SlugSat Spring Report

Testing Equipment Subteam

ECE 129 Senior Capstone

September 2019 - June 2020

Project Professors: Stephen Petersen (AC6P), Tela Favaloro

Subteam Lead: Caleb Cotter (KN6GEE)

Subteam Members: Ali Elagamawi, Nick Jannuzzi, (KN6GEN), Juan Ayala (KN6AMD)

Table of Contents

Table of Contents 1

List of Tables 5

List of Figures 6

Acronyms and Abbreviations 9

Introduction 10
- Motivation 10
- Space Environment 10
- NASA Testing Requirements for CubeSats 11

3-Axis Helmholtz Coil 12
- Introduction 12
- Design Requirements 13
  - Performance Requirements 13
  - Safety Requirements 14
  - User Interface Requirements: 16
- Prior Art 17
- Proposed Technical Solution 18
- Coil Design 19
  - Relevant Design Requirements 19
  - MATLAB Calculations & Theoretical Properties 21
  - Coil Cage Mechanical Design 22
- Low Friction Satellite Mount 26
- Measuring Coil Properties 28
- Control Electronics 35
  - Control Electronics Enclosures and Panel 47
- Control Design 48
- User Interface 49
  - Relevant Design Requirements 49
  - Overview of User Interface Architecture 50
  - UI Layout 51
  - USB Communication Protocol 51
  - Null Offset Calibration Routine 53
- Results 55
- Standard Operating Procedures 55

Future Work 56
Conclusion 57
Mechanical Shock Test Stand (STS) 58
Introduction 58
Design Requirements 59
Prior Art 60
Intro to Shock Testing 60
Pyrotechnic vs. Classical Shock 60
Pyro 60
Classical 61
Proposed Technical Solution 62
Selection of Shock Delivery Method 62
Form and Function of Solution 63
Shock Testing Procedure (Proposed) 64
Derivation of Engineering Requirements 64
Generation of the Shock Response Spectrum 64
Translation to Subsystem Requirements 66
Shock Delivery Mechanism 67
Shock Delivery Method 67
Test Stand Structure Design 68
Form Factor of Test Stand 68
Impact Hammer 68
Shock Deflection Plate 69
Hardware Mounting 72
Summary of Mechanical Design 73
Data Acquisition System 74
Engineering Requirements 74
Sensor Selection 75
Microcontroller Selection 76
Block Diagram 76
Data Recording 77
Firmware Architecture 79
Data Post-Processing 79
Summary of Data Acquisition System 81
Standard Operating Procedure 82
Future Work 83
State of the Deliverable 83
State of the Subsystems 84

Data Acquisition System 84
Shock Delivery Mechanism 84
Conclusion 85
Solar Vector Simulator 86
Introduction 86
Design Requirements 87
Spectral Match 87
Spatial Non-uniformity 89
Temporal Instability 89
Performance Requirements 90
Prior Art 91
Proposed Technical Solution 94
Light Sources: LED and QTH Spectrum 95
Relevant Design Requirements 95
Theory and Relevant Equations 95
Matlab Spectral Match Simulation 99
Light Source Components 100
Solar Simulator Panel 103
Relevant Design Requirements 103
Light Source Position 103
Panel Housing 105
Power Distribution and Driver 105
Relevant Design Requirements 105
Theory and Relevant Equations 106
Power Supply 108
Power Distribution 108
Driver Circuit 109
Solar Simulator Prototype. 110
Non-Uniformity Experiment 111
Future Work 115
State of Deliverable 115
Conclusion 116
Proposed Future Testing Equipment 117
Vibroacoustic Test Stand 117
Appendix A: HHC Engineering Schematics 120
Software A.1 120
Mechanical A.2 123

Electrical A.3 129

Appendix B: STS Engineering Schematics 134

B.1 Software 134

Appendix C: SVS Engineering Schematics and Scripts 145

[C.1] Spectral Match Assessment Matlab Script 145

[C.2.1] Spectral Match and Non-Uniformity Matlab Script 150

[C.2.2] LED Irradiance Model 158

[C.3] Light Source Driver Circuit 160

[C.4] Light Sources Part Table 161

[C.5] Non-Uniformity Experiment Matlab Script 161

Appendix D: References 163

List of Tables

TE
Table TE.1: *Table of Requirements for Vibroacoustic Test Stand Rev. 1 (12/5/19)*

HHC
Table HHC.1a: Magnetic and Power Requirements for 3-Axis HHC Rev. 2 (3/8/20)
Table HHC.1b: Safety Design requirements for 3-Axis HHC Rev. 2 (3/8/20)
Table HHC.1c: UI Design requirements for 3-Axis HHC Rev. 2 (3/8/20)
Table HHC.2: Table of Theoretical Coil Properties given Nominal Parameter Selection
Table HHC.3: Coil Inductance Theoretical vs Experimental
Table HHC.4: Coil Series Resistance Theoretical vs Experimental
Table HHC.5: Coil Frequency Response Simulation Result Comparison
Table HHC.6: Tradeoff analysis of USB vs. RS-232 serial communication Rev. 1.0 6/7/20
Table HHC.7: Flowchart of Coil Field Measurement Process Rev. 1.0 6/3/20

STS
Table STS.1: Design Requirements for STS Rev. 2 (6/6/20)
Table STS.2: Tradeoff Analysis of Shock Delivery Methods Rev. 1 (4/29/20)
Table STS.3: Tradeoff Analysis of Shock Delivery Methods Rev. 1 (4/29/20)
Table STS.4a: Table of Derived Accelerometer Requirements for STS DAQ Rev. 1 (5/11/20)
Table STS.4b: Table of Derived Encoder Requirements for STS DAQ Rev. 1 (5/11/20)
Table STS.5: Comparison of Accelerometers for STS Rev. 1 (6/7/20)
Table STS.6: Selected Encoder Relevant Specifications Rev. 1 (6/7/20)
Table STS.7: STS design requirements with compliancy status as of Spring 2020 Rev. 1 (6/4/20)

SVS
Table SVS.1: ASTM Spectral Match Classification Comparison Chart Rev. 1 (6/7/20)
Table SVS.2: ASTM Spectral Match Classification Rev. 1 (6/7/20)
Table SVS.3: ASTM Non-uniformity Classification Rev. 1 (6/7/20)
Table SVS.4: ASTM Temporal Instability Classification Rev. 1 (6/7/20)
Table SVS.5: SVS Performance Requirements Rev. 3 (5/31/20)
Table SVS.6: Radiometric and photometric quantities and units
Table SVS.7: Chosen components for color spectrum Rev. 2 (6/7/20)
Table SVS.8: Classification of Spectral Match for multiple waveband Rev. 1 (6/7/20)
Table SVS.9: Required Voltage Source for LED strings Rev. 1 (6/7/20)
Table SVS.10: State of Deliverable table Rev. 1 (6/7/20)

List of Figures

TE

Figure TE.1 CubeSat General Testing Flow Diagram
Figure TE.2: Proposed Block Diagram of Vibroacoustic Test Stand (12/7/19)

HHC

Figure HHC.1: SlugSat Year 3 Single Axis HelmHoltz Coil
Figure HHC.2: Single Axis Helmholtz Coil Illustration
Figure HHC.3: Nested Helmholtz Coil Pairs to form 3-Axes of Orthogonal Magnetic Fields
Figure HHC.4: Helmholtz System Block Diagram Illustrating a One of Three Axis Rev. 1 (12/1/19)
Figure HHC.5: Helmholtz Mechanical Design Overview Render (2/12/2020)
Figure HHC.6: Assembled Bare Frame Mockup
Figure HHC.7: CAD Render of HHC Core Ready for Transport, Narrowest Width is 24”
Figure HHC.8: Photo of Progress on HHC, Coils Complete with Electronics Mockup (3/11/2020)
Figure HHC.9: Diagram of Airflow and Air Bearing Principle of Operation (5/3/2020)
Figure HHC.10: Diagram of Airflow and Air Bearing Principle of Operation (5/3/2020)
Figure HHC.11: 1-Dimensional Stability Example (5/3/2020)
Figure HHC.12: Air Bearing Mounting on Satellite Riser in HHC (5/3/2020)
Figure HHC.13: Agilent 4263B, Frequency = 100KHz, LVL = 1000mV
Figure HHC.14: LCR Meter Test Setup
Figure HHC.15: Coil Resistance and Inductance vs Frequency
Figure HHC.16: VNA Test Setup for Coil Properties
Figure HHC.17: Smith Chart Results from Coil VNA Test
Figure HHC.18: VNA Audio Calibration Results
Figure HHC.19: LC Tank Circuit Used to Measure Inductance via Resonance
Figure HHC.20: Theoretical LC Tank Frequency Response of Circuit in Fig. 3 w/ L coil of 10mH
Figure HHC.21: Multimeter Coil Resistance Test Setup
Figure HHC.22: Schematic Coil Simulation-Series LR (Rev 1.0 3/10/20)
Figure HHC.23: Current through Inductor L1 for Series LR (Rev 1.0 3/10/20)
Figure HHC.24: Test Results for Current vs ADC Reading - ACS711EX Current Sensor (2/18/2020)
Figure HHC.25: X-axis H-bridge Input Signals (Direction & PWM) (3/2/2020)
Figure HHC.26: Fuse Indication Circuit across H-bridge Voltage Input (3/2/2020)
Figure HHC.27: X-axis H-bridge to X-axis Current Sensor Wiring (3/2/2020)
Figure HHC.28: Power Regulation for Logic Voltages (3/2/2020)
Figure HHC.29: Motherboard PCB Layout V0.6 (3/25/2020)
Figure HHC.30: Motherboard 3D Top View (reference [A.3.5] for Bottom View) V0.6 (3/25/2020)
Figure HHC.31: HHC Motherboard Perfboard Version (Top & Bottom View) V2.0, 5/13/2020
Figure HHC.32: USB Connection Galvanic Isolation Block Diagram
Figure HHC.33: HMI Electrical Schematic V1.0

Figure HHC.34: HMI 3D Isometric View Rev 1.0
Figure HHC.35: Finished Perfboard Electronics in Enclosure (6/1/2020)
Figure HHC.36: HHC Control Panel Wired and Mounted on Coil Cage (6/1/2020)
Figure HHC.37: Block Diagram of HHC Control Architecture Rev. 1.0 6/4/20
Figure HHC.38: Block Diagram of HHC PID Controller Library Rev. 2 (3/8/20)
Figure HHC.39: Overview of the user interface architecture Rev. 1.0 6/7/20
Figure HHC.40: Example of NMEA0183 encoded message Rev 1.0 6/8/20
Figure HHC.41: User Interface error handling routine Rev 2.0 6/7/20
Figure HHC.42: User Interface null offset calibration routine Rev. 2.0 6/7/20
Figure HHC.43: HHC UI after calibration, with null offsets displayed on screen
Figure HHC.44: Flowchart of Coil Field Measurement Process Rev. 1.0 6/3/20

STS

Figure STS.1: Pyroshock Testing of a 3U Cubesat
Figure STS.2: Shock Test Stand Proposal Diagrams (Rev. 1.0 12-13-19)
Figure STS.3: Flowchart of SRS Generation Process (Rev. 1 5-12-20)
Figure STS.4: SRS Generation Results for 3250G Sine - 740us Period (Rev. 1 5-10-20)
Figure STS.5: Angled View of STS System Assembly (Rev. 2.0 6-3-20)
Figure STS.6: Flowchart of STS Impact System Design Process (Rev. 1.0 6-4-20)
Figure STS.7: Closeup view of STS arm encoder and arm mounting (Rev. 1.0 6-4-20)
Figure STS.8: STS Data Acquisition System Block Diagram Rev. 2 (5/29/2020)
Figure STS.9: STS Data Sampling w/ Ping-Pong Buffers Flowchart Rev. 1 (6/7/2020)
Figure STS.10: Screenshot of STS Data-Analysis Program (Rev. 2 6-4-2020)
Figure STS.11: Finished STS Electronics with SD Memory Installed (Rev. 1 6-2-2020)
Figure STS.12: Block Diagram of STS standard operation procedure Rev 2.0 6/4/20

SVS

Figure SVS.1: 2000 ASTM Standard Extraterrestrial Spectrum Reference. Orange shade is visible light spectrum Rev. 1 (6/7/20)
Figure SVS.2: Typical LED Solar Simulator Rev. 1 (6/7/20)
Figure SVS.3: Solid State Solar Simulator
Figure SVS.4: Solar Vector Simulator System Diagram Rev. 4 (6/7/20)
Figure SVS.5: Illustration for calculating irradiance Rev. 1 (6/7/20)
Figure SVS.6: Lambertian Distribution Pattern and formula Rev. 1 (6/7/20)
Figure SVS.7: Illustration for calculating irradiance Rev. 1 (6/7/20)
Figure SVS.8: Luminosity Intensity Equation Rev. 1 (6/7/20)
Figure SVS.9: Spectral Irradiance of each component Rev. 2 (5/30/20)
Figure SVS.10: Spectral Irradiance of components Rev. 2 (6/7/20)
Figure SVS.11: Component XY position in elevated plane of 0.118m Rev. 2(6/7/20)
Figure SVS.12: Irradiance distribution on test area surface Rev. 2 (6/7/20)
Figure SVS.13: Solar Simulator Prototype Housing Rev. 1 (6/7/20)
Figure SVS.14: Power supply and driver system diagram Rev. 2 (6/7/20)

Figure SVS.15: Luminous flux vs current curve of white LED provided by OSRAM [D.10]
Figure SVS.16: Forward Current flux vs voltage of white LED. OSRAM [D.10]
Figure SVS.17: Simplified Driver Circuit Rev. 2 (6/7/20)
Figure SVS.18: Images of Solar Simulator prototype. Mid & Left: Front Solar Simulator. Right: driver circuits and back of Solar simulator Rev. 2 (6/10/20)
Figure SVS.19: The power distribution circuit using boost converters Rev. 1 (6/7/20)
Figure SVS.20: Image on the left is the current source with heat sinks. On the right is the PWM to analog converter and potentiometers Rev. 1 (6/10/20)
Figure SVS.21: Non-Uniformity Experiment. The solar simulator is within a box and emits light onto a piece of paper. The Camera senses the “intensity” of the image. Rev. 1 (6/10/20)
Figure SVS.22: Non-Uniformity Experiment Setup Front view. Rev. 1 (6/10/20)
Figure SVS.23: Non-Uniformity Experiment Setup Back view. Rev. 1 (6/10/20)
Figure SVS.24: Non-Uniformity Experiment Image Capture at sheet of paper. The image on the right is the cropped image only including the area within the drawn square. Rev. 1 (6/10/20)
Figure SVS.25: Average Experimental Uniformity Rev. 1 (6/10/20)

Acronyms and Abbreviations

ACS: Attitude Control System
ADC: Analog To Digital
ASTM: ASTM
CLK: Clock
DCM: Direction Cosine Matrix
DMA: Direct Memory Access
EMF: Electromotive Force
eV: Electron Volt
FCC: Federal Communications Commission
FSM: Finite State Machine
GPIO: General Purpose Input Output
GPS: Global Positioning System
I^2C: Inter-Integrated Circuit
IC: Integrated Circuit
IGRF: International Geomagnetic Reference Field
IMU: Inertial Measurement Unit
IR: Infrared
SNR: Signal to Noise Ratio
SOP: Standard Operating Procedure
SPI: Serial Peripheral Interface
SVS: Solar Vector Simulator
UTC: Universal Coordinated Time
XO: Crystal Oscillator
2U: Two Unit
LEO: Low Earth Orbit
LPF: Low Pass Filter
NED: North East Down (Reference Frame)
PCB: Printed Circuit Board
PID: Proportional Integral Derivative (Controller)
PMSM: Power Modes State Machine
P-Pod: Poly-Picosatellite Orbital Deployer
PPS: Pulse Per Second
PWM: Pulse-Width Modulation
QTH: Quartz Tungsten Halogen
RBF: Remove Before Flight
RTC: Real Time Clock

Introduction

The testing equipment subteam was created to build tools and fixtures to aid in the development and validation of the spacecraft.

Motivation

The year 4 team was the first SlugSat team to have a testing equipment subgroup. Previous project years have focused on the construction of a functional prototype called a FlatSat. This prototype is only designed to function on a lab bench and be used as test hardware to validate the satellite bus and payload technologies. The majority of the SlugSat team will be focusing on the construction of the engineering qualification unit and the ground station.

This year's team goal necessitated the creation of testing equipment to validate the integrated EU and its subsystems. The testing equipment subteam team was in charge of overseeing the design of the qualification testing equipment for the EU, such as the 3-Axis Helmholtz Coil system for attitude control system (ACS) tuning and testing.

Space Environment

On-orbit and during launch, the spacecraft will be subjected to a variety of harsh environmental factors. During launch of the craft it will be subject to powerful acoustic vibrations and rapid depressurization as the launch vehicle ascends. Once the craft is placed in orbit, it will be subjected to large thermal differentials from the sun-facing side, high energy radiation, and vacuum conditions. Launch is a one-time opportunity and therefore it is critical that the engineering unit is tested in the same conditions the craft will see during its mission. This ensures that design flaws are uncovered on the ground and gives time to address and fix issues before the vehicle is launched. Before launch can be approved, the craft must conform to NASA General Environmental Verification Standard (GEVS, GSFC-STD-7000).

NASA Testing Requirements for CubeSats

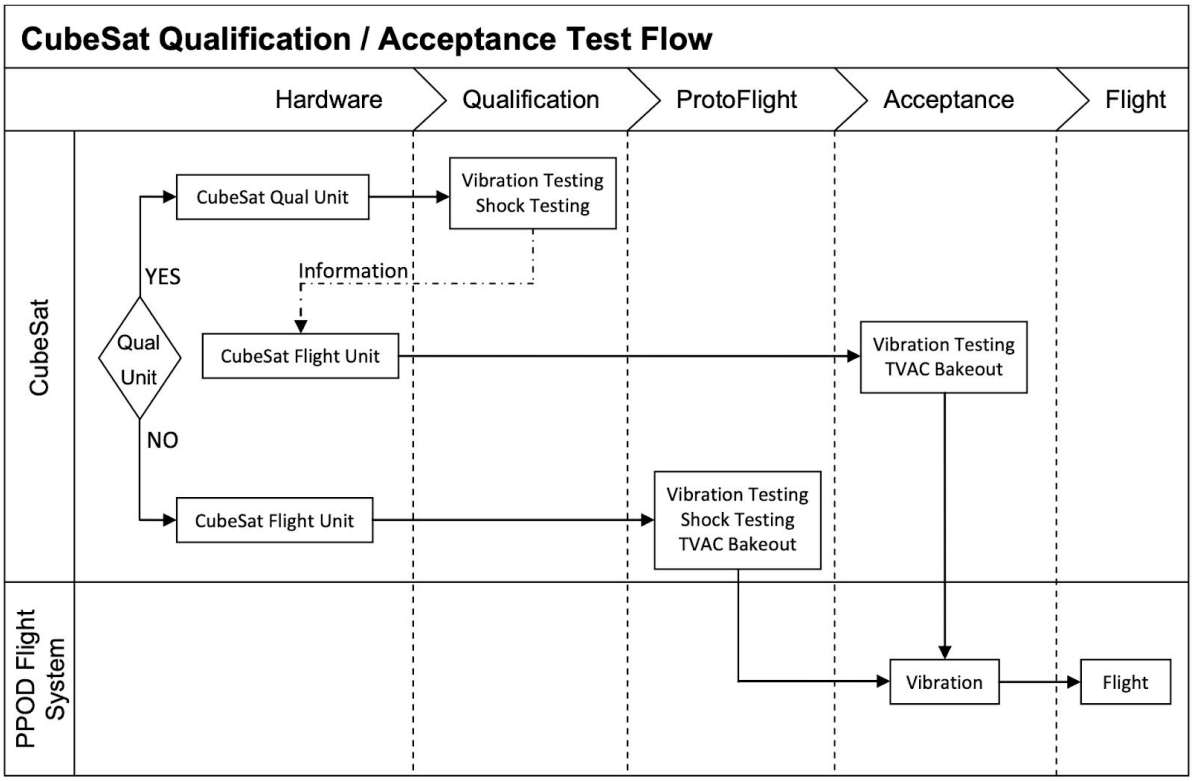


Figure TE.1 CubeSat General Testing Flow Diagram

(Cubesat Design Specification - pg. 15 - Revision 13 - 2/20/2014)

Cubesat qualification, validation, and acceptance guidelines are set and managed by NASA and satellite launch providers. The requirements of a launch provider supersede the NASA guidelines for compliance to the launch vehicle. Figure M.1 is provided in the cubesat design specification which shows the recommended test flow for development. This is the testing process that the SlugSat will be following. Year 4 is focusing on developing the CubeSat Qual Unit which is also known as the Engineering Unit or EU for short.

Official testing for acceptance of the flight unit must be done on industry grade calibrated equipment and will be done by the launch provider pre and post PPOD integration. The test equipment constructed by the SlugSat team will be reference equipment and instead of being used for official qualification, it will guide the engineering process and initial qualification.

3-Axis Helmholtz Coil

Caleb Cotter, Nick Jannuzzi, Ali Elagamawi

Introduction

The ACS of SlugSat utilizes magnetorquers to interface with ambient magnetic fields, and utilizes magnetic fields as a reference vector in the ACDS software. Therefore, a method of producing controllable magnetic fields was deemed necessary by last year's team. This requires the construction of an HHC, which is commonly used in satellite development. Additionally, since our ACS accuracy requirements necessitate the usage of reaction wheels in addition to the magnetorquers, a frictionless surface to simulate a 3DOF environment was also identified as a needed form of testing equipment.

A helmholtz coil is a device used to produce a region of nearly uniform magnetic field. Consisting of 2 electromagnets comprising a single axis, helmholtz coils can also be used to negate an existing magnetic field, such as the Earth's. For the testing conditions required for the ACS, we determined that a 3-Axis HHC would be necessary to test the full stack of the satellite and validate the ACDS algorithm. In addition, the construction of a frictionless air bearing comprising the HHC's payload mounting platform would also allow testing of the reaction wheels' functionality.

Project Documentation, Code, CAD Models, and Schematics: https://github.com/SlugSat/3HC

Design Requirements

The key design requirements of each coil of the 3-Axis HHC relate to the produced magnetic and any safety considerations arising from the power supply necessary to generate the desired magnetic field. Additionally, design requirements also included volume constraints, to allow easy disassembly and transferring of the HHC between lab spaces.

Performance Requirements

Requirement ID	Requirement
HHC_001	Magnetic Field Strength shall be greater than +/- 130uT per-axis.
HHC_002	Magnetic Field Resolution shall be less than 50 nT / LSB.
HHC_003	Magnetic Field Ripple shall be less than 600 nT peak to peak.
HHC_004	Power Drawn from the outlet shall be less than 15A @ 120VAC.
HHC_005	Power Connector to outlet shall be NEMA 5-15P.
HHC_006	Air bearing shall use less than 2.6 S.C.F.M @ 100 PSI of compressed air.
HHC_007	Device must be made to be transportable and shall have a collapsed width of less than 24 inches in one dimension.

Table HHC.1a: Magnetic and Power Requirements for 3-Axis HHC Rev. 2 (3/8/20)

HHC_001:

With the maximum field strength of the Earth's magnetic field being +/- 65 microTeslas in any one direction, requirement HHC_001 requires the ability to generate a field of +/- 130 microTeslas per-axis. This will allow full negation of the Earth's field component and up to 65 microTeslas in the opposite direction.

HHC_002:

The World Magnetic Model or WMM, is a large spatial scale representation of the Earth's magnetic field [M.1], and will be used as the reference for the IMU on the ACS. One of several standard geomagnetic models, it consists of the 12th degree and order spherical harmonic expansion of the magnetic potential of the geomagnetic field generated in the Earth's core. With the WMM having an uncertainty value for each component, HHC_002 sets a minimum field resolution greater than the uncertainty of the WMM.

HHC_003:

This requirement is set to limit the amount of the AC currents induced inside the coil. The reason for selection of 600nT as the limit

HHC_004:
This establishes a maximum power draw of 15A at 120V AC, allowing operation of the HHC from a single 15A power supply connected to a standard US wall outlet.

HHC_005:
Continuing from HHC_004, HHC_005 requires the power connector to the wall outlet to be a standard NEMA 5-15P plug.

HHC_006:
Relating to the air bearing, HHC_006 requires that the air bearing shall use less than 1 S.C.F.M @ 100 PSI, allowing the air bearing to run on a standard pancake air compressor

HHC_007:
Due to the uncertainty over where the HHC would be stored long term, HHC_007 was created to allow it to fit through any standard US door width once disassembled.

Safety Requirements

Requirement ID	Requirement
HHC_008	All voltage rails >5.25VDC shall be isolated from the control computer.
HHC_009	No high voltages shall be exposed outside of the enclosure.
HHC_010	Electrical cabinets shall be able to lock to avoid accidental access by users.
HHC_011	Warning labels exterior to any enclosure which contains dangerous voltage or current.
HHC_012	An “E-Stop” or Emergency Stop shall be easily accessible from the outside of the enclosure.
HHC_013	All system voltages <50VDC, except for the 120VAC line voltage from the wall to the power supply.
HHC_014	System shall be protected from overcurrent through the use of either fuses, magnetic thermal circuit breakers or other equivalent protection device.

Table HHC.1b: Safety Design requirements for 3-Axis HHC Rev. 2 (3/8/20)

HHC_008:
In order to protect the control computer, any voltage rails greater than 5.25V DC must be isolated from the computer per HHC_008.

HHC_009:
For general safety measures, HHC_009 requires any high voltages to be unexposed from outside surfaces.

HHC_010:
To avoid access from those not familiar with the safety standards of the HHC, HHC_010 will require the electronics cabinet of the HHC to be locked to prevent unwanted access to internal electronics.

HHC_011:
Per HHC_011, standard warning labels indicating high voltages and high current will be placed on the exterior of any cabinet containing said hazards.

HHC_012:
In order to quickly stop the HHC in case of an emergency, HHC_012 will require an easily accessible E-Stop button to be present on the outside of the electronics enclosure.

HHC_013:
As any voltage higher than 50VDC would be considered by OSHA as capable of passing through the human body, all voltages other than the 120V AC power supply must fall under this threshold as part of HHC_013.

HHC_014:
HHC_014 protects the system from overcurrent by requiring either fuses, thermal circuit breakers, or other equivalent circuit protection methods.

User Interface Requirements:

Requirement ID	Requirement
HHC_015	The HHC User Interface shall require no installation of additional software
HHC_016	The User Interface shall be compatible with multiple platforms (Mac, Windows, Linux)
HHC_017	Software shall be easy to maintain, relying on well documented, open-source libraries
HHC_018	User interface shall be able to perform bidirectional communication with HHC controller via USB

Table HHC.1c: UI Design requirements for 3-Axis HHC Rev. 2 (3/8/20)

HHC_015:
In order to allow easy setup of the HHC, the HHC UI application will be a standalone executable, requiring no additional installations other than the application itself.

HHC_016:
Adding on to HHC_015, HHC_016 specifies that the application will have versions compatible with each platform, such that the application can be run on any computer.

HHC_017:
For the purpose of maintainability and readability, HHC_017 requires the usage of well documented libraries with stable releases for all features.

HHC_018:
As the HHC UI will be both displaying data readouts and sending commands to the control firmware, bidirectional communication via USB interface is required. USB was selected due to being present on all computers, as well as stable libraries for USB existing in multiple programming languages.

Prior Art

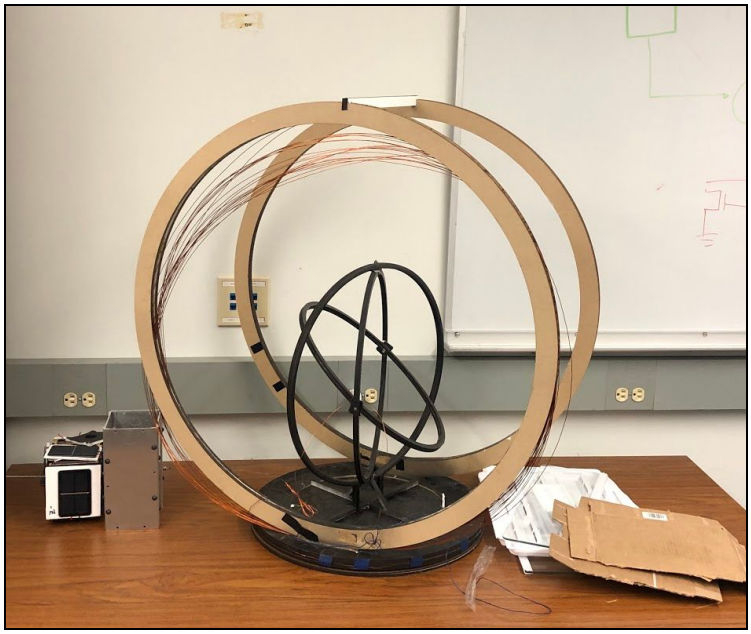


Figure HHC.1: SlugSat Year 3 Single Axis HelmHoltz Coil

In Year 3 of the SlugSat Program, the team constructed a small wooden Helmholtz Coil as a proof of concept. This design allowed only the testing of a single axis at a time, requiring the gimbal in the center to be turned by hand to test different axes. As the gimbal is not a frictionless rotating element, accurate testing of the reaction wheels would not be possible using this design.

PolySat Helmholtz Coil

Cal Poly San Luis Obispo, PolySat Program designed and constructed a 3-Axis Helmholtz Coil for the purposes of ACS Testing for their CubeSat program. Prior to design construction of our solution, we read their reports and documentation to find parts that we could improve upon. In their solution, we saw 3 key areas for improvement, software design, electronics, and integration.

Proposed Technical Solution

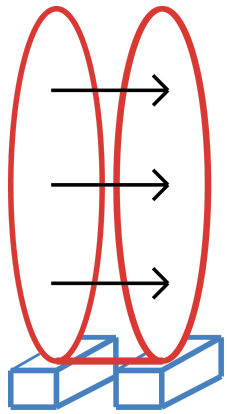


Figure HHC.2: Single Axis Helmholtz Coil Illustration

A single pair of Helmholtz coils form the basis of the design. The goal of the system is to neutralize the local magnetic vector due to Earth's dipole. A helmholtz coil is composed of two identical electromagnetic coils connected in series and separated one half radius apart. By Kirchoff's current law, the current flowing through both coils and thus the magnetic field generated is identical. The geometry of the pair creates a highly-uniform field in the center.

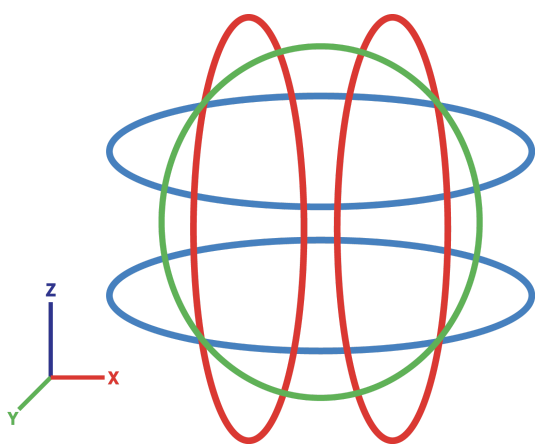


Figure HHC.3: Nested Helmholtz Coil Pairs to form 3-Axes of Orthogonal Magnetic Fields

By nesting 3 pairs of coils such that the fields are orthogonally aligned, all components of the field can be controlled. If the coils are designed to generate double the natural field on earth, then the natural field can be neutralized and any net field can be generated. This is the principle our design shall use to simulate the changing magnetic field on orbit to validate and tune the satellites attitude control system and attitude determination system.

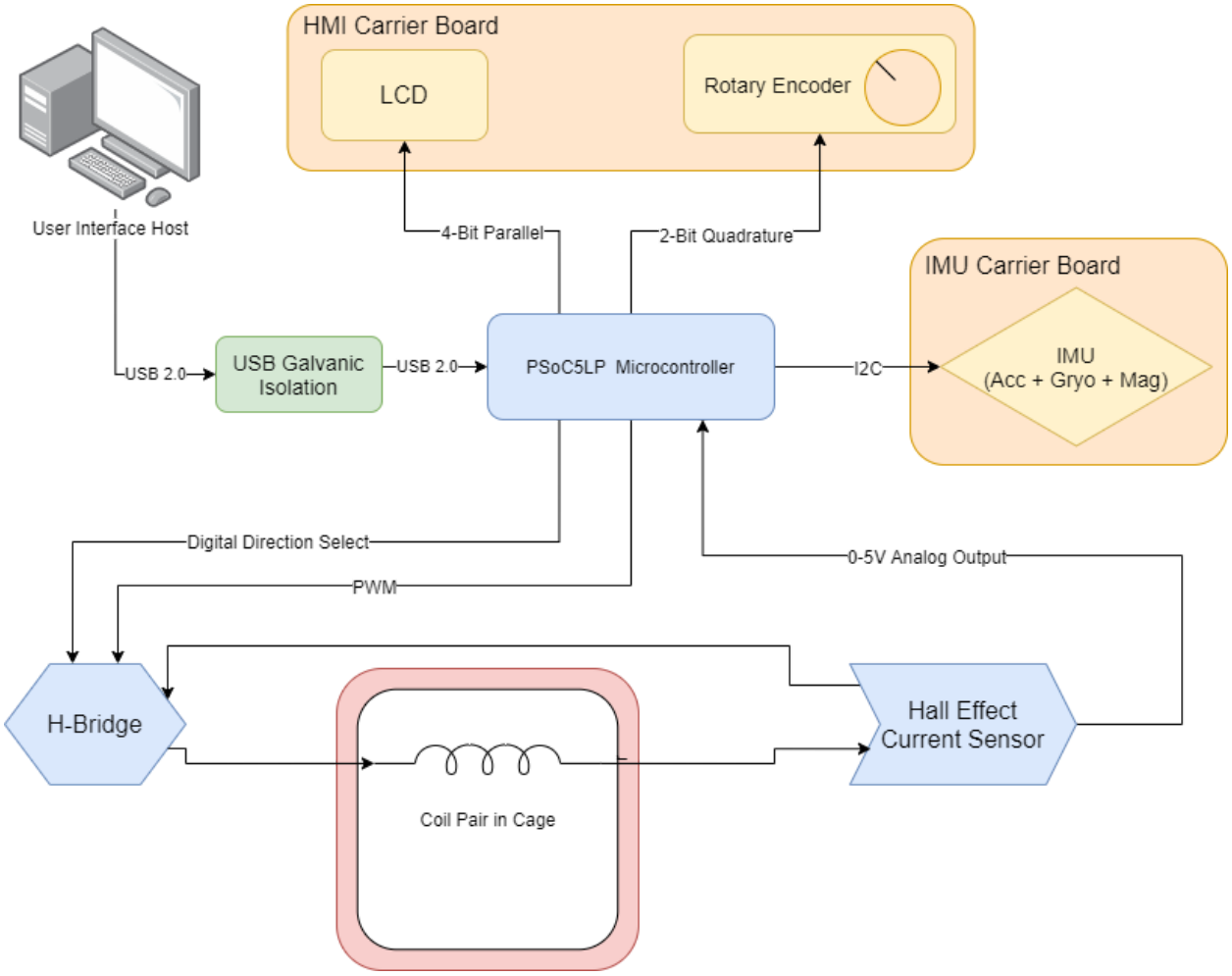


Figure HHC.4: Helmholtz System Block Diagram Illustrating a One of Three Axis Rev. 1 (12/1/19)

Coil Design

Relevant Design Requirements

In the design of electromagnet coils, the relevant requirements to this section are as follows. These requirements set the field that each pair needs to generate, and place limits on the maximum power draw and system voltage to generate said field.

- HHC_001
- HHC_004
- HHC_013

Theory and Relevant Equations

A Helmholtz coil is a single pair of coils spaced apart to create a highly-uniform magnetic field in the center. The two coils are wired in series and as such ensures that the current is exactly equal and can be proven by Kirchoff's Current Law. Magnetic field is generated by current flowing through the turns on each coil. The magnetic field generated for a given set of coil parameters is governed by equation M.1.

$$B = \frac{4N\mu_o I}{\pi a(1+\gamma^2)\sqrt{2+\gamma^2}}$$

Equation HHC.1: Equation for the Magnetic Field for a Square Helmholtz Coil

- N is the number of turns per coil in a pair
- B is the magnitude of the field to be generated at the center of the coil pair
- a is the side length for the square coil
- U_o is the magnetic permeability of the coil core
- I is the current flowing through the coil
- Gamma is the ratio of the coil spacing to the radius

Equation 1 can be rewritten as M.2 such that the design requirements can be used to calculate the number turns necessary.

$$N = \frac{B\pi a\left(1+\gamma^2\right)\sqrt{2+\gamma^2}}{4\mu_o I}$$

Equation HHC.2: Equation for the Number of Turns for a Square Helmholtz Coil

Due to the significant length of windings the coil will have a non-zero resistance. According to Ohm's law the coil will be current limited if the voltage is too low, thus the voltage supplying the coil must be great enough such that when the coil is directly connected to the power supply the maximum desired current is drawn through the coil pair. The resistance of the wound coil can be determined using the following relationship.

$$R = \frac{\rho L}{A}$$

Equation HHC.3: Resistance of a Wire

- R is the series resistance of a single coil

- Rho is the resistance of the wire material
- L is the length of the winding wire used
- A is the cross-sectional area of the winding wire

$$L = Na^2$$

Equation HHC.4: Length of Winding Given Number of Turns and Coil Side Length

Plugging equation M.3 and M.4 into Ohm's Law yields the relationship between the coil properties and the required supply voltage to run a coil pair see in M.5. The Numerator is doubled since the coils are connected in series, which doubles the resistance.

$$V_{supply} = I_{coil}\frac{2\rho N a^2}{A}$$

Equation HHC.5: Supply Voltage for a Coil Pair

Using these relationships a script was written to calculate the properties of a Helmholtz Coil Pair.

MATLAB Calculations & Theoretical Properties

Using the design equations outlined in the previous section the numerical values must be calculated based on the relevant design requirements. This was done with the MATLAB code shown in appendix C.1 & C.2.

The design script outputs decimal values of the ideal system however, in the real implementation nominal values are necessary for operation. Once a nominal system voltage and an integer number of winding turns are selected, the coil outputs and electrical properties must be recalculated. Given that a single supply rail will be used and each coil is slightly different, the maximum output, sensitivity, and scale factor will vary however, there is an additional 20uT of margin on the initial calculations to give headroom for the software running the coil to calibrate and adjust. In addition to software corrections, the mechanical design of the coil allows for adjustable spacing of the coil separation, so field corrections can also be adjusted physically after construction.Checking the sum of these properties the coils still have about 20uT over the minimum spec of 130uT specified in HHC_001.

The following parameters are identical for all 3 coil pairs.

- Vss: 36.00V
- N Turns per Coil: 29
- 20AWG Copper Wire

	X	Y	Z	Total
Average Side Length	1200.87mm	1170.77mm	1140.67mm	N/A
Magnetic Field	151.69 uT	157.134 uT	162.82 uT	N/A
Field Resolution	148 nT/LSB	153 nT/LSB	159 nT/LSB	N/A
Voltage Ripple Sensitivity	4.213 uT/V	4.364 uT/V	4.523 uT/V	N/A
Length of Wire per Axis	278.60m	271.62m	264.63m	814.85m
Resistance Per Axis	9.258 Ω	9.026 Ω	8.794 Ω	N/A
Current Per Axis	3.89 A	3.99 A	4.09 A	11.97 A
Power Per Axis	139.98 W	143.58 W	147.37 W	430.93 W
Inductance Per Axis	11.664 mH	11.330 mH	11.000 mH	N/A

Table HHC.2: Table of Theoretical Coil Properties given Nominal Parameter Selection

Coil Cage Mechanical Design

Traditionally, Helmholtz Coils are constructed using circular coil pairs, however construction of circular rings at the size necessary (approx 1.2m) is difficult. PolySat utilized square coils and proved that this method worked.

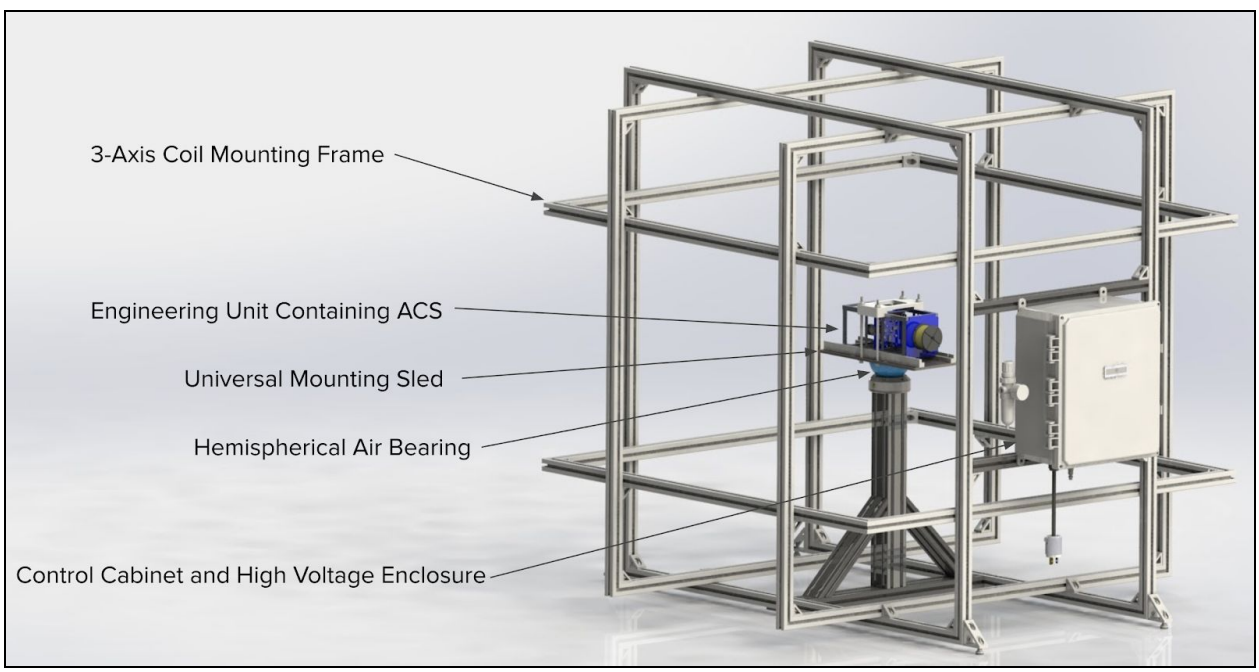


Figure HHC.5: Helmholtz Mechanical Design Overview Render (2/12/2020)

The Helmholtz coil must physically hold the 3 pairs of nested coils, support electronics, and support the satellite on a riser in the center region where the field is generated. In Figure 6 a full render of the proposed system can be viewed. The key design requirement that applies to the frame design is HHC_007 which restricts one of the dimensions to a max width of 24in so that the frame can fit through a single width doorway.

Construction Method

To form the frame, we choose an aluminum T-Slot construction. This allowed for rapid assembly and disassembly of the system for transportation. Aluminum extrusions are used commonly for fixtures in industry. By using this construction, we were able to design the system such that the inner YZ coils could be unbolted leaving just the core pictured in figure 8 which is 23.97” in width.

CAD Modeling

The nested coil cage was designed using the mechanical CAD software, Solidworks.

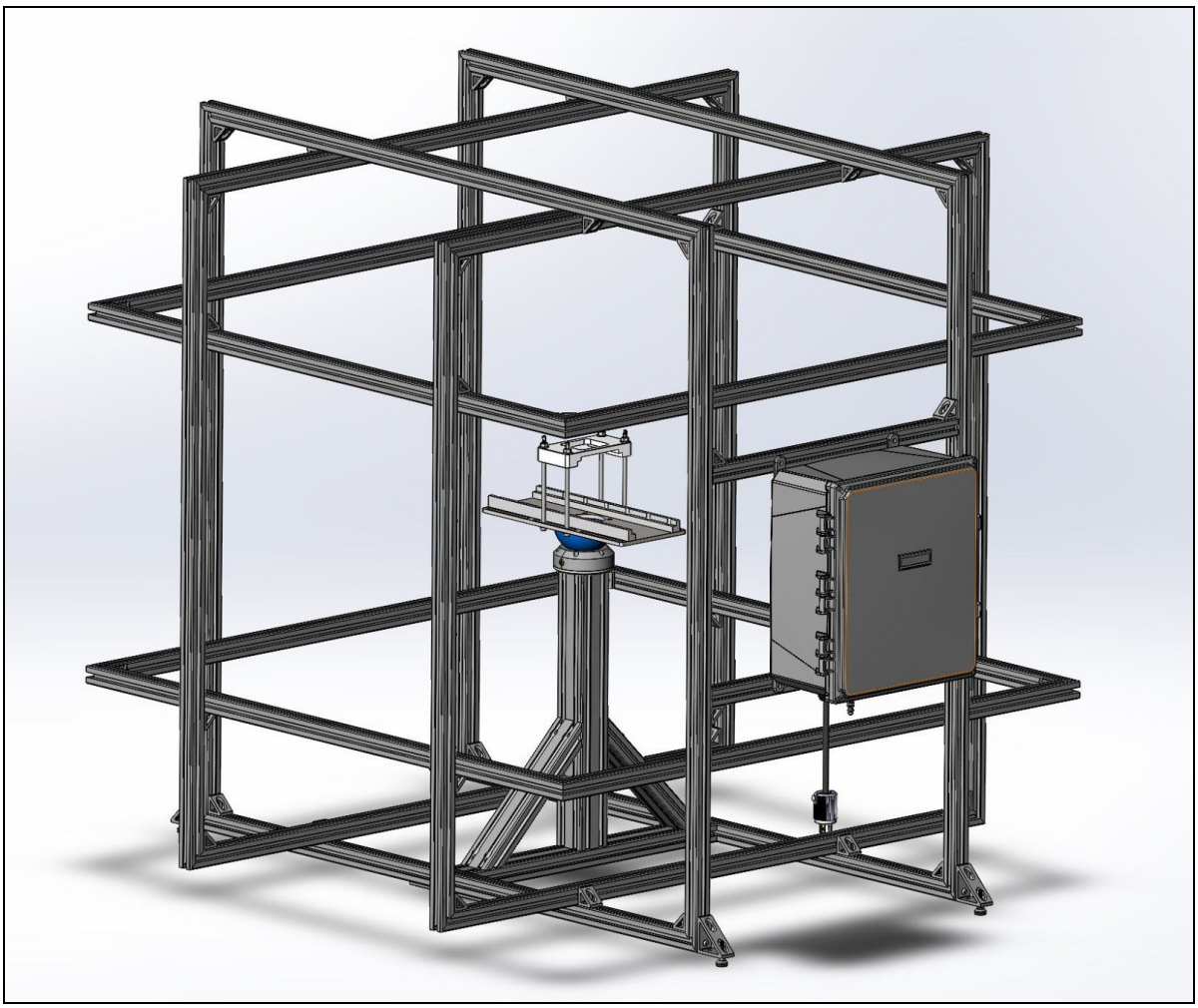


Figure HHC.6: Assembled Bare Frame Mockup

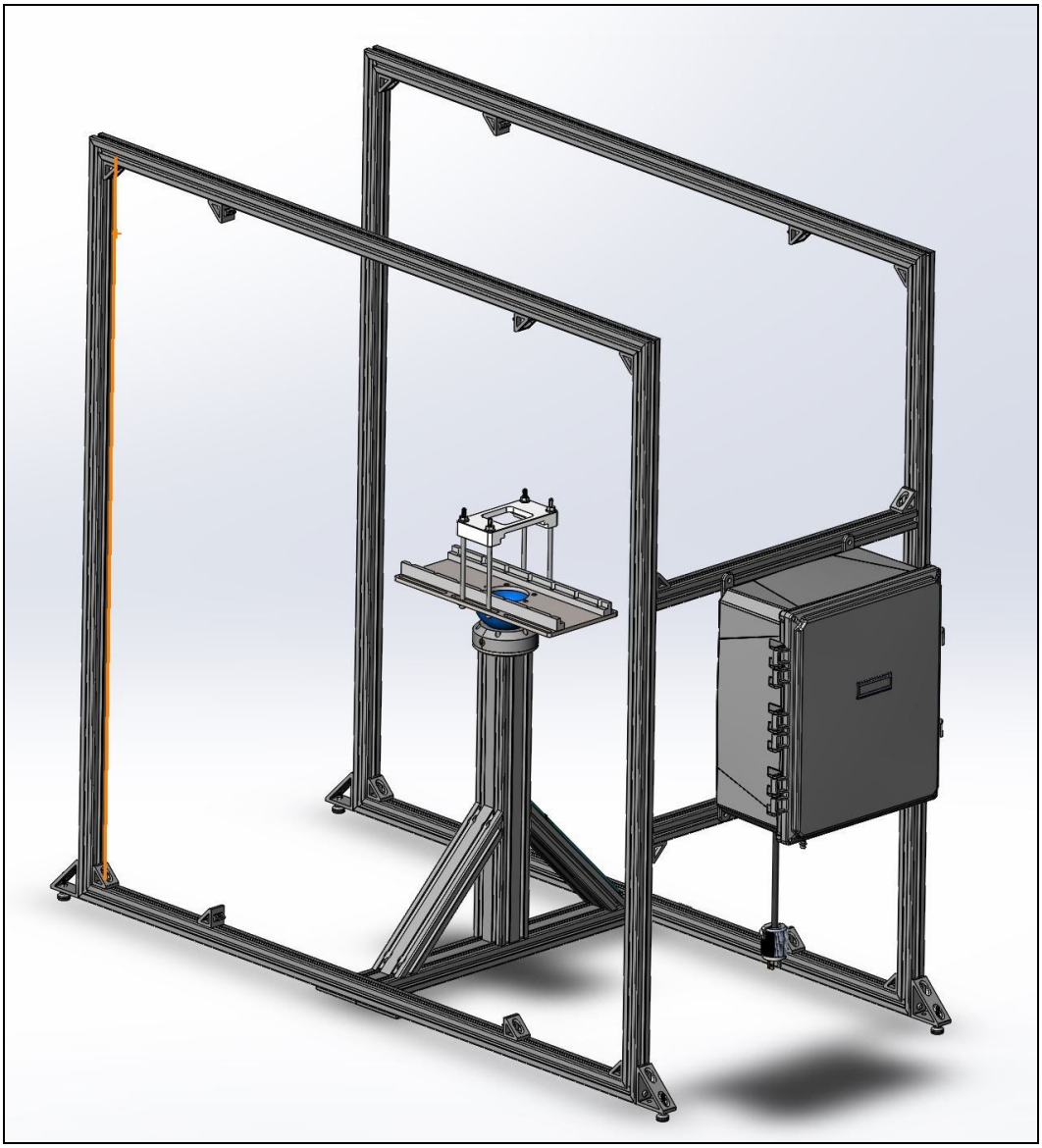


Figure HHC.7: CAD Render of HHC Core Ready for Transport, Narrowest Width is 24”

Mechanical Drawings & Bill of Materials

All mechanical drawings and the bill of materials are included in the appendix.

