertson's in-progress thesis project [25].

5.5 Low Pass Filter Design

Low pass filters convert the output of the microcontroller PWM outputs to constant voltages. The PWM outputs from the Atmel microcontroller at 120 MHz from 0 to 3.3 V as stated previously in Table 6. The bandwidth attenuates the PWM frequency and produces a DC value that can change quickly. The low pass filter achieves a bandwidth of 10 kHz with a capacitance of 1.6 nF and a resistance of 10 kΩ according to equation (5.1). The closest 10% capacitor value of 1.8 nF causes a slight decrease in bandwidth to 8.84 kHz.

$$f_{3dB} = \frac{1}{2\pi \times R_{LPF} \times C_{LPF}} \tag{5.1}$$

This low pass produces a DC voltage on the voltage feedback pin of the LT8705, which also connects to the voltage divider. The voltage divider typically provides the only feedback source, but connecting a low pass filter through a PWM output allows the microcontroller to alter the feedback. This method to connect the two circuits uses an isolation resistor to prevent the capacitor in the low pass filter from excessively interfering with the feedback. The original thought entailed sizing this resistor to enable the microcontroller to swing the output voltage up or down even in the worst case conditions. After some experimental tests, higher resistance provided higher precision control since the output voltage changed in smaller increments. The original 5.6 kΩ resistor increased to 105.6 kΩ for steady state testing to increase input voltage range on the DC-DC converter and neural network combination. This larger resistance limits the swing to 0.5 V on V_FB, which ranges from 0 to 2.5 V with the 5.6 kΩ resistor. During tests with the microinverter, the higher isolation resistance does not pass enough voltage to successfully control the DC-DC converter. For these tests, the originally calculated 5.6 kΩ resistor kept the microinverter powered on. Figure 17 shows the circuit where V1 represents the PWM output, V_OUT comes from the output of the DC-DC converter, and V_FB connects to the feedback node on the converter.

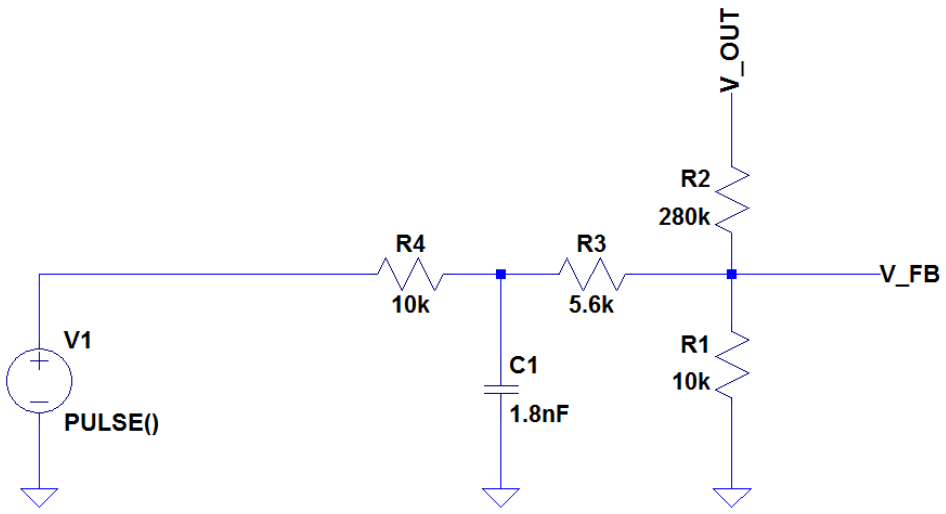

Figure 17. Connection between PWM Low-Pass Filter and Feedback Resistive Divider

5.6 Inverter Characteristics

The inverter sometimes behaves predictably. The inverter Maximum Power Point Tracking (MPPT) algorithms lower voltage and raise current, until they achieve the maximum power. This behavior normally benefits energy harvesting; however, lowering the voltage causes the DC-DC converter to run extremely inefficiently. The control system must give precedence to the DC-DC converter and fight any changes in voltage imposed by the inverter. Moreover, a drop below 22 V on the DC-DC converter output voltage node due to the microinverter MPPT turns the microinverter off. The control system keeps the DC-DC converter regulation steady to keep the system harvesting energy more consistently.

5.7 Feedforward Artificial Neural Network

A feedforward ANN designed for the DC-DC converter adjusts the duty cycle, improving performance characteristics. The inputs to the neural network include the input voltage, input current, difference between output voltage and nominal output voltage, and the load current. Most applications of neural networks to DC-DC converters directly control the duty cycle of the converter. However, an LT8705 controller chip contains many needed features, such as current protection. Therefore the implementation maintains functionality of this controller, which contains a feedback loop, and adds a learning aspect. The method ideally breaks the feedback loop, where a microcontroller measures the required values, but this thesis project opts to modify the

feedback voltage as previously discussed. An ANN then implements a feedforward path to adjust what the controller sees on the feedback pin. The feedback pin has a reference of 1.208 V. If the voltage on that pin exceeds the reference voltage, then the duty cycle adjusts based on Table 11. With similar voltages, the neural network does not act and outputs a reference voltage at 1.208 V. When the voltages deviate, the neural network controls the feedback pin to determine the appropriate voltage level.

5.7.1 ANN BACKGROUND INFO AND LEARNING DISCUSSION

Neural networks update the weights between each neuron with different types of learning. Two types of learning exist: supervised learning or learning with a teacher and unsupervised learning or learning without a teacher [26]. The neural network implementation adapts to varying settings on the elliptical trainer. Within the category of supervised learning lie many different algorithms, and each algorithm contains many variations. The main group of algorithms directly updates the weights of the neuron connections. Learning with a critic or reinforcement learning, an unsupervised learning method, supplies a reward or punishment to the ANN for an output within the specified range or not. The ANN then optimizes the weights to either maximize the rewards or minimize the punishments. This method could prove viable with some modification or precisely chosen parameters, but typically this learning slowly improves over time. A block diagram of reinforcement learning, Figure 18, shows the environment and the learning system inside the feedback loop.

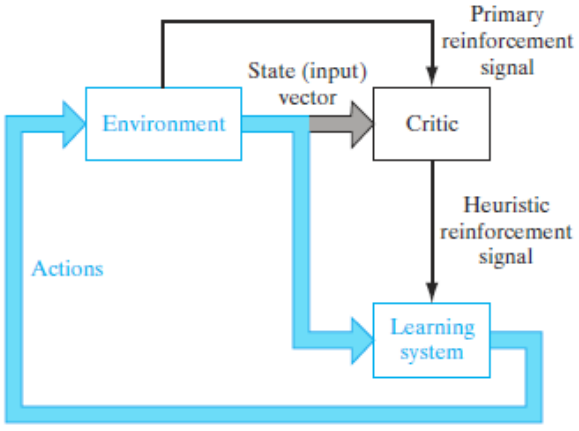

Figure 18. Reinforcement Learning Block Diagram [26]

Returning to supervised learning, the block diagram in Figure 19 shows how to place a learning system parallel to its teacher, and the error of the learning system updates the system modeling. Different algorithms use this error signal in different ways.

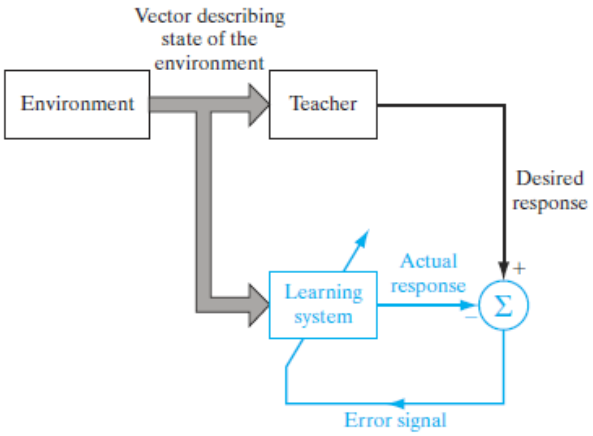

Figure 19. Supervised Learning Block Diagram [26]

Neural networks in this paper employ supervised learning to update the actual response. First, learning starts with a system randomly initializing conditions. The error signal is used to update the weights with a predefined algorithm, until the neural network outputs match the desired system outputs. Then, weights do not update until error exceeds the defined threshold.

5.7.2 BACK-PROPAGATION ALGORITHM AND LEVENBERG-MARQUARDT METHOD

In several other instances, researchers choose to use back-propagation algorithms to regulate constant voltages [12]-[14]. However, these do not include buck-boost topologies, but rather only buck converters. Studying the methods in these sources helps expand them to a buck-boost topology. One buck-boost topology, the flyback converter, produces good results from neural network control in one instance [15]. The downside to this topology includes lower efficiencies than with a four-switch buck/boost topology. This info points towards implementing an ANN to control the four-switch buck/boost topology. The back-propagation algorithm is a training algorithm for a feedforward neural network. It updates the weights backwards based on the error signals; the actual neurons only have forward connections. Figure 20 illustrates the direction of the error signals and neuron connections for an arbitrary, but simple neural network.

Neural networks easily map nonlinearities. To map nonlinear functions, the perceptron network must contain multiple layers. The network contains at least one hidden layer that helps map nonlinear patterns to a different space where they become linearly separable. The DC-DC converter's nonlinearities require multilayer perceptron networks.

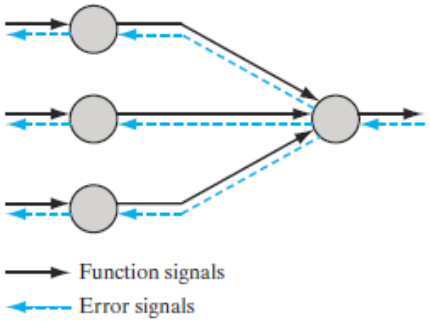

Figure 20. Back-Propagation Neural Network Example [26]

Equation 5.2 shows the general equation to update connection weights, with neuron connections going from j to i. The symbol w_{ij} then corresponds to the connection weight between neuron j and i. The symbol $\Delta w(n)$ corresponds to the value the error signal back propagates to neuron n. The symbol η, the learning rate, is usually a constant value chosen between 0 and 1.

$$w_{ij}(n+1) = w_{ij}(n) + \eta \Delta w(n) \tag{5.2}$$

The back-propagation method may converge slowly by itself. To increase the convergence speed, we use the Levenberg-Marquardt Method included as Equation 5.3 [12]. This method requires taking the inverse of a matrix. For the function this algorithm tries to optimize, g(n) corresponds to its gradient vector and H corresponds to its Hessian matrix. The symbol I represents the identity matrix of the same dimensions as H and λ ensures that the sum [H + λI] is positive and definite [26].One problem with this method relates to the inverse matrix calculation speed. If the matrix becomes larger than 5x5, then calculation time could dramatically increase.

$$\Delta w(n) = [H + \lambda I]^{-1} g(n) \tag{5.3}$$

The back-propagation algorithm uses the chain rule from calculus to find the gradient vector g(n) in Equation 5.4. The symbol ε refers to the cost function, included in Equation 5.5 [26]. The symbol e denotes the error signal for neuron n.

$$g(n) = -\frac{\partial \varepsilon}{\partial w_{ij}} \tag{5.4}$$

$$\varepsilon = \frac{1}{2} e^2(n) \tag{5.5}$$

5.7.3 NEURAL NETWORK STRUCTURE

The optimal structure of an ANN is never obvious. Many beginners in the field start by changing the number of neurons in each layer by trial and error. Certain models may only work for very specific scenarios. Having too many neurons may reduce the calculation efficiency.

First, start with the input. In [13], input voltage V_{in}, load current I_L, and deviation on output voltage $[V_{out}(t)-V_{out}(t-1)]$ each account for one input neuron, used in this structure as well. Since that DC-DC converter only steps down voltages, these inputs allow good regulation.

To expand this to a buck/boost converter, the neural network needs two more inputs to detect the operation region of the converter. Examining Table 11, transistor M1 turns on in boost mode and M4 when in buck mode. When in the middle region, we want the neural network to do nothing, so the weights do not adjust, and the voltage keeps adjusting in the same direction. We move forward with the assumption that it passes through this region quickly, and, in this region, the control switches between each mode quickly counteracting the effects of each. Additionally, the mode should not matter, so the neural network ignores it.

Instead, I add two other inputs, measuring the input current to the DC-DC converter I_{in} and the input current to the inverter I_{inv}. The I_{inv} current differs from the current out of the DC-DC Converter I_L by the capacitor current between the two. If the inverter tries increasing the current, it can pull current from the capacitor, which then causes the inverter to increase the current further, since the power reads higher at that current level. This forms a positive feedback loop, where current increases until the capacitor cannot source any current or until reaching the DC-DC Converter's current limit. To keep the DC-DC Converter's current limit below the absolute maximum, the neural network limits it. The input current I_{in} can help the neural network infer the output current, to maintain power in about equal to power out, while taking varying efficiency into account.

The network also needs one hidden layer, which allows simpler calculations than having two hidden layers. The network only needs two layers, if one layer does not provide acceptable results. This hidden layer contains 10 neurons, a half arbitrary and half guessed number from typical neuron numbers in other systems. These neurons then feed to the duty cycle of the transistors in normal applications. In our case, the output neuron feeds to the feedback voltage pin. This neuron correlates to a range of voltages from 0 to 2.5 V after some post processing, which uses the opposite calculation when normalizing the inputs. The feedback strength increases as the voltage moves away from the 1.208 V threshold voltage of the LT8705 converter chip. The proposed neural network architecture, shown in Figure 21, fits within the overall block diagram in the level 1 diagram.

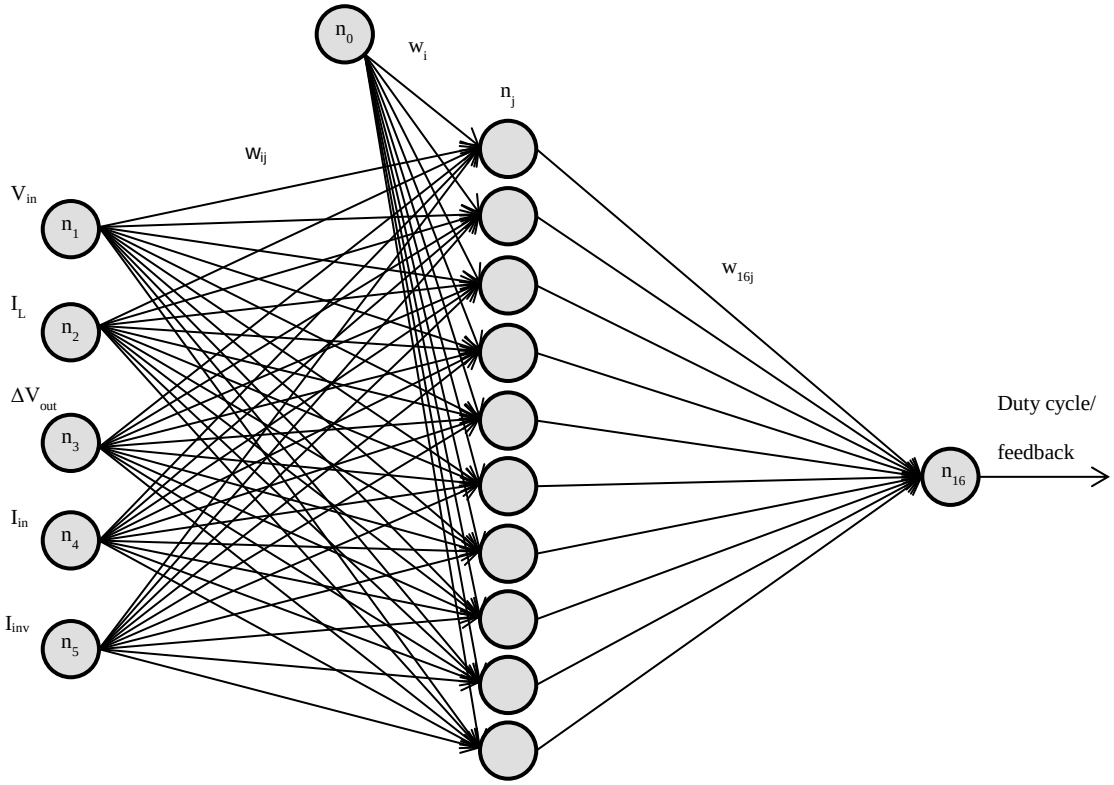

Figure 21. Initial Neural Network Structure

I abandoned the initial structure, since measuring the current before the DC-DC converter's output capacitance required cutting traces to insert a sense resistor. The revised neural network in Figure 22 reflects this change. Experimental data determines the number of

hidden neurons, eliminating arbitrary estimation. The hidden layer contains 18 neurons for fastest calculation speed and least error.

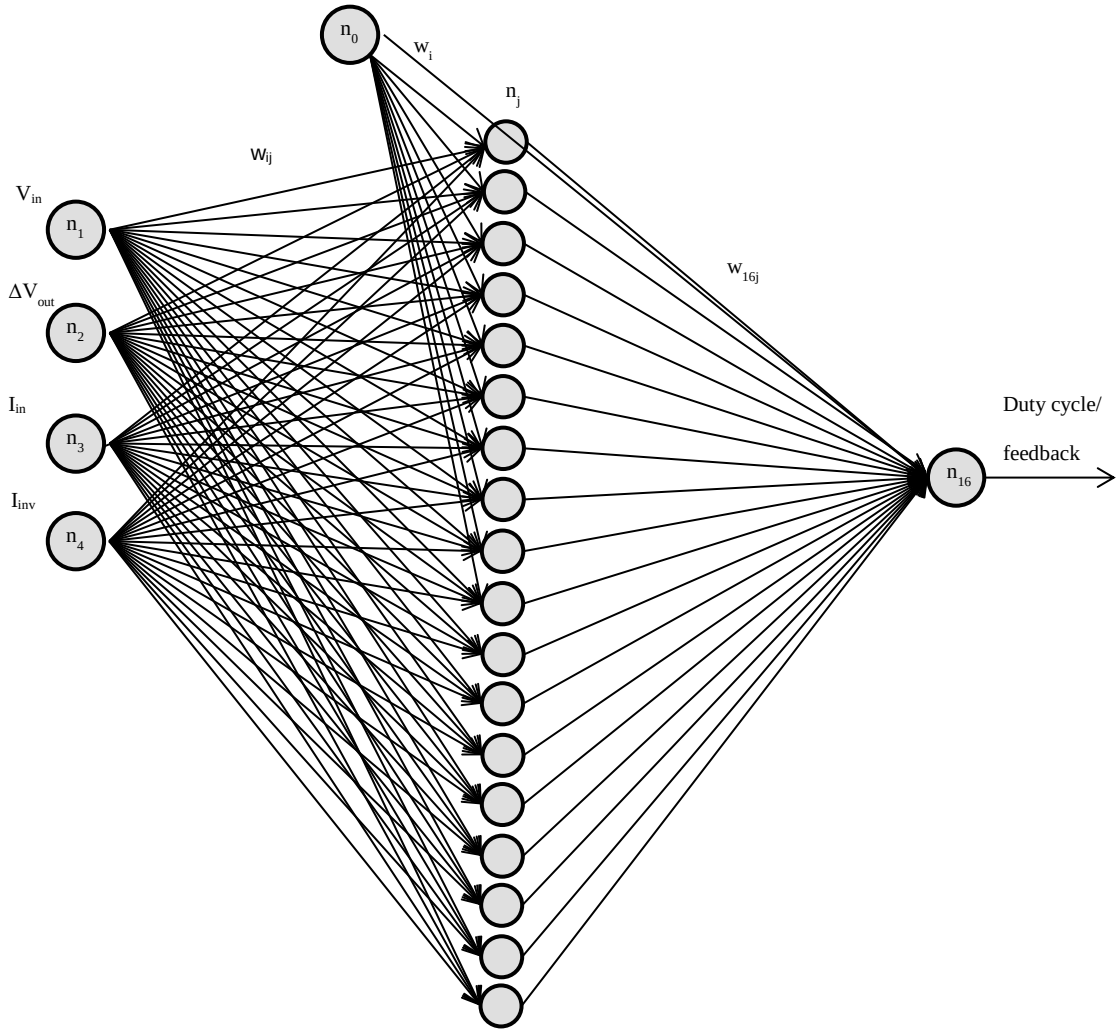

Figure 22. Final Neural Network Structure – 18 Hidden Neurons

5.7.4 OUTPUTS

In this topology, several options exist for the output neuron depending on the control method. To decide the control method, one must have a comprehensive understanding of the entire system. The elliptical trainer and input protection circuitry gives the DC-DC Converter some input power. The converter then aims to modify the output power, by keeping either output current or voltage constant. In our specific application, either could work. The inverter adjusts the output voltage and current to achieve its maximum power point, known to pull voltage lower to increase the current in nominal conditions. Therefore, the control limits the currents from rising above their

absolute maximum ratings. The maximum current out of the converter must not exceed 7 A and the voltage must not go outside the 5-60 V range previously determined for the input voltage. To prevent overshooting the 7 A maximum, the neural network keeps the current below 5 A to allow for overshoot and maintain the DC-DC converter's optimal current range. This analysis focuses on current limits more than voltage, making a case for current control. Additional reason to use current control comes from the Enphase solar panel inverter used.

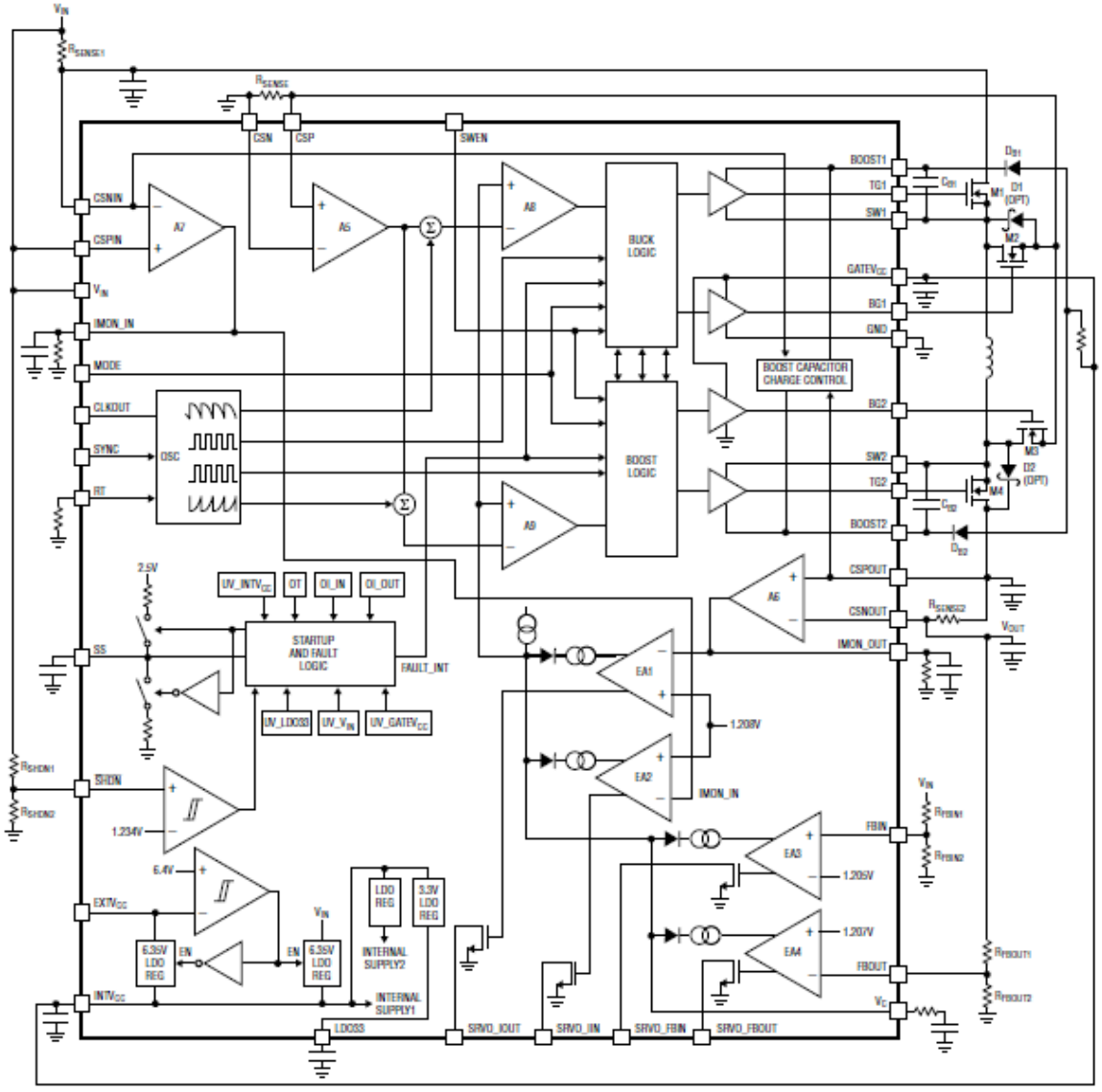

Figure 23. LT8705 Block Diagram [23]

Enphase designed the M215 inverter for use in photovoltaic (PV) systems. PV systems normally expect nearly constant current inputs, since solar panels output DC current. The inverter

may raise or lower the voltage to seek higher current and power. In this application with the DC-DC converter, lowering voltage can cause an excessive overcurrent. So, we must create a feedback control to prevent the inverter from receiving too much power as it decreases current. The method to do this drops the voltage across the inductor, decreasing the current output. The LT8705 chip allows for this control in the IMON_OUT pin. When the voltage on this pin rises above 1.208 V, the voltage pin Vc drops, limiting the current through the inductor. Therefore, the algorithms train the neural network to limit this current when the output current I_L differs from the input current to the inverter, I_{inv} by more than 15%. This prevents large amounts of current from the capacitance between the DC-DC Converter and inverter adding into the inverter when it tries to increase current. If it increases current and gets significantly more power, then it continues to do so until the DC-DC Converter can no longer handle the current. By throttling the current through the DC-DC converter, the inverter stops increasing current since power starts decreasing. The neural network also reduces output current, if it reaches more than about 6.5 A, under the 7 A absolute maximum.

