



---

# 3D Printing by Selective Laser Sintering

---

Group 12

Luis  
Carrillo

Michael  
McMahon

Michael  
Rogatinsky

Daniel  
Valledor

Electrical  
Engineering

Photonic  
Engineering

Computer  
Engineering

Electrical  
Engineering

# Table of Contents

1.	Executive Summary.....	1
2.	Project Description .....	2
2.1	Project Motivation .....	2
2.2	Goals and Objectives .....	2
2.3	Requirements Specifications.....	3
2.4	House of Quality Analysis.....	4
2.5	Overview Block Diagram .....	5
3.	Project Research and Part Selection .....	7
3.1	Existing Similar Projects and Products.....	7
3.2	Possible Architectures.....	9
3.3	Relevant Technologies.....	11
3.3.1	3D printing technologies.....	11
3.3.2	Material Extrusion .....	11
3.3.2.1	Vat Polymerization .....	12
3.3.2.2	Powder Bed Fusion.....	13
3.3.3	AC – DC Converter .....	14
3.3.4	Voltage Regulator .....	15
3.3.5	Laser Scanning Technologies .....	16
3.3.5.1	Lens-based Scanning.....	16
3.3.5.2	Modulation and Mechanical-based Scanning.....	18
3.3.6	Electromechanical Linear Motion.....	19
3.3.7	Motors.....	19
3.3.8	Relevant Software.....	20
3.3.8.1	Rust.....	20
3.3.8.2	Python.....	22
3.3.8.3	C.....	25
3.3.8.4	C++ .....	27
3.3.9	Displays.....	28
3.3.10	Laser Source.....	29
3.4	Strategic Components and Part Selection .....	34
3.4.1	Mechanical Components.....	34
3.4.1.1	Threaded Rod and Flanged Nut.....	34
3.4.1.2	Smooth Guide Rod and Linear Bearing .....	35
3.4.1.3	Self -Manufactured Components.....	35
3.4.1.4	Stepper Motor .....	36
3.4.2	Electrical Components .....	37
3.4.2.1	Stepper Motor Driver.....	37
3.4.2.2	Laser Driver Operational Amplifier .....	38
3.4.2.3	Power MOSFET .....	39
3.4.2.4	AC/DC Converter .....	40
3.4.2.4.1	Transformer.....	40
3.4.2.4.2	Full Bridge Rectifier.....	41
3.4.2.4.3	Capacitor .....	42
3.4.2.4.4	Power Cord.....	42

3.4.2.5	DC/DC Regulator .....	43
3.4.2.6	Display.....	44
3.4.2.7	Microcontroller.....	44
3.4.2.7.1	Hardware Class .....	44
3.4.2.7.2	Instruction Set Architecture .....	45
3.4.2.7.3	Microcontroller.....	45
3.4.2.7.4	Final Microcontroller Design Decision.....	46
3.4.3	Optical Components.....	46
3.4.3.1	Laser Selection .....	46
3.4.3.2	Lens Selection.....	48
3.4.3.3	Material.....	50
3.4.3.4	Beam Shaping.....	51
4.	Standards and Design Constraints .....	54
4.1	Standards.....	54
4.1.1	NEMA ICS 16-2001 .....	54
4.1.2	ASME B1.5-1997 .....	54
4.1.3	Optical Standards.....	54
4.1.4	NEMA Power Cords.....	55
4.1.5	Software Standards .....	56
4.2	Design Constraints.....	59
4.2.1	Economic and Time Constraints.....	59
4.2.2	Health and Safety constraints.....	59
4.2.2.1	Electrical Safety.....	60
4.2.2.2	Laser Safety .....	60
4.2.2.3	Safety of Moving Parts .....	61
4.2.3	Environmental Constraints.....	62
4.2.4	Manufacturability Constraints .....	62
5.	Project Hardware and Software Design.....	64
5.1	Mechanical Design.....	64
5.1.1	Enclosure.....	64
5.1.2	Reservoir .....	64
5.1.3	Plate Subsystem .....	65
5.1.4	Sweeper Subsystem .....	67
5.1.5	X-Y Track System .....	68
5.1.6	Final Assembly .....	69
5.2	Electrical Design.....	70
5.2.1	Stepper Motor Driver Circuit.....	71
5.2.2	Laser Driver Circuit .....	73
5.2.3	AC-DC Converter .....	76
5.2.4	DC-DC Converter .....	78
5.2.4.1	5V Regulator.....	79
5.2.4.2	9V Regulator.....	83
5.2.5	Microcontroller .....	84
5.3	Optical Design .....	90
5.3.1	Laser Power .....	90
5.3.2	Collimating Lens.....	92

5.3.3	Focusing Lens .....	93
5.3.4	Beam Shaping .....	93
5.3.5	Laser Module .....	93
5.3.6	Simulation Optimization.....	95
5.4	Software Design.....	96
5.4.1	Operating System.....	96
5.4.2	Programming Language.....	97
5.4.3	Final Software Design.....	99
5.5	Possible Future Capabilities.....	99
6.	Prototype Construction and Coding .....	100
6.1	PCB Vendor and Assembly .....	100
6.1.1	Laser Driver PCB.....	101
6.1.2	Stepper Motor Driver PCB .....	102
6.1.3	Power Module PCB .....	103
6.1.4	MCU Module PCB .....	104
6.2	Mechanical System Construction .....	105
6.3	Final Coding Plan .....	106
6.3.1	General Code Structure.....	106
6.3.2	PlanerScan Coding Structure .....	108
6.3.3	ZSystem Coding Structure.....	110
6.3.4	LaserManager Coding Structure .....	112
6.3.5	PrintModel Coding Structure.....	113
6.3.6	GUI Coding Structure .....	115
7.	Project Prototype Testing Plan .....	117
7.1	Hardware Testing.....	117
7.1.1	Stepper Motor Testing .....	117
7.1.2	Linear Actuator Testing .....	119
7.1.3	Laser Driver Testing .....	123
7.1.4	Kerf Testing .....	125
7.1.5	Powder Delivery Testing .....	126
7.1.6	Microcontroller Board Testing .....	128
7.1.7	AC-DC Converter Testing .....	129
7.1.8	DC/DC Regulators Testing.....	131
7.1.9	Laser Diode Testing.....	132
7.1.10	Lens Testing.....	135
7.1.11	Laser Module Testing .....	136
7.1.12	Laser Sintering Testing.....	138
7.2	Software Testing .....	140
7.2.1	PlanerScan Testing.....	141
7.2.2	ZSystem Testing .....	142
7.2.3	LaserManager Testing .....	143
7.2.4	Final Software Testing.....	145
8.	Administrative Content.....	146
8.1	Project Milestone .....	146
8.2	Budget and Finance Discussion .....	147
8.3	User Instruction Manual .....	148

8.3.1	Software Setup .....	148
8.3.2	Print File Setup .....	148
8.3.3	Printer Setup and Use.....	148
9.	Appendix .....	vi
9.1	Appendix A – Copyright Permissions .....	vi
9.2	Appendix B – Miscellaneous .....	vi
9.3	Appendix C – References .....	vii

## Table of Figures

Figure 2.4-1: House of Quality .....	5
Figure 2.5-1: Overview Block Diagram .....	6
Figure 3.2-1: X-Y Scanner Concept .....	10
Figure 3.2-2: X-Y Track System Concept .....	11
Figure 3.3-1: Fused Deposition Modeling.....	12
Figure 3.3-2: Stereolithography .....	13
Figure 3.3-3: Selective Laser Sintering .....	14
Figure 3.3-4 Pre-objective Scanning.....	17
Figure 3.3-5 Galvanometer Scanner .....	18
Figure 3.3-6 RECI Carbon Dioxide laser. ....	30
Figure 3.3-7 Energy-level diagram in a CO <sub>2</sub> laser.....	31
Figure 3.3-8 Laser Diode Characteristics .....	33
Figure 3.4-1 Laser diode collimation by an aspheric lens. ....	49
Figure 3.4-2 Comparison of Powder Creation Technologies .....	50
Figure 3.4-3 A Cylindrical Lens Pair Circularizes and Collimates a Beam .....	52
Figure 3.4-4 Comparison of Beam Shaping Methods .....	53
Figure 4.1-1 NASA NPR 7123.1 Overview [7].....	58
Figure 5.1-1: Plate System 3D Model.....	66

