Emulation of Aerospace Actuation Systems

Mid Project Report
Fall Semester 2021

Senior Design Team Members:
Kori Eliaz, Electrical Engineering
Jake Dorsett, Electrical Engineering
Dylan Gaub, Computer Engineering

Department of Electrical and Computer Engineering
Colorado State University
Fort Collins, Colorado 80523

Project Advisors: Dr. James Cale, Matt Heath
Approved by: Dr. James Cale, Matt Heath
Date of Approval: 12/8/2021

Abstract

The trend towards More Electric Aircraft (MEA) is expected to revolutionize the aerospace industry in the coming years, with promises of reduced weight, maintenance, and additional data analytics for real-time system prognostics compared to traditional drive systems. To verify these claims without incurring tremendous R&D costs to the customer, a controller-hardware in the loop (CHIL) platform allows for emulation of an expensive physical system using inexpensive software to test the behavior and benefits of an electromechanical thrust reverser actuation system (TRAS). Current TRAS systems use hydraulic and pneumatic mechanisms to effectuate thrust reversal on an airplane. These systems are lower efficiency, higher maintenance requirements, and slower response times than potential electromagnetically powered systems. According to Woodward, Inc. subject matter experts, the current state of the art deployment time for a TRAS system is 2.5 seconds. If an EM-TRAS system can be designed to match or reduce this deployment time, it would mark a generational change in the design of such systems and allow for the mass production of more environmentally conscious systems.

The Emulation of Aerospace Actuation Systems Project is part of an effort to validate a controller hardware-in-the-loop (CHIL) laboratory capability for use by CSU students and faculty. One application of this project is to validate an electrical drive alternative being developed by Woodward, Inc. to replace the pneumatically/hydraulically operated Thrust Reverser Actuation Systems (TRAS) currently deployed on commercial and military aircraft. This project interfaces a TI TMS320F28379D microcontroller with an OPAL-RT real-time processor to emulate several electrically driven aerospace actuation scenarios using a CHIL platform. Team members have programmed both the TI board and the OPAL-RT system in C to read/write synchronized digital and analog signals that reflect the behavior of a simplified DC machine system. In addition, the team is producing a detailed system model of the code and associated testbed hardware using the principles of Model Based Systems Engineering in Cameo Systems Modeler. The overall product is a well-documented system model and thoroughly tested code package that can be used by CSU students and faculty for CHIL demonstration and verification purposes on current projects and in the future.

A CHIL platform can help confirm the validity of any system by allowing a model of the system to be loaded onto a real-time processor using an industry-standard modeling platform like Simulink and tested for response time. So far, the work performed on this senior design project has established a working interface between the test case code loaded on a microcontroller and a real-time machine that can serve as a foundation for future tests, such as those required to validate the EM-TRAS system. Future work required to support this effort is the design of a graphical user interface (GUI) to interface with a PXI system that will allow users of the CHIL platform to easily manipulate inputs and view the corresponding behavior of their emulated system. Future work in the general field of MEA is further validation of different EM-TRAS designs involving various quantities of linear actuators, actuation lines, and other factors using CHIL platforms and mechanical testbeds.

Table of Contents

Chapter 1. INTRODUCTION **3**

Chapter 2. OBJECTIVES **5**

Chapter 3. CONSTRAINTS **6**

Chapter 4. SYSTEM DESIGN **7**

System Modeling in Cameo 7

Hardware Design 10

Software Design 12

Chapter 5. DESIGN VALIDATION **15**

Test Plan 15

Risk Mitigation Plan 19

Chapter 6. ETHICAL/CONTEMPORARY ISSUES **20**

Chapter 7. STANDARDS **21**

ISO/IEC/IEEE 15288 Standard 21

ISA 101 HMI Standard 22

TI Code Composer Studio IDE & Libraries 22

Chapter 8. CONCLUSIONS **22**

Chapter 9. FUTURE WORK **23**

GUI Design 24

Identifying Origin of Noise 25

Modifying Microcontroller Code to Allow Variable Inputs 25

Fixing Motor Speed Output Plot 25

Running Documentation Tool on Code 26

REFERENCES **27**

BIBLIOGRAPHY **28**

APPENDICES **29**

Appendix A. Abbreviations 29

Appendix B. Budget 30

Appendix C. Timeline Progression 31

Appendix D. Spring Semester Plan 35

Appendix E. MBSE Artifacts 36

ACKNOWLEDGMENTS **42**

Chapter 1. INTRODUCTION

The quest for more electric aircraft is an industry-wide initiative to optimize aircraft performance, reduce operation and maintenance costs, and reduce the environmentally damaging effects of operating aircraft. Specifically, this concept prompts the redesign of all non-propulsive systems to use electric power rather than hydraulic or pneumatic techniques. Targeted pursuit of MEA technologies is a critical step towards improvements in aircraft fuel consumption, reliability, and carbon neutrality.

Current TRAS systems employ mechanically focused solutions using hydraulic or pneumatic actuation to control an aircraft. Over the last two years, Woodward, Inc. and the CSU Systems Engineering Department have partnered to identify and validate an electric TRAS system at the ASET Lab. EM-TRAS systems are also currently in development at major aerospace contenders such as Honeywell Aerospace and Collins Aerospace; Woodward, Inc. has been at the forefront of mechanical TRAS systems for decades and is actively participating in the race to establish the most effective solution to implement this revolutionary concept.

The inherently large scope of any aerospace project in terms of cost and physical real-estate demands a solution that can validate control and analytics algorithms for electrical drive alternatives without having access to full-sized aircraft hardware. Through a partnership between Woodward, Inc. and the Systems Engineering Department at Colorado State University, a representative hardware testbed was designed and is now being constructed by members of the ASET Lab. The power-converter (voltage/current source) for this testbed employs a TI TMS320F28379D microcontroller. To complement the hardware testbed, there is a need to develop a platform to enable rapid firmware development for the power-converter by evaluating the hardware specific (I/O) functions of the microcontroller without endangering the physical testbed itself. By using controller-hardware-in-the-loop (CHIL) to emulate the physical system that would provide inputs to the microcontroller, team members can set the stage for the ASET team to accomplish the following goals without risk to the physical testbed:

- Enact test failures and develop algorithms to mitigate and prevent these failures from occurring in real time
- Rapidly unit-test newly developed firmware
- Enable automated test sweeps to ensure that changes to the microcontroller code do not negatively impact code that was already validated

A generic CHIL platform using the hardware described in this report follows the schematic shown in Figure 1. CHIL platforms generally consist of a real time machine that can load a system model and perform operations in simulated real time. This is connected to a microcontroller with code that reflects a control model for the closed loop. Both of these elements are then connected to a data acquisition box (DAQ) that captures both input and output signals from the hardware components and allows for further analysis and manipulation of this data using a GUI or additional data processing software.

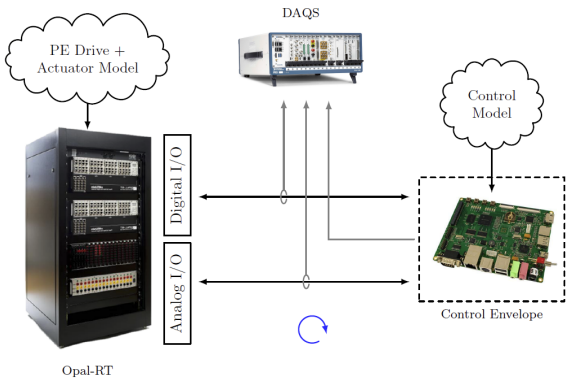


Figure 1 - Controller Hardware in the Loop

In addition to this general need to investigate complex systems like the EM-TRAS systems, CHIL platforms in general serve as a useful research mechanism for students to engage with and validate their own product concepts. To this end, the CHIL platform being developed by this project team will be used to further CSU's goals to increase accessibility of testing platforms for the rapid prototyping, design, and development of student projects.

This team is designing the microcontroller code to interface with an OPAL-RT real time processor that is running a simplified model of a DC machine through a CHIL platform. Code has been written in C and includes various functions that allow for digital to analog (and vice versa) conversion of signals between the real time machine and the microcontroller that are adjustable to specific voltage limitations. Eventually, the code will include unit tests that trigger specific failure modes in the supplied system model. These failure modes will be documented, and algorithms will be written to mitigate and prevent these failure modes from resulting in fatal errors to the overall system. By designing and rigorously testing this code on an internal model rather than on the physical testbed located in the ASET Lab, the team can ensure the longevity of the mechanical parts used to construct the testbed and can design solutions for the overall EM-TRAS system (or any complex system) to operate nominally despite potential failure modes that may occur. These failure modes are addressed in a proprietary document and will be tracked accordingly during code development to ensure that all customer requirements are being met.

As this project is not a continuation of previous work, this report will serve as an initial source of information regarding the steps that must be taken to identify the required interfaces between the OPAL-RT machine and the TI TMS320F28379D microcontroller, lessons learned in the implementation of Simulink models loaded onto the OPAL-RT, coding and verification of I/O connections between the two pieces of hardware, ways to adjust inputs and outputs between the two pieces of hardware, restrictions on these inputs and outputs, and the behavior of the data being transmitted between them. Neither piece of hardware is extensively well-described in the public domain since they have highly specific uses and are not commonly interfaced together; this report aims to describe their connection and the lessons learned in implementing this interface.

Furthermore, the concept of Model-Based Systems Engineering (MBSE) was employed in the definition of the CHIL platform being created by this team using Cameo Systems Modeler, a tool widely used in industry to revolutionize the way products are defined, validated, and maintained.

The principles of MBSE allow for the documentation and validation of a system design that can be evolved and maintained over a long period of time using modeling languages like SysML and tools like Cameo to digitally track its development. Since the CHIL platform is relatively simple in comparison to other systems described with MBSE principles, the key diagrams produced are less numerous than what one may find in the system model for a larger system. However, the CHIL platform model can and is being expanded to support other aspects of Woodward's efforts to design an EM-TRAS system and are supporting this team's goal to deliver not just a working CHIL platform, but a well-documented one at the conclusion of the senior design term.

Chapter 2 lists the specific objectives of this senior design project in a list format for ease of reference. Chapter 3 discusses the constraints imposed on the team in regards to the technology used to accomplish the objectives in Chapter 2. Chapter 4 delves into the three aspects of system design for the CHIL platform: System Modeling in Cameo using the principles of MBSE and related diagrams/requirements; the evolution of the Hardware Design and relevant hardware-based interfaces including specific connectors used, pin mappings, and component boards; and finally, the evolution of the Software Design being implemented on the TI microcontroller and the OPAL-RT machine including a description of the resulting output plots generated by the relevant hardware components. Chapter 5 discusses the design validation plan, specifically risks that were identified at project inception and what has been done since to mitigate those risks, as well as the general test plan that has already been executed to verify the interactions between hardware and software and those that still need to be performed in the coming semester. Chapter 6 discusses ethical and contemporary issues that revolve around the design of a CHIL platform such as this and the related EM-TRAS system that inspired the development of this project. Chapter 7 describes the standards used to design C code loaded onto the microcontroller and the system model in Cameo. Chapter 8 discusses principal findings of the work performed this semester and discusses the legitimacy of project continuation, as well as identification of guidelines for further steps to be taken. Chapter 9 discusses the next steps this team will take to complete execution of the project goals outlined in Chapter 2.

Chapter 2. OBJECTIVES

The objectives of this project are split up into primary and secondary categories. After discussion with the project advisor, these are deliverables that can and will support the overall goal to develop a platform for use by CSU students and faculty to analyze system models of various complexity in a closed loop.

Additional project deliverables were completed by members of the team in regards to defining system requirements for the physical components of a proprietary Woodward product; these deliverables will not be discussed in this report due to their confidential nature.

Primary Objectives:

- Determine the analog and digital I/O requirements for the microcontroller and OPAL-RT
- Develop C code for running a closed-loop CHIL experiment, where the OPAL-RT executes a real-time DC machine system model
- Run the closed-loop CHIL experiment with set failure modes as described by the system creator and collect measurement information

- Document the design steps and experimental results in a test report
- Include system requirements flow-through and design information on the CHIL platform in the existing Cameo repository for the testbed

Secondary Objectives:

- Design a Graphical User Interface (GUI) in LabView to represent the current state of the testbed during all tests as well as error codes that are flagged during these tests and the system's real-time response to mitigate these errors
- Create a Python script to parse through the microcontroller code and establish code coverage of all conditional branches to ensure that all code is being thoroughly tested and verified
- Run a documentation tool (e.g. Doxygen) on the finished code to extract an HTML report of all code documentation and increase readability/ease of understanding for future senior design members and CHIL platform users

All primary objectives have been initiated and are either fully or most of the way to completion. The project scope has changed several times through the semester; this will be elaborated on in Chapter 4. Secondary objectives will be completed by the close of the spring semester.

Chapter 3. CONSTRAINTS

The major constraints imposed on this project were the hardware timing and voltage limitations. These were not always directly available from datasheets and needed to be extracted through physical testing using oscilloscopes and function generators. The two major hardware components used to design the CHIL platform (the OPAL-RT machine and the TI microcontroller) were identified to have the following technical constraints:

Table 1: Hardware Constraints

OPAL 5600 RT Machine	**TI TMS320F28379D Microcontroller**
● I/O: -30V to 30V ● OPAL 5620 8 I/O Flat Carrier: ○ output: +/- 12VDC @ 1.2A (mezzanine A/B) ○ output: +/- 18VDC @ 0.8A (mezzanine A/B) ○ input: +/- 12VDC @ 6A; +/- 5VDC @ 3A ● OPAL 5142 Digital I/O: 0-3 V ● Onboard XILINX Spartan 3A FPGA: 3.3V (LVTTL) ● 12 nodes maximum ● Scope output can only be ready by a miniBNC connector	● 3.3V I/O design ● -0.3V min to 4.6V max (0-3.3 V nominal operating range) ● -20mA to 20mA ● 50 kHz synchronization rate ● Requires code written in C (no object-oriented functionality) ● Requires code to be written in the TI Code Composer Studio IDE

Since the TI microcontroller has a 50 kHz synchronization rate, the minimum sampling period that the OPAL-RT can operate with is 20 μs. The loaded Simulink model is currently running at a 50 μs timestep to accommodate additional delays in transmission. This follows the general standard that approximately 10 samples should be achieved per switching cycle.

The team is also limited by the specific hardware chosen to achieve the CHIL platform. The ASET lab had already purchased the OPAL RT machine and the specific TI controller, as well as an NI PXI machine with which to acquire and process data. Therefore, all code loaded onto the microcontroller had to be written in C using the TI Code Composer Studio IDE and provided libraries. The GUI to control input/output manipulation and data acquisition from the closed loop must be designed in LabView to interface with the PXI machine. Furthermore, any analysis of output data from the OPAL machine using an external oscilloscope needed to be performed with a specific miniBNC connector, which was not available in the lab and needed to be ordered separately.

Financial constraints were also imposed on the team; each member received a baseline of $200 from the department for the entire year. So far most of the hardware was already purchased by Dr. Cale, so the team is still within budget for the rest of the project timeline. More information on the budget can be found in Appendix B.

Chapter 4. SYSTEM DESIGN

System Modeling in Cameo

The first step to defining any system is the creation of verifiable and actionable system requirements that specify what the system shall do, how well, and under what conditions. As such, several major functional and non-functional requirements were defined for the CHIL platform. These requirements spanned the hardware, software, GUI, and general functionality of the CHIL platform. The full list of these requirements can be found in Appendix E.