Now we have two instances that decrease the DC-DC Converter's output current: 1. if the output current approaches the absolute maximum and 2. if the inverter's current receives too much current from the DC-DC Converter's output capacitance. Both types of control require current control, which the LT8705 chip allows. In addition, the converter may need voltage control, depending if the output voltage regulates near the nominal 36 V output. The LT8705 chip allows for this control as well in a separate node. In an effort to reduce hardware, the network may combine the logic of these two connections into one. Since the LT8705 datasheet shows all nodes connected to its own buck/boost logic box, it may not matter which node sends an overcurrent or overvoltage signal, as shown in Figure 23. Since the datasheet does not clarify this, hardware must test it and determine the results.

5.7.5 NETWORK TRAINING DATA

Neural networks need some set of data to train the connections to achieve the desired performance. Training data must not get confused with the network verification data, described in the next section. The network training data contain input-output pairs of data.

These data work best when all data have similar ranges. The data collected do not initially have the same ranges, but normalization fixes this discrepancy. A few basic ways exist to normalize data. Since the transfer function of the activation layer uses the hyperbolic tangent sigmoid function, the data should span from negative one to one. One method centers the data in this range on the mean, another way centers on the median and the chosen method linearly scales all values according to the minimum and maximum values [26].

5.7.6 NETWORK VALIDATION DATA

Network validation data shows if the network trains sufficiently. The input neurons all receive realistic values across some representative data set, and the user compares the neural network's outputs against their expectations. This does not show that the overall feedback system works, just that the training data trained the neural network as intended. Usually, the training data contains some subset of about 70 to 80% of all data; while validation contains 10 to 15% and a test set consist of about the same size. In this application, testing obtains new data on the experimental hardware. Training and validation datasets do not differ in collection methods.

Chapter 6. ANN Offline Training

Chapter 6 focuses on artificial neural network (ANN) offline training. Offline training refers to training the network with a set of previously collected data points. This teaches the artificial network to work perfectly on that data, and interpolates those points for untrained data when used in the real system. The real system uses online training to improve its response.

6.1 Introduction to Offline Training

Chapter 5.7 describes offline training data sets, neural network structure, and other issues from the perspective of initial design. This structure slightly changes throughout later chapters for possible or necessary performance improvements.

6.2 ANN Implementation Using MATLAB Neural Network Toolbox

First, the quickest approach implements the neural network, which uses preexisting toolboxes and functions in MATLAB software packages. This creates issues, however, since debugging or examining individual functions becomes difficult. The issues arise when the preexisting functions do not behave as expected or desired; therefore, this approach only helps approach the right solution. Additionally, these first tests only include load currents of 1 and 2 A, as the rest falls into place once these work correctly and avoids special difficult cases for the start.

6.2.1 MATLAB CODE

The code in MATLAB modifies a previous homework assignment employing an ANN. The code specifies a neural network with five input neurons, a hidden layer of a certain number of neurons, and one output neuron. Varying the number of hidden layer neurons determines the optimal layer size, but for now assume the layer must contain a minimum of 5 neurons, corresponding to the number of input neurons, and a maximum of some large number of neurons. The next sections show that this upper limit approaches 30 neurons, whereas some networks may employ much larger hidden layers.

The first two blocks of code included in Appendix B define the neural network. The number of neurons in each layer, the training algorithm, and activation functions all affect the performance. Chapter 5 discusses the number of neurons in each layer and training algorithms.

This code allows changes in the activation functions. This project only considers two activation functions. This project first considers the default activation functions, the hyperbolic tangential sigmoid function ‘tansig’ for the hidden layer and the linear function for the output neuron. These seem to produce the most accurate results. The other option changes the output neuron layer to the logarithmic sigmoid function ‘logsig’ since the network outputs either 0 or 1. This activation function increases the mean squared error (MSE). The increase in MSE should make the ANN run additional epochs, but the network does not seem to run as many epochs as expected. Later sections discuss possible reasons. The network randomly initializes bias and weights of each neuron, following standard procedure.

Next, I train and test the neural network, and then compare the neural network’s output to the desired output. Two different plots appear. The first plot shows the neural network properly initialized and its untrained output, while the second plot shows the trained output overlaid on the desired output. When training the neural network, the code sets the RMSE of the error threshold, so after training the actual RMSE is examined to ensure proper training. Sometimes, the error greatly exceeds the value set, so this serves as a good check on the training algorithms.

6.2.2 PERFORMANCE WITHOUT NORMALIZED DATA

Theoretically, no data needs normalizing since the network changes its weights appropriately. In practice, this usually does not hold true. Regardless, examining the response before normalizing the data helps get a sense of the network.

Because the network does not train to small MSE values as desired, no validation data set is used for these. The same training data validates the network to illustrate its training.

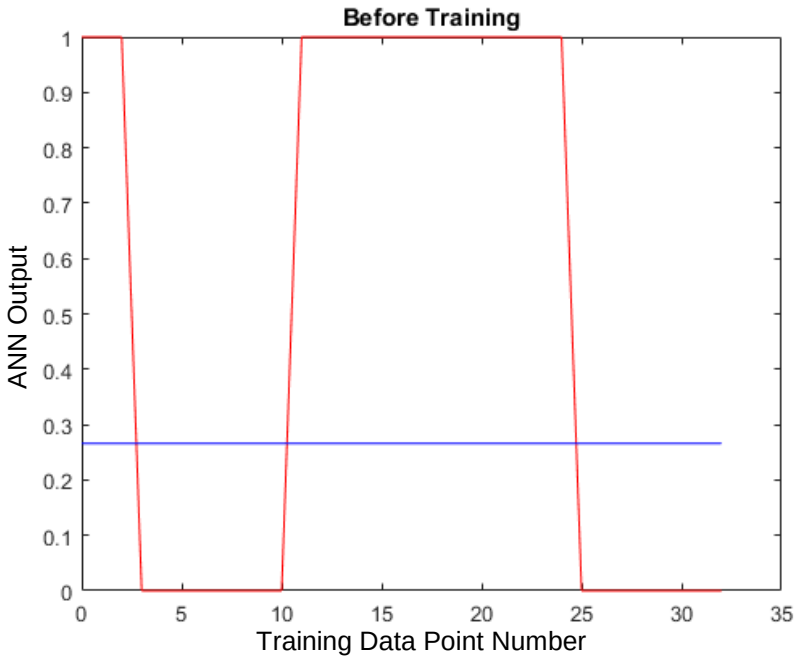

Figure 24. Initial Output with Neural Network Toolbox, Example 1 – Ideal Output of ANN (Red), Actual Output of ANN (Blue)

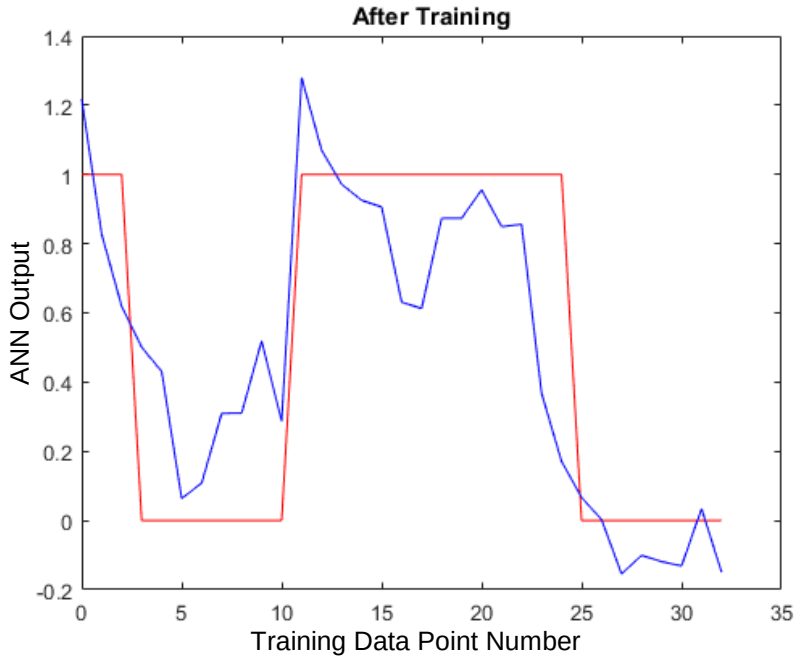

Figure 25. Trained Output with Neural Network Toolbox, Example 1 – Ideal Output of ANN (Red), Actual Output of ANN (Blue)

In the first example, the network initializes to Figure 24 as shown with the flat blue line.

The network goes through 7 iterations, and appears as shown in Figure 25 with the jagged blue

trend. In this run the MSE of 0.0867 exceeds the set goal of 0. The MSE typically is set to a small number, but setting it at 0 results in the best response for the meantime.

In a separate run, the neurons initialize differently as shown in Figure 26 and the network trains in a different manner. In this case, the MSE of just 0.007 after 18 iterations comes close to the training data shown in Figure 27. This example shows the correlation between smaller MSE values and increased resemblance of the network to the training data. This notably gives a strong case that once the network actually trains to the specified MSE as intended, the network classifies inputs properly.

These two examples summarize the endpoints of the network behavior, with many results in the middle. At this point, not using normalized data appears to throw the standard MATLAB functions off since they expect similar ranges in each data set. The next test normalizes the data to see if the neural network toolbox learns correctly.

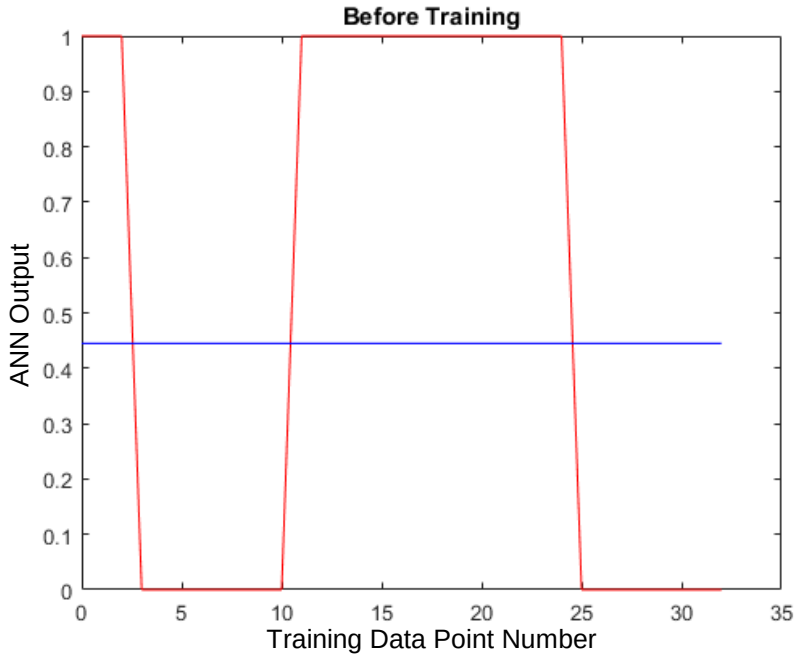

Figure 26. Initial Output with Neural Network Toolbox, Example 2 – Ideal Output of ANN (Red), Actual Output of ANN (Blue)

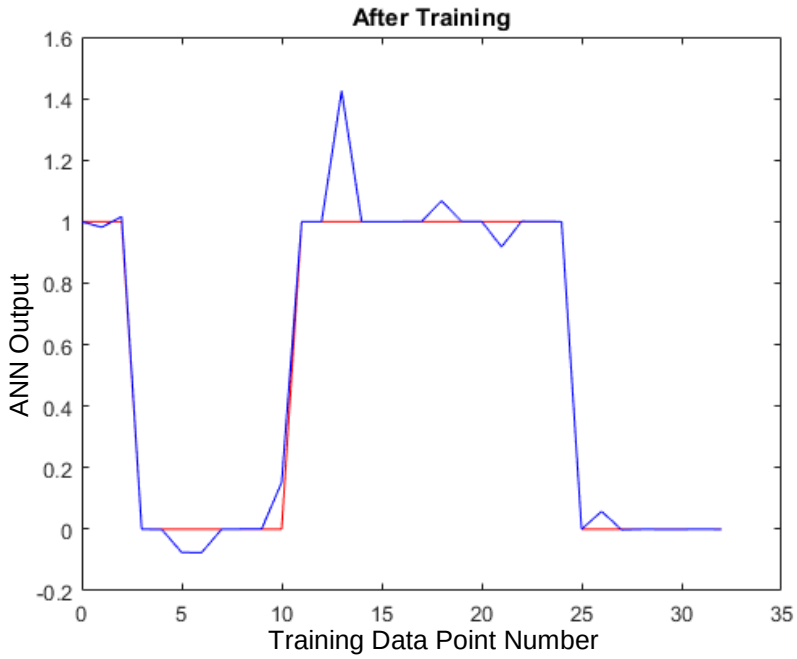

Figure 27. Trained Output with Neural Network Toolbox, Example 2 – Ideal Output of ANN (Red), Actual Output of ANN (Blue)

6.2.3 PERFORMANCE WITH NORMALIZED DATA

This simulation normalizes the data by linearly scaling numbers down to fit within the desired range. This entails finding the minimum, maximum, and using equation 6.1 to find the normalized data points.

$$x_{normalized} = \frac{x - x_{min}}{0.5 \cdot (x_{max} - x_{min})} - 1 \tag{6.1}$$

Using normalized data in the MATLAB toolbox did not give different responses than previously non-normalized data, so the next step develops lower level MATLAB code. Future neural network implementations use the normalized data.

6.3 ANN Implementation Using Low Level MATLAB Code

Low level MATLAB code allows full control over the learning algorithm. MATLAB's neural network toolbox works great theoretically, but when experiencing inadequate responses, no way exists to investigate where the issues arise. The starting point for my back propagation learning algorithm adapts an online blog post [27]. Using pre-existing code as a starting point saves time and reduces the number of errors. I proofread every line of their code and compared its functionality to the equations to ensure it operates properly. I learned part-way through early

simulations of one incorrect equation. I also replace the sigmoid activation function with a hyperbolic tangent function since the initial simulations show this produces adequate results and converges faster.

The first step tests the code to see if it works. Teaching the neural network the XOR function serves as a preliminary test. This function contains desirable features including its simplicity, small dataset, and is linearly inseparable. This simple function verifies important capabilities essential for the intended application. The output neuron ideally output 0, 1, 1, and 0 to mimic the XOR function.

The next step adds an iteration stop limit to the code. This calculates the error after every iteration of the learning algorithm and stops, if the error becomes less than the desired error threshold. Appendix F contains the earliest simulation iterations discovering coding errors and training the XOR function to test the code. The next section includes optimization using the final low level MATLAB code.

6.4 Determining Initial Connection Weights for Steady State Tests Using Offline Training in MATLAB

To use the neural network on hardware, it needs to start with weights close to their ideal values. These simulations determine those initial weights. These tests prove the neural network functions properly for steady state tests, replacing the static feedback with the neural network control. Passing this test enables more complex control for dynamic tests with the microinverter in the next section. This method of control directly translates to one operation condition of the dynamic tests, providing an integral building block. Therefore, finding the minimum hidden neuron layer size for this test provides a minimum for future tests.

These tests differ from early tests found in Appendix F after determining the feedback pin of the LT8705 chip acts as proportional gain instead of a comparator as previously assumed.

6.4.1 RESULTS FROM 16 HIDDEN NEURONS

Optimization starts with 16 neurons after all previous simulations hit the iteration limit of 40,000 every time. Fewer iterations from the randomly initialized weights correlates to the time the neural network takes to correct itself in the microcontroller implementation, so minimization

improves system performance. The RMSE of the error equaled 0.0772 after the last iteration, above the 0.07 threshold, so this simulation hit the iteration limit.

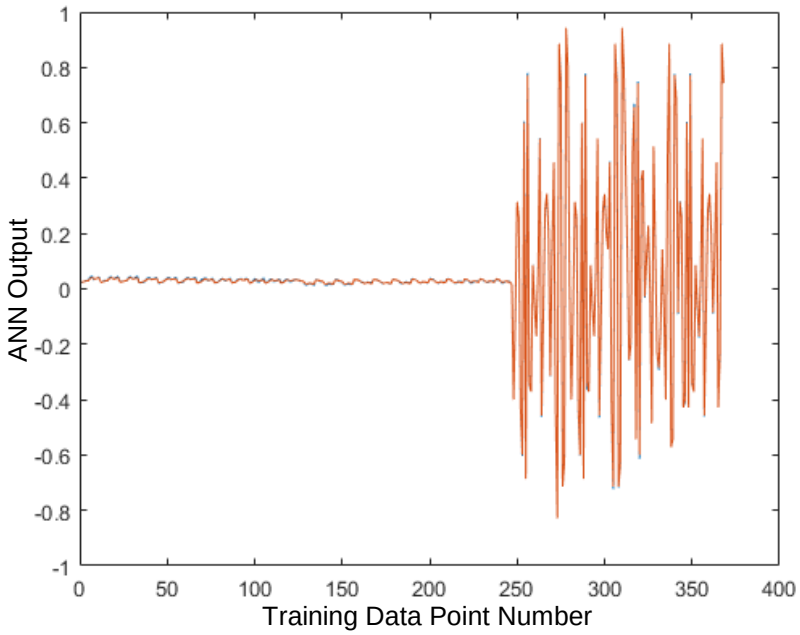

Figure 28. Training Data – Steady State Optimization 16 Neurons, Eta of 0.03 – Ideal Output of ANN (Red), Actual Output of ANN (Blue)

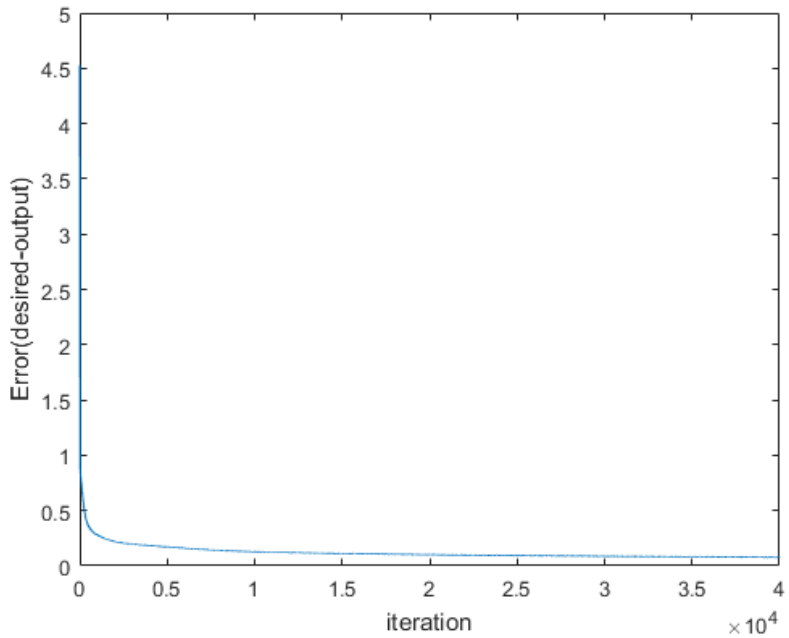

Figure 29. RMSE of Error after Each Training Data Iteration – Steady State Optimization 16 Neurons, Eta of 0.03

Figure 28 shows the network trained well, showing no obvious deviation from the expected and actual outputs, while Figure 29 shows the RMSE of the error decreasing with each iteration ran. The initial learning coefficient value of 0.03 undergoes optimization after optimizing the hidden neuron layer. The neural network fit the data nicely considering the validation data, data not part of the training dataset, produced a MSE of 0.0014 and shown in Figure 30.

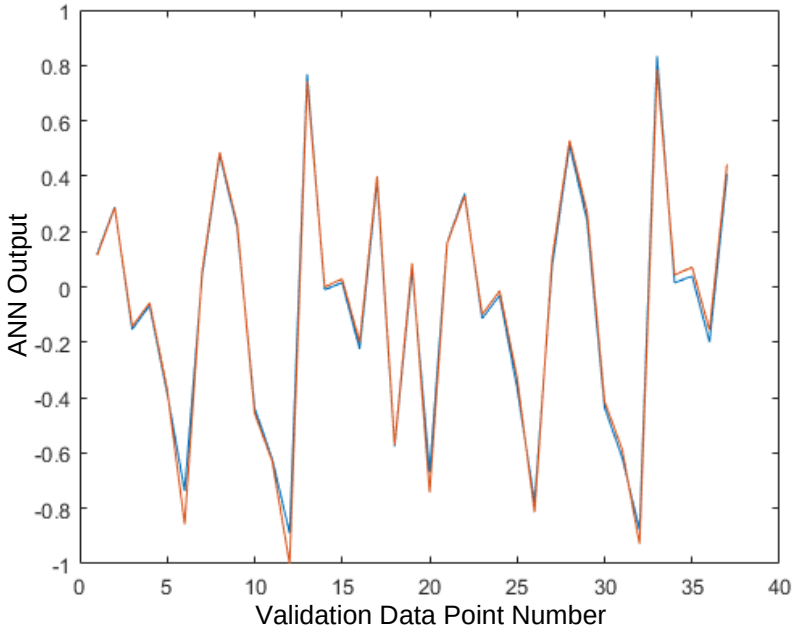

Figure 30. Network Validation Data – Steady State Optimization 16 Neurons, Eta of 0.03 – Ideal Output of ANN (Red), Actual Output of ANN (Blue)

In this test, a decrease in just one hidden neuron decreases accuracy. The plots show one example from the many runs of this test to ensure repeatability.

6.4.2 OPTIMIZED CASE EMPLOYING 18 HIDDEN NEURONS

After many tests adjusting the hidden neuron layer and the leaning rate, this test determines 18 hidden neurons as the optimal case. The number of iterations greatly decreases while the validation dataset error stays about the same at 0.0012. Figure 31 shows the network trained well and Figure 33 shows the performance on the validation data.