Figure 5.1-2: Sweeper Subsystem.....	68
Figure 5.1-3: X-Y Track Subsystem.....	69
Figure 5.1-4: Final Assembly .....	70
Figure 5.2-1: Full-Scale Current [8] .....	71
Figure 5.2-2: DRV8825 Schematic.....	72
Figure 5.2-3: Laser Driver Fundamental Schematic .....	74
Figure 5.2-4: Laser Driver Final Schematic .....	76
Figure 5.2-5: Eagle Schematic of AC/DC Converter .....	77
Figure 5.2-6: 185E16 Transformer Wiring from Datasheet .....	78
Figure 5.2-7: WEBENCH Design for 5V Regulator .....	79
Figure 5.2-8: 5V Regulator Eagle Schematic .....	82
Figure 5.2-9: 9V Regulator Schematic .....	83
Figure 5.2-10: 3.3V Regulator for RP2040 Power Input.....	84
Figure 5.2-11: RP2040 with Decoupling Capacitors.....	85
Figure 5.2-12: Flash Memory Design for RP2040 .....	86
Figure 5.2-13: RP2040 External Crystal Circuit.....	86
Figure 5.2-14 USB-to-UART circuit.....	89
Figure 5.2-15 ATMEGA2560 Schematic .....	89
Figure 5.3-1 L-I Curve of the PLPT9 450 LB-E Diode.....	91
Figure 5.3-2 Schematic of the Optical Layout.....	92
Figure 5.3-3 Exploded View of Endurance’s Laser Module .....	94
Figure 5.3-4 Laser Module Housing.....	95
Figure 5.3-5 Zemax Simulation Results .....	95
Figure 5.3-6 Optimized Optical Layout.....	96
Figure 6.1-1: Final PCBs Mounted.....	100

Figure 6.1-2: Laser Driver Module PCB Layout .....	101
Figure 6.1-3: Final Laser Driver Module PCB .....	101
Figure 6.1-4: Stepper Motor Driver Module PCB Layout.....	102
Figure 6.1-5: Final Stepper Motor Driver Module PCB.....	103
Figure 6.1-6: Power Supply Module PCB Layout.....	103
Figure 6.1-7: Final Power Supply Module PCB.....	104
Figure 6.1-8: Microcontroller Module PCB Layout.....	105
Figure 6.1-9: Final Microcontroller Module PCB .....	105
Figure 6.2-1: (Left) Final Mechanical Construction with Side Panel Open. (Right) Final Mechanical Construction with Side Panel Open. ....	106
Figure 6.3-1 General Code Structure .....	108
Figure 6.3-2 PlanerScan.....	110
Figure 6.3-3 ZSystem.....	112
Figure 6.3-4 LaserManager.....	113
Figure 6.3-5 PrintModel.....	115
Figure 7.1-1: Stepper Motor Breadboard Testing.....	118
Figure 7.1-2: “Smiley” Model .....	121
Figure 7.1-3: XY Track System Results.....	122
Figure 7.1-4: Max Laser Driver Current.....	125
Figure 7.1-5: Confectioners’ Sugar Test Results.....	128
Figure 7.1-6: Iced Tea Mix Results .....	128
Figure 7.1-7: 6V Regulator 30ms Soft Start .....	132
Figure 7.1-8 Spectrum of Laser Diode .....	134
Figure 7.1-9 Layout of Laser Module Lens Tests.....	135
Figure 7.1-10 Lens Test Results .....	136
Figure 7.1-11 Printed Lens Mounts .....	137

Figure 7.1-12 Final Laser Module Results .....	137
Figure 7.1-13 ‘Smiley’ Sintering Test Results .....	138
Figure 7.1-14 UCF Pegasus Sintering Test Results.....	139
Figure 7.1-15 Differences in Sintering Quality .....	139
Figure 7.1-16 High Speed Box Test Results.....	140
Figure 9.2-1: Stepper Motor Schematic.....	vi
Figure 9.2-2: DRV8825 Test Module.....	vi



*This page intentionally left blank.*

# 1. Executive Summary

Turning abstract thoughts into tangible products is the driving force of the human species. From the beginnings of ancient society, tools were created by means of carving or layering materials. In modern society, there are many methods to create the products used by billions of people, but these methods lack something unique and powerful: direct writing of materials in three-dimensions. This is the perfect method to create something; no milling, no forging – direct writing. These methods of 3D printing have been around for only the last 40 years, yet they've opened a door full of rich possibilities.

The most consumer-driven 3D printers print thermoplastic materials layer by layer on an x-y stage. Spools of material are fed through a heated extruder which melts the plastic as the extruder head writes the design on the print bed. Eventually, the pattern is transferred entirely to the print bed, and minimal post processing is needed. For these types of prints, supports are needed, and the materials are limited to thermoplastics.

Less consumer-driven types of 3D printing include Selective Laser Sintering (SLS). The SLS process for 3D printing begins with a print bed. The print bed is covered with a thin layer of powder. When powder particles are fed enough energy, they can sinter, or fuse, together. A laser system is used to deliver energy to the powder in a predefined pattern which sinters the powder together, resulting in a sintered shape on the powder bed. The powder bed is then lowered, and another layer of powder is rolled on top of the previously sintered layer. The process repeats, where laser exposure of the powder bed layer by layer creates a product within the enclosed build volume.

SLS printers have many advantages over traditional extruder-based thermoplastic 3D printers, such as no need for supports, wide material selection, and superior resolution. Unfortunately, SLS printers are not widely available to the 3D printing hobbyist, with the cost of most printers exceeding \$10,000. This project, 3D Printing by Selective Laser Sintering, seeks to design and create a low-cost SLS printer, while retaining accuracy and performance of more costly printers, only at the expense of decreased throughput. The project will also seek to expand the material capabilities of SLS printers by including not only thermoplastics, but also sugar.

This will be accomplished by the construction and assembly of multiple subsystems. A mechanical subsystem will provide the printer with the necessary accuracy to create powder layers of 120 microns in height. An optical subsystem will deliver energy to the powder bed with a spot size of 175 microns. Together, the mechanical and optical subsystems will deliver scanning speeds of greater than 30 mm/s, an improvement on the cheapest SLS printer available. These subsystems will be powered by intelligent electrical design with excellent processing control. Altogether, the SLS printer will cost only 10% of the most affordable printer available on the market at \$600, bringing the capabilities of SLS printers to the comfort of hobbyist's homes.

## 2. Project Description

### 2.1 Project Motivation

Transforming data from one medium to another has been a process known to humans for thousands of years. Printing in the form of woodblock on textiles or paper was known and widely used throughout East Asia since 220 AD. The printing press, a revolutionary device which could print thousands of pages per day, was created in 1440. The art of lithography, which is pattern transfer and replication from one material medium to another, has been around since 1796. Of course, digital printing, where computer data is converted to paper, has manifested as laser printing, inkjet printing, and other types, since the early 1900s.

The nearly 2000-year history of printing has one common theme: all printing materials were two dimensional in nature. With the advent of computers, and thus computer aided design, there was a need to develop printing methods based in three dimensions that didn't involve extensive manufacturing, such as metal machining or forging. First discussed in the 1940s and 1950s, 3D printing came to fruition as additive manufacturing in the 1980s.