After defining the system requirements, the next step was to model the Operational Viewpoint of the system. This was done by identifying the major domains of the system: the software, hardware, and user interface. Furthermore, there are six main types of users who will be interacting with the system at any given time: at least one on-site user, remote user, on-site coder, remote coder, on-site technician, and the powerhouse central power (not a human, but an important factor to consider since it dictates specific use cases of the system). These users are significant in their interaction with the various domains of the system as well as their involvement in use case diagrams, an example of which is shown in Figure 2.

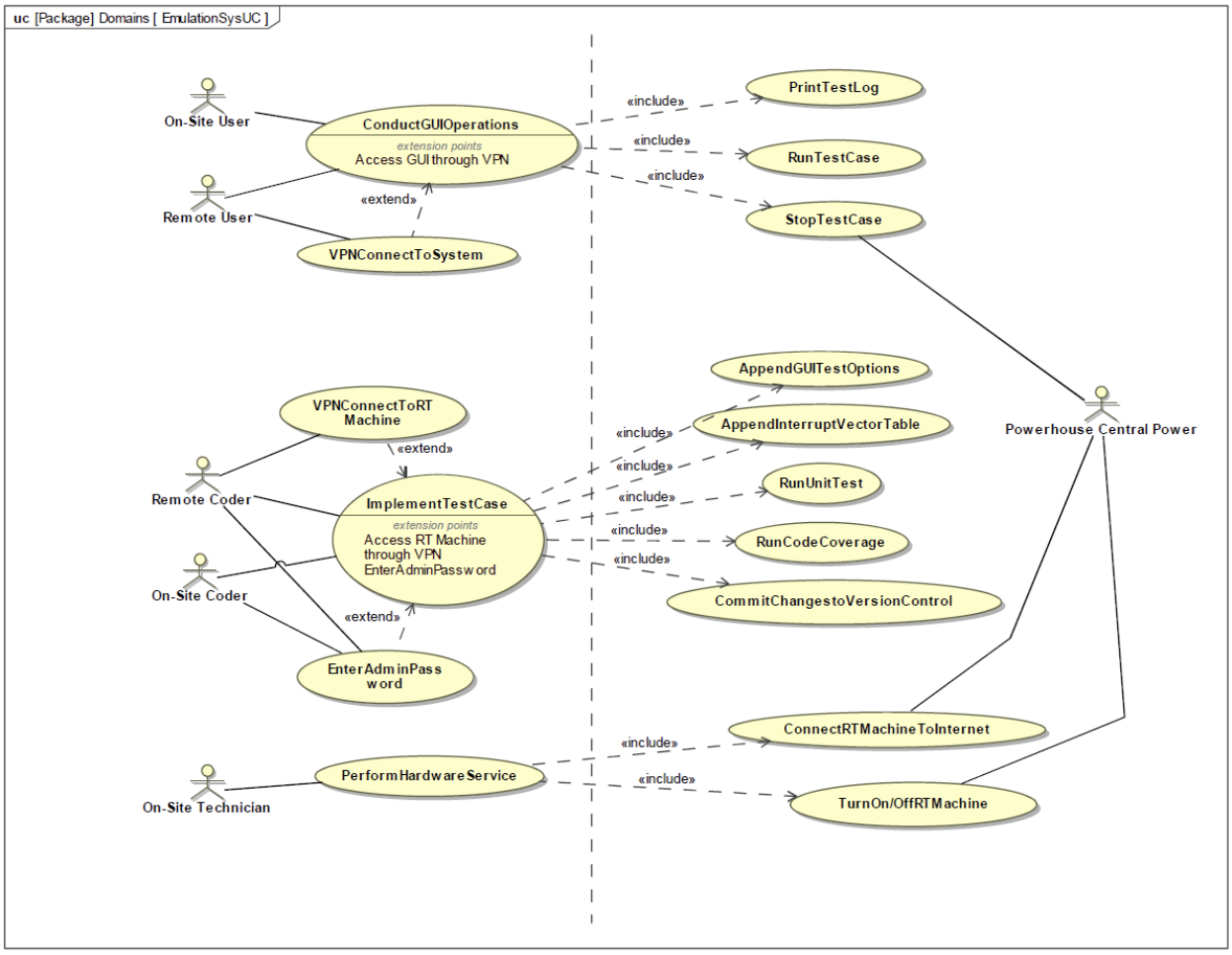


Figure 2 - Use Case Diagram for the CHIL System

The use case diagram in Figure 2 shows how the various actors in the system, listed earlier, are interacting with different use cases to demonstrate various high-level functionalities of the system. For example, a remote coder ust access the GUI through a VPN, and in so doing can then either print a test log, run a test case, or stop a test case. The powerhouse central power has the unique ability to stop test cases and turn on/off the OPAL-RT machine since the lab computer and OPAL machines are reliant on central power to operate. If the powerhouse loses power, then any running test cases are stopped because the OPAL is automatically shut off.

This kind of diagramming is significant, especially early in the project life cycle, because it provides the team with a thorough understanding of all the various tasks that the system must be able to achieve. By iterating through the use cases of the CHIL platform, the specific tasks to be completed by the software and hardware designers on the team were better understood and documented in further detail.

Another important aspect of MBSE is that the system model must constantly evolve as additional information about the system is acquired and understood. In the beginning of the project, the constraints on the TI microcontroller and OPAL-RT system, especially the constraints on the interface between the two elements, were not well understood. After spending time manually

testing input and output min/max voltage ranges of both hardware components, as well as determining the floor and ceiling of sampling rates between the two elements, additional diagrams that delve into the more specific functional and physical viewpoints of the system were composed.

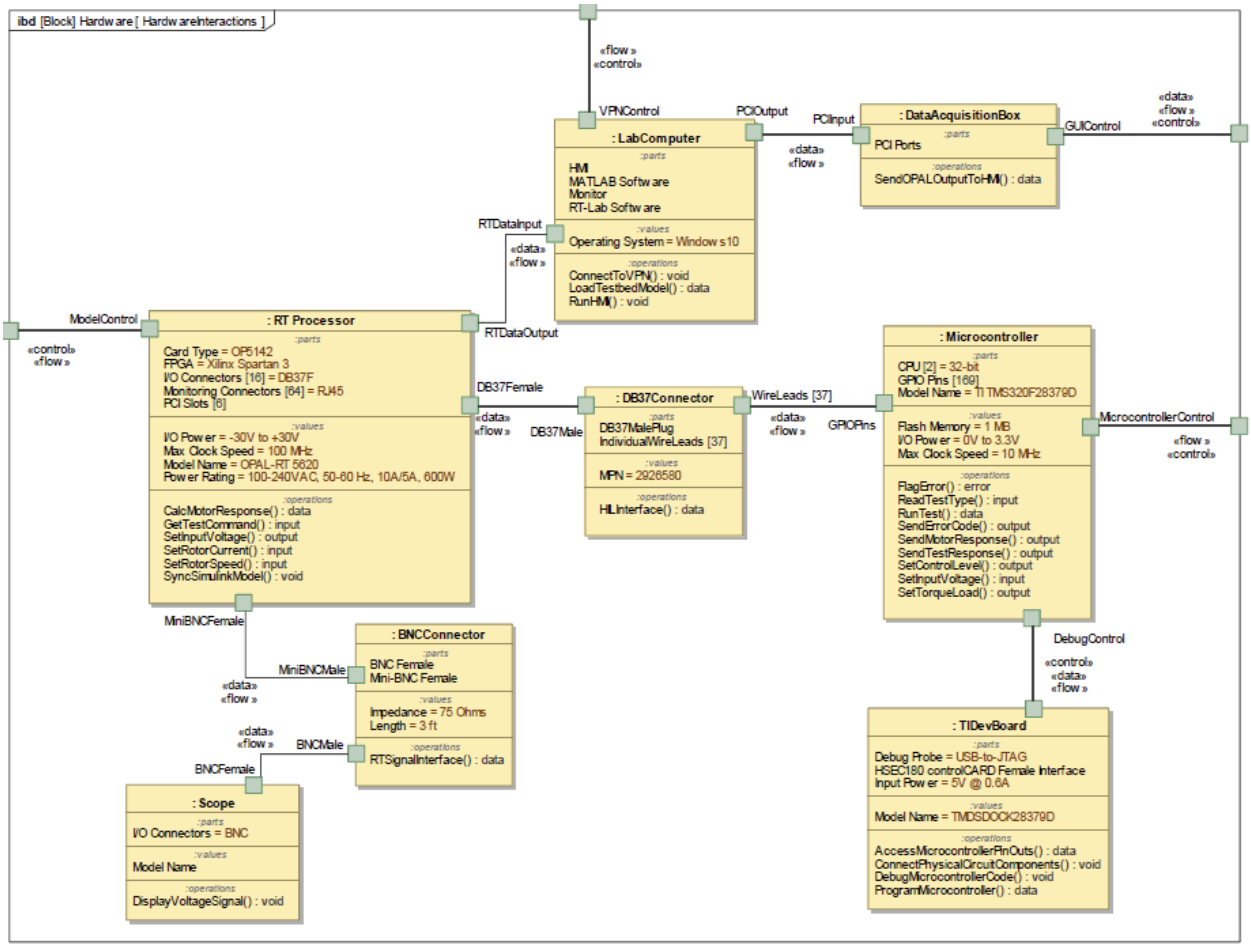


Figure 3 - Physical Hardware Interactions Diagram

Figure 3 shows an internal block diagram that shows the physical hardware interfaces of the system. This diagram combines both the functional and physical viewpoints of the system in that it shows the specific details about every piece of hardware being used in the system as well as the functional data flows between elements. Each hardware element is described as a block of the hardware domain and contains specific values that help to define its functionality (such as power requirements, model name, power rating, etc.); parts that compose the whole piece of hardware (such as I/O connectors, PCI ports, and other physical subcomponents); and operations that it must be able to complete to achieve the total functionality of the item. Data flows between the individual components are shown. The green boxes on the outline of each block are ports that demonstrate both physical and functional input and output capabilities of each hardware component.

Figure 4 shows a sequence diagram that documents the process flow and timing constraints if a user wants to enact a test to be run on the CHIL platform. First the test is initiated via the user

interface that will be loaded onto the DAQ box. Then the test is processed through the model loaded onto the RT processor, with inputs set and sent to the microcontroller for processing. Inside the microcontroller the commands required to execute the specific test are located in the interrupt vector table, which is called because the polling has been interrupted by a call to a function, and then an error log is generated and returned to the user via the user interface to show the result of the test. Analysis of this diagram will show the general timing constraints imposed on each step of this sequence.

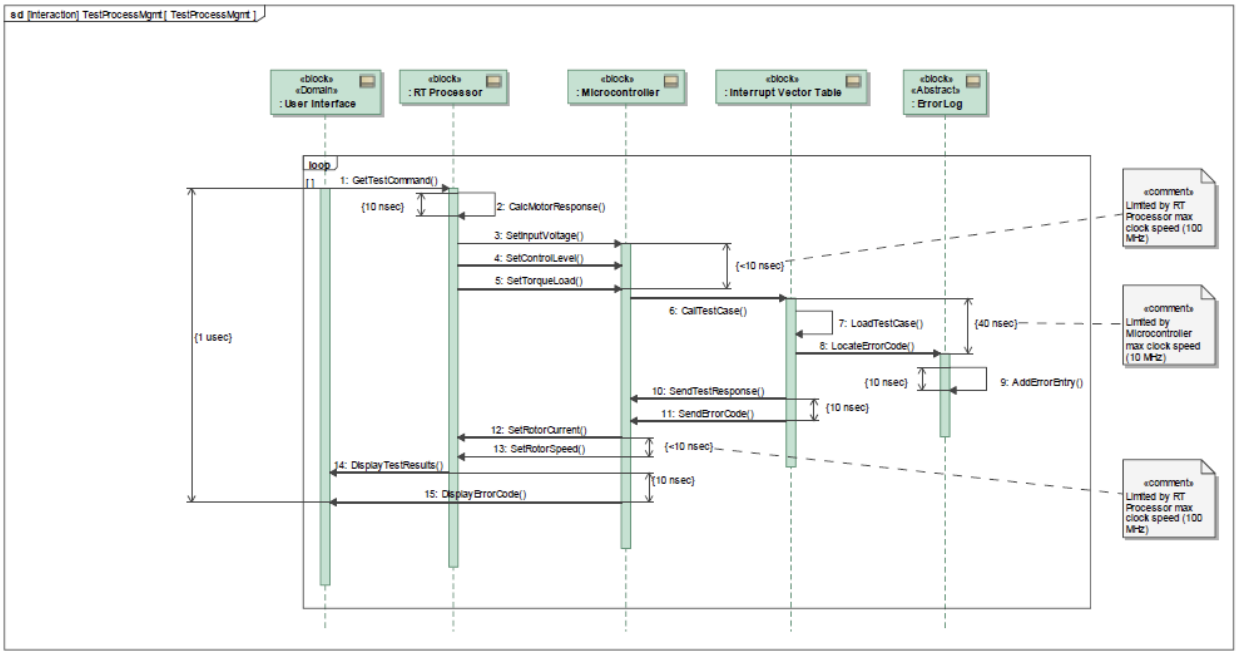


Figure 4 - Test Process Sequence Diagram

Many more diagrams and aspects of the system were created in accordance with the tenets of MBSE, and these can be found in Appendix E. Overall, the major function of including the MBSE aspect to this project was to have a central location to store all specific functional and physical information dictating the design of the system so that team members could always reference the major system requirements, save time on troubleshooting, and adhere to predefined standards to ensure a high quality product.

Hardware Design

When the project was initially proposed, it was expected that a signal conditioning board may be needed to scale down the voltages from the OPAL-RT to fit within the 0-3.3V that the TI microcontroller can accept. However, after learning how to use the OPAL-RT system, it was determined that the voltages could be scaled down in the simulink model, and no hardware was needed. This changed the scope of the project significantly and allowed for more testing to be performed on the hardware, as well as design of neater interfaces between the OPAL-RT system and the microcontroller.

The figures below show the TI microcontroller after the connections were cleaned up. To minimize the setup time and reduce the occurrence of incorrectly connected wires, we soldered terminals on the TI developer (dev) board and left the connections from these to the microcontroller pins in place (red and yellow wires, signal and ground respectively). As you can see in the left figure, the terminals are labeled I1, I2, O1 and O2. These labels correspond to the signals seen from the microcontroller's perspective. I1 and I2 are the terminals receiving the OPAL's output signals, Armature Current and Motor Speed, while O1 and O2 are the terminals associated with the Duty Cycle and Load Torque. This setup was first tested with a multimeter's conductivity function, ensuring that there was proper connection between each dev board pin all the way to the terminal connection.

Below are the figures shown for how the team interfaced the OPAL with the TI-Microcontroller. After further evaluation and experimentation. As seen in Figure 5, the 8 pins were established (4 of which are GND pins). Referencing the terminals previously stated, we configured the HSEC pins which were discovered through the TI Pinout configuration sheet. More details on exploration of these discoveries can be found in the following Software section. In Figure 6, the cables on the left are the two DB-37 cables connecting directly to the OPAL, one for input and one for output. Here, upon careful inspection of pinout configurations of the DB-37 cables, we place four separate coaxial cables, each with a signal and ground cable. These then can be easily interfaced with the terminal port connections connecting directly into the TI Board.