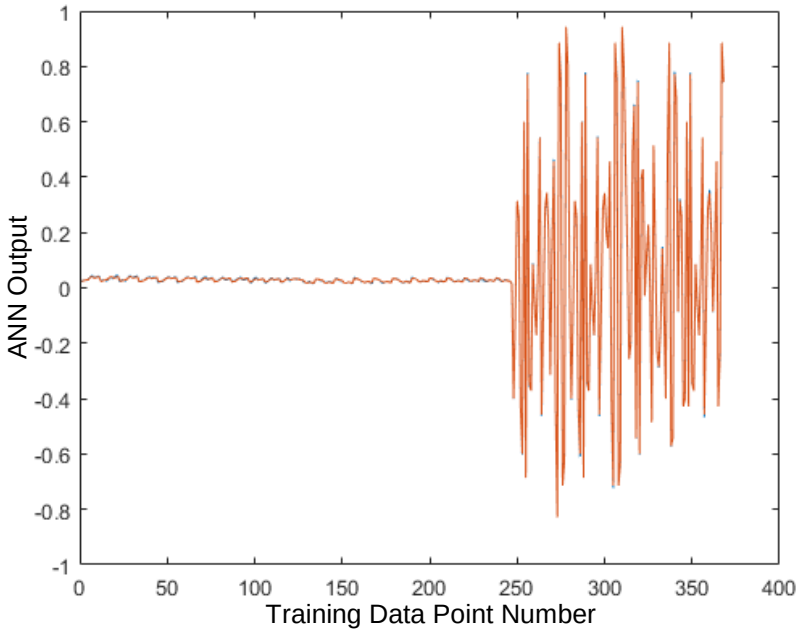

Figure 31. Training Data – Steady State Optimization 18 Neurons, Eta of 0.04 – Ideal Output of ANN (Red), Actual Output of ANN (Blue)

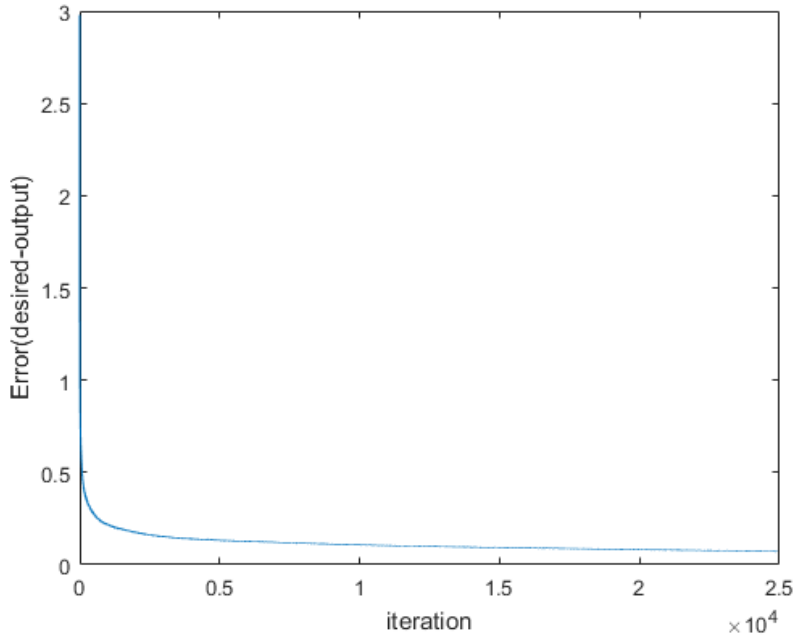

Figure 32. RMSE of Error after Each Training Data Iteration – Steady State Optimization 18 Neurons, Eta of 0.04

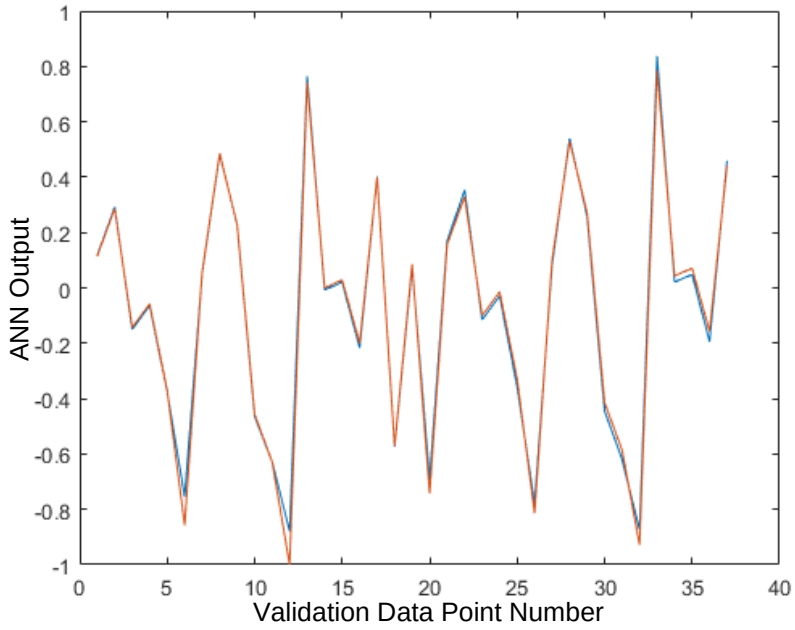

Figure 33. Network Validation Data – Steady State Optimization 18 Neurons, Eta of 0.04 – Ideal Output of ANN (Red), Actual Output of ANN (Blue)

After some optimization, this run shows a more optimal neural network for this application. With 18 hidden neurons, the validation data set shows a small MSE of 0.0014 and a number of iterations to achieve that performance less than most other runs. The number of iterations decreases with larger values of eta, in this case, 0.04 works well without causing validation data wellness of fit to decrease. Furthermore, setting the error threshold slightly higher would greatly decrease number of iterations shown by the error graph in Figure 32. However, during offline training a closer fit takes priority over fewer iterations as a better fit allows less error in the experimental run.

This stops optimization, since more neurons provide similar results, but decrease computational speed on the microcontroller.

Chapter 7. Hardware Testing

This chapter explains configuration of the microcontroller that connects to the voltage and current sensing circuits, which then implements the neural network to control the buck-boost converter.

7.1 Atmel SAM4S Xplained Pro Configuration

The SAM4S Xplained Pro evaluation board contains an interface to program the SAM4SD32C at the heart of the board. The AREF adjustment potentiometer in Figure 34 produces an AREF measurement of 2.6 V, after tuning, that the current amplifier output voltage range and AREF minimum voltage determine. Other features in Figure 34 and their functions include: extension header 1 (pins 3:4 ADC out Channels 0:1, pin 7 PWM output), extension header 2 (pins 3:4 ADC out Channels 4:5), power header (pin 2 ground reference, pin 4 3.3 V output voltage), and LED0 (UART test).

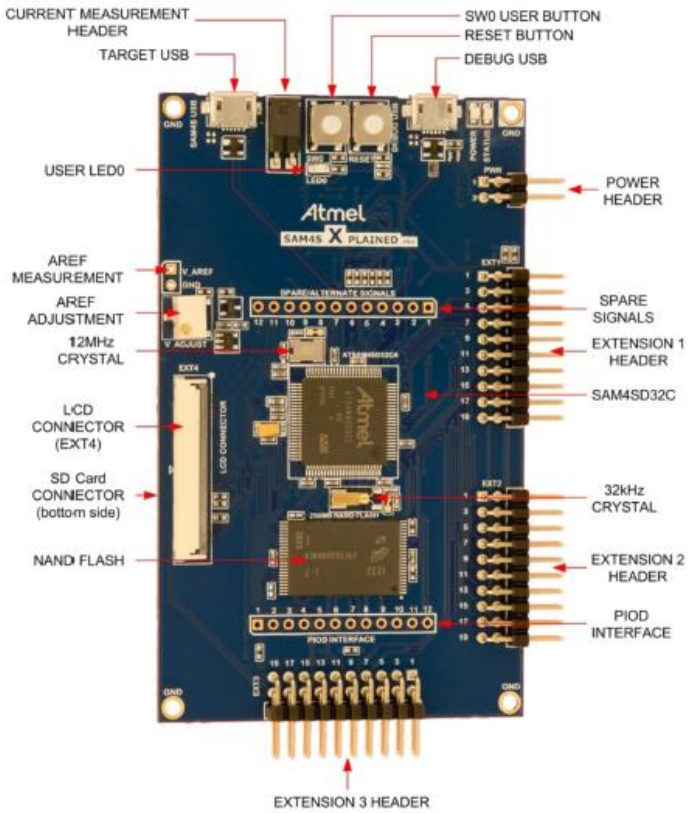

Figure 34. SAM4S Xplained Pro Evaluation Board (SAM4SD32C) [28]

7.2 Atmel Software Framework

This project uses Atmel Studio 6.2, since the website for this microcontroller directs to downloading version 6.2. Before discovering this, I tried using version 7, but after working initially it mysteriously stopped reading the correct ADC values after no code changes. This experience echoes the recommendations from Crivelli's senior project, which also provides the starting point for my code along with Funsten and Kiddoo's senior project [18], [29]. My code starts with running their code, then stripping it down to just the real time timer (RTT), ADC, UART, and general housekeeping code before adding in unique code. To get started in Atmel Studio 6.2, Crivelli's senior project report contains a great guide which I followed as well to change the required header file [29]. Atmel Studio contains a number of software modules, listed below, that need including to run the code in Appendices D and E.

ASF modules needed:

- Generic board support (driver)
- System Clock Control (service)
- Delay routines (service)
- GPIO – General Purpose Input/Output (service)
- IOPORT – General Purpose I/O service (service)
- USART – Serial interface (service)
- Standard serial I/O (stdio) (driver)
- ADC – Analog-to-Digital Converter (driver)
- DACC – Digital-to-Analog Converter (driver)
- PIO – Parallel Input/Output Controller (driver)
- PWM – Pulse Width Modulation (driver)
- RTT – Real Time Timer (driver)
- TC – Timer Counter (driver)
- WDT – Watchdog Timer (driver)

The UART responds to the commands in Table 12. The UART helps debug and test primarily, since it cannot output as fast as the microcontroller runs.

Table 12. UART Command List

Command	Response
‘a’	Toggles LED0, prints a to UART
‘p’	Prints all 4 ADC values to UART
‘f’	Data capture mode: only used to collect monitoring ADC data directly to terminal
‘r’	Runs neural network program
‘v’	Tests PWM output at 10% duty cycle, then 50%, then off
‘s’	Stores ADC readings (0.125 seconds of data max)
‘o’	Outputs stored data
‘m’	Captures median data
‘w’	Outputs median data to terminal

7.3 Atmel Studio and Microcontroller Quirks

Sometimes, programming the microcontroller may seem less than trivial. When the microcontroller fails to program new code, pressing the reset button on the SAM4S Xplained Pro evaluation board then reprogramming the device easily solves this issue.

Additionally, the usual warnings from Atmel Studio sometimes disappear without any reason. Upon creating an error in the code then fixing that error, the warnings reappear as expected. Without usual warnings, I do not have confidence that Atmel Studio built the project properly, although behaviors seemed unchanged.

7.4 Summary of Prototyping Board to Measure Voltages and Currents

The board created for this thesis attempted to employ color schemes for ease of use and male header pins to easily connect to any microcontroller male headers. After adding additional ground wires, the color code became less clear. Figure 35 and Table 13 show each header and protruding wire’s uses. Any headers not circled or labeled provided intermediate nodes used for testing purposes only. The green wires connect to ground along with the header at the almost exact center of the board. The Blue wires on the left side of the board connect to the positive current sense nodes, while the orange wires connect to the negative nodes. The header pin in the middle right allows easy connection of power from the 3.3 V voltage of the Atmel Xplained Pro evaluation board or an external power supply. The header pin on the top right connects to the

PWM output of the microcontroller to create a DC voltage on the wire on the right side of the prototyping board. The four light and dark turquoise circles around the ground header connect to the ADC pins of the microcontroller. The white wires on the top and bottom of the board connect to the voltages measured before and after the DC-DC converter.

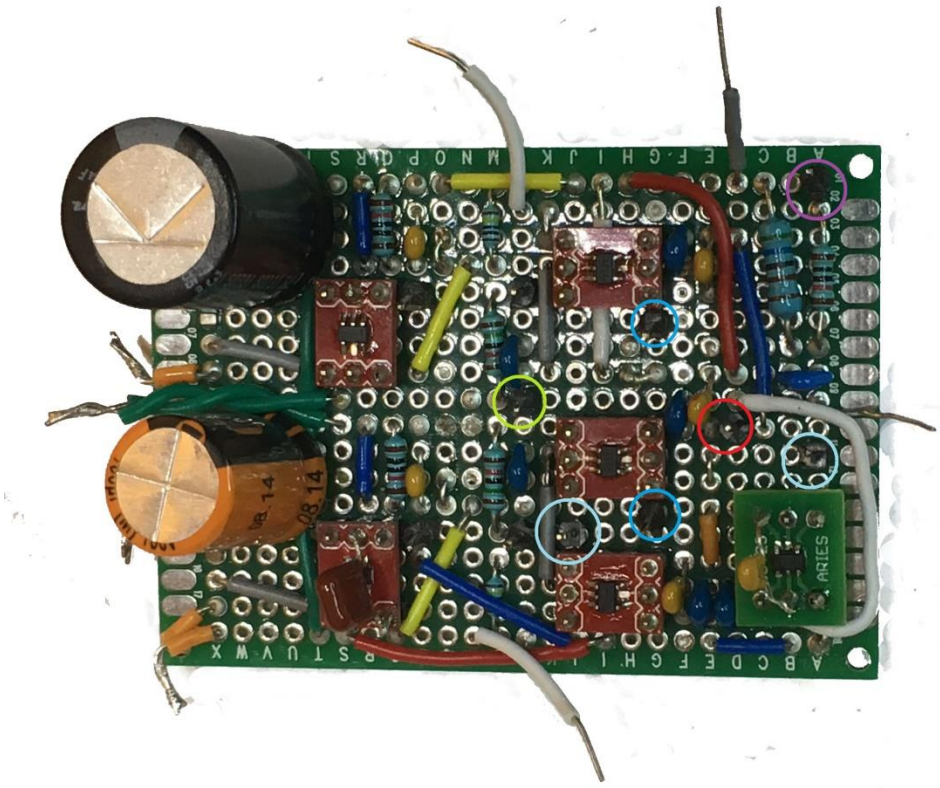

Figure 35. Prototyping Board with 2 Voltage and 2 Current Sense Circuits

Table 13. Pin Descriptions of Prototyping Board – 2 Voltage and 2 Current Sense Circuits

Color	Description	Correlated Connection
Green	Wires and Circled Header	Ground
Red	Circled Header	3.3 V Power to Voltage Followers from Microcontroller
Orange	Protruding Wires	Negative Side Current Sense Resistor
Blue	Protruding Wires (behind capacitors)	Positive Side Current Sense Resistor
Light Turquoise	Circled Headers	Current Sense Voltage Outputs to ADCs
Dark Turquoise	Circled Headers	Voltage Sense Voltage Outputs to ADCs
Purple	Circled Header	PWM Low Pass RC Filter from Microcontroller
White	Protruding Wires Top and Bottom	Input Voltage to Voltage Sense Dividers
Gray	Protruding Wire Top Right	PWM + Isolation Resistor Output to DC-DC Converter

With the prototype board complete and each component working during calibration, the next steps integrate the board into the full system.

7.5 Monitoring Input Neuron Data

To monitor data for the input neurons of the ANN, the test setup emulates the actual EHFEM circuitry. The problem observed in Andrew Forster's thesis, is the microinverter pulls the output voltage down until the inverter shuts off below 22 V, which allows the DC-DC converter to properly regulate only for the cycle to repeat [9]. The neural network training attempts to eliminate this cycle, reacting to the microinverter to keep the DC-DC converter's output voltage more consistent.

7.5.1 MONITORING METHODS AND BACKGROUND

Ideally, the system monitoring works at the same speed the code runs in the final implementation. Unfortunately, the UART interface only runs at 115,200 baud, drastically slower than the ADC value sampling. The most obvious work around slows down the system to a speed the UART could handle. In doing so, data packet size output to the UART decreases by stripping down print statements and UART code to the bare minimum. For instance, decreasing the size of the ADC values sampled with the 10 bit ADCs to 16 bit unsigned integers instead of 32 bits, saves memory writing time. Then printing only the necessary ADC values decreases the number of characters output to the terminal. Adding timestamps with Realterm produces a time scale without significantly increasing communication through the UART. The desire for timestamps with millisecond resolution necessitated coding custom timestamps in the command prompt, which only Realterm 3.0.0.31 beta version supports. This version has slight bugs, but no bugs without easy workarounds for this use.

One issue with terminal timestamps is that the timestamps occur as the terminal receives the print statement, rather than when the ADC actually takes its measurements. Assuming a consistent delay between the print statement and ADC readings, coded as close together as possible, the relative time holds true. An attempt at double checking timestamps against a more precise timer employs the RTT code once again. The RTT claims the measurements take 5.565 seconds to complete, while the timestamps show a difference of 5.8 seconds. Upon measuring

values of a saw tooth and sine wave as outlined in section 7.5.2 at 1 Hz, the timestamps correlate to a period of 0.999 seconds and the RTT correlates to a period of 0.953 seconds. Due to the unknown source of the RTT speed increase, this test concludes the UART attaches more accurate timestamps from recording waveform frequencies consistent with the function generator.

An alternative solution explores the Live Watch feature inside Atmel Studio 6.2. This method should work as Atmel claims near real-time monitoring of variables when output to a text file. In practice, this feature slows down the code, since Atmel Studio essentially pauses the code to update all ADC values and then resumes. Each attempt to solve this feature led to new tests and ultimately ended with Atmel Studio crashing on my computer. I ran Atmel Studio in Windows 10 on an older laptop. I think, although likely less common, having a newer laptop with at least an i5 or equivalent processor running Windows 7, since Atmel likely designed and supported Atmel Studio 6.2 on Windows 7, would increase functionality of this feature.

Later, I program a timer since the RTT clock of 32 kHz does not provide accurate timestamping when sampling in the 10 microsecond range. This code, commented out in the final version, provides timestamps for data capturing. It also provides a method for timing neural network calculations, since, depending on the inputs, the calculation time varies drastically. The timer ran at 1.875 MHz and makes four ADC readings as well as storing the timestamp among other less significant functions in about 20.34 clock cycles. This means reading the ADC values takes 10.85 microseconds. This value exceeds the time measured with the RTT in section 7.7, but justifiably so, considering the additional code and hence instructions to take this measurement. Incrementing the index takes time as well as additional if statements, which justify the increase of about 6 microseconds.

7.5.2 TESTING MONITORING SPEED CAPABILITIES

Although monitoring the data directly to the terminal and storing the data each have their own theoretical speeds, timers determine the actual speed. First, the RTT timer measured the speed of outputting the ADC values directly to the terminal. Due to function generator limitations, waveforms only span from 0 to 10 V. I choose a sinewave input to one voltage channel and a

sawtooth to the other, providing some insight into the filtering on each channel. Due to function generator low current source capabilities, this method only allows voltage channel tests.

7.5.3 MONITORING TESTS FOR POWER SUPPLY INPUT AND ELECTRONIC LOAD

The easiest test case uses a test setup where all voltages and currents remain steady. This test aims to use the electronic load, but upon failing easy test cases, the 10.3 Ω, 300 W resistor replaces the electronic load momentarily. The resistive load keeps values consistent with calibration tests. Figure 36 measures current with differential voltage measurements across 1 mΩ sense resistors. Using short leads whenever possible minimizes noise from long wire inductances.

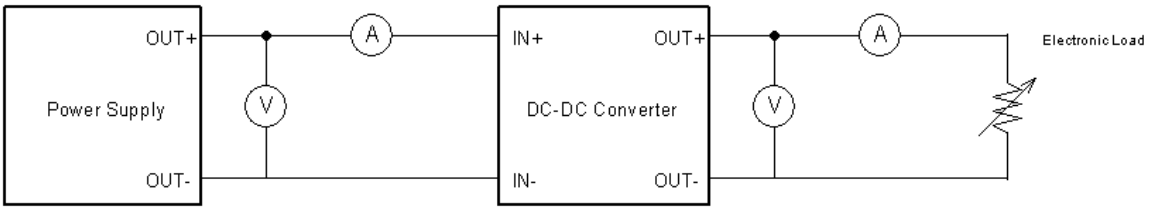

Figure 36. Test Setup to Monitor Current and Voltages from Power Supply to Electronic Load

Required equipment:

- BK Precision Power Supply 540 W
- BK Precision Electronic Load 1200 W (or 10 Ω power resistor for initial check)
- Agilent 3630A Triple Output DC Power Supply
- Andrew Forster's Buck-Boost DC-DC Converter
- Prototyping Board with Current and Voltage Sense Circuitry
- Atmel SAM4S Xplained Pro Evaluation Board
- 5 digital signal wires
- 4 banana-to-spade wires
- 5 banana-to-grabber wires
- 4 alligator clips
- 1 bag of short leads

Calibration proves using 1 or 2 ADC channels simultaneously correctly measures voltages on the corresponding sense circuits. Due to the limitations in number of high power resistors in my possession, calibration initially only measures current with one current sense circuit at a time. When measuring on all 4 ADC channels simultaneously, the values of 512, 328, and 256 appear consistently. These numbers suspiciously correspond to one high bit in a binary representation. After extensive troubleshooting, I notice the first channel ADC, corresponding to the input voltage, would not exceed 512 consistently despite it doing so during calibration. Therefore, this test returns to calibration once again to see if the code limits this channel to a differential ADC measurement. This test suspects differential mode due to its accuracy at 30 V input voltage, where all channels read fairly accurately, and its maximum value of 512 at 40, 45, and 50 V input voltage. The initial reference voltage of 1.2 V on the ADC AREF pin constituted the primary problem. Increasing the voltage to 2.6 V places the reference voltage comfortably above the required 2.4 V minimum. Then modifying microcontroller code set ADC gain values for all channels to 2 to read the values closer to full range.

A second issue with this test setup came to light after fixing the previous problem. I return to calibration of the ADC channels with the new reference voltage. During this test, the ADCs sometimes read steady voltages and currents inconsistently, appearing as voltage and current ripple expected from a DC-DC converter, with magnitudes outside the BK Precision DC power supply's specifications. Troubleshooting determined the power supply causes the ripple and the ripple creates a whining noise from the power supply. The voltage ripples occur most often after starting up the power supply, while rarely occurring when changing the power supply voltage. Waiting for the power supply to regulate with smaller ripple solves this issue, which takes over 15 minutes sometimes. After realizing this issue, I always keep oscilloscope probes on the measured values during steady state tests to help recognize this problem. Figure 37 shows the voltage ripple seen on my voltage divider circuit at a 5 V power supply voltage, which translates to 35.7% of the average output voltage and causes headaches while calibrating. When running tests not at steady state, I make sure to listen for the whining noise from the power supply.

Next, this test implements timestamping with a 1.875 MHz clock to monitor and capture data with the microinverter connected. This provides about 0.13 seconds of data with less than 11 microsecond resolution. To test the data capturing with known values first, this test has a power supply on input and electronic load on output. The DC-DC converter supplies a constant 36 V out, giving a good value to check for consistency while changing the others and taking data. At steady state the test primarily captures ripple current passing through the RC filter on my current measurement circuit at max load.

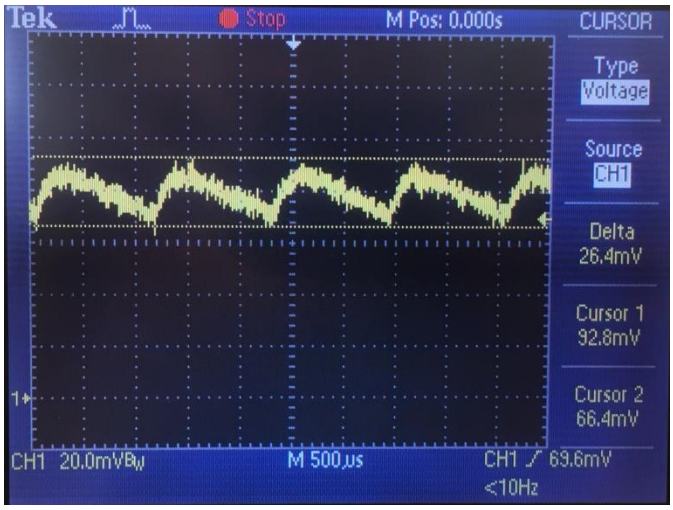

Figure 37. Scope Capture of Power Supply Voltage Ripple Seen on Voltage Divider Circuit

Initially the ADC reads currents somewhat correlated to the actual current, but nowhere near similar to calibration. Troubleshooting determines that various sources of noise prevent the LT6101 from receiving an adequate power supply. The chip receives power from the positive current sense side, which becomes noisy after connecting the DC-DC converter. Reducing the noise involves replacing all long banana-to-banana wires with short leads, adding 0.1 µF capacitors directly across the LT6101 power terminals for higher frequency noise, and large capacitors across the power terminals local to the current sensing circuits to act as charge reservoirs. Capacitor placement puts them in parallel with the input and output capacitance of the DC-DC converter, effectively increasing these values. Adding smaller 10 nF capacitors on the DC-DC converter directly attempt to remove very high frequency noise, but their effects prove difficult to quantify. These noise mitigation techniques reduce ground plane noise to 400 mV_{p-p}.

The largest noise spikes only last tens of nanoseconds, making further reduction difficult with inductance from relatively long leads between circuit components dominating any smaller capacitances. Noise reduction prevents needing to take the median of data points, increasing speed. Taking the median of three data points increases measurement time about fourfold, but reduces measurement error to tens of counts. The neural network implementation avoids taking the median, since so few samples read incorrectly, and the neural network helps mitigate noise.

7.5.4 MONITORING TESTS FOR POWER SUPPLY INPUT AND MICROINVERTER LOAD

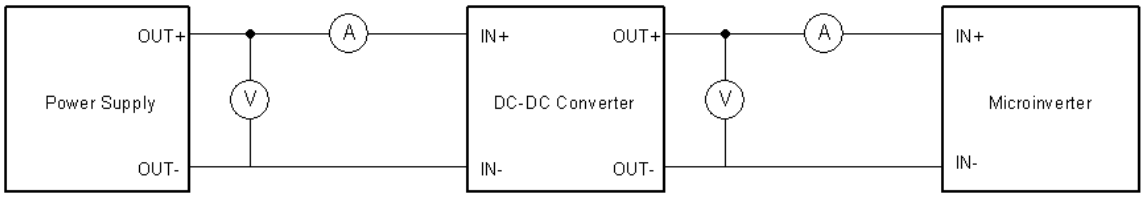

Figure 38. Test Setup to Monitor Current and Voltages from Power Supply to Microinverter

Required equipment:

- BK Precision Power Supply 540 W
- Enphase M215 Microinverter
- Agilent 3630A Triple Output DC Power Supply
- Andrew Forster's Buck-Boost DC-DC Converter
- Prototyping Board with Current and Voltage Sense Circuitry
- Atmel SAM4S Xplained Pro Evaluation Board
- 5 digital signal wires
- 2 banana-to-spade wires
- 5 banana-to-grabber wires
- 3 banana-to-banana wires
- 4 alligator clips
- 1 bag of short leads

The next test setup takes data using a power supply to replace the elliptical, as a stepping stone to the full system. Upon completing the previous test, integration of the microinverter as the load obtains detailed data on the microinverter's effects on the EHFEM

system. This test uses the detailed connection diagram from the microinverter to the $240V_{AC}$ wall from James Ralston's senior project report [30]. Figure 38 shows the full test setup diagram. The ground of the sensing circuit connects to the power supply ground. The ground before the DC-DC converter cannot connect to the ground after the converter without tripping the ground fault interrupt (GFI) of the microinverter as described in Andrew Forster's thesis [9]. Figures 39 through 42 show data captures for one test, noting the suspected issue presents itself as the clear issue through this test.