Fabrication

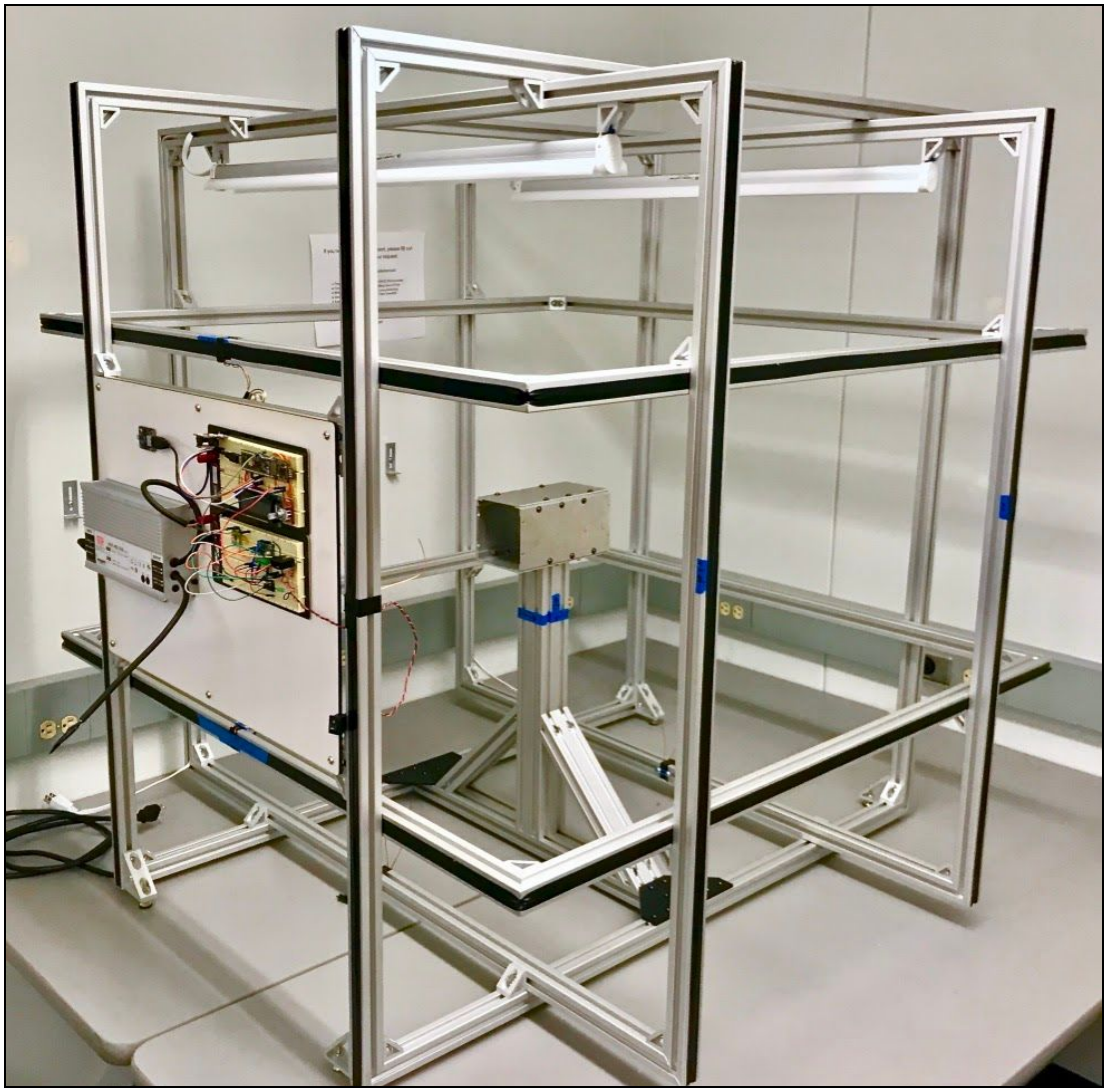


Figure HHC.8: Photo of Progress on HHC, Coils Complete with Electronics Mockup (3/11/2020)

The construction of the HHC is not complete at the time of this writing and this section will be updated when the air bearing system and electronics cabinet is finished in spring.

Low Friction Satellite Mount

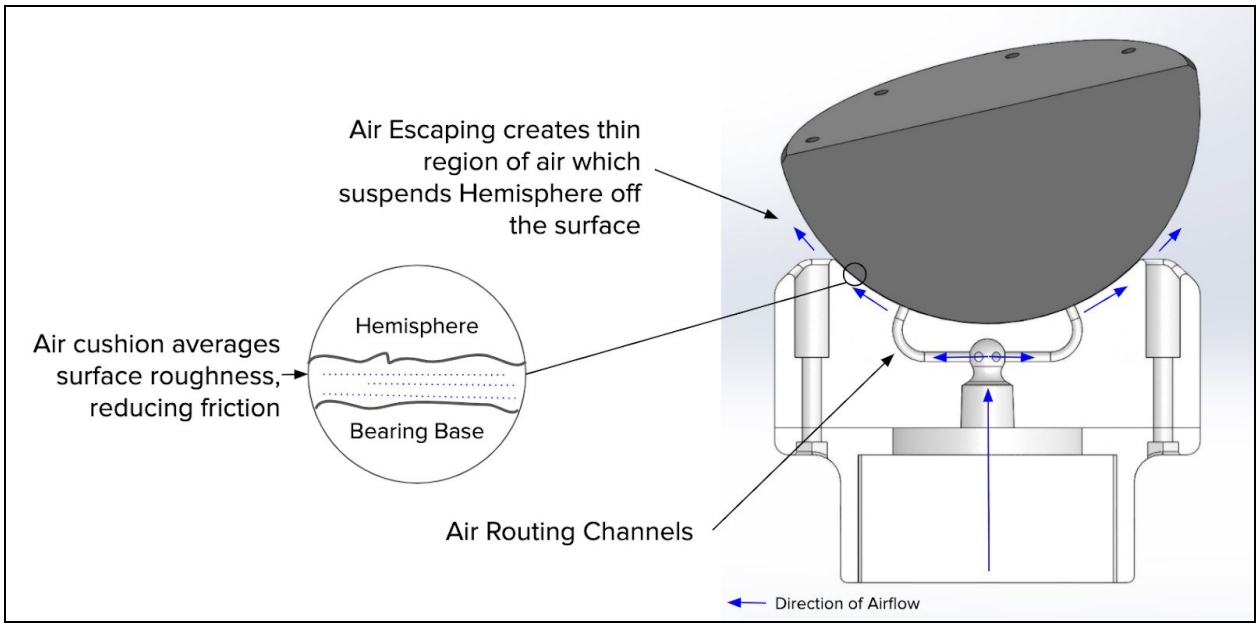


Figure HHC.9: Diagram of Airflow and Air Bearing Principle of Operation (5/3/2020)

Air Bearing Natural Stability

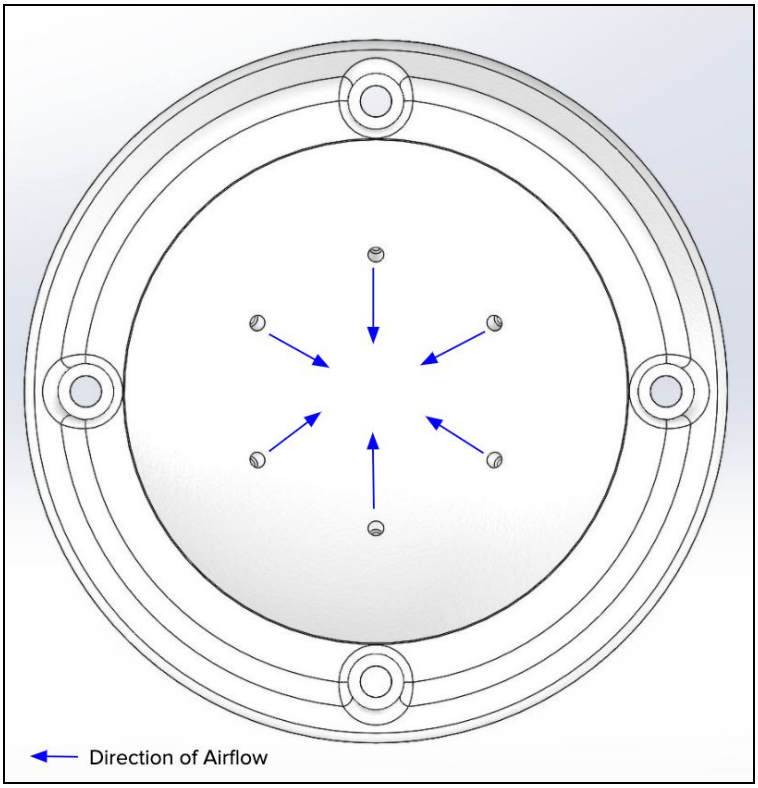


Figure HHC.10: Diagram of Airflow and Air Bearing Principle of Operation (5/3/2020)

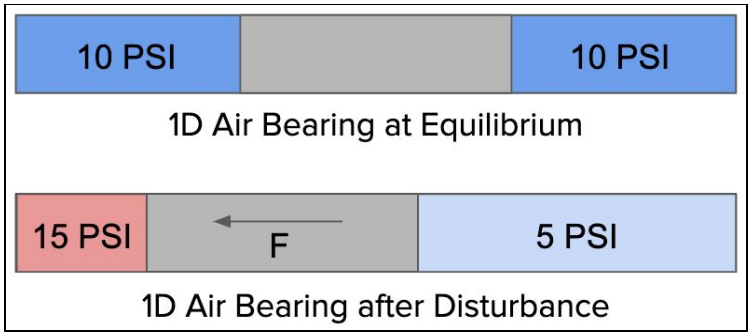


Figure HHC.11: 1-Dimensional Stability Example (5/3/2020)

Integration with HHC

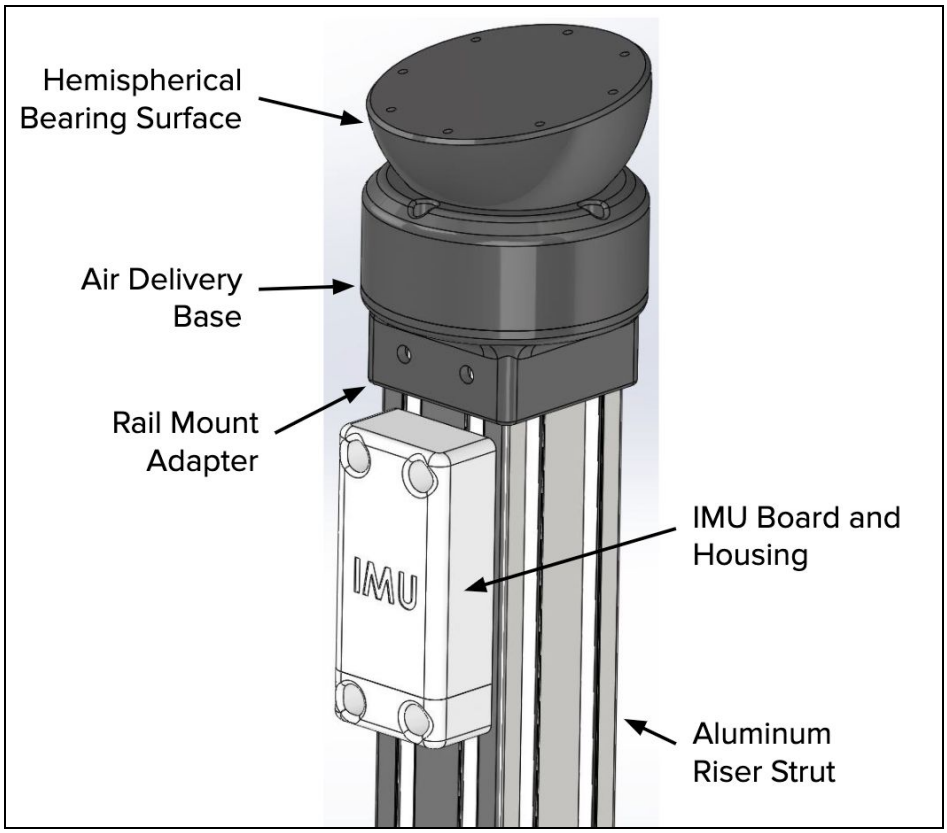


Figure HHC.12: Air Bearing Mounting on Satellite Riser in HHC (5/3/2020)

Measuring Coil Properties

From the simulated Matlab values for the coil properties, a calculated hypothesis was set for the measured actual coil characteristics. The main parameters of concern were inductance and resistance. The inductance corresponded to any antenna characteristics that the coil pairs created as well as any resonation that might occur. While the resistance is responsible for the voltage and current amounts that are needed to create the wanted field.

The first apparatus that was used for measuring the coil characteristics was the LCR meter. This is a testing equipment used to measure the inductance, capacitance, and resistance of an electronic component. The Agilent 4263B was used as it allows you to simulate testing under the correct conditions: 100 Hz, 120 Hz, 1 kHz, 10 kHz, and 100 kHz as well as adjusting the voltage levels.

Figure HHC.13: Agilent 4263B, Frequency = 100KHz, LVL = 1000mV

Tests were run for a single axis under a constant voltage level of 1000mV, while varying the frequencies that the apparatus offers. Multiple values from each frequency were collected and an average was taken. Anomalies were not taken into account as there was some fluctuation in the results seen. From the averages a mapping of the values was created of resistance and inductance vs frequency to check for linearity and to compare against our hypothesis.

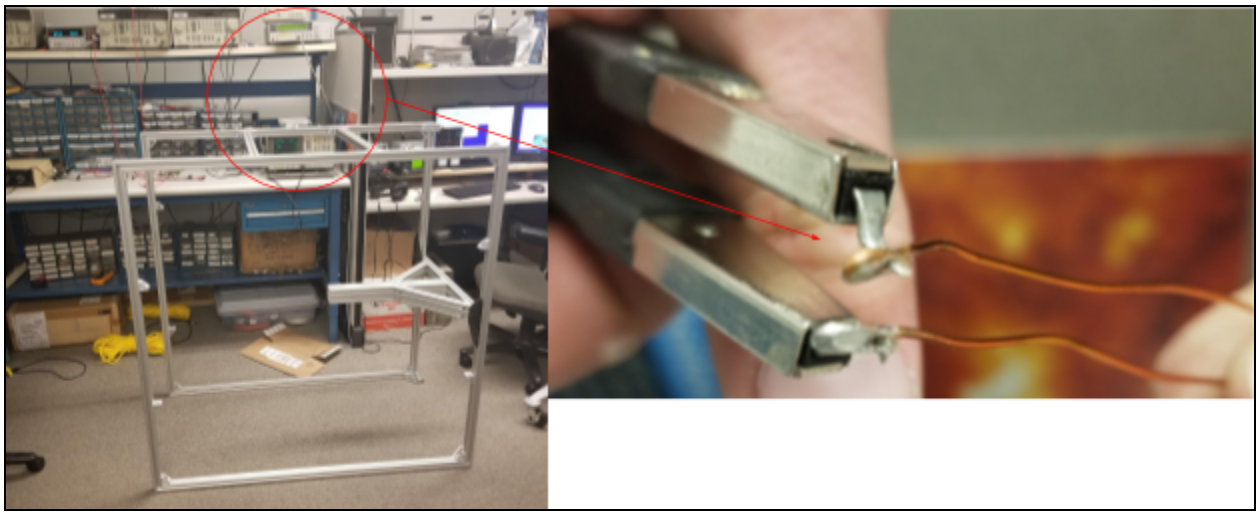


Figure HHC.14: LCR Meter Test Setup

Frequency (Hz)	Resistance (mΩ)	Inductance (uH)
100	390.61	96.30
120	398.05	96.50
1000	572.94	21.55
10000	644.98	7.68
100000	929.00	6.69

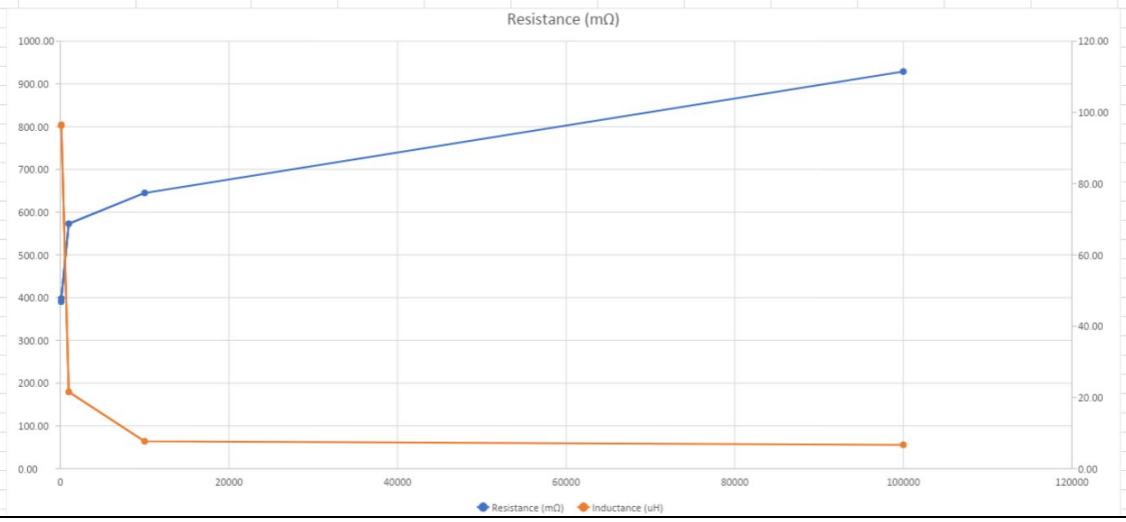


Figure HHC.15: Coil Resistance and Inductance vs Frequency

As seen above the results for our inductance and resistance were non-linear and did not match the original hypothesis. The values had an order of magnitude of 100x less than what was calculated, thus we concluded that this method of characterization was not valid. Given that the coils would act as a magnetic loop antenna and had very high inductance, the LCR meter proved to be inefficient.

The next method for characterization that was decided, was to use a VNA. VNA commonly measures s–parameters because reflection and transmission of electrical networks are easy to measure at high frequencies. This made it difficult for the functionalities of the coils, as the frequencies used for the PWM input are supposed to be as low 5Khz.

The DG8SAQ VNA was used as it operates under a frequency coverage of 1Khz to 1.3Ghz. An SMA connector, a crew terminal and an RG58U coaxial cable were soldered together and used as a way to connect the coils to the TX out of the VNA. The added parameter of the Coax was not taken into account, as they are negligible at such low frequencies. See figure 4 for test setup.

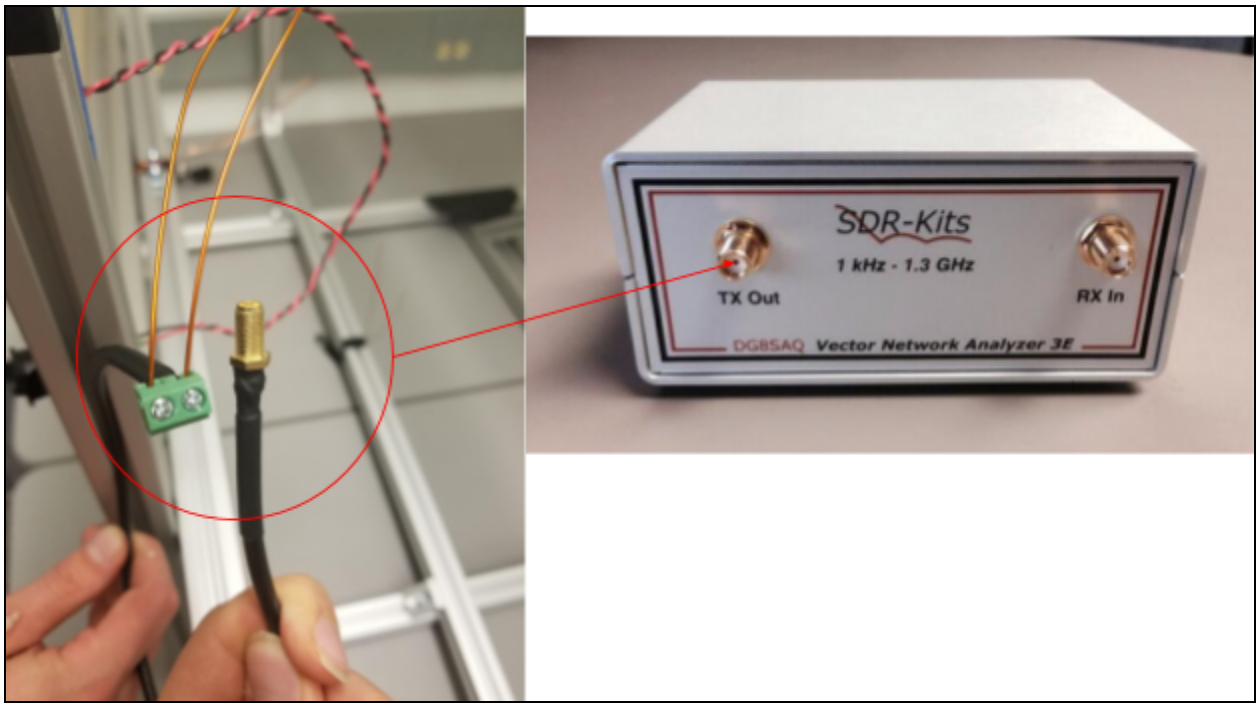


Figure HHC.16: VNA Test Setup for Coil Properties

After the calibration of the VNA, different frequencies, and frequency range sweeps were experimented with to find a stable value. The Smith Chart showed values that were constantly fluctuating and did not resemble the calculated values as seen in figure 5. These values were not usable nor applicable for the characterization of the coils. Two factors that could have caused these results would be the frequency range being too low and the audio calibration. The DG8SAQ uses the audio card as its reference for wave calibration, but it did seem compatible with the audio cards we had available. As seen in figure 6 the waves set from the calibration were distorted and not sinusoidal as they should have been.

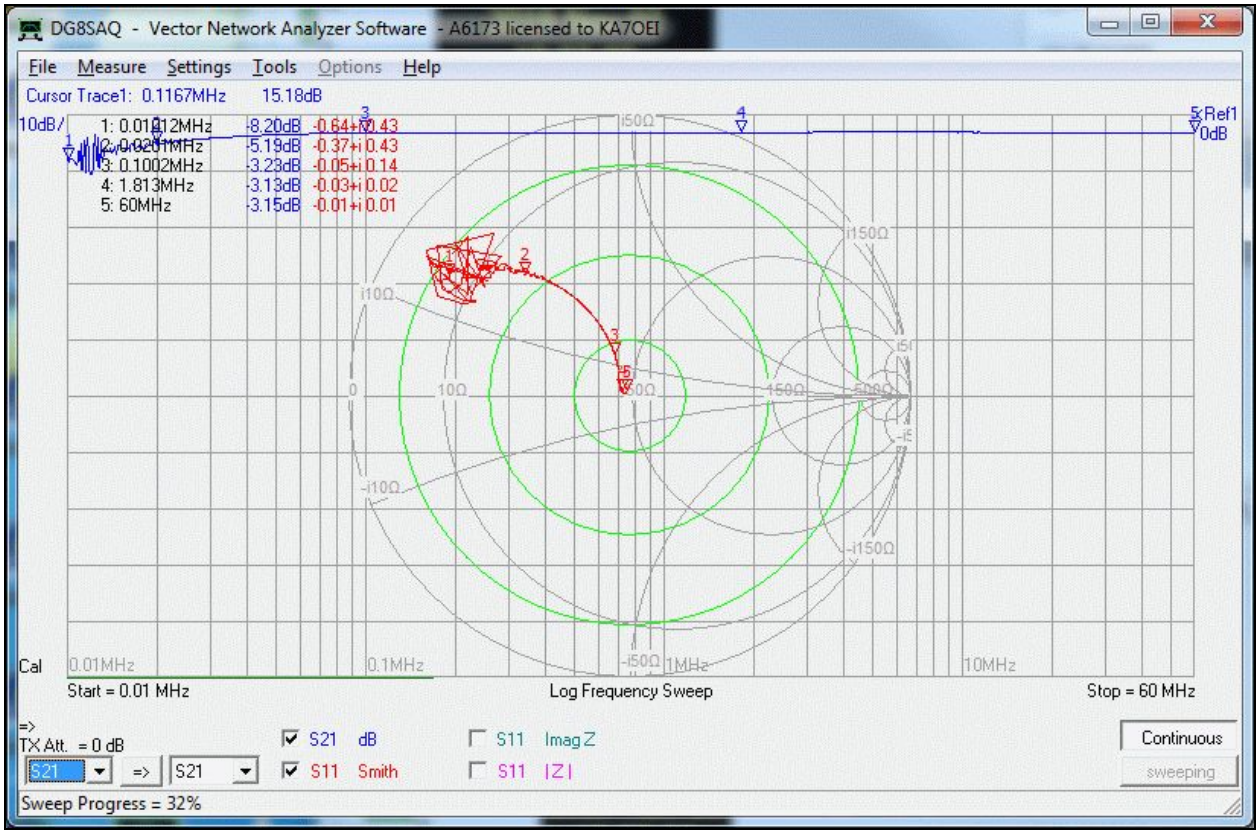


Figure HHC.17: Smith Chart Results from Coil VNA Test

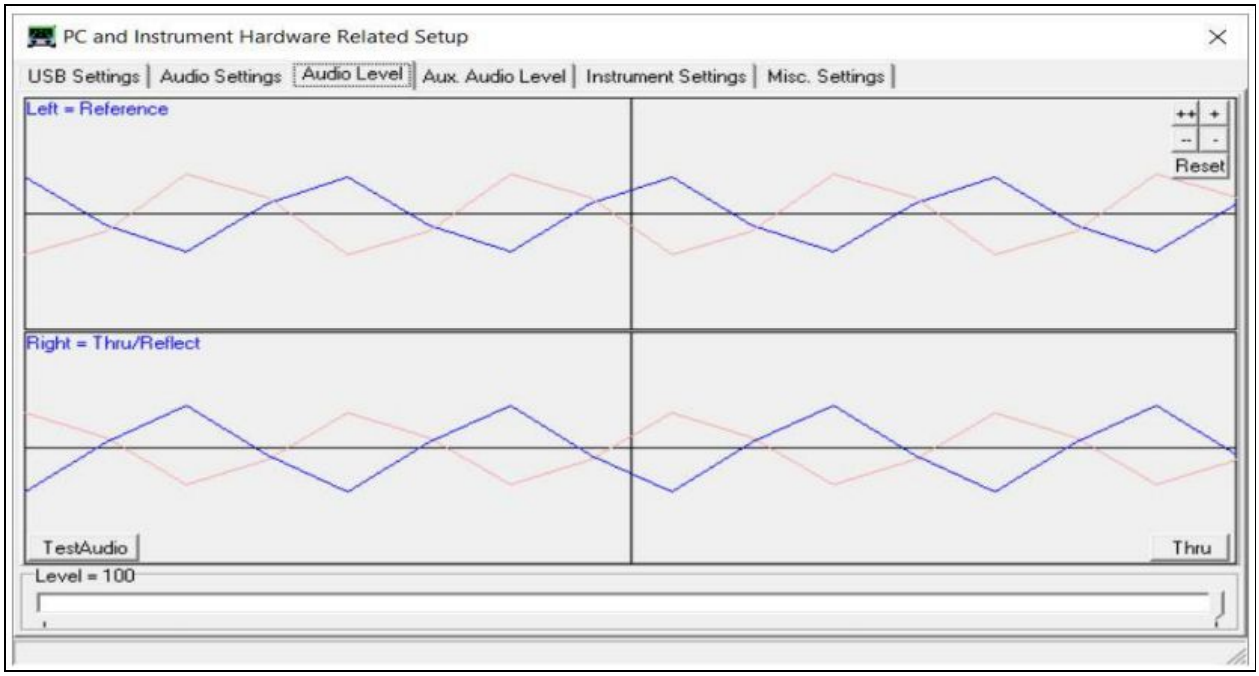


Figure HHC.18: VNA Audio Calibration Results

Finally, a utilization of the resonant characteristics of the coil inductance was used. An LC tank circuit with a precision measured capacitance, see the circuit in figure 16. An LC tank is a resonant circuit, and by exploiting this property we can perform a frequency sweep and find the frequency which results in the greatest amplitude on the output. By using a known capacitance and the following relationship, we can experimentally find the inductance of the coil.

$$f_c = \frac{1}{2\pi\sqrt{LC}}$$

Equation HHC.6: Resonant Frequency Relationship

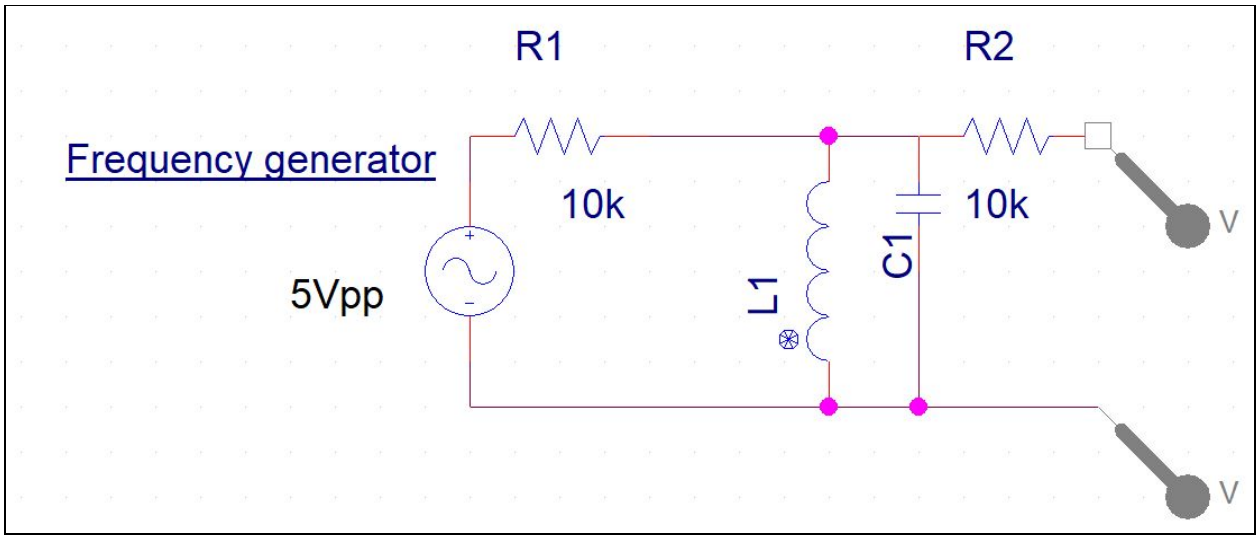


Figure HHC.19: LC Tank Circuit Used to Measure Inductance via Resonance

Initially we simulated the LC Tanks frequency response to get an idea of how the bode plot is going to look, and to get a rough estimate of what to expect, the results are shown in the figure below.

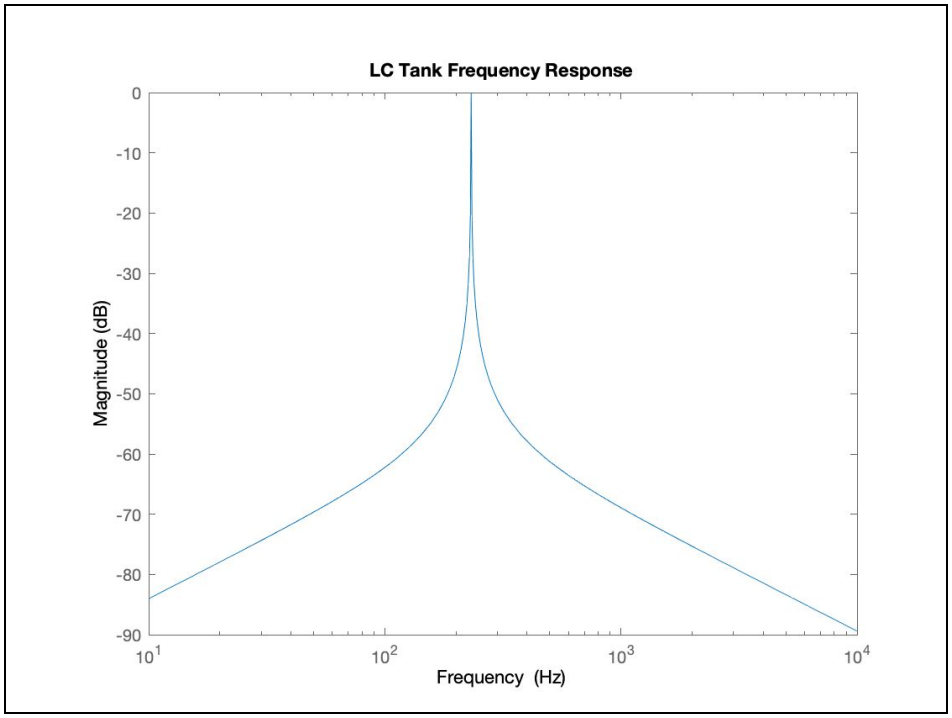


Figure HHC.20: Theoretical LC Tank Frequency Response of Circuit in Fig. 3 w/ L coil of 10mH (Rev. 2 - 3/8/20)

An experiment using multiple known capacitors was conducted, and average of the resonant frequencies per capacitor were taken. Using the frequency to inductance relation shown above a resultant inductance was calculated per capacitor. An average of the inductance was then calculated and used as the characterized value and the results are shown below in table 1.

Coil Series Inductance			
	X-Axis	Y-Axis	Z-Axis
Theoretical	11.664 mH	11.330 mH	11.000 mH
Measured	10.96 mH	10.650 mH	10.341 mH

Table HHC.3: Coil Inductance Theoretical vs Experimental

The resistances were measured using a multimeter for each of the individual coils. Adding the resistance of the coils yields the series resistance of each axis of the coil cage. See figure 9 for test setup and table 2 for test results. The inductance and resistance values measured from these two methods, were the ones used for later calculation, as they are much closer to the theoretical values seen above in previous sections of the chapter.

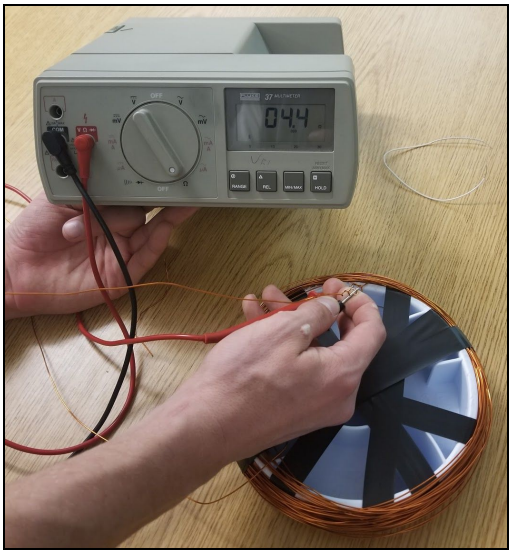


Figure HHC.21: Multimeter Coil Resistance Test Setup

Coil Series Resistance			
	X-Axis	Y-Axis	Z-Axis
Theoretical	9.258 Ω	9.026 Ω	8.794 Ω
Measured	9.0 Ω	8.7 Ω	8.4 Ω

Table HHC.4: Coil Series Resistance Theoretical vs Experimental

Power Supply Selection

The supply chosen should be able to satisfy the following conditions:

- 36VDC Nominal Output
- > 11.06A Supply
- > 398 Watts

The following power supplies were chosen as candidates:

MEANWELL SE-600-48/36
MEANWELL SP-750-48/36
MEANWELL HEP-480-36A

The HEP-480-36A power supply was chosen because it eliminates the need for a high-voltage cabinet and additional fusing as the power supply is fully encased and protected. This improved the overall safety of the system and reduced cost since enclosures were close to $200 each and would require additional fittings.

Control Electronics

In order to power the system and achieve the desired functionality, we engineered an electric design to meet the design specifications. The design consisted of the control electronics box which houses the control modules. These consist of: the H-bridges, hall effect sensors, microcontroller, HMI, and motherboard. Furthermore, the carrier board houses the magnetometer and its electronics which are planted in the center of the cage.

Electrical Design Goals

The HHC requires the ability to drive current through the coils in both directions and modulate the applied voltage for precise control. We utilized H-brides and PWM signals to achieve this feature. We can find the magnetic field strength of a coil set using the equation, stated in equation HHC.1. We set the area of the coil as our control variable, as this is dependent on our size requirement. That means that our generated field is dependent on the number of turns and current input. We found an exponential increase in copper wire reel prices depending on gauge and size, therefore we had a set maximum wire length and we controlled our current. The copper reel we sourced was a 20 gauge, 3125ft wire. Using these variables we were able to deduce that a maximum current of 4A with a voltage of 36V which would be sufficient for the design requirements as seen in table HHC.2.

H-Bridge

We needed to source an H-bridge that is able to provide a constant 4A current and drive the coil pairs at the required voltage. This proved to be tough given that most H-bridges that can provide high power is usually very high voltage, or too expensive for our budget. We were able to find an H-bridge that can operate in either dual-channel mode for independent bidirectional control of two modules or single-channel mode for driving a single module with increased current. This was the Toshiba TB67H420FTG. This H-bridge is able to drive up to 4.2A of constant current at a maximum of 50V [A.3.1].

Our current delivery system had to account for the current ripple outlined by requirement HHC_003. Using PSPICE we simulated multiple electrical schematics primarily concentrating on capacitors and inductors, which helped to help us reach our requirement [A.3.2]. We tested the module using a square wave with a $36V_{PP}$, 5.8KHz with a 50% Duty Cycle. We experimented with different circuits and all the schematics are shown in the appendix. The testing conditions were kept the same for a consistent environment and to have fair, reliable tests. Finally, we accumulated the results from all the schematics, and they are portrayed below.

Coil Simulation Ripple Results		
Simulation	Current Ripple (mA_{PP})	Magnetic Field Ripple (nT_{PP})
Constant Conduction Mode	130	5008
Series LCR	130	5008
Parallel LCR	600	23,808
Series LR - Rev 1.0	13.5	520
Series LR - Rev 2.0	10	385

Table HHC.5: Coil Frequency Response Simulation Result Comparison

Through simulation we discovered that not adding any discrete components gave us our desired ripple. This meant that we only relied on the inherent low pass filter that the coil inductance created. Our successful model was the Series LR circuit which consisted of no added components. The final decision was to go with the Series LR circuit rev 1.0. Even Though Series LR rev 2.0 had a lower ripple, it required the use of extra inductance which would require filtering to block any unwanted magnetic fields. The following figures represent the desired circuits and results.

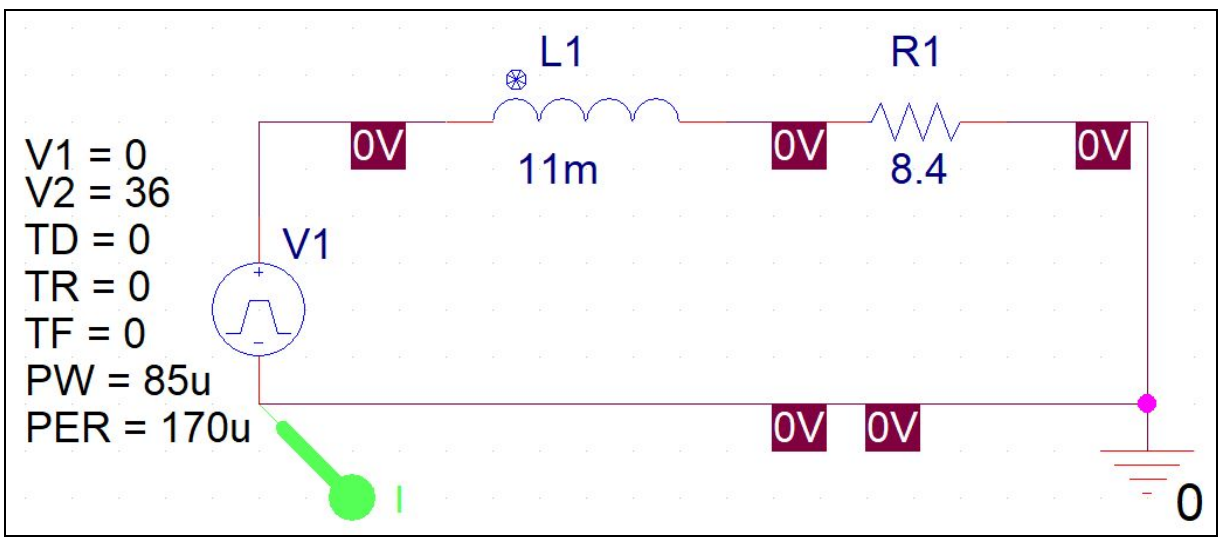


Figure HHC.22: Schematic Coil Simulation-Series LR (Rev 1.0 3/10/20)

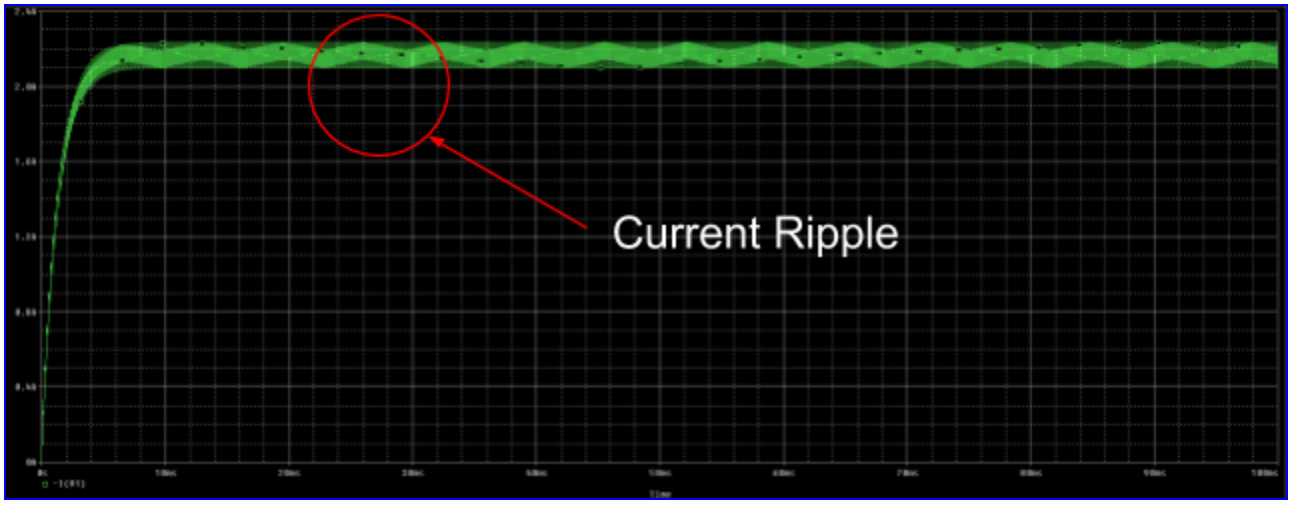


Figure HHC.23: Current through Inductor L1 for Series LR (Rev 1.0 3/10/20)

Current Sensors

For our closed loop feedback system we required current sensors to read the actual PWM current flowing into the coils. The current sensors read the change in current out of the H-bridges and adjust the inputs accordingly. This grants us more control over the system, and accounts for errors coming out of the drivers which would help us achieve requirement HHC_002. This would also inform us about the actual generated magnetic field in relation to the current being fed for more accurate testing results of the ACS.

The current sensors that we sourced were the Allegro's ±31 A ACS711 Hall effect-based linear current sensor [A.3.3]. It offers a low-resistance current path and electrical isolation up to 100 V. This version accepts a bidirectional current input with a magnitude up to 31 A and outputs a proportional analog voltage centered at Vcc/2 with a typical error of ±4%. This component was sourced as it hit out current, voltage and sensitivity specifications. It also came on a 2-oz copper dev board which would decrease in system power losses, and increase in reading accuracy. Finally, it is a Hall effect sensor, which means the IC is able to electrically isolate the current path from the sensor's electronics, which companies with our main safety specifications for voltage isolation.

In order to determine the linearity, accuracy and functionality of the device, we tested the current sensors alongside the microcontroller we were planning on using, the PSOC5. We tested the sensors using the desired 5V rail for the logic, and a Delta Sigma ADC inside the PSOC5. We varied the current load starting from 0.0 A to 5A using the bench power supply. Using a 12 bit PWM for the output, we were able to record our readings at 100mA increments.