The OPAL can be interfaced remotely through the RT-LAB V2021.2 software. The steps are fairly straight forward as this is a similar process to remotely log into a machine or server. Upon opening the RT-LAB software you can connect with a new target by right clicking on Targets and then inputting the correct IP address of the OPAL. From here you can execute Simulink and MatLab scripts.

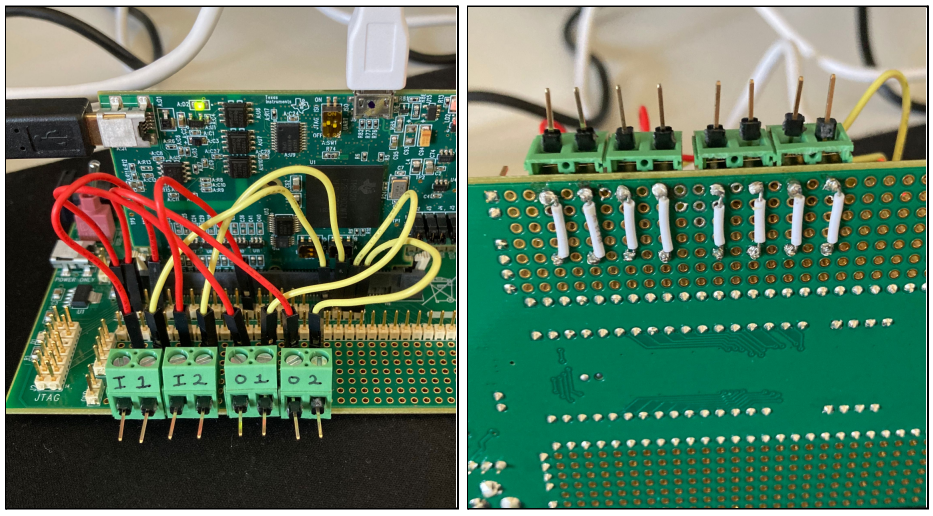


Figure 5 - TI Microcontroller & I/O Connection Points

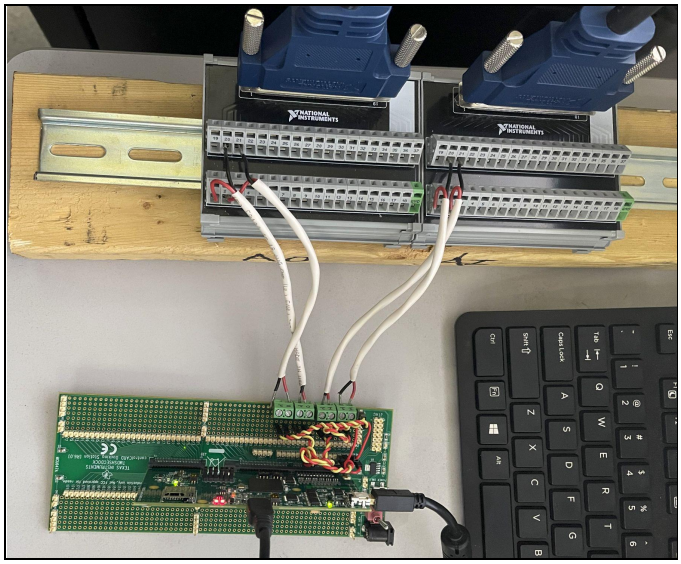


Figure 6 - TI Microcontroller Connected to OPAL-RT via DB37 Pigtail Connector

Software Design

A key portion of the success for the software side was found within the TI-Microcontroller libraries and the TI One Day learning course, which is linked in the reference section below. Important keys to success for learning how to use the C code that is constructed on the TI Board was heavily dependent on the one day learning course. Here the producers of the board show how to directly interface with the TI Board. Lab 1 showed how to run and compile code, Lab 2 walked us through using one input and one output, and finally Lab 3 showed us how to use two analog inputs and two analog outputs. This task seemed like a basic, straightforward one. However the one flaw in this process was that the learning course was using a slightly different microcontroller. So this dramatically changed the I/O within the program. We had to carefully reference the pin out documentation and examine the provided code that provided register memory control.

With a successfully coded program we are able to build and enter debug mode within Code Composer Studio. Here, we execute the program and set up the Graph I/O dual mode option with the correct corresponding registers. That is how Figure 10 was created.

Figure 7 is a code snippet of the successful pinouts that we used for our interfacing with the microcontroller.

```
134 void ConfigureDAC(void)
135 {
136     EALLOW;
137     DacaRegs.DACCTL.bit.DACREFSEL = 1;          // Use ADC references (HSEC Pin 09)
138     DacaRegs.DACCTL.bit.LOADMODE = 0;           // Load on next SYSCLK
139     DacaRegs.DACOUTEN.bit.DACOUTEN = 1;         // Enable DAC
140     DacbRegs.DACCTL.bit.DACREFSEL = 1;          // Use ADC references (HSEC Pin 11)
141     DacbRegs.DACCTL.bit.LOADMODE = 0;           // Load on next SYSCLK
142     DacbRegs.DACOUTEN.bit.DACOUTEN = 1;         // Enable DAC
143     EDIS;
144 }
169
170 void SetupADCEpwm(void)
171 {
172     // Select the channels to convert and end of conversion flag
173     EALLOW;
174     AdcaRegs.ADCSOC0CTL.bit.CHSEL = 2;          // SOC0 will convert pin A0 (HSEC Pin 15)
175     AdcaRegs.ADCSOC0CTL.bit.ACQPS = 14;         // Sample window is 100 SYSCLK cycles
176     AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 7;        // Trigger on ePWM2 SOCA/C
177     AdcaRegs.ADCINTSEL1N2.bit.INT1SEL = 0;      // End of SOC0 will set INT1 flag
178     AdcaRegs.ADCINTSEL1N2.bit.INT1E = 1;        // Enable INT1 flag
179     AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;      // Make sure INT1 flag is cleared
180
181     // Also setup ADC-C3 in this example
182     AdccRegs.ADCSOC0CTL.bit.CHSEL = 3;          // SOC0 will convert pin C3 (HSEC Pin 33)
183     AdccRegs.ADCSOC0CTL.bit.ACQPS = 14;         // Sample window is 100 SYSCLK cycles
184     AdccRegs.ADCSOC0CTL.bit.TRIGSEL = 7;        // Trigger on ePWM2 SOCA/C
185     EDIS;
```

Figure 7 - Configuring the input and output registers within the C code of the Microcontroller

Other key implementations with the code is converting the voltage range of the microcontroller. Unfortunately the OPAL and the TI Board are not a perfect one to one ratio with corresponding voltages that they require to use. Below are the HEX conversions and voltage ranges corresponding to these HEX values.

```
12  #include "F28x_Project.h"           // Device Header File and Examples Include File
13
14  // Variables for output
15  Uint16 resultsIndex;
16  Uint16 ToggleCount = 0;
17  Uint16 mmSpeed = 0x000;         // {-600, 600} [rad/s] - Motor Mechanical Speed rad/s | {0.0 V, 3.3 V}
18  Uint16 maCurrent = 0x000;       // {0, 100} [A] - Motor Armature Current in A | {0.0 V, 3.3 V}
19
20  //Variables for input
21  Uint16 dacOutput;
22  Uint16 LoadTorque = 0.1;      // {-0.2, 0.2} [Nm] - Load Torque in Nm | {0.0 V, 3.0 V}
23  Uint16 DutyCycle = 95;       // {0, 100} [%]  - Load Torque in % | {0.0 V, 3.0 V}
24
25
26  // Definitions for PWM generation
27  #define PWM1_PERIOD 0xC350          // PWM1 frequency = 2kHz
28  #define PWM1_CMPR25 PWM1_PERIOD>>2  // PWM1 initial duty cycle = 25%
29
30  // Function Prototypes
31  void ConfigureADC(void);
32  void ConfigureDAC(void);
33  void ConfigureEPWM(void);
34  void SetupADCEpwm(void);
35  void InitEPwm1(void);               // Configure ePWM module 1
36  void InitEPwm2(void);               // Configure ePWM module 2
37  void InitEPwm5(void);               // Configure ePWM module 5
38  interrupt void adca1_isr(void);     // ADC interrupt service routine
39
```

Figure 8 - Basis of voltage and range conversion for inputs and outputs

The microcontroller produced outputs that fed into the emulated hardware system loaded on to the OPAL-RT machine. Figure 9 shows the emulated H-bridge circuit schematic used to create a Simulink model that was provided inputs by the code shown above. The H-Bridge model switches polarity of the voltage applied to the load so that the DC motor can run both forwards and backwards.This is a very simplified design meant to showcase the functionality of the CHIL platform. Any circuit schematic or model can be loaded onto the OPAL-RT so long as its complexity does not exceed the provided 12 nodes on the machine.

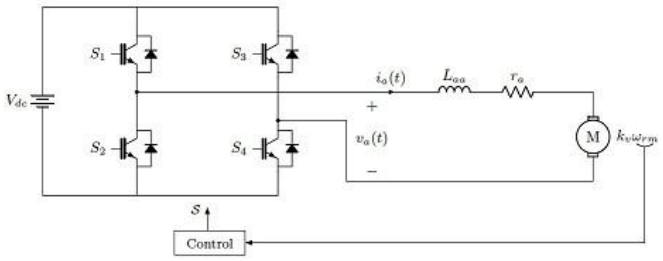


Figure 9 - Circuit Design of the Emulated H Bridge DC Machine Model

Figure 10 shows the autogenerated user interface created by the OPAL-RT once the basic Simulink model is loaded onto it. This user interface allows for internal viewing of the behavior of the loaded model in regards to motor current (iL_scope) and motor speed (wm_scope). The switches on the left side of the model allow the user to decide if they want to view the results on the OPAL-RT (purely simulated) or if they want to view the analog results through the TI microcontroller (closed loop). The two inputs to the system have sliding scales similarly to what will be programmed in LabView.

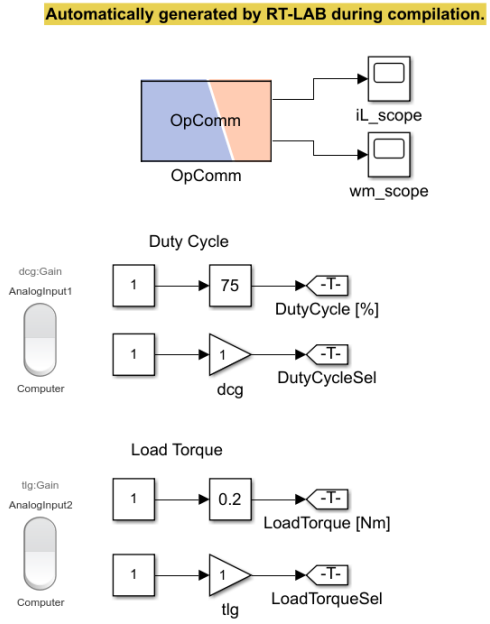


Figure 10 - OPAL-RT Autogenerated User Interface

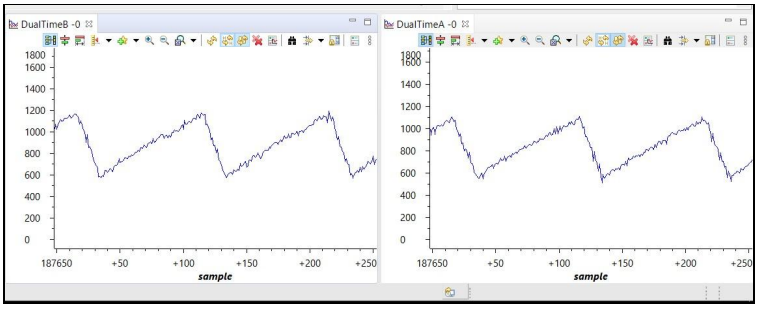


Figure 11 - OPAL-RT Outputs w/ Noise (Motor Current, left & Motor Speed, Right)

Upon our final tests this semester, we were receiving a noticeable amount of noise in our incoming signal into the TI Board. We are under the impression that we do not want this noise and that this noise is not acceptable, at the moment. As we are still troubleshooting this error, we have attempted several solutions. These steps included shortening our connecting wire, taking out jumper cables, and firmly securing any connections to a flat, stable (non vibrating) surface. Other steps that we will take in the future is exploring our grounds for our wires and taking out any bias that may be present or calculating the Signal to Noise Ratio. As a last case resort, we can add an RC circuit for filtering noise.

Another flaw is that while we are correctly receiving our current, the OPAL-RT is only sending out one signal to the TI board. That is why the two graphs are duplicated signals. Further in the future we will expand on these outputs to also receive the duty cycle and current.

Chapter 5. DESIGN VALIDATION

Test Plan

This system required testing of both hardware (TI microcontroller and OPAL-RT machine) and software (microcontroller code) components. To accomplish this, a set of hard-coded inputs were passed to the TI microcontroller to simulate test conditions iterated in the following pages. These inputs triggered specific hardware outputs in the form of a voltage range that were representative of a certain quantity (load torque, duty cycle ratio, etc.) passed via connector to the OPAL-RT machine, causing an appropriate response within a constant Simulink model. This model generated the appropriate response to represent the behavior of the emulated testbed in the form of a motor speed and armature current to be passed back via connector to the TI microcontroller and read in as input.

Specific targets tested within the system included:

- I/O voltage ranges
- Effects of changing control level (duty cycle)
- Effect of changing load torque
- Effect on simulated motor speed
- Effect on simulated armature current
- Effect on simulated motor temperature

When developing the C code required to interface the TI microcontroller with the OPAL-RT, the

team initially used a Digilent Analog Discovery 2 USB oscilloscope, logic analyzer, and wave generator in place of the OPAL-RT. By using this device, the team was able to develop and debug code quickly and efficiently, without needing to be on site at the CSU Powerhouse where the OPAL-RT is located. This method also allowed the team to work with a much more familiar device, potentially avoiding any hazards associated with damaging the OPAL-RT.

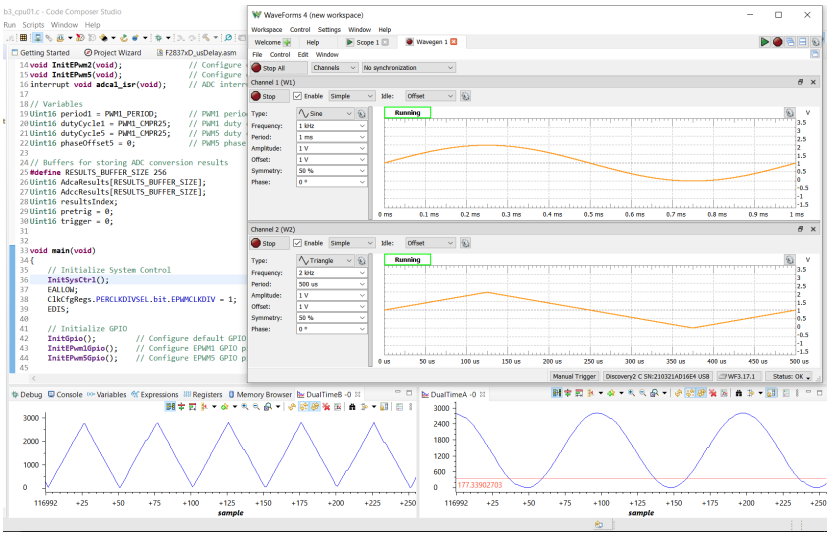


Figure 11 - Analog Discovery Test Outputs

Using the Analog Discovery's wave generator functionality, the team sent two sinusoidal signals serving as the signals generated on the OPAL-RT, and connected them to ADC pins configured on the microcontroller. The oscilloscope functionality on the Analog Discovery was used to measure outputs from the microcontroller's DAC configured pins. With this configuration, the team was able to send two DC signals from the microcontroller to the Analog Discovery serving as the Duty Cycle and Load Torque and two AC signals from the Analog Discovery to the microcontroller serving as the Armature Current and Motor Speed.