As is expected with any new technology, the 3D printers in the mid-1980s cost hundreds of thousands of dollars. Nearly 40 years later, 3D printing is an affordable table-top hobby enjoyed by millions of people worldwide.

Today, 3D printing takes on many forms. Enthusiasts can purchase printers which extrude heated thermoplastic into arbitrary shapes for less than \$1000. These are the prototypical 3D printers, but they have limitations. Some of these limitations include necessitating support structures for the builds, as well as constraining users to only printing models with thermoplastic materials, such as acrylonitrile butadiene styrene (ABS) or polylactic acid (PLA).

3D printing with different materials can be done with other types of printers. A subset of laser additive manufacturing, selective laser sintering (SLS) is a 3D printing method which uses a laser to selectively fuse together, or sinter, a powder into arbitrary shapes. One of the biggest advantages of this method is its ability to sinter any powdered material; the limitations arise only from thermal properties of the powder itself. Unfortunately, the cheapest SLS printers can cost upwards of over \$18,000, still significantly out of the common enthusiast's price range. Further, these systems are usually only intended for single-material type use. Multi-material SLS printers can exceed \$100,000.

### 2.2 Goals and Objectives

Prior designs of SLS printers, as mentioned before, are prohibitively expensive. These systems have lasers exceeding 50 watts which are a significant contribution to the cost. Further, these systems use heated beds to ease the sintering process and include automated material retrieval. The available SLS printers also use high-speed galvanometers to scan at speeds in excess of 500mm/s. Our system will be designed to maintain high resolution but with decreased throughput.

To build a low-cost implementation of SLS, we will use a low-power laser assembly. One company, Sintratec, has developed one of the cheapest SLS systems at \$24,000 using a 5W blue laser diode as the laser source. Our optical system will also use a blue laser diode of similar power. Using blue light as the thermal sintering source is beneficial over CO<sub>2</sub> or Nd-YAG as the blue light has higher absorption in a wider range of materials and is much cheaper than CO<sub>2</sub> or Nd-YAG layouts. However, the previously mentioned Sintratec system is only designed for one type of material. We will design our optical system to be robust in its material-sintering capabilities by designing a laser driver to operate the laser at predefined material-specific parameters.

Laser scanning systems vary widely in their performance, whether by resolution, speed, or scanning area. Initially, our system was going to feature an x-y scanning system inspired by galvanometers. However, galvanometers are difficult to use, are expensive, and have very high maximum scan speeds. They also use two mirrors, which, for high-power applications, can also raise the cost due to the protective coatings. Because of this, we will develop an x-y scanning system based on track-and-rail movements that will scan the laser across the powder bed. This removes the need for high-resolution motors, high-power capacity mirrors, and a scanning lens to correct for field curvature of the scan area, all of which contribute to excessive costs. While much cheaper, this comes at a reduction of scan speed and product throughput.

Systems on the market can achieve layer heights within 70-100 microns. These layer heights work well with our selection of blue laser diodes as its penetration depth exceeds this. Our powder delivery system will ensure this same level of micron height by using high resolution stepper motors. Micro-stepping is achieved by both the stepper motor, the motor's driver, and the amount of threads on a supporting rod, and is not a considerable addition to the total cost.

The printer's processing will be done using a microcontroller. The microcontroller is responsible for reading the g-code and controlling the sub-systems accordingly. SPI, a communication protocol designed by Motorola, will be used to transmit data to the user interface screen.

## 2.3 Requirements Specifications

The SLS project's engineering specifications arise from motivations including ease of use, budget, and small size. Some specifications are very similar to more expensive printers on the market. Some of these competitive specifications include the bed print area and build height, material capabilities, and resolution (determined by the 175-micron spot size and the material size).

However, there are some specifications that had to be worse than what is currently on the market to make way for decreased budget. Specifically, the scanning speed is very slow. Comparable to the cheapest available SLS printer, it is faster. This is due to the motors used in the x-y scanning architecture. But, when compared with the more expensive industrial printers, the scanning speed is abysmal. This is due to the scanning type used; industrial scanners use galvanometric scanning, which is a high-speed, high-precision, and

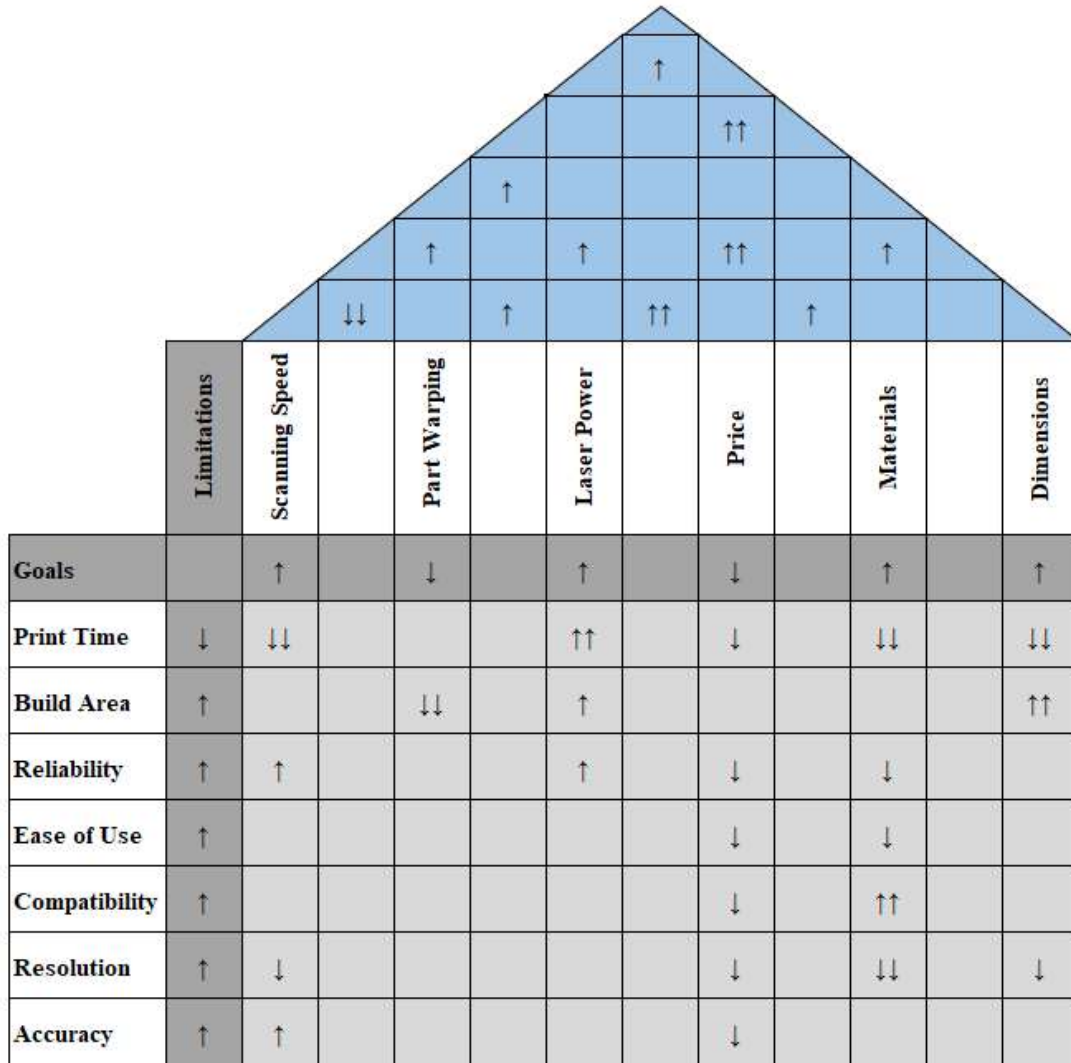
high-cost scanning method. Table 2.3-1 shows an overview of the requirements specifications for the SLS printer.