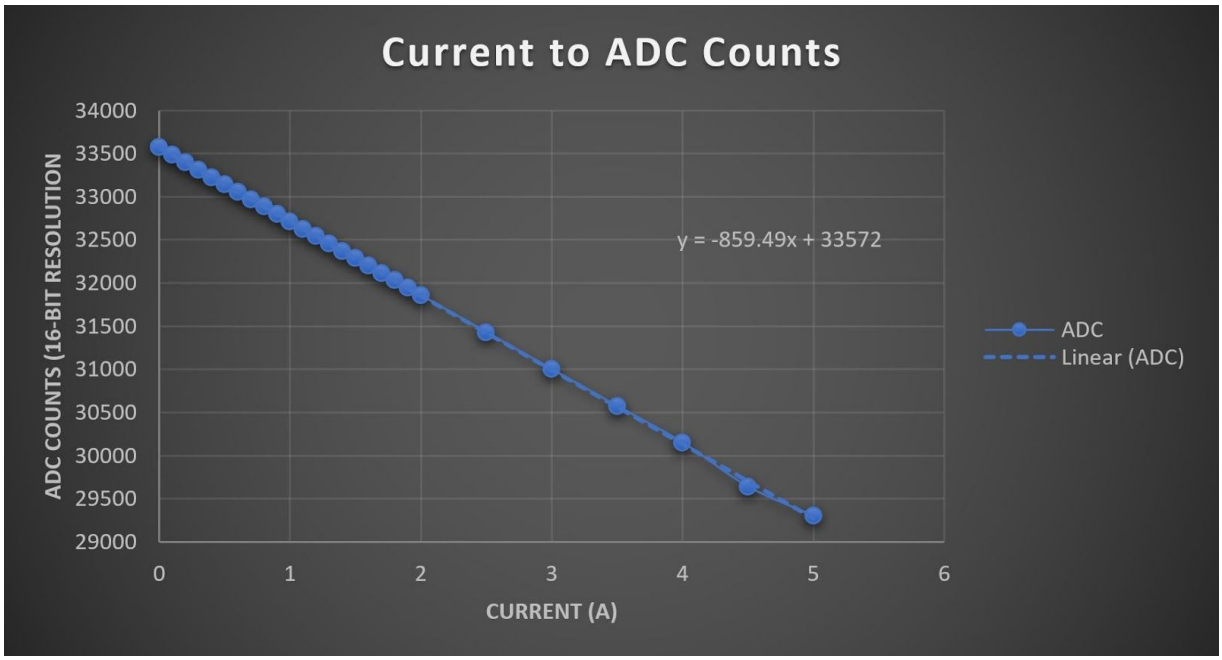


Figure HHC.24: Test Results for Current vs ADC Reading of ACS711EX Current Sensor (2/18/2020)

We were able to confirm that the interface between PSOC, current sensor and HMI was successful, and proved the linearity of the current sensors. We found that we get 862 raw ADC value counts/A, which converts to 1.16mA/LSB. This is equivalent to 43nT/count which meets our requirement for being under 50nT/count, thus confirming the sensors functionality.

Motherboard

In order to connect all electronic modules together, a motherboard was required. The motherboard consisted: the 3 H-bridges, 3 current sensors, and the microcontroller.

The H-bridge we sourced required both forward and reverse direction inputs to be inserted into the driver to work depending on the desired driving direction. A hexinverter was used in order to invert the signal sent from the microcontroller and both signals were fed into their respective inputs on the driver.

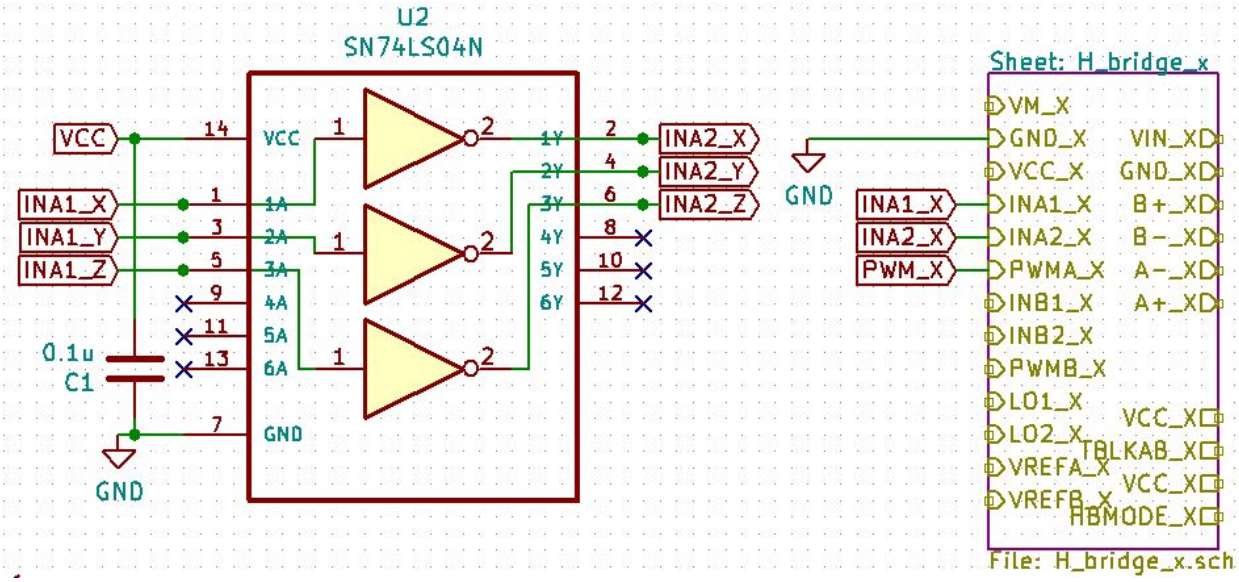


Figure HHC.25: X-axis H-bridge Input Signals (Direction & PWM) (3/2/2020)

We had to establish a fusing system to comply with the HHC_014 requirement. We had two sets of fast acting fuses: three 5A fuses, for the H-bridge inputs, and a 15A fuse for the power supply. They were all linked to LEDs, where the LEDs would light up if the fuse breaks as the impedance across the fuse would greatly increase.

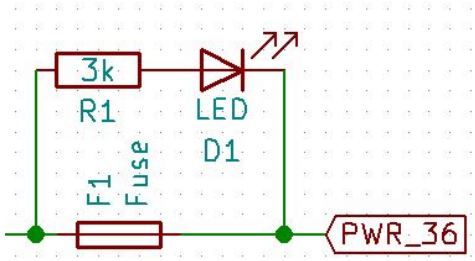


Figure HHC.26: Fuse Indication Circuit across H-bridge Voltage Input (3/2/2020)

After confirming that a series LR circuit (as seen in figure HHC.19) was the configuration we were going to use, we now had to connect our H-bridges to our current sensors. Given there were no external circuitry then the only component needed was L1 in figure HHC.24, where L1 in the X-axis coil pair.

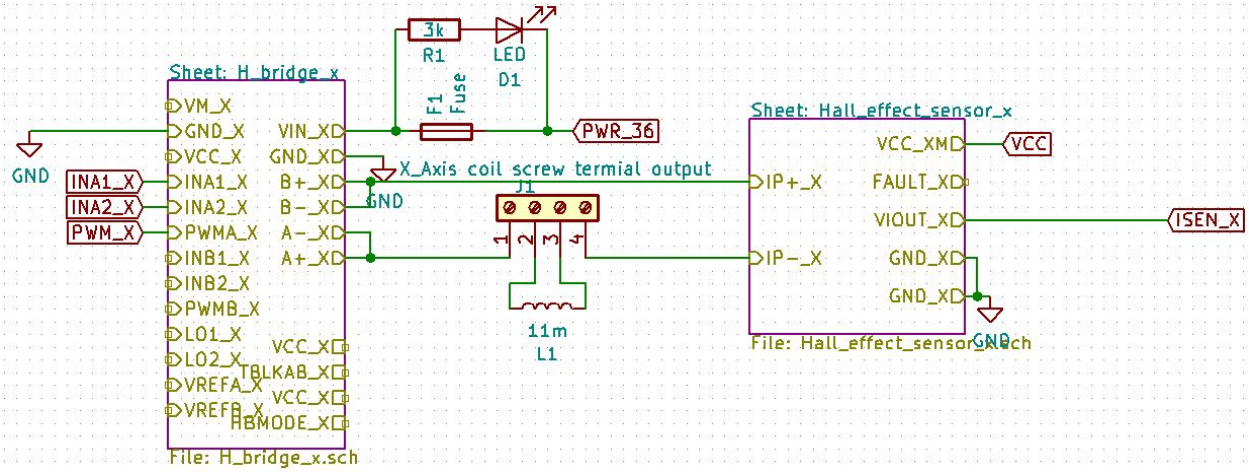


Figure HHC.27: X-axis H-bridge to X-axis Current Sensor Wiring (3/2/2020)

We had to cover two components in the power input, the E-stop module which was connected directly to the power supply. And the 5V switching voltage regulator, which provided the voltage rail for all the modules on the motherboard, this was linked to the 15A mentioned in the section above.

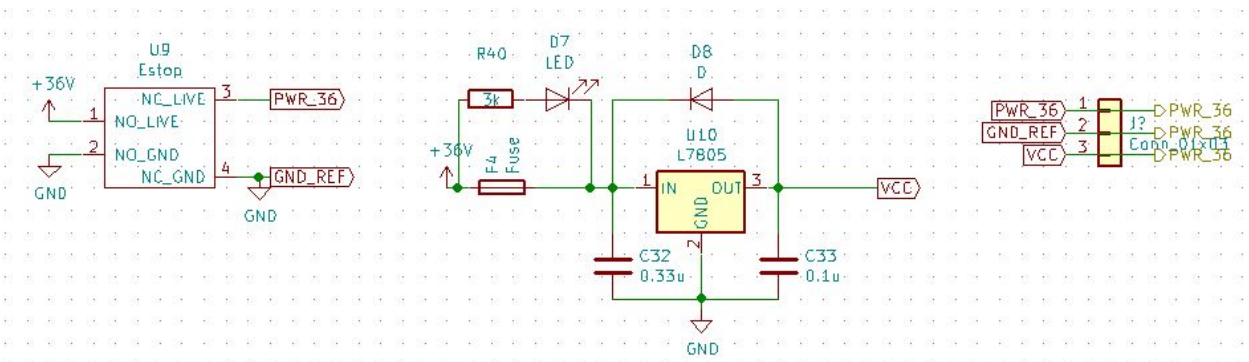


Figure HHC.28: Power Regulation for Logic Voltages (3/2/2020)

Finally, we had two female buses: one connected to the IMU module (our magnetometer), and one the other to the HMI module. These buses carried both the power and signal inputs for the respective modules to be able to drive them. For the completed schematic, reference [A.3.4].

PCB Layout

For the PCB layout our biggest concern was the trace widths of the power traces. This was to secure the systems power delivery, and reduce thermal noise and parasitics. Given that we were driving high power, we had to be conscious of the amount of current these traces could handle. Using the following equation we were able to calculate the minimum require trace widths:

$$I = K * dT^{0.44} * (W*H)^{0.725}$$

Equation HHC.6: trace width to current input and temperature rise

Where I is the maximum current in amps
dT is the temperature rise above ambient in degrees C
W,H are the widths and thickness in mils
K is equal to 0.024 for internal traces, or 0.048 for externa traces

From equation HHC.6 we were able to confirm our minimum required trace widths. We decided on using 80mil traces for high power, 40mil for low power and 20mil traces for routing. This was around double the minimum requirement we calculated, and given that envelope was not a constraint we tried to minimize parasitics as much as possible. For the main power line coming from the power supply, we kept it as a cooper fill to decrease its inductance and have a bigger thermal spread across the board. We also added it on the top side of the board rather than the bottom to minimize noise interference.

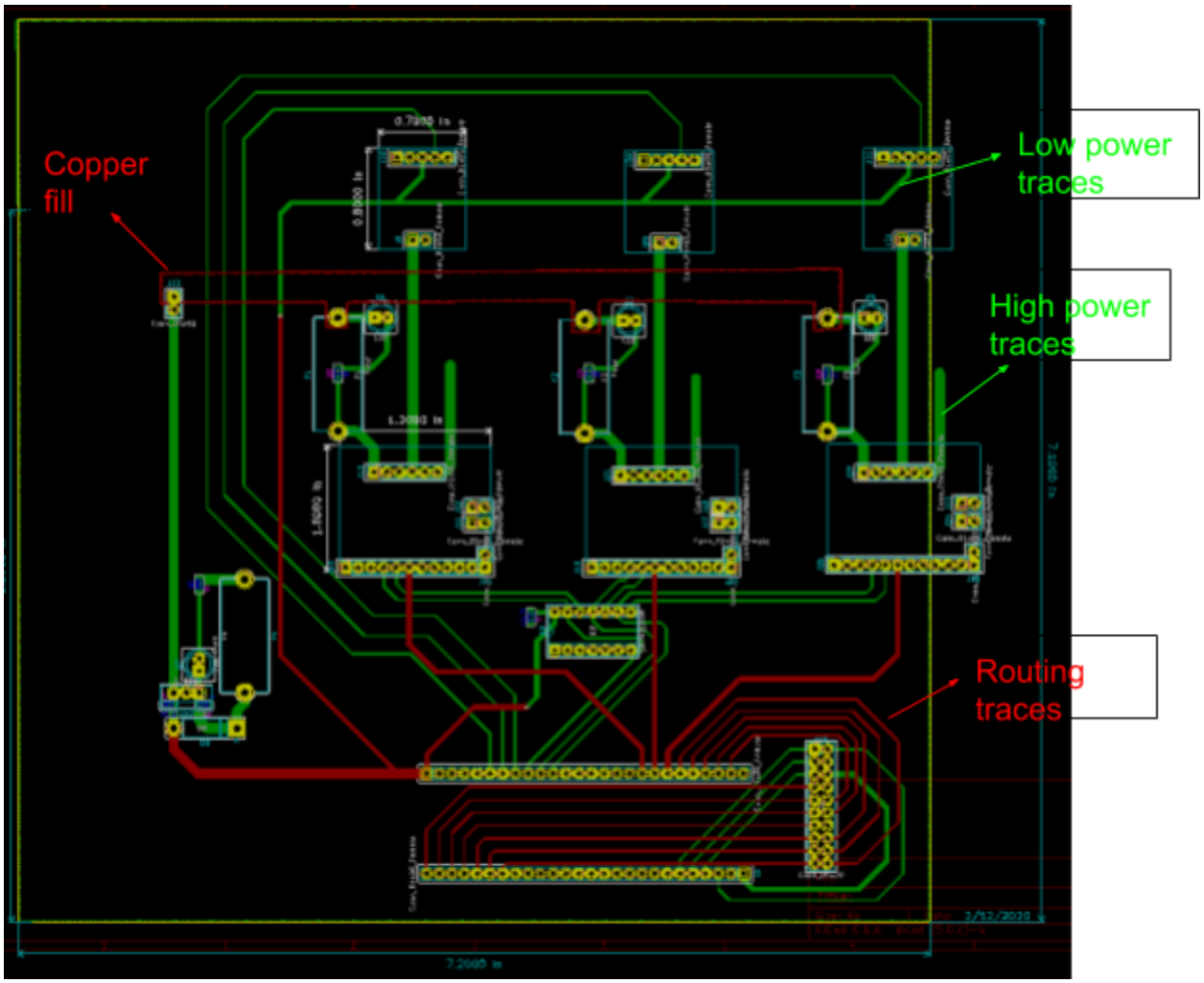


Figure HHC.29: Motherboard PCB Layout V0.6 (3/25/2020)

The motherboard ended up being a 7.2inX7.2in board. We tried to modularize all our components in case they need to be switched out. That meant that they were all connected to the motherboard through header pins. We laid out the modules so that they are as close as possible while having sufficient spacing between routing traces to reduce noise.

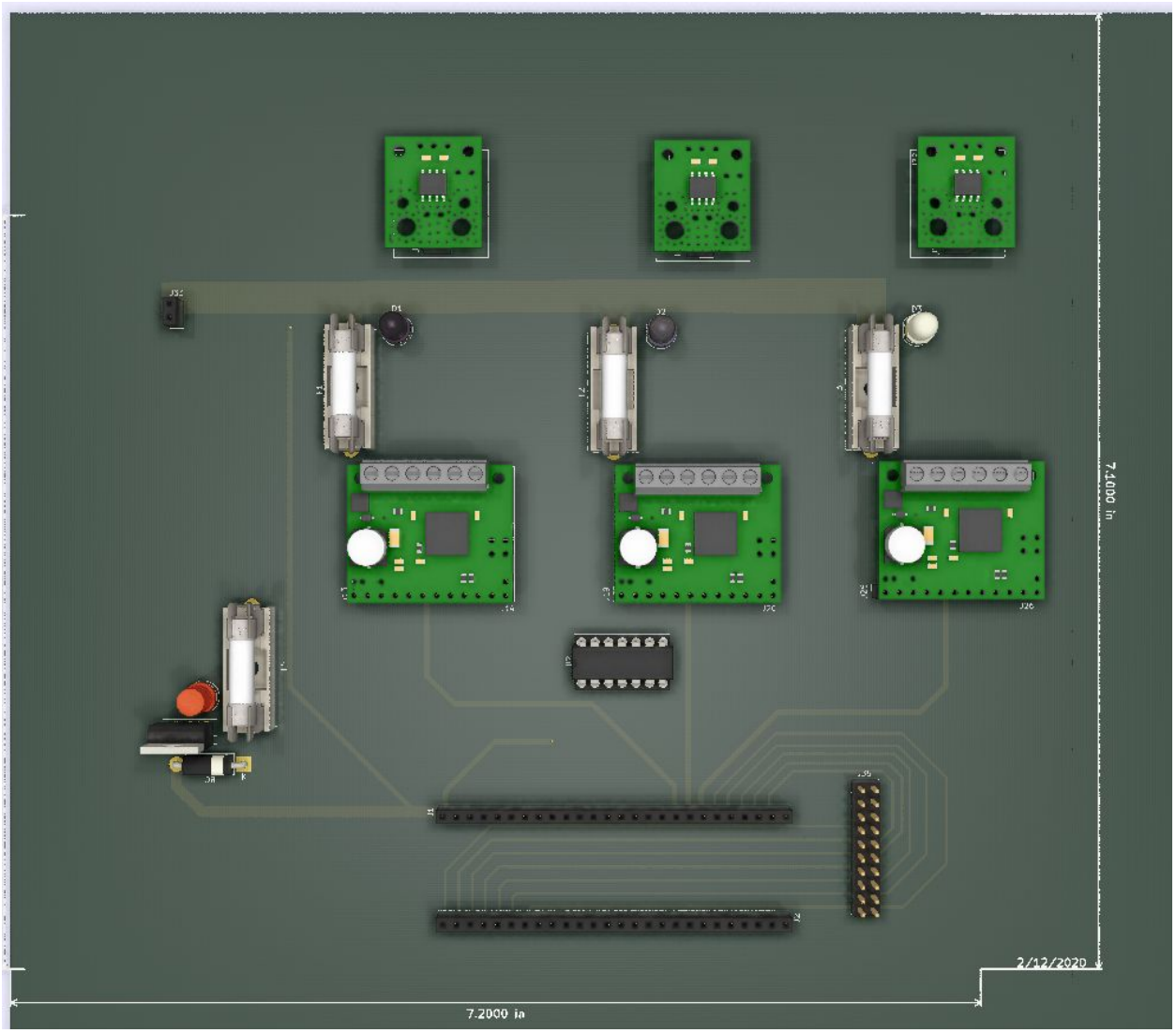


Figure HHC.30: Motherboard 3D Top View (reference [A.3.5] for Bottom View) V0.6 (3/25/2020)

Fabrication

Due to COVID-19 we did not have access to the M-60 milling machine, to mill the motherboard, so we ended up resorting to a perf board rather than a PCB. The main differences in the perf board implementation, is the use of copper tape wiring for our high power traces and power input, as it has a better thermal spread than a normal copper/ threaded copper wire. We also added a 100µF capacitor across the output of the voltage regulator to stabilize the voltage.

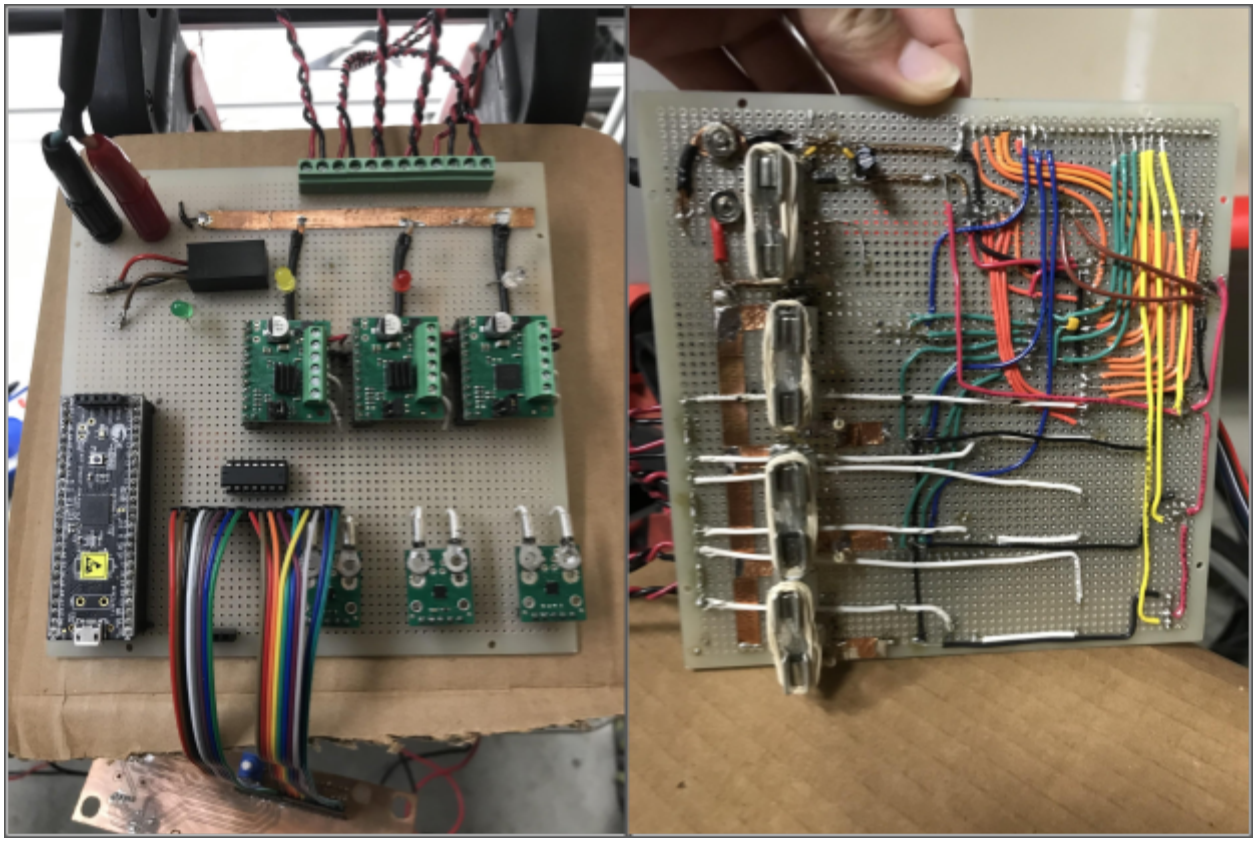


Figure HHC.31: HHC Motherboard Perfboard Version (Top & Bottom View) V2.0, 5/13/2020

Isolation

In order to complete the safety requirement HHC_014 we needed a method to fully isolate the user from the circuity. We decided on using galvanic isolation. Galvanic isolation is a design technique that separates electrical circuits to eliminate stray currents. Signals can pass between galvanically isolated circuits, but stray currents, such as differences in ground potential or currents induced by AC power, are blocked. This is done through separating the GND signals using inductive coils to transmit data.

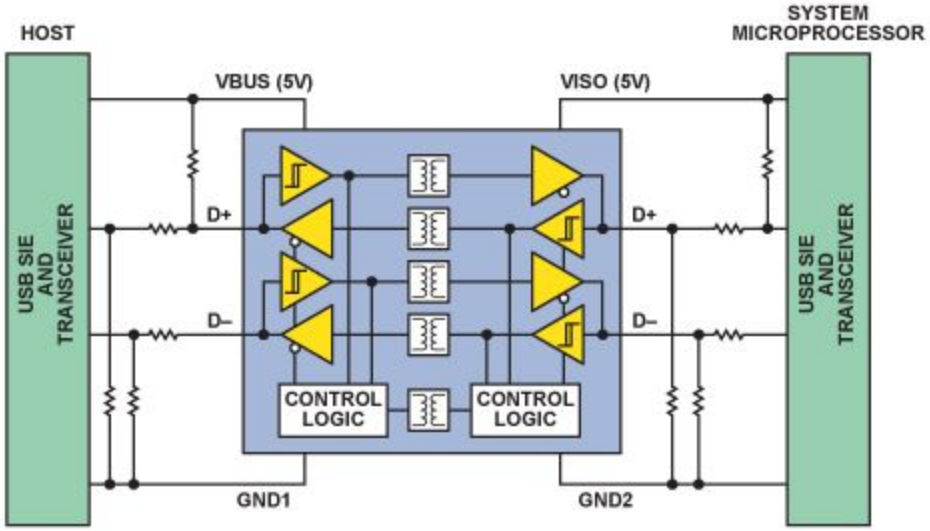


Figure HHC.32: USB Connection Galvanic Isolation Block Diagram

We sourced the adafruit USB isolator-100mA Isolated Low/Full Speed USB. It runs on the ADUM-4180 and ADUM-5000 chips which are galvanic usb isolators for the 1.5Mbps (low speed) or 12Mbps (full speed) offered by USB communications. The device fit all our power specifications and a good fit for our communication methods, given that we planned on using USB [D.9].

HMI

For a system control, display and debugging methods, an HMI for the control electronics was needed. The HMI consisted of an LCD screen for display, and a quadrature rotary encoder to control and input commands. For our LCD controller/driver we sourced the hitachi HD44780U (LCD-II). This LCD was sourced as it is compatible with our voltage rails and microcontroller. It also corresponds to high speed MPU bus interfaces at around 2MHz which works well with our communication method, USB. As for the rotary encoder we went with the EC12PLRGBSDVBF. This was a standard 5V RGB rotary encoder that was available to us, and fit our design.

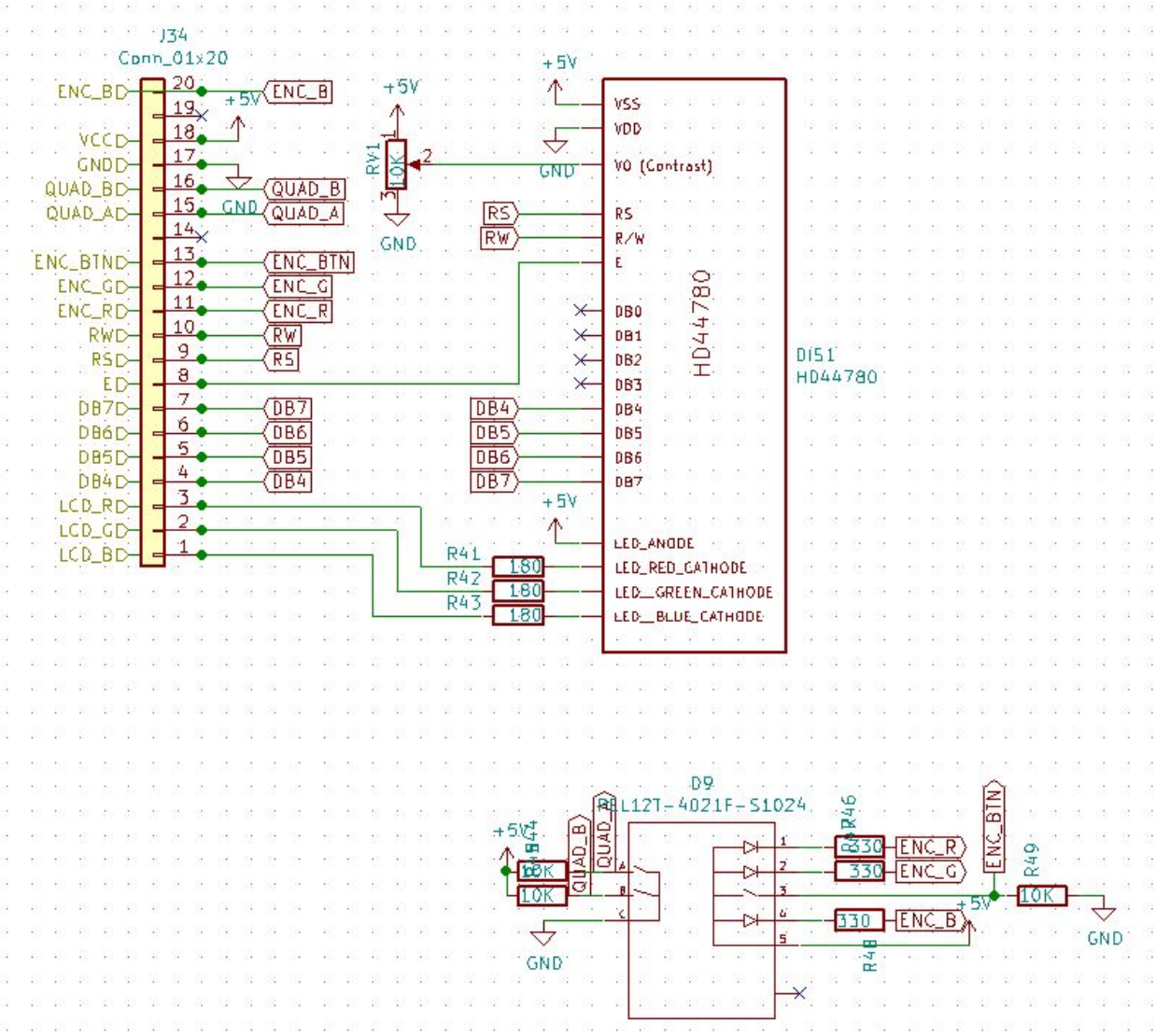


Figure HHC.33: HMI Electrical Schematic V1.0

HMI

We layed out and millied the HMI using the M-60 machine that we had available to us. The HMI is meant to go on the outside of the control electronics box, and thus we tried to hide the circuitry and components on the back side of the board, and only keeping the LCD and encoder on the front. We use the same trace width on the HMI as we did on the motherboard. We set the HMI board dimensions to be 1.5inX5.5in for ease of control by people; the encoder is sufficiently away from the LCD [A.3.6]

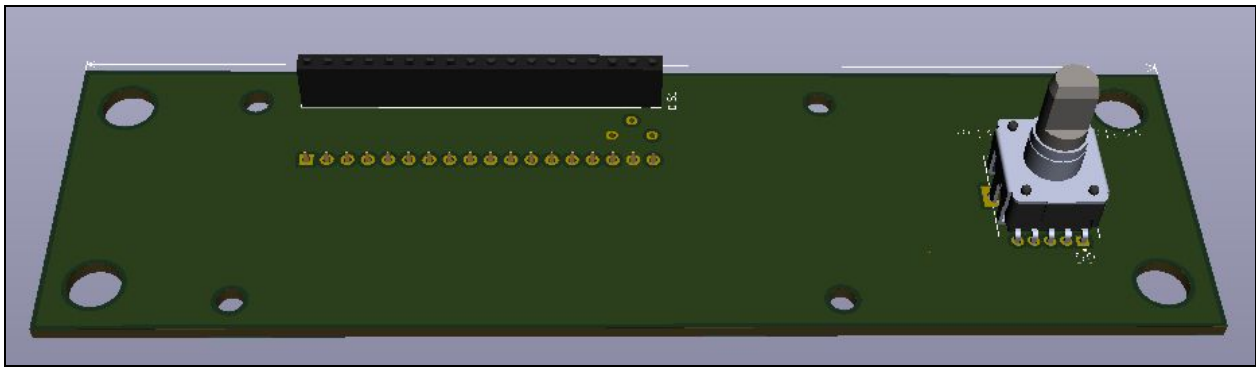


Figure HHC.34: HMI 3D Isometric View Rev 1.0

Control Electronics Enclosures and Panel

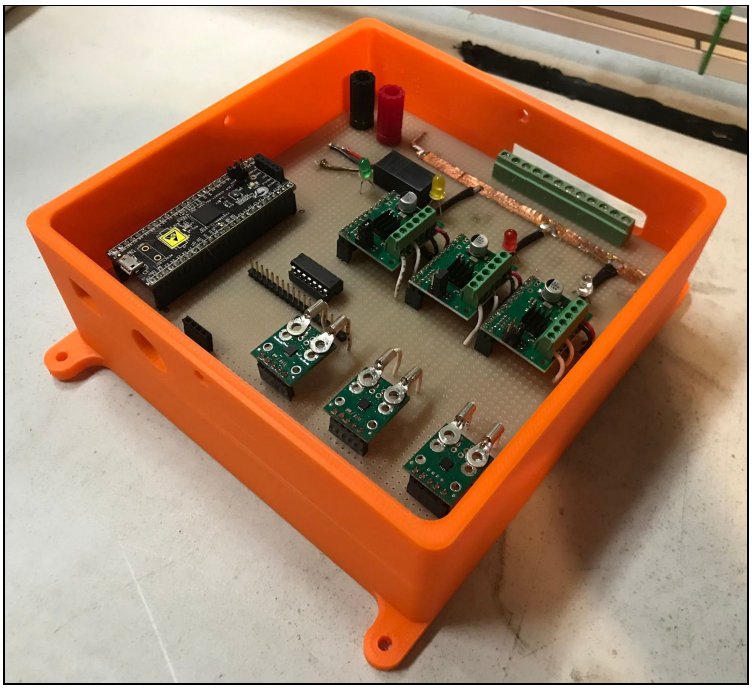


Figure HHC.35: Finished Perfboard Electronics in Enclosure

Figure HHC.36: HHC Control Panel Wired and Mounted on Coil Cage

Control Design

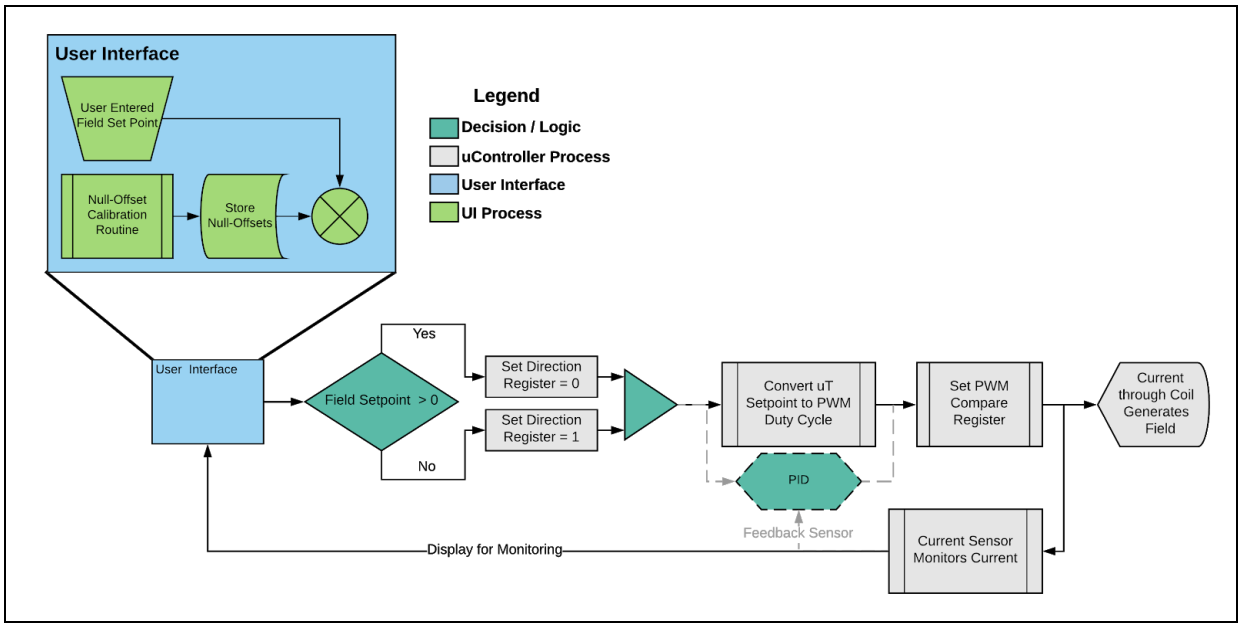


Figure HHC.37: *Block Diagram of HHC Control Architecture Rev. 1.0 6/4/20*

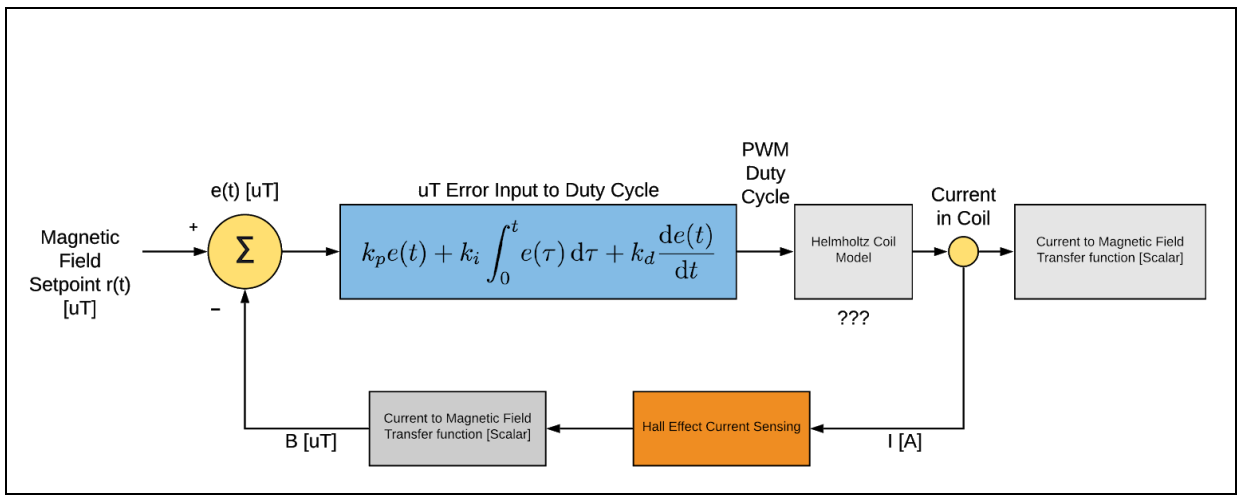


Figure HHC.38*: Block Diagram of HHC PID Controller Library Rev. 2 (3/8/20)*

User Interface

In order to stream data from the HHC and control the magnetic field itself, a method of interfacing with the HHC was required. A standalone user interface that could be run on a separate computer or laptop enables live data streaming from the HHC, along with the ability to control the HHC, from a single application. Additionally, the user interface will allow calibration of the system upon startup to ensure that the ambient magnetic field is accounted for and offset such that the field can be zeroed properly.

Relevant Design Requirements

The relevant design requirements for the HHC user interface are as follows:

- HHC_015
- HHC_016
- HHC_017
- HHC_018

These requirements are aimed at creating a user interface that can be run on multiple different platforms with limited additional setup(HHC_015 & HHC_016), and is easy to maintain and add or modify features in the future (HHC_017). As the user interface will be both sending and receiving information from the HHC's motherboard, the application will also need to be able to perform bidirectional communication with the system (HHC_018).

Overview of User Interface Architecture

The user interface was developed using the PyQt Python wrapper of the Qt UI design suite, chosen based on the quality of its documentation and compatibility with other python libraries. The architecture of the user interface is contained within an application class run as a while loop within the main of the script. Illustrated below is an abstracted view of the application's key elements.

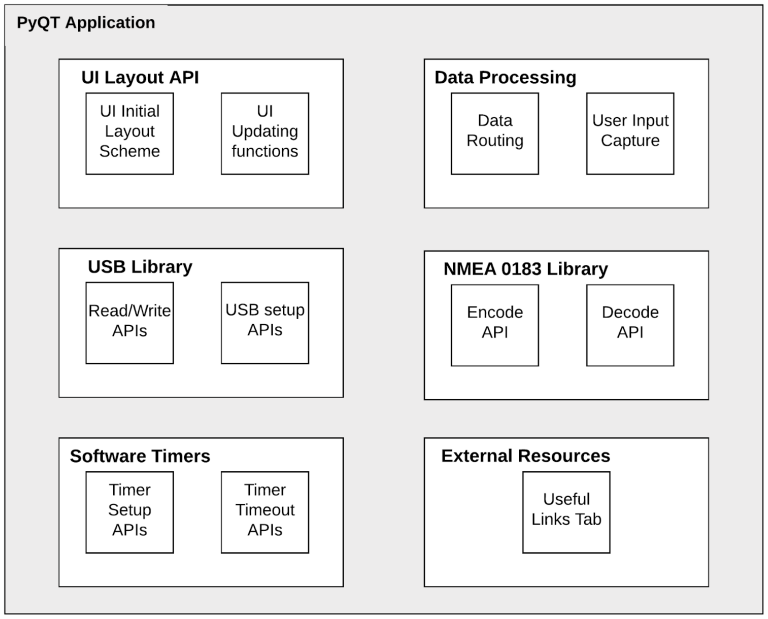


Figure HHC.39: Overview of the user interface architecture Rev. 1.0 6/7/20

The user interface can be broken down into 6 major components within the application:

- UI Layout API: specific text boxes, labels, buttons, tabs, text input areas upon startup. Update APIs alter the data displayed by a label or a label's appearance when called by other functions
- Data Processing: handles routing of data to appropriate labels, filtering of data captured from user input or the sent to/from the HHC. Includes built in error checking to prevent & resolve USB errors.
- USB Library: Utilizes pyusb to implement serial communication between the interface and the HHC. USB setup and read/write functions connect and stream data between host and device. Error handling functions reboot USB connection when data processing detects an error.
- NMEA 0183 Library: USB transfer uses NMEA 0183 encoding scheme for communications. Encoded and decodes all transmitted data to assist data processing functions with data routing.

- Software Timers: Software timers allow data sampling and writing on fixed intervals specified by the designer, which can be enabled and disabled at will. Connected APIs are triggered upon timer timeout events.
- External Resources: Built in links to various resources needed to safely and properly operate the HHC, such as the user manual, SOP, and reference material on magnetic fields.

UI Layout

PyQt simplifies the process of creating the user interface layout via built in APIs for various interface elements. Display labels

USB Communication Protocol

We selected USB as the communication protocol for usage with the HHC, due to the reliability and documentation of the pyusb python library. The pyserial RS-232 serial communications library was also considered briefly, and presented a possibly more efficient and robust method of data transfer than USB. However, the ability to directly connect the HHC to any computer with a USB port, along with the available documentation for pyusb, ultimately outweighed the performance benefits of RS-232.

Communication Protocol								
Design	Efficiency	Complexity	Reliability	Documentation	Additional Hardware	+ Sum	- Sum	Total
Weighting	1x	2x	3x	2x	1x			
USB (pyusb)	+	++	+	++	+	12	0	12
RS-232 (pyserial)	++	+	+	+	-	9	-1	8

Table HHC.6: Tradeoff analysis of USB vs. RS-232 serial communication Rev. 1.0 6/7/20

NMEA 0183 Encoding Scheme

Both the user interface and the HHC firmware itself utilize a simplified version of the NMEA 0183 encoding protocol for packaging data for communication between the two devices. Used primarily in GPS and sonar, NMEA 0183 was selected due to being utilized in a prior class at UCSC, which would allow future engineering students working with the HHC to already have some familiarity with the protocol. A NMEA 0183 message consists of:

- A $ to indicate the start of a message
- A message ID, usually identifying what the data is representing, followed by a comma (,)
- The message payload itself, which is traditionally a number, followed by an asterisk (*)
- A checksum generate by taking the one-byte XOR of the bytes in the message ID and payload
- A terminating newline character (\n), stored internally but not normally displayed

An example of a NMEA_0183 message can be seen below:

$$\$\underbrace{XCUR}_{\text{ID}},\underbrace{1.756}_{\text{Payload}}*\underbrace{1b}_{\text{Checksum}}$$

Figure HHC.40: Example of NMEA0183 encoded message Rev 1.0 6/8/20

NMEA 0183 libraries were created in both C and Python for the firmware and user interface respectively, containing full encoding and decoding functionality, along with checksum verification to handle bad messages. The NMEA 0183 Library allows all incoming and outgoing data for both parts of the HHC to be routed based upon the message ID, simplifying data handling down to a simple state machine in both cases.

Error Handling for USB Comms

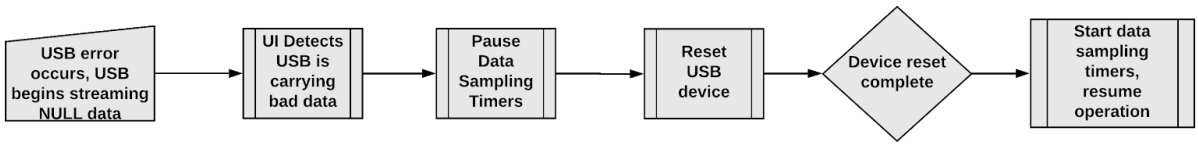


Figure HHC.41: User Interface error handling routine Rev 2.0 6/7/20

During implementation of the USB communications, we had experienced issues with singular USB data packages becoming corrupted and corrupting subsequent messages, which would cause the system to stall and crash. While assumed to be an issue at the protocol layer rather than an issue with the physical signal itself, a comprehensive diagnosis of the problem was not conducted. We resolved this through the implementation of an error handling protocol, outlined above, which allows the interface to detect and catch these errors and reset the device to resolve them. This is a blocking operation that must pause USB communications while it is occurring, after which data transfer will resume as normal.

Null Offset Calibration Routine

Prior to arming the HHC, the ambient magnetic field strength along each axis must be accounted for, such that zeroing all three axes results in zero magnetic field at the center of the cage. This was achieved via the implementation of a calibration routine into the user interface.

Calibration is done via taking a series of magnetometer readings on each axis and calculating the average magnetic field from those readings, which is stored as the null offset for each axis. These offsets are displayed on the UI and added to each axes' setpoint range, changing from +/- 150uT to +/- 150uT plus the null offset value for that axis. The calibration process is illustrated in the figure below:

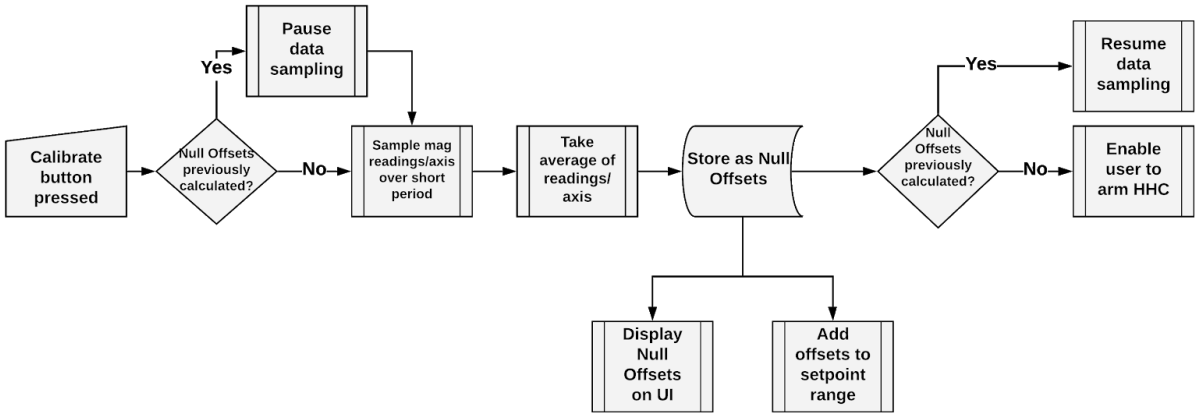


Figure HHC.42: User Interface null offset calibration routine Rev. 2.0 6/7/20

The calibration routine must be run at least once before arming the HHC when first starting up the interface, and arming the HHC is actually impossible without first calibrating the system. Once calibrated the first time, the HHC can be armed at any time, indicated by the calibration label turning green(Purple circle below) and the null offsets being displayed (Red circle below). The calibration process may be repeated at any point during testing to update the system's null offsets. If a recalibration is initiated while the HHC is armed, the calibration routine will pause the system to recalibrate before resuming normal operations once complete.