When testing our code for correctness in receiving data, we utilized the function generator on the Analog Discovery, and connected the pins to the ADC input pins on the microcontroller. We started a test by setting the Analog Discovery wave generator functions to two different signals such as a sinusoidal signal and a sawtooth signal. Because the acceptable input voltage of the microcontroller is 0 to 3.3V, we only needed to ensure the voltage being sent to the microcontroller was within these values. Once we verified the signals were appropriate to send to the microcontroller, we connected these two signals to the microcontroller ADC pins. We then used Code Composer Studio to graph the data stored in the ADC results registers to verify that we see the same signal graphed in Code Composer that we sent to the microcontroller.

Once we verified that our code could handle receiving data from the Analog Discovery, we moved to verifying the output functionality of our code. Using a simple conversion from the desired range of input values (i.e. 0% to 100% Duty Cycle or -0.2Nm to 0.2Nm Load Torque), and the maximum allowable output voltage (i.e. 0V to 3V) of the microcontroller, we were able to convert a value of input (Duty Cycle or Load Torque) to a bit representation between 0 and 4095 bits. Once we had this conversion, we could simply enter in a value for both the Duty Cycle

and Load Torque, convert this to a number of bits, and send this through the DAC which in turn converts this number to a voltage between 0 and 3V.

To verify the correctness of this conversion and subsequent DAC output, we simply entered values for both the Duty Cycle and Load Torque, noting the allowable range for these values, and calculated the expected voltage to be seen on the Analog Discovery oscilloscope inputs. For example, a Duty Cycle of 50% would yield a bit value of 2047 (half of 4095, rounded down) and therefore we should see a value of half the maximum microcontroller output voltage, 1.5V. If we were to run the microcontroller code and see an input voltage on the Analog Discovery oscilloscope of 1.5V +/- 3%, we would consider this a confirmation that the code is correct and operating as intended. Direct test cases can be seen in tests 1.2 and 1.3.

Test 1.1	TI GPIO Connections
Components	TI Microcontroller, TI dev board, jumper cables, function generator
Constraints	● Microcontroller can receive 0-3.3V (-0.3V absolute min to 4.6 V absolute max @ -20 to 20 mA) ● Microcontroller can output 0-3V (+/- 0.3%)
Stimulus	Sinusoidal waveform (1V amplitude, 1V DC bias @ 1 kHz) generated by function generator
Expected Response	TI sees the input from the Analog Discovery and transmits a shifted output (by 1V) visible on an oscilloscope.
Test Plan	1. Generate the 1V @ 1kHz sinusoidal waveform using the function generator. 2. Connect the voltage output from the function generator to the TI Dev Board on pins 9 (ADC A0) and 11 (ADC A1). 3. Compile and load Actuation_CPU1.c file onto TI microcontroller to take in input from pins 9 and 11, shift by 1V constant value, and output using pins 15 (ADC A2) and 17 (ADC A3). 4. Connect the original input from pins 9 and 11, and output from pins 15 and 17 to an oscilloscope and confirm that the output is shifted by 1V.

Test 1.2	Open-Loop Control Level (within bounds)
Components	TI Microcontroller, OPAL-RT, TI dev board, jumper cables, DB37 breakout boards, BNC-miniBNC connector, oscilloscope

Constraints	● Microcontroller can receive 0-3.3V (-0.3 absolute min to 4.6 V absolute max @ -20 to 20 mA) ● Microcontroller can output 0-3V (+/- 0.3%) ● Output value limited to 12 bits ● Control level (duty cycle) range limited to 0-100% (b/w 0-3 V) ● Values above 3V will saturate at 100% duty cycle (cannot exceed 100%)
Stimulus	Hexadecimal value representing a control level percentage, controlled by the user via a slider scale in the system GUI or hard-coded in the microcontroller code.
Expected Response	Example: if the user enters a 50% control level, the voltage transmitted to the OPAL should be 1.5V (+/- 0.3%).
Test Plan	1. Assign a constant control level value of 50% (hard-coded or via GUI). 2. Convert 50% to a 12 bit hexadecimal value (0x7FF). 3. Transmit hexadecimal value via pins 9 and 11 (ADC A0 and A1). 4. Connect jumpers at pins 9 and 11 on the microcontroller to CH00, Group 1B on the OPAL-RT via the DB37 breakout board. 5. Connect Group 1B scope output to an oscilloscope using the BNC-miniBNC connector. 6. Confirm that the output corresponds to test input (following test case in expected response described above).

Test 1.3	Open-Loop Load Torque (within bounds)
Components	TI Microcontroller, OPAL-RT, TI dev board, jumper cables, DB37 breakout boards, BNC-miniBNC connector, oscilloscope
Constraints	● Microcontroller can receive 0-3.3V (-0.3V absolute min to 4.6 V absolute max @ -20 to 20 mA) ● Microcontroller can output 0-3V (+/- 0.3%) ● Output value limited to 12 bits ● Load torque range limited to -0.2 to 0.2 Nm (mapped to 0-3V) ● Values above 3V will NOT saturate (will exceed maximum load torque)
Stimulus	Hexadecimal value representing a load torque, controlled by the user via a slider scale in the system GUI or hard-coded in the microcontroller code.
Expected Response	Example: if the user enters 0 Nm (no load torque), the voltage transmitted to the OPAL should be 1.5V (+/- 0.3%).

Test Plan	1. Assign a constant load torque value x (hard-coded or via GUI). 2. Convert x to a 12 bit hexadecimal value. 3. Transmit hexadecimal value via pins 9 and 11 (ADC A0 and A1). 4. Connect jumpers at pins 9 and 11 on the microcontroller to CH01, Group 1B on the OPAL-RT via the DB37 breakout board. 5. Connect Group 1B scope output to an oscilloscope using the BNC-miniBNC connector. 6. Confirm that the output corresponds to test input (following test case in expected response described above).

Risk Mitigation Plan

The original risks enumerated in the project plan were written with the assumption that a signal conditioning board and wiring harness would need to be created to scale down the voltage from the OPAL-RT machine to the microcontroller. Once this was determined to be unnecessary, the risk factors weighing over this project became significantly less. Some risks that are still relevant to the current design understanding are:

- Potential miscalculation of voltage/current relationships between the OPAL-RT and the microcontroller
- Unintentional modification of the Simulink model loaded onto the OPAL-RT
- Physical damage to the OPAL-RT
- Staff illness due to COVID-19 and the new Omicron variant
- Unintentional disclosure of confidential components of the project

These risks have been assessed and mitigated according to the following risk mitigation table.

Table 2: Risk Analysis Table

#	Risk Event	Probability	Impact (hrs)	Score (hrs)	Effect	Risk Mitigation Plan	Responsible Person(s)
1	Miscalculation of voltage/current relationships between OPAL and microcontroller	0.3	100	30	TI microcontroller or OPAL-RT are damaged as a result	Validate inputs and outputs using the Analog Discovery before manually closing the loop	Jake/Dylan
2	Unintentional modification of Simulink model loaded onto OPAL	0.1	5	0.5	The model will have to be reloaded onto the OPAL.	The OPAL will be kept on and the model will stay constant so that it doesn't need to be reloaded or adjusted.	Jake/Dylan
3	Physical damage to OPAL	0.1	500	50	The CSU SE/ECE department will have to repair or order a new OPAL. This is *extremely* costly	Team members will not touch the physical OPAL machinery other than to connect inputs/outputs. Members will always ground themselves	All

					and will delay the entire project for an indefinite period of time.	before touching the machine.	
4	Staff illness	0.1	18	1.8	Team is down in manpower and tasks take longer to complete	Non-sick team members make availability in their weekly schedule to cover time-sensitive tasks assigned to sick team member. Ensure project schedule and Trello are kept up to date with tasks/deadlines and time requirements	All
5	Unintentional disclosure of confidential components of the project	0.3	10	3	Proprietary information is given to the public and patent timeframes are tightened, severely affecting customer's timeline	All team members signed an NDA. Reiterating which portions of the project are proprietary at weekly meetings. All deliverables proofed by advisors.	Kori
6	Noise in the OPAL-RT output	1.0	10	10	Signal is not as pure as possible, leading to potential errors in data processing.	Adjust signal acquisition rate, twist wires, and process data using FFT to identify the frequency of the noise.	Jake, Dylan
			Total Risk:	**95.3**			

Chapter 6. ETHICAL/CONTEMPORARY ISSUES

The CHIL platform as it is being designed is intended for use by CSU students and faculty who would like to emulate a system on a CHIL platform as a means to validate their design before purchasing expensive parts and equipment. This directly supports the IEEE Code of Ethics in its goal to "improve the understanding of individuals and society of the capabilities and societal implications of conventional and emerging technologies," specifically as this platform is being used to verify an EM-TRAS system that may change the way aircraft operate globally [1].

By creating a system that allows researchers and engineers to validate their system completely virtually before ever requisitioning parts and materials, this project is eliminating waste generated by improper planning and/or insufficient understanding of the system requirements. This is an important factor when considering any engineering product because anything electronic generates a considerable amount of waste both in production, during use, and after consumption.

One potential source of ethical concern is the fact that for this project to successfully allow both in person and remote operation of the CHIL platform, the OPAL-RT machine has to remain powered at all times. The powerhouse is a "100,000 square-foot green building that is a model for sustainable building practices," but a building of this size still draws a significant amount of power, and the OPAL is a large machine. There is a tradeoff between creating this accessibility for research while also still consuming large amounts of power, and as engineers, it's important to assess whether this tradeoff is justified [2].

No major conflicts arose during the course of this project. Conflict resolution skills employed in this project were mostly needed during periods of miscommunication. Due to the variety of tasks being undertaken between the CHIL element of this project and the other Woodward projects related to EM-TRAS, the scope of the project was often changing or unclear. Graduate students involved with the project were not always available to support the team and needed more frequent prodding to provide the requisite support. There was also a question of ethics from a teaching perspective. For instance, work had previously been performed by another student to identify the proper way to write code that would interface with the microcontroller. The team and advisor all faced an ethical dilemma of whether or not it would be acceptable to bypass the learning process associated with senior design work for the sake of time savings. In the end, the decision was made that it would be acceptable, although a delayed response from the graduate student who was meant to provide the code made it so that the team spent the extra time learning how to interface with the TI from scratch. The goals of the semester were still achieved, albeit at a slower pace than expected, but the learning that came from spending those additional hours understanding the code and reasoning behind it made for a better overall understanding of the system.

Chapter 7. STANDARDS

ISO/IEC/IEEE 15288 Standard

The ISO/IEC/IEEE 15288 standard describes a common framework for identifying the life cycle of a system, specifically the processes, keywords, and definitions of associated hardware, software, and data components. This standard explains how both functional and non-functional requirements should be formatted to most accurately describe the system, and how to ensure that they are verifiable [3]. It also describes how the system should be modeled using the principles of MBSE to ensure that the system has defined boundaries, hierarchical relationships, a complete list of users (both human and non-human), and general functionality that can expand to other similar systems (i.e. follows the MBSE principles of loose coupling and generalization).

This standard was a key factor in the model-based design of this system and provided for the team to make well-informed design decisions when creating the interfaces between the OPAL-RT and the microcontroller. Initially, the standard governed the creation of high-level functional requirements for the CHIL system that were then expanded into more specific requirements regarding timing, voltage limits, and other physical constraints. By following the rules outlined in this standard, changes made to the scope of the project did not drastically affect the overall system design and allowed for a smooth transition to the new goals of the system. The

MBSE artifacts seen in Appendix E demonstrate how the system diagrams produced adhere to the 15288 standard.

Furthermore, the 15288 standard helped to create component-level requirements for the Woodward customer as part of the proprietary portion of this project that will not be discussed in this report.

ISA 101 HMI Standard

The ISA 101 HMI standard describes a set of guidelines for human-machine interfaces (HMI) that contribute to a reduction in human error and unintended manipulation of the back-end. While an HMI has not yet been established for this system (this is part of the Spring semester objectives), the standard has been reviewed and incorporated into the system's high level requirements as a verification rubric for the finalized GUI design. The standard explains how to go about designing an HMI such that a user is cognitively limited to the full functional scope of the system, while simultaneously making them aware of the method which they may follow to "transform, reduce, store, recover, and use sensory input" [4].

What this means for the CHIL system is that the HMI designed in LabView must display specific elements (e.g. buttons, combo boxes, and other interactive items) that allow the user to enact test cases, stop test cases, view the behavior of the system, produce error logs, produce outputs logs, and furthermore, informs them of the steps they can ultimately take to process the data produced by the system into a format that aligns with their specific needs.

In addition, the ISA 101 HMI standard describes best practices in regards to density of information, minimalist and effective UI design, and clear indicators of alarm/emergency functionality of the system. There is a fully defined style guide that the team will adhere to when designing the HMI in LabView that provides specifications all the way down to scripting, embedded logic, and use of color to convey certain elements of consideration.

More specific requirements will be written in regards to this standard once the work on designing an HMI begins in the following semester.

TI Code Composer Studio IDE & Libraries

While not an official standard, all the microcontroller's code was written in the TI Code Composer Studio IDE using the embedded libraries. This was done to ensure that the I/O and ADC/DAC functions of the CHIL system were performed in compatibility and compliance with the microcontroller's requirements.

Chapter 8. CONCLUSIONS

The principal findings of this semester's work are as follows:

- The TI successfully outputs analog DC signals in a range of 0-3 V.
- The TI successfully accepts analog inputs in a range of 0-3.3 V.
- The OPAL-RT accepts analog signals compatible with the TI microcontroller.

- A closed-loop between the TI and the OPAL-RT has been successfully implemented and is ready for the inclusion of a GUI to moderate the inputs and view the outputs of the CHIL system.
- The output signals produced by the OPAL-RT are noisy. After testing and modifying the wiring setup, it appears that this noise is a result of some internal properties of the OPAL-RT, but this needs to be verified through further analysis.
- The MBSE diagrams created to model the CHIL system are useful points of reference for current and future teams to understand timing relationships between the OPAL-RT machine and the microcontroller, identify all required physical components in the system, and verify the behavior of the system under specified conditions.

Based on these findings, the work proposed for the next semester in Chapter 9 can continue, although some uncertainties and additional tasks remain to fully close out the objectives of this semester. These uncertainties are as follows:

- Currently, the microcontroller code sends values to the OPAL-RT machine via hard-coded values provided as 12-bit hexadecimal numbers. The code needs to be compiled and built every time this number is changed. Because of this, the team needs to replace these values with variables that can be modified via system interrupts.
- The output of the OPAL-RT is somewhat noisy. Tests have been performed to identify the source of this noise, but the result is still inconclusive. The team needs to continue investigating the source of the noise and, if possible, find a programmatic way to eliminate it.
- The OPAL-RT should be producing a plot of motor current and a separate plot of motor speed. However, it currently only produces duplicate plots of motor current. The team needs to identify the reason for this, which is likely due to a minor error in the model loaded onto the OPAL-RT, and fix it so that the proper plot is being generated.