**Table 2.3-1: SLS Printer Requirements Specifications**

Specifications	
Print Bed Area	10 cm x 10 cm $\pm$ 5%
X-Y Build Area	9 cm x 9cm
Print Height	9 cm
Power Source	120V Wall Power
User Interface	Local Host Server
Motor Torque	40 - 45 N*cm
Laser Diode Output Power	2W - 5W
Laser Diode Wavelength	447 $\pm$ 15 nm
Beam Focal Length	75 $\pm$ 10 mm
X-Y Scanning Speed	$\geq$ 30 mm/s
Material Type	Sugar, Polyamides, other TPE/TPU
Laser Spot Size	< 200 $\mu$ m

## 2.4 House of Quality Analysis

The House of Quality is a visualization tool to help people understand the relations between different parts of the project and associated goals. As seen in Figure 2.4-1 some of the main goals of our project are the print time, build area, reliability, ease of use, compatibility, resolution, and accuracy. The main limiting factors to contend with will be the scanning speed, part warping, laser power, price, materials, and the projects dimensions. And of course, as shown on the roof of the house, those limiting factors can also impact one another.



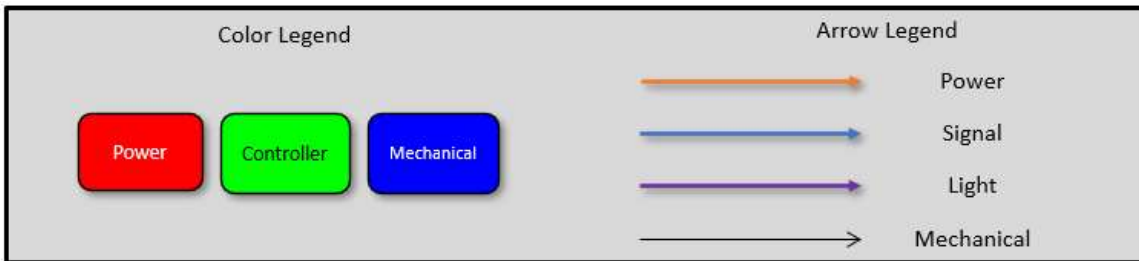
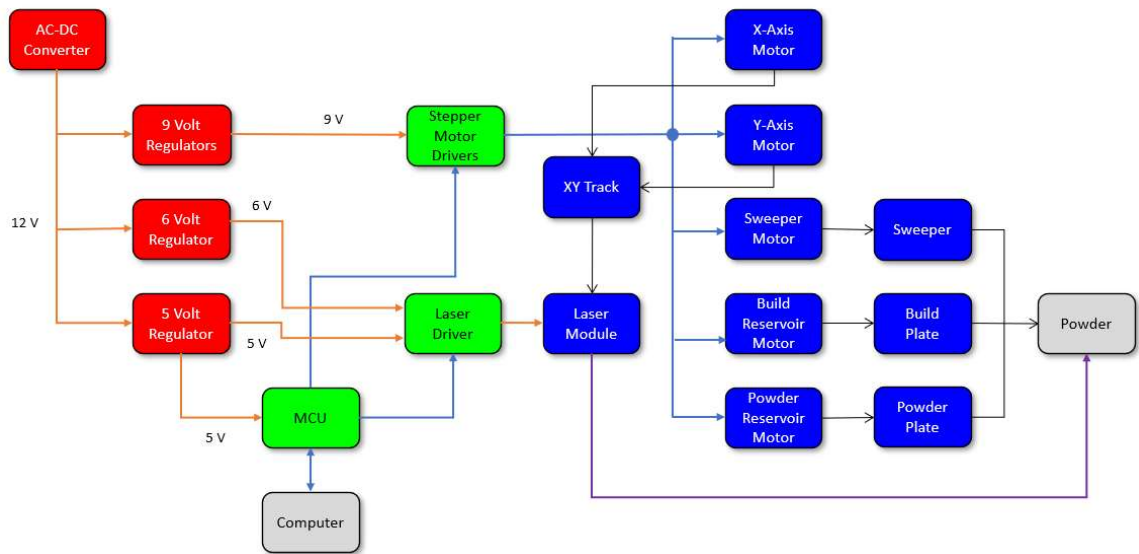
↑↑	Strong Positive Correlation
↑	Positive Correlation
	No Correlation
↓	Negative Correlation
↓↓	Strong Negative Correlation

**Figure 2.4-1: House of Quality**

## 2.5 Overview Block Diagram

The SLS printer project is broken down into several different subsystems. Figure 2.5-1 shows the block diagram overview for subsystem responsibility. Luis Carillo, an electrical engineer, will be responsible for the power supply and MCU design. The power system design includes an AC-DC converter and multiple DC-DC converters to power the subsystem. Michael McMahon, a photonics engineer, will be responsible for the laser

module design and powder material selection. The laser module design includes the laser diode, lens selection and thermal design. Michael Rogatinsky, a computer engineer, will be responsible for the system software, user interface, and enclosure design. Daniel Valledor, an electrical engineer, will be responsible for the electromechanical subsystems and the laser driver design. The electromechanical systems include the x-y track, print and powder bed, and powder sweeper. The subsystems are driven by motors therefore motor drivers will also be designed.



**Figure 2.5-1: Overview Block Diagram**

## 3. Project Research and Part Selection

### 3.1 Existing Similar Projects and Products

Products available on the market regarding selective laser sintering include printers, kits, and components such as laser modules which are commonly used for laser engraving or laser cutting. SLS printing is not as consumer friendly as other types of thermoplastic printers, such as extrusion printers, stereolithography (SLA), or fused deposition modelling. This is due to the components and operation of SLS printers: they are inherently more dangerous than other printers because of the high-powered laser assembly; the occupation volume of one machine is quite large; SLS powder is still very expensive; and the process of dealing with the powder itself is very involved. In this project, we seek to decrease some of these complexities which limit SLS printers' entrance into the consumer market.

The cheapest SLS printer available on the market costs \$5,827. The Sintratec Kit, as it is called, has a volume of  $9 \times 9 \times 9 \text{ cm}^3$ , a laser speed of 5-20 mm/s, and a layer height of 100-150 microns. It offers customizable parameters, namely laser speed and layer height, and supports multiple materials. This is one of the few workbench-type SLS printers on the market.

Perhaps the next most inexpensive SLS printer, the LISA BASIC by Sinterit costs only \$6990. The affordability is due to its inexpensive laser source and modest resolution: a 5W infrared laser diode operating at 808 nm with a minimum wall thickness of 400 microns, which might imply the minimum spot size is 400 microns. The accuracy of the scanning system is 50 microns, with a variable layer height between 75 to 175 microns. Its maximum print volume is somewhat larger than Sintratec Kit is, at  $11 \times 16 \times 13 \text{ cm}^3$ . While the size of the printer is not necessarily large, measuring an average 20 inches across in each dimension, it weighs 96.8 pounds, making this product somewhat unsuitable for tabletop usage. Some nice features of the Sinterit LISA BASIC include a 4-inch resistive LCD screen for user interactions, as well as Wi-Fi and USB connectivity. Because the laser is invisible in the infrared, the protection class is only Class 1. While on the inexpensive side, the LISA BASIC is still quite costly, and certainly there are areas to be improved upon.

Formlabs is a 3D printing company that has excelled since its inception in 2011. They are known for their consumer-driven products that bring high quality prints at an affordable price. Recently, Formlabs has burst into the SLS market with their own SLS printer called the Fuse 1. It costs significantly more than their SLA products at \$18,499 but maintains the company's high quality. The laser is an Ytterbium fiber operating at 1065 nm. The optical power output is considerable at 10W and delivers a spot size of 200 microns. The printer itself is capable of 110-micron layer heights and a build volume of  $16.5 \times 16.5 \times 30 \text{ cm}^3$ . The scanning system used in the Fuse 1 is a custom-made galvanometer scanner. User interfaces include a 10.1" touchscreen, boasting a resolution of 1200 x 800, with connectivity options of Wi-Fi, Ethernet, or USB 2.0. Because the Fuse 1 is very large (it weighs 251.3 pounds without the build chamber or powder, with a recommended



operational footprint of 145.5 x 149.5 x 167.5 cm<sup>3</sup>), it is certainly not an accessible tabletop machine. It also is incapable of printing with other materials except Nylon 12, leaving the user with limited design capabilities.