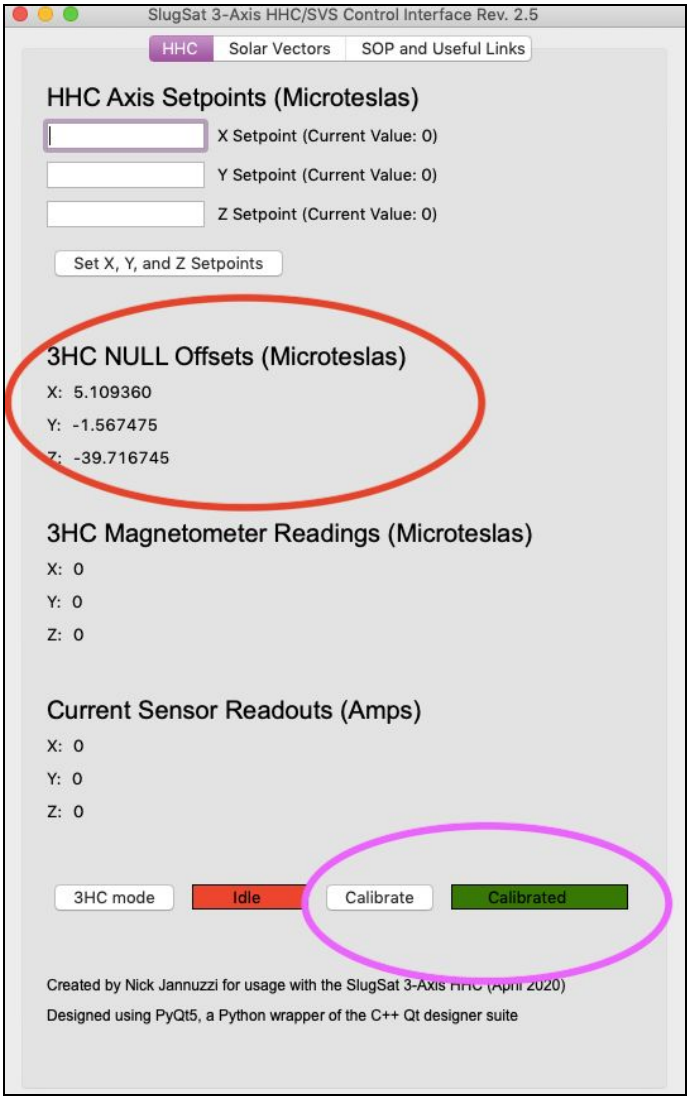


Figure HHC.43: HHC UI after calibration, with null offsets displayed on screen

Results

Measurement and Validation of Field Generated in Cage

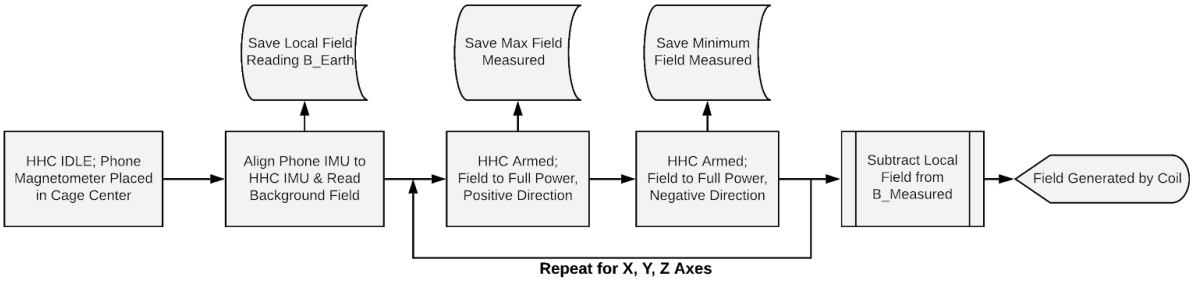


***Figure HHC.44**: Flowchart of Coil Field Measurement Process Rev. 1.0 6/3/20*

	X(raw)	X(corrected)	Y(raw)	Y(corrected)	Z(raw)	Z(corrected)
Max	+157.6	+141.2*	+169.7	+148.9*	+101.0	+141.74*
Min	-128.8	-145.2*	-125.6	-146.4*	-179.0	-138.26*

Table HHC.7: Flowchart of Coil Field Measurement Process Rev. 1.0 6/3/20

*HHC_001 Spec: +/- 130uT range per-axis

Standard Operating Procedures

Setup & Calibration

1. Mount Test Article to Hemispherical Air Bearing
2. Add counterweights until center of mass is at the center of the bearing
3. Connect the air line to the air bearing.
4. Increase air pressure until the bearing lifts off the surface. Note air pressure.
5. Turn off air.
6. Connect HHC to power and connect the USB to the host computer.

Future Work

Reassemble the HHC in the Lab
The HHC had to be disassembled into pieces for storage after this year. The first order of business will be to reassemble the unit in a permanent lab space. We shot a video guiding on how to disassemble and reassemble the unit which can be found in the drive.

Implement and Integrate Closed Loop Control of the Magnetic Field
This year we were not able to get the closed loop control method working. Instead we were able to get usable results with an open loop control method. This was simply a factor of time and the hardware support is already done to be able to support it.

Improve Magnetometer Calibration Method
The magnetometer was calibrated by using a phone IMU placed beside it and measuring the error. This error was used to iteratively adjust the scale factor of each axis. This is a quick and dirty method of calibration and should be improved using a least squares regression method and tumble testing. This method of calibration is taught in-depth in the Sensors and Sensor Technologies class at UC Santa Cruz.

Finish Designing and Fabrication of the Satellite Testing Mount
We were able to fabricate a working version of the hemispherical air bearing using a 3D printer. This was tested working. The next step is to design and fabricate a way to mount the satellite to it for ACS testing.

Fabricate a PCB Version of the Control Electronics
Due to the COVID crisis, we were unable to mill or send out for PCB fabrication. Instead we hand soldered a perf board prototype. This is a working demonstration but should be improved with a proper PCB.

Explore the use of a DAQ with High Current Amplifier
In our design, we chose to use PWM to control the current through the coil. This method relies on the inductance of the coil to filter the current and create a DC average field. While this method works, it is prone to ripple. In the future, we would recommend the use of a DAC with dithering and voltage controlled current sources to create a more stable and less noisy field.

Conclusion

Despite COVID related limitations and setbacks, the team was able to successfully design and implement a functioning 3-Axis Helmholtz Coil over the course of the year. While we were originally planning to implement a more complex control scheme and calibration process, limited access to lab equipment required us to make do with what was available. Several delays due to the need to replace faulty components ultimately stalled completion of the HHC until the end of spring quarter rather than the middle of spring as originally planned. While this did not affect the HHC ultimately, it came at a cost of reduced time to work on the STS accordingly.

Future work will need to start with reassembly of the HHC and validation that it is functioning properly. Once this has been done, upgrades to the control software and electronics will be the primary area of improvement. Closed loop PID control libraries have already been created, but will need to be tuned and integrated with the HHC firmware. The control motherboard can potentially be remade as a PCB given access to a mill, and the user interface software can be improved and refined to better optimize performance and functionality. Overall, the HHC constructed this year is a functional design, with obvious areas where additional work can be done next year.

Mechanical Shock Test Stand (STS)

Caleb Cotter & Nick Jannuzzi

Introduction

During the launch of the satellite, multiple separation events of launch vehicle stages will occur which will create large mechanical shocks on the spacecraft. To qualify the assembly before launch, the spacecraft will need to be shock tested to verify that the components will survive launch. A system capable of replicating the shock loading generated during launch can enable in house testing of both subsystems and the full craft. As validation is required before sending the flight unit to the launch provider for final checkoff, the ability to test in-house can greatly speed up the cubesat's development process.

Project Documentation, Code, CAD Models, and Schematics: https://github.com/SlugSat/STS

Design Requirements

<table>
<tr><th>Requirement. ID</th><th>Design Requirement</th><th>Reference</th></tr>
<tr><td>STS_001</td><td>STS shall deliver a shock to the test mass up to 4.00kg with the following shock response spectrum (SRS) characteristics. All SRS values assume Q = 10.
<table>
<tr><th>SRS Fundamental Frequency</th><th>Acceleration</th></tr>
<tr><td>30Hz</td><td>>= 5g</td></tr>
<tr><td>100Hz</td><td>>= 100g</td></tr>
<tr><td>700Hz</td><td>>= 1500g</td></tr>
<tr><td>1000Hz</td><td>>= 2400g</td></tr>
<tr><td>1500Hz</td><td>>= 4000g</td></tr>
<tr><td>5000Hz</td><td>>= 4000g</td></tr>
<tr><td>10000Hz</td><td>>= 2000g</td></tr>
</table></td><td>NASA GEVS Standard (Document Number GSFC-STD-7000 Rev. A) & CubeSat Design Specification Rev. 13</td></tr>
<tr><td>STS_002</td><td>STS shall provide mounting such that a cubesat upto 3U can be mounted and shock delivered in the X+,X-,Y+,Y-,Z+,Z- directions.</td><td>Shock Delivered in All Directions</td></tr>
<tr><td>STS_003</td><td>STS shall fit in an envelope no larger than 32” x 76” x 76” when not in use.</td><td>US Minimum Door Opening</td></tr>
<tr><td>STS_004</td><td>STS Data Acquisition system shall be powered by NEMA 5-15P a.k.a Type B plug</td><td>Compatible with standard US wall outlet</td></tr>
<tr><td>STS_005</td><td>STS shall have a data acquisition system that can record acceleration data such that the NASA GEVS shock response spectrum can be calculated</td><td>Allow collection and transfer of data to computer for processing</td></tr>
</table>

Table STS.1: *Design Requirements for STS Rev. 2 (6/6/20)*

STS_001 - Shock Response Spectrum Requirement
NASA GEVS mechanical qualification standards specify a set of fundamental frequencies and specific minimum accelerations per each frequency that the CubeSat must be able to survive. The SRS itself is a plot of the max acceleration at each frequency of interest. The shock test stand must generate an SRS of either equal or greater magnitude at each frequency.

STS_002 - Mounting Requirements for CubeSat Test Article
The shock must be delivered in each orthogonal direction to verify that the satellite can survive shock conditions on launch no matter the mounting configuration. Cubesats are secondary payloads and typically do not get to choose the placement and orientation, therefore qualification for all mounting orientations is essential.

STS_003 - Size Requirements
To allow the STS to be moved between lab spaces when necessary, the entire system must be able to fit through a standard US single door frame (32” x 76” x 76”) when not in use**.**

STS_004 - Power Interface
The data acquisition system must be able to run on a standard 115V U.S. outlet. The use of a standard power connector ensures compatibility in as many use cases as possible.

STS_005 - Data Recording System Function Requirement
In order to verify that applied shock impacts meet the NASA GEVS spec, the STS must be able to record the data into a format capable of being processed to generate an SRS that can be compared to the NASA GEVS standard.

Prior Art

Intro to Shock Testing

Shock testing is performed on spacecraft to qualify and validate hardware prior to launch. A shock test imparts a finite impulse of energy on an object which results in a high acceleration across a short duration. The functionality and performance of the unit under test is performed before and after the shock to ensure the device can withstand the shock.

Pyrotechnic vs. Classical Shock

Mechanical shock testing can be done either using a pyrotechnic charge or mechanical system. Shock with a pyrotechnic charge allows for high energy impulses under a short duration which is best suited for testing high frequency shock response. Mechanical shock test mechanisms are known as classical shock tests and can be done with a variety of mechanisms.

Pyro

A pyroshock test is done by mounting a CubeSat to a resonant plate and then detonating a charge against the plate. This imparts an impulse to the system which is shaped by the resonance plate and coupled to the satellite.

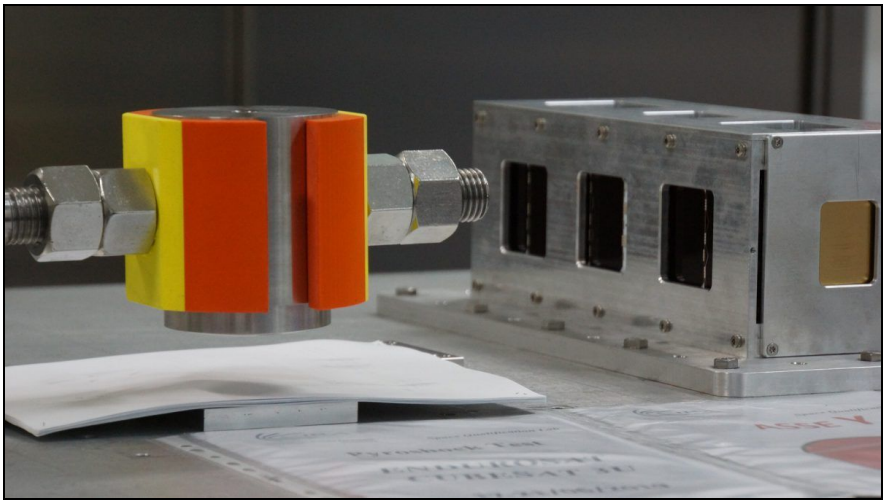


Figure STS.1: Pyroshock Testing of a 3U Cubesat

(Photo Credit: EnduroSat - 6/7/2020)

Classical

Classical shock tests do not involve any explosive charges but do not come without their own hazards. These types of shock tests involve using a moving mass and using kinetic energy transfer into a resonant plate which the test article is mounted to.

Drop Test

Shock testing using a drop test is the most mechanically simple method of shock testing. In a drop test, the test article is mounted on a set of vertical rails to guide the fall and the unit it lifted to height and then released and shock is imparted when the unit impacts the ground.

Pneumatic

In pneumatic shock testing, a piston is used with a mass mounted at the end. Using pressurized air and a high speed valve, the piston is accelerated into the test article delivering a blow. The strength of the impact is varied by adjusting the air pressure feeding the regulator.

Proof Mass

Proof mass shock testing is similar to the drop test, except instead of moving the test article, it is fixed and a mass is lifted on a pivot arm or rails and then dropped to deliver a blow to a resonant plate which is mounted to the test article. In this test, the proof mass and lift height can be varied to change the shock delivered.

Proposed Technical Solution

Selection of Shock Delivery Method

For our solution, we chose to build a proof mass based shock test stand. This choice was made using the tradeoff analysis in the following table. For the design, we chose to favor safety and manufacturability since the final solution needed to be operable in an office like lab space and SlugSat has limited availability of fabrication tools. As seen in the trade of analysis, the choice was close between the mass swing arm and the drop test method. Ultimately the mass swing arm was decided on since the drop test requires the test operator to be closer to the impact zone where the test hammer allows the operator to release the hammer and it will swing away from them.

Shock Delivery Method										
Design	Safety	Cost	Time	Manufacturability	Max Impact Force	Impact Accuracy	Impact Repeatability	+ Sum	- Sum	Total
Weighting	3x	1x	2x	3x	2x	2x	1x			
Pneumatic Piston	--	-	--	+	-	-	--	3	-13	-12
Mass Swing Arm	++	++	+	++	++	--	++	18	-4	18
Pyrotechnic	-	--	---	-	-	-	+	1	-17	-17
Controlled Drop	+	+	+	+	+	+	++	12	0	15

Table STS.2: *Tradeoff Analysis of Shock Delivery Methods Rev. 1 (4/29/20)*

Form and Function of Solution

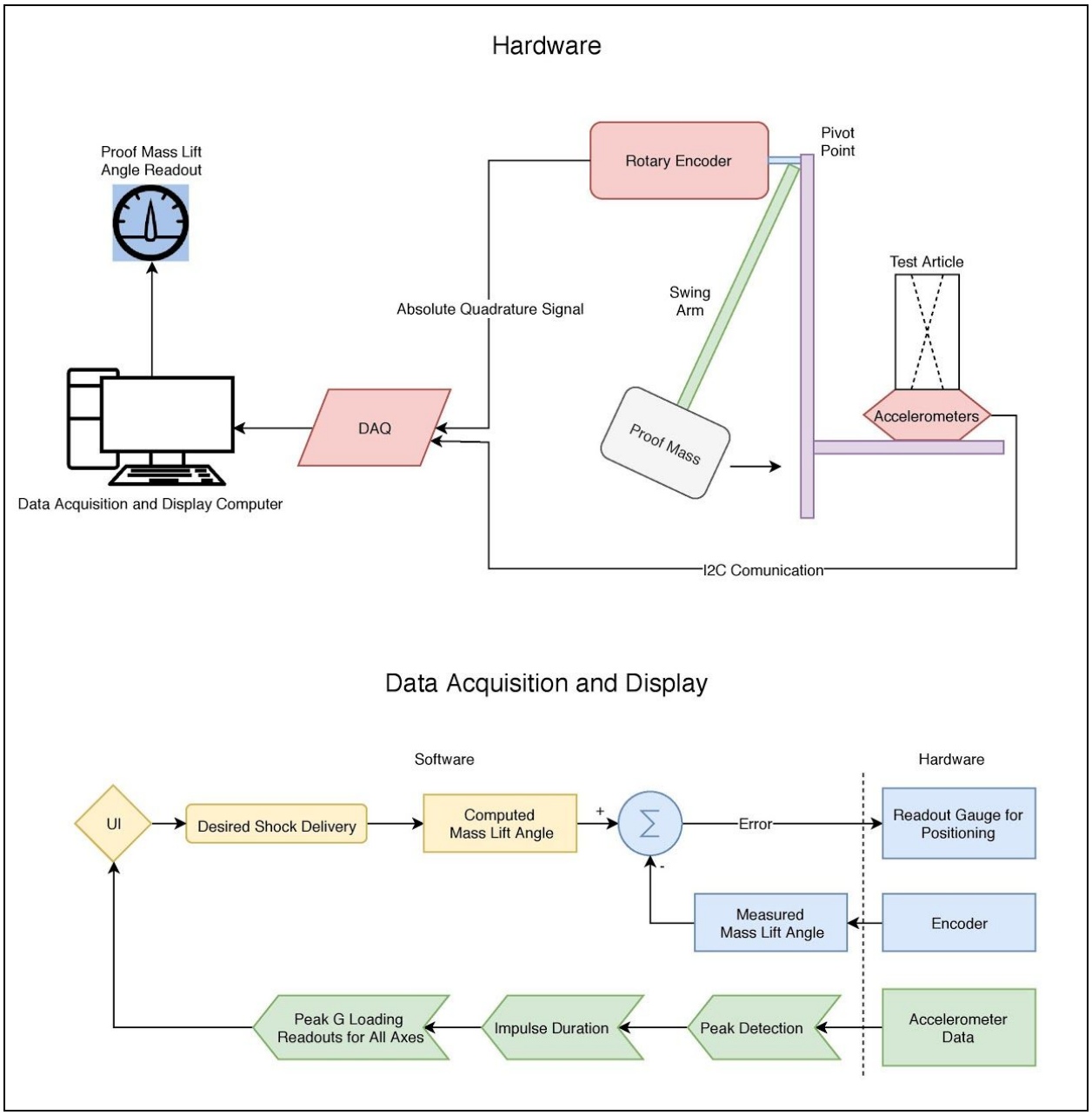


Figure STS.2: *Shock Test Stand Proposal Diagrams (Rev. 1.0 12-13-19)*

Shock Testing Procedure (Proposed)

1. Weigh test article
2. Perform functionality test of all features of test article and photograph test article
3. Mount and secure test article to STS
4. Enable data acquisition system
5. Calculate lift angle necessary to generate SRS for given test article mass
6. Ensure area in and around STS is clear on non-operator personnel
7. Lift proof mass to the required angle as shown on DAQ readout
8. Release proof mass
9. Wait for data recording to finish
10. Load data into post processing software
11. Analyze and compare data to NASA GEVS spec
12. Perform functionality test of all features of test article and photograph test article; compare to data gathered in step 2
13. Combine photos, functionality note, and SRS test data into report for engineering use

Derivation of Engineering Requirements

Generation of the Shock Response Spectrum

From design requirements, the system needs to generate a shock impact with the SRS defined by NASA GEVS [HHC_001]. We started by researching what the SRS is and how it is different from a FFT of the accelerometer signal. We found that the SRS is created by simulating the response of n-number of 2nd order systems with different fundamental frequencies and then taking the peak shock load for each system and plotting it against the fundamental frequency.

SRS Generation Process

1. Choose a damping ratio for the SRS (5% damping / Q=10 is most commonly used)
2. Use a digital filter to model a Single Degree of Freedom (SDOF) system of frequency fn and damping factor ξ.
3. Apply the transient impulse as an input to the system and calculate the response acceleration waveform.
4. Retain the peak positive and negative responses occurring during the pulse's duration and afterward.
5. Select one of these extreme values and plot it as the spectrum amplitude at fn.
6. Repeat these steps for each (logarithmically spaced) fn desired.

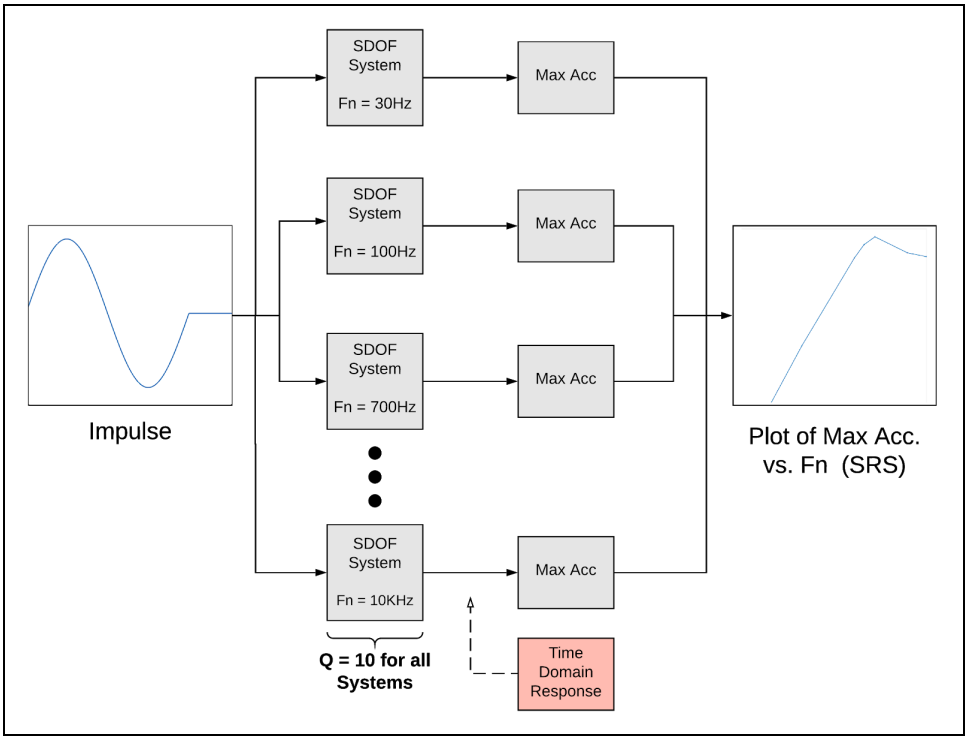


Figure STS.3: *Flowchart of SRS Generation Process (Rev. 1 5-12-20)*

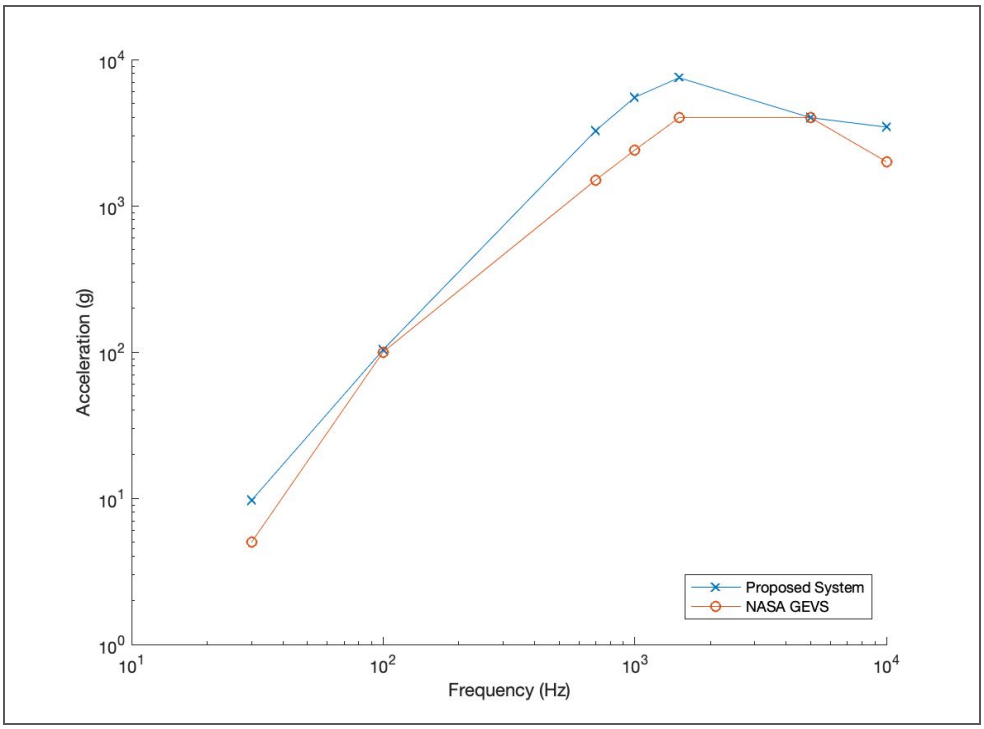


Figure STS.4: *SRS Generation Results for 3250G Sine - 740us Period (Rev. 1 5-10-20)*

In the original effort, we looked into ways to reverse the calculation process. This method would involve starting with a known SRS and using it to back into the acceleration profile necessary. However, multiple acceleration curves can generate the same SRS, thus there is not a 1-to-1 relationship between an SRS and the acceleration profile.

Instead, the best way we found to select an acceleration profile was by iterative trial-and error. The preceding steps were followed to narrow in on an acceleration profile:

1. Generate time-series acceleration data of single sine-wave period acceleration profile.
2. Run the script to generate SRS.
3. Evaluate the SRS vs Spec
4. If the magnitude is too low, increase the acceleration magnitude & vice versa.
5. If the peak frequency response occurs at a lower frequency, decrease the period of the impulse.
6. Repeat this procedure until SRS is above the NASA GEVS at all points.

We found that the best way to dial in the acceleration profile is to use the intuition of how linear systems behave to a given impulse. Since there is no closed-form analytical solution, this process relies on the engineer's intuition.

Translation to Subsystem Requirements

After following the iterative process for generating an SRS and finding an acceleration profile that meet STS_001, we determined that a single period sine impulse of 3250G peak magnitude with a 740us period would generate the SRS shown in figure STS.4, which meets spec.

Acceleration Profile

- Sine Wave
- Single Period
- 3250G Peak Magnitude
- 740us Period (1350Hz)

This acceleration profile directly drives the subsystem requirements. The mechanical system will need to be capable of imparting this acceleration on a test article up to 4.00Kg in all orthogonal directions [STS_002] and fit within the required envelope [STS_003]. To allow evaluation of performance, a data recording system must be capable of recording the acceleration profile such that an SRS can be generated for a test impact [STS_005]. These requirements are further broken down into component level requirements in the following sections.

Shock Delivery Mechanism

Shock Delivery Method

Based on the previously conducted tradeoff analysis, we choose to construct the STS around a proof mass design. The kinetic energy of a swinging proof mass must be converted into the required acceleration profile of the test article. The dynamics of this can be modeled as a classical 2nd-order system.

2nd Order Mechanical Systems

There exists multiple ways of implementing a 2nd-order mechanical system, most notably being the discrete mass-spring damper and the deflection plate. A discrete mass-spring damper system involves mounting the test article on linear rails coupled to a discrete mass, spring, damper much like that of the suspension found on a car. Striking the test article causes it to resonate based on the properties of the spring and the damper. A key downside is that the mass-spring damper requires a high stiffness and damping not well approximated at small scales using discrete components.

The deflection plate mounts the test article to a solid plate rigidly mounted at one end. This plate can be modeled as a cantilever beam. Impact of the proof mass on the free end of the beam results in linear elastic deformation of the plate, imparting acceleration on the mounted test mass. The material properties and shape of the deflection plate approximate a 2nd order spring with a high degree of stiffness and damping with relatively low mass in a singular component.

Due to the difficulty of obtaining discrete components with the necessary stiffness and damping factor to design a mass-spring damper system, we elected to utilise a deflection plate for our design.

Shock Mechanism								
Design	Cost	Manufacturability	High Stiffness	Low Mass	Low Damping	+ Sum	- Sum	Total
Weighting	1x	1x	3x	2x	1x			
Discrete Mass-Spring-Damper	---	-	-	-	+	1	-9	-8
Deflection Plate	+	+	+	+	-	7	-1	6

Table STS.3: *Tradeoff Analysis of Shock Delivery Methods Rev. 1 (4/29/20)*

Test Stand Structure Design

Form Factor of Test Stand

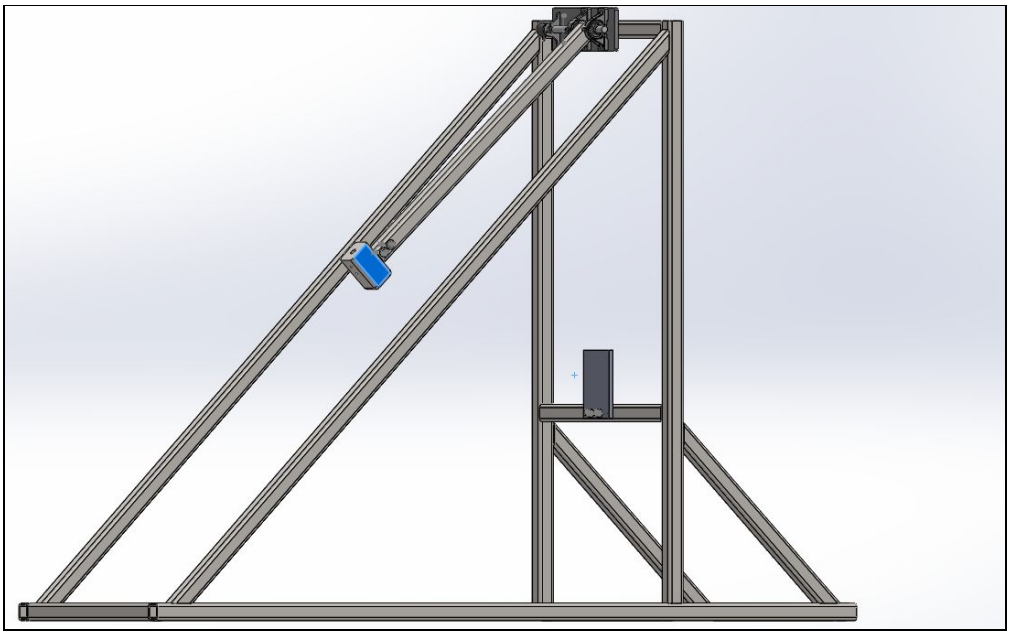


Figure STS.5: *Angled View of STS System Assembly (Rev. 2.0 6-3-20)*

Impact Hammer

The impact hammer is the first half of the shock delivery method, and is responsible for generating the force that will be transferred as an impulse to the deflection plate and by extension, the test article. The hammer can be represented as a uniform rod with a point mass mounted on one end, known as a compound pendulum. In the design of the shock test stand, the primary values we can adjust are the length of the arm, the dimensions of the head itself (length, width, height), and the material of both the arm and the head. The following equations outline the significant parameters of the impact hammer.

EQ STS.1: Mass of a rectangular hammer head

$$M_{Hammer} = L_{Hammer} W_{Hammer} H_{Hammer} \rho_{material}$$

EQ STS.2: Moment of Inertia of a Compound Pendulum

$$I_{Hammer} = \frac{1}{3} M_{Arm} L_{arm}^2 + M_{Hammer} L_{Arm}^2$$

EQ STS.3: Center of Mass of a Pendulum

$$X_{CM} = \frac{(\frac{M_{Arm}L_{Arm}}{2} + M_{Head}L_{Arm})}{M_{Head} + M_{Arm}}$$

*EQ STS.4: Period of a Complex Pendulum**

$$T_{pendulum} = \pi\sqrt{\frac{I_{Hammer}M_{Hammer}}{gM_{Arm}X_{CM}}}$$

* Since the hammer arm only completes a downward swing before striking the plate, this value is half of what the period would normally be.

EQ STS.5: Angular Velocity of a Pendulum

$$\omega_{hammer} = 2\pi\frac{1}{T_{Hammer}}$$

EQ STS.6: Kinetic Energy of a Pendulum

$$K_{Hammer} = \frac{1}{2}I_{Hammer}\omega_{hammer}^2$$

Shock Deflection Plate

The shock deflection plate is the second half of the shock delivery method, acting as the means of transferring the kinetic energy of the hammer into an acceleration acting upon the test article. Mounted such that the deflection plate can be represented as a cantilever beam, the collision between the impact hammer and the plate causes a linear elastic deflection to impart the acceleration on the test article. The primary variables we can adjust for the deflection plate are the dimensions (Length, Width, Thickness) and material of the plate, and the distance from mounting that the force is applied at. The equations that follow outline major values necessary for determining the SRS generated from the impact:

EQ STS.7: Mass of a plate from dimensions and material density:

$$M_{Plate} = L_{Plate}W_{Plate}T_{Plate}\rho_{Plate}$$

EQ STS.8: Total Mass of Plate and Test Article:

$$M_{Total} = M_{Plate} + M_{TestArticle}$$

EQ STS.9: Equivalent Spring Constant of Plate:

$$K_{Plate} = \frac{W_{Plate} T_{Plate}^3 E_{Plate}}{4 L_{Plate}^2}$$

EQ STS.10: Damping Coefficient of the Plate:

$$\zeta_{Plate} = \frac{\zeta_{ratio}}{2\sqrt{K_{Plate} M_{Total}}}$$

EQ STS.11: Where Damping Ratio is:

$$\zeta_{ratio} = 0.5 * 10^{-4}$$

EQ STS.12: 2nd Moment of Inertia of a Plate:

$$I_{Plate} = \frac{L_{Plate} T_{Plate}^3}{12}$$

EQ STS.13: Natural Frequency of the Plate:

$$\omega_{n\,Plate} = 1.875^2 * \sqrt{\frac{E_{Plate} I_{Plate}}{M_{Total} L_{Plate}}}$$

Deflection Calculations

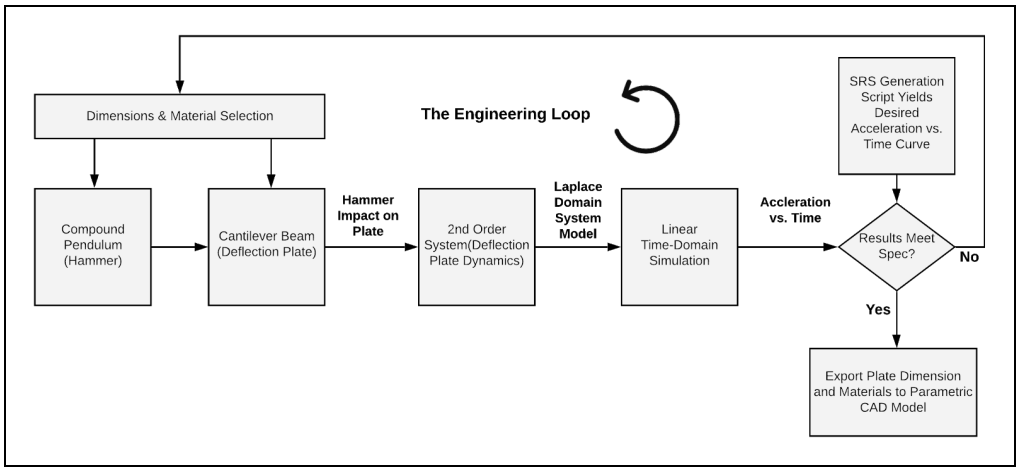


Figure STS.6: *Flowchart of STS Impact System Design Process (Rev. 1.0 6-4-20)*

Sizing of both the plate and the hammer was attempted via simulations, combining the above equations STS.1-STS.14 with the SRS generation algorithm to attempt to find parametric values for components analytically. We determined that experimental setups for the shock delivery method would be needed to properly derive the correct dimensions for the impact hammer and plate. The MATLAB script utilized for this can be seen in the appendix and found on the SlugSat Github.

Plate Mounting

As the deflection plate is a cantilever beam, it must be mounted to the STS frame at one end. Due to the potential need to create multiple interchangeable plates for different tests or replace fatigued plates over time, two large bolts were chosen instead of directly welding the plate to the frame.

Hardware Mounting

Arm Position Encoder

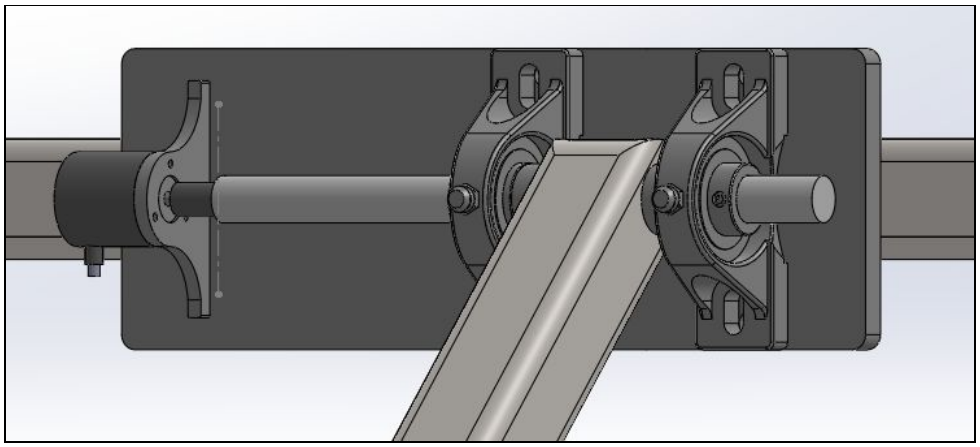


Figure STS.7: *Closeup view of STS arm encoder and arm mounting (Rev. 1.0 6-4-20)*

A heavy mounting plate was designed to mount both the impact hammer arm and the arm angle encoder. Large ball bearing mounts were selected to hold the hammer arm, and the mounting shaft is extended to connect to the arm angle encoder mounted at the end of the plate. In order to alleviate the stress on the encoder's shaft, an additional mounting bracket was designed to fit with the screw holes on the encoder. This entire fixture will be welded and centered on the top cross bar of the STS frame.

Shock Accelerometer

The shock accelerometer will need to be mounted to the test article in question in order to properly collect data. Additionally, the DAQ will need to be located as close to the accelerometer as possible to limit noise during operation. While not designed during this year, a proposed solution would likely utilize a mounting clamp connected to the accelerometer board to securely attach to the test article in any of the 6 orthogonal configurations.

Test Article Mounting

Our proposed method of mounting a Cubesat to the deflection plate involves the PolySat testing mount. Based out of Cal Poly San Luis Obispo, PolySat routinely lends out a standardized test mount for usage by Cubesat groups that is comparable to the PPOD the satellite will be stored in during launch. Rather than develop an entirely new test mount, we elected to reach out to PolySat about obtaining a mount, as this would require simply adding a hole pattern to the plate to match the design. While we had sent them a message about obtaining mechanical specs of the mount in order to add the hole pattern to the deflection plate, we have not heard back from PolySat as of the time of writing.

Summary of Mechanical Design

The mechanical design has been finalized in regards to the mounting of both the deflection plate and the hammer. While the final parametric values of the impact hammer and deflection plate need to be determined from the MATLAB script, the CAD model has been designed such that the values can be easily changed without having to reassemble the model. PolySat will need to be contacted again regarding the test mount, and the DAQ and accelerometer mounts will need to be designed and mounted.

Data Acquisition System

The Data Acquisition System or DAQ, must be capable of recording data such that the rapid shock acceleration can be captured and stored for post processing. DAQ acquisition systems can be purchased off the shelf for most applications but are too costly for us to acquire and thus we developed our own embedded system in-house.

Engineering Requirements

Accelerometer

The following requirements were developed based on the need to capture the acceleration profile of 3250G's with a 740us period.

Property	Requirement	Reason
Dynamic Range	+/- 3500 G	Max acceleration for designed acceleration profile
Transducer Bandwidth	> 2.7 KHz	Shock impulse frequency
Sample Rate	> 30KHz	10x Nyquist Frequency to capture acceleration profile
Measurement Resolution	< 3 G / LSB	Reduced Quantization Error
NIST Traceable Calibration	Required	Calibrated sensor to guarantee accuracy of test results

Table STS.4a: *Table of Derived Accelerometer Requirements for STS DAQ Rev. 1 (5/11/20)*

Rotary Encoder

Property	Requirement	Reason
Angular Resolution	> 360 P/R	Angle of the lift arm should be able to resolve the angle to a 1 degree resolution.
Angular Rate	> 115 rpm	Based on the max height of the unit of 76in we can assume that this is the upper bound for the arm length and thus the max angular velocity of the arm will be ~113rpm.

Table STS.4b: *Table of Derived Encoder Requirements for STS DAQ Rev. 1 (5/11/20)*

Sensor Selection

Accelerometer

Accelerometers are widely used in automotive, smartphones, and other consumer electronics. Typically these are low-cost MEMS devices that are widely available. However, a MEMS-based device is typically limited to a maximum sensing range of 200Gs and is limited in sensor bandwidth. MEMS commonly use I2C for the interface, which limits the update rate to ~500Hz. These factors led us to ruling out the use of a MEMS device for measuring acceleration.

Instead we found that a Piezoelectric-based analog accelerometer would meet the required specifications and they are specialized for use in shock applications. The accelerometer of choice was a TE Connectivity 834M1.

Tri-Axial Accelerometer Comparison			
Model	MPU9250	ADXL377	834M1
Technology	MEMS	MEMS	Piezoelectric
Dynamic Range	+/- 16G	+/- 200G	+/- 6000G
Transducer Bandwidth	0 - 260Hz	0 - 500Hz	2 - 6000Hz
Interface	I2C	Analog	Analog
Sample Rate	500Hz	Depends on ADC	Depends on ADC

Table STS.5: *Comparison of Accelerometers for STS Rev. 1 (6/7/20)*

Rotary Encoder

In order to control the amount of impulse delivered for different test masses, the angle of the arm needs to be well-known. The user can view the read-out of the DAQ to set the lift angle. The selection of the rotary encoder was straight-forward given that the requirements are not outside of the range of a common encoder. We chose an encoder from Sparkfun with the following specifications:

Rotary Encoder	
Model	YUMO E62B-CWZ3E
Angular Resolution	1024 P/R
Angular Rate	6000 rpm
Interface	Quadrature

Table STS.6: *Selected Encoder Relevant Specifications Rev. 1 (6/7/20)*

Microcontroller Selection

In a data acquisition system, the sensor requirements drive the design / selection of the data recording and processing. The choice of an accelerometer with an analog interface with a high sample rate narrowed down the available options for microcontrollers. On other test equipment the team developed, we used a PSoC5LP, and thus this was the first microcontroller we evaluated. Onboard, it has a SAR ADC with 12-bits of resolution capable of running at up to 1MSPS as well as a plethora of other peripherals. We chose to stick with this uController for commonality between systems, we have already written libraries for it that we could leverage.

Block Diagram

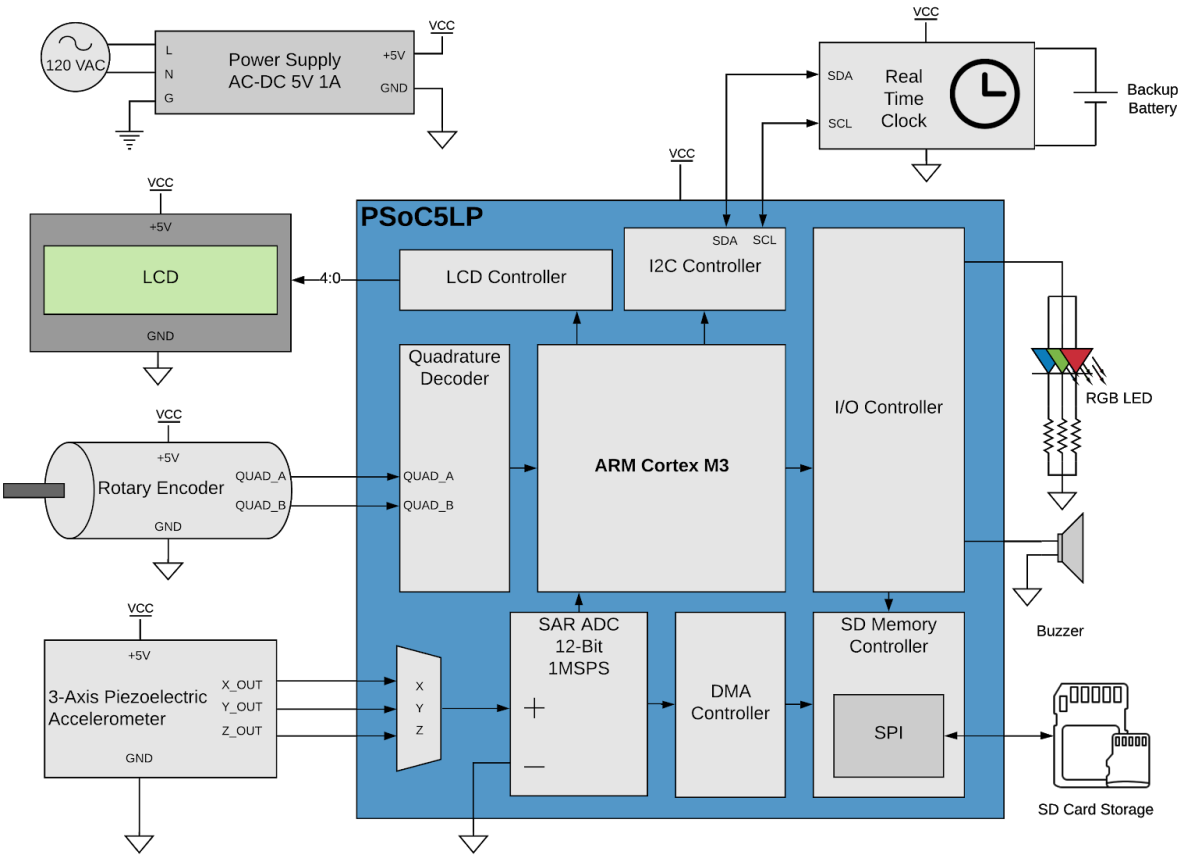


Figure STS.8: *STS Data Acquisition System Block Diagram Rev. 2 (5/29/2020)*

The data acquisition system is built around the PSoC5LP. This SoC has a 32-Bit ARM Cortex M3 running at 48 MHz coupled to FPGA-like configurable digital logic and analog signal processing blocks. The most notable part of the design is along the bottom of the figure. This is the high-speed data recording signal and information chain which records acceleration data and formats it to the SD card.

Data Recording

High Speed Sampling Process

During a test, the accelerometer will convert the acceleration into an analog voltage for all 3 axes. The PSoC has 2 SAR ADCs so the signal must be multiplexed access within the SoC. Only one of the ADCs is used to reduce the overhead associated with managing multiple ADCs and eliminates the concern of keeping them synchronized.

Due to the high sample rate of the ADC and criticality of timing, it is not feasible to use the processor to read, save, and format the data into memory. Instead, direct memory access or DMA, is used to sample the ADC and store the information in a ping-pong buffer. Once a buffer is full, the DMA controller switches buffers and triggers an interrupt for the processor to copy the data from the buffer to the SD card in the raw binary format. After the time critical portion of the test is complete, the processor will then go back to the raw binary data and reformat it as a CSV file for post processing.