Future steps should include updates to the MBSE model to ensure that all new changes in the system are being tracked. Timing requirements may change as the GUI is implemented, since the inclusion of another hardware element may increase the amount of time required to execute a test condition and receive the corresponding output.

Chapter 9. FUTURE WORK

Based on the discussion in previous chapters, the major objectives for the coming semester include:

- Creation/validation of a GUI in LabView to be loaded onto an NI DAQ box following the ISA 101 HMI standard that allows users to perform the following functions (at a minimum):
    - Manipulate inputs to the system via sliding scales.
    - View the resulting behavior of the system as raw data output.
    - Generate a test report describing the inputs and resulting outputs.
- Identify the cause for the noise in the output of the OPAL-RT and mitigate this by following the risk mitigation plan. Mitigation efforts not already attempted include:
    - Extracting and processing OPAL-RT output data using the Fast Fourier Transform functionality in MATLAB to identify the frequency of the noise.

- Modify the microcontroller code to support a dynamically changing input (not just a hard-coded value) using the hardware interrupt functionality.
- Identify the reason why the motor current and motor speed plots are visually the same when outputting from the OPAL-RT. The motor current plot is correct, but the motor speed plot should be a sinusoidal waveform.
- Run a documentation tool on all written code in MATLAB, Simulink, LabView, and the Code Composer Studio to generate an HTML package and user guide.

These objectives are explained in more detail in the following pages.

GUI Design

Due to shipping limitations, the PXI DAQ that was originally planned to be available in the beginning of the spring semester will not arrive until the middle of the spring semester. As such, the team will have to develop the GUI using a cRIO DAQ that already exists in the lab. This will only implicate the team in slightly extending the schedule to accommodate for the transition between machines. The PXI DAQ has a built-in display, whereas the cRIO does not, and requires data to stream via an IP address that can then be processed and displayed on an external monitor. Once the PXI is acquired, the LabView code written to interface with the cRIO can be modified relatively painlessly to perform on the PXI.

Presently, no team members have experience with LabView. In order to design the GUI, a practical understanding of the language must be acquired. Over the winter break and in the first couple of weeks of the semester, time has been allocated in the schedule to learn LabView via free online tutorials and labs. The GUI should be fully designed and validated in no more than two months because of the wealth of online and in-lab support with the LabView language. Especially in comparison to the lack of information about the OPAL-RT and TI microcontroller interfaces this semester, the GUI design process should be relatively straightforward.

The GUI will include sliding scales for the user to manipulate the two inputs to the system: control level (duty cycle from 0-100%) and the load torque (from -0.2 Nm to +0.2 Nm). These scales will be limited to these values so that they cannot exceed the boundaries of what the CHIL system has been programmed to accept. The GUI will display the raw output of the system and provide the user with the ability to export this raw data in a CSV or equivalent format that can be processed using a commercially available data processing tool (e.g. MATLAB).

Time permitting, the GUI will provide the functionality to generate a comprehensive report including the raw data and processed data in one centralized document, as well as a description of the inputs to the system and a log of any errors that occurred. Further, the GUI could include a list of predefined test conditions for the system being emulated, represented as a combo box that the user can then select from to run on their system. The test log would also include a description of the type of test run, the resulting behavior of the system, and other such information. However, this is a stretch goal, as these test conditions have not yet been fully defined and would require a considerable amount of time spent on understanding the behavior of the system model as well as the inner workings of LabView, which may not be realistically accomplished in the remaining project timeline.

Identifying Origin of Noise

The noise seen in the output of the OPAL-RT (Figure 9) is a result of many potential factors, namely:

- Improper/loose wiring
- Improper signal scaling
- Insufficient sampling rate
- Improper wire length
- Physical vibrations from surrounding equipment
- Electromagnetic interference from surrounding equipment
- Model design (H-bridge)

So far, the team has improved wiring by creating twisted pairs, shortening wire length, scaling the signals to their maximum limits, and moving the physical OPAL-RT/microcontroller interface away from other electronics and vibrations. Despite these changes, the noise persists. Future steps should include adjusting the sampling rate to see if an increase or decrease may reduce the noise. If that yields no benefits, the output data should be exported from the OPAL-RT to a readable format that can be processed in MATLAB. Using the FFT command, the frequency spectrum of the signal can be investigated. If the noise is occurring at high frequencies, the likelihood of the noise coming from the emulated H-Bridge model loaded onto the OPAL-RT can be eliminated. By reviewing the frequency spectrum of the noise, the team should be able to track down the primary source of the noise and mitigate accordingly. It is important to try to eliminate the noise programmatically rather than adding in additional hardware components such as an RC filter because additional hardware creates more opportunities for phase delay, attenuation, and additional noise.

Modifying Microcontroller Code to Allow Variable Inputs

Currently the two inputs to the microcontroller code are hardcoded 12 bit values. The team will spend a couple of weeks in the beginning of the next semester revising the code to allow variable inputs that can be adjusted via the GUI using the interrupt functionality in the hardware. The code will poll constantly for a change to the inputs. When this change is detected, the value will be stored in a register and then retrieved and processed through the appropriate function call in the existing microcontroller code during an interrupt sequence.

This change is mandatory early in the start of the next semester to allow for seamless transition with the GUI. In the code's current state, the GUI will not be able to operate on the inputs to the TI microcontroller.

Fixing Motor Speed Output Plot

The motor speed plot is incorrect. It is currently plotting the output of the motor current (which is a sawtooth wave), when it should be a sinusoidal wave. This is probably due to some selection inside the Simulink model loaded onto the OPAL-RT that needs to be accessed by the lab advisor, Chris Lute. When the team returns next semester, they will work with Chris to address this issue, since they didn't work with the Simulink model and need to check with him before

making modifications. This should be a relatively simple fix that doesn't require much troubleshooting, but ample time will be allowed in the project schedule.

Running Documentation Tool on Code

The team has elected to use Doxygen, an open source tool that parses through code of all kinds (LabView, MATLAB, C code from the microcontroller) to generate an HTML report of all the documentation written in the code. This provides a great resource and central repository of information on how to understand, modify, and use the code. This will be a very important deliverable because it will allow for users of the CHIL system to be able to make well-informed decisions when using the GUI or modifying the functionality of the system.This step will be carried out at the end of the semester as part of the closing deliverables and will be linked into the GUI for ease of access by users of the CHIL system.

REFERENCES

[1] “IEEE Code of Ethics.” Website. https://www.ieee.org/about/corporate/governance/p7-8.html (accessed Dec 5, 2021).

[2] “CSU Powerhouse.” Website. https://energy.colostate.edu/powerhouse/ (accessed Dec 4, 2021).

[3] *Systems and software engineering - System life cycle processes*, ISO/IEC/IEEE 15288:2015, 2015.

[4] J. Benitz. “What You Need to Know about the ISA 101 HMI Standard: Empowering the Operator.” Website. https://graymattersystems.com/the-4-things-you-need-to-know-about-isa-101-hmi-standard/ (accessed Dec 4, 2021).

BIBLIOGRAPHY

TI C2000 Training Workshop Videos and Labs. Website. https://training.ti.com/c2000-f2837xd-microcontroller-one-day-workshop-series?context=1134645.

R.S. Carson, “Requirements Completeness,” *Proc. INCOSE 2004 - 14th Annual International Symposium Proceedings*. 2004, pp. 930-944, doi: 10.1002/j.2334-5837.2004.tb00546.x

J.M. Borky and T.H. Bradley, *Effective Model-Based Systems Engineering,* Springer, 2019.

B.W. Kernighan and D.M. Ritchie, *The C Programming Language,* 2nd ed. Pearson, 1988.

APPENDICES

Appendix A. Abbreviations

Acronym	Definition
MEA	More Electric Aircraft
TRAS	Thrust Reverser Actuation System
EM-TRAS	Electromechanical TRAS
CHIL	Controller Hardware-in-the-Loop
GUI	Graphical User Interface
PXI	NI product; PXI = PCI eXtensions for Instrumentation
ASET	Aerospace System Emulation and Test
MBSE	Model-Based Systems Engineering
DAQ	Data Acquisition Box
cRIO	NI product; cRIO = Compact Reconfigurable I/O Module
FFT	Fast Fourier Transform
ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
HMI	Human Machine Interface
NI	National Instruments
TI	Texas Instruments
ECE	Electrical and Computer Engineering

Appendix B. Budget

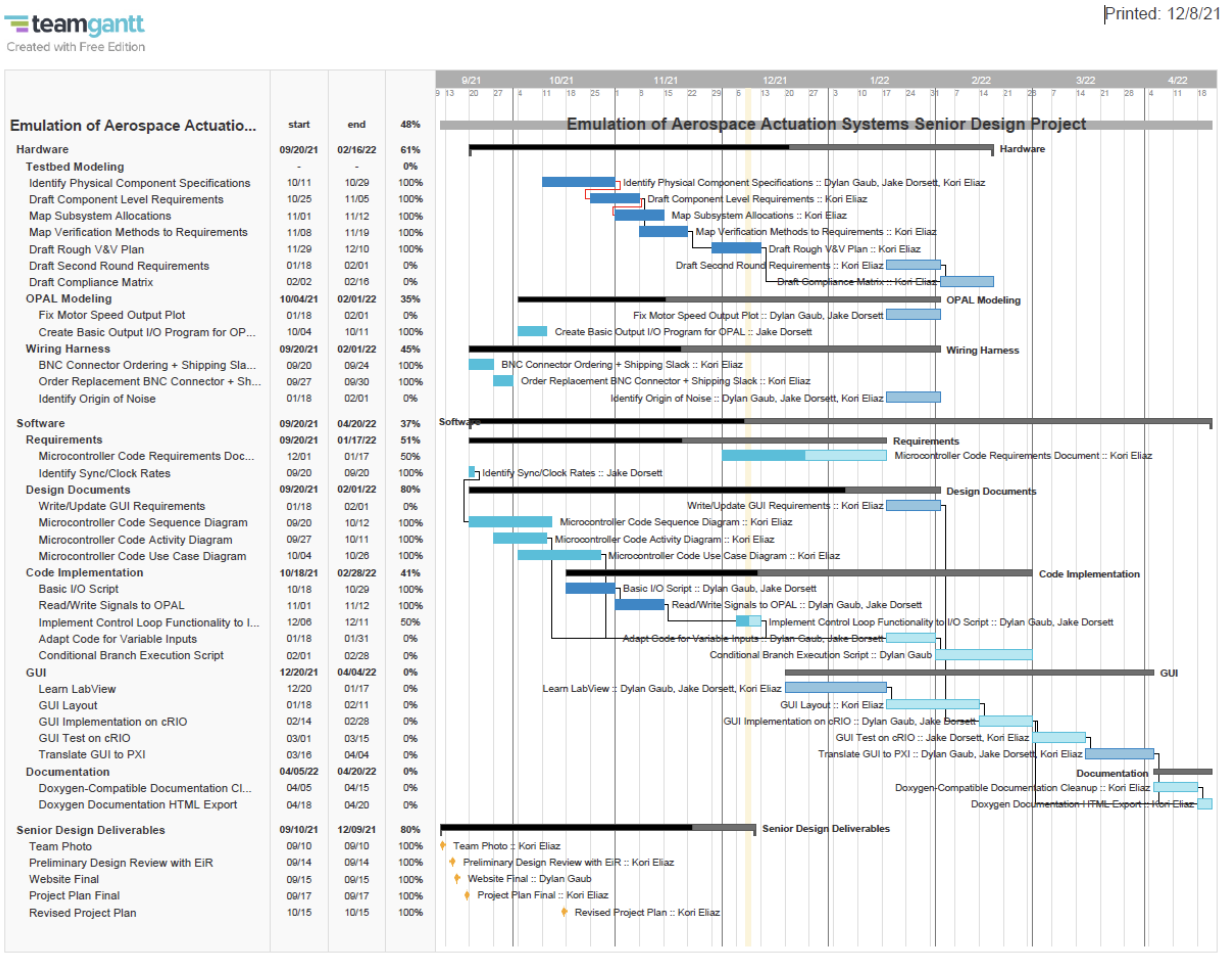
Table 3 shows the budget breakdown for this project, including the expenses already made during the fall semester and expected expenses for the spring semester. These estimates are made based on research performed or previous expenses made by the team's advisors. All expenses were paid by a student and then reimbursed through the ECE department. Because the first connector ordered was incorrect, it was returned and the team has reimbursed the ECE department the difference via check. All transactions are logged in this spreadsheet and in the individual team members' senior design notebooks. Because the majority of the expensive equipment was already purchased by the project's advisors and the majority of the work performed is software based, the budget is relatively simple for this senior design project. The team does not expect to exceed the $600 budget allocation. In the unlikely event that this value is exceeded, the team shall discuss potential funding opportunities with the professor and industry sponsor.

Table 3: Budget Breakdown

Expense	**Amt**	**QTY**	**Status**	**Purchased**	**Reimbursed**	**Justification**
BNC to MiniBNC Connector	$11.57	1	Returned	9/21/2021	9/28/2021	Connector required to interface between OPAL-RT analog output and an oscilloscope to verify signal output matches expectations.
Replacement Connector	$7.63	1	Arrived	9/27/2021	9/28/2021	The connector ordered on 9/21 did not fit the OPAL-RT. This new connector is confirmed to be the right size and does fit. We returned the wrong connector and reimbursed the department.
Additional TI Controller	$250.00	1	Pending	1/2022	TBD	An additional microcontroller will allow for the project team to keep a constant closed loop connection between the TI microcontroller and the OPAL-RT, allowing for remote manipulation of the CHIL system. This will lead to more rapid troubleshooting and accelerated project completion.
E-Days Materials	$50.00	1	Pending	4/2022	TBD	Posters, 3D printed pieces, and stickers/business cards will need to be produced for our E-Days Showcase in the Spring Semester.