Approaching industrial grade SLS printers, 3D Systems' ProX SLS 6100 has a hefty price of over \$100,000. Weighing 3000 pounds, the ProX is equipped with a 100 W CO<sub>2</sub> laser with lightning-fast scan speeds of 12,700 mm/s. It has a large build volume of 38.1 x 33 x 46 cm<sup>3</sup> with minimum layer heights of 80 microns. For the high price, the ProX's main feature is its high throughput. Material selections available to use with the ProX are limited to the company's own thermoplastic selection called Duraform, which are essentially derivatives of Nylon 11 and Nylon 12. Further contributions to the cost include a 95% refresh rate on powder material. Refresh rate for SLS printers is the amount of powder able to be reused after a print is completed. Variables that can reduce refresh rate include particle size, accuracy of the laser scanning system, and temperature control. For comparison, the prior two printers (LISA BASIC and Fuse 1) have a refresh rate of 75% and 70% respectively. To determine the refresh rate, the remaining unsintered powder is sifted through a sieve of particular size to get rid of any particles that have not stuck to the main print but are too big to use in another sintering process. In other words, the refresh rate can be used as a figure of merit for the efficiency of a printer.

The EOS P 770 is a \$500,000 SLS printer with fascinating features. It uses two 70 W CO<sub>2</sub> coupled with a custom-made scanning module to scan at 20,000 mm/s. The idea of using multiple lasers to increase throughput is seen not only in 3D printing but also in areas of fabrication in general. Micro and nanofabrication with electron beam lithography is traditionally slow, but architectures have been built which split the original beam into thousands or even millions of beams. A yet-to-be-released SLS printer, the QLS 350 by Nexa3D, will be equipped with four 100 W CO<sub>2</sub> lasers to increase throughput. The EOS P 770's minimum layer height is 60 microns in a build volume of 70 x 38 x 58 cm<sup>2</sup>. It can use up to 10 different commercially available thermoplastics. The printer also features spot pyrometers to constantly measure the temperature of the beam, which is fed to a feedback loop to maintain consistent temperature for an optimal sintering environment. Further, the accuracy of the scanning system is so high as to ensure the overlap between sintered line scans is nearly zero; one part of the scanning system is an F-theta lens, which was a possible part to include in our printer if it weren't for the cost. Because this is a true industrial SLS printer, the size and weight are massive: 189 x 189 x 118 inches at 5071 pounds. Additional optional features include an attachable CoolDown® station which ensures parts post-sintering are cooled in a way that promotes maximum structural integrity.

Table 3.1-1 below shows a comparison of the advertised printers as well as some others that were not discussed. From the table, it can be seen the layer thickness of the commercial printers is similar to the layer height of the project's specifications.

**Table 3.1-1 SLS Printer Process Parameters on the Market**

Printer	Cost	Material	Laser Type (nm)	Laser Power (W)	Layer Thickness (μm)	Scan Speed (mm/s)
ProX SLS 6100	\$100,000	Polyamide	CO <sub>2</sub> λ=10640	100	80-150	500
Sinterit Lisa Pro	\$17,000	Polyamide Polyurethane	Diode λ=808	5	75-175	N/A
Formlabs Fuse 1	\$32,000	Polyamide	Fiber λ=1065	10	110	N/A
EOS P 110 Velocis	\$175,000	Alumide Polyamide Polystyrene Polyetheramide	CO <sub>2</sub> λ=10640	30	60-120	500

## 3.2 Possible Architectures

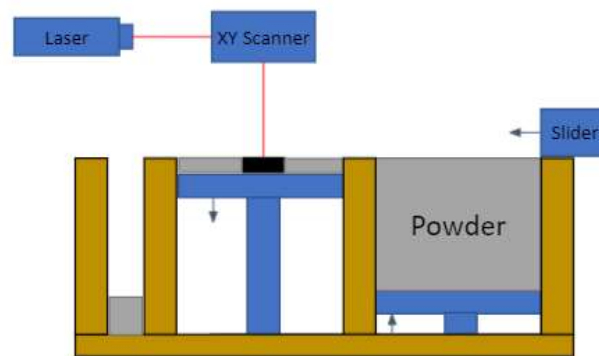
The main functionality of a SLS printer is sintering powder on top of previously sintered material to achieve a three-dimensional object. Given the main functionality, we had to figure out how to deliver the powder on top of previously sintered powder, and how to move the laser beam to accurately sinter the powder. We researched different ways design engineers have previously answered these two questions and deduced which solutions will work best for our application.

From our research, we found two methods to achieve powder delivery. The first method is directly placing powder over the previously sintered object and sintering it immediately. The method is used in a type of laser additive manufacturing process called Laser Metal Deposition (LMD). [1] The approach is also similar to the functionality of a conventional thermoplastic printer. There is added complexity when using this method since exact powder delivery is needed to achieve quality prints. There is also a physical limitation in part complexity using this method. Even though they use a very similar material delivery system, thermoplastic printers are able to print complex parts due to supports. Supports are low density structures that break away from the part once the print is complete. The slicing software creates a slight enough gap so that the plastic doesn't fully bond to the support. Due to the nature of laser sintering, sintered supports will not break away like thermoplastic supports.

The second method is delivering a full layer of powder to the whole laser sintering area for each layer. Once a layer is complete, the sintering area drops, and a new layer of powder covers the previous layer. Precise powder delivery is less of a priority when using this method. We also have the capability of creating very complex parts using this approach. The non-sintered powder that remains in every layer act as support material for the sintered 3D object. Relative to the first method, the second method wastes more powder, but it is

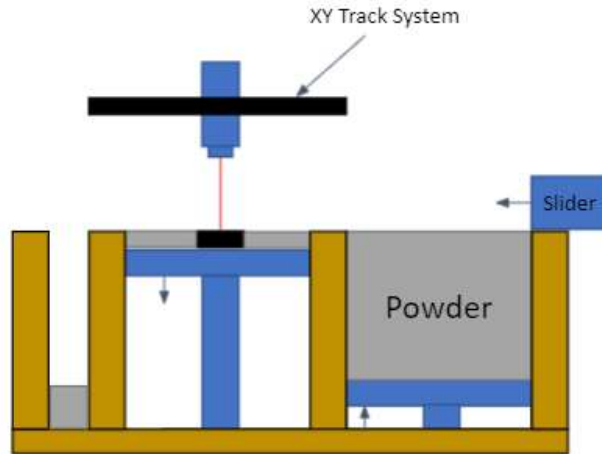
capable of printing complex part structures. In order for our design to compete with conventional thermoplastic printers, we will use the second method for powder delivery.

In order to create a successful three-dimensional print, we need to ensure the laser beam moves precisely. From research and team discussion, we came up with two methods to move the laser beam. The first method is using a x-y galvanometer scanner. A x-y galvanometer scanner works by rotating two mirrors to successfully move the beam in a cartesian plane. Galvanometer scanning can achieve high enough speeds to be used in persistence of vision applications. These scanners are very expensive, some ranging from \$1,000 to \$5,000. Therefore, we would apply the same scanning concept to something we can build. With the use of two stepper motors, a 3D printed bracket, and mirrors, we could build our own x-y scanning system. Figure 3.2-1 is a two-dimensional concept drawing of the x-y scanner and powder delivery system.