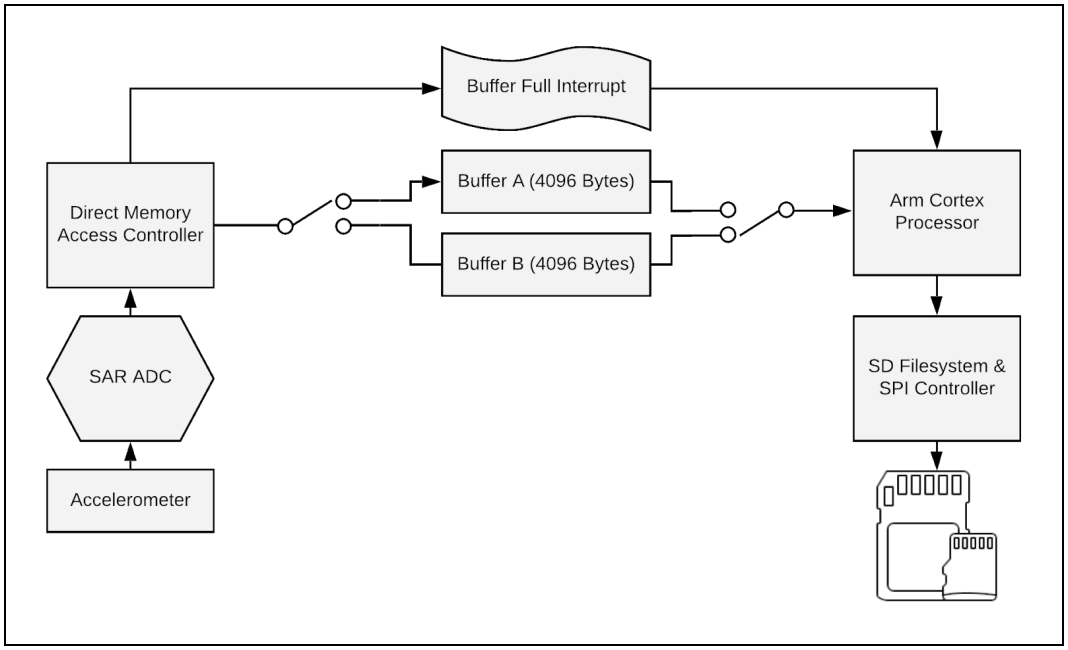


Figure STS.9: *STS Data Sampling w/ Ping-Pong Buffers Flowchart Rev. 1 (6/7/2020)*

Sampling, Memory and Read/Write Speeds

- 3 Accelerometer Channels (XYZ Axis) - 12-Bits SAR ADC (Stored as 16-Bit Integer)
- +/- 6000 G => (12000 G / 2^12 Bits) = 2.93 G / LSB
- ADC Conversions @ 666 KHz Sample Rate / 3 Channels = 222 KHz / Channel
- (666 KHz * 2 Bytes / Sample) = 1.332 MBytes / second
- SD Card via SPI with 25 MHz Clock Speed Writes at 3.125 MBytes / second
- Data Rate Margin (1 - (1.332 MBytes / second) (3.125 MBytes / second)) = 57.8% Margin
- (1.332 MBytes / second)(1.020 seconds) = 1.358 MBytes per Test

Data Storage Method

To record testing data, we decided to use a standard SD card interface as this made the design stand-alone and reduced system complexity while still meeting functionality and usability requirements. The interface to the SD card is done through a SPI interface running at 25MHz bit-clock rate. This will allow for data recording upto 3.125MBytes / second, while the data is generated at ~1.3MBytes / second.

To interface with the SD card and SPI interface we needed a filesystem and formatting method. The chosen system was FAT32 formatting to allow for modern sizes of SD cards (> 2GB). Normally, this library would need to be written down to the low-level to implement file handling and read / write communications to the SD memory. During development, it was discovered that there is an implementation for PSoC5LP already that we could leverage to implement data recording and a full filesystem without an OS. Using these libraries, we were able to get the SD card working and ran a test harness provided with the library to verify SD card setup and performance.

This library however does not support DMA like we originally intended and rather uses the CPU to handle read / write. This was an issue since we can use DMA to load data from the ADC into the SRAM of the microcontroller to buffer it and set and interrupt for the CPU to load data. The CPU can handle 32-bits at a time on the data bus and will run at 48MHz so with buffering from the ADC to SRAM, the SD write dependent on the CPU will be satisfactory.

Timestamping Data

During the development of testing for the SD card interface, we found that the system needed a clock to timestamp the data and set the files write times appropriately. To accomplish this, we needed to add an RTC with battery backup.

We chose to use the DS1307 RTC module since it is widely used and can maintain time upto 10 years on battery. This module communicates using I2C to the PSoC. From the HHC we already have an I2C library and this made implementation of the module fairly quick and smooth.

The RTC has been set and is now working smoothly with the SD card to timestamp files and metadata on the file system about file write and creation times.

Firmware Architecture

- main
  - Real-Time Clock Driver
    - I2C Driver
      - I2C Master
  - emFile System
    - FAT32
    - SD Card Driver
      - SPI Controller
  - Matrix Math Library
  - LCD Driver
  - Quadrature Decoder
  - Component APIs
  - PSoC Timers & Interrupts
- config
  - Rotary Encoder Parameters
  - Accelerometer Calibration Matrix
  - Physical Parameters

Data Post-Processing

After the test is complete, the SD card contains the tri-axial time-series acceleration data in a CSV file. This data must be processed into a displayable format and the SRS spectrum must be calculated. The post-processing software was written using MATLAB and the built-in UI builder. MATLAB was the framework of choice since there is already a vast library of data handling tools and the UI builder supports compilation of a stand-alone executable.

SRS Generation Routine

During development of the subsystem requirements, we wrote software to generate the shock-response spectrum from a generated set of time-series acceleration data. In the post-processing scripts, we were able to leverage this work as the original simulation used time-series data. This was ported to load data from a CSV file instead of generating a waveform.

UI Design

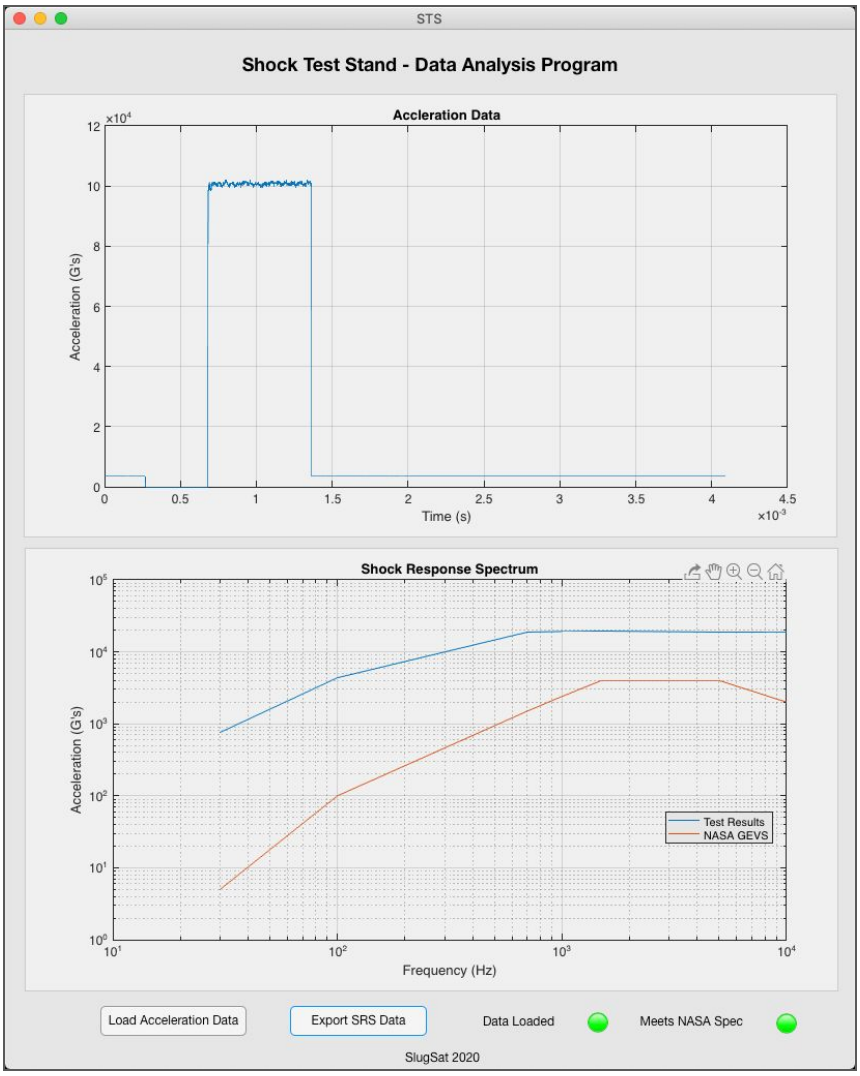


Figure STS.10: *Screenshot of STS Data-Analysis Program (Rev. 2 6-4-2020)*

The UI in STS.10 was built using the MATLAB UI builder. The layout was meant to be simple and include only what the user needs. The top chart displays acceleration vs time and the lower display shows the SRS from the test overlaid with the NASA standard. To use the interface the user simply needs to click “Load Acceleration Data” and select the desired CSV file of test data from the DAQ.

Automatic Test Result Check

Upon generation of the SRS, the software will automatically check if the test results meet the NASA GEVS spec. This is done by searching through the generated SRS data and comparing it to the 7 points outlined by STS_001. This quickly lets a user know if the test was valid and passed. The addition of the feature allows an unfamiliar user the ability to run tests using the stand on their subsystems.

Summary of Data Acquisition System

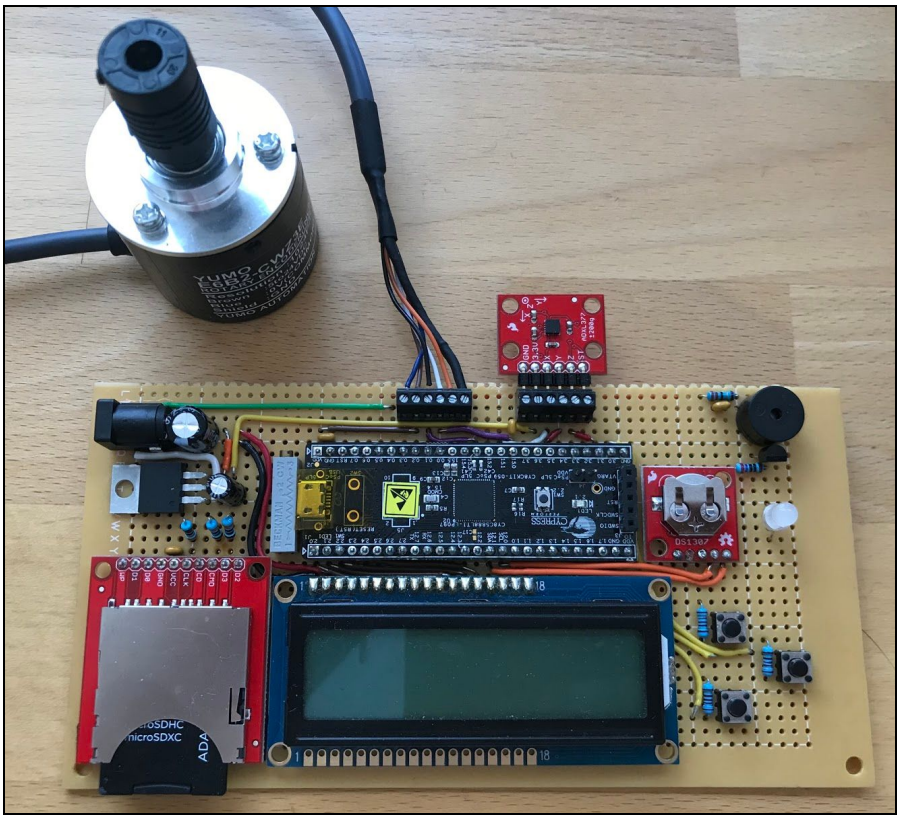


Figure STS.11: *Finished STS Electronics with SD Memory Installed (Rev. 1 6-2-2020)*

So far the electronics have been prototyped and built out on a perfboard as seen in STS.11. This was used to develop the firmware and testing process. These electronics are functional but need to be tested and should likely be turned into a proper PCB with an enclosure.

Standard Operating Procedure

Preparing the Test Article

1. Weigh test article
2. Perform functionality test of all features of test article and photograph test article
3. Calculate lift angle necessary to generate SRS for given test article mass

Setup & Calibration

4. Mount and secure test article to STS
5. Enable data acquisition system
6. Ensure area in and around STS is clear on non-operator personnel

Running the Test

7. Lift proof mass to the required angle as shown on DAQ readout
8. Release proof mass
9. Wait for data recording to finish

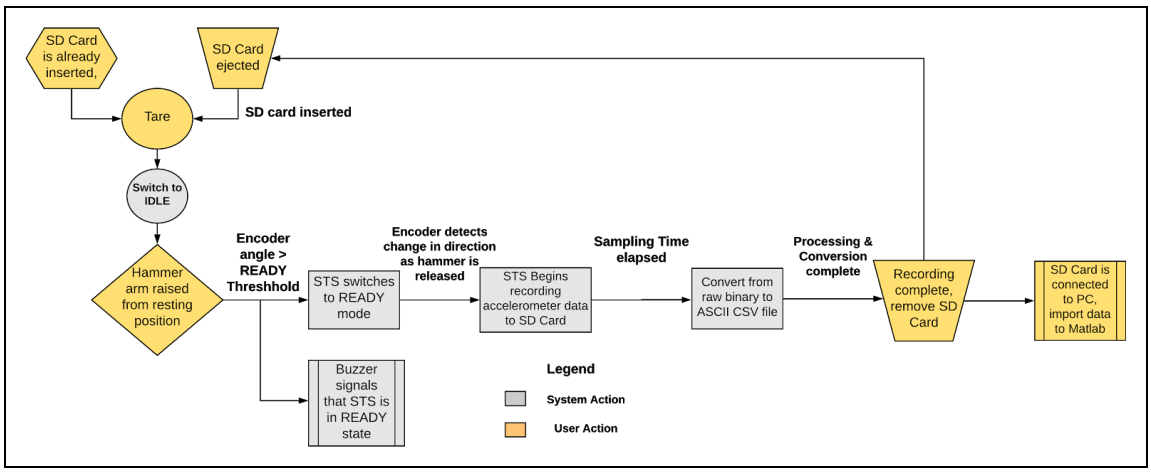


Figure STS.12: Block Diagram of STS standard operation procedure Rev 2.0 6/4/20

Post-Processing & Analysis

10. Load data into post processing software
11. Analyze and compare data to NASA GEVS spec
12. Perform functionality test of all features of test article and photograph test article; compare to data gathered in step 2
13. Combine photos, functionality note, and SRS test data into report for engineering use

Future Work

State of the Deliverable

<table>
<tr><th>Requirement ID</th><th>Design Requirement</th><th>Reference</th><th>Compliance</th></tr>
<tr><td>STS_001</td><td>STS shall deliver a shock to the test mass up to 4.00kg with the following shock response spectrum (SRS) characteristics. All SRS values assume Q = 10.
<table>
<tr><th>SRS Fundamental Frequency</th><th>Acceleration</th></tr>
<tr><td>30Hz</td><td>>= 5g</td></tr>
<tr><td>100Hz</td><td>>= 100g</td></tr>
<tr><td>700Hz</td><td>>= 1500g</td></tr>
<tr><td>1000Hz</td><td>>= 2400g</td></tr>
<tr><td>1500Hz</td><td>>= 4000g</td></tr>
<tr><td>5000Hz</td><td>>= 4000g</td></tr>
<tr><td>10000Hz</td><td>>= 2000g</td></tr>
</table></td><td>NASA GEVS Standard (Document Number GSFC-STD-7000 Rev. A) & CubeSat Design Specification Rev. 13</td><td>In Progress</td></tr>
<tr><td>STS_002</td><td>STS shall provide mounting such that a cubesat upto 3U can be mounted and shock delivered in the X+,X-,Y+,Y-,Z+,Z- directions.</td><td>Shock Delivered in All Directions</td><td>In Progress</td></tr>
<tr><td>STS_003</td><td>STS shall fit in an envelope no larger than 32” x 76” x 76” when not in use.</td><td>US Minimum Door Opening</td><td>Compliant</td></tr>
<tr><td>STS_004</td><td>STS Data Acquisition system shall be powered by NEMA 5-15P a.k.a Type B plug</td><td>Compatible with standard US wall outlet</td><td>Compliant</td></tr>
<tr><td>STS_005</td><td>STS shall have a data acquisition system that can record acceleration data such that the NASA GEVS shock response spectrum can be calculated</td><td>Allow collection and transfer of data to computer for processing</td><td>Compliant</td></tr>
</table>

Table STS.7: STS design requirements with compliancy status as of Spring 2020 Rev. 1 (6/4/20)

State of the Subsystems

Data Acquisition System

Verifying Data Recording Rate / Firmware
The data acquisition system firmware and hardware has been prototyped and is functional. However, the data rates and data integrity has not been verified. This is the recommended starting place for the system ADC timing and DMA may need to be adjusted to bring the sampling rate upto full speed. The system and test process is functional so far.

PCB Layout & Fabrication
During the development of this test stand, we did not have access to any methods for making a proper PCB. We built a prototype on a breadboard and then soldered the system into a perfboard, which is seen in the DAQ section. This board needs to be captured and laid out as a PCB after the Data Recording Rate and firmware have been verified.

Piezoelectric Accelerometer Carrier PCB
The selected piezoelectric accelerometer is a SMD device and is a special made unit. Thus, it is not available on a breakout board. To test the prototype DAQ we purchased an ADXL377 on a breakout board which is a MEMS device that is electrically similar to the 834M1. To bring the system up to full spec, the 834M1 will need to be purchased and a breakout / carrier board will need to be made.

Shielded Wiring Harness for Accelerometer
In addition to the breakout board, a wiring harness will need to be built for the accelerometer. As stated previously, this is an analog device and thus signal integrity is important from the output of the transducer to the ADC. It is recommended that this wiring harness be made with shielding and be kept as short as possible.

Shock Delivery Mechanism

Shock Delivery Mechanism FEA & Analysis
The simulation we constructed can be used to determine the force needed to deflect the plate a specified distance. While this can be used to back calculate the size of the impact hammer, more in depth analysis of the system needs to be done, namely FEA analysis on the deflection plate itself. As the impulse is currently approximated as a point load on a cantilever beam, a more thoroughly informed understanding of the plate dynamics will be necessary before finalizing the mechanical design of the STS

Experimental Test Setup to Verify & Refine Simulation
In order to determine the ideal size of the impact hammer to use, experimental setups of multiple impact hammers and deflection plates will need to be used in addition to the existing simulations

to verify that the simulation has been implemented correctly before adding parametric values to the CAD model.

Finalized CAD Model & Drawings

Once the FEA analysis and experimental setups have determined mechanical parameters that will generate the required acceleration profile, the CAD model will need to be updated with these values. Additionally, both the DAQ and the accelerometer mounting will need to be designed and integrated to the CAD model. Once the CAD model has been finalized, mechanical drawings and a bil of materials will need to be made, and both will need to undergo Critical Design Review (CDR) and be approved for fabrication.

Fabrication of Physical System

After the previously mentioned CDR has been approved, components and materials for fabrication can be purchased. As the frame and arm mounting will be welded, an individual experienced in welding will need to handle assembly of the frame. Mounting hardware and fasteners will need to be purchased as well, likely from McMaster Carr. Once assembled, a formal Standard Operating Procedure (SOP) and Safety Protocol will need to be developed before using the STS.

Conclusion

The shock test stand has been a positive learning experience for the team. This quarter we were able to lay much of the groundwork for the design. Originally we had planned to develop and fabricate the test stand, however the challenging conditions due to COVID extended the time the team spent on finishing the HHC which compressed work on the STS to 4 weeks and shifted the goal to a simulated design of the test stand with the data recording electronics.

Given the rushed timeline, we were still able to hit a majority of the deliverables. In the end, we completed the design calculations and developed a parametric CAD model of the design. Next year, the team will need to validate the calculations and build the mechanical system. For the data acquisition and test-analysis, we were able to develop and build a working prototype that could run a full test-sequence and post-process the data.

The work done this year will make it feasible for next year's team to develop a working shock test stand for validating the subsystems of the craft.

Solar Vector Simulator

Juan Ayala

Introduction

As the Helmholtz coil validates the ACDS algorithm in calculating the magnetic inertial vector, the Solar Vector System (SVS) simulates the solar rays emitted on the cubesat to validate the algorithm to determine the inertial solar vector. To implement the SVS, it requires a solar simulator that can imitate the light rays at Low Earth Orbit (LEO).

To define a solar simulator, we followed the classification provided by the American Society for Testing and Materials (ASTM) in Code E927:

> Solar simulators are classified according to their ability to reproduce a reference spectral irradiance distribution, the uniformity of total irradiance across the test plane, and the stability of total irradiance over time [I]

A solar simulator typically has three major components: light sources, optics, and controls to operate the simulator [I]. For the testing conditions required, we determined that a hybrid solar simulator consisting of Light Emitting Diodes (LEDs) and Quartz Tungsten Halogen (QTH) lamps would be necessary to test the solar vector algorithm and solar sensors. This report discusses the design and implementation of a unit solar simulator prototype that can be multiplied to simulate on any size test area.

Project Documentation, Code, CAD Models, and Schematics: https://github.com/SlugSat/SVS

Design Requirements

ASTM classifies a solar simulator within 3 classes (A, B, or C) for each of the following categories: spectral match to the reference solar irradiance spectrum, spatial non-uniformity over the measured area, and temporal instability of the irradiance.

Spectral Match

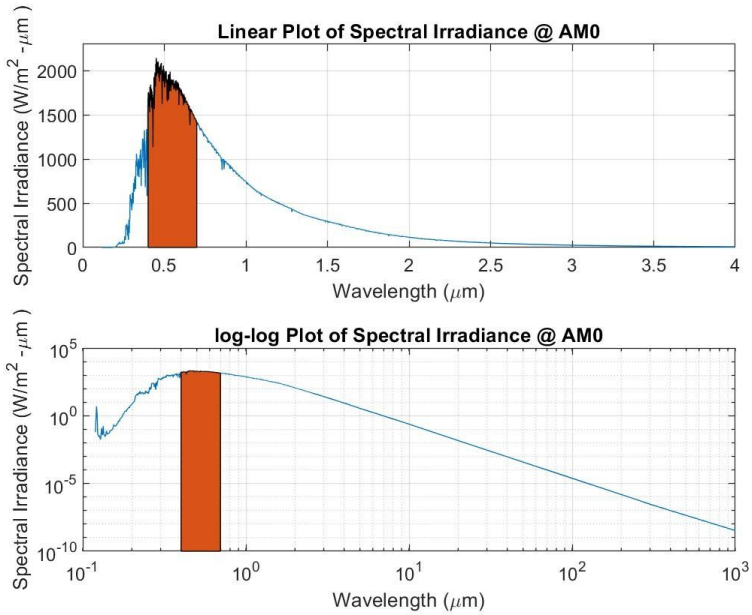


Figure SVS.1: 2000 ASTM Standard Extraterrestrial Spectrum Reference. Orange shade is visible light spectrum Rev. 1 (6/7/20)

ASTM provides a reference spectrum quantifying the solar irradiance per wavelength for different optical paths away from the Earth's atmosphere. For the cubesat in low earth orbit, the air mass coefficient is AM0. To classify the spectral match, the ratio of the irradiance within a specific waveband and the total irradiance from the reference (1366.1 $W*m^{-2}$) is compared to the following table.

Table SVS.1 : ASTM Spectral Match Classification Comparison Chart Rev. 1 (6/7/20)

Wavelength Interval [nm]	%
400-500	18.4
500-600	19.9
600-700	18.4
700-800	14.9
800-900	12.5
900-1100	15.9

The maximum ratio between the solar simulator spectrum irradiance and table SVS.1 must fall within a range to be classified as shown within table SVS.2.

Table SVS.2: ASTM Spectral Match Classification Rev. 1 (6/7/20)

Class	Percentage Range of Comparison Chart
A	0.75-1.25
B	0.6-1.4
C	0.4-2.0

Spatial Non-uniformity

The test area is divided into a finite amount (at least 36) grid and the irradiance is measured within each divided area. The max irradiance and minimum irradiance found characterize the spatial non-uniformity. The spatial non-uniformity is calculated as shown in Equation SVS.1.

$$Non - Uniformity(\%) = \frac{E_{max} + E_{min}}{E_{max} - E_{min}} * 100$$

Equation SVS.1 - Equation for Non-uniformity

Table SVS.3: ASTM Non-uniformity Classification Rev. 1 (6/7/20)

Class	Non-Uniformity (%)
A	2
B	5
C	10

Temporal Instability

Temporal instability is the change of irradiance on the test area within a finite amount of time after steady state. The procedure to find temporal instability is by measuring the max and minimum irradiance found 1-2 seconds after steady state. The equation is the same as Equation SVS.1 but values are found within a set amount of time as opposed to divided areas.

Table SVS.4: ASTM Temporal Instability Classification Rev. 1 (6/7/20)

Class	Temporal Instability (%)
A	2
B	5
C	10

Performance Requirements

With each of the previous categories considered, the following performance requirements are set.

Requirement ID	Requirement
SVS_001	A Solar Simulator Panel shall be capable of illuminating an area of 211 cm^2
SVS_002	Irradiance Non-Uniformity shall be less than 10% in ASTM standards
SVS_003	Solar Simulator shall emit a spectral match least class C in ASTM standards
SVS_004	Solar Vector Simulator shall change intensity simulating the change in distance from sun
SVS_005	Solar Simulator shall emit a spectral range between 400-1100 nm
SVS_006	Solar Simulator shall illuminate cubesat with incident rays less than 20 degrees
SVS_007	Solar Vector Simulator shall interface with user to control solar environment
SVS_008	Solar Vector Simulator shall emit a temporal instability less than 10%

Table SVS.5: SVS Performance Requirements Rev. 3 (6/7/20)

SVS_*001:*

The 2U cubesat has a face area of 211cm^2 where a sun sensor is placed. The solar simulator must be able to illuminate this amount of area.

SVS_*002:*

The irradiance non-uniformity with less than 10% places the simulator within ASTM classification of class C.

SVS_*003:*

The spectral match to the solar rays at LEO will provide an accurate response from what the cubesat will experience. The spectrum will be matched to the ASTM Solar reference shown in figure SVS.

SVS_*004:*
The cubesat will be experiencing different levels of irradiances during orbit. Being able to change the radiant flux to correlate with the position of orbit will validate the solar vector algorithm in all cases.

SVS_*005:*
According to the solar irradiance spectrum, the percentage of irradiance past the wavelength of 1100 is 25.67%. However, the wavelength response of most solar sensors like the repurpose spectrolab prime triple junction do not exceed past 1200nm.[II]

SVS_*006:*
Photovoltaic devices respond to the incidence of light rays and the light rays on the atmosphere are collimated due to the very far source of light that is the sun.

SVS_*007:*
The solar simulator needs to change radiant flux to simulate the different irradiance received by the cubesat within orbit. To alter the irradiance the user must have a method of changing the radiant flux.

SVS_*008:*
The solar simulator cannot be flickering at a high rate during simulation as the instantaneous readings of the solar sensor on the cubesat will influence the ACS algorithm. With a temporal instability less than 10%, the sensors will not detect a large change to contaminate the data.

Prior Art

Previous versions of solar simulators have used incandescent or gas discharge lamps as light sources. Incandescent and gas discharge lamps are capable of emitting high radiant flux and are close in proximity to the solar spectrum. However, the implementation of these light sources heavily rely on the use of filters and optics to produce and control the irradiance. The short life span and high power demand of these light sources required frequent maintenance Most solar simulators did not use LEDs until the development of high power LEDs.

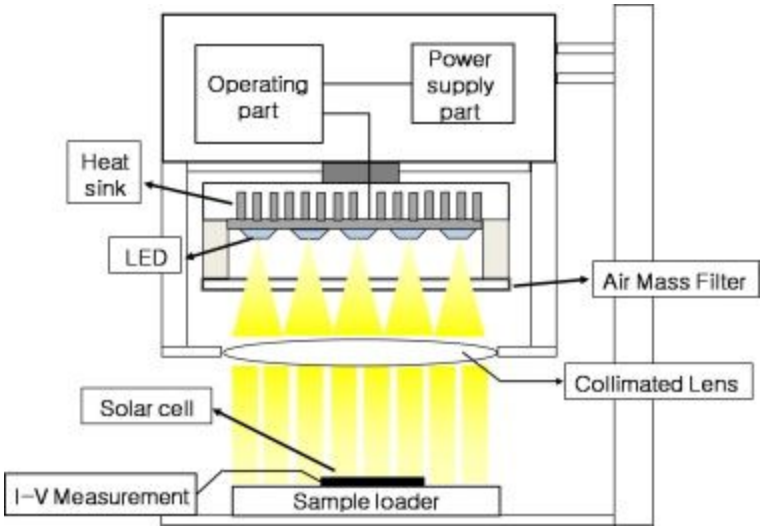


Figure SVS. 2: Typical LED Solar Simulator Rev. 1 (6/7/20)
Image Provided by Jang, Sun Ho

Figure SVS.1 displays a conventional LED simulator. The LED simulator has high power LEDs attached to a large heatsink for cooling and has a collimated lens to collimate the rays to the test area.

In 2012, researchers at University of Illinois at Urbana-Champaign created an LED simulator that emits the AM1.5 solar spectrum with Class C uniformity [D.5]. The spectral range of the solid state solar simulator was designed within the UV and visible light spectrum only.

The design of the solid state simulator does attempt to match the solar spectral irradiance but rather match the relative spectral irradiance. In addition, the solar simulator does not include the infrared spectrum which is about 30% of the total spectral irradiance.

Figure SVS.3: Solid State Solar Simulator used only LEDs. The colors were cool and warm white, UV, cyan, and green
Images Provided by: A.M. Bazzi Rev. 1 (6/7/20){D.5]

Proposed Technical Solution

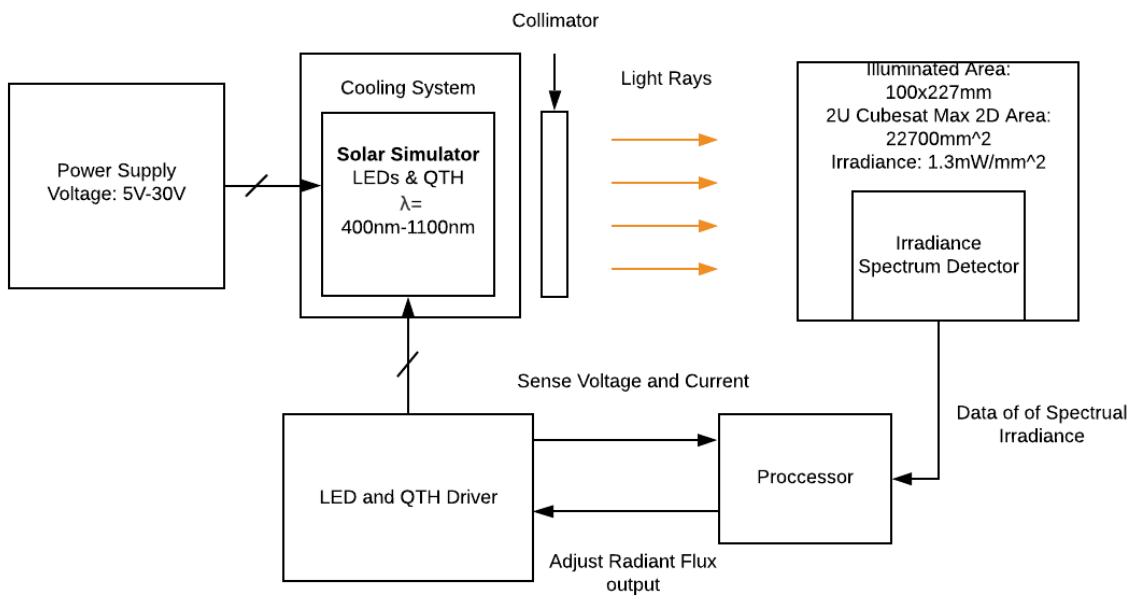


Figure SVS.4: Solar Vector Simulator System Diagram Rev. 4 (6/7/20)

The Solar Vector Simulator is a closed loop system that adjusts the radiant flux output of the solar simulator based on a detector at the test area. For the Solar simulator, the light source is a hybrid of LEDs and QTH lamps that radiate a light spectrum between 400-1100nm. The solar simulator consists of different color LEDs to create the UV and visible spectrum while the QTH lamps create the IR spectrum onto the test area. Due to the different light sources being used, they are positioned relative to each other to produce a high uniform irradiance on the test area. This is further explained in the Solar Simulator Panel section.

The LED and QTH driver adjust the radiant flux of the solar simulator by controlling the sink current. The radiant flux of an LED is nearly proportional to the current. This allows the intensity to change and simulate the dynamic irradiance around orbit. There is a current source for each color and lamp string to allow calibration for the output spectral irradiance.

The power supply and distribution will consist of different voltages based on the string length of each light source spectrum. This is implemented with boost converters.

Light Sources: LED and QTH Spectrum

Relevant Design Requirements

In the decision of the light sources, the relevant requirements to this section are as follows. These requirements set the irradiance spectrum the light sources need to create.

- SVS_003
- SVS_005

Theory and Relevant Equations

<u>Conversion from Photometry to Radiometry Units</u>

Radiometry and photometry are both the detection and measurement of optical light but there are important distinctions to make. Although both study optical electromagnetic waves, photometry is a subset of radiometry that focuses on the human response to visible light. Each science has their own measurements and units. For example, radiometry describes the emitted power of light as radiant flux with units in watts while in photometry it is luminous flux with units in lumens.

	Radiometric		Photometric	
Quantity	**Symbol**	**Units**	**Symbol**	**Units**
Wavelength	λ	nanometer (nm)	λ	nanometer (nm)
Radiant & luminous energy	Q	watt-seconds (W-s)	Q_v	lumen-seconds (lm-s)
Radiant & luminous energy density	U	watt-seconds/m^3 (W-s/m^3)	U_v	lumen-seconds/m^3 (lm-s/m^3)
Radiant & luminous flux (power)	Φ	watts (W)	Φ_v	lumens (lm)
Irradiance & illuminance	E	watts/cm^2 (W/cm^2) or watts/m^2 (W/m^2)	E_v	lux (lx; lm/m^2) or footcandle (fc; lm/ft^2)
Radiance & luminance	L	watts/m^2/steradian (W/m^2/sr)	L_l	lumens/m^2/steradians (lm/m^2/sr)
Radiant & luminous intensity	I	watts/steradian (W/sr)	I_v	candela (cd; lm/sr)

Table SVS.6: Radiometric and photometric quantities and units
Source: Illumination Fundamentals[D.6]

When classifying the solar simulator, the characteristics of light are described in radiometry since the solar simulator is not dependent on how the human perceives the light. Radiometric quantities are noted with subscript e while photometric are noted with subscript v. However, datasheets for LEDs and lamps describe information within photometry. For that reason, we need to convert photometry units into radiometry. To convert from photometry units to radiometry, we use Equation SVS.2.

$$X_v = K_m \int_0^\infty X_e(\lambda) * V(\lambda) d\lambda \quad K_m = 683\ lm/W$$

Equation SVS.2: Radiometric to Photometric units

X is a placeholder for the quantity for conversion and λ is the wavelength. Any function using wavelength is known as the spectral distribution. The subscript *v* is for a photometric unit and e is for a radiometric unit. K_m is a constant.The function V(λ) is known as the photopic spectral luminous efficiency function which contains the magnitude of impact to a certain wavelength that is based on the eye sensitivity to the wavelength. The luminous efficiency function is based on the data set found from experimentation. This data set is used throughout this paper [D.7].

<u>LED Irradiance Model</u>

The chosen LEDs need a net irradiance spectrum that matches the ASTM irradiance reference. This section describes the LED irradiance model used to simulate the net irradiance within matlab.

The LED has a lambertian distribution pattern in which the intensity depends on the viewing angle (α) [D.8] as shown in Figure SVS.6.

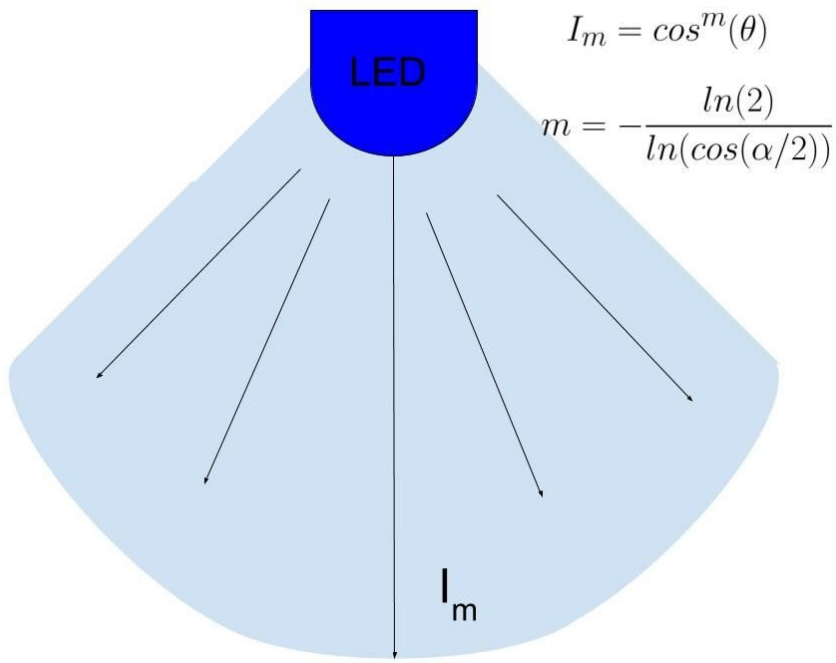


Figure SVS.6: Lambertian Distribution Pattern and formula Rev. 1 (6/7/20)

The irradiance on a surface illuminated by one LED can be determined in Equation SVS.3. $I_{v,o}$ is the max intensity output of the LED. The coordinate x_o,y_o, and z_o is the position of the LED above the surface.

$$\Phi_e(x, y, z) = \frac{I_{e,o}\, {z_o}^{m-1}}{\sqrt{(x_o + x)^2 + (y_o + y)^2 + (z_o)^2)}^{\,m+1}}$$

Equation SVS.3: Irradiance Equation Rev. 1 (6/7/20)

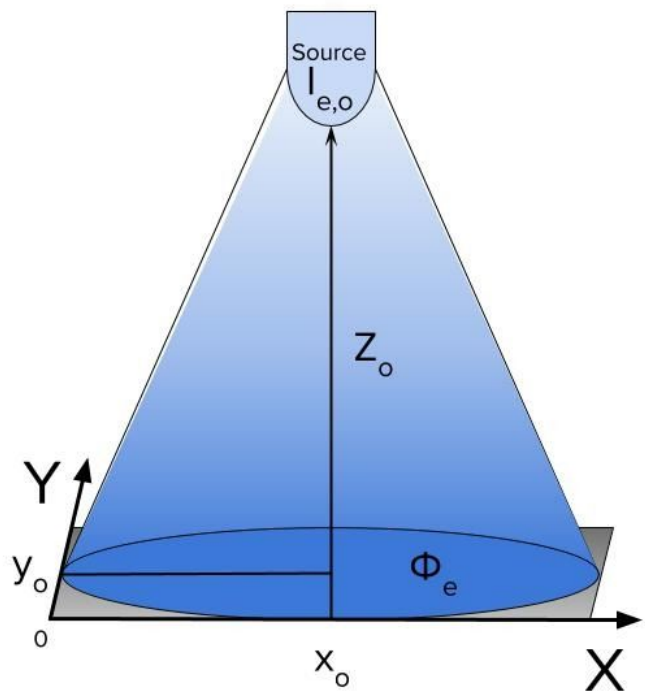


Figure SVS.7: Illustration for calculating irradiance Rev. 1 (6/7/20)

If there are multiple LED sources, then the irradiance at the point is the net sum of all sources. For the total irradiance produced by the light source, Equation SVS.4 is integrated over the test area.

$$\Phi_e = \int\int \frac{I_{e,o}\, {z_o}^{m-1}}{\sqrt{(x_o + x)^2 + (y_o + y)^2 + (z_o)^2)}^{\;m+1}} dxdy$$

Equation SVS.4: Total irradiance equation Rev. 1 (6/7/20)

Calculating Radiant Intensity

Radiant intensity is the radiometric unit of radiant flux per steradian or in other words the amount of energy going in a specific direction from a light source. Datasheets and manufacturers do not always report the radiant intensity so it was often necessary to calculate using the information provided. This section describes how the radiant intensity is calculated.

First, the Luminous intensity is calculated using Equation SVS.5 with the viewing angle (α) and luminous flux (Φ_v) provided in the datasheet.

$$I_v = \frac{\Phi_v}{2\pi(1 - cos(\alpha))}$$

Equation SVS.5: Luminosity Intensity Equation Rev. 1 (6/7/20)

Next, readable ray files from manufacturers are attained. These rayfiles contain datasets that plot the *relative* spectral irradiant intensity of the component. The relative spectral radiant intensity is the normalized spectral radiant intensity so the scale factor needs to be found to get the absolute spectral radiant intensity. The scale factor can be found by manipulating Equation SVS.2.

$$I_v = K_m \int_{\lambda_{min}}^{\lambda_{max}} I_\lambda(\lambda) * V(\lambda) d\lambda$$

$$I_v = K_m \int_{\lambda_{min}}^{\lambda_{max}} k_{sf} I_{relative}(\lambda) * V(\lambda) d\lambda$$

$$k_{sf} = \frac{I_v}{\int_{\lambda_{min}}^{\lambda_{max}} I_{relative}(\lambda) * V(\lambda) d\lambda}$$

Equation SVS.6: Calculation Scale factor for relative radiant intensity Rev. 1 (6/7/20)

Gaussian Estimation of Relative Intensity

Previous methods in finding the absolute spectral intensity need the relative spectral intensity. There are cases that the relative spectral intensity cannot be found for all accessible light sources. In these cases, the spectral intensity is modeled as a Gaussian estimation [D.11].

$$S(\lambda_p, \lambda_{FWHM}, \lambda) = exp\left\{\frac{-2.7725(\lambda - \lambda_p)^2}{\lambda^2{}_{FWHM}}\right\}$$

The wavelength λ_p is the peak wavelength and λ_{FWHM} is the wavelength at which the intensity is at half maximum.

Matlab Spectral Match Simulation

Using the equations outlined in the previous section, LED components can be determined to create a high spectral match to the ASTM solar reference. The calculations and methods were implemented with MATLAB code shown in appendix [C.1] & [C.2].

The design script takes files containing the relative intensity spectrum of chosen LEDs and the photopic spectral luminous efficiency data set as inputs. The programmer can change the amount of LEDs emitting on a surface as long as they are the predefined commercial components. However, the user may add more commercial components. The design script outputs graphs illustrating the irradiance spectrum of each component, the integrated irradiance spectrum, color coded wavebands for ASTM grading, and prints out the spectral match for each waveband.

Light Source Components

Through trial and error, the following 3W LEDs color and QTH in Table SVS.7 are chosen for the unit Solar Simulator prototype. Material list of components are shown in appendix [C.4].

Table SVS.7: Chosen components for color spectrum Rev. 2 (6/7/20)

Color	Amount
Warm White	9
Royal Blue	1
Deep Blue	1
Green	4
Red	3
UV	3
Violet	2
Cyan	3
IR 20W 24 V QTH	2

The irradiance spectrum of the components were simulated over an area of 2500mm^2 (50x50mm), at a distance of 0.118m. A visual representation of the simulation for all light sources are in figure SVS.8. The spectral irradiance is calculated for each wavelength by using the spectral radiant intensity within equation SVS.2 instead of radiant intensity. The spectral irradiance equation is shown in equation SVS.7.

$$\Phi_e(\lambda) = \sum_{k=1}^{N} \int \int \frac{I_k(\lambda)\, {z_o}^{m_k-1}}{\sqrt{(x_o+x)^2+(y_o+y)^2+(z_o)^2)}\;^{m_k+1}} dxdy$$

Equation SVS.7: Spectral Irradiance using spectral intensity Rev. 1 (6/7/20)

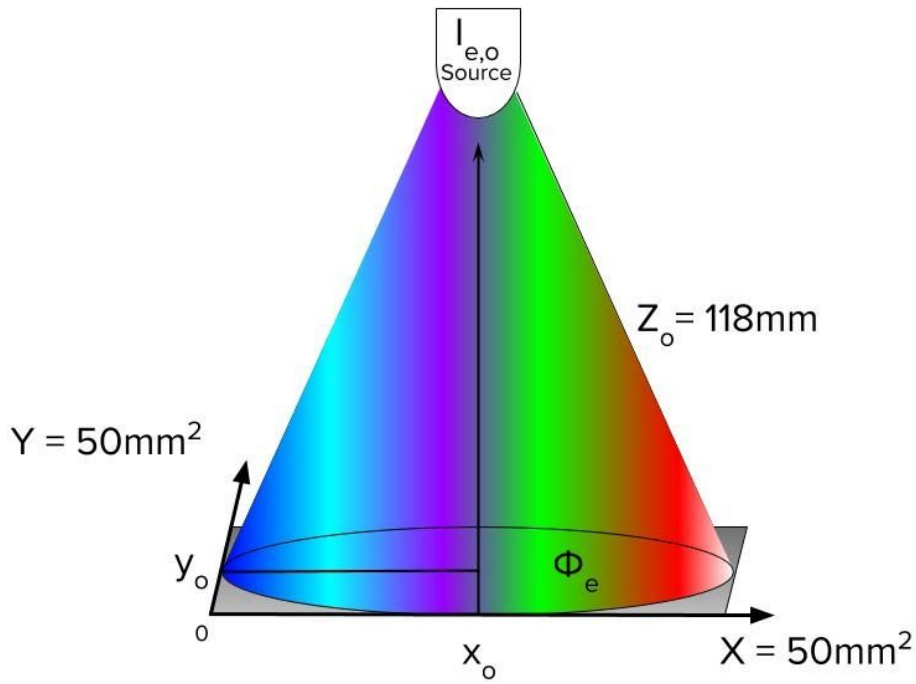


Figure SVS.8: Luminosity Intensity Equation Rev. 1 (6/7/20)

The spectral irradiance for each component is shown in Figure SVS.9 and the summation of each spectral irradiance is shown in Figure SVS.10. Figure SVS.10 clearly shows the spectral irradiance from the components does not exactly match the ASTM solar reference. If it was desired to achieve a near perfect match, more light sources are needed to expand the irradiance spectrum. However for the simulator, the spectral match is within classification of ASTM. Shown in table SVS.8, the spectral match is of Class B.

Waveband (nm)	Spectral Irradiance/ Total Reference Irradiance	Class
400-500	13.4%	A
500-600	13.7%	A
600-700	10.3%	A
700-800	8.3%	B
800-900	7.5%	A
900-1100	14.7%	A

Table SVS.8: Classification of Spectral Match for multiple waveband Rev. 1 (6/7/20)

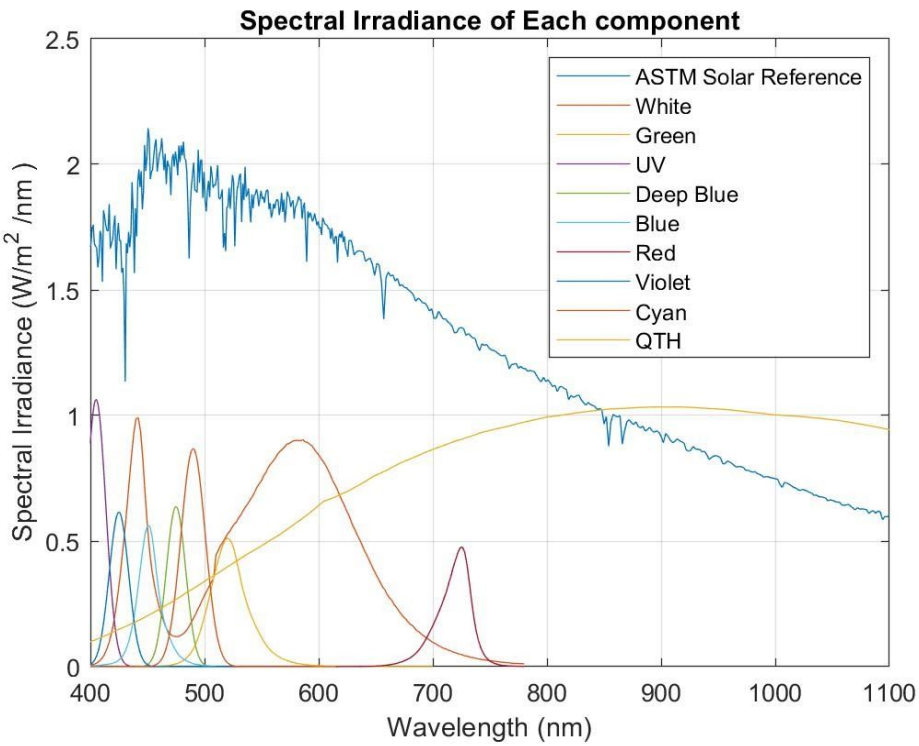


Figure SVS.9: Spectral Irradiance of each component Rev. 2 (5/30/20)

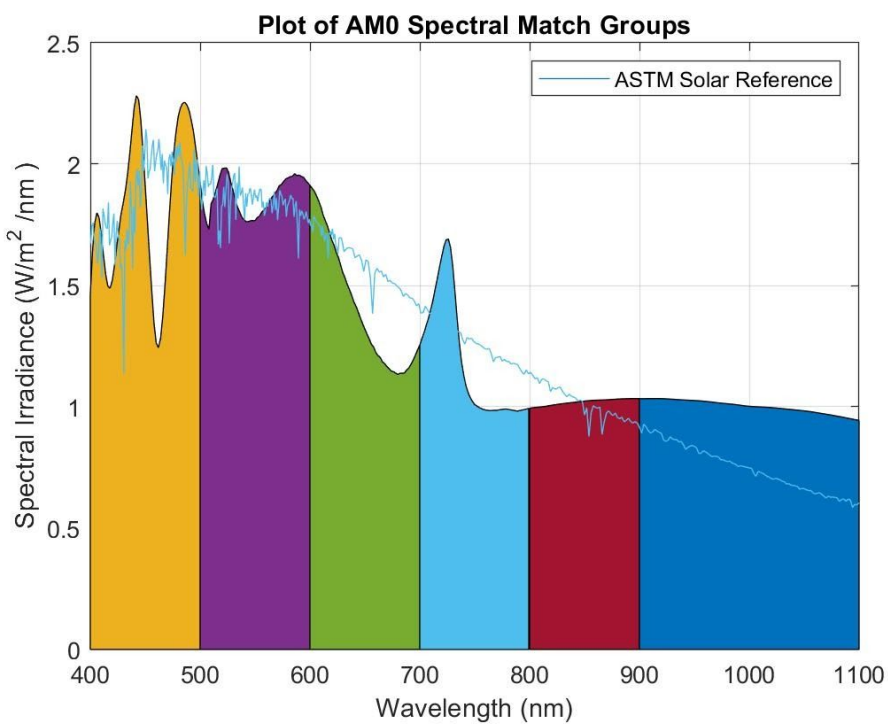


Figure SVS.10: Spectral Irradiance of components Rev. 1 (6/7/20)

Solar Simulator Panel

Relevant Design Requirements

In the decision of the light sources, the relevant requirements to this section are as follows. These requirements set the irradiance uniformity on the test area.