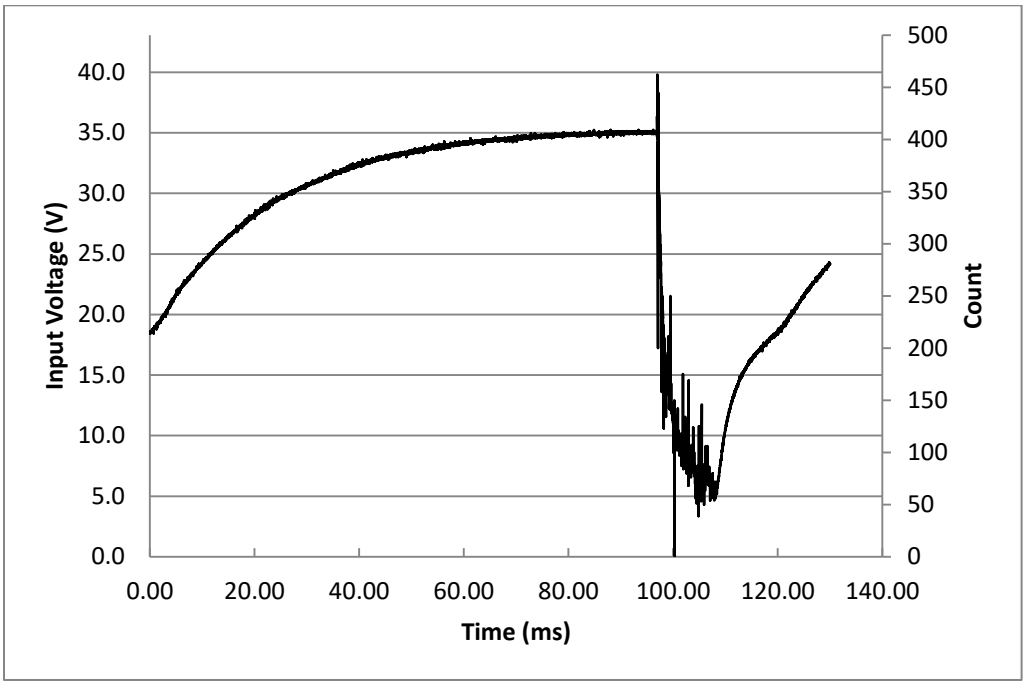

Figure 39. Input Voltage Monitor with Microinverter Load to DC-DC Converter – (Voltage = [Count + 0.5972] / 11.621)

The current graphs both hit max values of 1023, which correlates to currents above 12 A on the output of the DC-DC converter and 9.1 A on the input, despite the input power supply current limit set to 7 A. The microinverter must change its input resistance fast enough that the power supply cannot regulate fast enough, since its voltage dips and current exceeds its limit. The power supply current also only supplies current up to 9.1 A, which could limit the input current to that value, but since this coincides with the maximum value that current sense circuit can measure, this test cannot determine the cause. The large amount of capacitance on the DC-DC converter and current sensing circuits likely cause large currents for short durations without

as large of currents coming from the power supply. The current and voltage sensing circuit enables capturing this data since scope probes ground the reference, tripping the GFI of the microinverter.

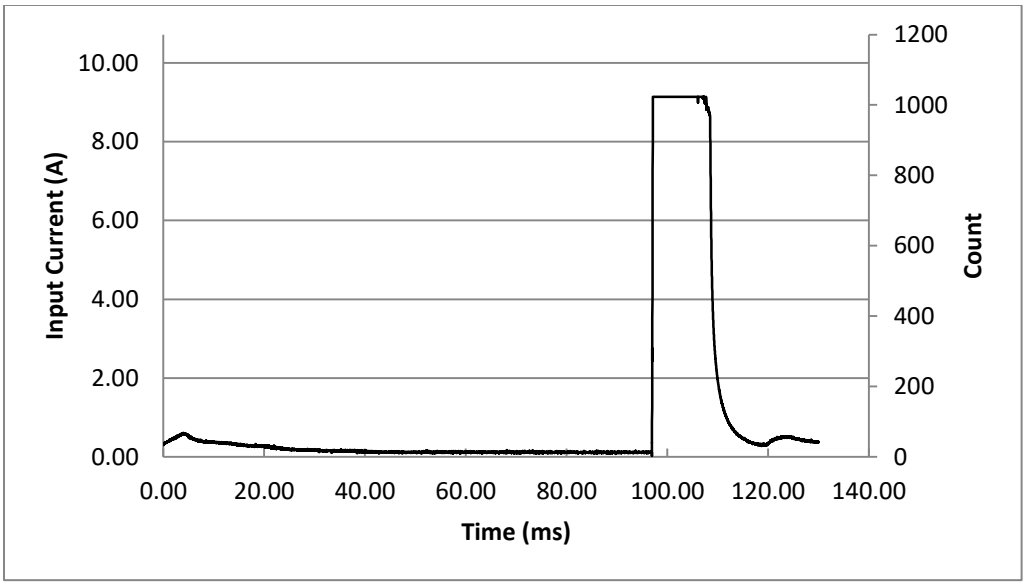

Figure 40. Input Current Monitor with Microinverter Load to DC-DC Converter – (Current = [Count + 8.1723] / 112.67)

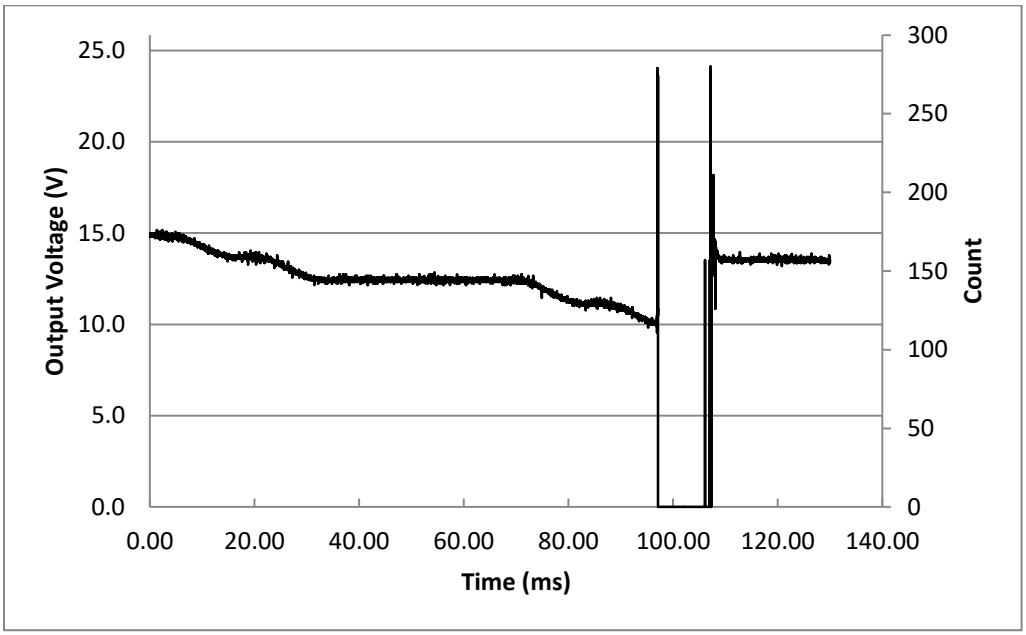

Figure 41. Output Voltage Monitor with Microinverter Load to DC-DC Converter – (Voltage = [Count + 0.9844] / 11.646)

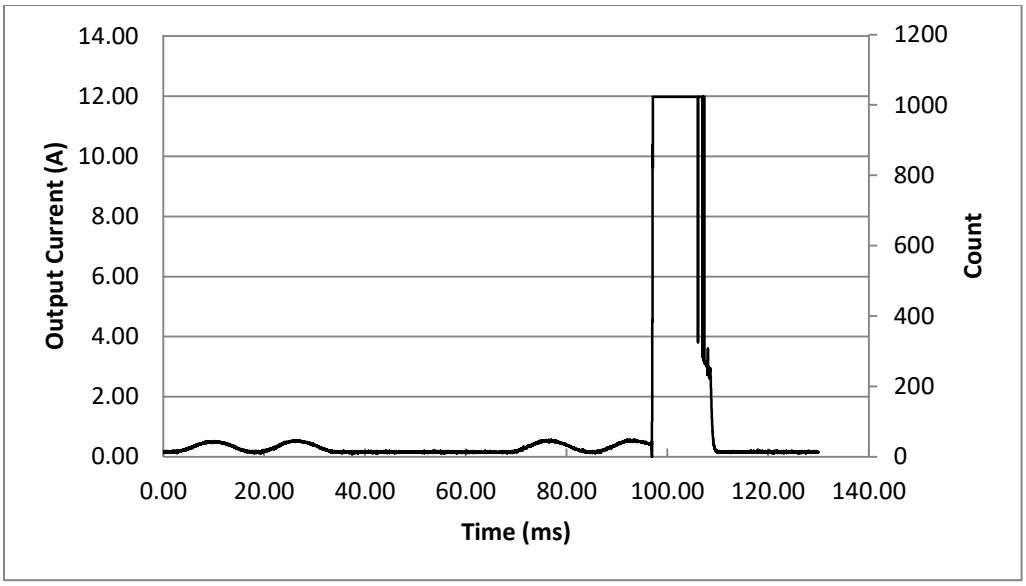

Figure 42. Output Current Monitor with Microinverter Load to DC-DC Converter – (Current = [Count - 4.1987] / 85.093)

With the issues characterized neural network code implements one attempted solution.

7.6 Neural Network Implementation in C Code

Neural networks provide the control to react to changes in currents and voltages to undesired levels. Since Chapter 6 simulates the neural network to acceptable performance levels, this test adds similar code to the microcontroller. The basics include converting the ADC count readings to normalized values compatible with the neural network, asking the neural network what the duty cycle should change to, and using the performance to refine the weights further. This code employs online learning, but does not update neuron weights if the neural network performs as desired to save computation resources.

7.6.1 COMPILER SETTINGS

Initial checks to prove the neural network code worked on the microcontroller found the microcontroller had issues regarding floating point numbers and using math functions. The ANN uses the hyperbolic tangent function, for which Atmel Studio provides a faster approximation, but it defaults to disabled. To fix the problem in Atmel Studio 6.2 go to Project -> “Project name” Properties -> Toolchain -> ARM/GNU Linker -> Miscellaneous. In the linker flags field type “--specs=nano.specs -lc -u _printf_float” to enable the proper compiler settings. The printf support for float numbers enables easier debugging and verification of code operation, if desired. In the

same toolchain window scroll to the folders ARM/GNU C Compiler and ARM/GNU Linker and find the folder called Optimization in each. Check the box labelled “Enable fast math (-ffast-math)” to define the faster tanh() function used in the code in Appendix E. Now that the floating point numbers compute correctly, steady state tests start using the neural network control.

7.6.2 STEADY STATE EXPERIMENTAL RESULTS WITH ANN

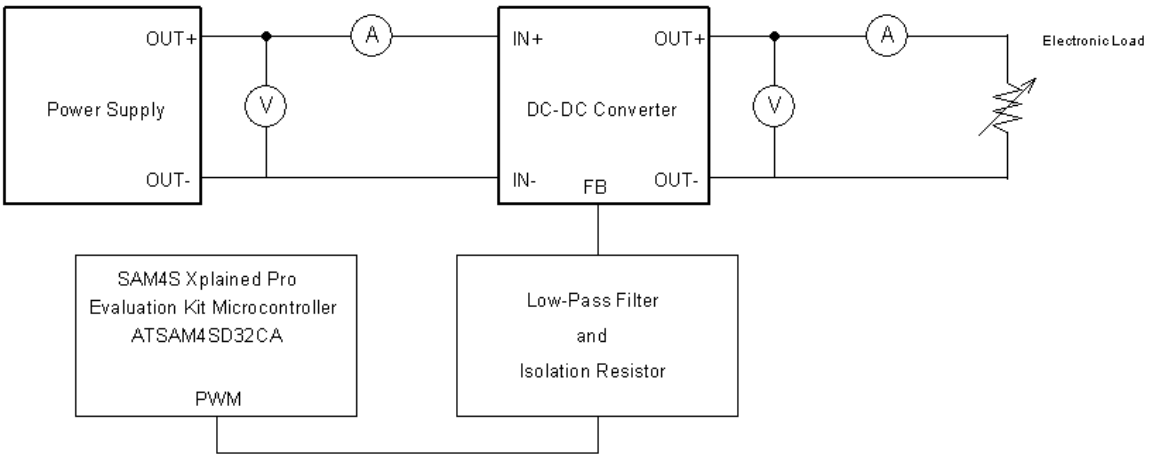

Figure 43. Steady State Test Setup to Control DC-DC Converter Feedback – from Power Supply to Electronic Load

Required equipment:

- BK Precision Power Supply 540 W
- BK Precision Electronic Load 1200 W
- Agilent 3630A Triple Output DC Power Supply
- Andrew Forster’s Buck-Boost DC-DC Converter
- Prototyping Board with Current and Voltage Sense Circuitry
- Atmel SAM4S Xplained Pro Evaluation Board
- 6 digital signal wires
- 4 banana-to-spade wires
- 6 banana-to-grabber wires
- 5 alligator clips
- 1 bag of short leads

This test compares system efficiency and operation with and without the neural network. Figure 43 shows the test setup, which adds the microcontroller to control the feedback voltage pin of the DC-DC converter. The PWM passes through a low pass filter and isolation resistor as shown in Figure 43. Figure 17 contains the schematic for this block interfaced with the feedback voltage divider. The isolation resistor starts at 105.6 kΩ, reducing the pull of the microcontroller to only a 0.5 V range. This range noticeably affects the output voltage of the DC-DC converter when the PWM initializes to 100% duty cycle, pulling the output voltage down to about 31 V. Running the neural network code restores the 36 V operating voltage even as input voltage and load currents change.

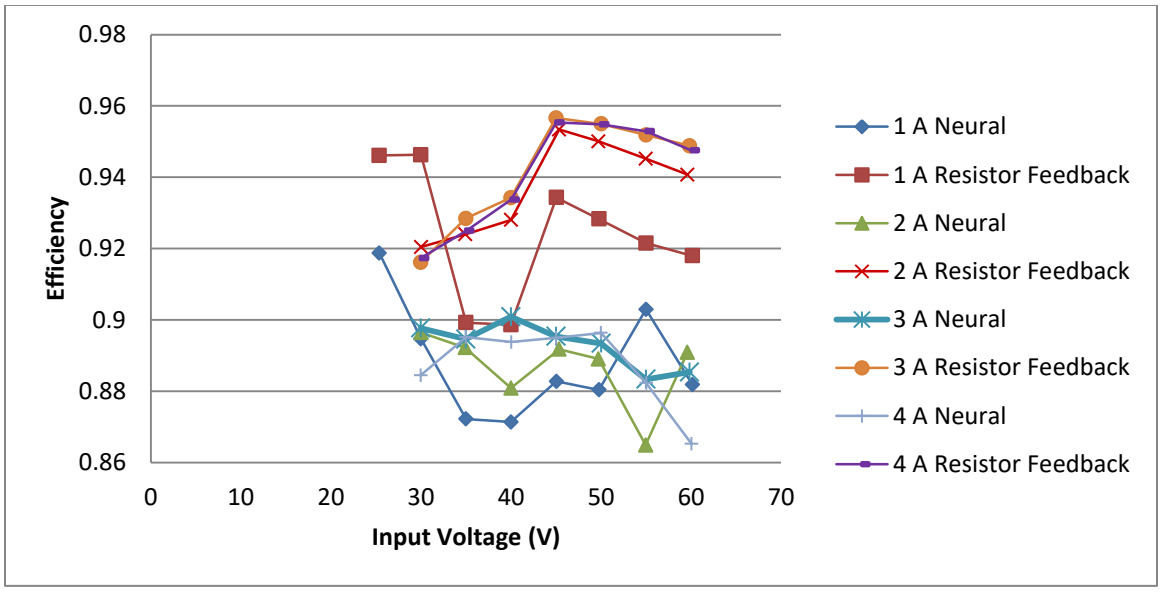

Figure 44. Efficiency Comparison with and without Neural Network Controller – 105.6 kΩ Isolation Resistance

Figure 44 shows the efficiency comparison with the neural network controller using a 105 kΩ isolation resistance. Most noticeably are decreased efficiencies, likely caused by control speed. Figure 45 shows the same results, but with the 5.6 kΩ isolation resistance as planned and used in microinverter testing. The smaller isolation resistance correlates to lower efficiencies at input voltages that the converter and neural network could properly regulate, although, during microinverter testing, the input voltage varied though these levels without the same regulation issue. Figures 44 and 45 show the possibilities of future neural network implementations with

additional optimization. The test with only a 5.6 kΩ resistor does not regulate the voltage on as wide of an input voltage range. Employing PWM with finer voltage increments could allow more precise control, which most likely causes this reduced input voltage range, along with higher efficiencies. The neural network test points have two additional 1 mΩ sense resistors in the power path, not helping efficiency, although they affect it minimally. With the high amount of noise that necessitated filtering capacitors, come losses through ESR of capacitors especially in the high frequency range.

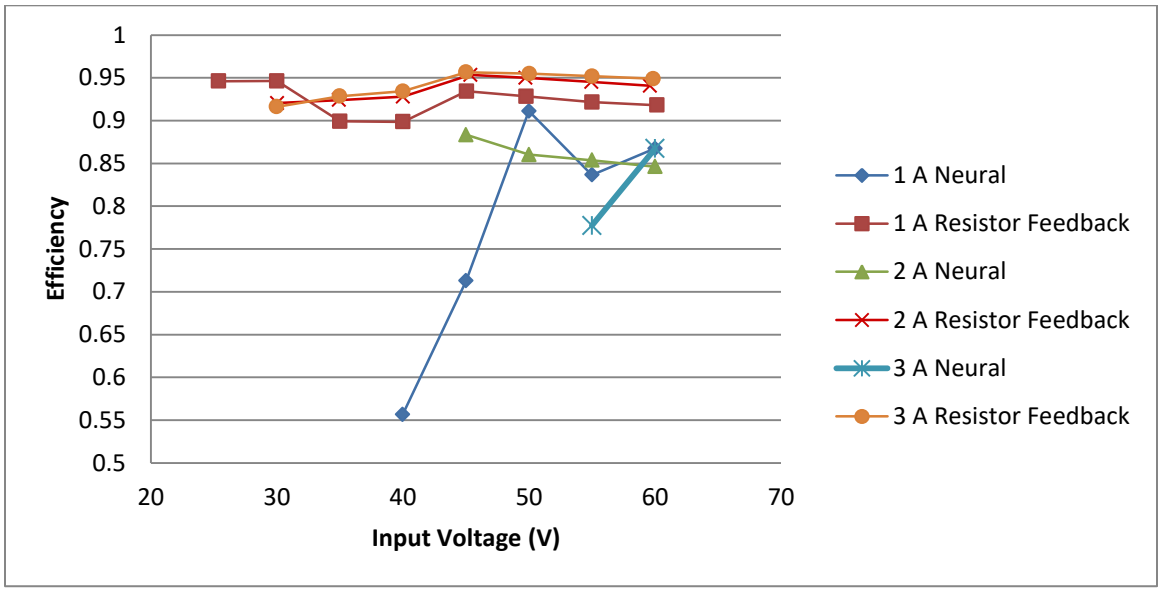

Figure 45. Efficiency Comparison with and without Neural Network Controller – 5.6 kΩ Isolation Resistance

Figure 46 validates the neural network control as not causing the input resistance of the converter to vary significantly. Theoretically, the neural network could try to maintain a 10 Ω resistance on the input of the DC-DC converter, but this does not align with the conventional DC-DC converter IV curves. Maintaining a 10 Ω input resistance requires current control as well, separate from voltage, which requires additional inputs to the converter.

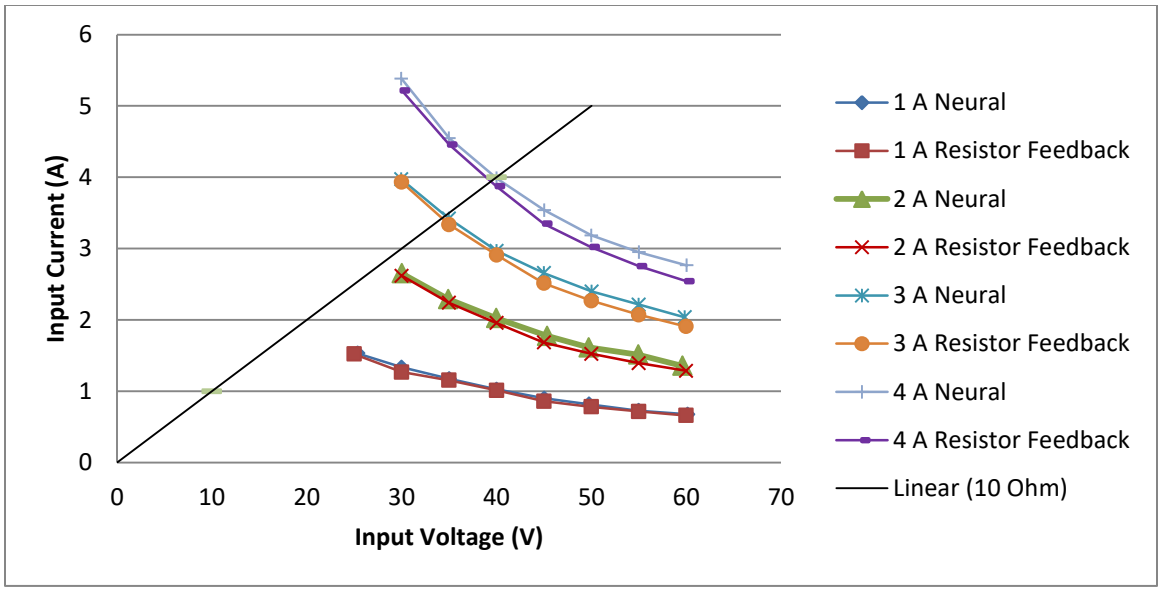

Figure 46. Input Resistance to DC-DC Converter with and without Neural Network Controller

7.6.3 DYNAMIC EXPERIMENTAL RESULTS WITH ANN

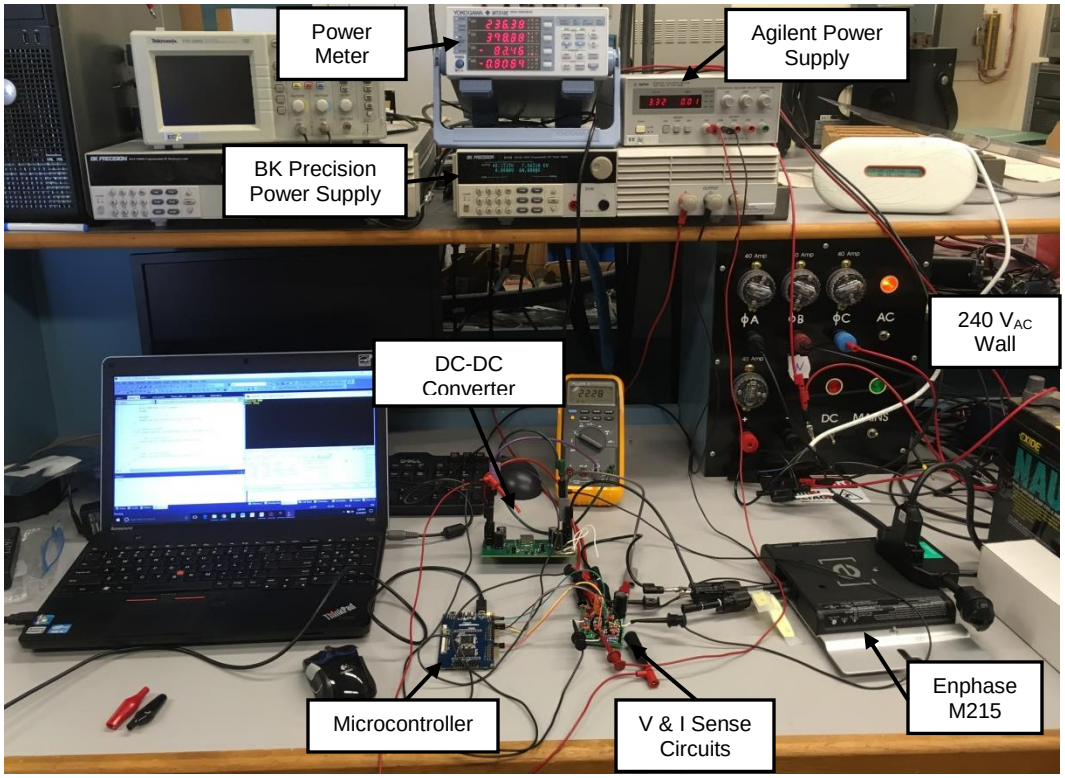

Figure 47. Lab Bench Setup at Cal Poly in 20-150 – ANN and Microinverter Tests

Required equipment:

- BK Precision Power Supply 540 W
- Enphase M215 Microinverter
- Agilent 3630A Triple Output DC Power Supply
- Andrew Forster's Buck-Boost DC-DC Converter
- Prototyping Board with Current and Voltage Sense Circuitry
- Atmel SAM4S Xplained Pro Evaluation Board
- Yokogawa WT310E Digital Power Meter
- 6 digital signal wires
- 2 banana-to-spade wires
- 6 banana-to-grabber wires
- 5 alligator clips
- 1 bag of short leads

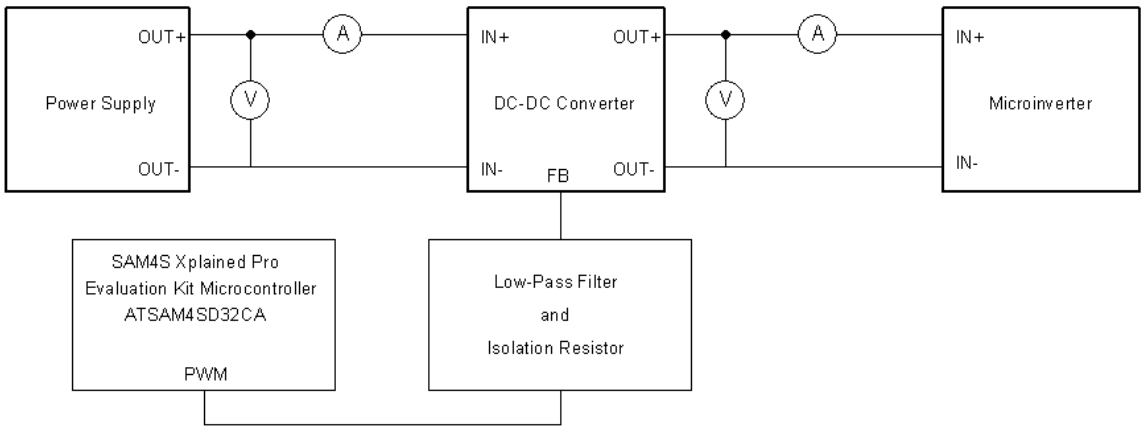

Figure 48. Dynamic Test Setup to Control DC-DC Converter Feedback – from Power Supply to Microinverter

Figure 47 contains the test setup on the lab bench for testing the ANN with the microinverter and Figure 48 contains the detailed block diagram. The microinverter changes its input resistance to find the maximum power point, making these tests change dynamically. Due to the variance of all voltages and currents, individual data points become difficult to record, while maintaining any type of general meaning. The power supply current limit of 7 A helps reduce

large currents, but current can rise above 7 A momentarily before the power supply can reduce it. Additional current limits under the same test did not achieve the same results, which imply the current limit aids in system functionality.