**Figure 3.2-1: X-Y Scanner Concept**

There are factors that go into designing an x-y scanner that adds complexity. The first factor is geometric translation to the print area. The microcontroller would need to make the proper trigonometric calculations to convert cartesian coordinates from the g-code to proper mirror angles. The second factor is making up for inconsistent beam lengths. As the scanner moves the mirrors, the beam changes length therefore changing the focal point. As the complexity might be too great for the scope of this project, we considered a second method of moving the laser beam. The second method consists of placing the laser module on an x-y track. X-Y track systems are used in many manufacturing technologies such as CNC machines and laser cutters. Using the X-Y track would eliminate the need for complex calculations. The X-Y track system would be able to work off of the g-code since it would be a one-to-one matching. Furthermore, the laser beam will keep a consistent length throughout travel. Figure 3.2-2 is a two-dimensional concept drawing of the x-y track and powder delivery system. In order to simplify the system and refrain from overdesigning, we will use the x-y track system to move the laser module.



**Figure 3.2-2: X-Y Track System Concept**

## 3.3 Relevant Technologies

### 3.3.1 3D printing technologies

Over the last 40 years additive manufacturing, more commonly known as 3D printing has developed at great pace. Industries have identified a plethora of benefits from its use and therefore accelerated the number of resources going towards 3D printing research and development. This has brought along a diverse number of additive manufacturing technologies based on entirely different concepts, where each type has a completely different set of features. Some key differences across technologies are materials used, process time length, accuracy, and cost. Given that our goal is to design a selective laser sintering printer affordable for hobbyists we will focus on discussing the 3D printing technologies that use widely available printing materials and have little to none post-processing. With this in mind, there are three groups of technologies separated by the operation concept that we will dive into. They are material extrusion, vat polymerization, and powder bed fusion.

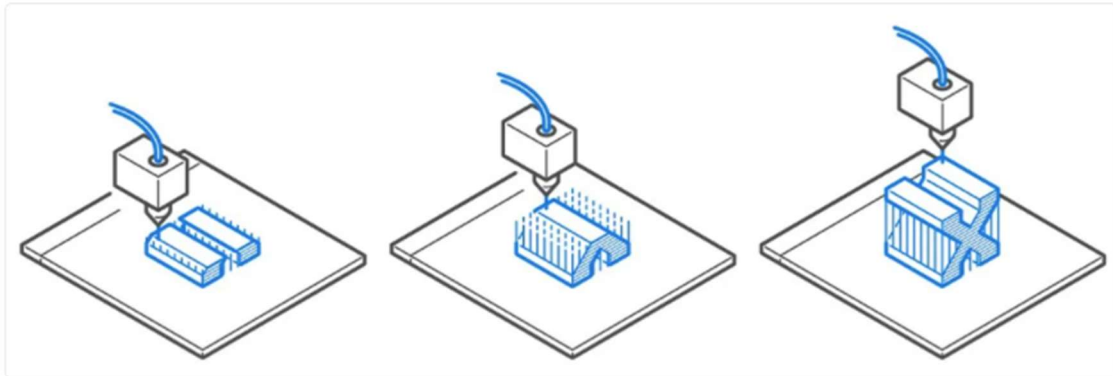
### 3.3.2 Material Extrusion

Starting with material extrusion, the main technology that falls into this category is Fused Deposition Modeling (FDM) also referred to as Fused Filament Fabrication (FFF). It is important to remark that this is the type of printer that is most commonly used, specially by our target audience. This is because it ranks well among all of the features that we deemed significant for hobbyist 3D printing, especially on materials used and cost. Figure 3.3-1 shows a simplified view of the printing process for this technology.

Regarding materials, FDM is compatible with a wide range of filaments. For the low costs FDM 3D printers' materials are limited to plastic filaments, however, this still includes popular materials like PLA, nylon, or carbon fiber. As far as cost, FDM printers are the cheapest option in the market for additive manufacturing with the lower end devices

starting under \$220. The reason behind FDM being the lowest priced option comes from its design simplicity.

The material extrusion concept is rather straightforward and can be easily executed. All it requires is a heating component connected to the nozzle that melts the material used as it is pushed through the extrusion head as well as a track that enables the extrusion head to move in three dimensions in order to place the material where needed. Typically, this is done layer by layer, meaning that it will place all required material in an X-Y plane before moving in the Z direction. However, it is also due to this mode of operation that the final result from a print might have visible layers.



**Figure 3.3-1: Fused Deposition Modeling**

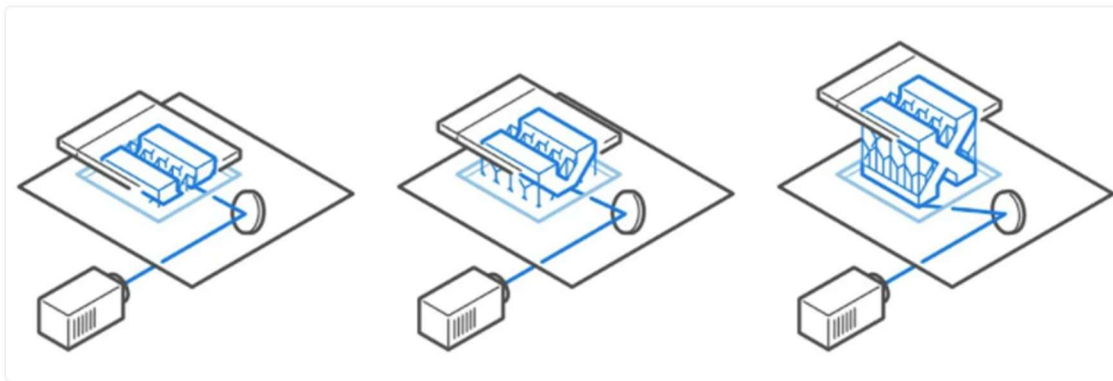
Depending on the product needs, FDM prints might often need to go through post-processing in order to achieve smoother textures. Additionally, when the object being printed has overhang sections, FDM printers need to include support structures. This adds to the printing time as well as post-processing given that those structures will need to be removed post print. Some FDM printers support special filaments for support structures made of material that dissolves on water or special solutions. Additive manufacturing technologies that do not require post-processing are outside of the typical hobbyist's budget. Making, FDM the best 3D printing technology by far at low price ranges.

### 3.3.2.1 Vat Polymerization

The next category of 3D printing technologies we will discuss is vat polymerization. The printers in this category are characterized by the use of resin and a light source as the building tool. Stereolithography (SLA) is a type of vat polymerization printer. It has a printing bed that is held by an elevator arm. In this case as seen in Figure 3.3-2, the print is done in the inferior side of the plate. At the start of the print the plate is lowered down until there is only about two tenths of a millimeter between itself and the bottom of the translucent bottom of the resin reservoir. This allows a laser beam to be directed to resin. This is where a galvanometer (a pair of carefully placed mirrors) takes care of adjusting the laser beam to cure or solidify the resin where needed to additively build the print as the plate rises. Just like FDM, some pieces with overhang might need support structures. Thus, post-processing is still required to remove such structures and because we are only

building with resin, is not possible to create support with other materials that might facilitate the removal of supports.

SLA printers have a similar price range to FDM, but it produces prints that are significantly smoother and more accurate. However, FDM remains the dominant 3D printer due first to popularity and exposure and second due to running costs. Even though you can have a FDM and an SLA printer of similar size and quality at the same price range, filaments used in FDM tend to be approximately half of the cost of the equivalent amount of printing resin. Since we are looking at them from a hobbyist point of view, this could have a great impact in the variety of printing material you could have. Additionally, the resin tank has to be replaced after printing a few liters of resin. This is because resin slowly tarnishes the bottom of the tank. This impacts the transparency of it and therefore degrades the effect of the laser beam. A new tank could cost around \$50. This type of expense is not present in FDM printers. Thus, by choosing FDM, users can more easily own a variety of color filaments as well as solid or flexible types.