- SVS_002

Light Source Position

The irradiance uniformity is dependent on the optics and position of the light source relative to the test area. For this design, optical components are not considered due to their high expense and time consumption in implementing simulation.

The simulation script for calculating the irradiance distribution on the test area is in appendix[C.2]. The core of the script uses Equation SVS.7 to find all irradiance contributions from each light source. The light sources are positioned on a parallel surface 118mm above the test area. The script produces figures presenting the irradiance distribution, the position of the components, and the non-uniformity value calculated using Equation SVS.1.

Figure SVS.11 displays the determined positions of the LED components and Figure SVS.12 displays the irradiance distribution on a test area of 2500mm^2. The components were designed to be evenly spaced throughout the plane. The non-uniformity of this configuration is 6.5% which is classified as class C. To improve the unfirmority, more components can be used but the price would increase. On the other hand, the non-surfacemountLEDs for the prototype are not small enough to fit close to each other.

The class C non-uniformity satisfies the design requirements.

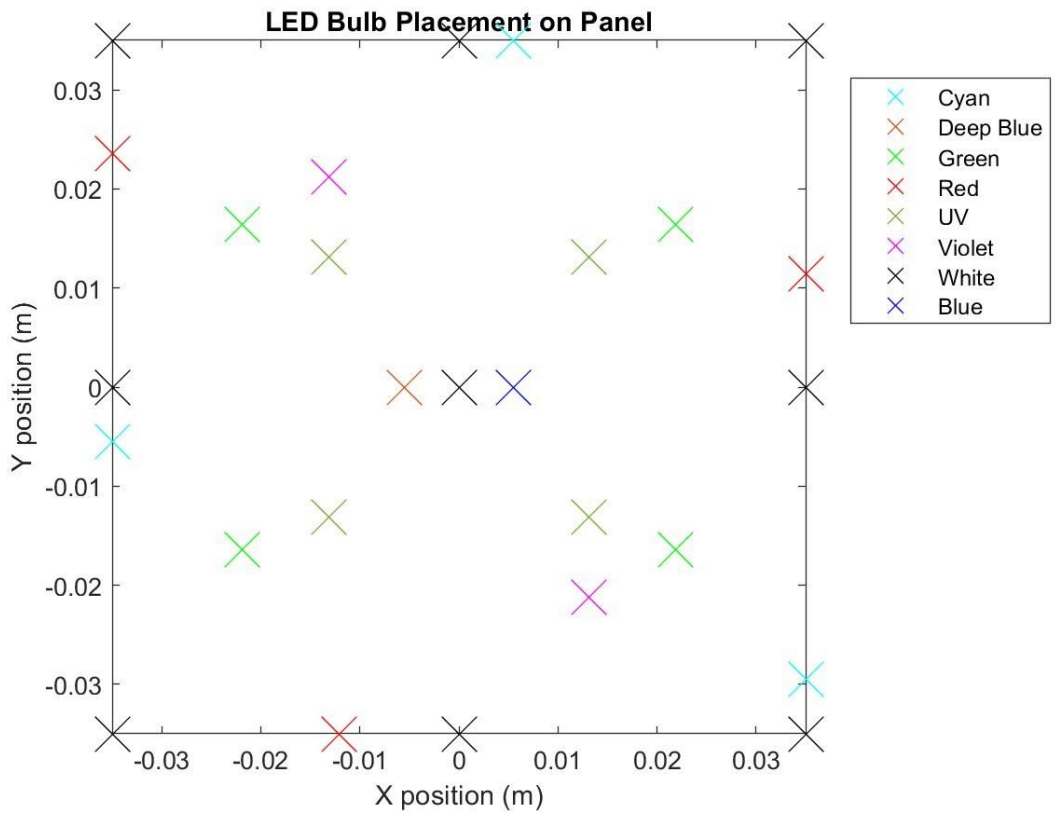


Figure SVS.11: Component XY position in elevated plane of 0.118m Rev. 1 (6/7/20)

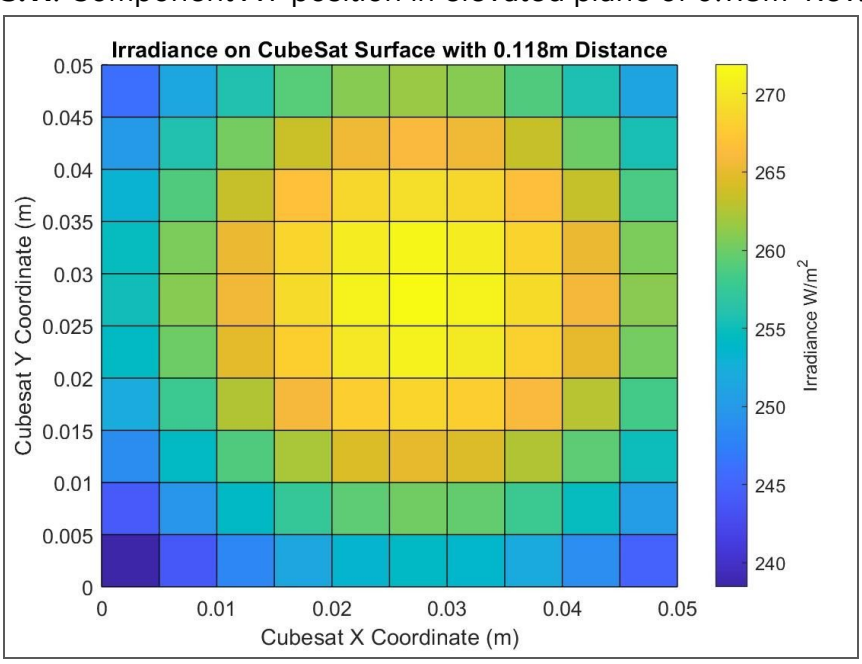


Figure SVS.12: Irradiance distribution on test area surface Rev. 2 (6/7/20)

Panel Housing

The housing to hold the light sources in the same position as the simulation is designed within Solidworks. The housing of the prototype solar simulator is shown in Figure SVS.13. The panel is 3.75x3.75in with a quarter of an inch in depth. It is designed for the LEDs and components to slide into their positions.

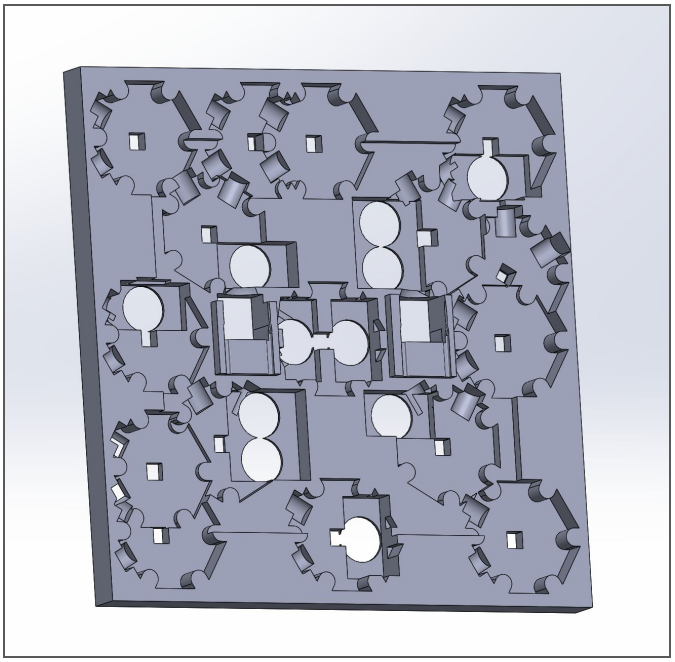


Figure SVS.13: Solar Simulator Prototype Housing Rev. 1 (6/7/20)

Power Distribution and Driver

Relevant Design Requirements

In the design of the power distribution, the relevant requirements to this section are as follows. These requirements set the voltage and current regulation of the power distribution and driver.

- SVS_007
- SVS_008

Figure SVS.14 demonstrates the power distribution and driver system. There are different light source strings for each different color. Due to the different number of light sources within each string, there is a different voltage source provided by a boost converter. The light sources are controlled by the LED driver which implement current sources for each color string. Each color string will be calibrated to emit its respective radiant flux to output a spectral match using a

potentiometer and set PWM signal. After the calibration, the PWM signal can change the overall radiant flux but not the relative spectral distribution.

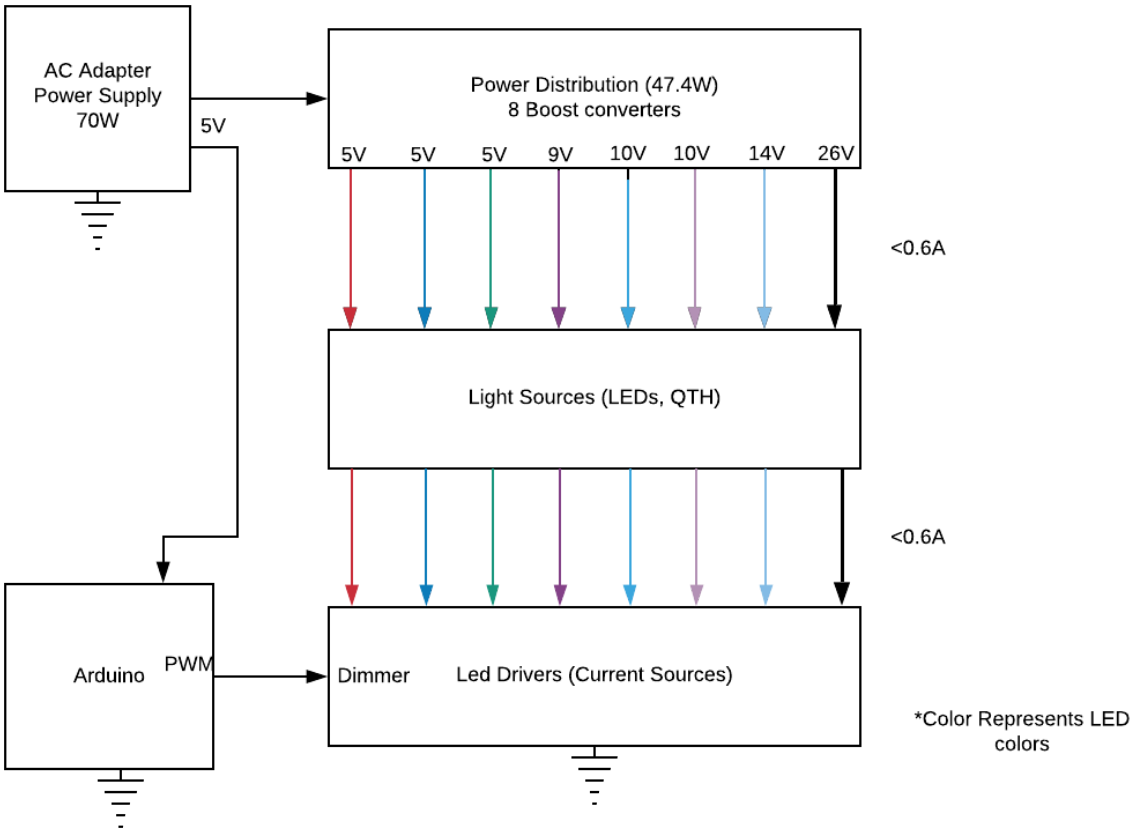


Figure SVS.14: Power supply and driver system diagram Rev. 2 (6/7/20)

Theory and Relevant Equations

Allowable voltage/current ripple

As previously described, the solar simulator needs a temporal instability of less than 10%. The change of temporal instability is caused by the change of irradiance within a set amount of time. The irradiance is proportional to the radiant flux emitted by the light sources. To minimize the change in the radiant flux, the current and voltage need to be stable to a certain degree. This section describes how the maximum voltage and current ripple are determined to have a temporal instability of less than 10%.

To begin, the irradiance in equationSVS. is converted to radiant flux and is equated to the max temporal instability of 10%. Equation SVS. represents the temporal instability when there is a

percentage change from the assigned radiant flux. The equation is used to find the max allowable percentage of change in the radiant flux to stay at most 10% in temporal instability.

$$0.10 = \frac{(k + \Phi_e) - (\Phi_e - k)}{(k + \Phi_e) + (\Phi_e - k)}$$

Equation SVS.8: Temporal Instability equation set for 10% and radiant flux change of k Rev. 1 (6/7/20)

$$k = 0.1\Phi_e$$

Equation SVS.9: Allowable radiant change Rev. 1 (6/7/20)

According to equation SVS.9, the radiant flux can only change ±10% of its original to achieve 10% of temporal instability. From the datasheet, we can use the luminous flux and current curve to find the ripple in current from our bias current which in our application is 350mA.

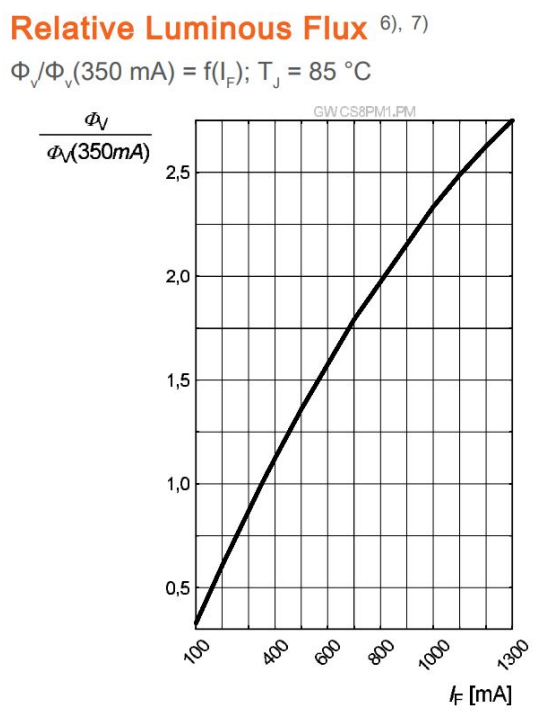


Figure SVS.15: Luminous flux vs current curve of white LED. Image provided by OSRAM [D.10]

The luminous flux is proportional to the radiant flux so the curve is applicable in determining the change in radiant flux. Then the current vs voltage curve within the datasheet can be used to determine the allowable voltage ripple to attain a temporal instability of 10%. If a lower temporal instability is desired, the voltage ripple is less than the prior calculated value.

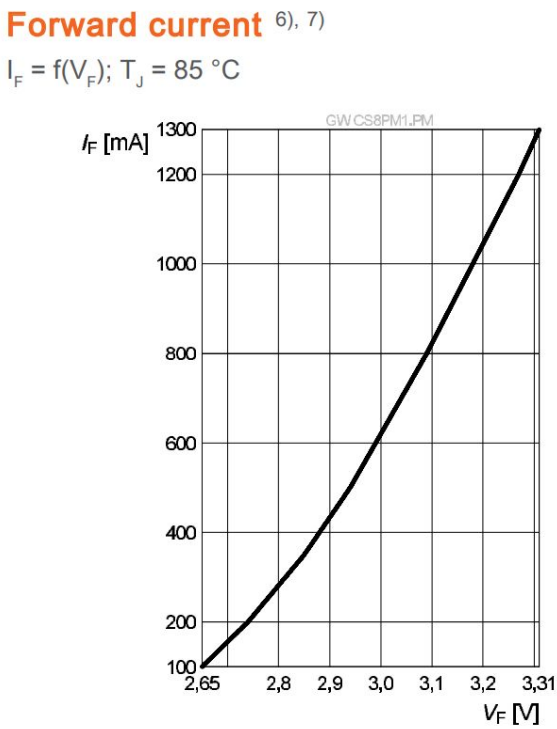


Figure SVS.16: Forward Current flux vs voltage of white LED. Image provided by OSRAM [D.10]

Power Supply

The radiant intensity to simulate the spectral match in matlab had the operational current of 350mA through each LED. Since the solar simulator needs to have flexibility in the output radiant flux, the LEDs are given a max of 600mA as a possible scenario. With 26 LEDs in operation of 3V and max 600mA as well as a QTH in 20W, the power supply must output at least 66.8W. To supply this, an AC adapter with a power output of 70W and 5V is chosen for the prototype.

Power Distribution

Light sources of the same wavelength are in series to minimize the amount of driver circuitry while allowing calibration of the irradiance spectrum. Table SVS.9 shows the source voltage needed to operate the light source strings.

Table SVS.9: Required Voltage Source for LED strings Rev. 1 (6/7/20)

Color	# in Series	Voltage Source
White	9	26
Royal Blue	1	5
Deep Blue	1	5

Green	4	14
Red	3	10
UV	3	12
Violet	2	9
Cyan	3	10
IR	1	24

Since the power supply is 5V and many of the component strings require voltages above that, a boost converter is implemented for the required strings. A boost converter module is selected due to the capability of changing output voltage. The boost converter module is the XL6009 based power module. This module has 5V within its input voltage range and the output is capable of voltages between 3V to 32V with a maximum output voltage of 4A.

Using the method of finding the maximum voltage stated in previous sections, the maximum voltage ripple allowed is 165mV. The XL6009 based power module has a maximum ripple of 50mV which will cause the temporal instability to be less than 10%.

Driver Circuit

Each color component string will have their own driver circuit. Since the radiant intensity is influenced by the current, the driver circuit is a constant current driver. We used an op amp and MOSFET to implement a current source. Figure SVS.17 is a simplified version of one led driver. This allows the current source to be adjusted at the non-inverting input of the opamp.

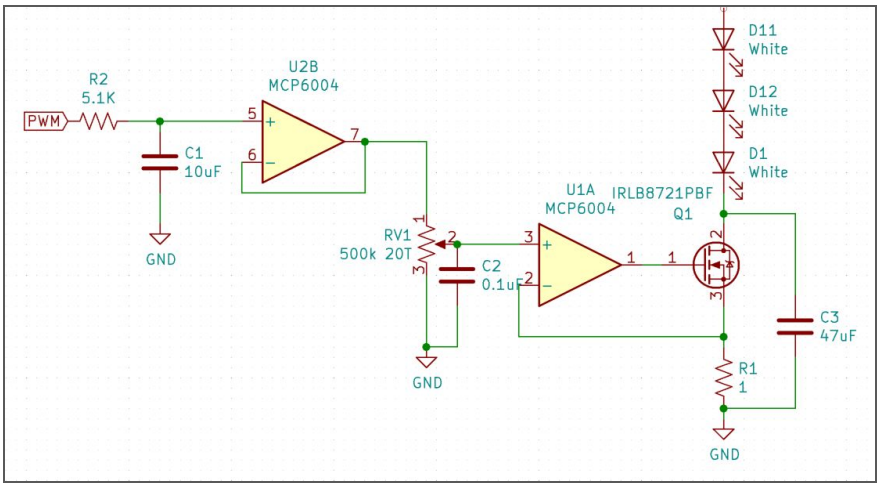


Figure SVS.17: Simplified Driver Circuit Rev. 2 (6/7/20)

The current source is the op amp U1 and the gate Q1. This simple circuit uses negative feedback to control the voltage across the resistor R1. If the voltage is set at the non-inverting input of U1A, the voltage at R1 is the same, setting the current through the gate to be $V_{\text{non-inverting input}}$/R1. If the components are ideal, the current can be predicted if the voltage at the non-inverting input is known. Even with the discrepancies of the components, the current is nearly linear to the voltage in.

The purpose of the potentiometer, RV1, is to calibrate the voltage for the individual LED drivers. This is the interface to adjust the radiant flux of a color to achieve the spectral match.

The RC filter and buffer U2B are signal processing a PWM signal from the microcontroller to an analog signal. The conversion of the PWM signal to an analog is to reduce the flickering of the light sources and remain within the required temporal instability. This portion of the circuit is used by all other current sources. The one PWM source signal makes the brightness of the solar simulator adjust uniformly without changing the relative spectral match.

Solar Simulator Prototype.

The Solar Simulator panel is shown in Figure SVS.18. The housing is manufactured using a 3D printer and the circuits are soldered onto perfboards. There are 4 separate boards connected with stranded wires.

Figure SVS.18: Images of Solar Simulator prototype. Mid & Left: Front Solar Simulator. Right: driver circuits and back of Solar simulator Rev. 2 (6/10/20)

The solar simulator is able to adjust its brightness through a PWM signal from an arduino. The panel can reach a max current flow of 1A through each component but the drivers are capable of demanding more current.

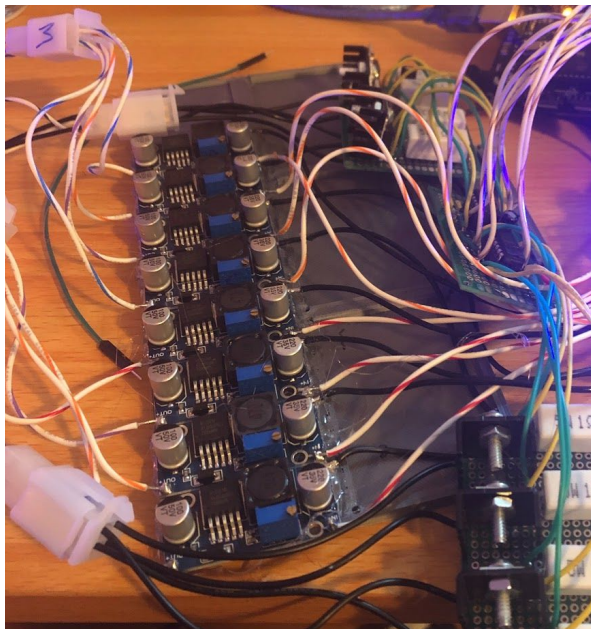


Figure SVS.19: The power distribution circuit using boost converters Rev. 1 (6/7/20)

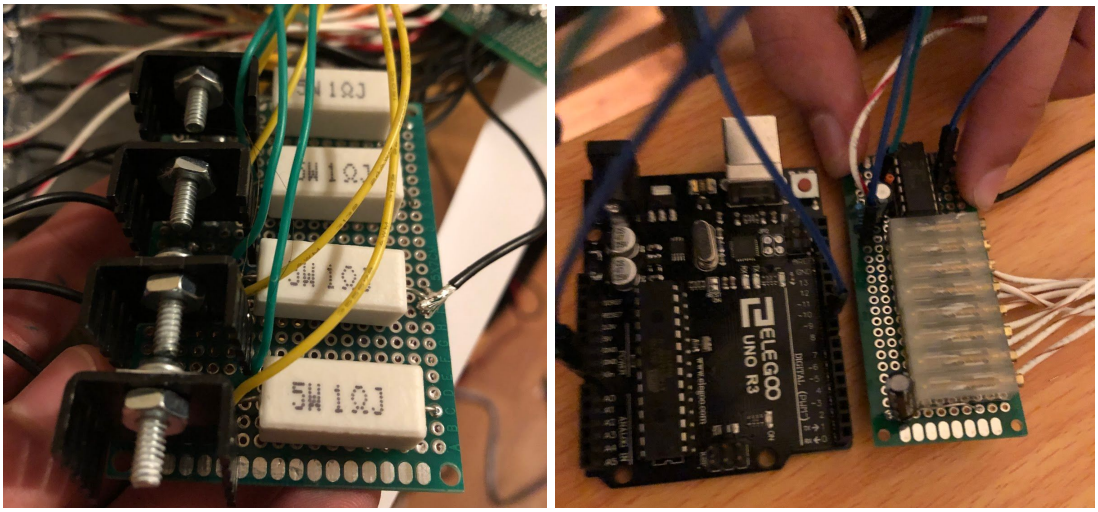


Figure SVS.20: Image on the left is the current source with heat sinks. On the right is the PWM to analog converter and potentiometers Rev. 1 (6/10/20)

Non-Uniformity Experiment

It is difficult to test the solar simulator and compare it to the simulation results without proper equipment. However, it is possible to gain a perspective on the non-uniformity of the panel. The method is to use a camera to take a picture of the illuminated area the same size as the test area. The image is processed and analyzed to find the “intensity” or the camera response to radiant intensity throughout the image. Figure SVS.21 illustrates the experimental design and Figure SVS.22 is an image of the experiment.

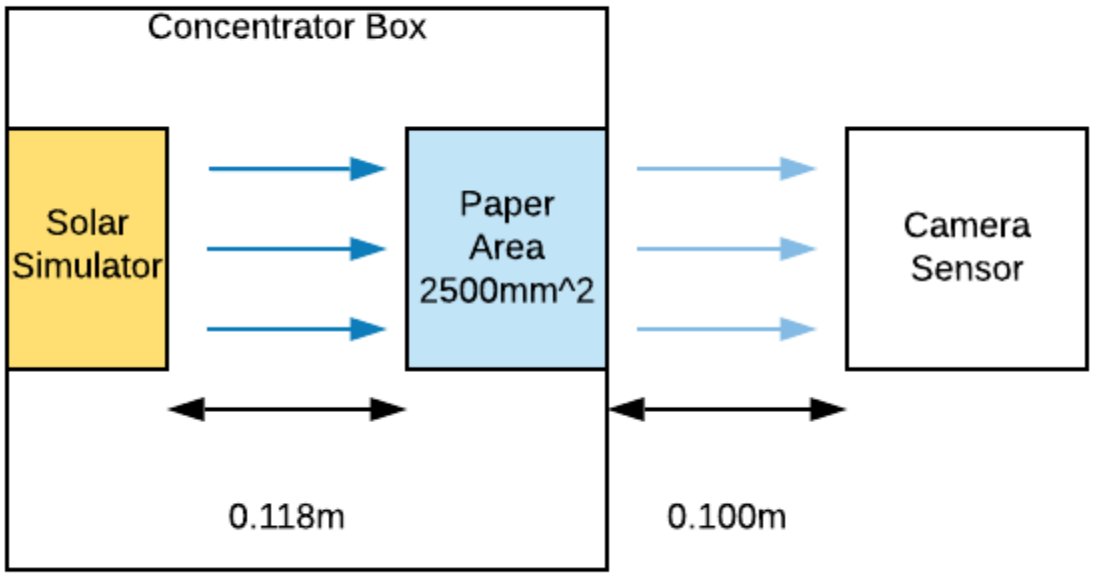


Figure SVS.21: Non-Uniformity Experiment. The solar simulator is within a box and emits light onto a piece of paper. The Camera senses the "intensity" of the image. Rev. 1 (6/10/20)

The solar simulator is held within a box where on the other end is a thin piece of paper for the light to emit on. The piece of paper has a square drawn on it with an area of 2500mm^2 which is the area used within the simulation. Once the image is captured, the image is cropped to only encase the test area. The output of the solar simulator is set up to create the simulated spectral irradiance and non-uniformity. This is by setting up the current through each component as 350mA. The solar simulator is distanced from the sheet of paper by 0.118 as in the simulation. The experiment was held within an enclosed environment so no additional light sources can contribute to the readings from the camera.

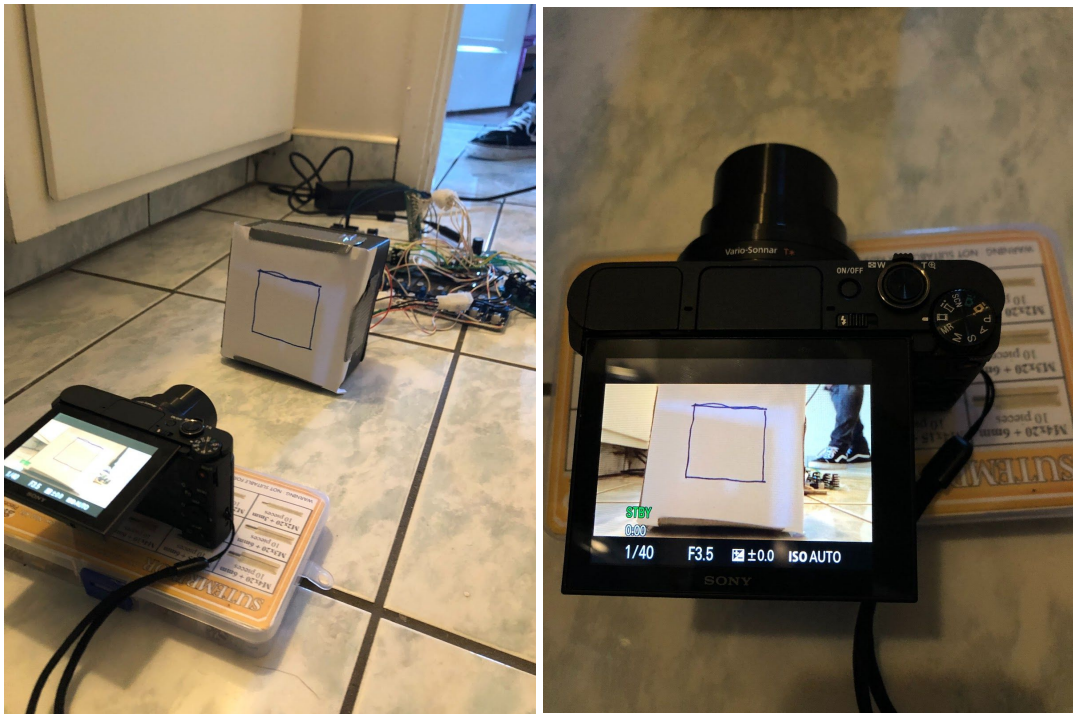


Figure SVS.22: Non-Uniformity Experiment Setup Front view. Rev. 1 (6/10/20)

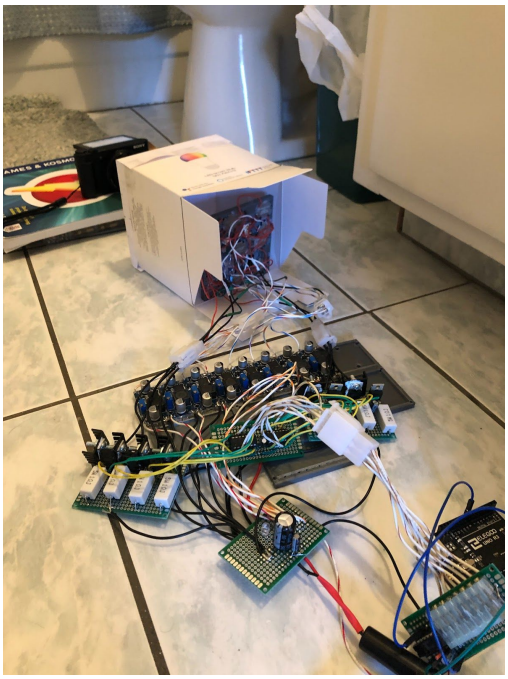


Figure SVS.23: Non-Uniformity Experiment Setup Back view. Rev. 1 (6/10/20)

The image captured is shown in Figure SVS.24. The image is cropped to only include the test area and is processed in matlab using the script in appendix [C.5].

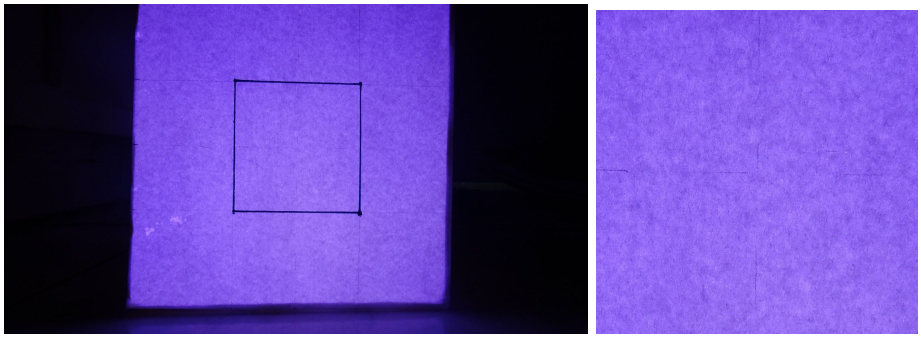


Figure SVS.24: Non-Uniformity Experiment Image Capture at sheet of paper. The image on the right is the cropped image only including the area within the drawn square. Rev. 1 (6/10/20)

The matlab script turns the image into a grayscale matrix where each number represents the shade of each pixel. The shade correlates loosely to the brightness and color differences on the paper surface. Comparing the different shades on the image can give the same comparison of the measured irradiance for non-uniformity.

After averaging the matrix within a 10x10 grid, the difference of “intensity” can be seen in Figure SVS.25. Calculating the non-uniformity uses Equation SVS.1 but uses the “intensity” of the image instead of irradiance. The resulting uniformity is 9.4% which is classified as class C by ASTM standards.

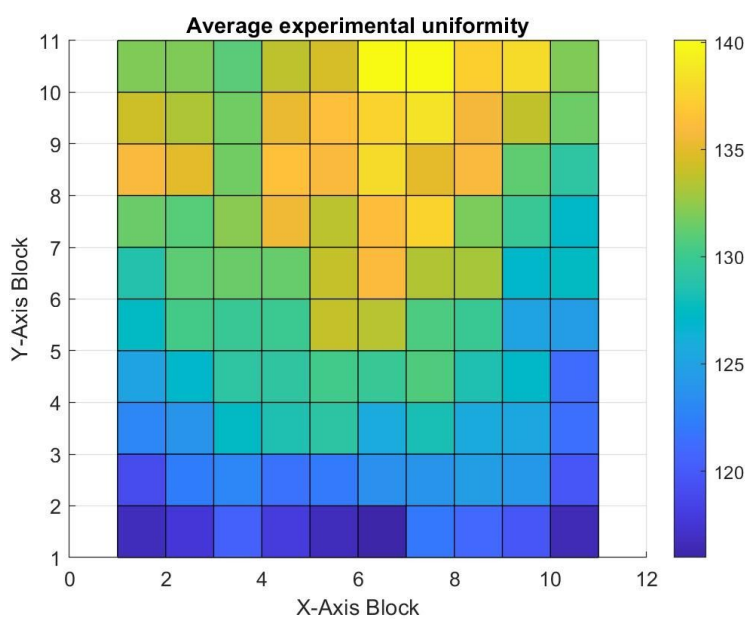


Figure SVS.25: Average Experimental Uniformity Rev. 1 (6/10/20)

Within this experiment, the solar simulator non-uniformity matches with the simulation in that the classification is C. However, the simulated non-uniformity is 6.5% as opposed to the experimental

value of 9.4%. This difference is determined by the testing equipment that is the camera as opposed to a photodiode array or spectrometer.

Future Work

State of Deliverable

Table SVS.10: State of Deliverable table Rev. 1 (6/7/20)

Requirement ID	Requirement	Reference	Compliance
SVS_001	A Solar Simulator Panel shall be capable of illuminating an area of 211 cm^2	Expandable Unit satisfying SVS_002 & SVS_003	Compliant
SVS_002	Irradiance Non-Uniformity shall be less than 10% in ASTM standards	Matlab Simulation comparison with ASTM Solar Reference	Simulated Compliance
SVS_003	Solar Simulator shall emit a spectral match at least class C in ASTM standards	Matlab Simulation comparison with ASTM Solar Reference	Simulated Compliance
SVS_004	Solar Vector Simulator shall change intensity simulating the change in distance from sun	PWM signal drives constant current drivers	Compliant
SVS_005	Solar Simulator shall emit a spectral range between 400-1100 nm	Matlab Simulation comparison with ASTM Solar Reference	Simulated Compliance
SVS_006	Solar Simulator shall illuminate cubesat with incident rays less than 20 degrees	Matlab Ray tracing Simulation	Future Work
SVS_007	Solar Vector Simulator shall	Arduino program	Compliant

	interface with user to control solar environment	can adjust output brightness and potentiometers calibrate	
SVS_008	Solar Vector Simulator shall emit a temporal instability less than 10%	Boost Converters	Compliant

The prototype solar simulator was created but the characteristics of the panel were not measured due to the available resources. To experiment with the prototype solar simulator a spectrometer or radiometer is required.

The prototype is only one unit of the larger solar simulator. If this unit is able to produce the characteristics that were simulated, the prototype can be multiplied together to produce a larger panel that can illuminate a test area of 211cm^2.

<u>Solar Vector Simulator</u>

- Design Collimator lens to collimate light onto surface
- Design PCB for surface mount LEDs to decrease size of prototype
- Implement software to adjust radiant flux according to actual orbit
- Design photodiode array circuit to sense the output uniformity and spectral irradiance to feedback and adjust system

Conclusion

The SVS system is designed to test the solar vector calculated from ACS. This led to the development of the solar simulator prototype. Due to the amount of resources, the prototype could not be adequately measured. However, according to the simulations and design, the panel is of Class B in spectral match, class C in non-uniformity, and class A in temporal instability. The conclusion of the non-uniformity experiment provided more evidence that the non-uniformity is of class C but better equipment should be used. The test area for the solar simulator is 2500mm^2 which is only a portion of a face of the Cubesat. But this unit panel can be multiplied and scaled for the area as the current classifications only impact the specified area and does not influence the area around it much.

Proposed Future Testing Equipment

Vibroacoustic Test Stand

To perform qualification testing on the subcomponents and integrated satellite, vibration testing is required. Vibration testing simulates the mechanical vibration of the satellite during powered ascent on the launch vehicle. The test stand requires an actuator to create the mechanical vibrations.

SlugSat envisions a future project being the construction of a vibroacoustic test stand that meets the following specifications:

Vibroacoustic Test Stand					
Criteria	Indicator	Type	Units	Result/Goal	Measurement Strategy
Vibration Table Accurately and Repeatedly Emulates the Launch Vehicle Conditions for All Axes Of the Craft	ASD @ 20 Hz	Quantitative	ASD (g^2/Hz)	0.026	Accelerometer FFT Data
	ASD @ 20-80 Hz	Quantitative	ASD (g^2/Hz)	+6dB/oct	Accelerometer FFT Data
	ASD @ 80-500 Hz	Quantitative	ASD (g^2/Hz)	0.16	Accelerometer FFT Data
	ASD @ 500-2000 Hz	Quantitative	ASD (g^2/Hz)	-6dB/oct	Accelerometer FFT Data
	ASD @ 2000 Hz	Quantitative	ASD (g^2/Hz)	0.026	Accelerometer FFT Data
	Mounting Interfaces Hold UUT on Table	Qualitative	Components Fit Securely in Mounts	Mounts Hold Components in all 3-Axis Orientations	Physical Test Fit / CAD Model GD&T Analysis

Table TE.1: *Table of Requirements for Vibroacoustic Test Stand Rev. 1 (12/5/19)*

This year's team has not spent any time developing a solution to this but after preliminary research in the fall, we laid out a concept of what a solution could look like in the following figure.

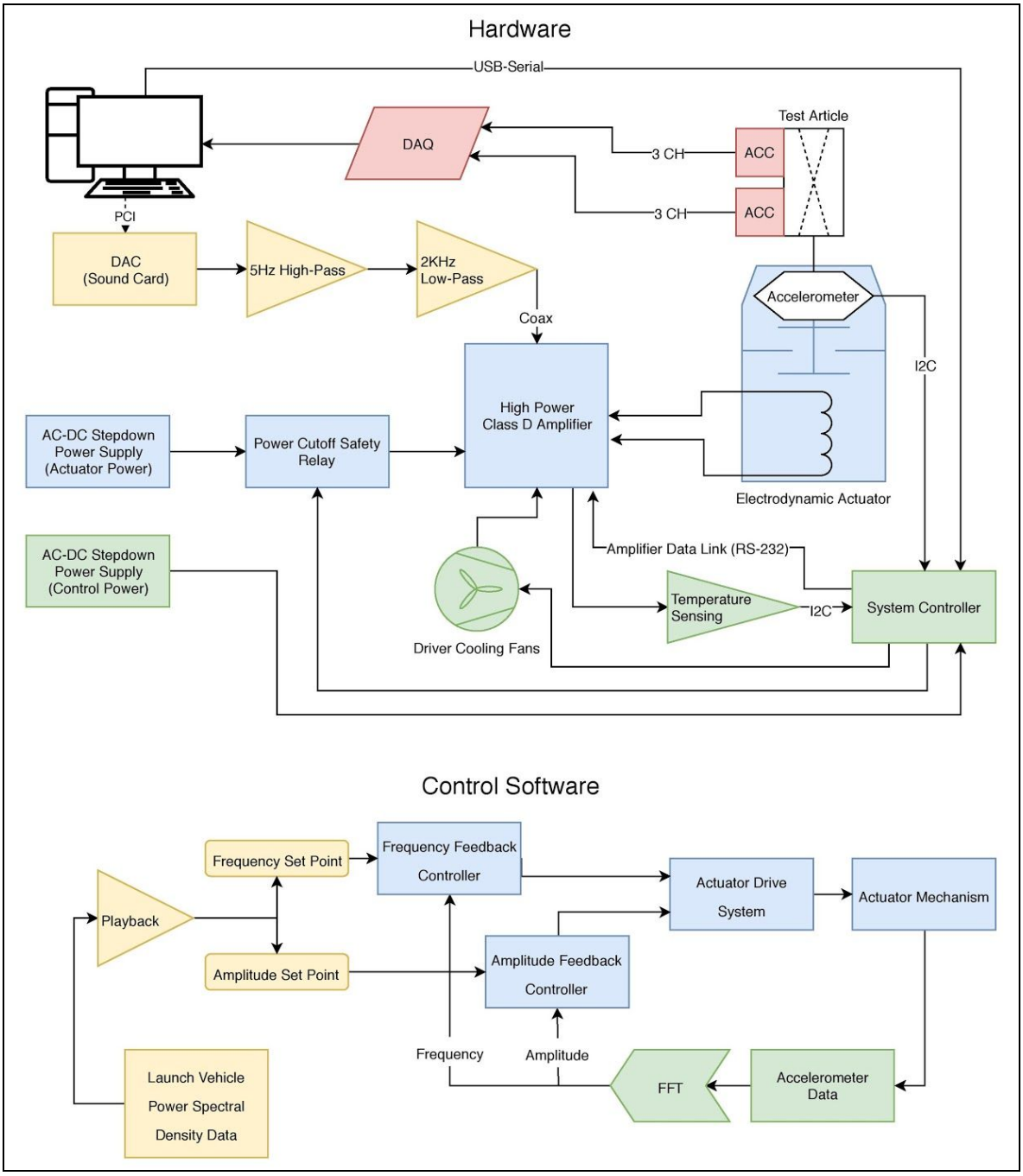


Figure TE.2: *Proposed Block Diagram of Vibroacoustic Test Stand (12/7/19)*

Conclusion

The Testing equipment subteam began the development of in-house CubeSat validation equipment following the limited conceptual work on the HHC done by last year's team. While this year was hit with unforeseen challenges due to COVID-19, this year's Testing Equipment team was able to make significant progress on the goals set at the beginning of the year.

The Testing Equipment team was able to design and fabricate a fully working 3-Axis Helmholtz coil and control software that can be used to validate the craft's Attitude Control System. The HHC has been dismantled and stored at BSOE until a permanent residence for it can be found. Preliminary CAD modeling for the Shock Test Stand has been completed, and the data acquisition system and post processing script have been developed and tested with sample data.

From this point, future work for the Test Equipment team will involve reassembling the HHC once space for it has been found at BSOE; documentation outlining how to reassemble the HHC has been created to assist in this process. While the HHC is fully functional, there is room for improvement in both the electronics and the software. Access to a bench setup will be needed to implement a more robust feedback controller into the firmware.

Next year's work on the STS will need to begin with understanding the impact dynamics surrounding the shock delivery mechanism. Current simulations will need to be revisited and refined to determine the parametric values for the impact hammer and deflection plate. Once this has been thoroughly validated, the main focus will be on finalizing the CAD model and moving towards the fabrication and integration of the complete STS rig.