Without the neural network controller, the converter clicks at about 1 Hz, converting no power. About an additional 1 minute after the microinverter turns on, the converter then attempts to regulate the voltage, but only for the microinverter to turn off when the voltage on its input drops below 16 V [9], [30].

With the ANN controller, the inverter doesn't turn off, or at least not long enough to cause it to act negatively. I only run tests at 60 V on the input, since that seems optimal for the neural network and DC-DC converter combination. The power supply voltage varies from 15 to 60 V and the output voltage sits around 22 V. The current protection code, that lowers output voltage when current is too high, most likely determines this voltage. After regulating at 22 V for over 20 seconds, the converter output voltage momentarily drops to 18 V for a few seconds. Attempting to raise the output voltage at this point allows excessive currents, disqualifying any proposed solutions. The microinverter starts up at 22 V across its input terminals and turns off at 16 V, but the correlation of these levels to observed levels does not produce an argument for causation. The voltage never drops below 16 V on the voltmeter, which means the microinverter doesn't turn off. The system ran as described at a 7 A current limit for 6 minutes without the microinverter turning off, and, after the inductor got hot, I decided to end testing.

In this test, the converter converts more power. When the ANN code runs, currents rise to maximums of 8 A, although they usually do not exceed the 7 A current limit, and do not often go below 1 A. Finding the average power converted becomes unreasonable since the microcontroller cannot collect data long enough to get a meaningful result. However, this test measures power sent to the grid and power from the supply to produce Table 14. These values do not necessarily represent the average power due to the inability to monitor voltages and currents with the microcontroller without slowing the neural network down, therefore altering its performance. This method uses a Yokogawa power meter to measure the output power and the table finds input power from the voltage and current readings on the power supply. Due to quickly

changing values, taking pictures of readings then transferring data to the table produces more accurate results. Still, this test achieves efficiencies over 100%, most likely due to a moving average feature on the Yokogawa power meter.

Table 14. Power Data from Microinverter Load Tests with the ANN Controlling the DC-DC Converter

Vin (V)	Iin (A)	Input Power (W)	Vout (V)	Iout (A)	PFout	Output Power (W)	Efficiency (%)
39.7	0.59	23.46	236.16	0.388	0.8851	81.01	345.3
47.7	1.71	81.77	236.19	0.386	0.8888	82.98	101.5
23.4	0.59	13.76	236.20	0.362	0.8712	74.50	541.3
37.6	5.88	220.62	236.24	0.378	0.8849	78.93	35.8
30.0	8.28	248.55	236.20	0.363	0.8837	75.75	30.5
39.6	8.03	318.07	236.17	0.356	0.8688	73.10	23.0
47.4	7.89	374.23	236.17	0.356	0.8748	71.79	19.2
59.6	6.81	406.16	236.25	0.365	0.8487	73.11	18.0
80.1	3.03	243.01	236.27	0.386	0.8695	79.25	32.6
38.9	0.61	23.60	236.24	0.348	0.8061	66.20	280.5
23.1	0.73	16.79	236.27	0.366	0.864	74.70	444.8
24.1	2.41	58.01	236.22	0.360	0.8635	73.43	126.6
22.3	2.47	55.04	236.21	0.360	0.8726	74.24	134.9
21.8	3.24	70.55	236.21	0.332	0.8388	65.83	93.3
34.2	6.00	205.31	236.24	0.362	0.8671	74.14	36.1
37.1	7.25	269.07	236.25	0.352	0.8684	72.31	26.9
59.0	7.09	418.11	236.13	0.362	0.881	75.35	18.0
55.1	3.90	214.86	236.14	0.363	0.8631	73.89	34.4
23.5	0.98	23.01	236.27	0.365	0.884	76.33	331.8
46.8	7.22	338.05	236.39	0.366	0.8838	76.50	22.6
59.2	6.72	397.43	236.32	0.357	0.8632	72.90	18.3
27.1	3.42	92.52	236.38	0.360	0.8786	74.83	80.9
52.8	3.71	195.91	236.43	0.384	0.8862	80.41	41.0
36.2	2.88	104.04	236.36	0.361	0.8517	72.72	69.9
29.7	7.07	209.89	236.43	0.383	0.905	82.05	39.1
39.7	6.00	238.45	236.40	0.369	0.8784	76.61	32.1
57.9	5.77	333.89	236.34	0.376	0.8765	77.94	23.3
45.8	7.94	363.49	236.38	0.373	0.8835	77.93	21.4
32.5	1.05	34.20	236.28	0.350	0.8489	70.13	205.0
40.9	1.33	54.33	236.31	0.360	0.8822	75.10	138.2
37.2	0.95	35.37	236.29	0.377	0.8825	78.71	222.5
30.7	4.95	152.10	236.24	0.385	0.8816	80.21	52.7
57.2	7.62	436.34	236.28	0.369	0.8793	76.69	17.6
32.7	5.93	193.93	236.27	0.367	0.8795	76.19	39.3
50.6	6.69	338.90	236.25	0.382	0.8816	79.55	23.5
27.7	1.18	32.57	236.22	0.368	0.8853	76.88	236.1
44.6	5.39	240.74	236.33	0.362	0.8847	75.72	31.5
Totals:		**7076.13**				**2797.91**	**39.5**

During testing the output power on the Yokogawa power meter seemed to remain too constant, considering the quickly changing input power. Since this test suspects a moving average on the output, Table 14 sums the input and output power of all measurements to find the overall efficiency. This efficiency of 39.5% represents only an estimate of what the overall efficiency might be, so future work must develop an accurate method of measuring power. The easiest solution might just use two Yokogawa power meters to measure input and output power consistently with identical equipment, but at the submission time of this project, no other power meters were available.

This test suggests the system might work better with current control, but ideally voltage and current stay more constant and within maximum efficiency points. Current likely carries more weight in the MPPT, and enabling direct control of it, we could create the knee in a solar panel IV curve. The MPPT algorithm would then want to stay at the same point the EHFEM system achieves highest efficiency.

During testing, the Atmel board occasionally shuts down. It appears to happen during high currents or voltages, possibly providing too much voltage on those ADC pins, and triggering overvoltage or overcurrent conditions. My computer might have also decided to act up, causing this issue itself. This problem did not appear during any efficiency tests, but did after trying the microinverter tests and returning to high current tests with the electronic load. It also happened at low currents, though rarely, so this further shrouds the root cause.

7.7 Timing Speed of Control

To accurately control the converter, the controller must respond to changes in input neurons quickly. For this, the Atmel microcontroller uses the real time timer (RTT). The RTT counts clock cycles of the internal 32.768 kHz crystal oscillator. To determine the base point, the RTT reads 4.3 µs as the time the microcontroller takes to read all four ADC values.

When running the steady state tests, the RTT allows counting clock cycles similarly to before. The RTT timer determines the average run time for the neural network across 1,000 weight updates to be 1.12 milliseconds. Decreasing this time would provide faster reaction times, since this time resides within a borderline acceptable region.

Several attempts to increase floating point speed ended in the DC-DC converter accepting a smaller input voltage range. The first attempt used the -funroll-loops optimization flag in the ARM/GNU C Compiler to step through the embedded loops faster. The second attempt set -funsafe-math-optimizations in the optimization settings in both the ARM/GNU C Compiler and the AM/GNU Linker toolchains. Both decreased neural network regulation and all tests omitted them.

Chapter 8. Conclusion and Future Work

This thesis project explores and confirms neural networks can regulate the voltage on the LT8705 DC-DC converter chip. Despite the datasheet not describing exactly what each input node does inside the control architecture, experiments determine the general effects. At steady state, the neural network worked consistently, which supports future work optimizing the neural network to improve efficiency.

The neural network ran into many issues, with not enough time to fix every one. Upon Andrew Forster completing the DC-DC converter, only one quarter remained for lab experiments. Unfortunately a few incorrect assumptions about the LT8705 control scheme prevented hardware interfaces from working perfectly the first design iteration. After fixing each hardware design, setbacks limited neural network optimization time. Neural network optimization could continue indefinitely and refining the initial network weights could prevent converter output voltage overshoot upon startup. Additional optimization could increase the Atmel microcontroller code execution speed, or a digital signal processor could interface with the Atmel SAM4S to handle floating point operations quicker. This project explored some compiler optimization to increase operation speed, but the microcontroller also caused the DC-DC converter to accept a smaller input range.

Another idea lowers the PWM low pass filter cutoff frequency, then slowing the PWM switching speed down to allow finer adjustments on the output voltage. A gain stage would allow a higher isolation resistor value, which reduces the effects of the larger low pass filter capacitance.

Creating additional hardware to control the current and voltage in the LT8705 chip could enable neural network regulation to find proper output voltage and current to maintain a near 10 Ω resistor on the input of the DC-DC converter. This would drastically change the purpose of the neural network, but could improve workout consistency for the EHFEM system end user.

WORKS CITED

[1] V. Chau, J. Roecks, S. Spurr, and D. Webb, “Exercise Bike Power Generator,” Senior Project Report, California Polytechnic State University, 2007.

[2] Energy Efficient Gyms, “ReRev Technology in the Wake Forest Gym,” *Energy Efficient Gyms,* 2012. [Online]. Available: http://energyefficientgyms.weebly.com/rerev-at-wake.html. [Accessed: Nov. 7, 2015].

[3] T. Newcomb, “In the Gym: Clean Energy from Muscle Power,” *TIME Inc.,* Dec. 2, 2010. [Online]. Available: http://content.time.com/time/business/article/0,8599,2032281,00.html. [Accessed: Nov 7, 2015].

[4] Sports Art, “G845 Elliptical,” *Sports Art,* 2017. [Online]. Available: http://us.gosportsart.com/product/g845-elliptical-2/. [Accessed: June 20, 2017].

[5] J. Arakaki, P. Lawrence and A. Nakamura. “Energy Harvesting from Exercise Machines Cal Poly Recreation Center Implementation,” Cal Poly Digital Commons. 2010. [Online]. http://digitalcommons.calpoly.edu/eesp/41/

[6] K. Leslie, "Cal Poly to see its biggest freshman class ever this fall," The Tribune, Aug. 9, 2013. [Online]. Available: http://www.sanluisobispo.com/news/local/education/article39451995.html. [Accessed: Dec. 11, 2015].

[7] A. Hilario, “Energy Harvesting From Exercise Machines Using Four-Switch Buck-Boost Topology,” M.S. Thesis, *Cal Poly Digital Commons.* 2011. [Online]. http://digitalcommons.calpoly.edu/theses/511/

[8] A. Nadhir and T. Hiyama, "Maximum Power Point Tracking Based Optimal Control Wind Energy Conversion System," in *Second International Conference on Advances in Computing, Control and Telecommunication Technologies (ACT), 2010, December 2-3, 2010, Jakarta.* IEEE, 2010. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5675847&isnumber=5675795. [Accessed: October 11, 2015].

[9] A. Forster. “Energy Harvesting from Exercise Machines: Buck-Boost Converter Design,” Cal Poly Digital Commons. 2017. [Online]. http://digitalcommons.calpoly.edu/theses/1702

[10] M. DeSando. “Universal Programmable Battery Charger With Optional Battery Management System,” M.S. thesis, *Cal Poly Digital Commons*. 2015. [Online]. http://digitalcommons.calpoly.edu/theses/1409

[11] University of Texas at Austin, "Chapter 15: Ratio Control," University of Texas at Austin. [Online]. Available: http://www.che.utexas.edu/course/che360/lecture_notes/chapter_15.ppt. [Accessed: Dec. 11, 2015].

[12] W. Li and X. H. Yu, "A Self-tuning Controller for Real-time Voltage Regulation," *Neural Networks, 2007. IJCNN 2007. International Joint Conference on,* Orlando, FL, 2007, pp. 2010-2014.

[13] X. H. Yu, W. Li, and Taufik, "Design and Implementation of a Neural Network Controller for Real-Time Adaptive Voltage Regulation," *Neurocomputing*, vol. 73, no. 1-3, p. 531-535, Dec. 2009. [Online]. Available: Cal Poly Digital Commons, http://digitalcommons.calpoly.edu/eeng_fac/177/. [Accessed: March 18, 2016].

[14] W. Li and X. H. Yu, "Improving DC Power Supply Efficiency with Neural Network Controller,"*Control and Automation, 2007. ICCA 2007. IEEE International Conference on,* Guangzhou, 2007, pp. 1575-1580.

[15] W. M. Utomo, S. S. Yi, Y. M. Y Buswig, Z. A. Haron, A. A. Bakar, and M. Z. Ahmad, "Voltage Tracking of a DC-DC Flyback Converter Using Neural Network Control," *International Journal of Power Electronics and Drive System,* vol.2, no.1, p. 35-42, March 2012. [Online]. Available: http://iaesjournal.com/online/index.php/IJPEDS/article/view/154. [Accessed: June 3, 2016].

[16] Automation Direct, "What do the NEMA Ratings Mean?," *NEMA Electrical Enclosures and Industrial Cabinets*, April 15, 2015. [Online]. Available: http://www.automationdirect.com/static/specs/encnemaratings.pdf. [Accessed: March 15, 2016].

[17] Atmel Corporation, "Atmel | SMART ARM-based Flash MCU," Atmel, Jun. 9, 2015. [Online]. Available: http://www.microchip.com/wwwproducts/en/ATSAM4SD32C. [Accessed: Oct. 17, 2015].

[18] E. Funsten and C. Kiddoo. “Protection System: For the Energy Harvesting from Exercise Machines (EHFEM) project,” Cal Poly Digital Commons. 2014. [Online]. http://digitalcommons.calpoly.edu/eesp/259/

[19] Texas Instruments, "TMS320F2806x Piccolo Microcntrollers," Texas Instruments, July 2014. [Online]. Available: http://www.ti.com/lit/ds/symlink/tms320f28069m.pdf. [Accessed: Dec. 8, 2015].

[20] Enphase, "Enphase M215," Enphase, 2015. [Online]. Available: http://www.enphase.com/sites/default/files/M215_DS_EN_60Hz.pdf. [Accessed: Dec. 1, 2015].

[21] J. Yuen, M. Lum, C. Cinkornpumin, and J. Chan. “Energy Harvesting From Exercise Machines (EHFEM) Self-generator Elliptical Machine,” *Cal Poly Digital Commons*. 2008. [Online]. http://digitalcommons.calpoly.edu/eesp/12

[22] R. Ford and C. Coulston, *Design for Electrical and Computer Engineers*. McGraw-Hill, 1st ed., 2007.

[23] Linear Technology, "LT8705," Linear Technology, 2015. [Online]. Available: http://www.linear.com/product/LT8705. [Accessed: March 19, 2016].

[24] D. Braun. “Post-Leave Report for Spring 2012 Sabbatical,” San Luis Obispo: Cal Poly State Univ., Electrical Engineering. 15 Oct 2012.

[25] A. Robertson. “Input Protection Optimization for Energy Harvesting from Exercise Machines” Cal Poly Digital Commons. In-progress.

[26] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed., New Jersey: Pearson Prentice Hall, 2009.

[27] V. Lugade, "Neural Networks – A Multilayer Perceptron in Matlab," Matlab Geeks, June 9, 2011. [Online]. Available: http://matlabgeeks.com/tips-tutorials/neural-networks-a-multilayer-perceptron-in-matlab/. [Accessed: Apr. 11, 2017].

[28] Atmel Corporation, "SAM4S Xplained Pro User Guide," Atmel, Jun., 2015. [Online]. Available: http://www.atmel.com/Images/Atmel-42075-SAM4S-Xplained-Pro_User-Guide.pdf. [Accessed: May. 7, 2017].

[29] C. Crivelli. “Current Protection for the Energy Harvesting from Exercise Machines (EHFEM) Project,” Cal Poly Digital Commons. 2015. [Online]. http://digitalcommons.calpoly.edu/eesp/295/

[30] J. Ralston. “Microinverter Improvement for the Energy Harvesting from Exercise Machines Project,” Cal Poly Digital Commons. 2016. [Online]. http://digitalcommons.calpoly.edu/eesp/354

[31] P. Schewe, "Preventing Blackouts: Building a Smarter Power Grid," *Scientific American*, vol. 296, no. 5, May 2007. [Online]. Available: http://www.scientificamerican.com/article/preventing-blackouts-power-grid/. [Accessed: March 15, 2016].

[32] M. Haji, K. Lau and A. Agogino, "Harnessing Human Power for Alternative Energy in Fitness Facilities: A Case Study," ResearchGate, University of California, Berkeley, Feb. 5, 2015. [Online]. Available: https://www.researchgate.net/publication/268352134_Harnessing_Human_Power_for_Alternative_Energy_in_Fitness_Facilities_A_Case_Study. [Accessed: Oct. 17, 2015].

[33] NASA Earth Science Communications Team, "Climate change: How do we know?," NASA Global Climate Change, California Institute of Technology, Dec. 3, 2015. [Online]. Available: http://climate.nasa.gov/evidence/. [Accessed: Dec. 11, 2013].

[34] California Energy Commission, "California Renewable Energy Overview and Programs," CA.gov, 2015. [Online]. Available: http://www.energy.ca.gov/renewables/. [Accessed: Nov. 8, 2015].

APPENDICES

Appendix A. Analysis of Senior Project Design

Project Title: DC-DC Converter Control System for the Energy Harvesting from Exercise Machines System

Student's Name: Alexander Sireci

Advisor's Name: Dr. David Braun

1. Summary of Functional Requirements

The control system limits the voltage and current, and hence the power going into the DC-DC converter and inverter. The controller changes the duty cycle of the DC-DC converter to keep it running at an optimal point. In the ideal design, the controller implements the DC-DC converter's feedback. This allows increased control of the transfer function to change during operation, instead of limiting the feedback to discrete components. The system must accurately control changing inputs within microseconds, making a quick response time essential. Chapter 2.2 contains a complete explanation of the functional requirements.

2. Primary Constraints

The main challenge presents itself in keeping costs low enough to make the system pay for itself over its lifetime. Due to electricity's low cost, the minimal amount of energy human exert during exercise, and the non-constant usage, the EHFEM must last many years to become economically beneficial [5]. Another challenge arises from the variable power output that can cause voltage and current spikes every microsecond. As a result, the microcontroller must respond extremely quickly.

Another difficulty lies in interfacing the controller to the rest of the system. The inputs to the ADC can have magnitudes up to 51 V, which exceeds the microcontroller's 2.4 V to 3.6 V range [17], [18]. An attenuator reduces the magnitude and adds an offset voltage to meet this specification. Attenuating the voltage increases the need for a high ADC resolution. Section 2.2 outlines the full specification list and expands on the requirements.

These constraints come from the customers' needs, which Section 2.1 fully explains. Cal Poly's REC Center places the biggest constraint on safety of gym goers, so they require a proper enclosure that seals the electronics from possible spillage.

3. Economic

The project directly impacts the REC Center, where they buy the EHFEM to save on electricity costs. Since they capture energy instead of dissipating it as heat, they save money on air cooling systems as well. The electricity savings cause the utility company's profits to decline slightly, but also helps prevent blackouts by reducing the load at peak times. Preventing blackouts could cause a positive economic advantage for supplying uninterrupted service [31]. Buying the system gives the university most profits, since they sponsored project development. In the short run, profits remain minimal since the system sells almost at-cost. Since Dr. Braun leads the project, he sees the most compensation, primarily since Cal Poly pays him for some of his time spent working on the project. The Earth's resources benefit from the project, reducing the consumption of fossil fuels destroying the atmosphere. Cal Poly reimburses up to $150 of development costs and the rest comes out of my pocket, or companies donate the parts the project needs.

During the product's lifecycle, ideally the REC Center only benefits. The system should start reducing carbon emissions and REC Center costs immediately after installation. Costs only increase if components require extensive maintenance. Due to the system's intentionally robust design, very few systems should need maintenance after initial testing. The system specifications specify a lifetime of at least 7 years after installation, currently slated for after the 2017-2018 school year. Overall project development time would therefore span about 12 years, with the control system development lasting 20 months according to the Gantt chart in Figures 6 and 7. Upon the project's completion, the system gets installed at the Cal Poly REC Center as the beta testing version. I anticipate several senior projects simplifying the system, combining PCB boards to cut costs, and redesigning past projects to improve reliability.

As stated previously, Cal Poly largely funds development costs. The project's continual advances over many years complicate development cost estimates for the entire project. I estimate the control system development costs at $18,072, but labor wages contribute to most costs. Considering only the parts costs, the entire project estimates cost at $140. Cal Poly's Senior Project Fund exceeds this amount, so they cover these costs. The equipment present in the EHFEM system already works, so no needs for additional equipment arise.

4. Commercial Manufacturing

At first, only 6 to 10 elliptical machines at Cal Poly's REC Center utilize this new technology, but once the system proves high reliability and cost effectiveness, as many as 1,000 gyms could utilize this technology totaling 10,000 units. Production takes time to ramp up, but at full scale, sales can reach about 1,000 units. The projected life of 7-10 years implies at least 7,000 units simultaneously producing electricity. The current goals keep the parts cost below $250, which makes the system cost about $400 after manufacturing costs. Installation costs run around $50 per unit, assuming at least 10 simultaneous installations. The estimated purchase price around $500 allows for some investment into cheaper manufacturing processes in the future. Profit can therefore reach $50,000 per year using the same 1,000 units per year assumption. The user should have no cost to operate the device, only electricity generation during equipment use.

5. Environmental

The overall system uses a battery, so mining the lithium produces the most carbon emissions and demands the scarcest material. The controller uses silicon, copper, and ceramics, among other plentiful resources. All parts follow RoHS compliant requirements to minimize environmental impacts disposing of any materials after their lives. During production and transportation of the components, carbon emissions hurt the atmosphere, but these negatives get more than offset over the products' lifetimes. All living species continue to face challenges adapting to global warming's increased effects on the environment, but this project aims to reduce these long-term effects. A case study performed at University of California, Berkeley, evaluates the effects of installing 28 ReRev machines at their recreation center [32]. Although the

EHFEM system provides both a cheaper and more efficient solution, the analysis supplies a bearish estimate. The study concludes that production of the 28 systems produces 9.9 metric tons (MT) of carbon dioxide emissions, which offsets the emissions in just 3 years [32]. Since the EHFEM project projects a useful lifecycle of at least seven years, harnessing the electric energy saves over twice the emissions produced during system production. As global warming becomes one of the biggest problems of the modern world, the ability to save the environment provides enough reason to invest in such systems. Evidence that climate change is a large problem includes the Paris United Nations climate conference and that the carbon dioxide levels continue to surpass 33% higher than ever in the last 650,000 years [32].