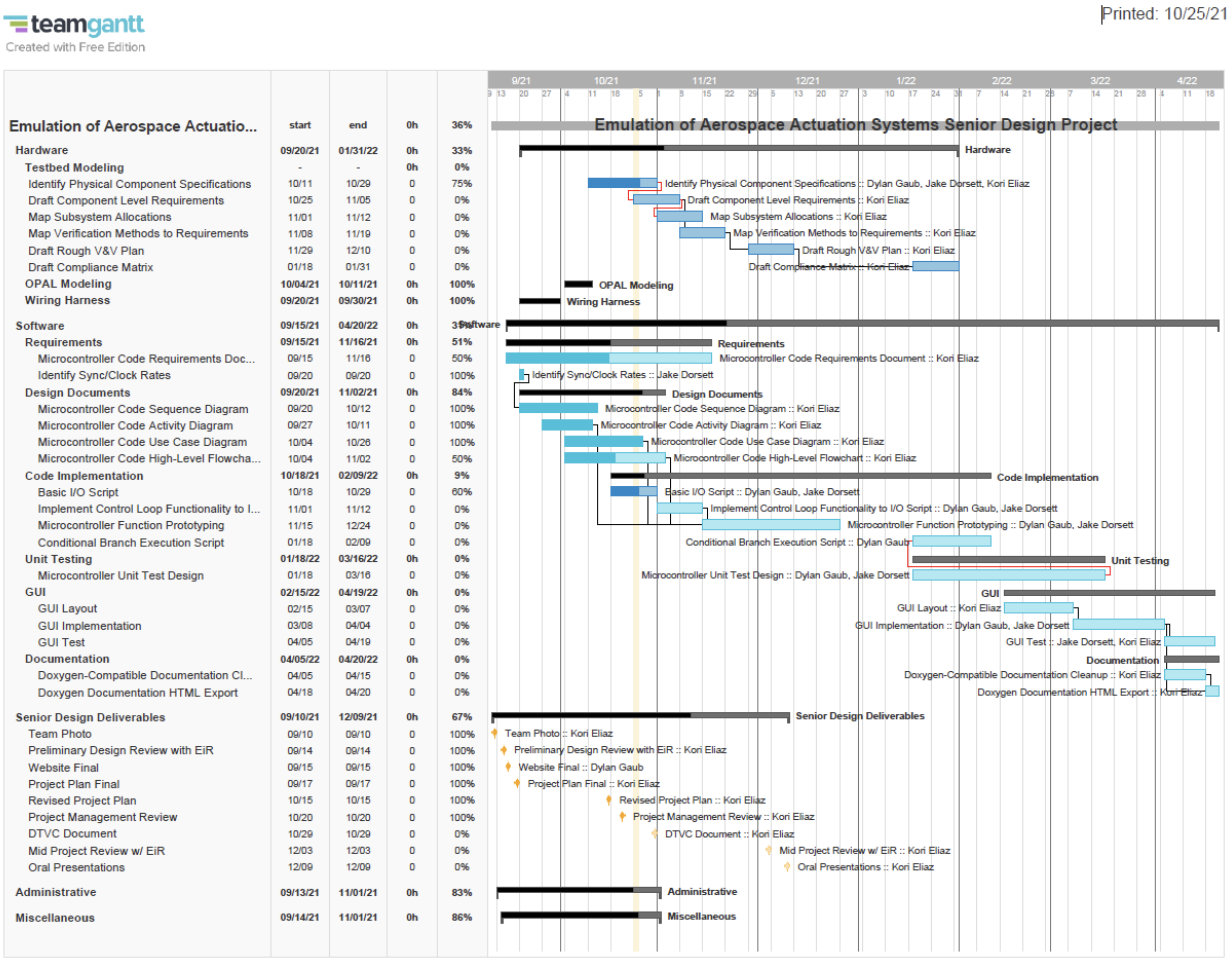
Starting Balance	**$600.00**
Total Spent	**$(319.20)**
Total Remaining	**$280.80**

Appendix C. Timeline Progression



Emulation of Aerospace Actuatio...	start	end	48%
Hardware	09/20/21	02/16/22	61%
Testbed Modeling	-	-	0%
Identify Physical Component Specifications	10/11	10/29	100%
Draft Component Level Requirements	10/25	11/05	100%
Map Subsystem Allocations	11/01	11/12	100%
Map Verification Methods to Requirements	11/08	11/19	100%
Draft Rough V&V Plan	11/29	12/10	100%
Draft Second Round Requirements	01/18	02/01	0%
Draft Compliance Matrix	02/02	02/18	0%
OPAL Modeling	10/04/21	02/01/22	35%
Fix Motor Speed Output Plot	01/18	02/01	0%
Create Basic Output I/O Program for OP...	10/04	10/11	100%
Wiring Harness	09/20/21	02/01/22	45%
BNC Connector Ordering + Shipping Sla...	09/20	09/24	100%
Order Replacement BNC Connector + Sh...	09/27	09/30	100%
Identify Origin of Noise	01/18	02/01	0%
Software	09/20/21	04/20/22	37%
Requirements	09/20/21	01/17/22	51%
Microcontroller Code Requirements Doc...	12/01	01/17	50%
Identify Sync/Clock Rates	09/20	09/20	100%
Design Documents	09/20/21	02/01/22	80%
Write/Update GUI Requirements	01/18	02/01	0%
Microcontroller Code Sequence Diagram	09/20	10/12	100%
Microcontroller Code Activity Diagram	09/27	10/11	100%
Microcontroller Code Use Case Diagram	10/04	10/26	100%
Code Implementation	10/18/21	02/28/22	41%
Basic I/O Script	10/18	10/29	100%
Read/Write Signals to OPAL	11/01	11/12	100%
Implement Control Loop Functionality to I...	12/06	12/11	50%
Adapt Code for Variable Inputs	01/18	01/31	0%
Conditional Branch Execution Script	02/01	02/28	0%
GUI	12/20/21	04/04/22	0%
Learn LabView	12/20	01/17	0%
GUI Layout	01/18	02/11	0%
GUI Implementation on cRIO	02/14	02/28	0%
GUI Test on cRIO	03/01	03/15	0%
Translate GUI to PXI	03/16	04/04	0%
Documentation	04/05/22	04/20/22	0%
Doxygen-Compatible Documentation Cl...	04/05	04/15	0%
Doxygen Documentation HTML Export	04/18	04/20	0%
Senior Design Deliverables	09/10/21	12/09/21	80%
Team Photo	09/10	09/10	100%
Preliminary Design Review with EiR	09/14	09/14	100%
Website Final	09/15	09/15	100%
Project Plan Final	09/17	09/17	100%
Revised Project Plan	10/15	10/15	100%

Fig C1 - Dec 8, 2021 Project Schedule



Emulation of Aerospace Actuatio...	start	end	0h	36%
Hardware	09/20/21	01/31/22	0h	33%
Testbed Modeling	-	-	0h	0%
Identify Physical Component Specifications	10/11	10/29	0	75%
Draft Component Level Requirements	10/25	11/05	0	0%
Map Subsystem Allocations	11/01	11/12	0	0%
Map Verification Methods to Requirements	11/08	11/19	0	0%
Draft Rough V&V Plan	11/29	12/10	0	0%
Draft Compliance Matrix	01/18	01/31	0	0%
OPAL Modeling	10/04/21	10/11/21	0h	100%
Wiring Harness	09/20/21	09/30/21	0h	100%
Software	09/15/21	04/20/22	0h	35%
Requirements	09/15/21	11/16/21	0h	51%
Microcontroller Code Requirements Doc...	09/15	11/16	0	50%
Identify Sync/Clock Rates	09/20	09/20	0	100%
Design Documents	09/20/21	11/02/21	0h	84%
Microcontroller Code Sequence Diagram	09/20	10/12	0	100%
Microcontroller Code Activity Diagram	09/27	10/11	0	100%
Microcontroller Code Use Case Diagram	10/04	10/26	0	100%
Microcontroller Code High-Level Flowcha...	10/04	11/02	0	50%
Code Implementation	10/18/21	02/09/22	0h	9%
Basic I/O Script	10/18	10/29	0	60%
Implement Control Loop Functionality to I...	11/01	11/12	0	0%
Microcontroller Function Prototyping	11/15	12/24	0	0%
Conditional Branch Execution Script	01/18	02/09	0	0%
Unit Testing	01/18/22	03/16/22	0h	0%
Microcontroller Unit Test Design	01/18	03/16	0	0%
GUI	02/15/22	04/19/22	0h	0%
GUI Layout	02/15	03/07	0	0%
GUI Implementation	03/08	04/04	0	0%
GUI Test	04/05	04/19	0	0%
Documentation	04/05/22	04/20/22	0h	0%
Doxygen-Compatible Documentation Cl...	04/05	04/15	0	0%
Doxygen Documentation HTML Export	04/18	04/20	0	0%
Senior Design Deliverables	09/10/21	12/09/21	0h	67%
Team Photo	09/10	09/10	0	100%
Preliminary Design Review with EiR	09/14	09/14	0	100%
Website Final	09/15	09/15	0	100%
Project Plan Final	09/17	09/17	0	100%
Revised Project Plan	10/15	10/15	0	100%
Project Management Review	10/20	10/20	0	100%
DTVC Document	10/29	10/29	0	0%
Mid Project Review w/ EiR	12/03	12/03	0	0%
Oral Presentations	12/09	12/09	0	0%
Administrative	09/13/21	11/01/21	0h	83%
Miscellaneous	09/14/21	11/01/21	0h	86%

Fig C2 - Oct 25, 2021 Project Schedule

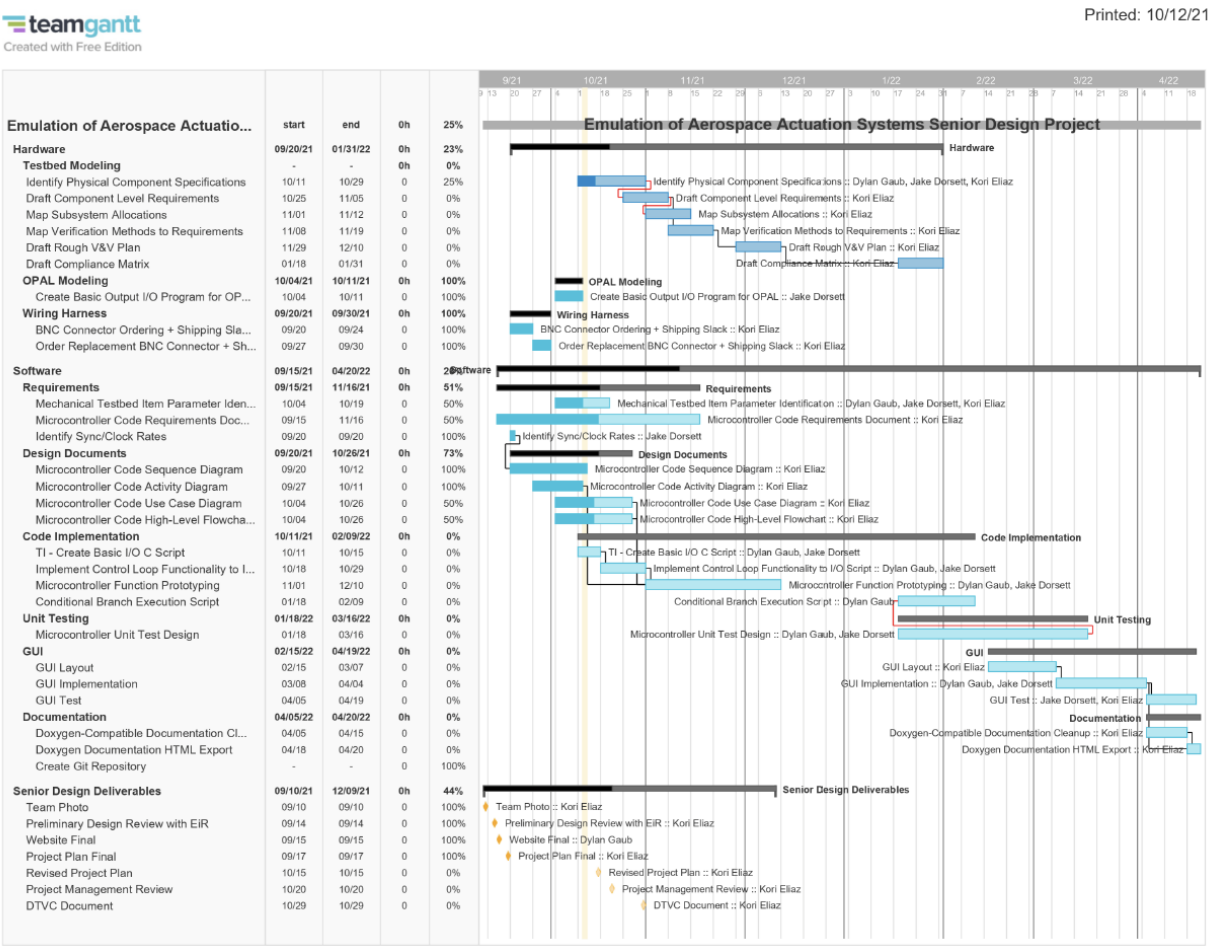


Fig C3 - Oct 12, 2021 Project Schedule

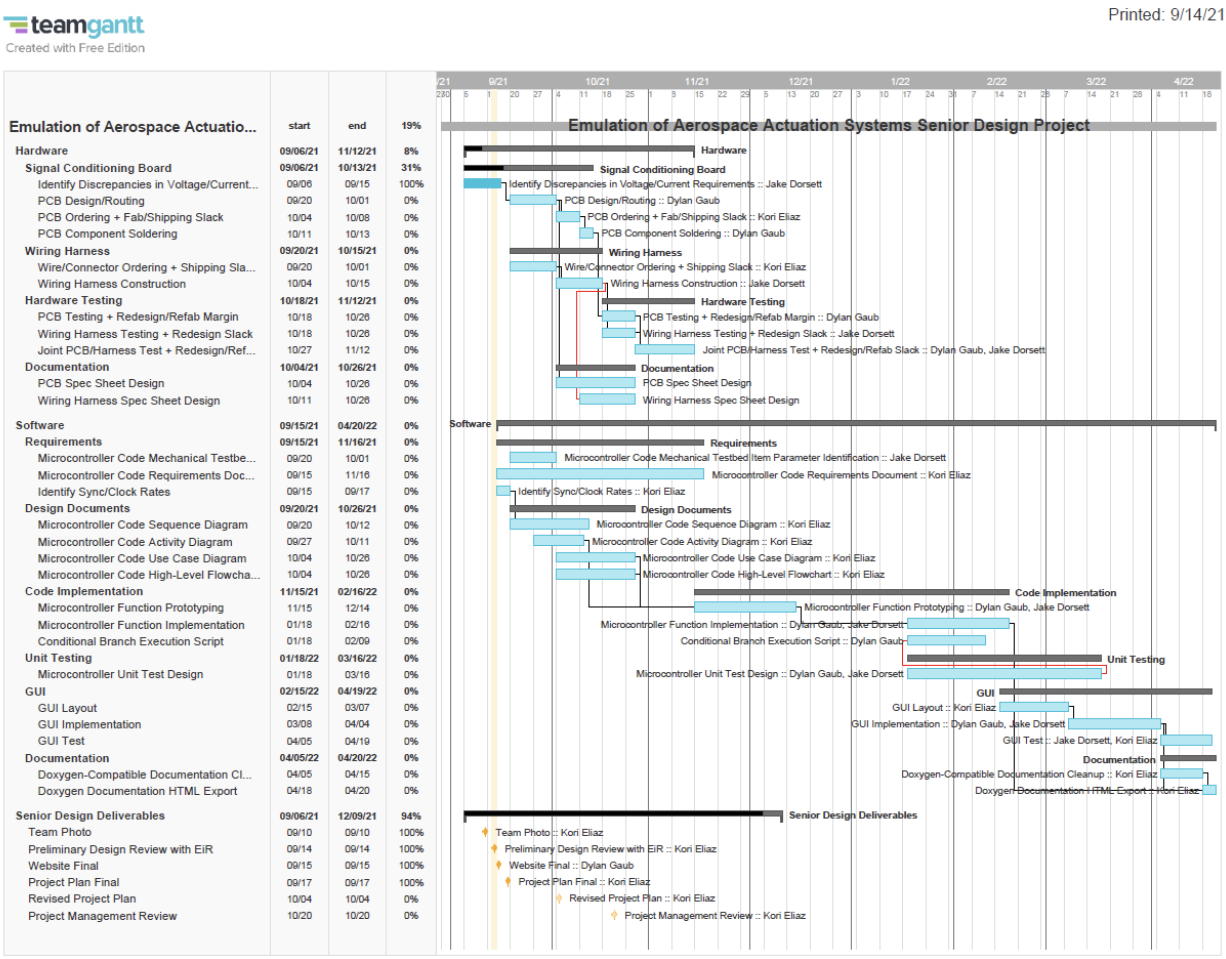


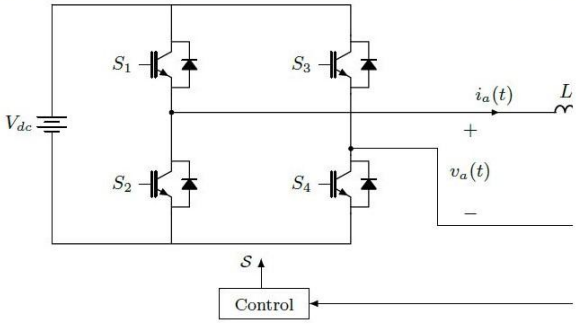
Fig C4 - Sep 14, 2021 Project Schedule

Appendix D. Spring Semester Plan

Deliverable	**Duration**	**Due Date**	**Assigned Member(s)**
Fix Motor Speed Output Plot	1 week	1/25/2022	Jake, Dylan
Adapt Code to Allow Variable Inputs	2 weeks	2/1/2022	Jake, Dylan
Write/Update GUI Requirements	2 weeks	2/1/2022	Kori
Identify Origin of Noise	2 weeks	2/1/2022	All
Design/Validate GUI for cRIO	8 weeks	3/14/2022	All
Translate GUI to PXI	2 weeks	4/4/2022	All
Run Documentation Tool	1 week	4/11/2022	Kori
Design E-Days Display	1 week	4/18/2022	All