Appendix A: HHC Engineering Schematics

Software A.1

Helmholtz Coil

[A.1.1] MATLAB Coil Calculation Script

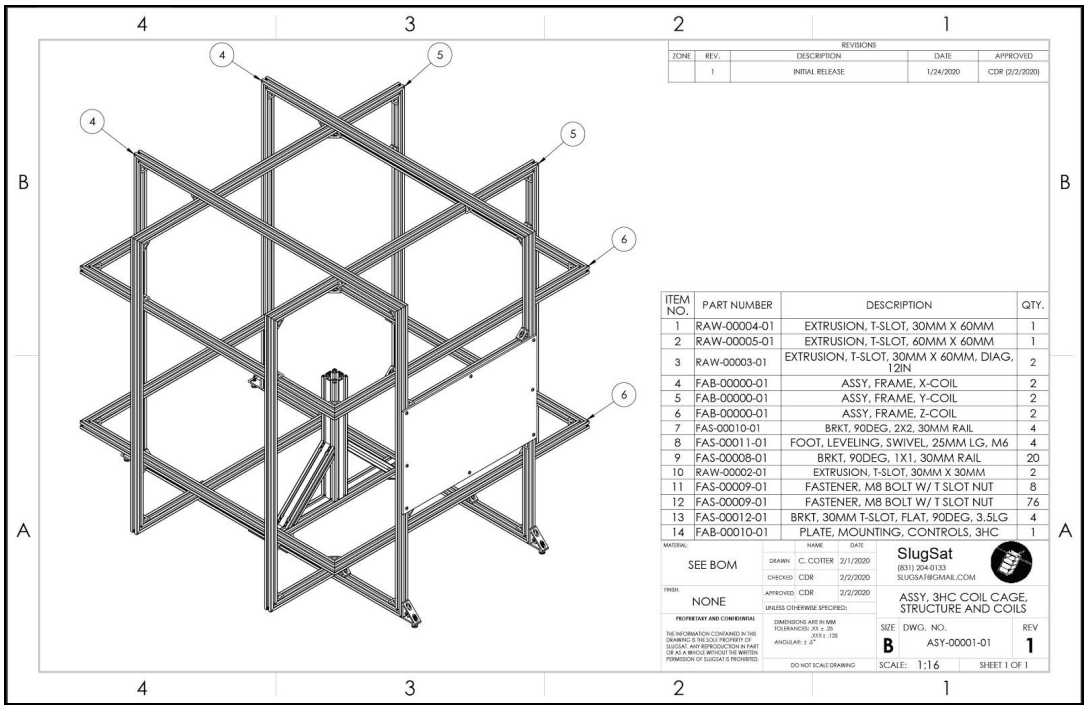
```
% ------------------------------------------------------------------- %
% Helmholtz Coil Pair Winding Calculator
%
% Created By:Caleb Cotter and Ali Elagawami
% Last Updated: 11/19/19
%
% Current Rev: 0.1
%
% Revision History:
% 0.1 - Initial Design - 11/19/19
% 0.2 - Corrected Coil Spacing Errors, Added Prints and Ripple 1/31/20
% Calculations - 1/31/20
% ------------------------------------------------------------------- %
close all;
clear all;
clc;

% Input Parameters
B = 150E-6;     %Teslas
I = 3.8685;     %Coil Current
L = 1.17077;    %Coil Side Length in Meters
d = 0.663;      %Coil Separation in Meters
uo = pi*4E-7;   %Permittivity of a Vacuum
D_wire = 0.8118E-03;    %Wire Conductor Diameter
p = 1.72E-8;    %Ohms Per Meter of Coil Material
pwm_bits = 10;  %H-Bridge PWM Resolution (n-bits)
V_ripple = 1.00 %Voltage Ripple

% Calculations
A_wire = pi*(D_wire/2)^2;    %Wire Cross Sectional Area
gamma = d / L;
n = (B * pi * (L/2) * (1+gamma^2) * sqrt(2+gamma^2)) / (4 * uo * I);
l = (4 * n * L);
R_coil = (2* p * l) / A_wire;
Vss = I*R_coil;
S = (B / 2^pwm_bits);
P = I*Vss;
B_ripple = n / (( pi * (L/2) * (1+gamma^2) * sqrt(2+gamma^2)) / (4 * uo * (V_ripple/R_coil)))


fprintf('--- Results ---\n')
```

```
fprintf('# of Turns Per Coil: %f \n',n);
fprintf('Supply Voltage: %f V\n',Vss);
fprintf('Resistance per Pair: %f Ohms\n',R_coil);
fprintf('Current per Pair: %f A\n',I);
fprintf('Power: %f W\n',P);
fprintf('Length of Wire per Coil Pair: %f m \n',2*l);
fprintf('Field Resolution: %f uT/LSB \n',S*1E6);
fprintf('Voltage Ripple Sensitivity: %f uT/V \n',B_ripple*1E6);


% ------------------------------------------------------------------------- %
% Helmholtz Coil Pair Dimensions to Coil Properties
%
% Created By:Caleb Cottert
% Last Updated: 2/2/20
%
% Current Rev: 0.1
%
% Revision History:
% 0.1 - Initial Design - 2/2/20
% ------------------------------------------------------------------------- %
close all;
clear all;
clc;

% Input Parameters
W = 1.14067;
H = 1.20087;
L_side_avg = (W+H)/2; %Average Coil Side Length in Meters
d = 0.663;                    %Coil Separation in Meters
n = 29;                       %Number of Turns Per Coil
V_supply = 36;                %Nominal Power Supply Voltage
V_ripple = 1.000              %Voltage Ripple of Power Supply
D_wire = 0.8118E-03;          %Wire Conductor Diameter in Meters
pwm_bits = 10;                %H-Bridge PWM Resolution (n-bits)

p = 1.72E-8;    %Ohms Per Meter of Coil Material
uo = pi*4E-7;   %Permittivity of a Vacuum
ur = 1;         %Relative Permittivity of Loop Core

% Calculations
A_wire = pi*(D_wire/2)^2;    %Wire Cross Sectional Area
gamma = d / L_side_avg;
L_wire = n*(2*W + 2*H);
R_coil = (2* p * L_wire) / A_wire;
I_max = (V_supply/R_coil);
B = n / (( pi * (L_side_avg/2) * (1+gamma^2) * sqrt(2+gamma^2)) / (4 * uo * I_max));
B_ripple = n / (( pi * (L_side_avg/2) * (1+gamma^2) * sqrt(2+gamma^2)) / (4 * uo * (V_ripple/R_coil)));
S = (B / 2^pwm_bits);
```

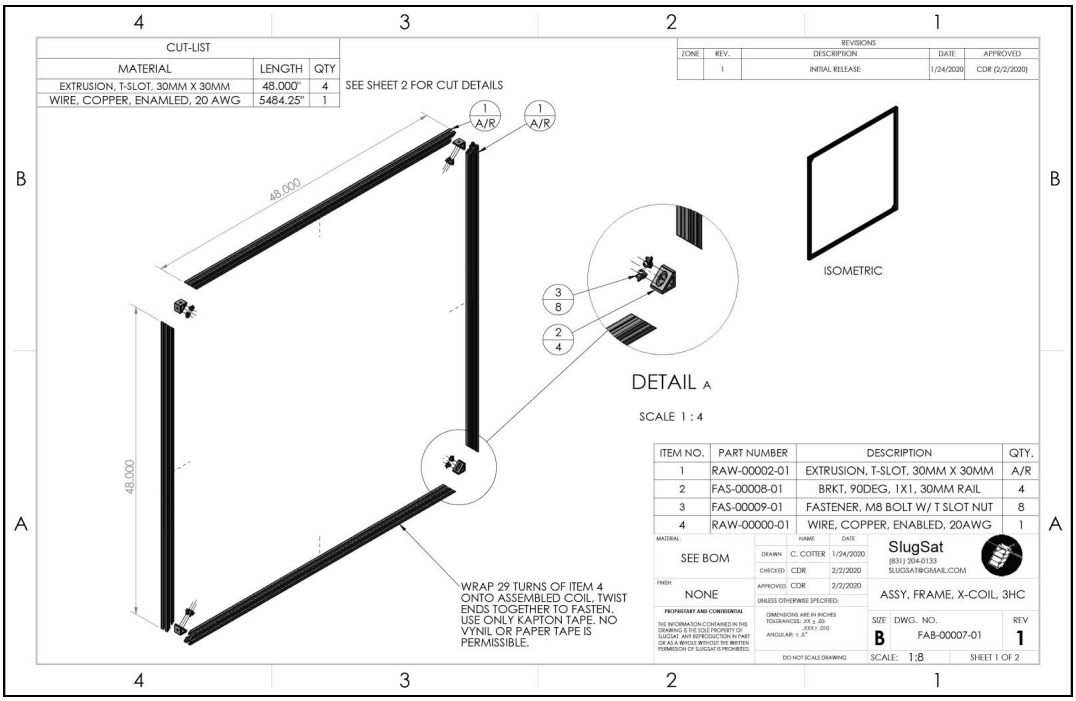
```
L = n^2*(uo*ur/pi)*(-2*(W+H)+2*sqrt(W^2+H^2)-H*log((H+sqrt(W^2+H^2))/W)-W*log((W+sqrt(W^2+H^2))/H)+H*log(2*H/(D_wire/2))+W*log(2*W/(D_wire/2)));


fprintf('Magnetic Field: %f uT\n',B*1E6);
fprintf('Field Resolution: %f uT/LSB \n',S*1E6);
fprintf('Magnetic Field Ripple: %f uT/V \n',B_ripple*1E6);
fprintf('Length of Wire per Coil Axis: %f m \n',2*L_wire);
fprintf('Resistance per Axis: %f Ohms\n',R_coil);
fprintf('Current per Axis: %f A\n',I_max);
fprintf('Power per Axis: %f W\n',I_max*V_supply);
fprintf('Inductance per Coil: %f H\n',L);
fprintf('Inductance per Axis: %f H\n',L*2);
```

Mechanical A.2

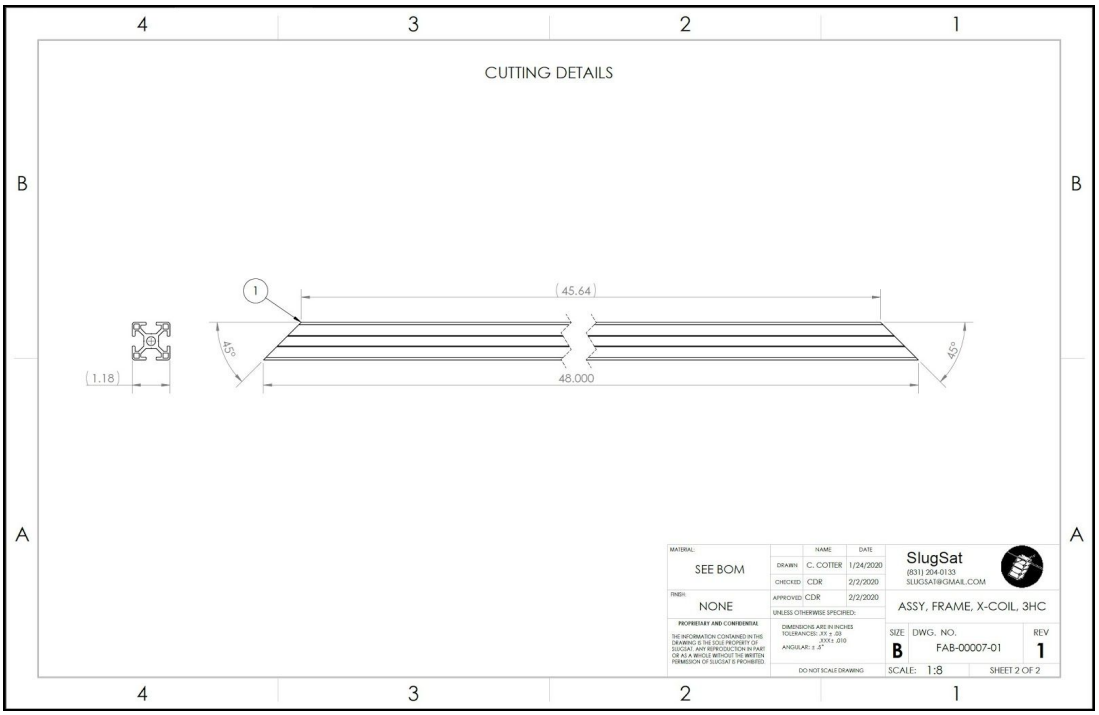
Mechanical Drawings



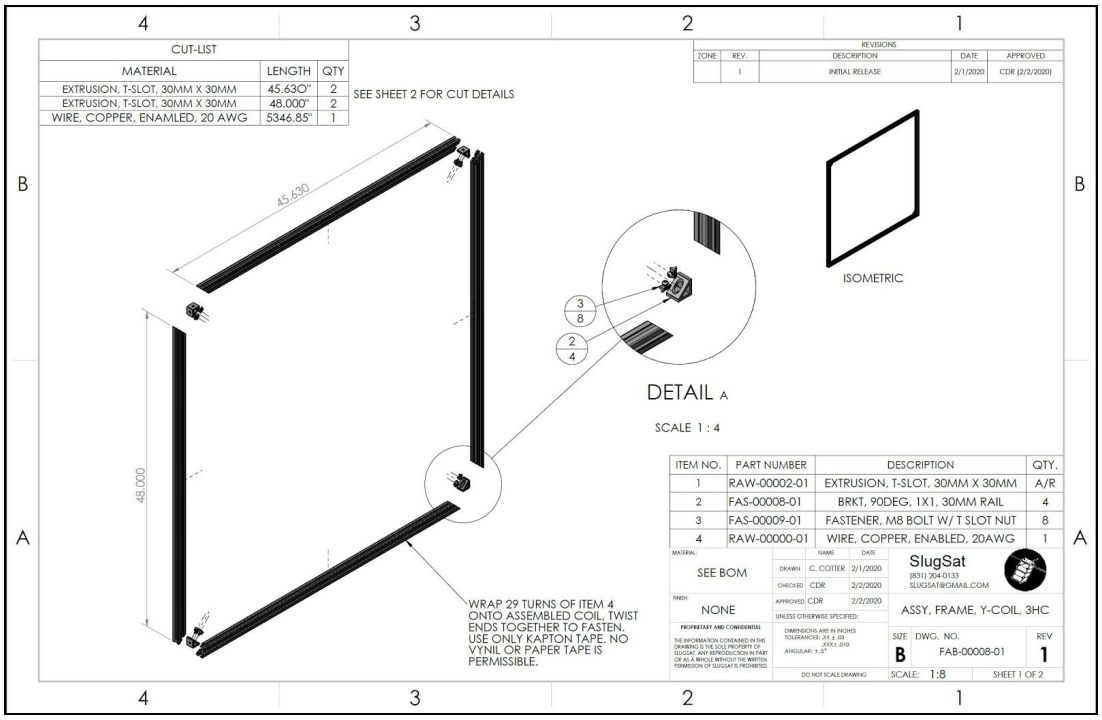
[**A.2.1**] ASY-00001-01



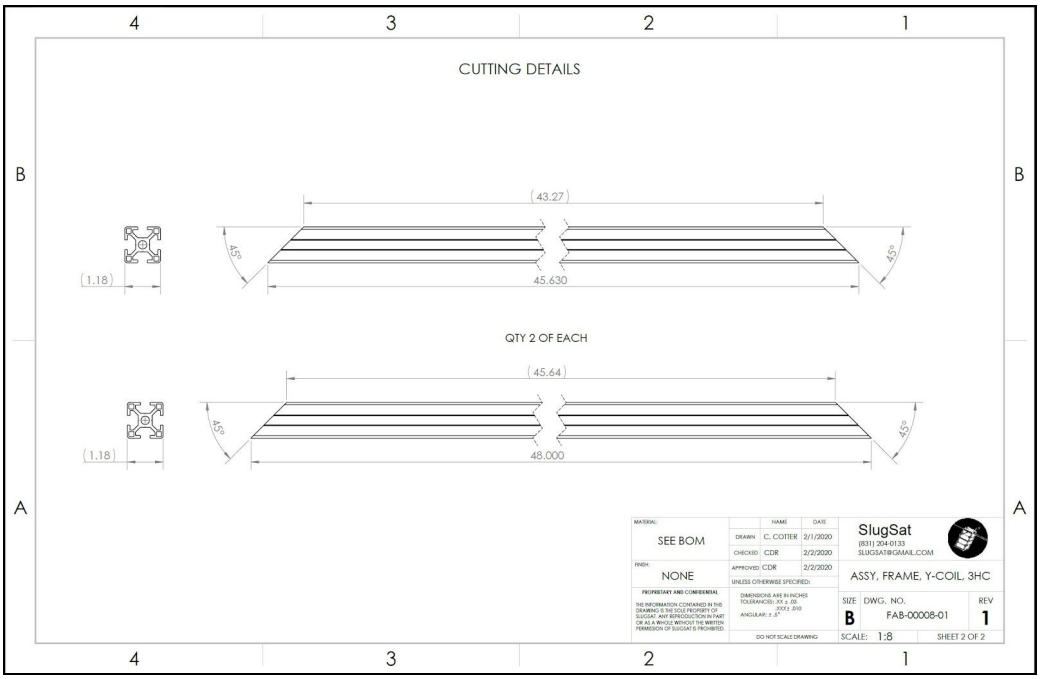
[**A.2.2**] FAB-00007-01 (SHEET 1 OF 2)



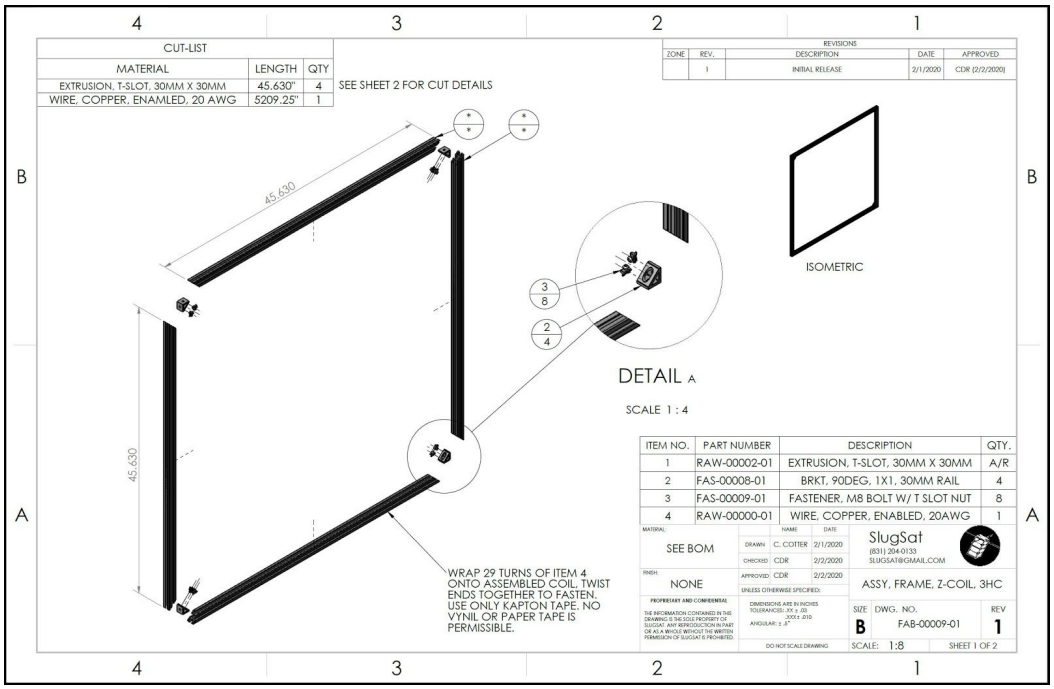
[**A.2.3**] FAB-00007-01 (SHEET 2 OF 2)



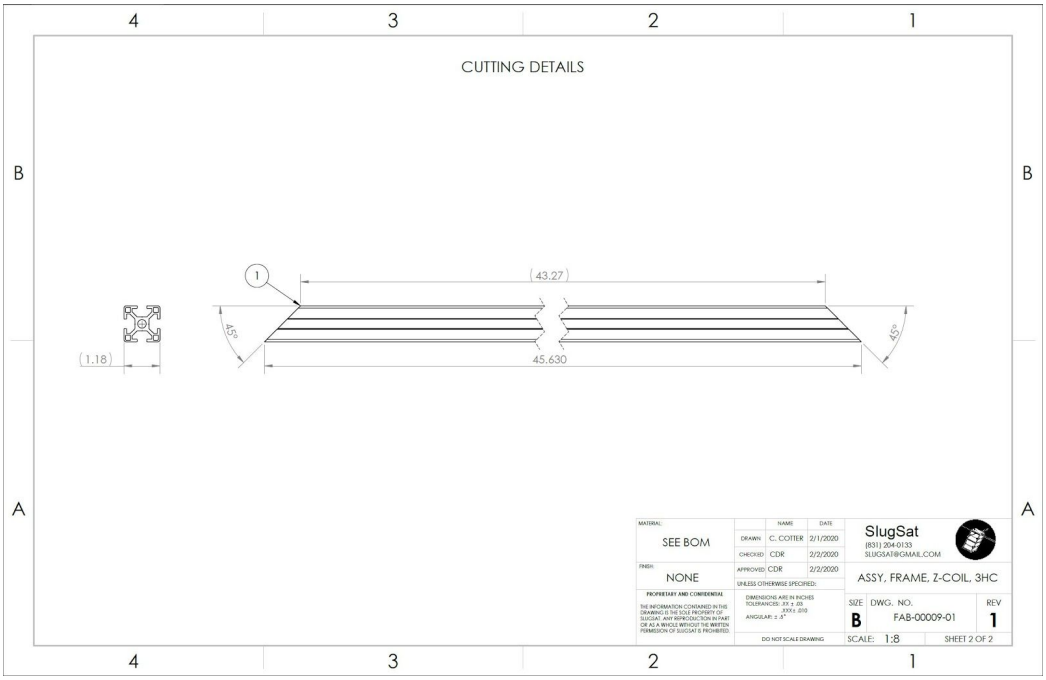
[**A.2.4**] FAB-00008-01 (SHEET 1 OF 2)



[**A.2.5**] FAB-00008-01 (SHEET 2 OF 2)



[**A.2.6**] FAB-00009-01 (SHEET 1 OF 2)



[**A.2.7**] FAB-00009-01 (SHEET 2 OF 2)

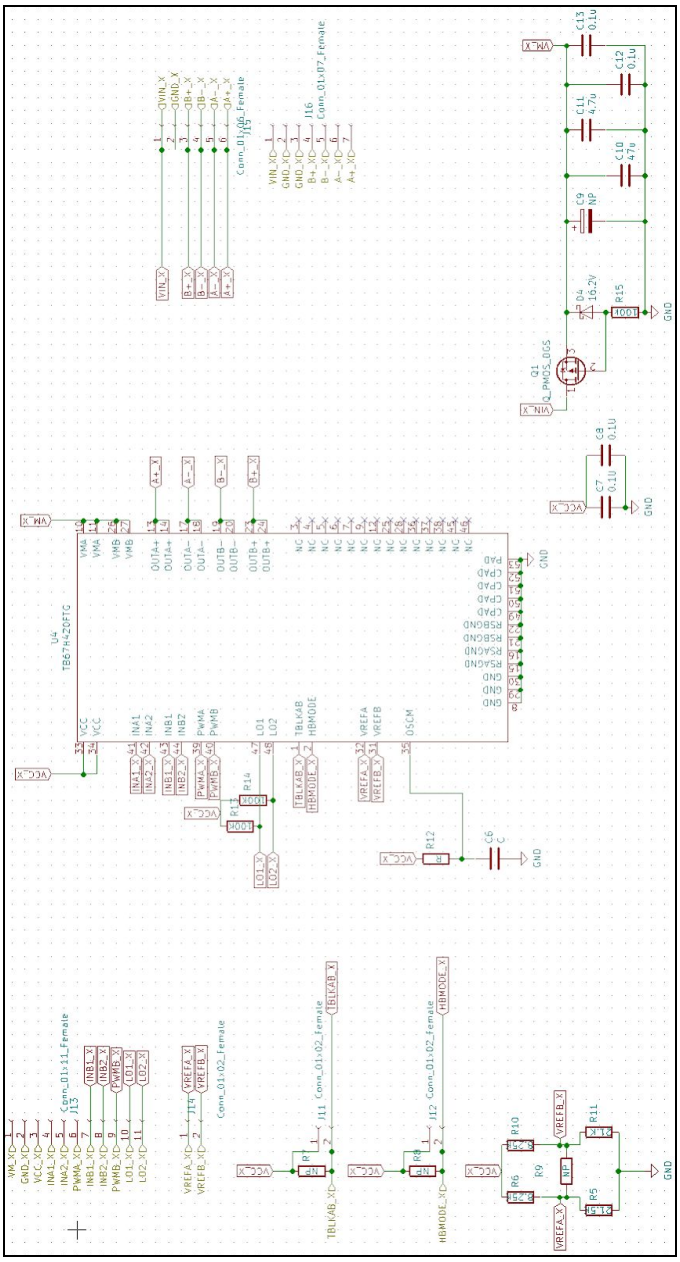
Bill of Materials

3-Axis Helmholtz Coil Parts List / Budget							
Level	**Part Number**	**Description**	**Vendor**	**Qty**	**Units**	**Unit Price**	**Line Price**
0	**ASY-000000-01**	**ASSY, 3HC COIL CAGE**		**1**	**EA**	**$1,500.28**	**$1,500.28**
	PSU-00001-01	PSU, 36VDC, 480W, ENCLOSED, IP65	TRC Electronics	1	EA	$181.00	$181.00
1	**ASY-000001-01**	**ASSY, 3HC COIL CAGE, STRUCTURE AND COILS**		**1**	**EA**	**$1,319.28**	**$1,319.28**
	RAW-00002-01	EXTRUSION, T-SLOT, 30MM X 30MM	McMaster-Carr	3.6	FT	$4.05	$14.58
	RAW-00003-01	EXTRUSION, T-SLOT, 30MM X 60MM, DIAG, 12IN	McMaster-Carr	2	EA	$28.38	$56.76
	RAW-00004-01	EXTRUSION, T-SLOT, 30MM X 60MM	McMaster-Carr	2	FT	$8.88	$17.76
	RAW-00005-01	EXTRUSION, T-SLOT, 60MM X 60MM	McMaster-Carr	2	FT	$23.30	$46.60
	RAW-00006-01	SHEET, ACRYLIC, WHITE, 24X24X0.25	McMaster-Carr	1	EA	$39.860	$39.860
	FAS-00011-01	FOOT, LEVELING, SWIVEL, 25MM LG, M6	McMaster-Carr	4	EA	$5.600	$22.400
	FAS-00012-01	BRKT, 30MM T-SLOT, FLAT, 90DEG, 3.5LG	McMaster-Carr	4	EA	$13.080	$52.320
	FAS-00010-01	BRKT, 90DEG, 2X2, 30MM RAIL	McMaster-Carr	4	EA	$10.080	$40.320
	FAS-00008-01	BRKT, 90DEG, 1X1, 30MM RAIL	McMaster-Carr	20	EA	$5.790	$115.800
	FAS-00009-01	FASTENER, M8 BOLT W/ T SLOT NUT	McMaster-Carr	84	EA	$1.610	$135.240
2	**FAB-000007-01**	**ASSY, FRAME, X-COIL, 3HC**		**2**		**$130.34**	**$260.68**
	RAW-00002-01	EXTRUSION, T-SLOT, 30MM X 30MM	McMaster-Carr	16	FT	$4.05	$64.80
	FAS-00008-01	BRKT, 90DEG, 1X1, 30MM RAIL	McMaster-Carr	4	EA	$5.790	$23.160
	FAS-00009-01	FASTENER, M8 BOLT W/ T SLOT NUT	McMaster-Carr	8	EA	$1.610	$12.880

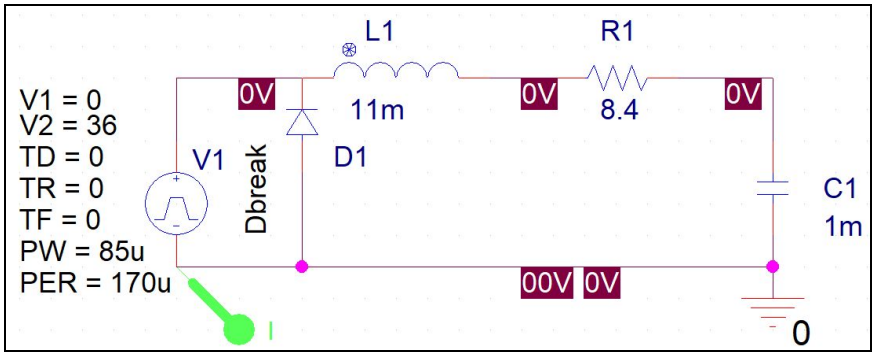
	RAW-00 001-01	WIRE, COPPER, ENABLED, 20AWG	Powerwerx	456.85	FT	$0.065	$29.499
2	**FAB-00 008-01**	**ASSY, FRAME, Y-COIL, 3HC**		**2**		**$129.61**	**$259.22**
	RAW-00 002-01	EXTRUSION, T-SLOT, 30MM X 30MM	McMaster-Carr	16	FT	$4.05	$64.80
	FAS-000 08-01	BRKT, 90DEG, 1X1, 30MM RAIL	McMaster-Carr	4	EA	$5.790	$23.160
	FAS-000 09-01	FASTENER, M8 BOLT W/ T SLOT NUT	McMaster-Carr	8	EA	$1.610	$12.880
	RAW-00 001-01	WIRE, COPPER, ENABLED, 20AWG	Powerwerx	445.57	FT	$0.065	$28.770
2	**FAB-00 009-01**	**ASSY, FRAME, Z-COIL, 3HC**		**2**		**$128.87**	**$257.74**
	RAW-00 002-01	EXTRUSION, T-SLOT, 30MM X 30MM	McMaster-Carr	16	FT	$4.05	$64.80
	FAS-000 08-01	BRKT, 90DEG, 1X1, 30MM RAIL	McMaster-Carr	4	EA	$5.790	$23.160
	FAS-000 09-01	FASTENER, M8 BOLT W/ T SLOT NUT	McMaster-Carr	8	EA	$1.610	$12.880
	RAW-00 001-01	WIRE, COPPER, ENABLED, 20AWG	Powerwerx	434.1	FT	$0.065	$28.030

[**A.2.8**] 3-Axis Helmholtz Coil Parts List / Budget

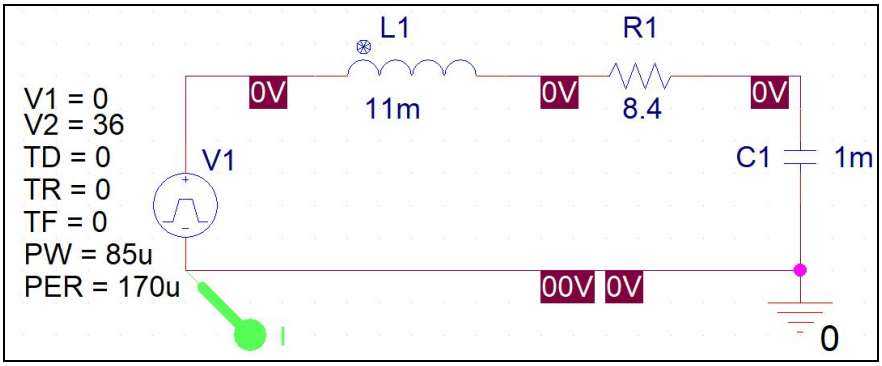
Electrical A.3



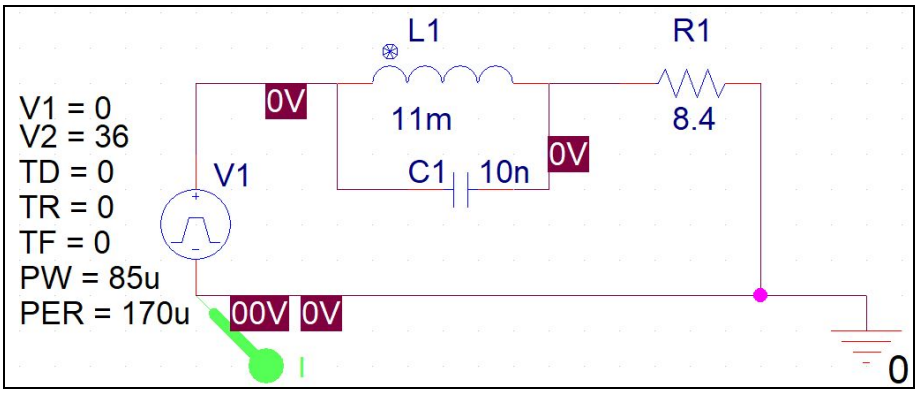
[A.3.1] H-bridge (Toshiba TB67H420FTG) schematic



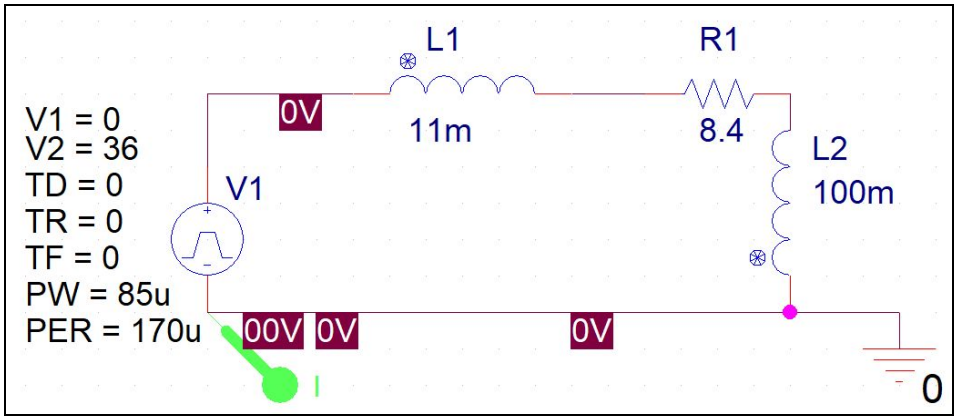
[A.3.2a] Coil simulation-Constant Conduction Mode schematic



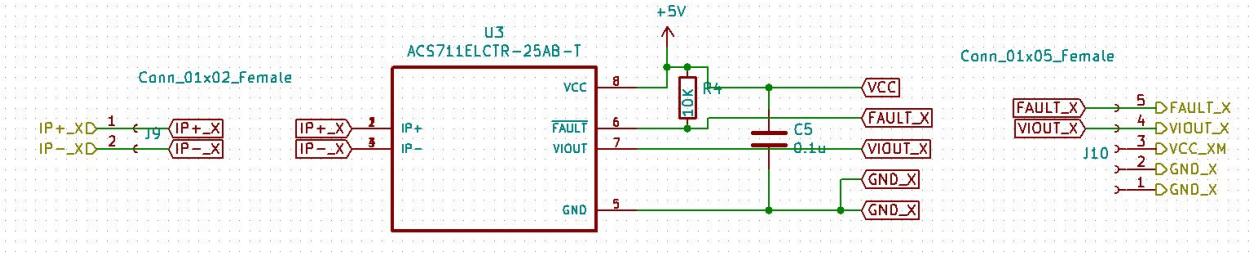
[A.3.2b] Coil simulation-Series LCR schematic



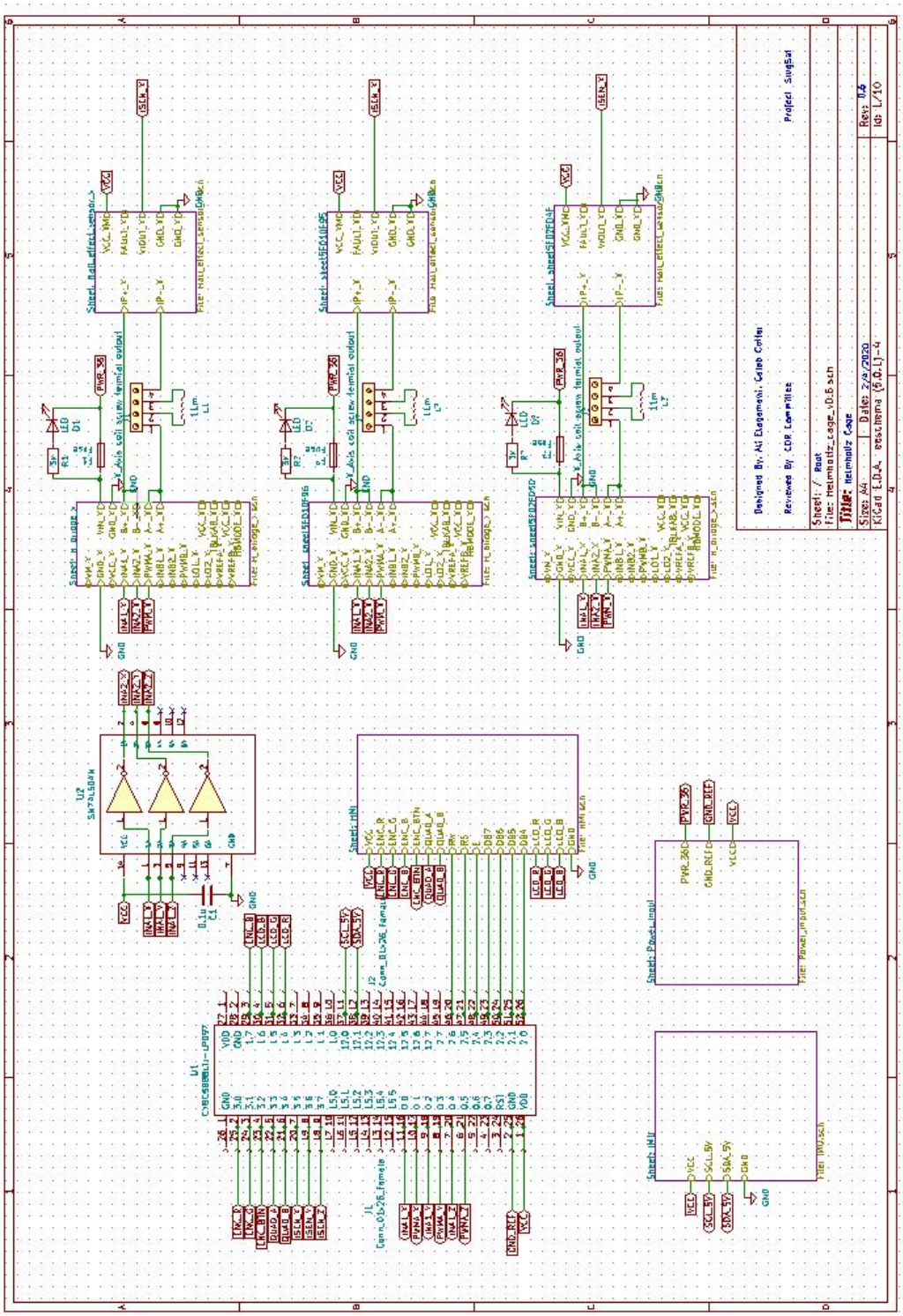
[A.3.2c] Coil simulation-Parallel LCR schematic



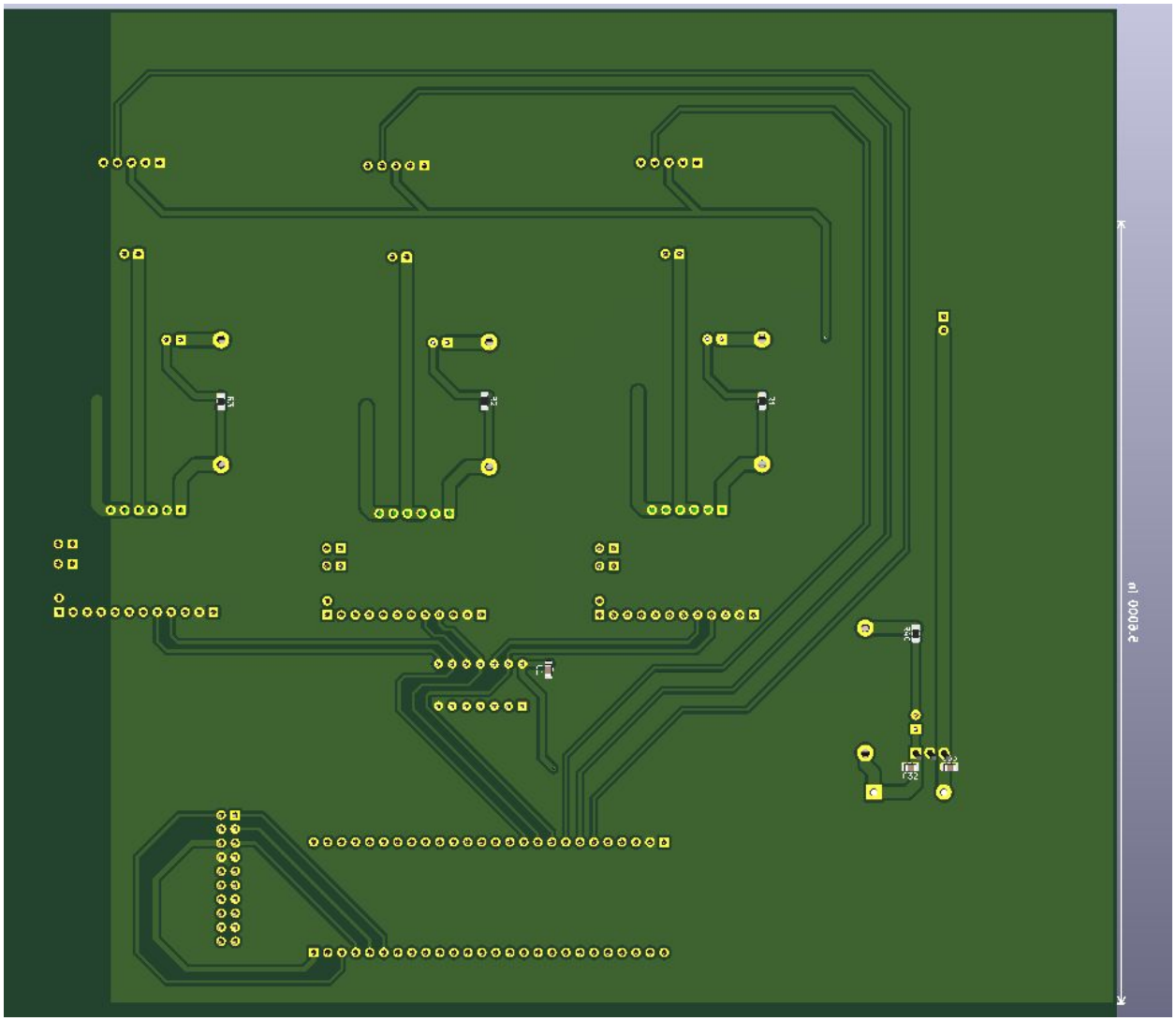
[A.3.2d] Coil simulation-Series LR rev 2.0 schematic



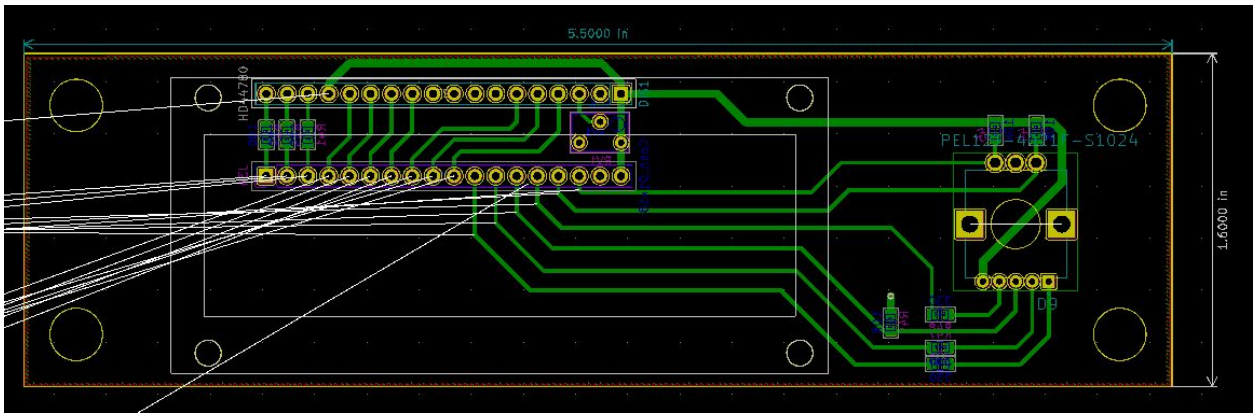
[A.3.3] current sensor (ACS711EX) schematic



[A.3.4] full motherboard electric schematic



[**A.3.5**] motherboard 3D isometric bottom view



[**A.3.6**] HMI layout V1.0 (2/22/2020)

Appendix B: STS Engineering Schematics

B.1 Software

[B.1.1] SRS Generation Script

```
%%
% STS Shock Test Stand
% SRS Generation Script
%
% Caleb Cotter
%
% 4/30/20
%
%% Program Inputs
close all;
clear all;
clc;

t_sim = 40E-3; %seconds
t_step = 0.00000005; %seconds

% Constants
g = 9.80665;

% SRS Test Frequencies
f_n = [30, 100, 700, 1000, 1500, 5000, 10000]; %Hz
Q = 10;

% SRS Goals
acc_g_n = [5, 100, 1500, 2400, 4000, 4000, 2000]; %G's

%Sine Shock Pulse
t_pulse_duration = 0.74E-3; %seconds (original)
t_pulse_duration = 8.775802299165001e-04; %seconds
a_peak_pulse = 3150; %G's

[u_pulse,t_pulse] = gensig('sin', t_pulse_duration, t_pulse_duration, t_step); %Half-Sine
u_pulse = u_pulse*a_peak_pulse*g; %Scale to m/s^2
t_zeros = ((length(t_pulse)*t_step):t_step:t_sim)';
u_zeros = zeros(length(t_zeros),1);
t = vertcat(t_pulse,t_zeros);
u = vertcat(u_pulse,u_zeros);

% plot(t*1000,u,'LineWidth',5);
% set(gca,'FontSize',20)
```

```
% xlabel("Time (ms)");
% ylabel("Acceleration (g)");
% legend("Acceleration Profile of UUT");
% title("Shock Acceleration Profile");

%% Processing
k = 1;
zeta = 1/(2*Q);
w_n = f_n*2*pi; %rad/s

% Simulate System Responses
s = tf('s');
sys_resp = zeros(length(w_n),length(t));
for i = 1:length(w_n)
    sys = (k*w_n(i)^2)/(s^2+2*zeta*w_n(i)*s+w_n(i)^2);
    sys_resp(i,:) = lsim(sys,u,t);
end
set(gca,'FontSize',20)
%%
% Extract Max Magnitude of Each Frequency
acc_max_w_n = zeros(1,length(w_n));
for i = 1:length(w_n)
    acc_max_w_n(i) = max(sys_resp(i,:));
end

hold on;
plot(f_n,acc_max_w_n./g,'x-','LineWidth',3);
plot(f_n,acc_g_n,'o-','LineWidth',3);
set(gca, 'XScale', 'log', 'YScale', 'log');
xlabel("Frequency (Hz)");
ylabel("Acceleration (g)");
legend("Proposed System","NASA GEVS",'Location','best');
set(gca,'FontSize',20)
delta_x = 0.5*a_peak_pulse*t_pulse_duration^2

 % --- Check Results against NASA GEVS Spec ---
meetsRequirements = 1;
for i = 1:length(f_n)
    if(acc_max_w_n(i)/g < acc_g_n(i))
        meetsRequirements = 0;
    end
end
if(meetsRequirements)
    disp("Meets GEVS spec!");
else
    disp("Does not meet GEVS spec!");
end
```

[B.1.2] Hammer Impact Calculations

```
%% HammerImpactCalcs.m
%   For usage with development of the SlugSat Shock Test Stand (STS)
%   Created by : Nick Jannuzzi & Caleb Cotter
%   Property of UCSC SlugSat Program
%   Each section can be run individually to calculate different portions
%   for the system.
close all;
clear all;
clc;

%% Deflection Plate Dynamics Transfer Function
% Plate dimensions and material parameters (in meters)
Plate_Length = 0.2;
Plate_Width = 0.1;
Plate_Thickness = 0.00635*2;

% Look up this value for the material (no there is not an Old's Modulus)
Youngs_Modulus = 69e9;          % Pascals
Desired_Deflection = 0.135e-3;  % Meters

% Calculate the equivalent spring constant K of the beam, in Newtons / Meter
Cantilever_K = (Plate_Width*Plate_Thickness^3*Youngs_Modulus)/(4*Plate_Length^2);

% Calculate mass of plate from M = DV
% Material Density
Material_Density = 2700;                                % Kilograms / Meters^3
Plate_Volume = Plate_Length*Plate_Width*Plate_Thickness;    % Meters^3
Plate_Mass = Material_Density*Plate_Volume;

% Natural Frequency of Plate + Satellite
Sat_Mass = 2;                   % Kilograms
T_Mass = Plate_Mass + Sat_Mass; % Kilograms
Wo = sqrt(Cantilever_K/T_Mass); % Radians/Second

% Damping ratio of Material (Look this up)->Damping Coefficient
Damp_Ratio = 0.5e-4; %(Aluminum)

% Damping coefficient (zeta) = viscous coefficient(lambda)/2sqrt(km) since
% our beam is of non-negligible thickness, lambda is the damping ratio of
% the material rather than of air.
Damp_Coefficient = Damp_Ratio/(2*sqrt(Cantilever_K*T_Mass));

% Q-Factor (Don't really need this, but put it here anyways
Q_Factor = 1/(2*Damp_Coefficient);
```

```
% 2nd Order System Model (Force -> Deflection Transfer Function)
s = tf('s');
G = 1/((1/Wo^2)*s^2+(2*Damp_Coefficient/Wo)*s+1)
% Need to set step amplitude to non unit value
opt = stepDataOptions('StepAmplitude', Desired_Deflection);
step(G,opt)
Period = 1/(2*pi*Wo)

% Energy stored in the plate (Joules)
U_Plate = 0.5*Cantilever_K*Desired_Deflection^2;

%% Calculation of Hammer Arm Kinetic Energy Given Masses and Length of Arm
% Constants
g = 9.80665; % m/s

% Hammer Arm Parameters
M_Arm = 1;
M_Head = 1;
L_Arm = 1;

% Center of mass location for Compound Pendulum
Xcm = (M_Arm*L_Arm/2 + M_Head*L_Arm)/(M_Head+M_Arm)

% Hammer arm inertia can be calculate by representing it as a uniform rod
% of length L with a point mass attached at L
Hammer_MoI = 1/3*M_Arm*L_Arm^2 + M_Hammer*L_Arm^2;

% Compound pendulum period equation
Compound_Pendulum_Period = 2*pi*sqrt(Hammer_MoI/(M_Hammer*M_Arm*g*Xcm));

% Frequency of compound pendulum
Compound_Pendulum_Frequency = 1/Compound_Pendulum_Period;

% Angular Velocity
AngV = 2*pi*Compound_Pendulum_Frequency;

% Kinetic Energy of Pendulum
KE_Pendulum = 0.5*Hammer_MoI*AngV^2;

%% Calculate Deflection in Cantilever Beam given Force
% Parameters
Elasticity_Modulus = 1; %(Young's Modulus)
Beam_Height = 1;
Beam_Thickness = 1;

P = 1; %(Point Load)
% Distance from support to P
```

```
a = 1;
% Distance from P to end of cantilever beam
b = 1;%(we never actually use this, don't worry)

% Area Moment of Inertia for Cantilever Beam
AI = 1;

% Deflection of a beam, with a concentrated load P at distance a from the
% support can be found from the equation:
%
% Beam_Deflection = (P*a^2)/(6*AI*Elasticity_Modulus)*(2*a);
%
% This can be rearranged to find the load needed to obtain a desired
% deflection.

% Deflection we need for acceleration profile
Desired_Deflection = 0.135e-3;

% Calculate impulse force needed to obtain deflection in Aluminum plate
Load_Needed = (6*AI*Elasticity_Modulus*Desired_Deflection)/(2*a^3);

%% Calculating Strain Energy Stored in a Beam
```