6. Manufacturability

Most components bought from existing manufacturers have no manufacturability issues. The only problem arises while assembling the PCB if components cannot be wave-soldered. Installation into the elliptical machines requires an electrician to run the wires and place the EHFEM system inside the elliptical. No issues arise from finding an electrician to install the systems, since we can provide detailed installation instructions, distributed with each system sold. Wiring the elliptical machines to the grid requires a path from each elliptical machine to the central circuit breaker in each gym. This wiring causes complications to reduce obstructions to the walkway.

7. Sustainability

The product achieves complete sustainability in the near term. In the long term, lithium ion abundance decreases, but new battery technology reduces the use of lithium in batteries. In addition, the system reduces the carbon dioxide emissions by over double the emissions created during system production [32]. Upgrading the system requires a complete redesign due to compatibility issues, unless the new parts use consistent interfaces with the current design. Some possible upgrades include tuning the control code for faster response times, using a higher speed microcontroller, and reducing the number of instructions needed to calculate the feedback adjustment.

8. Ethical

This project aligns with the IEEE Code of Ethics. For instance, the focus on health and safety to the gym goers aligns with the first and ninth planks [22]. Since the EHFEM system positively preserves the environment, contrary to the norm for new technology, it reinforces the first plank further. The IEEE code of ethics also requires engineers to present all claims in honest light, based on data [22]. This senior project and master's thesis document presents all data and claims as realistic as possible to improve the readers' understanding of the project. Helping others involved in the EHFEM project improvement aligns with the sixth and tenth planks [22]. I also seek a tremendous amount of technical advice through every avenue available to me. I have Dr. Braun, a professor in electrical engineering at Cal Poly, who knows the EHFEM system well, as the first reviewer of my work. Additionally, multiple other professors offer advice in their specialties, such as Dr. Helen Yu, who specializes in control systems. Before completion of my degrees, I must also defend my thesis, which means at least three other industry experts or professors review my work. The high amount of peer review causes my reliability to increase, although I have no reputation in industry yet. Producing a comprehensive project and report bolters my reputation for future endeavors.

This product also follows good ethics according to Ethical Principlism. The EHFEM system has non-maleficence due to its sustainable design and RoHS compliant materials. The product has beneficence to many, providing cleaner air for the public to breathe. In addition, the REC Center's electricity costs see reductions. Following the fairness aspect, the gym buying the system rightly saves more money than they invest in the product, essentially paying them to keep the air clean. The system also treats users fairly by offering a system to capture energy without altering their workouts. The EHFEM system grants freedom to every gym goer to decide if they want to use the machines or not. In the early stages of the project, choice between standard elliptical machines and ones outfit with the EHFEM system forces no one to use this product.

9. Health and Safety

No health or safety concerns exist from specific usage of this product, except those already there from strenuous workouts. Strenuous workouts can cause dehydration and dizziness

due to low water and oxygen levels, but the EHFEM system does not increase the risk of such experiences. The user must factor in the possibility of these experiences before using the equipment provided by their local gym. The project maintains the user experience, so no additional safety concerns appear. A chance for harm exists for drivers transporting the product, but this project does not increase any risk already present. Wiring may cause safety hazards if not properly routed, but strategic placement with floor or wall wiring connections can reduce obstructions to the walkway. Other issues arise if water spills reach the electrical components, producing possible shocks to the user. The system design does not provide any paths for water to reach the electronics, creating no shocking hazard. Moreover, a safety issue arises from unexpectedly fast cycling speeds, causing capacitors and integrated circuits to receive too much power and the possibility of explosions or even fire. The overvoltage and overcurrent protection circuits throughout the system prevent overloads on any of the electronics. When these protection circuits trigger, they create another safety issue from the physical resistance reducing to almost nothing. The user's pedaling speed increases under such conditions immensely, possibly causing feet to leave the pedals and muscles to cramp. The control system measures all voltages and currents and strives to reduce the possibility of such experiences. In the future, adding a possible warning message to the elliptical user interface increases safety.

This project reduces the carbon emissions into the atmosphere, so actually benefits the health and safety of the public. Since the air is cleaner, citizens in cities with a large amount of EHFEM systems could see real reductions in pollution. Reduction in pollution causes reduction in children with asthma, among other respiratory problems.

10. Social and Political

The EFHEM project does not present many social or political issues, since its design helps reduce the issue of global warming. The design, produced through university funds and student time, has no political or social effects. The manufacturing of most parts come from China, which may produce some resistance to activists striving for domestically made products. Politically, every citizen supports capturing of clean energy already produced. Instead of throwing

it into the atmosphere as heat, the EHFEM system captures it. Socially, it provides cleaner air in high density populations, since cities contain more gyms and normally higher pollution rates.

The project impacts almost everyone, but in a positive way: reducing carbon emissions. The direct stakeholders include the REC Center, users of the elliptical machines, and everyone near the energy production facilities. The REC center saves money from the systems, while the users receive the motivation of producing clean energy. Everyone who breathes air near non-renewable energy plants benefits from a decrease in energy production and therefore cleaner air to breathe. Producers of every integrated circuit in the design and the PCB manufacturers receive increases in revenue. Since California requires 33% of all energy to come from renewable sources by 2020, California benefits in reaching this goal more easily [34]. PGE also benefits from their requirement to meet these restrictions and the project simplifies issues with meeting peak demand. California and PGE have indirect stakes in this project.

Cal Poly, another direct stakeholder, benefits from articles and publicity showing it invests in industry leading solutions to climate change. The general population of college applicants hears about their support, and becomes more likely to both apply and accept admission to the university.

11. Development

This project requires attaining new knowledge in digital controls as well as feedforward design techniques to speed up response times. Feedforward systems produce stability issues, so I perform the appropriate stability analysis. I need to learn the Atmel software to implement the microcontroller as well. I learn a great deal designing the electrical system to write estimates down and determine the parts that work best for my application. Instead of having one unknown in a problem, many unknowns exist, as well as design tradeoffs between every choice made.

I learn more types of sources can help offer more insight into design of any system. I already use the internet, books, datasheets, and databases to find reliable information, but now I additionally use patent literature, IEEE journal articles, and past senior projects and theses to offer more specific knowledge as shown from the literature search in the References section. In addition, I determine the quality and accuracy of each source before citing it, to ensure I use only

the best information. In my literature search, I find many other companies producing products to solve the same problem the EHFEM system addresses. Learning about the competition allows me to shoot for performance levels above current systems. DeSando controls a DC-DC converter in a similar fashion to how I control one, so learning his approach and some problems he had to solve, simplifies my approach.

Appendix B. MATLAB Code Using Neural Network Toolbox

```
net = feedforwardnet(5,'trainlm');    %# hidden neuron, Leven-Marq
net.inputs{1}.size = 5;     %set number of input neurons
% net.layers{1}.transferFcn = 'logsig';
% net.layers{2}.transferFcn = 'logsig';

net.initFcn = 'initlay';
net.layers{1}.initFcn = 'initwb';
net.layers{2}.initFcn = 'initwb';
net.inputWeights{1,1}.initFcn = 'rands';
net.inputWeights{2,1}.initFcn = 'rands';
net.biases{1,1}.initFcn = 'rands';
net.biases{2,1}.initFcn = 'rands';
net = init(net);
% generate input/output pair
p1 = (20*rand(1,500));
p0 = (20*rand(4,500));

% Normalized
M = csvread('C:\Users\Alex\Documents\Senior Project\Trial_data1-
    CCMResults-norm_mean.csv',11,1,[11 1 43  9]);
N = csvread('C:\Users\Alex\Documents\Senior Project\Trial_data1-
    CCMResults.csv',1,14,[1 14 11 22]);

% normalized training data
dv_out = M(:,4)';
i_out  = M(:,2)';
v_in   = M(:,1)';
i_inv  = M(:,5)';
i_in   = M(:,3)';
t1 = M(:,9)';
t = M(:,9)';

%validation data
dv_out_V = N(:,5)';
i_out_V  = N(:,2)';
v_in_V   = N(:,1)';
i_inv_V  = N(:,6)';
i_in_V   = N(:,4)';

p2 =[dv_out;i_out;v_in;i_inv;i_in];
ps = 0:1:32;
p = p2;

net = configure(net, p2, t1);
initial_output = net(p);
net.trainParam.epochs = 100;
%net.performFcn = 'sse';
%net.trainParam.goal = 1e-07;    % sets MSE???
net = train(net,p2,t1);
test_output = net(p);

%plot desired output red, actual blue
figure(1)
```

```
plot(ps,t,'r',ps,initial_output,'b')
title('Before Training')
figure(2)
plot(ps,t,'r',ps,test_output,'b')
title('After Training')

%check MSE
%MSE = immse(t,test_output)    %either method works
MSE = mse(net,t,test_output)
```

Appendix C. Final MATLAB Neural Network Code without Toolboxes

```
% Max-min normalized data
M = csvread('C:\Users\Alex\Documents\Senior Project\Trial_data1-
CCMResults_new_out.csv',11,1,[11 1 378 6]);
input = M(:,1:4);
N = csvread('C:\Users\Alex\Documents\Senior Project\Trial_data1-
CCMResults_new_out.csv',379,1,[379 1 415 6]);
valid = N(:,1:4);

% Desired output of XOR
output = M(:,6);
outputv = N(:,6);
% Initialize the bias
bias = -1;

coeff = 0.04;            % Learning coefficient eta << 1 typically
iteration_limit = 40000;     % Number of learning iterations
error_threshold = 0.07; % Desired iteration endpoint
iterations = 0;    % initialization
scale = 1;    %scales activation function to make steeper
num_in = 4;
num_hid = 18;

% predetermine sizes for speed; removes warning
err = zeros(1,iteration_limit);
delta2 = zeros(num_hid,1);
H = zeros(num_hid,1);
x2 = zeros(num_hid,1);
Ht = zeros(num_hid,1);
x2t = zeros(num_hid,1);
Hv = zeros(num_hid,1);
x2v = zeros(num_hid,1);

%- Calculate weights randomly using shuffle.
rng('shuffle');    %ensure truly random numbers
weights = -1 + 2.*rand(num_hid,num_in);  %random weight matrix, -1 to 1
weightb = -1 + 2.*rand((num_hid+1),1);   %random bias matrix, -1 to 1
weighto = -1 + 2.*rand(1,num_hid);       %random output matrix, -1 to 1

for i = 1:iteration_limit
    out = zeros(368,1);
    numIn = length(input(:,1));    %detect input data size
%out = zeros(numIn,1); % doing this here breaks code
   r = randperm(numIn);
   for j = 1:numIn
      % Hidden layer
      for h = 1:num_hid
        H(h) = bias*weightb(h,1) + input(r(j),1)*weights(h,1)...
               + input(r(j),2)*weights(h,2) +
input(r(j),3)*weights(h,3)...
               + input(r(j),4)*weights(h,4);
        % edit number of terms to match number input neurons
        % Send data through hyperbolic tangent function
        x2(h) = (exp(2*scale*H(h))-1)/(exp(2*scale*H(h))+1);
```

```
        end
        % Output layer
        x3_1 = bias*weightb((num_hid+1),1);
        for h = 1:num_hid
           x3_1 = x3_1 + x2(h)*weighto(1,h);
        end
           % edit number of terms to match number of hidden neurons
        out(r(j)) = (exp(2*scale*x3_1)-1)/(exp(2*scale*x3_1)+1);

        % Adjust delta values of weights
        % For output layer:
        % delta(wi) = xi*delta,
        % tanh: delta = (1-actual output)*(1+actual output)*
        %    (desired output - actual output)
        delta3_1 = (1-out(r(j)))*(1+out(r(j)))*(output(r(j))-out(r(j)));
        % Propagate the delta backwards into hidden layers
        for h = 1:num_hid
          delta2(h) = (1-x2(h))*(1+x2(h))*weighto(1,h)*delta3_1;
        end
        % Add weight changes to original weights
        % And use the new weights to repeat process.
        % delta weight = coeff*x*delta
        %
        % Bias cases
        for h = 1:num_hid
          weightb(h,1) = weightb(h,1) + coeff*bias*delta2(h);
        end
        weightb((num_hid+1),1) = weightb((num_hid+1),1) + ...
            coeff*bias*delta3_1;

        for k = 1:num_in % input layer size
          % Input cases to neurons
          for h = 1:num_hid
            weights(h,k) = weights(h,k) + coeff*input(r(j),k)*delta2(h);
          end
        end

        for h = 1:num_hid % hidden layer size
              weighto(1,h) = weighto(1,h) + coeff*x2(h)*delta3_1;
        end
    end
    iterations = iterations + 1;
    error = output - out;
    err(i) = sqrt(sum(error.*error));
    if err(i) < error_threshold
        break;
    end

end

err(i)
% error
MSE = immse(out,output)
figure(1);
plot([out output]);
```

```
iterations

% plot error vs. iterations to see if approaching some small value
    % used to set neuron #, coeff, etc.
x = 1:1:iterations;
figure(2);
plot(x,err(1:iterations));
xlabel('iteration');
ylabel('Error(desired-output)');

outt = zeros(368,1);
% Training verification
for j = 1:numIn
      % Hidden layer
      for h = 1:num_hid
        Ht(h) = bias*weightb(h,1) + input(j,1)*weights(h,1)...
               + input(j,2)*weights(h,2) + input(j,3)*weights(h,3)...
               + input(j,4)*weights(h,4);
        x2t(h) = (exp(2*scale*Ht(h))-1)/(exp(2*scale*Ht(h))+1);
      end
      % Output layer
      x3_1t = bias*weightb((num_hid+1),1);
      for h = 1:num_hid
          x3_1t = x3_1t + x2t(h)*weighto(1,h);
      end
      outt(j) = (exp(2*scale*x3_1t)-1)/(exp(2*scale*x3_1t)+1);
end
errort = output - outt;

MSEt = immse(outt,output)
figure(3);
plot([outt output]);

figure(4);
plot(errort);

numInv = length(valid(:,1));    %detect input data size
outv = zeros(37,1);
% Validation
for j = 1:numInv
      % Hidden layer
      for h = 1:num_hid
        Hv(h) = bias*weightb(h,1) + valid(j,1)*weights(h,1)...
               + valid(j,2)*weights(h,2) + valid(j,3)*weights(h,3)...
               + valid(j,4)*weights(h,4);
        x2v(h) = (exp(2*scale*Hv(h))-1)/(exp(2*scale*Hv(h))+1);
      end
      % Output layer
      x3_1v = bias*weightb((num_hid+1),1);
      for h = 1:num_hid
          x3_1v = x3_1v + x2v(h)*weighto(1,h);
      end
      outv(j) = (exp(2*scale*x3_1v)-1)/(exp(2*scale*x3_1v)+1);
end
```

```
MSEv = immse(outv,outputv)
figure(5);
plot([outv outputv]);
```

Appendix D. Microcontroller Code for Monitoring Voltages and Currents

This appendix contains the microcontroller code used while only monitoring voltages and currents. At one point, the code took the median over 3 data points, which essentially eliminated noise. The median code appears commented out here after deciding against this approach due to slower response times.

```
 /**
 * Median code commented out. Index edits and variable changes are needed to
 * insert median back in.
 */
/**
 * \mainpage User Application template doxygen documentation
 *
 * \par Empty user application template
 *
 * This is a bare minimum user application template.
 *
 * For documentation of the board, go \ref group_common_boards "here" for a link
 * to the board-specific documentation.
 *
 * \par Content
 *
 * -# Include the ASF header files (through asf.h)
 * -# Minimal main function that starts with a call to board_init()
 * -# Basic usage of on-board LED and button
 * -# "Insert application code here" comment
 */
/*
 * Include header files for all drivers that have been imported from
 * Atmel Software Framework (ASF).
 */
/*
 * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel
 * Support</a>
 */
#include <asf.h>
#include <stdbool.h>
#include <stdio.h>
#define ADC_CLOCK 22000000
#define PWM_FREQ 1000000
#define PWM_PERIOD 5
#define FALSE 0
#define TRUE !(FALSE)
#define PWM_DAC IOPORT_CREATE_PIN(PIOA, 23)
#define TIMER_TC0 IOPORT_CREATE_PIN(PIOA, 29)
#define TIMER_OUT IOPORT_CREATE_PIN(PIOA, 24)
#define SHADOW_SIZE 6000 // 12300 almost fills RAM takes < 0.13 seconds of data
#define MEDIAN_COUNT 3 // take median of 3 ADC readings to mitigate noise
#define MEDIAN_LOCATION ((MEDIAN_COUNT-1)/2)  // assumes MEDIAN_COUNT is odd
#define TC_CAPTURE_TIMER_SELECTION TC_CMR_TCCLKS_TIMER_CLOCK3

int counter;
uint32_t j = 0;
```

```
//uint32_t k = 0;  // median index
_Bool FEED = FALSE;
_Bool STORE = FALSE;
//_Bool MEDIAN = FALSE;
_Bool VALUE = 0;
uint16_t v_in_default;
uint16_t i_in_default;
uint16_t v_out_default;
uint16_t i_out_default;
//uint16_t v_in_med;
//uint16_t i_in_med;
//uint16_t v_out_med;
//uint16_t i_out_med;
//uint16_t v_in[MEDIAN_COUNT] = {0};
//uint16_t i_in[MEDIAN_COUNT] = {0};
//uint16_t v_out[MEDIAN_COUNT] = {0};
//uint16_t i_out[MEDIAN_COUNT] = {0};
uint16_t v_in_shadow[SHADOW_SIZE] = {0};
uint16_t i_in_shadow[SHADOW_SIZE] = {0};
uint16_t v_out_shadow[SHADOW_SIZE] = {0};
uint16_t i_out_shadow[SHADOW_SIZE] = {0};
uint32_t timestamp_shadow[SHADOW_SIZE] = {0};

//volatile int duty_calc=0;//debugging, check internal ADC readings match terminal

pwm_channel_t pwm_channel_instance;

static void configure_rtt(void)
{
    uint32_t ul_previous_time;
    // configure RTT for an interrupt per slow clock (32.768 kHz) period
    rtt_init(RTT, 1);
    ul_previous_time = rtt_read_timer_value(RTT);
    while (ul_previous_time == rtt_read_timer_value(RTT));
}

// configure UART console
static void configure_console(void)
{
    const usart_serial_options_t uart_serial_options = {
       .baudrate = CONF_UART_BAUDRATE,
       .paritytype = CONF_UART_PARITY};
    // configure console UART
    sysclk_enable_peripheral_clock(CONSOLE_UART_ID);
    pio_configure_pin_group(CONF_UART_PIO, CONF_PINS_UART, CONF_PINS_UART_FLAGS);
    stdio_serial_init(CONSOLE_UART, &uart_serial_options);
}

/*Use to find median */
void swap(uint16_t *p,uint16_t *q)
{
    uint16_t t;

    t=*p;
    *p=*q;
    *q=t;
}
```

```
void sort(uint16_t arr[])
{
    int l,m;

    for(l=0;l<MEDIAN_COUNT;l++)
    {
        for(m=0;m<MEDIAN_COUNT-l-1;m++)
        {
            if(arr[m]>arr[m+1])
            swap(&arr[m],&arr[m+1]);
        }
    }
}

/** * ADC Interrupt Handler * Reads in from 4 ADC channels */
void ADC_Handler(void)
{
    // Check the ADC conversion status
    if ((adc_get_status(ADC) & ADC_IER_EOC5) == ADC_IER_EOC5)
    {
        //adc_disable_interrupt(ADC, ADC_IDR_EOC5); // slows interrupt for UART
        if (STORE == TRUE)
        {
            /* store data locally before output */
            /* comment out channel disable/enables in this ISR */

            /* Get latest digital data value from ADC and can be used by
             * application */
            // for median code
            //v_in[k] = adc_get_channel_value(ADC, ADC_CHANNEL_0);
            //i_in[k] = adc_get_channel_value(ADC, ADC_CHANNEL_1);
            //v_out[k] = adc_get_channel_value(ADC, ADC_CHANNEL_4);
            //i_out[k] = adc_get_channel_value(ADC, ADC_CHANNEL_5);

            v_in_shadow[j] = adc_get_channel_value(ADC, ADC_CHANNEL_0);
            i_in_shadow[j] = adc_get_channel_value(ADC, ADC_CHANNEL_1);
            v_out_shadow[j] = adc_get_channel_value(ADC, ADC_CHANNEL_4);
            i_out_shadow[j] = adc_get_channel_value(ADC, ADC_CHANNEL_5);
            timestamp_shadow[j] = (unsigned int) tc_read_cv(TC,
                TC_CHANNEL_CAPTURE);
            j++;
            //k++;
            //if (k >= MEDIAN_COUNT)
            //{
                //sort(v_in);
                //v_in_shadow[j] = v_in[MEDIAN_LOCATION];
                //sort(i_in);
                //i_in_shadow[j] = i_in[MEDIAN_LOCATION];
                //sort(v_out);
                //v_out_shadow[j] = v_out[MEDIAN_LOCATION];
                //sort(i_out);
                //i_out_shadow[j] = i_out[MEDIAN_LOCATION];
                //timestamp_shadow[j] = (unsigned int) tc_read_cv(TC,
                //    TC_CHANNEL_CAPTURE);
                //j++;
                //k = 0;
            //}
```

```
            if (j >= SHADOW_SIZE)
            {
                STORE = FALSE;
                puts("Stored\r");
                tc_disable_interrupt(TC, TC_CHANNEL_CAPTURE, TC_SR_COVFS);
                j = 0;
            }
        }
        //else if (MEDIAN == TRUE)
        //{
            ////adc_disable_interrupt(ADC, ADC_IDR_EOC5);
            //// Get latest value from ADC and can be used by application
            //v_in[k] = adc_get_channel_value(ADC, ADC_CHANNEL_0);
            //i_in[k] = adc_get_channel_value(ADC, ADC_CHANNEL_1);
            //v_out[k] = adc_get_channel_value(ADC, ADC_CHANNEL_4);
            //i_out[k] = adc_get_channel_value(ADC, ADC_CHANNEL_5);
            //k++;
            //if (k >= MEDIAN_COUNT)
            //{
                //MEDIAN = FALSE;
                //puts("Median\r");
                //k = 0;
            //}
        //}
        /*Data capture only - Comment out if controlling hardware*/
        //else if (FEED == TRUE)
        //{
            //printf("%04u %04u %04u %04u\n\r", v_in_default, i_in_default,
                v_out_default, i_out_default);
        //}
        else
        {
            v_in_default = adc_get_channel_value(ADC, ADC_CHANNEL_0);
            i_in_default = adc_get_channel_value(ADC, ADC_CHANNEL_1);
            v_out_default = adc_get_channel_value(ADC, ADC_CHANNEL_4);
            i_out_default = adc_get_channel_value(ADC, ADC_CHANNEL_5);
        }

        adc_start(ADC);
        //adc_enable_interrupt(ADC, ADC_IER_EOC5); // slows interrupt for UART

        /* uncomment for RTT timing */
        /* time = rtt_read_timer_value(RTT) / [counter=10000] / [SCLK=32.768kHz]*/
        //if(++counter == 10000)
        //{
        //printf("Time: %u\n\r", (unsigned int)rtt_read_timer_value(RTT));
        //counter = 0;
        //configure_rtt();
        //}
    }
}

// configure ADC
static void adc_setup(void)
{
    sysclk_enable_peripheral_clock(ID_ADC);
    adc_init(ADC, sysclk_get_cpu_hz(), ADC_CLOCK, 8);
```

```
    adc_configure_timing(ADC, 0, ADC_SETTLING_TIME_3, 1);
    adc_set_resolution(ADC, ADC_MR_LOWRES_BITS_10);
    adc_enable_channel(ADC, ADC_CHANNEL_0);
    adc_enable_channel(ADC, ADC_CHANNEL_1);
    adc_enable_channel(ADC, ADC_CHANNEL_4);
    adc_enable_channel(ADC, ADC_CHANNEL_5);
    adc_disable_anch(ADC); //sync Gain, Offset and Differential mode values to Ch0
    //set gain of 2 for all channels
    adc_set_channel_input_gain(ADC, ADC_CHANNEL_0, ADC_GAINVALUE_2);
    NVIC_EnableIRQ(ADC_IRQn);
    adc_enable_interrupt(ADC, ADC_IER_EOC5);
    adc_configure_trigger(ADC, ADC_TRIG_SW, 0);
    //adc_configure_trigger(ADC, ADC_TRIG_SW, ADC_MR_FREERUN_ON);
}

/* PWM Handler * updates PWM duty cycle */
void PWM_Handler(void)
{
    static uint32_t ul_duty = 0;
    uint32_t ul_status;
    static uint8_t uc_count = 0;
    static uint8_t uc_flag = 1;
    ul_status = pwm_channel_get_interrupt_status(PWM);
    if ((ul_status & PWM_CHANNEL_0) == PWM_CHANNEL_0)
    {
        uc_count++;
        if (uc_count == 10)
        {
           if (uc_flag)
                {
                    ul_duty++;
                    if (ul_duty == 100)
                        uc_flag = 0;
                }
                else
                {
                    ul_duty--;
                    if (ul_duty == 0)
                        uc_flag = 1;
                }
                uc_count = 0;
                pwm_channel_instance.channel = PWM_CHANNEL_0;
                pwm_channel_update_duty(PWM, &pwm_channel_instance, ul_duty);
        }
    }
}

static void pwm_setup (void)
{
    pio_configure_pin(PWM_DAC, PIO_TYPE_PIO_PERIPH_B);
    sysclk_enable_peripheral_clock(ID_PWM);
    pwm_channel_disable(PWM, PWM_CHANNEL_0);
    pwm_clock_t clock_setting = {
        .ul_clka = PWM_FREQ * PWM_PERIOD,
        .ul_clkb = 0,
        .ul_mck = sysclk_get_cpu_hz()
    };
```

```
    pwm_init(PWM, &clock_setting);
    pwm_channel_instance.ul_prescaler = PWM_CMR_CPRE_CLKA;
    pwm_channel_instance.polarity = PWM_HIGH;
    pwm_channel_instance.ul_period = PWM_PERIOD;
    pwm_channel_instance.ul_duty = 0;
    pwm_channel_instance.channel = PWM_CHANNEL_0;
    pwm_channel_init(PWM, &pwm_channel_instance);
    pwm_channel_enable_interrupt(PWM, PWM_CHANNEL_0, 0);
}

// Configure TC TC_CHANNEL_CAPTURE in capture operating mode
static void tc_capture_initialize(void)
{
    sysclk_enable_peripheral_clock(ID_TC_CAPTURE);
    // Initialize TC to capture mode
    tc_init(TC, TC_CHANNEL_CAPTURE,TC_CAPTURE_TIMER_SELECTION);
}

// Interrupt handler for the TC TC_CHANNEL_CAPTURE
void TC_Handler(void)
{
    // do nothing - needs this in handler to work properly
    if ((tc_get_status(TC, TC_CHANNEL_CAPTURE) & TC_SR_COVFS) == TC_SR_COVFS)
    {
        /*uncomment to output timer frequency to Ext 1 pin 5*/
        /*timer frequency = [frequency on pin 5] * [2^16=counter overflow value]*/
        //VALUE = !VALUE;
        //ioport_set_pin_level(TIMER_OUT, VALUE);
    }
}

static void tc_setup (void)
{
    pio_configure_pin(TIMER_TC0, PIO_TYPE_PIO_PERIPH_B);
    tc_capture_initialize();
    NVIC_DisableIRQ(TC_IRQn);
    NVIC_ClearPendingIRQ(TC_IRQn);
    NVIC_SetPriority(TC_IRQn, 0);
    NVIC_EnableIRQ(TC_IRQn);
}

int main (void)
{
    int PWM_count = 0;
    int index;

    sysclk_init();
    board_init();

    // disable watchdog
    WDT->WDT_MR = WDT_MR_WDDIS;

    // to measure timer frequency only
    ioport_init();
    ioport_set_pin_dir(TIMER_OUT, IOPORT_DIR_OUTPUT);

    // insert application code here, after the board has been initialized
```

```
configure_console();
adc_setup();
pwm_setup();
tc_setup();

// output example information
puts("Hello World!\r");
printf("Clock: %u\n\r", sysclk_get_cpu_hz());
counter = 0;
adc_start(ADC);
pwm_channel_enable(PWM, PWM_CHANNEL_0);

char input;

while (1)
{
    input = getchar();
    switch(input)
    {
        // test serial connection
        case 'a':
        printf("%c\n\r", input);
        ioport_toggle_pin_level(LED_0_PIN);
        break;

        //Print out the 4 ADC values
        case 'p':
        printf("ADC Values: %u %u %u %u\n\r", v_in_default, i_in_default,
            v_out_default, i_out_default);
        //duty_calc = (int) ((2.6/1024*V_in*2)/3.3*100);
        //pwm_channel_update_duty(PWM, &pwm_channel_instance, duty_calc);
        break;

        //case 'm':
        //MEDIAN = TRUE;
        //break;

        //case 'w':
        //for(index=0;index<MEDIAN_COUNT;index++)
        //{
            //printf("%u %u %u %u\n\r",v_in[index],i_in[index],v_out[index],
            //    i_out[index]);
        //}
        //sort(v_in);
        //v_in_med = v_in[MEDIAN_LOCATION];
        //sort(i_in);
        //i_in_med = i_in[MEDIAN_LOCATION];
        //sort(v_out);
        //v_out_med = v_out[MEDIAN_LOCATION];
        //sort(i_out);
        //i_out_med = i_out[MEDIAN_LOCATION];
        //for(index=0;index<MEDIAN_COUNT;index++)
        //{
            //printf("Sort %u %u %u %u\n\r", v_in[index], i_in[index],
            //    v_out[index], i_out[index]);
        //}
        //printf("Med %u %u %u %u\n\r", v_in_med, i_in_med, v_out_med,
```

```
                //    i_out_med);
                //break;

                //Data capture only
                //Comment out if controlling hardware
                //case 'f':
                //if (FEED == FALSE)
                //{
                    //configure_rtt();
                    //printf("Time: %u\n\r", (unsigned int)rtt_read_timer_value(RTT));
                //}
                //FEED = !FEED;
                //if (FEED == FALSE)
                //{
                    //printf("Time: %u\n\r",(unsigned int) rtt_read_timer_value(RTT));
                    //configure_rtt();
                //}
                //break;

                // store SHADOW_SIZE number of ADC values at max speed
                case 's':
                tc_enable_interrupt(TC, TC_CHANNEL_CAPTURE, TC_SR_COVFS);
                tc_start(TC, TC_CHANNEL_CAPTURE);
                STORE = TRUE;
                break;

                // output stored data to terminal
                case 'o':
                for (index = 0; index < SHADOW_SIZE; index++)
                {
                    printf("%04u %04u %04u %04u %u\n\r", v_in_shadow[index],
                        i_in_shadow[index], v_out_shadow[index], i_out_shadow[index],
                        timestamp_shadow[index]);
                }
                break;

                //Output PWM voltage to test
                case 'v':
                PWM_count++;
                if (PWM_count == 1)
                    pwm_channel_update_duty(PWM, &pwm_channel_instance, 1);
                else if (PWM_count == 2)
                    pwm_channel_update_duty(PWM, &pwm_channel_instance, 4);
                else
                {
                    pwm_channel_update_duty(PWM, &pwm_channel_instance, 0);
                    PWM_count = 0;
                }
                puts("PWM Duty Cycle Changed\r");
                break;

                default:
                ioport_set_pin_level(LED_0_PIN, !LED_0_ACTIVE);
            }
        }
    }
```