Appendix E. MBSE Artifacts

Table E - High Level CHIL Requirements

#	Name	Text	Id	Satisfied By	Applied Stereotype
1	1 Overall Mission Requirements	Compiled Emulation System Requirements	1	1.1 Overall Functional Requirements 1.2 Overall Non-Functional Requireme	Requirement [Class] HyperlinkOwner [Element]
2	1.1 Overall Functional Requireme	Specific functional capabilities of the Emulation System	1.1	1.1.1 Hardware 1.1.2 Software	Requirement [Class] HyperlinkOwner [Element]
3	1.1.1 Hardware	Specific functional capabilities of all hardware components	1.1.1	1.1.1.1 Microcontroller 1.1.1.2 Real-Time HIL Simulator	Requirement [Class] HyperlinkOwner [Element]
4	1.1.1.1 Microcontroller	The Emulation System shall employ a TI TMS320F28379D microcontroller to process input commands from the real-time HIL simulator.	1.1.1.1	1.1.1.1.1 Microcontroller CPU Usage 1.1.1.1.2 Microcontroller GPIO Low Vc 1.1.1.1.3 Microcontroller GPIO High V 1.1.1.1.4 Microcontroller Input Voltag	Requirement [Class]
5	1.1.1.1.1 Microcontroll	The Microcontroller shall only enable 1 of 2 CPU's to ensure that only 1 debugger is active at a time.	1.1.1.1.1		Requirement [Class]
6	1.1.1.1.2 Microcontroll	The Microcontroller GPIO low pin voltage shall be set to not greater than 1.5 V.	1.1.1.1.2		Requirement [Class]
7	1.1.1.1.3 Microcontroll	The Microcontroller GPIO high pin voltage shall be set to not less than 1.6 V.	1.1.1.1.3		Requirement [Class]
8	1.1.1.1.4 Microcontroll	The Microcontroller shall not accept digital input signals outside the range of 0-3.3 V.	1.1.1.1.4		Requirement [Class]
9	1.1.1.2 Real-Time HIL Sim	The Emulation System shall employ an OPAL-RT 5600 real-time simulator to create input commands sent to the testbed microcontroller.	1.1.1.2	1.1.1.2.1 RT Simulator Signal Conditio 1.1.1.2.2 RT Simulator Model Node Cc	Requirement [Class]
10	1.1.1.2.1 RT Simulator	The RT Simulator shall not send digital signals outside the range of 0-3.3 V to the microcontroller without a proper signal conditioning board.	1.1.1.2.1		Requirement [Class]
11	1.1.1.2.2 RT Simulator	The RT Simulator shall run a model of the testbed with not more than [TBD-02] nodes to match the capabilities of the machine as dictated by the quantity of its processing cores.	1.1.1.2.2		Requirement [Class] TBD [Element]
12	1.1.2 Software	Specific functional capabilities of all software components	1.1.2	1.1.2.1 Simulated Model 1.1.2.2 Microcontroller Libraries 1.1.2.3 Synchronization Rate	Requirement [Class] HyperlinkOwner [Element]
13	1.1.2.1 Simulated Model	The Emulation System shall operate based on a DC machine Simulink model loaded onto the OPAL-RT machine that reflec 	1.1.2.1		Requirement [Class]
14	1.1.2.2 Microcontroller Lib	The software shall adhere to pre-defined functions in the TI driverlib library to streamline code implementation.	1.1.2.2		Requirement [Class]
15	1.1.2.3 Synchronization R	The software shall sample inputs from the OPAL-RT at no less than the Nyquist rate of [TBD-01] Hz.	1.1.2.3		Requirement [Class] TBD [Element]
16	1.2 Overall Non-Functional Requi	Extraneous/non-functional capabilities of the Emulation System	1.2	1.2.1 GUI Standard 1.2.2 CSU Facility Constraints 1.2.3 CSU Safety Guidance 1.2.4 Woodward Proprietary Guidanc 1.2.5 Allowed Languages 1.2.6 Documentation Export	Requirement [Class] HyperlinkOwner [Element]
17	1.2.1 GUI Standard	The Emulation System shall be monitored by a Graphical User Interface that adheres to the ISA101 HMI Standard.	1.2.1		Requirement [Class] Non-Functional Requirement [Element
18	1.2.2 CSU Facility Constraints	The Emulation System shall comply with CSU facility constraints on space and power.	1.2.2		Requirement [Class] Non-Functional Requirement [Element
19	1.2.3 CSU Safety Guidance	The Emulation System shall comply with CSU safety guidance parameters.	1.2.3		Requirement [Class] Non-Functional Requirement [Element
20	1.2.4 Woodward Proprietary	The Emulation System shall comply with restrictions imposed in the Woodward NDA signed by all team members.	1.2.4		Requirement [Class] Non-Functional Requirement [Element
21	1.2.5 Allowed Languages	All code written to interface with the TI Microcontroller shall be written in C. All scripts used to parse through code results shall be written in Python.	1.2.5		Requirement [Class] Non-Functional Requirement [Element
22	1.2.6 Documentation Export	The Emulation System shall package and export all software documentation into an HTML file using the Doxygen documentation tool.	1.2.6		Requirement [Class] Non-Functional Requirement [Element