[B.1.3] Data-Analysis Program / UI

```
classdef STS_Test_Analysis < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        STSUIFigure                 matlab.ui.Figure
        LoadAccelerationDataButton  matlab.ui.control.Button
        ExportSRSDataButton         matlab.ui.control.Button
        ExportSRSFigureButton       matlab.ui.control.Button
        ShockTestStandDataAnalysisProgramLabel  matlab.ui.control.Label
        Panel                       matlab.ui.container.Panel
        AccelerationGraph           matlab.ui.control.UIAxes
        Panel_2                     matlab.ui.container.Panel
        SRSGraph                    matlab.ui.control.UIAxes
        MeetsNASASpecLampLabel      matlab.ui.control.Label
        MeetsNASASpecLamp           matlab.ui.control.Lamp
        DataLoadedLampLabel         matlab.ui.control.Label
        DataLoadedLamp              matlab.ui.control.Lamp
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: LoadAccelerationDataButton
        function LoadAccelerationDataButtonPushed(app, event)

            % Prompt User for File
            [file,path] = uigetfile('*.csv');
            filename = fullfile(path, file);

            % Load Data if File Exists
            if(exist(filename,'file') == 2)
                app.DataLoadedLamp.Color = 'green';
                data = readtable(filename);
            else
                app.DataLoadedLamp.Color = 'red';
                return;
            end

            acc_mat = table2array(data);


            % --- Generate Fake Data for Test ---
            t_sim = 5E-3; %seconds
            t_step = 0.00000005; %seconds
```

```
% Constants
g = 9.80665;

% SRS Test Frequencies
f_n = [30, 100, 700, 1000, 1500, 5000, 10000]; %Hz
Q = 10;

% SRS Goals
acc_g_n = [5, 100, 1500, 2400, 4000, 4000, 2000]; %G's

% Half-Sine Shock Pulse
t_pulse_duration = 0.37E-3; %seconds
a_peak_pulse = 3250; %G's

[u_pulse,t_pulse] = gensig('sin', 2*t_pulse_duration, 2*t_pulse_duration, t_step); %Half-Sine
u_pulse = u_pulse*a_peak_pulse*g; %Scale to m/s^2
t_zeros = ((length(t_pulse)*t_step):t_step:t_sim)';
u_zeros = zeros(length(t_zeros),1);
t = vertcat(t_pulse,t_zeros);
u = vertcat(u_pulse,u_zeros);

%--- Prep & Test Data ---

plot(app.AccelerationGraph,t,u);

% --- SRS Generation & Plot ---
k = 1;
zeta = 1/(2*Q);
w_n = f_n*2*pi; %rad/s

% Simulate System Responses
s = tf('s');
sys_resp = zeros(length(w_n),length(t));
for i = 1:length(w_n)
    sys = (k*w_n(i)^2)/(s^2+2*zeta*w_n(i)*s+w_n(i)^2);
    sys_resp(i,:) = lsim(sys,u,t);
end

% Extract Max Magnitude of Each Frequency
acc_max_w_n = zeros(1,length(w_n));
for i = 1:length(w_n)
    acc_max_w_n(i) = max(sys_resp(i,:));
end

acc_max_w_n = acc_max_w_n/g;

%Plot SRS
```

```
            hold(app.SRSGraph, 'on');
            plot(app.SRSGraph,f_n,acc_max_w_n);
            plot(app.SRSGraph,f_n,acc_g_n);
            legend(app.SRSGraph,"Test Results","NASA GEVS",'Location',"best");
            hold(app.SRSGraph, 'off');

            % --- Check Results against NASA GEVS Spec ---
            meetsRequirements = 1;
            for i = 1:length(f_n)
                if(acc_max_w_n(i) < acc_g_n(i))
                    meetsRequirements = 0;
                end
            end
            if(meetsRequirements)
                app.MeetsNASASpecLamp.Color = 'green';
            else
                app.MeetsNASASpecLamp.Color = 'red';
            end

        end

        % Button pushed function: ExportSRSFigureButton
        function ExportSRSFigureButtonPushed(app, event)
            [file, path, filterindex] = uiputfile({'*.png', 'PNG (*.png)'; '*.*', 'All Files (*.*)'});
            filename = fullfile(path, file);

%               FigureSave = figure;
%               copyobj(app.SRSGraph,FigureSave)
%               saveas(FigureSave,filename,'png');
        end

        % Button pushed function: ExportSRSDataButton
        function ExportSRSDataButtonPushed(app, event)
            [filename, pathname, filterindex] = uiputfile({'*.csv', 'CSV (*.csv)'; '*.*', 'All Files (*.*)'});

        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create STSUIFigure and hide until all components are created
            app.STSUIFigure = uifigure('Visible', 'off');
            app.STSUIFigure.Color = [0.902 0.902 0.902];
```

```
        app.STSUIFigure.Position = [100 100 820 1098];
        app.STSUIFigure.Name = 'STS';

        % Create LoadAccelerationDataButton
        app.LoadAccelerationDataButton = uibutton(app.STSUIFigure, 'push');
        app.LoadAccelerationDataButton.ButtonPushedFcn = createCallbackFcn(app,
@LoadAccelerationDataButtonPushed, true);
        app.LoadAccelerationDataButton.Position = [24 28 141 29];
        app.LoadAccelerationDataButton.Text = 'Load Acceleration Data';

        % Create ExportSRSDataButton
        app.ExportSRSDataButton = uibutton(app.STSUIFigure, 'push');
        app.ExportSRSDataButton.ButtonPushedFcn = createCallbackFcn(app,
@ExportSRSDataButtonPushed, true);
        app.ExportSRSDataButton.Position = [335 28 132 29];
        app.ExportSRSDataButton.Text = 'Export SRS Data';

        % Create ExportSRSFigureButton
        app.ExportSRSFigureButton = uibutton(app.STSUIFigure, 'push');
        app.ExportSRSFigureButton.ButtonPushedFcn = createCallbackFcn(app,
@ExportSRSFigureButtonPushed, true);
        app.ExportSRSFigureButton.Position = [184 28 132 29];
        app.ExportSRSFigureButton.Text = 'Export SRS Figure';

        % Create ShockTestStandDataAnalysisProgramLabel
        app.ShockTestStandDataAnalysisProgramLabel = uilabel(app.STSUIFigure);
        app.ShockTestStandDataAnalysisProgramLabel.HorizontalAlignment = 'center';
        app.ShockTestStandDataAnalysisProgramLabel.FontSize = 18;
        app.ShockTestStandDataAnalysisProgramLabel.FontWeight = 'bold';
        app.ShockTestStandDataAnalysisProgramLabel.Position = [213 1042 396 46];
        app.ShockTestStandDataAnalysisProgramLabel.Text = 'Shock Test Stand - Data Analysis
Program';

        % Create Panel
        app.Panel = uipanel(app.STSUIFigure);
        app.Panel.Position = [19 566 785 470];

        % Create AccelerationGraph
        app.AccelerationGraph = uiaxes(app.Panel);
        title(app.AccelerationGraph, 'Acceleration Data')
        xlabel(app.AccelerationGraph, 'Time (s)')
        ylabel(app.AccelerationGraph, 'Acceleration (G''s)')
        app.AccelerationGraph.Box = 'on';
        app.AccelerationGraph.BoxStyle = 'full';
        app.AccelerationGraph.Color = 'none';
        app.AccelerationGraph.XGrid = 'on';
        app.AccelerationGraph.YGrid = 'on';
```

```
    app.AccelerationGraph.Position = [41 18 704 434];

    % Create Panel_2
    app.Panel_2 = uipanel(app.STSUIFigure);
    app.Panel_2.Position = [20 86 785 470];

    % Create SRSGraph
    app.SRSGraph = uiaxes(app.Panel_2);
    title(app.SRSGraph, 'Shock Response Spectrum')
    xlabel(app.SRSGraph, 'Frequency (Hz)')
    ylabel(app.SRSGraph, 'Acceleration (G''s)')
    app.SRSGraph.Box = 'on';
    app.SRSGraph.BoxStyle = 'full';
    app.SRSGraph.XMinorTick = 'on';
    app.SRSGraph.YMinorTick = 'on';
    app.SRSGraph.Color = 'none';
    app.SRSGraph.XGrid = 'on';
    app.SRSGraph.YGrid = 'on';
    app.SRSGraph.XScale = 'log';
    app.SRSGraph.YScale = 'log';
    app.SRSGraph.Position = [41 18 704 434];

    % Create MeetsNASASpecLampLabel
    app.MeetsNASASpecLampLabel = uilabel(app.STSUIFigure);
    app.MeetsNASASpecLampLabel.HorizontalAlignment = 'right';
    app.MeetsNASASpecLampLabel.Position = [642 31 105 22];
    app.MeetsNASASpecLampLabel.Text = 'Meets NASA Spec';

    % Create MeetsNASASpecLamp
    app.MeetsNASASpecLamp = uilamp(app.STSUIFigure);
    app.MeetsNASASpecLamp.Position = [762 31 20 20];
    app.MeetsNASASpecLamp.Color = [0.8 0.8 0.8];

    % Create DataLoadedLampLabel
    app.DataLoadedLampLabel = uilabel(app.STSUIFigure);
    app.DataLoadedLampLabel.HorizontalAlignment = 'right';
    app.DataLoadedLampLabel.Position = [511 31 75 22];
    app.DataLoadedLampLabel.Text = 'Data Loaded';

    % Create DataLoadedLamp
    app.DataLoadedLamp = uilamp(app.STSUIFigure);
    app.DataLoadedLamp.Position = [601 31 20 20];
    app.DataLoadedLamp.Color = [0.851 0.3255 0.098];

    % Show the figure after all components are created
    app.STSUIFigure.Visible = 'on';
end
```

```
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = STS_Test_Analysis

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.STSUIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.STSUIFigure)
        end
    end
end
```

Appendix C: SVS Engineering Schematics and Scripts

[C.1] Spectral Match Assessment Matlab Script

```
%Spectral_Match_v1
%Calculates the Spectrum Match of LEDs and QTH based on datasheets relative
%intensity spectrum and radiant flux\luminous flux
%Simulates with least amount of components using White, UV, Green, red,
%and different waveband blues
Clc;
Clear;
close all;

%Retrieves ASTM Solar Reference Irradiance Spectrum
filename='ASTM_SolarIrradiance_AM0.csv';
Solar_reference = csvread(filename,1,0,[1 0 1697 1]);
total=trapz(Solar_reference(:,1),Solar_reference(:,2));
fprintf('Solar Reference Total Irradiance %.2f\n',total);
%Retrieves Photopic curve
filename='PhotopicCurve.csv';
photopic = csvread(filename,1,0,[1 0 471 1]);

%Retrieves QTH Irradiance Spectrum
filename='QTH_v1.csv';
QTH = csvread(filename,1,0,[1 0 212 1]);

%Retrieving OSRAM Deep Blue LED Relative Intensity Spectrum from CSV
watt=.660; %at 350mA
filename='GD_CSXPM1_14_20160712_spectrum.csv';
LED_B = csvread(filename,1,0,[1 0 66 1]);

%Calculates absolute Radiant flux Spectrum
B_intensity_multiplier = (watt ./ trapz(LED_B(:,1), LED_B(:,2)));
LED_B (:,2) = LED_B(:,2) .*B_intensity_multiplier;

%Retrieving OSRAM Far Red LED Relative Intensity Spectrum from CSV
watt = 0.383; % at 350mA
filename='far_red_spectrum.csv';
LED_R = csvread(filename,1,0,[1 0 136 1]);

%Calculates absolute Radiant flux Spectrum
red_intensity_multiplier = watt ./ trapz(LED_R(:,1), LED_R(:,2));
LED_R (:,2) = LED_R(:,2) .*red_intensity_multiplier; %should be in w/nm
```

```
%Creates relative intensity spectrum for UV LED from Gaussian curve
watt=.930*3/4; %with current being near 0.35A
LED_UV= guass_estimate(405,20);

%Calculates absolute Radiant flux Spectrum
UV_intensity_multiplier = watt ./ trapz(LED_UV(:,1), LED_UV(:,2));
LED_UV (:,2) = LED_UV(:,2) .*UV_intensity_multiplier; %should be in w/nm

%Creates relative intensity spectrum for Blue LED from Gaussian curve
lumen=45.7; %with current near 0.35A
LED_BB= guass_estimate(475,20);

%Retrieves photopic band that aligned with Blue LED band
photopic_band = photopic(photopic(:,1)>LED_BB(1,1) & photopic(:,1)<LED_BB(end,1),:);
LED_BB_interp=interp1(LED_BB(:,1),LED_BB(:,2),photopic_band(:,1)); % aligns data sets

%Calculates absolute Radiant flux Spectrum
Bblue_intensity_multiplier = (lumen) ./ (683*trapz(photopic_band(:,1),
LED_BB_interp.*photopic_band(:,2)));
LED_BB (:,2) = LED_BB(:,2) .*Bblue_intensity_multiplier; %should be in w/nm

%Retrieves Blue spectrum portion of White LED and converts it to scaled intensity
lumen = 140;
filename='white_blue_v2_spectrum.csv';
LED_W_B =csvread(filename,1,0,[1 0 66 1]);

photopic_band = photopic(photopic(:,1)>LED_W_B(1,1) & photopic(:,1)<LED_W_B(end,1),:);

LED_W_interp=interp1(LED_W_B(:,1),LED_W_B(:,2),photopic_band(:,1));
blue_intensity_multiplier=(lumen.*0.0426)./(683*trapz(photopic_band(:,1),LED_W_interp.*photopic
_band(:,2)));
LED_W_B (:,2) = LED_W_B(:,2) .*blue_intensity_multiplier; %should be in w/nm

%Retrieves yellow LED portion of white LED and converts it to scaled intensity
filename='white_yellow_v2_spectrum.csv';
LED_W_Y =csvread(filename,1,0,[1 0 136 1]);
photopic_band = photopic (photopic(:,1)>LED_W_Y(1,1) & photopic(:,1)<LED_W_Y(end,1),:);
LED_W_interp=interp1(LED_W_Y(:,1),LED_W_Y(:,2),photopic_band(:,1));
yellow_intensity_multiplier=(lumen.*0.9574)./(683*trapz(photopic_band(:,1),
LED_W_interp.*photopic_band(:,2)));
LED_W_Y (:,2) = LED_W_Y(:,2) .*yellow_intensity_multiplier; %should be in w/nm
LED_W= [LED_W_B(1:end-1,:);LED_W_Y];

%Retrieves Green LED portion of curve and converts it to scaled intensity
filename='true_green_spectrum.csv';
LED_G = csvread(filename,1,0,[1 0 150 1]);
lumen=143;
```

```
photopic_band = photopic (photopic(:,1)>LED_G(1,1) & photopic(:,1)<LED_G(end,1),:);
LED_G_interp=interp1(LED_G(:,1),LED_G(:,2),photopic_band(:,1));
green_intensity_multiplier= (lumen) ./ (683*trapz(photopic_band(:,1),
LED_G_interp.*photopic_band(:,2)));
LED_G (:,2) = (LED_G(:,2) .*green_intensity_multiplier); %should be in w/nm

%Number of Components
N_W_LEDS=9; %number of white LEDS
N_G_LEDS=4; %number of green LEDS
N_UV_LEDS=2; %number of UV LEDS
N_QTH=0.25; %Quartz tungsten lamp
N_B_LEDS=1; %number of Royal Blue LED
N_BB_LEDS=1;%number of Blue Leds
N_R_LEDS=3;%number of Red LEDs

%Add Absolute Spectrum Radiant Flux of all components
totalLED = combineSpectrum(LED_W,LED_UV,N_W_LEDS,N_UV_LEDS);
totalLED = combineSpectrum(totalLED,LED_G,1,N_G_LEDS);
totalLED = combineSpectrum(totalLED,LED_B,1,N_B_LEDS);
totalLED = combineSpectrum(totalLED,LED_BB,1,N_BB_LEDS);
totalLED = combineSpectrum(totalLED,LED_R,1,N_R_LEDS);

%Calculate the totalIrradiance assuming all radiant flux is emitted on test
%area
AREA=0.02; %Test area in m^2
totalIrradiance=totalLED;
totalIrradiance(:,2) = totalLED(:,2)./AREA;
totalIrradiance=combineSpectrum(totalIrradiance,QTH,1,N_QTH); %Add the irradiance of QTH

%Plots the ASTM Solar Reference Spectrum
figure (1);
tiledlayout(2,1)
nexttile
plot(Solar_reference(:,1),Solar_reference(:,2));
grid on;
axis([0 4 0 2300]);
title('Linear Plot of Spectral Irradiance @ AM0');
xlabel('Wavelength (\mum)');
ylabel('Spectral Irradiance (W/m^2 -\mum )');
nexttile
loglog(Solar_reference(:,1),Solar_reference(:,2));
grid on;
title('log-log Plot of Spectral Irradiance @ AM0');
xlabel('Wavelength (\mum)');
ylabel('Spectral Irradiance (W/m^2 -\mum )');

%Plots Irradiance Spectrum of LEDs
```

```
figure (2);
plot(Solar_reference(:,1).*1000,Solar_reference(:,2)./1000);
hold on
plot(LED_W(:,1),(LED_W(:,2).*N_W_LEDS)./AREA);
hold on
plot(LED_G(:,1),(LED_G(:,2).*N_G_LEDS)./AREA);
hold on
plot(LED_UV(:,1),(LED_UV(:,2).*N_UV_LEDS)./AREA);
hold on
plot(LED_BB(:,1),(LED_BB(:,2).*N_B_LEDS)./AREA);
hold on
plot(LED_B(:,1),(LED_B(:,2).*N_B_LEDS)./AREA);
hold on
plot(LED_R(:,1),(LED_R(:,2).*N_R_LEDS)./AREA);
hold on
plot(QTH(:,1),QTH(:,2).*N_QTH);
hold off

grid on;
axis([400 1100 0 2.5]);
title('Plot of Spectral Irradiance @ AM0 with Light Sources');
xlabel('Wavelength (nm)');
ylabel('Spectral Irradiance (W/m^2 /nm )');

%Creates graph with ASTM waveband sections color coded
figure(3);
plot(Solar_reference(:,1).*1000,Solar_reference(:,2)./1000);
hold on
axis([400 1100 0 2.5]);
section=totalIrradiance(totalIrradiance(:,1)>=300 & totalIrradiance(:,1)<=400,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
%fprintf('300-400 %.1f\n',((LED_sum)/f).*100);

hold on
section=totalIrradiance(totalIrradiance(:,1)>=400 & totalIrradiance(:,1)<=500,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
fprintf('400-500 %.1f\n',((LED_sum)/total).*100);

hold on
section=totalIrradiance(totalIrradiance(:,1)>=500 & totalIrradiance(:,1)<=600,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
fprintf('500-600 %.1f\n',((LED_sum)/total).*100);

hold on
```

```
section=totalIrradiance(totalIrradiance(:,1)>=600 & totalIrradiance(:,1)<=700,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
fprintf('600-700 %.1f\n',((LED_sum)/total).*100);

hold on
section=totalIrradiance(totalIrradiance(:,1)>=700 & totalIrradiance(:,1)<=800,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
fprintf('700-800 %.1f\n',((LED_sum)/total).*100);

IR_axis = 800:1:1500;
QTH_interp=interp1(totalIrradiance(:,1),totalIrradiance(:,2),IR_axis, 'spine');
IR = [IR_axis.' QTH_interp.'];
hold on
section=IR(IR(:,1)>=800 & IR(:,1)<=900,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
fprintf('800-900 %.1f\n',((LED_sum)/total).*100);

hold on
section=IR(IR(:,1)>=900 & IR(:,1)<=1100,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
fprintf('900-1100 %.1f\n',((LED_sum)/total).*100);

hold on
section=IR(IR(:,1)>=1100 & IR(:,1)<=1200,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
%fprintf('1100-1300 %.1f\n',((LED_sum)/total).*100);

hold on
section=IR(IR(:,1)>=1200 & IR(:,1)<=1300,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
%fprintf('1200-1300 %.1f\n',((LED_sum)/total).*100);

hold on
section=IR(IR(:,1)>=1300 & IR(:,1)<=1400,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
%fprintf('1200-1300 %.1f\n',((LED_sum)/total).*100);

hold on
section=IR(IR(:,1)>=1400 & IR(:,1)<=1500,:);
LED_sum=trapz(section(:,1),section(:,2));
```

```
area(section(:,1),section(:,2));
%fprintf('1200-1300 %.1f\n',((LED_sum)/total).*100);

hold on
plot(Solar_reference(:,1).*1000,Solar_reference(:,2)./1000);

hold off
grid on;
title('Plot of AM0 Spectral Match Groups');
xlabel('Wavelength (nm)');
ylabel('Spectral Irradiance (W/m^2 /nm )');
```

[C.2.1] Spectral Match and Non-Uniformity Matlab Script

```
%Using Values from Spectral_Matchv1, the intensity and flux can be calculated
%interchangeably. This Script places LED lambertian light sources 118mm
%above test area of 2500mm^2.
clc;
clear;
close all;

%Assigning test area and distance from
LENGTH = 0.05;
AREA = LENGTH.^2;
x=0:LENGTH/10:LENGTH;
y=0:LENGTH/10:LENGTH;
DISTANCE =0.118;

LED_W_Positions = [
   0 0 DISTANCE;
   0.035 0 DISTANCE;
   0.035 0.035 DISTANCE;
   0.035 -0.035 DISTANCE;
   -0.035 0.035 DISTANCE;
   -0.035 -0.035 DISTANCE;
   -0.035 0 DISTANCE;
   0 -0.035 DISTANCE;
   0 0.035 DISTANCE
   ];
sum = zeros(length(x),length(x));
for i = 1:9
   [X Y LED] = LED_irradiance(LED_W_Positions(i,:),0.2835,LENGTH, 120);
   sum =sum+LED;
end
W= trapz(y,trapz(x,sum,2));
W=W./AREA;
% figure (1);
% nexttile;
% surf(X,Y,sum);
```

```
% view(2);
% colorbar
% xlabel('Cubesat X Coordinate m');
% ylabel('Cubesat Y Coordinate m');
% zlabel('Irradiance W/m^2');
% zlim([180 200]);

% figure(2);
LED_DP_Positions = [
    -0.00551 0 DISTANCE
];
sumBuffer = zeros(length(x),length(x));
for i = 1:1
   [X Y LED] = LED_irradiance(LED_DP_Positions(i,:),0.21867,LENGTH, 120);
   sumBuffer =sumBuffer+LED;
end
DP= trapz(y,trapz(x,sumBuffer,2));
DP=DP./AREA;
% surf(X,Y,sumBuffer);
% colorbar
% view(2);
% xlabel('Cubesat X Coordinate m');
% ylabel('Cubesat Y Coordinate m');
% zlabel('Irradiance W/m^2');

sum = sum+sumBuffer;

% figure(3);
LED_G_Positions = [
   0.02183 0.01863 DISTANCE;
   0.02183 -0.01863 DISTANCE;
   -0.02183 0.01863 DISTANCE;
   -0.02183 -0.01863 DISTANCE;
];
sumBuffer = zeros(length(x),length(x));
for i = 1:4
   [X Y LED] = LED_irradiance(LED_G_Positions(i,:),0.089,LENGTH, 120);
   sumBuffer =sumBuffer+LED;
end
G= trapz(y,trapz(x,sumBuffer,2));
G=G./AREA;
% surf(X,Y,sumBuffer);
% colorbar
% view(2);
% xlabel('Cubesat X Coordinate m');
% ylabel('Cubesat Y Coordinate m');
% zlabel('Irradiance W/m^2');

sum = sum+sumBuffer;

% figure(4);
LED_R_Positions = [
   0.035 -0.01398 DISTANCE;
```

```
    -0.01207 -0.035 DISTANCE;
    -0.035 0.02356 DISTANCE;
];
sumBuffer = zeros(length(x),length(x));
for i = 1:3
    [X Y LED] = LED_irradiance(LED_R_Positions(i,:),0.08976,LENGTH, 120);
    sumBuffer =sumBuffer+LED;
end
R= trapz(y,trapz(x,sumBuffer,2));
R=R./AREA;
% surf(X,Y,sumBuffer);
% colorbar
% view(2);
% xlabel('Cubesat X Coordinate m');
% ylabel('Cubesat Y Coordinate m');
% zlabel('Irradiance W/m^2');

sum = sum+sumBuffer;

% figure(5);
LED_UV_Positions = [
    0.01313 0.01313 DISTANCE;
    0.01313 -0.01313 DISTANCE;
    -0.01313 0.01313 DISTANCE;
    -0.01313 -0.01313 DISTANCE;
];
sumBuffer = zeros(length(x),length(x));
for i = 1:3
    [X Y LED] = LED_irradiance(LED_UV_Positions(i,:),0.1273,LENGTH, 120);
    sumBuffer =sumBuffer+LED;
end
UV= trapz(y,trapz(x,sumBuffer,2));
UV=UV./AREA;
% surf(X,Y,sumBuffer);
% colorbar
% view(2);
% xlabel('Cubesat X Coordinate m');
% ylabel('Cubesat Y Coordinate m');
% zlabel('Irradiance W/m^2');

sum=sum+sumBuffer;

% figure(6);
LED_V_Positions = [
    -0.01313 0.02123 DISTANCE;
    0.01313 -0.02123 DISTANCE;
];
sumBuffer = zeros(length(x),length(x));
for i = 1:2
    [X Y LED] = LED_irradiance(LED_V_Positions(i,:),0.1089,LENGTH, 165);
    sumBuffer =sumBuffer+LED;
end
V= trapz(y,trapz(x,sumBuffer,2));
```

```
V=V./AREA;
% surf(X,Y,sumBuffer);
% colorbar
% view(2);
% xlabel('Cubesat X Coordinate m');
% ylabel('Cubesat Y Coordinate m');
% zlabel('Irradiance W/m^2');

sum =sum+sumBuffer;

% figure(7);
LED_C_Positions = [
   -0.035 -0.00551 DISTANCE;
   0.00551 0.035 DISTANCE;
   0.035 -0.029 DISTANCE;
];
sumBuffer = zeros(length(x),length(x));
for i = 1:3
   [X Y LED] = LED_irradiance(LED_C_Positions(i,:),0.1344,LENGTH, 120);
   sumBuffer =sumBuffer+LED;
end
C= trapz(y,trapz(x,sumBuffer,2));
C=C./AREA;
% surf(X,Y,sumBuffer);
% colorbar
% view(2);
% xlabel('Cubesat X Coordinate m');
% ylabel('Cubesat Y Coordinate m');
% zlabel('Irradiance W/m^2');

sum=sum+sumBuffer;

% figure(8);
LED_B_Positions = [
   %%
   %
   %  PREFORMATTED
   %  TEXT
   %
   0.00551 0 DISTANCE;

];
sumBuffer = zeros(length(x),length(x));
for i = 1:1
   [X Y LED] = LED_irradiance(LED_B_Positions(i,:),0.072188,LENGTH, 120);
   sumBuffer =sumBuffer+LED;
end
B= trapz(y,trapz(x,sumBuffer,2));
B=B./AREA;
% surf(X,Y,sumBuffer);
% colorbar
% view(2);
% xlabel('Cubesat X Coordinate m');
```

```
% ylabel('Cubesat Y Coordinate m');
% zlabel('Irradiance W/m^2');

sum=sum+sumBuffer;

figure(9);
surf(X,Y,sum);
cbar =colorbar;
cbar.Label.String = 'Irradiance W/m^2';

view(2);
xlabel('Cubesat X Coordinate (m)');
ylabel('Cubesat Y Coordinate (m)');
zlabel('Irradiance W/m^2');
title('Irradiance on CubeSat Surface with 0.118m Distance');

max = max(max(sum));
min=min(min(sum));
fprintf('The Non-Uniformity is %.1f%%\n',((max-min)./(max+min)).*100);

figure (10);
plot(LED_C_Positions(:,1),LED_C_Positions(:,2),'cX','Markersize',20);
hold on
plot(LED_DP_Positions(:,1),LED_DP_Positions(:,2),'X','Markersize',20);
hold on
plot(LED_G_Positions(:,1),LED_G_Positions(:,2),'gX','Markersize',20);
hold on
plot(LED_R_Positions(:,1),LED_R_Positions(:,2),'rX','Markersize',20);
hold on
plot(LED_UV_Positions(:,1),LED_UV_Positions(:,2),'X','Markersize',20);
hold on
plot(LED_V_Positions(:,1),LED_V_Positions(:,2),'mX','Markersize',20);
hold on
plot(LED_W_Positions(:,1),LED_W_Positions(:,2),'kX','Markersize',20);
hold on
plot(LED_B_Positions(:,1),LED_B_Positions(:,2),'bX','Markersize',20);
hold off
axis equal;
axis([-0.07/2 0.07/2 -0.07/2 0.07/2])
legend('Cyan','Deep Blue', 'Green', 'Red', 'UV','Violet','White','Blue');
title('LED Bulb Placement on Panel');
xlabel('X position (m)');
ylabel('Y position (m)');


%The following code is similar to Spectral_Match_v1 to check whether the irradiance  spectrum on the
%testing area is still ASTM Grade. There is also an inclusion of two LED colors: violent & Cyan
%*****************************************************************************************
%Retrieves Solar Spectrum and creates a matrix with the visible spectrum
filename='ASTM_SolarIrradiance_AM0.csv';
Solar_reference = csvread(filename,1,0,[1 0 1697 1]);
total=trapz(Solar_reference(:,1),Solar_reference(:,2));
slot = Solar_reference(Solar_reference(:,1)>0.4 & Solar_reference(:,1)<0.5,:);
```

```
first = trapz(slot(:,1),slot(:,2));
slot = Solar_reference(Solar_reference(:,1)>0.5 & Solar_reference(:,1)<0.6,:);
second = trapz(slot(:,1),slot(:,2));
slot = Solar_reference(Solar_reference(:,1)>0.6 & Solar_reference(:,1)<0.7,:);
third = trapz(slot(:,1),slot(:,2));
slot = Solar_reference(Solar_reference(:,1)>0.7 & Solar_reference(:,1)<0.8,:);
fourth = trapz(slot(:,1),slot(:,2));
slot = Solar_reference(Solar_reference(:,1)>0.8 & Solar_reference(:,1)<0.9,:);
fifth = trapz(slot(:,1),slot(:,2));
slot = Solar_reference(Solar_reference(:,1)>0.9 & Solar_reference(:,1)<1.1,:);
six = trapz(slot(:,1),slot(:,2));

%Retrieves Blue LED portion of curve and converts it to scaled intensity
filename='white_blue_v2_spectrum.csv';
LED_W_B =csvread(filename,1,0,[1 0 66 1]);

blue_intensity_multiplier = (W.*0.2368) ./ (trapz(LED_W_B(:,1), LED_W_B(:,2)));
LED_W_B (:,2) = LED_W_B(:,2) .*blue_intensity_multiplier; %should be in w/nm

%Retrieves yellow LED portion of curve and converts it to scaled intensity
filename='white_yellow_v2_spectrum.csv';
LED_W_Y =csvread(filename,1,0,[1 0 136 1]);

yellow_intensity_multiplier = (W.*0.7632) ./ (trapz(LED_W_Y(:,1), LED_W_Y(:,2)));
LED_W_Y (:,2) = LED_W_Y(:,2) .*yellow_intensity_multiplier; %should be in w/nm

LED_W= [LED_W_B(1:end-1,:);LED_W_Y];


%Reads Deep Blue from data
filename='GD_CSXPM1_14_20160712_spectrum.csv';
LED_B = csvread(filename,1,0,[1 0 66 1]);

B_intensity_multiplier = DP ./ trapz(LED_B(:,1), LED_B(:,2));
LED_B (:,2) = LED_B(:,2) .*B_intensity_multiplier; %should be in w/nm

%Getting Far red spectrum from data
filename='far_red_spectrum.csv';
LED_R = csvread(filename,1,0,[1 0 136 1]);

red_intensity_multiplier = R ./ trapz(LED_R(:,1), LED_R(:,2));
LED_R (:,2) = LED_R(:,2) .*red_intensity_multiplier; %should be in w/nm

%creates an estimate for the UV LED using a guasing
LED_UV= guass_estimate(405,20);
UV_intensity_multiplier = UV ./ trapz(LED_UV(:,1), LED_UV(:,2));
LED_UV (:,2) = LED_UV(:,2) .*UV_intensity_multiplier; %should be in w/nm

%creates an estimate for the Violet LED using a guasing
LED_V= guass_estimate(425,20);
V_intensity_multiplier = V ./ trapz(LED_V(:,1), LED_V(:,2));
LED_V (:,2) = LED_V(:,2) .*V_intensity_multiplier; %should be in w/nm
```

```
%creates an estimate for the Cyan LED using a guasing
LED_C= guass_estimate(490,25);
C_intensity_multiplier = C ./ trapz(LED_C(:,1), LED_C(:,2));
LED_C (:,2) = LED_C(:,2) .*C_intensity_multiplier; %should be in w/nm

%creates an estimate for the royal blue using a guasing
LED_BB= guass_estimate(475,20);
Bblue_intensity_multiplier = DP ./ (trapz(LED_BB(:,1),LED_BB(:,2)));
LED_BB (:,2) = LED_BB(:,2) .*Bblue_intensity_multiplier; %should be in w/nm

%Retrieves Green LED portion of curve and converts it to scaled intensity
% filename='GREEN_LED.csv';
% LED_G = csvread(filename,1,0,[1 0 100 1]);
filename='true_green_spectrum.csv';
LED_G = csvread(filename,1,0,[1 0 150 1]);
green_intensity_multiplier= G ./ (trapz(LED_G(:,1),LED_G(:,2)));
LED_G (:,2) = (LED_G(:,2) .*green_intensity_multiplier); %should be in w/nm

%Retrieves IR data
filename='QTH_V1.csv';
QTH = csvread(filename,1,0,[1 0 211 1]);
QTH(:,2)= QTH(:,2)/4; %Converting mW values to W values and multiply by 25 due to emitting closer

totalIrradiance = combineSpectrum(LED_W,LED_UV,1,1);
totalIrradiance = combineSpectrum(totalIrradiance,LED_G,1,1);
totalIrradiance = combineSpectrum(totalIrradiance,LED_B,1,1);
totalIrradiance = combineSpectrum(totalIrradiance,LED_BB,1,1);
totalIrradiance = combineSpectrum(totalIrradiance,LED_R,1,1);
totalIrradiance = combineSpectrum(totalIrradiance,LED_V,1,1);
totalIrradiance = combineSpectrum(totalIrradiance,LED_C,1,1);
totalIrradiance(:,2)=totalIrradiance(:,2).*1.5;
totalIrradiance=combineSpectrum(totalIrradiance,QTH,1,1);
figure (1);
plot(Solar_reference(:,1).*1000,(Solar_reference(:,2)./1000));
hold on
plot(LED_W(:,1),(LED_W(:,2)));
hold on
plot(LED_G(:,1),(LED_G(:,2)));
hold on
plot(LED_UV(:,1),(LED_UV(:,2)));
hold on
plot(LED_BB(:,1),(LED_BB(:,2)));
hold on
plot(LED_B(:,1),(LED_B(:,2)));
hold on
plot(LED_R(:,1),(LED_R(:,2)));
hold on
plot(LED_V(:,1),(LED_V(:,2)));
hold on
plot(LED_C(:,1),(LED_C(:,2)));
hold on
plot(QTH(:,1),QTH(:,2));
hold off
```

```
legend({'ASTM Solar Reference','White','Green','UV','Deep Blue','Blue','Red','Violet','Cyan','QTH'})
grid on;
axis([400 1100 0 2.5]);
title('Spectral Irradiance of Each component');
xlabel('Wavelength (nm)');
ylabel('Spectral Irradiance (W/m^2 /nm )');

figure (2);
plot(Solar_reference(:,1).*1000,Solar_reference(:,2)./1000);

hold on
axis([400 1100 0 2.5]);
section=totalIrradiance(totalIrradiance(:,1)>=300 & totalIrradiance(:,1)<=400,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
%fprintf('300-400 %.1f\n',((LED_sum)/f).*100);

hold on
section=totalIrradiance(totalIrradiance(:,1)>=400 & totalIrradiance(:,1)<=500,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
fprintf('400-500 %.1f\n',((LED_sum)/first).*100);

hold on
section=totalIrradiance(totalIrradiance(:,1)>=500 & totalIrradiance(:,1)<=600,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
fprintf('500-600 %.1f\n',((LED_sum)/second).*100);

hold on
section=totalIrradiance(totalIrradiance(:,1)>=600 & totalIrradiance(:,1)<=700,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
fprintf('600-700 %.1f\n',((LED_sum)/third).*100);

hold on
section=totalIrradiance(totalIrradiance(:,1)>=700 & totalIrradiance(:,1)<=800,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
fprintf('700-800 %.1f\n',((LED_sum)/fourth).*100);

IR_axis = 800:1:1500;
QTH_interp=interp1(totalIrradiance(:,1),totalIrradiance(:,2),IR_axis, 'spine');
IR = [IR_axis.' QTH_interp.'];
hold on
section=IR(IR(:,1)>=800 & IR(:,1)<=900,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
fprintf('800-900 %.1f\n',((LED_sum)/fifth).*100);

hold on
section=IR(IR(:,1)>=900 & IR(:,1)<=1100,:);
```

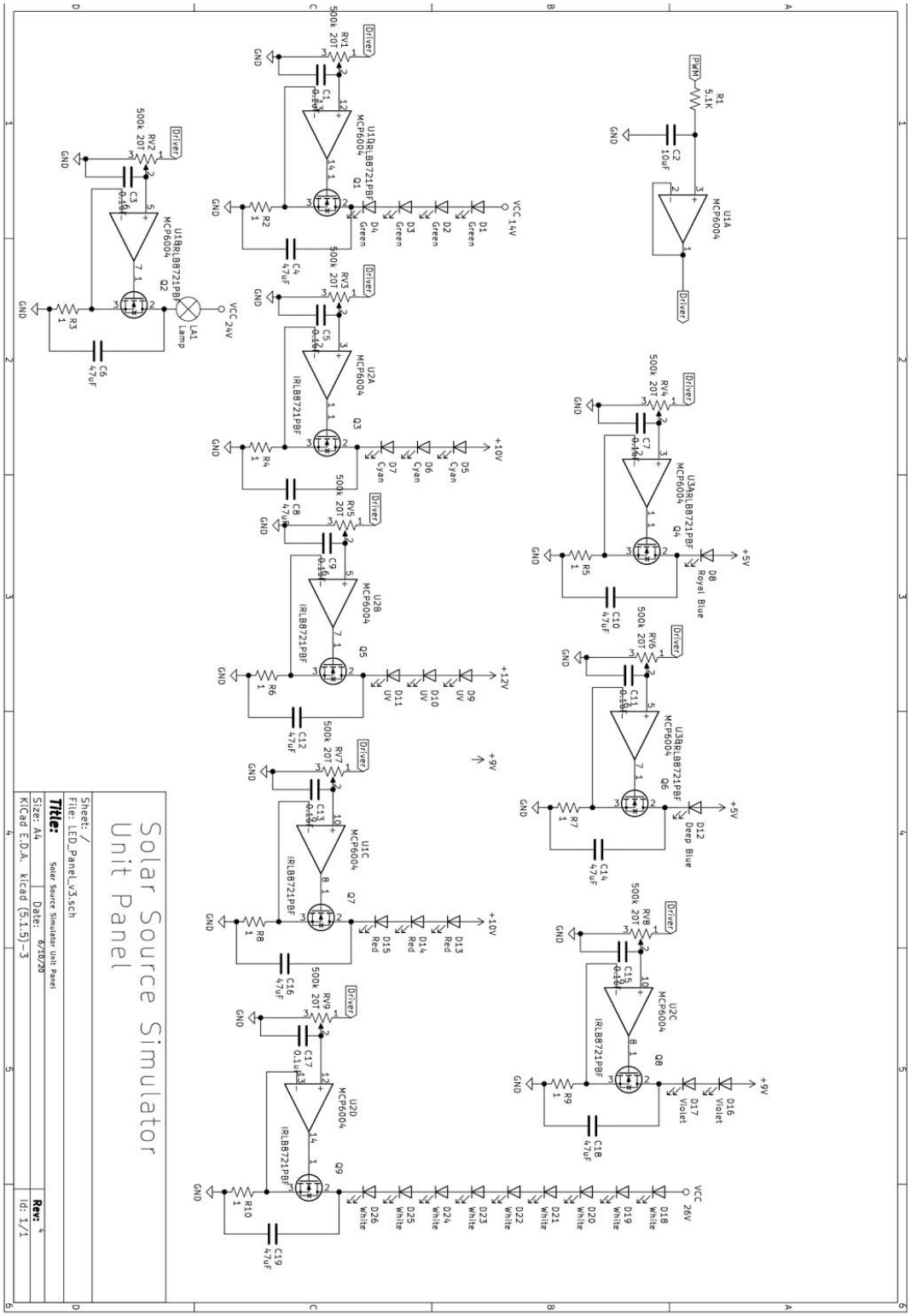
```
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
fprintf('900-1100 %.1f\n',((LED_sum)/six).*100);

hold on
section=IR(IR(:,1)>=1100 & IR(:,1)<=1200,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
%fprintf('1100-1300 %.1f\n',((LED_sum)/total).*100);

hold on
section=IR(IR(:,1)>=1200 & IR(:,1)<=1300,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
%fprintf('1200-1300 %.1f\n',((LED_sum)/total).*100);

hold on
section=IR(IR(:,1)>=1300 & IR(:,1)<=1400,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
%fprintf('1200-1300 %.1f\n',((LED_sum)/total).*100);

hold on
section=IR(IR(:,1)>=1400 & IR(:,1)<=1500,:);
LED_sum=trapz(section(:,1),section(:,2));
area(section(:,1),section(:,2));
%fprintf('1200-1300 %.1f\n',((LED_sum)/total).*100);

hold on
plot(Solar_reference(:,1).*1000,Solar_reference(:,2)./1000);
legend({'ASTM Solar Reference'});
hold off
grid on;
title('Plot of AM0 Spectral Match Groups');
xlabel('Wavelength (nm)');
ylabel('Spectral Irradiance (W/m^2 /nm )');
```

[C.2.2] LED Irradiance Model

```
%Function LED_irradiance has 4 inputs and 3 outputs. The 4 inputs are
%position, intensity, length of test area, and apex angle. With these input
%the Irradiance model calculates the irradiance on the designated surface.
%The output is an X,Y meshgrid and surface being the irradiance
function [x_surf y_surf surface] = LED_irradiance( position,intensity,length, apex)
   [x_surf y_surf] = meshgrid(0:length/10:length);
   M=-1.*(log(2)./log(cosd(apex/2)));
   position = position + [length./2 length./2 0];
   R=sqrt((x_surf-position(1)).^2 + (y_surf-position(2)).^2 + position(3).^2);
   surface = ((position(3).^(M+1)).*intensity)./(R.^(M+3));
```

```
end
```

[C.3] Light Source Driver Circuit



[C.4] Light Sources Part Table

Color	Manufacturer	Part Number
White 4000k	Opulent	LST1-01F05-5070-01
Far Red	Opulent	LST1-01F06-FRD1-00
Deep Blue	Opulent	LST1-01F06-RYL1-00
Green	Opulent	LST1-01F06-RYL1-00
Cyan	Chanzon	10DGL-DZ-3W-490
Purple/Violet	Chanzon	10DGL-DZ-3W-425
UV	Chanzon	10DGL-DZ-3W-405
Royal Blue	Chanzon	10DGL-DZ-3W-440
QTH IR	Newport	6319 20W

[C.5] Non-Uniformity Experiment Matlab Script

```
% This script looks at an image file and creates a grayscale intensity
% matrix. This grayscale intensity matrix can show the different of light
% intensity at certain points on the image. The image is divided by 100
% squares and the non-uniformity is calculated
clc;
clear;
close all;

%Read the image and converts it to an RGB matrix
P=imread('Uniformity_Experiment.jpg');

Gray=rgb2gray(P); %Converts image to gray scale to find relative intensity
[row column]=size(Gray);
myMeanFunction=@(block_struct) mean2(block_struct.data); %Assigns mean2 as a function for
image processing
blockMean=blockproc(Gray,[floor(row/10) ,floor(column/10)], myMeanFunction); %Uses mean2 for
1/10 of the image

%Finds the Maximum and Minimum Image Intensity on each block.The
```

```
%Non-uniformity is printed
max = max(max(blockMean));
min=min(min(blockMean));
fprintf('The Non-Uniformity is %.1f%%\n',((max-min)./(max+min)).*100);

%Display the relative image intensity
figure(1);
surf(blockMean);
view(2);
colorbar
xlabel('X-Axis Block');
ylabel('Y-Axis Block');
zlabel('Average Image Intensity');
title('Average experimental uniformity');

figure(2);
imshow(Gray);
```

Appendix D: References

[D.1] National Centers for Environmental Information and Ncei, “The World Magnetic Model - Accuracy, limitations, magnetic poles and error model,” World Magnetic Model - Model Limitations. [Online]. Available: https://www.ngdc.noaa.gov/geomag/WMM/limit.shtml. [Accessed: 13-Mar-2020].

[D.2] ASTM E927-19, Standard Classification for Solar Simulators for Electrical Performance Testing of Photovoltaic Devices, ASTM International, West Conshohocken, PA, 2019, www.astm.org

[D.3] Green, MA, Dunlop, ED, Hohl-Ebinger, J, Yoshita, M, Kopidakis, N, Ho-Baillie, AWY. Solar cell efficiency tables (Version 55). Prog Photovolt Res Appl. 2020; 28: 3– 15. https://doi.org/10.1002/pip.3228

[D.4] Jang, Sun Ho, and Moo Whan Shin. “Fabrication and Thermal Optimization of LED Solar Cell Simulator.” Current Applied Physics 10, no. 3 (2010): 537–39. https://doi.org/10.1016/j.cap.2010.02.035.

[D.5] A. M. Bazzi, Z. Klein, M. Sweeney, K. P. Kroeger, P. S. Shenoy and P. T. Krein, "Solid-State Solar Simulator," in IEEE Transactions on Industry Applications, vol. 48, no. 4, pp. 1195-1202, July-Aug. 2012, doi: 10.1109/TIA.2012.2199071.

[D.6] Taylor, Alma E.F. Illumination Fundamentals. Rensselaer Polytechnic Institute, n.d.

[D.7] “Useful Color Data.” RIT. Accessed June 8, 2020. https://www.rit.edu/cos/colorscience/rc_useful_data.php.

[D.8] Cool, Simon et al. “Development of a High Irradiance LED Configuration for Small Field of View Motion Estimation of Fertilizer Particles.” SENSORS 15.11 (2015): 28627–28645. Web.

[D.9] Adafruit USB Isolator - 100mA Isolated Low/Full Speed USB. [Online]. https://www.adafruit.com/product/2107 [Accessed: 21-Mar-2020].

[D.10] OSRAM Opto Semiconductors, “High Performance SSL Family LED,” GW CS8PM1.PM datasheet, March. 2020

[D.11] Khanh, T., Bodrogi, P., Trinh, Q., & Winkler, H. (2015). LED lighting : technology and perception . Wiley-VCH. 168

[D.12] NASA General Environmental Verification Standard, GSFC-STD-7000, https://standards.nasa.gov/standard/gsfc/gsfc-std-7000, April 2019

[D.13] PolySat Helmholtz Cage, Cal Poly San Luis Obispo, 2017 https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=1437&context=mesp

[D.14] CubeSat Space Qualification, February 2020, https://www.endurosat.com/space-qualification/