Appendix E. Microcontroller Code Final ANN Implementation

```
/**
 * ANN Network Code - press 'r' to enter network mode
 * Excess deleted for readability
 * Some features explored during experiments are commented out since didn't show
 * significant improvements.
 */
/**
 * \mainpage User Application template doxygen documentation
 *
 * \par Empty user application template
 *
 * This is a bare minimum user application template.
 *
 * For documentation of the board, go \ref group_common_boards "here" for a link
 * to the board-specific documentation.
 *
 * \par Content
 *
 * -# Include the ASF header files (through asf.h)
 * -# Minimal main function that starts with a call to board_init()
 * -# Basic usage of on-board LED and button
 * -# "Insert application code here" comment
 *
 */
/*
 * Include header files for all drivers that have been imported from
 * Atmel Software Framework (ASF).
 */
/*
 * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel
 * Support</a>
 */
#include <asf.h>
#include <stdbool.h>
#include <stdio.h>
#include <math.h>
#include <arm_math.h> // needed for float32_t typedef
#define ADC_CLOCK 22000000
#define PWM_FREQ 1000000
#define PWM_PERIOD 50
#define FALSE 0
#define TRUE !(FALSE)
#define PWM_DAC IOPORT_CREATE_PIN(PIOA, 23)
#define TIMER_TC0 IOPORT_CREATE_PIN(PIOA, 29)
#define TIMER_OUT IOPORT_CREATE_PIN(PIOA, 24)
#define TC_CAPTURE_TIMER_SELECTION TC_CMR_TCCLKS_TIMER_CLOCK3

#define NUM_HID 18
#define NUM_IN 4
#define BIAS -1
#define ETA 0.01 // learning coefficient

// Define resistor values (kOhms), reducing math in ANN loop
#define R2 270 // top feedback DC-DC converter
#define R1 10  // bottom feedback
```

```
#define R3 5.6 // isolation resistor
#define R4 10  // PWM R of Low-pass filter

uint16_t p = 0;

float32_t r1_par_r2;
float32_t r1_par_r2r3;
float32_t a, b;
float32_t c;

float input[NUM_IN] = {0};
float32_t H[NUM_HID] = {0};
float32_t x2[NUM_HID] = {0};
float32_t x3_1;
float32_t out;
float32_t weightb[(NUM_HID+1)] = {-1.1289, -0.4050,  0.5334, -0.9252, -0.1694,
                                  -0.9923,  0.2232, -1.3877, -0.1787, -2.3424,
                                  -0.4919,  1.2214,  0.8009,  0.0282, -0.0374,
                                   0.0108,  0.3165, -0.2409,
                                  -0.0010}; // output neuron bias in last index
float32_t weights[NUM_HID][NUM_IN] = { {-0.8723, -0.4713, -0.4880, -1.0827},
                                       { 0.5521, -0.1729,  0.6203,  0.4431},
                                       { 1.0083,  0.4620, -0.1606, -0.4845},
                                       {-0.5788,  0.0004,  1.0885, -0.7678},
                                       {-0.0694, -0.4512,  0.2015, -0.8104},
                                       { 0.2729, -0.4276, -0.2746, -0.3278},
                                       {-0.8403,  0.9638,  0.1556, -0.5760},
                                       { 0.2155, -0.1364,  0.8861, -0.3022},
                                       { 0.0820,  0.8244, -0.6873, -0.1296},
                                       { 0.1107, -0.1501,  0.0814,  2.1392},
                                       { 0.0859,  0.6584, -0.1520, -0.4005},
                                       { 0.5715,  0.7339, -0.8521,  0.2963},
                                       { 0.0797,  0.1608,  0.4125, -0.0641},
                                       { 0.5481, -0.1283, -0.8556,  0.2243},
                                       {-0.7602, -0.2452,  0.1496,  0.1151},
                                       {-0.0841,  0.9460,  0.6869, -0.4156},
                                       { 0.6410,  0.1255, -0.3265, -0.4245},
                                       { 0.1217,  0.5005,  0.8240,  0.2095} };
float32_t weighto[NUM_HID] = {-0.8557, -0.3514,  0.2440, -0.4725,  1.1281,
                               0.8227,  0.3846,  1.2816, -0.0343, -2.0152,
                               0.3341, -0.9898, -0.3077,  0.5729, -0.1583,
                              -0.1631, -0.6019,  0.4574};
float32_t delta2 [NUM_HID];
float32_t delta3_1;
float32_t des_output;
uint8_t duty = PWM_PERIOD;
float32_t duty_NN;
float32_t v_out;
float32_t v_pwm;
float32_t v_feedback_des;
float32_t v_feedback_NN;
float32_t v_FB_norm;

int counter;
uint32_t j = 0;
volatile _Bool DONE = 1;
uint16_t v_in_default;
```

```
uint16_t i_in_default;
uint16_t v_out_default;
uint16_t i_out_default;
uint16_t v_in_buf;
uint16_t i_in_buf;
uint16_t v_out_buf;
uint16_t i_out_buf;

pwm_channel_t pwm_channel_instance;

static void configure_rtt(void)
{
    uint32_t ul_previous_time;
    // configure RTT for an interrupt per slow clock (32.768 kHz) period
    rtt_init(RTT, 1);
    ul_previous_time = rtt_read_timer_value(RTT);
    while (ul_previous_time == rtt_read_timer_value(RTT));
}

// configure UART console
static void configure_console(void)
{
    const usart_serial_options_t uart_serial_options = {
        .baudrate = CONF_UART_BAUDRATE,
        .paritytype = CONF_UART_PARITY
    };
    // configure console UART
    sysclk_enable_peripheral_clock(CONSOLE_UART_ID);
    pio_configure_pin_group(CONF_UART_PIO, CONF_PINS_UART, CONF_PINS_UART_FLAGS);
    stdio_serial_init(CONSOLE_UART, &uart_serial_options);
}

/*Start ANN Functions */
// normalize input neurons
void precalculate_constants (void)
{
    r1_par_r2 = (float32_t) (R1 * R2) / (R1 + R2);
    a = r1_par_r2 / (R4 + R3 + r1_par_r2); // V out of PWM scaler
    r1_par_r2r3 = (R1 * (R3 + R4)) / (R1 + R3 + R4);
    b = r1_par_r2r3 / (R2 + r1_par_r2r3); // Vout scaler
    c = (float32_t) R1 / (R2 + R1); // feedback resistor voltage divider
}
void normalize_inputs(void)
{
    //Normalized values=1 when Currents are 8.68 A (in) or 8.23 A (out),
        //Voltages to 70 V
    input[0] = (((float32_t) v_in_buf) / 454) - 1.0; // v_in_buf normalized
    input[1] = (((float32_t) i_in_buf) / 428) - 1.0; // i_in_buf normalized
    input[2] = (((float32_t) v_out_buf) / 454) - 1.0; // v_out_buf normalized
    input[3] = (((float32_t) i_out_buf) / 352) - 1.0; // i_out_buf normalized
}
// Correlate Feedback Voltage to Duty Cycle
void update_duty (void)
{
    v_feedback_NN = (out + 1) * 1.25;
    v_pwm = (v_feedback_NN - (v_out * b)) / a;
    duty = (uint8_t) ((v_pwm / 3.3 * PWM_PERIOD) + 0.5);
```

```
    if (duty < (.9*PWM_PERIOD) && (i_out_buf > 425))
        duty = PWM_PERIOD;
    if ((duty>PWM_PERIOD) || (v_out_buf>523) || (i_out_buf>480) || (i_in_buf>600))
        duty = PWM_PERIOD;
    else if (duty < 0)
        duty = 30;
    pwm_channel_update_duty(PWM, &pwm_channel_instance, duty);
}
// Calculate desired output
void update_output_des (void)
{
    v_feedback_des = (v_out * c); //v feedback desired
    des_output = (v_feedback_des / 1.25) - 1.0;
}
// update output
void update_out(void)
{
    uint8_t i_h;
    uint8_t i_x1;

    // Hidden layer
    for (i_h=0;i_h<NUM_HID;i_h++)
    {
        H[i_h] = BIAS * weightb[i_h];
        for(i_x1=0;i_x1<NUM_IN;i_x1++)
        {
            H[i_h] += input[i_x1] * weights[i_h][i_x1];
        }
        x2[i_h] = tanhf(H[i_h]); //hyperbolic tangent function
    }
    // Output layer
    x3_1 = BIAS * weightb[(NUM_HID+1)];
    for(i_h=0;i_h<NUM_IN;i_h++)
    {
        x3_1 += x2[i_h] * weighto[i_h];
    }
    out = tanhf(x3_1);
    update_duty();
    update_output_des();
}
// update weights and bias values
void update_WnB(void)
{
    uint8_t i_h;
    uint8_t i_x1;

    if ((v_out_buf > 411) && (v_out_buf < 425))
    {
        return; // if voltage regulated well don't correct
    }
    // Adjust delta values of weights for output layer:
    // delta(wi) = xi*delta
    delta3_1 = (1-out)*(1+out)*((des_output)-out);
    // Propagate the delta backwards into hidden layers
    for (i_h=0;i_h<NUM_HID;i_h++)
    {
        delta2[i_h] = (1-x2[i_h])*(1+x2[i_h])*weighto[i_h]*delta3_1;
```

```
    }
    // Add weight changes to original weights; use new weights to repeat process
    // delta weight = coeff*x*delta
    // Bias cases
    for (i_h=0;i_h<NUM_HID;i_h++)
    {
        weightb[i_h] += ETA * BIAS * delta2[i_h];
    }
    weightb[NUM_HID+1] += ETA * BIAS * delta3_1;
    for (i_x1=0;i_x1<NUM_IN;i_x1++)
    {
        for (i_h=0;i_h<NUM_IN;i_h++)
        {
            weights[i_h][i_x1] += ETA * input[i_x1] * delta2[i_h];
        }
    }
    for (i_h=0;i_h<NUM_HID;i_h++)
    {
        weighto[i_h] += ETA * x2[i_h] * delta3_1;
    }
}

/** * ADC Interrupt Handler * Reads in from 4 ADC channels */
void ADC_Handler(void)
{
    // Check the ADC conversion status
    if ((adc_get_status(ADC) & ADC_IER_EOC5) == ADC_IER_EOC5)
    {
        v_in_default = adc_get_channel_value(ADC, ADC_CHANNEL_0);
        i_in_default = adc_get_channel_value(ADC, ADC_CHANNEL_1);
        v_out_default = adc_get_channel_value(ADC, ADC_CHANNEL_4);
        i_out_default = adc_get_channel_value(ADC, ADC_CHANNEL_5);
        adc_start(ADC);

        if (!DONE)
        {
            v_in_buf = v_in_default;
            i_in_buf = i_in_default;
            v_out_buf = v_out_default;
            i_out_buf = i_out_default;
            DONE = TRUE;
        }

        /* uncomment for RTT timing */
        /* time = rtt_read_timer_value(RTT) / [counter=10000] / [SCLK=32.768kHz]*/
        //if(++counter == 10000)
        //{
        //printf("Time: %u\n\r", (unsigned int)rtt_read_timer_value(RTT));
        //counter = 0;
        //configure_rtt();
        //}
    }
}

// configure ADC
static void adc_setup(void)
{
```

```
    sysclk_enable_peripheral_clock(ID_ADC);
    adc_init(ADC, sysclk_get_cpu_hz(), ADC_CLOCK, 8);
    adc_configure_timing(ADC, 0, ADC_SETTLING_TIME_3, 1);
    adc_set_resolution(ADC, ADC_MR_LOWRES_BITS_10);
    adc_enable_channel(ADC, ADC_CHANNEL_0);
    adc_enable_channel(ADC, ADC_CHANNEL_1);
    adc_enable_channel(ADC, ADC_CHANNEL_4);
    adc_enable_channel(ADC, ADC_CHANNEL_5);
    adc_disable_anch(ADC); //sync Gain, Offset and Differential mode values to Ch0
    //set gain of 2 for all channels
    adc_set_channel_input_gain(ADC, ADC_CHANNEL_0, ADC_GAINVALUE_2);
    NVIC_EnableIRQ(ADC_IRQn);
    adc_enable_interrupt(ADC, ADC_IER_EOC5);
    adc_configure_trigger(ADC, ADC_TRIG_SW, 0);
    //adc_configure_trigger(ADC, ADC_TRIG_SW, ADC_MR_FREERUN_ON);
}

/* PWM Handler * updates PWM duty cycle */
void PWM_Handler(void)
{
    static uint32_t ul_duty = 0;
    uint32_t ul_status;
    static uint8_t uc_count = 0;
    static uint8_t uc_flag = 1;
    ul_status = pwm_channel_get_interrupt_status(PWM);
    if ((ul_status & PWM_CHANNEL_0) == PWM_CHANNEL_0)
    {
        uc_count++;
        if (uc_count == 10)
        {
            if (uc_flag)
            {
                ul_duty++;
                if (ul_duty == 100)
                    uc_flag = 0;
            }
            else
            {
                ul_duty--;
                if (ul_duty == 0)
                    uc_flag = 1;
            }
            uc_count = 0;
            pwm_channel_instance.channel = PWM_CHANNEL_0;
            pwm_channel_update_duty(PWM, &pwm_channel_instance, ul_duty);
        }
    }
}

static void pwm_setup (void)
{
    pio_configure_pin(PWM_DAC, PIO_TYPE_PIO_PERIPH_B);
    sysclk_enable_peripheral_clock(ID_PWM);
    pwm_channel_disable(PWM, PWM_CHANNEL_0);
    pwm_clock_t clock_setting = {
        .ul_clka = PWM_FREQ * PWM_PERIOD,
        .ul_clkb = 0,
```

```
        .ul_mck = sysclk_get_cpu_hz()
    };
    pwm_init(PWM, &clock_setting);
    pwm_channel_instance.ul_prescaler = PWM_CMR_CPRE_CLKA;
    pwm_channel_instance.polarity = PWM_HIGH;
    pwm_channel_instance.ul_period = PWM_PERIOD;
    pwm_channel_instance.ul_duty = PWM_PERIOD; // start PWM driving voltage down
    pwm_channel_instance.channel = PWM_CHANNEL_0;
    pwm_channel_init(PWM, &pwm_channel_instance);
    pwm_channel_enable_interrupt(PWM, PWM_CHANNEL_0, 0);
}

// Configure TC TC_CHANNEL_CAPTURE in capture operating mode
static void tc_capture_initialize(void)
{
    sysclk_enable_peripheral_clock(ID_TC_CAPTURE);
    // Initialize TC to capture mode
    tc_init(TC, TC_CHANNEL_CAPTURE,TC_CAPTURE_TIMER_SELECTION);
}

// Interrupt handler for the TC TC_CHANNEL_CAPTURE
void TC_Handler(void)
{
    // do nothing - needs this in handler to work properly
    if ((tc_get_status(TC, TC_CHANNEL_CAPTURE) & TC_SR_COVFS) == TC_SR_COVFS) {}
}

static void tc_setup (void)
{
    pio_configure_pin(TIMER_TC0, PIO_TYPE_PIO_PERIPH_B);
    tc_capture_initialize();
    NVIC_DisableIRQ(TC_IRQn);
    NVIC_ClearPendingIRQ(TC_IRQn);
    NVIC_SetPriority(TC_IRQn, 0);
    NVIC_EnableIRQ(TC_IRQn);
}

int main (void)
{
    uint8_t PWM_count = 0;
    int index;

    sysclk_init();
    board_init();

    // disable watchdog
    WDT->WDT_MR = WDT_MR_WDDIS;

    // to measure timer frequency only
    ioport_init();
    ioport_set_pin_dir(TIMER_OUT, IOPORT_DIR_OUTPUT);

    // insert application code here, after the board has been initialized
    configure_console();
    adc_setup();
    pwm_setup();
    tc_setup();
```

```
precalculate_constants();

// output example information
puts("Hello World!\r");
printf("Clock: %u\n\r", sysclk_get_cpu_hz());
counter = 0;
adc_start(ADC);
pwm_channel_enable(PWM, PWM_CHANNEL_0);

char letter;

while (1)
{
    letter = getchar();
    switch(letter)
    {
        // test serial connection
        case 'a':
        printf("%c\n\r", letter);
        ioport_toggle_pin_level(LED_0_PIN);
        //printf("e: %.50f\r\n", tanh(.5));
        break;

        //Print out the 4 ADC values
        case 'p':
        printf("ADC Values: %u %u %u %u\n\r", v_in_default, i_in_default,
            v_out_default, i_out_default);
        break;

        /** run ANN **/
        case 'r':
        puts("Running ANN\r");
        // fill buffers
        DONE = FALSE;
        while(!DONE){}
        //let converter reach nominal output voltage; soft start
        //while((v_out_buf < 405) && (duty > 0))
        //{
            //DONE = FALSE;
            //while(!DONE){}
            //duty--;
            //pwm_channel_update_duty(PWM, &pwm_channel_instance, duty);
            //delay_ms(10); //delay some time preventing duty cycle undershoot
        //}
        puts("main loop\r");
        while(1)
        {
            /* uncomment for RTT timing */
            /*time=rtt_read_timer_value(RTT)/[counter=1000]/[SCLK=32.768kHz]*/
            //if(++counter == 1000)
            //{
                //printf("Time: %u\n\r", (unsigned int)
                //    rtt_read_timer_value(RTT));
                //counter = 0;
                //configure_rtt();
            //}
            DONE = FALSE;
```

```
                while(!DONE){}
                v_out = (((float32_t) v_out_buf) + 0.9844) / 11.646;
                normalize_inputs();
                update_out();
                update_WnB();
            }
            break;

            //Output PWM voltage to test
            case 'v':
            PWM_count++;
            if (PWM_count == 1)
                pwm_channel_update_duty(PWM, &pwm_channel_instance,PWM_PERIOD/10);
            else if (PWM_count == 2)
                pwm_channel_update_duty(PWM, &pwm_channel_instance, PWM_PERIOD/2);
            else
            {
                pwm_channel_update_duty(PWM, &pwm_channel_instance, PWM_PERIOD);
                PWM_count = 0;
            }
            puts("PWM Duty Cycle Changed\r");
            break;

            default:
                ioport_set_pin_level(LED_0_PIN, !LED_0_ACTIVE);
        }
    }
}
```

Appendix F. Early ANN Simulations Discovering Errors and Training XOR

This appendix includes early ANN simulations, before confirming the feedback pin to the LT8705 chip acts as proportional gain. Here the simulations assume the feedback connects to a comparator providing only high or low values to the buck-boost logic. Throughout this appendix the training and validation data graphs follow the pattern that the red data series contains the ideal output, while the blue has the actual ANN output. If the graphs do not contain color, the ideal output data series corresponds to the data series with corners or clear discontinuities.

F.1 Performance of Neural Network for XOR Function

The first attempt to learn the XOR function only uses 2 hidden neurons since this theoretically works. These simulations often work, but the main function continually decreases mean-squared error, as the error plots show in working simulations. In the first run, the output neuron shows -0.0024, 0.9916, 0.9819, and -0.0469. These values come close to the ideal values, but the MSE of 6.5e-04 after 100,000 iterations still exceeds the desired level after more than enough iterations.