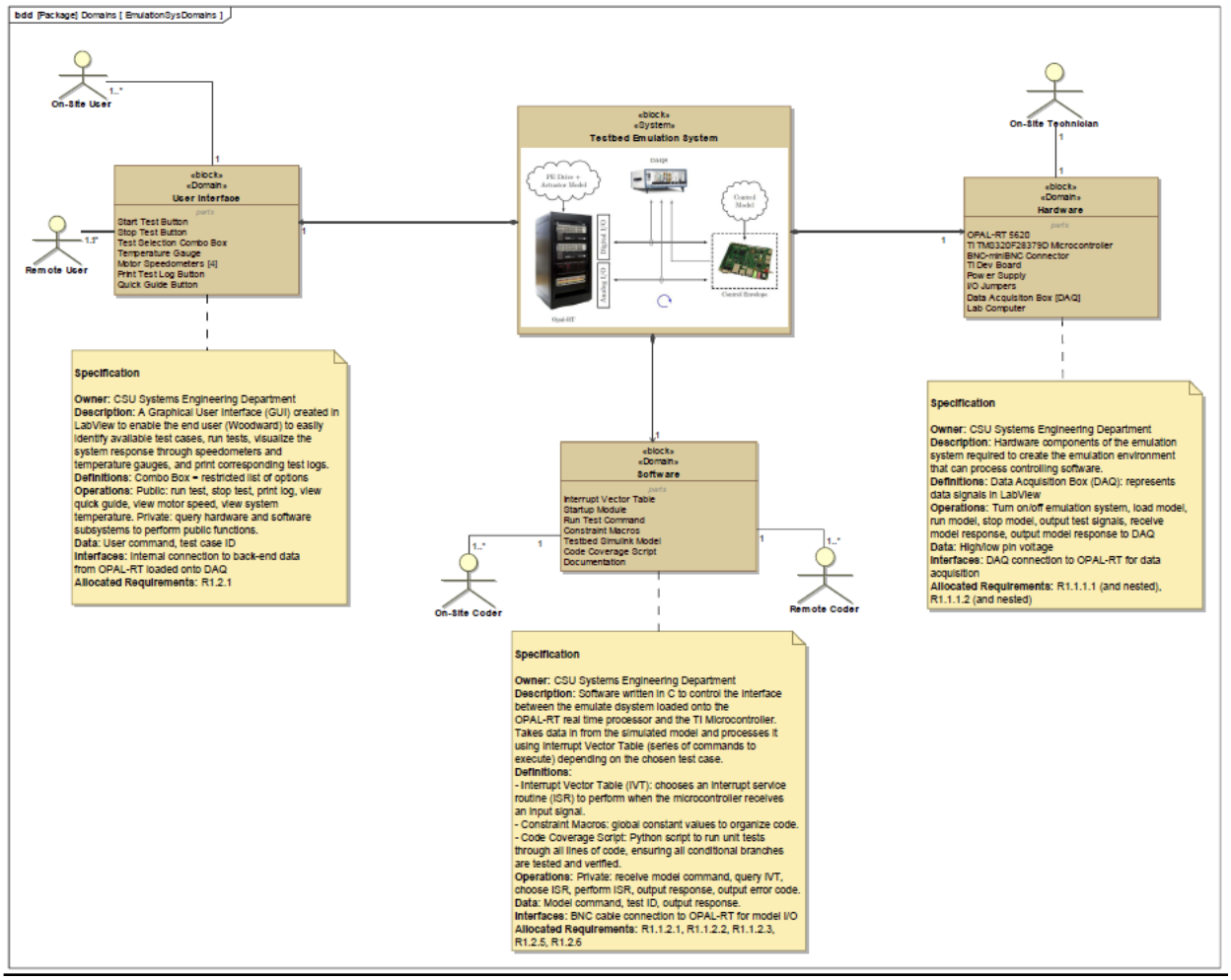


Fig E1 - Domains of the CHIL System

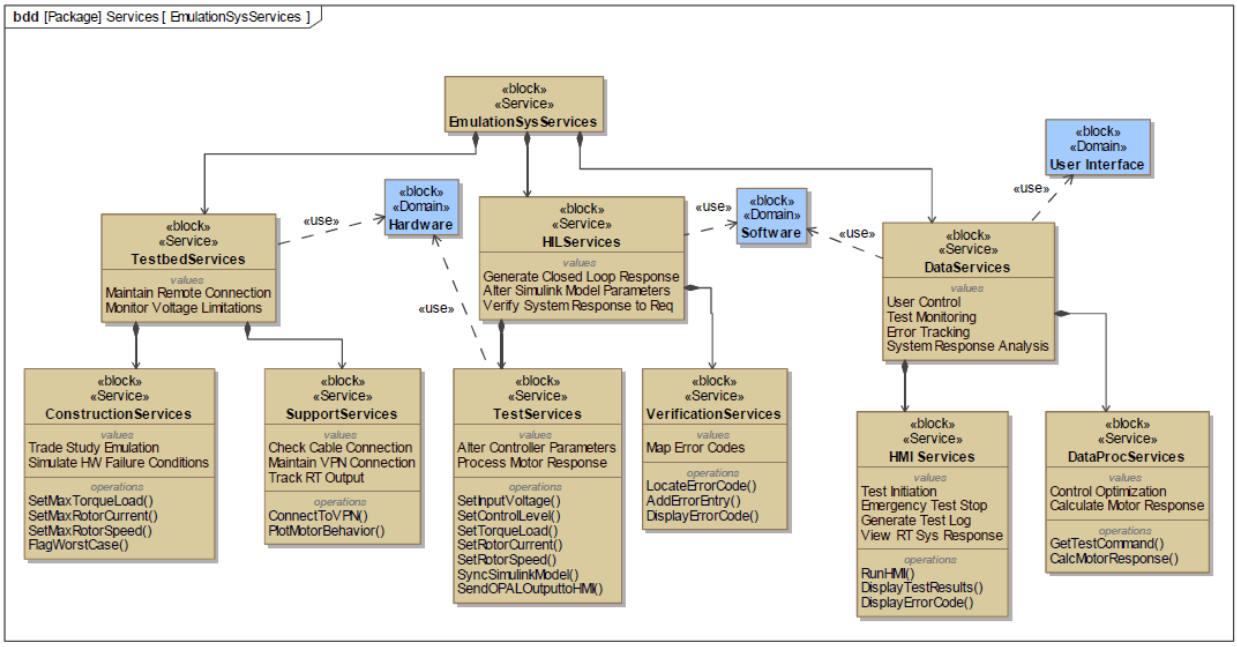


Fig E2 - CHIL Platform Services

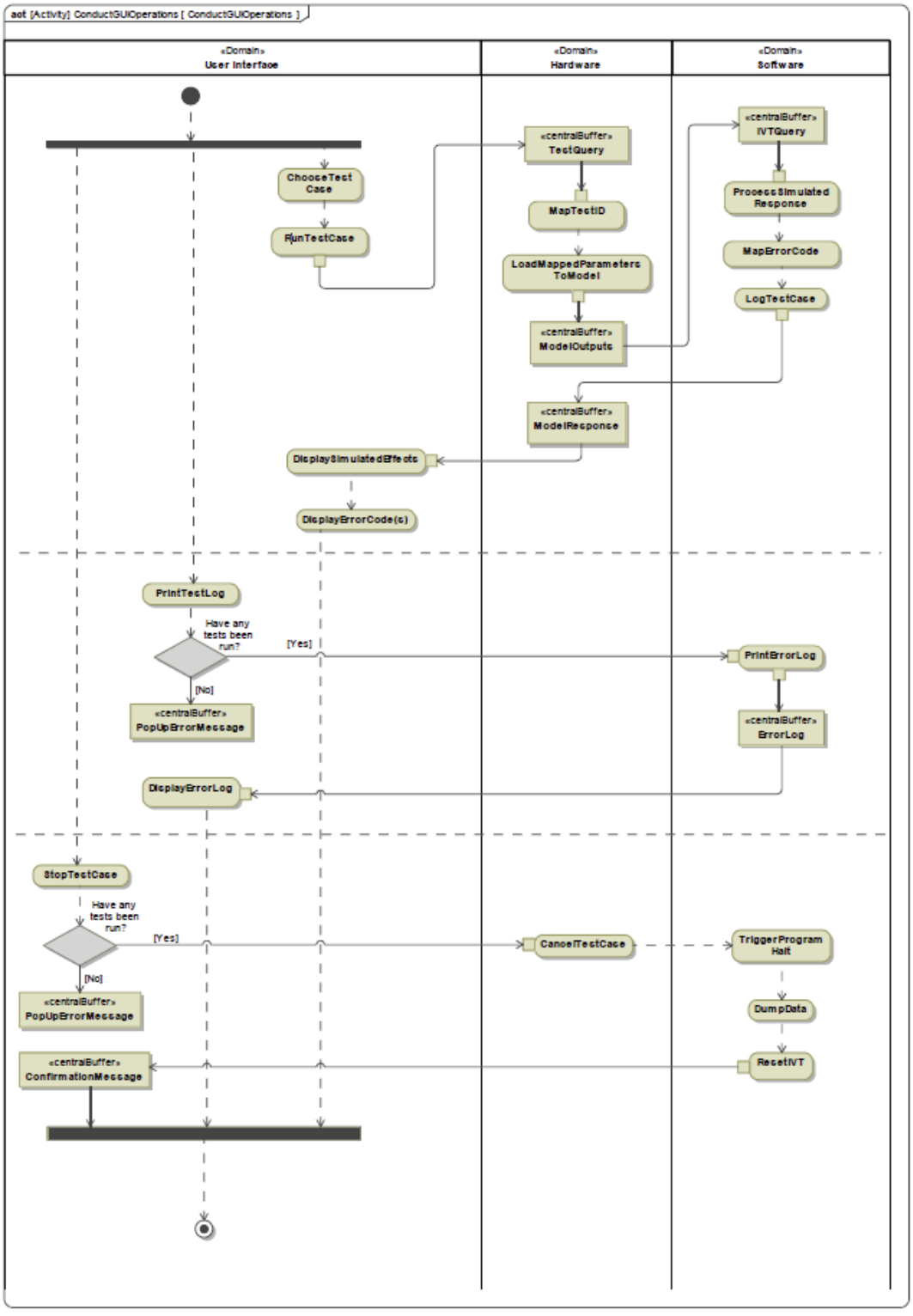


Fig E3 - Activity Diagram Showing GUI Operations

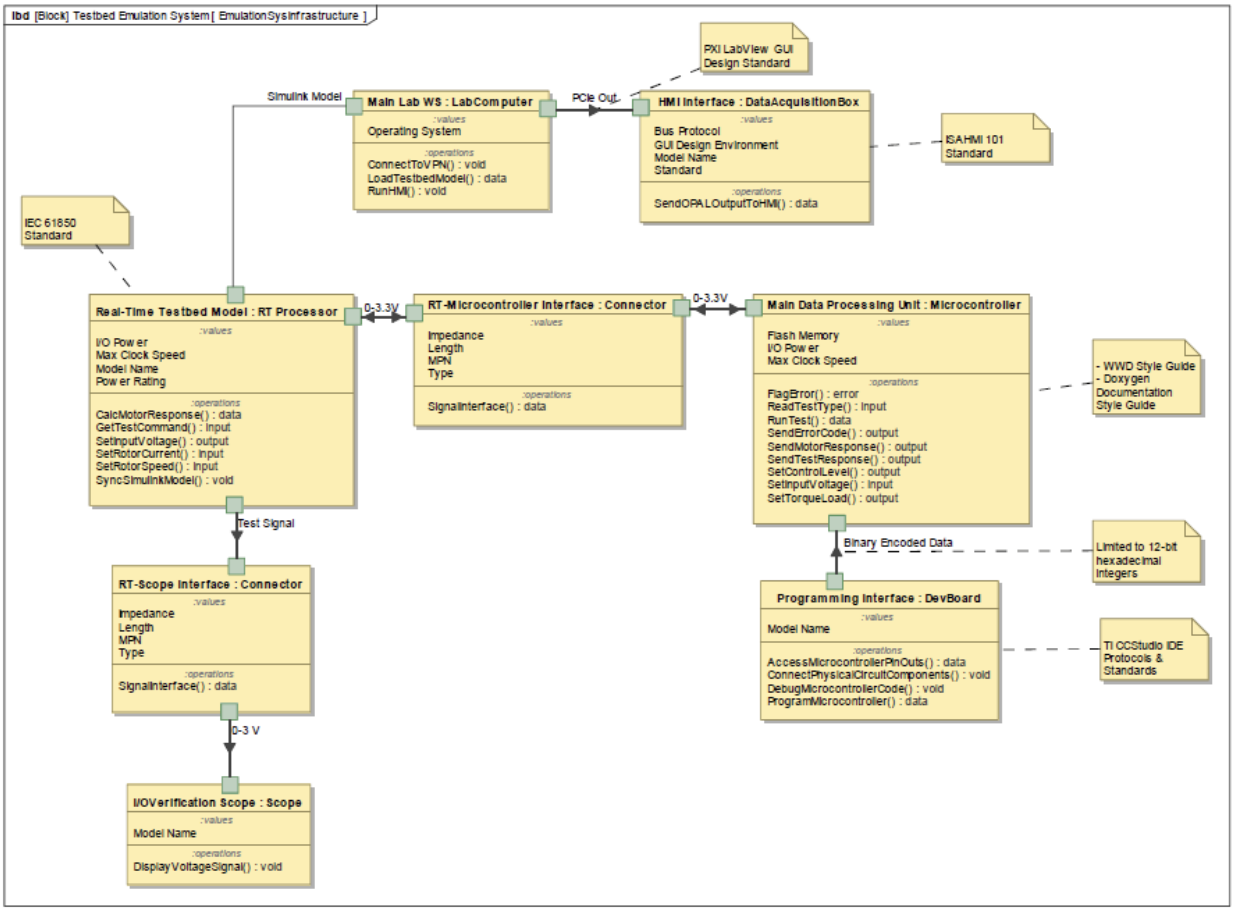


Fig E4 - Hardware Interfaces w/ Governing Standards/Protocols

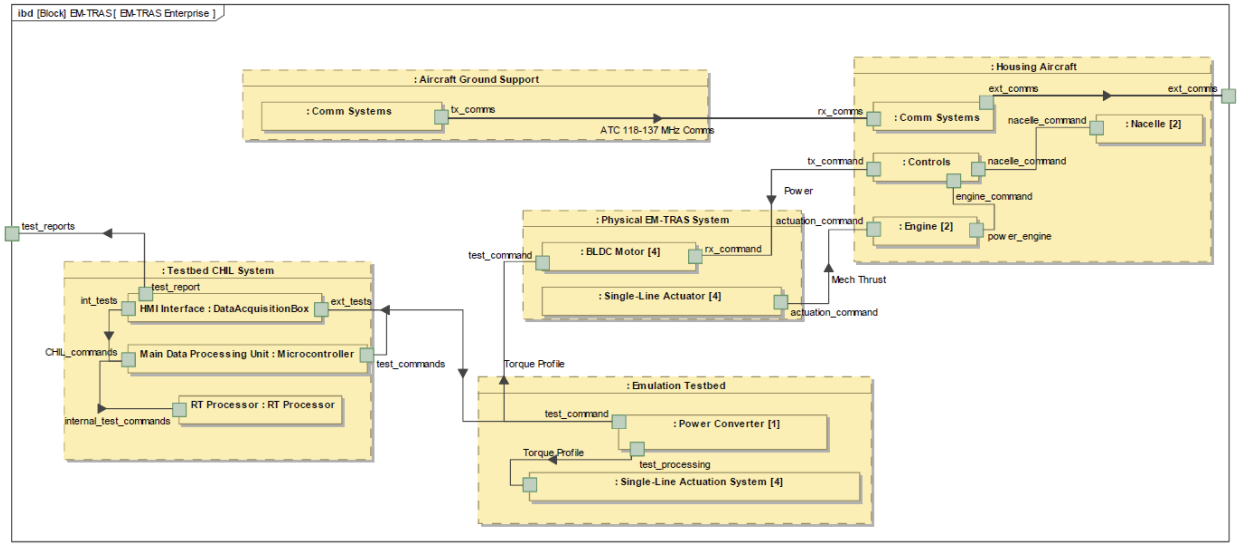


Fig E5 - CHIL Platform Role in the EM-TRAS Enterprise

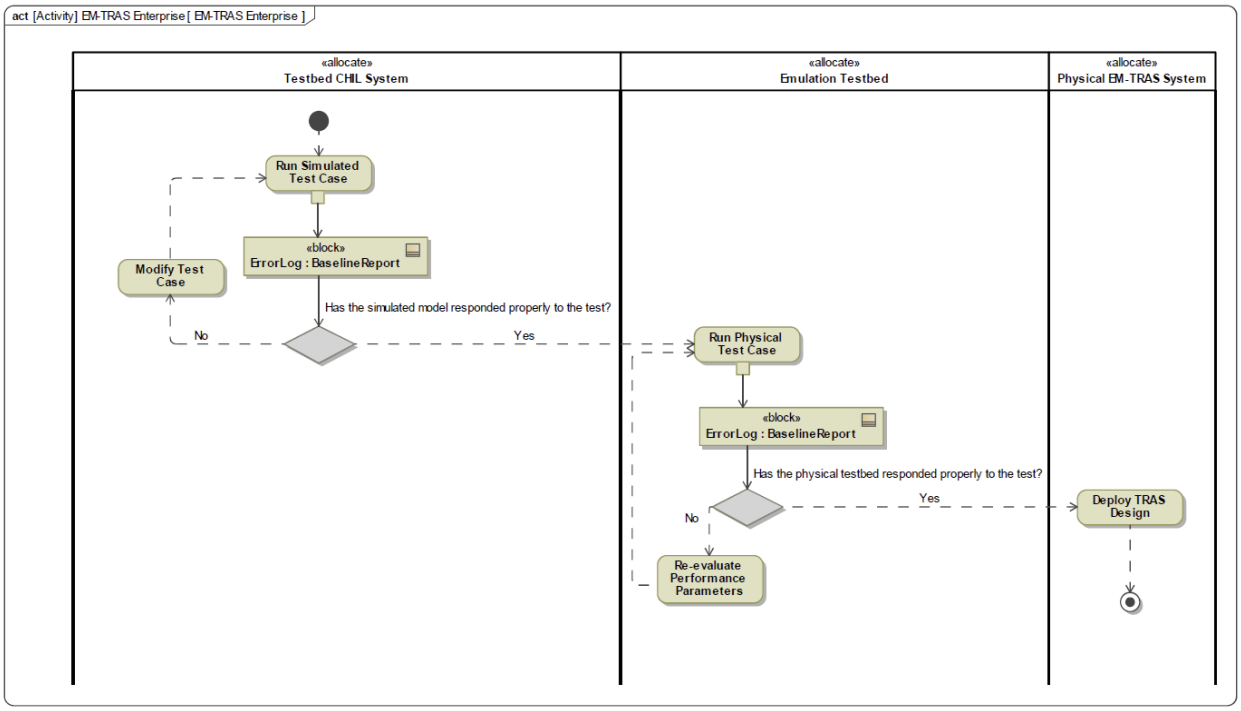


Fig E6 - Verification of EM-TRAS Design from CHIL to Physical Implementation

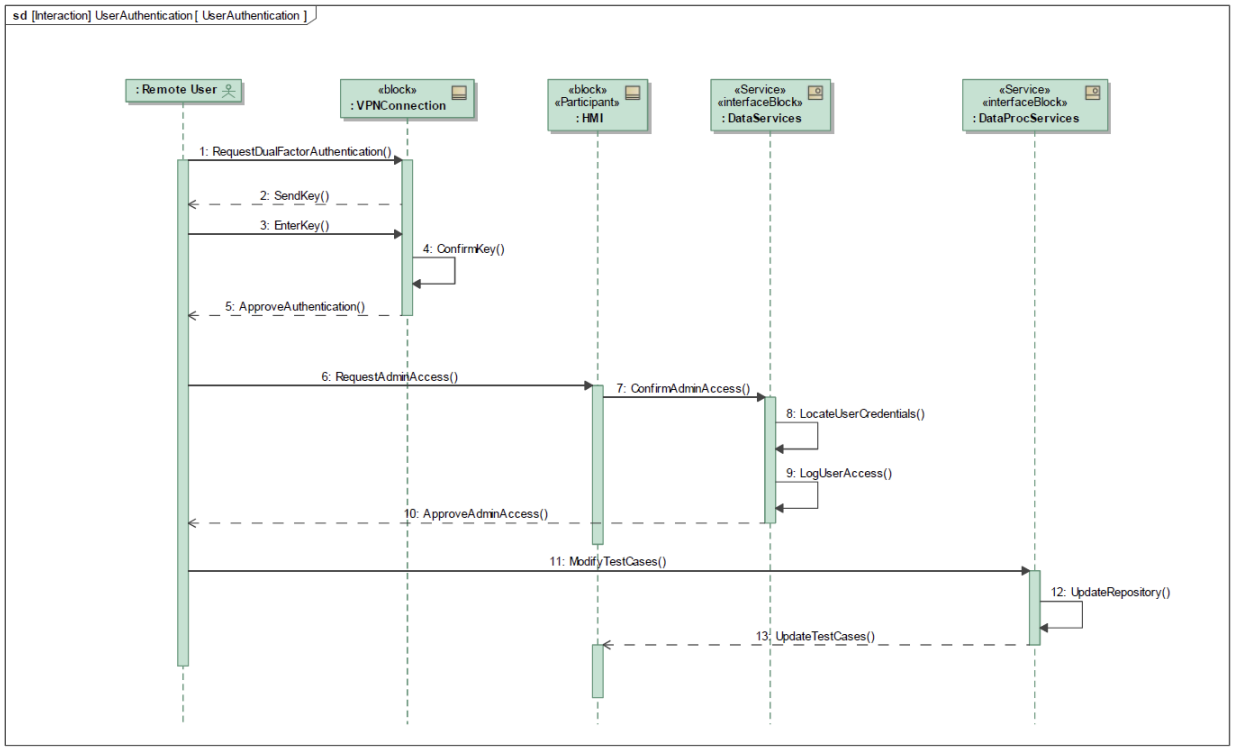


Fig E7 - Remote User Authentication Process

ACKNOWLEDGMENTS

We would like to acknowledge the following people for their support of our efforts this semester:

- *Dr. James Cale*, for giving us a fantastic opportunity to learn about C-HIL, to employ multiple facets of both electrical and systems engineering, a well-supplied lab space to work in, and to develop resume-building skills.
- *Matt Heath (EiR)*, for supporting our weekly meetings, providing much-needed input from a customer perspective, supporting our MBSE endeavors, and making himself available for questions.
- *Dr. Kamran Efektari Shahroudi*, for supporting our MBSE endeavors on the proprietary side of this project.
- *Chris Lute*, for his constant technical assistance in the lab and effort to continue our team's progress.
- *Selim Madcadi*, for allowing us to participate in the manual construction of the mechanical EM-TRAS testbed and supporting our MBSE endeavors.
- *Jayesh Narsinghani,* for supporting our MBSE endeavors and training us on proper requirements-writing format.
- *Claudio Lima,* for his support with the TI microcontroller code.
- *Ethan Schultz,* for providing a foundational understanding of the TI microcontroller.
- *Sarah Dorsett*, for designing an amazing logo.

We couldn't have made progress without the support of these key people and we are excited to continue working with those still involved in the project in the Spring semester.