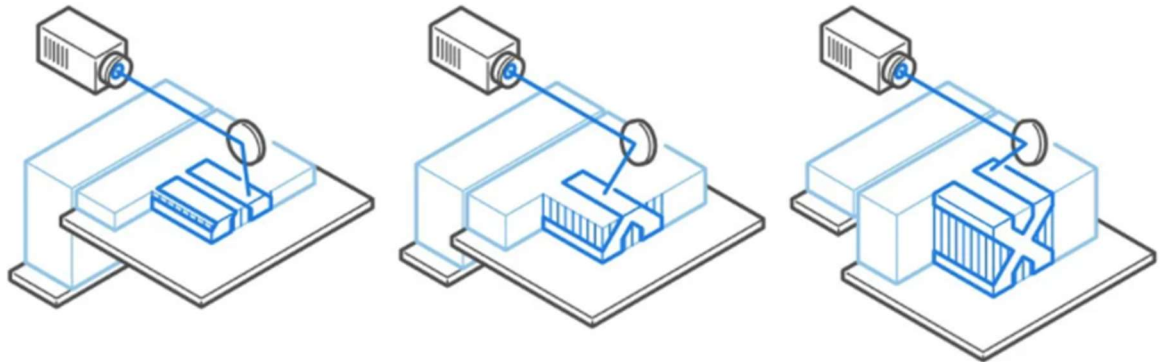
**Figure 3.3-2: Stereolithography**

### 3.3.2.2 Powder Bed Fusion

Lastly, we have the powder bed fusion. The printers in this category use a light source, typically some variation of a laser, to fuse powder particles of a specific material together. After fusing a layer of powder together, some kind of mechanism will apply another thin layer of powder on top from another plate containing the printing powder. This process is repeated until the print is done. Selective Laser Sintering (SLS) falls into this category. What makes it different from other powder bed fusion printers like Selective Laser Melting (SLM) is the type of material being used. SLS uses thermoplastics like nylon 6, or nylon 11, as well as ceramics. SLM is used on metals.

SLS printers use the same system as SLA to direct the laser beams into the printing material. Main difference is the type and power of the laser, material being used (powder vs resin), and the location of the laser system. In SLS printers, the laser is placed on top of the printing bed instead of under it. This change is key when looking at the benefits of SLS printing. Thinking back to SLA even though our print might be submerged in resin at points, when the plate continues to rise, the print stops being under resin and therefore

needs printed support structures. Because on SLS we are printing from the top and moving our print down as we add layers, it means that the unused powder stays filling the gaps of the print while it cools down, therefore working as an unattached support structure. This means that 100% of the printing time is spent on our print and not time is lost adding printed support. Additionally, even more time is saved when removing support structures only involves removing unfused powder. The extra powder can be recycled, meaning that no material is wasted on supports.



**Figure 3.3-3: Selective Laser Sintering**

Unfortunately, all of the benefits of SLS printing mentioned before, come at a significant price increase. While we said that FDM and SLA printers can be purchased for a few hundred dollars, SLS printers available start at and go well beyond \$6000. Thus, as mentioned in the goals and objectives section, we aim to make an affordable SLS printer.

### 3.3.3 AC – DC Converter

Due to internal resistance of non-ideal transmission lines, there will be power dissipated in them. In order to reduce these losses, power is transmitted at high voltages and low alternating currents (AC). This is why regular wall outlets deliver power at 120V 60Hz 15A. However, most, if not all electronics work with direct current (DC) power including our SLS printer. Thus, an AC/DC converter is necessary in order to successfully power our product.

There are two popular techniques used to not only turn the power in DC but also step it down. The first approach is the switching system, and the second option is the transformer system. Both of them use the same technique to stabilize the voltage at a single level. This is through the use of a full bridge rectifier and a capacitor. The full bridge will turn the negative side of the sinusoidal input into positive, while the capacitor will get charged and then attempt to maintain its voltage as the AC voltage decreases until the input starts charging the capacitor again. With correct capacitance values, the time constant will be large enough so that the voltage stays constantly at the peak of the wave.

The difference between technologies lies in whether this is done before or after stepping down the voltage. A switching AC/DC converter will have this stage right from the wall



input and then feed its high voltage output into a switching control circuit that uses PWM signals to step down the voltage depending on the duty cycle. This is the preferred option for most regular consumer adapters such as the ones needed to connect your phone or laptop to a wall outlet. The next option is to first take care of the step down and then convert into DC. This is done by using a transformer. The transformer will use the magnetic fields created in a coil due to the alternating current to induce current into the next coil with fewer wire turns to step down the voltage. Then this new sinusoidal wave with the desired amplitude will be fed into the bridge rectifier circuit with the capacitor to end up with the DC voltage. This option is not as popular mainly due to the weight and size of the transformer.

Additionally, other differences include that the transformer dissipates more heat than the switching circuit. However, the transformer does have the advantage of being a much simpler circuit, with less noise than the switching solution. Another relevant difference between the systems is that, since the rectification is happening at high voltage before the stepping down of the circuit, we must find components with adequate power ratings. This will often have a big impact on the cost of those components, making it cheaper to step down voltage as the first step.

### 3.3.4 Voltage Regulator

Given that our SLS has multiple power modules, it is likely that we will need different voltage levels. The required voltage for the computer module is 5V, our laser module requires 6V and our motors 9V. Thus, after converting the wall outlet AC 120V into a lower DC voltage, we need to use voltage regulators to create the different input levels that meet each powered module's needs. This is where voltage regulators come into place.

Just as AC/DC conversion, there are different technologies used to achieve our goal of producing different output levels. One of the common types of regulators that produce a DC output lower than the original DC input is a linear regulator. They are easy to use and do not require many external components. Additionally, some linear regulators are made so that they can be adjustable. Thus, depending on how they are connected, you can achieve different output levels with the same type of IC. The other type of regulators used are switching regulators. They have a much more complex internal circuit. This is due to the fact that they use a PWM signal to reduce the output into the average of the altered input voltage. This means that an inductor will be used in conjunction with a capacitor to make the output a constant voltage once again. Switching regulators can also be configurable to achieve different output voltages. This very different approach to regulation results in a drastic difference in efficiencies. Linear regulators can only be as efficient as the ratio of  $V_{out}/V_{in}$ . While switching regulators have efficiencies typically around 90%. In the next section of the report, we will be discussing part selection. Thus, we will make a pick between any of those technologies and explain the reasoning of the decision there.



## 3.3.5 Laser Scanning Technologies

SLS systems on the market make use of a variety of methods to scan the laser across the print bed. These scanning systems are usually high speed, but cheaper SLS systems can use slower, less expensive scanning architectures. This section will detail the existing scanning technologies and illustrate their pros and cons.

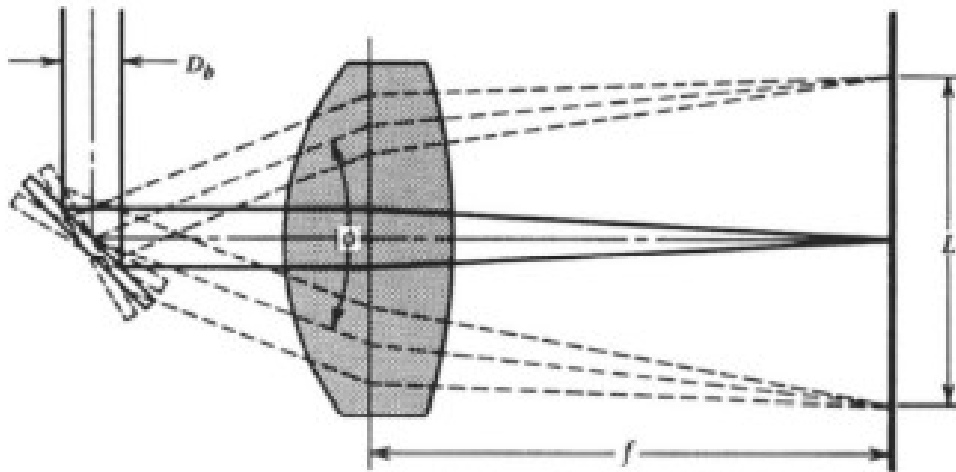
### 3.3.5.1 Lens-based Scanning

In lens-based scanning technologies, the laser beam is incident upon a focusing lens, which is then moved or rotated in a manner which directs the beam to trace a pattern over the print bed. In using the lens-based scanning, sometimes mirrors are used instead of lenses to manipulate where the beam falls on the print bed.