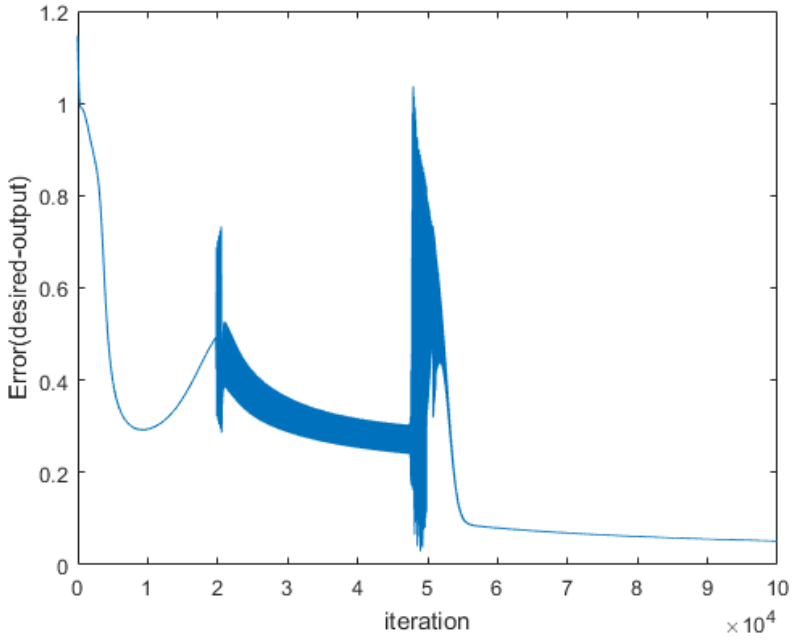

Figure 49. RMSE of Error after Each Iteration – Training XOR Function on Low Level Code with 2 Hidden Neurons, Example 1

Examining the error plotted against the number of iterations in Figure 49, the error moves up and down sporadically, which explains the large number of iterations without a smaller MSE. This run shows that more hidden neurons would help the neural network characterize the function quicker and closer to the theoretical values.

The next attempt to train the neural network increases the hidden neurons number to 4. The error threshold maintained at 0.01 still corresponds to a MSE of 2.5e-05. Running this code 30 times testing repeatability, each run achieved the error threshold before reaching the max iterations. The code also corrects updating of weights connecting from the input neurons to the hidden layer. Figure 50 shows the ideal error graph shape, where the error decreases with more iterations. Still the network could train with less than 72,620 iterations. This run demonstrates the hidden layer needs additional neurons to quickly characterize the XOR function.

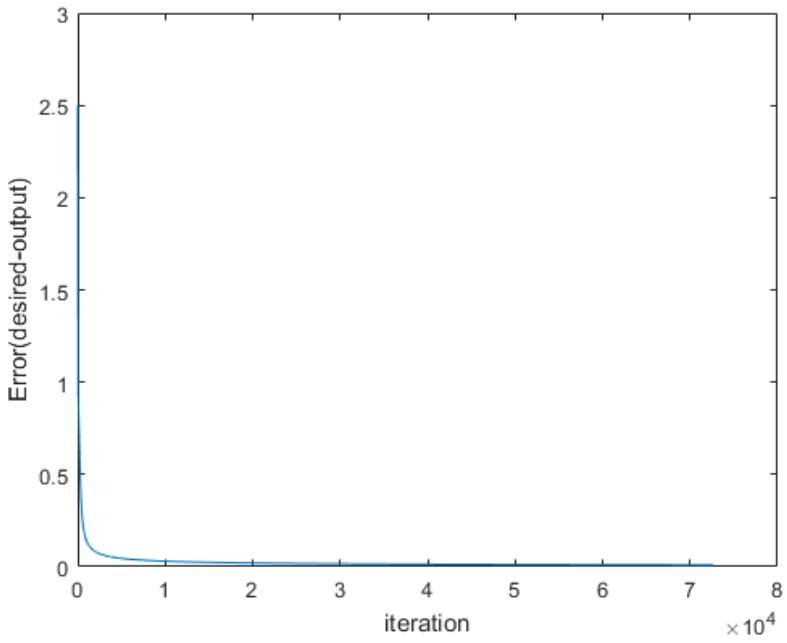

Figure 50. RMSE of Error after Each Iteration – Training XOR Function on Low Level Code with 4 Hidden Neurons

At this point, learning how to debug a simple function teaches me the same techniques that I need to employ to teach this same neural network to mimic my application specific data. Previously, I thought a steeper activation function would help produce binary output values as I want, but with this example, this seems unnecessary. To decrease convergence time for this

function, I could have tried optimizing the learning rate, also known as the η value, but I reserve this step for the actual dataset.

F.2 Performance of Neural Network for dVout Data with 4 Hidden Neurons

The next run aligns with a normal operation of neural networks with DC-DC converters. Usually the network regulates the voltage instead of using a controller chip. This run only uses 4 input neurons, excluding the inverter current. However, it uses all data points with load currents varying from 1 to 4 A as obtained from Andrew Forster's tests [9]. The error threshold increased to 0.1 allows the simulation to end sooner. I found this allows the simulation to reach this goal sooner, and the small MSE deems this test as adequate. The algorithm achieved a MSE of just 8.13e-05, while the only running 3,922 iterations.

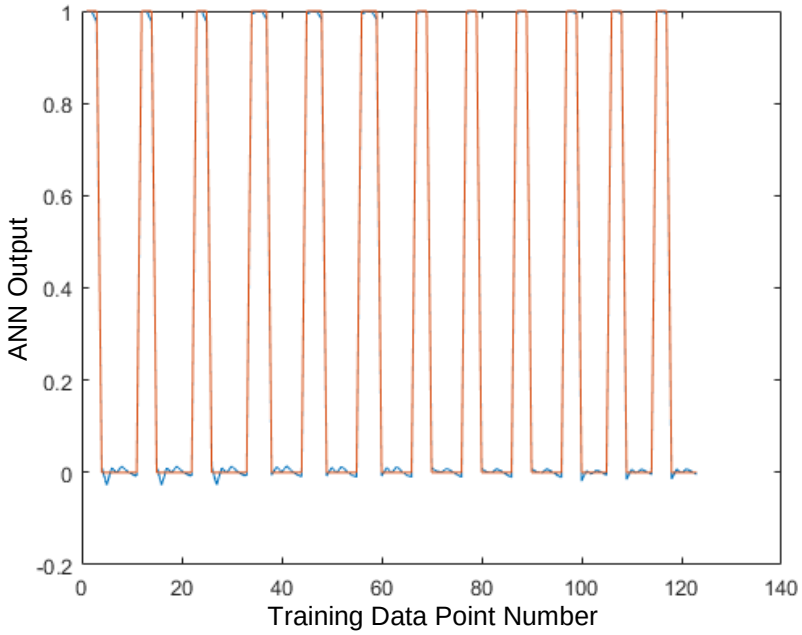

Figure 51. Training dVout Data on Low Level Code with 4 Hidden Neurons – Output

Figure 51 shows the output of the neural network after passing the inputs in. The red waveform, which has perfectly sharp corners, denotes the desired output, while the blue, which shows some ripple, denotes the output of the network. This simulation shows the ease with which this network controls the output voltage of the system. This response employs only 4 hidden

neurons, which shows the pattern is easily recognized by the network. To input the inverter current next, the input and hidden layers each must increase by one.

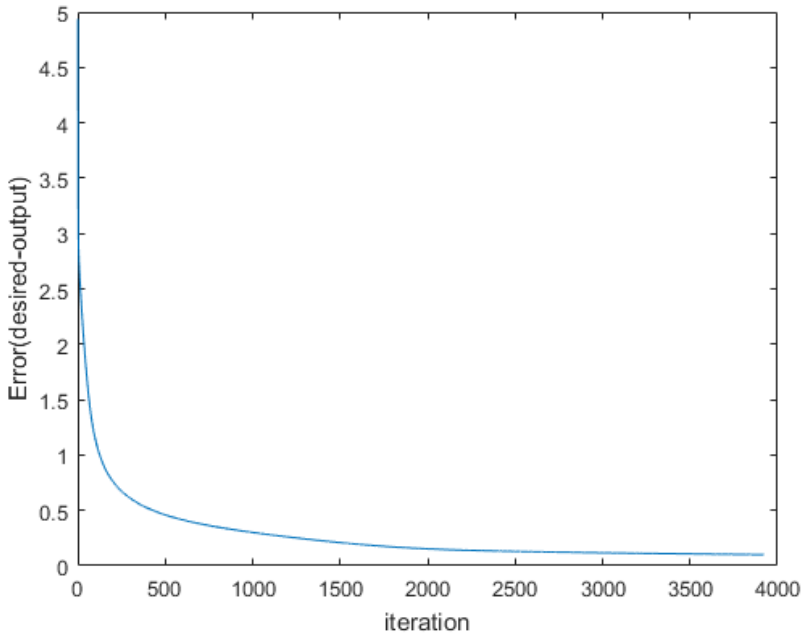

Figure 52. RMSE of Error after Each Iteration – Training dVout Data on Low Level Code with 4 Hidden Neurons

F.3 Performance of Neural Network for CCM Data with 5 Hidden Neurons

This revision on the neural network uses 5 input neurons for the input data, including the current into the inverter. The hidden layer usually exceeds the input layer in size, so the hidden layer is increased to keep it the same size as the input layer. Figure 53 shows the output of the neural network plotted on top of the desired output after reaching the error threshold of 0.1 at 20,122 iterations. The actual output does not have clean edges, since the training algorithm does not need to achieve perfection. This shows the neural network trained the specific data points to output the correct values. After running the code 10 times, the highest number of iterations hit a maximum at 100,000 and a minimum at 15,090. Therefore the network does not reach the error threshold once, but in all cases, the MSE is about 8.13e-05. The network output shows it clearly learns the right relationship between the inputs and outputs, supported by the small MSE. The neural network takes a long time to converge, however, so the next test investigates improvements.

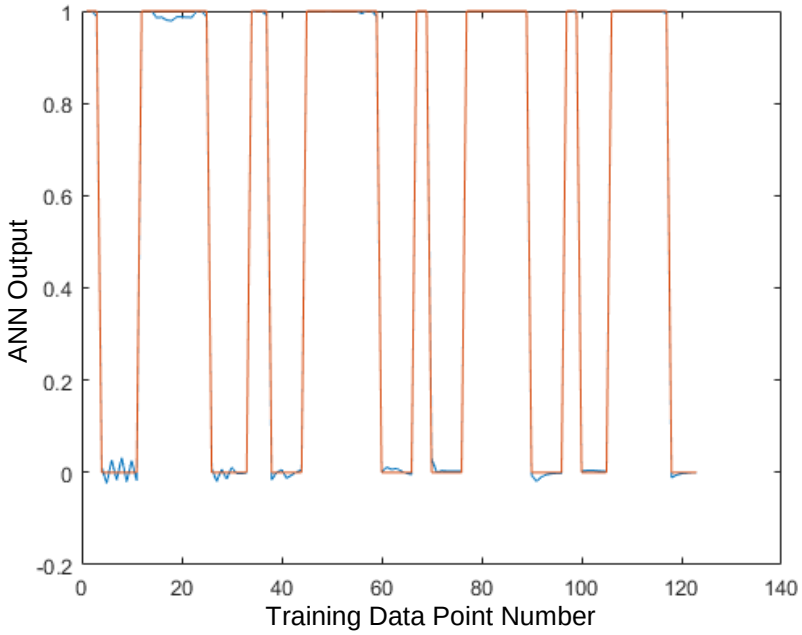

Figure 53. Training Dataset on Low Level Code with 5 Hidden Neurons – Output

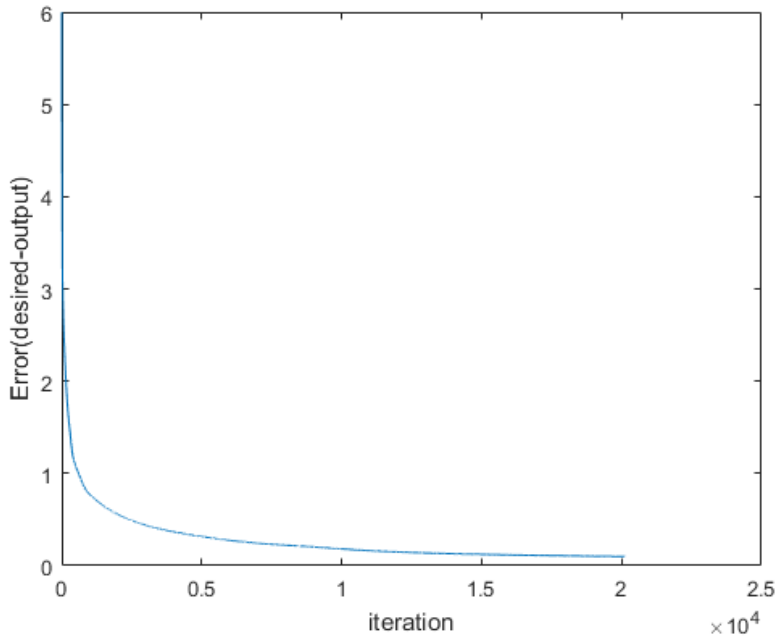

Figure 54. RMSE of Error after Each Iteration – Training Dataset on Low Level Code with 5 Hidden Neurons

F.4 Performance of Neural Network for CCM Data with 6 Hidden Neurons

This test increases the hidden neuron layer size by one neuron. The added neuron enables the neural network to fit the data easier in less time. As Figure 55 shows, the data fits; however, the maximum number of iterations needed to reach the same error threshold is 64,454. The network never reaches the maximum iteration limit, and the network achieves 15,375 as the least number of iterations. Interestingly, the minimum numbers for the network with 6 neurons lie slightly higher than with 5 neurons, but the iteration number varies across a smaller range with 6 neurons.

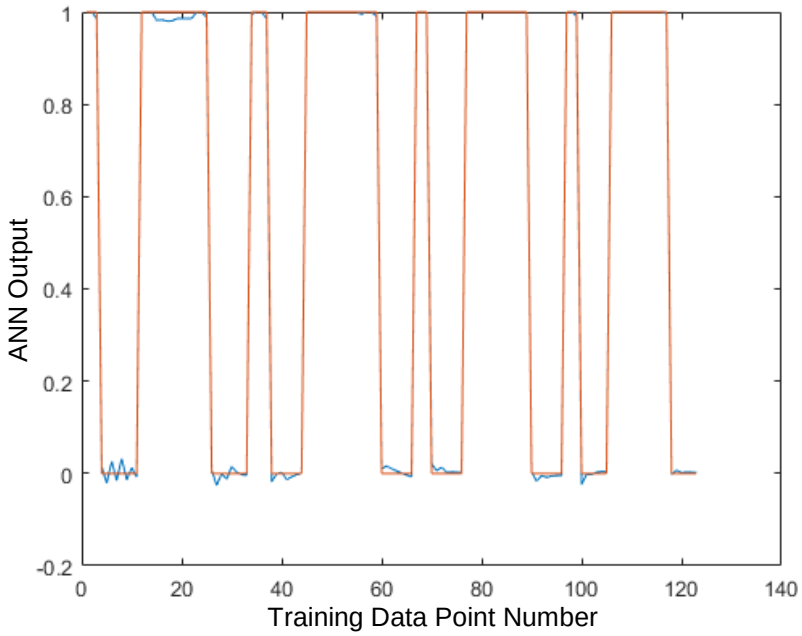

Figure 55. Training Dataset on Low Level Code with 6 Hidden Neurons – Output

At this point, optimization could continue, but this network does not test the real goal. The issue lies in that the output of the neural network only is tested at the point of training samples. Therefore the network always shows good output data. Moreover, the network could just memorize the order of the data samples and regurgitate them correctly without actually learning the desired patterns. The next section fixes these issues before optimizing the network to achieve the lowest number of iterations.

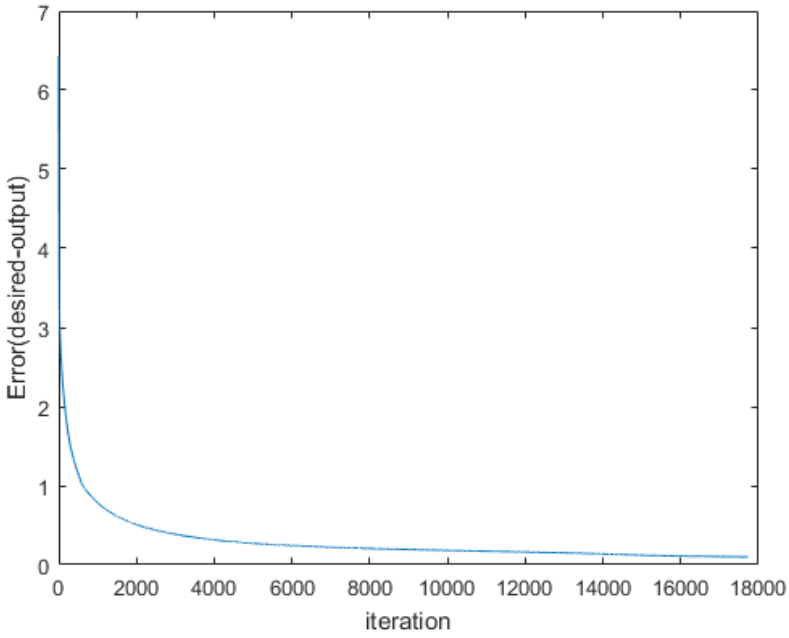

Figure 56. RMSE of Error after Each Iteration – Training Dataset on Low Level Code with 6 Hidden Neurons

F.5 ANN Implementation Using Low Level MATLAB Code and Validation Data

To train the network and verify the training properly, a validation data set must test the neural network ability. This test found a number of issues simultaneously. First, the use of training data to test the neural network makes the training much easier, since the neural network already minimizes the error at those specific data points. When creating a validation dataset, I realized the normalization and dVout data was flawed. The dVout data only includes values at steady state of the DC-DC converter, which does not include large voltage drops the inverter causes to the output voltage of the converter. The solution to this issue changes the output voltage on some of the data points. Some points must lie above and some below 0 V with an estimated possible range of 30 V. This means the normalization changes as well. Second, the code did not employ a random training data order. The network therefore learns the data, but could have just memorized the order of the data. To help ensure the network learns the actual patterns, the network presents training data in a random order to the input neurons for each iteration. Third, the code had some copy-paste bugs, which improperly updated neuron weights. To fix this problem and prevent this

mistake from repeating, this version adds for loops to step through the hidden neurons. The code becomes significantly shorter due to this edit, but completes the same tasks.

F.6 Low Level MATLAB Code with Validation Data Set and Small dVout Variation

The next iteration runs the validation data and corrected code. The validation data ideally passes through the neural network without updating the weights between neurons, but a copy-paste error left one variable name unchanged. Therefore the validation data output produced combined input data from the training data. This step produces more graphs during debugging and examination of training, but these additional graphs do not show any additional information. Rather, they just verify the neural proper network training at an intermediate step. Figure 57 shows the output of the neural network during the last iteration. During this run, the training dataset doubled from previous runs due to the duplicated data. In the duplicated set, dVout changes to the negative of the previous value so the network could learn that dVout may vary above or below the desired voltage at any data point. The change in dVout causes the input current of the DC-DC converter to update.

Additionally, this test reduces learning rate to 0.01. The low rate meant the simulation runs until it hits the iteration limit of 100,000 rather than the error threshold. Later simulation optimize the learning rate, but for now, choosing a small rate prevents the error from changing so much each data point that the output never converges to the desired output. Figure 58 shows this smaller learning rate decreasing the RMSE of the error slower than in previous iterations.

The validation data set, fit by the neural network and shown in Figure 59, produces a mean-squared error of 0.0046. This number likely would increase if the neural network reached the error threshold of 0.1, while the error only reaches 0.26. The validation data and the training data do not surpass 4 A, which correlates to the largest current data taken reliably from the DC-DC converter. The network learns these characteristics well, but training and validation data must encompass dVout more realistically for the next simulation.

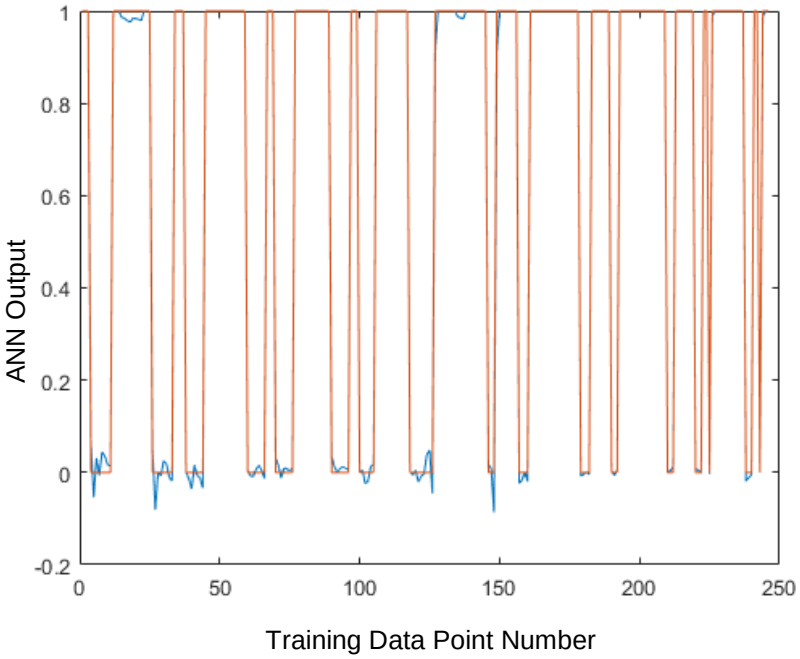

Figure 57. Training 246 Point Dataset on Low Level Code with 6 Hidden Neurons – Output

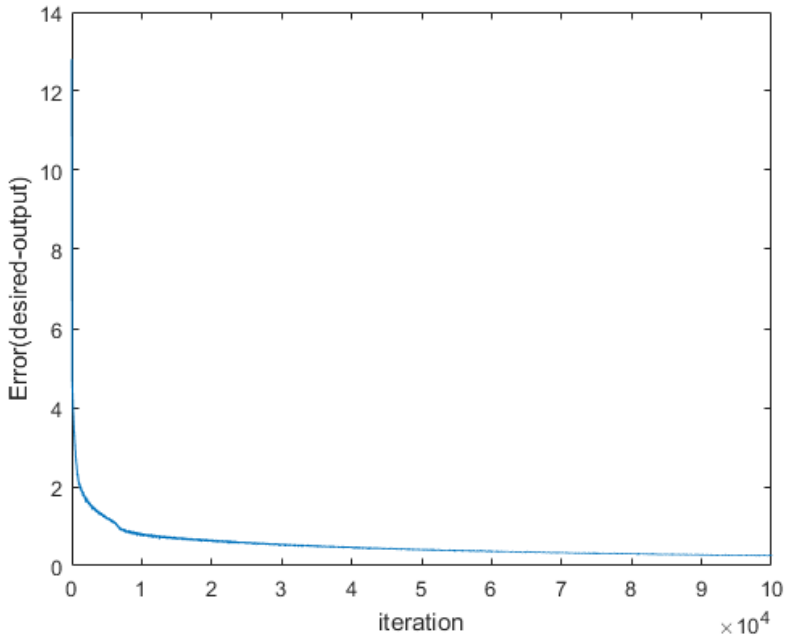

Figure 58. RMSE of Error after Each Iteration – Training 246 Point Dataset on Low Level Code with 6 Hidden Neurons

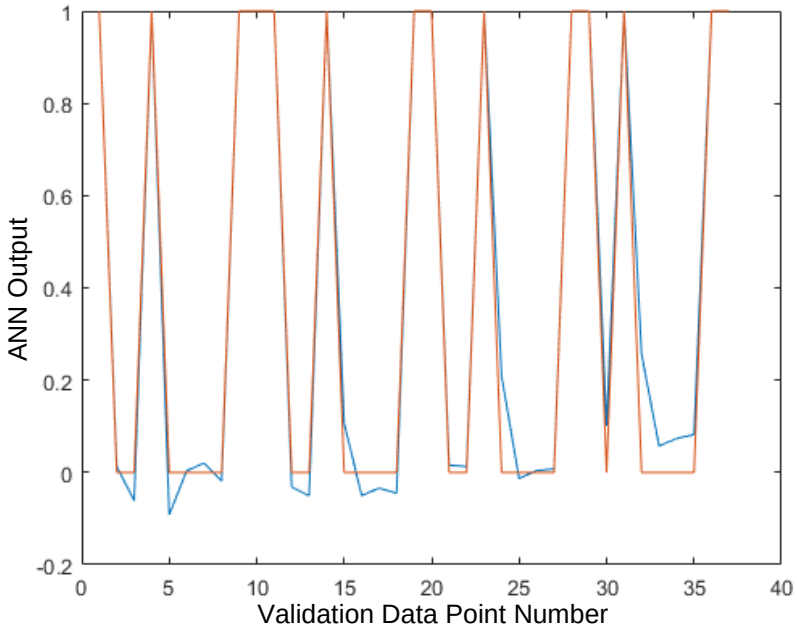

Figure 59. Training 246 Point Dataset on Low Level Code with 6 Hidden Neurons – Validation

The previous tests collectively prove the ANN code functions as desired and preserves the possibility of a working ANN controller in the EHFEM system.