One example of lens-based scanning is objective scanning. In objective scanning, the focusing lens moves about the optical axis (central to the beam) which translates the laser beam. The center of the lens is a distance  $x$  from the beam axis, and by rotating the lens, a circle of radius  $x$  can be drawn at a focal distance away. This type of configuration is relatively simple, but the translator of the lens must be precise. Further, it must be precise in not one dimension but both  $x$  and  $y$ . Inherently, because the movement of the lens is responsible for the pattern on the print bed, the focal field at the print bed exhibits curvature. This type of scanning system also has a 1:1 direct translation from lens-shift to pattern dimensions drawn, which is useful in certain cases – not so much in SLS printing. If the patterns drawn in SLS were perfect circles, objective scanning would be the best option.

Another example of lens-based scanning is called post-objective scanning. In post-objective scanning, the collimated laser beam oriented along the optical axis is incident upon a focusing lens. In this configuration, the optical axis is oriented parallel to the print bed. The focusing lens begins to focus the light a distance  $f$ . Further along the optical axis, a rotating mirror is placed which deflects the laser beam at a distance  $d$  from the lens. The optical axis is translated  $90^\circ$ , now perpendicular to the print bed. The laser beam focuses on the print bed at a distance  $f - d$  and with a half-angle of  $90^\circ - \theta$ . This system configuration has some advantages. For instance, the sizes of the lens and mirrors need not be much larger than the beam size itself, making the overall scanning space cost very small. Another advantage is that the scan size in one direction is directly linear with the scanning angle at a length  $L = (f - d) \cdot \theta$ . This makes scan field computations quite simple. Unfortunately, the shape of the scan field using post-objective scanning is curved significantly, and at long distances  $L$  this can become a serious problem, as field curvature changes the spot size with increasing distances away from the center point.

To correct for field curvature, an architecture known as pre-objective scanning can be implemented, shown in Figure 3.3-4. As the name suggests, the scanning is achieved before focusing, allowing the use of a special lens to focus light off-center. The incident laser beam is collimated and is deflected in the same manner as in post-objective scanning. Once the collimated light is deflected off the mirror, it enters a focusing lens



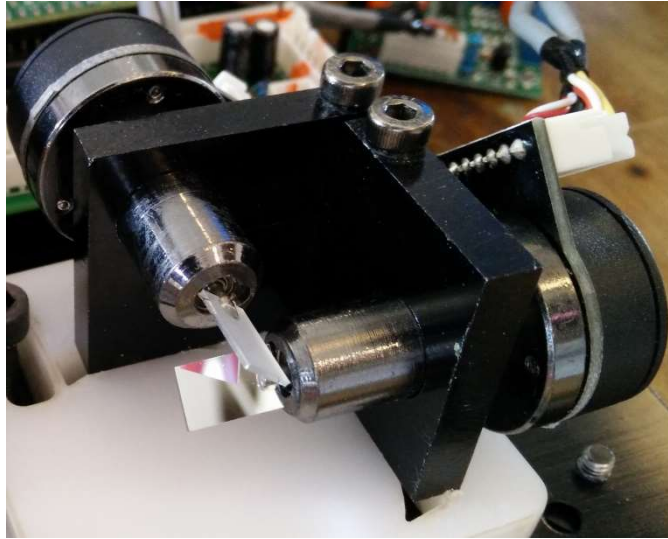
**Figure 3.3-4 Pre-objective Scanning**

which focuses the beam onto the print bed at one focal distance away. In this configuration, the length of the scan field in one dimension is no longer a linear function of the focal distance and the mirror deflection angle, so it is necessary to appropriately model the beam path for correct functioning. The added complexity, however, flattens the field curvature. While not ideally flat, it is much flatter than the post-objective scanning architecture.

To achieve a perfect zero-field-curvature scan, a type of pre-objective scanning known as telecentric scanning can be used. In this setup, the scanning mirror is placed one focal length away from a spherical lens, which is placed one focal length away from the scan field. This creates rays which all leave the focusing lens as parallel. A type of lens used in pre-objective scanning that eliminates field curvature due to entering the lens at long distances away from the center of the lens are called  $f - \theta$  lenses, or scanning lenses. These are ideal to use but cost thousands of dollars.

Achieving scanning in two dimensions requires a second scanning element. One scanning element scans in only the x or y direction, while the second scanning element scans in the other direction. This adds complexity to the system but is what the high-end SLS printers use.

The scanning elements in the high-end systems are based off the physical principles of galvanometers and boast high speed and high precision. Initially, the SLS design for this project was going to use a galvanometer-based scanning system such as the one shown in Figure 3.3-5. However, as mentioned before, the post-objective scanning configuration does not have a linear relationship with the scan field so the proper calibration would have been exceedingly difficult to achieve.



**Figure 3.3-5 Galvanometer Scanner**

### 3.3.5.2 Modulation and Mechanical-based Scanning

Aside from the use of lenses, laser scanning can be accomplished with modulation of the beam path, such as with electro-optic or acousto-optic modulators, or with mechanical displacement and reflection of the beam path, such as with polygon rotary scanners.

Modulation of the beam path with electro-optic modulators (EOM) or acousto-optic modulators (AOM) is an excellent way to deflect a laser if very small step sizes are necessary and at fast speeds. EOMs use birefringence to electrically modulate the beam. Once the beam passes through a birefringent crystal, the linearly polarized light becomes amplitude modulated because the birefringent crystal acts as a polarizer. The degree of birefringence can be modulated by the input voltage across the crystal. So, by varying the voltage, the amplitude of the laser beam can be modulated. This would be an excellent way to modulate the amplitude of the laser beam in the SLS printer for when the laser beam must turn off, because the rise time is exceptionally high. However, using an EOM would add significantly to the cost.

AOMs use ultrasonic sound waves to move a material which is a periodic, spatially varying refractive index. As a laser beam passes through the periodic crystal, such as fused quartz, the different indices of refraction it passes through bends the light at different angles. In this manner, the crystal can be used as a diffraction grating. Ultrasonic waves produced by an ultrasonic transducer can vibrate this crystal at megahertz frequencies. Higher amplitude sounds will push the light into other orders of diffraction. This creates an extremely fast scanning technology that may exceed the specification requirements of the SLS printer.

Rotary scanners are simply mechanically driven rotating mirrors. Mirrors are fused together at their ends to create shapes (polygons), with the number of mirrors determining the step size and resolution characteristics of the scanner. These polygonal mirror setups

must be created with extreme tolerances since each mirror will control one scan line. If the polygon scanner has 18 mirrors, then those are 18 chances to misalign a line scan by a tiny amount. Misalignment from the central axis by only thousands of an inch can cause a scanning system to produce a noticeably wider line than the beam width due to imprecise scanning dimensions.

### 3.3.6 Electromechanical Linear Motion

There are various ways to achieve precise linear motion with mechanical systems. All methods consist of converting rotational motion to linear motion. The two most popular methods of achieving linear motion using a motor are belt drive and lead screw. The belt drive design consists of a gear coupler attached to a motor and a timing belt with equally sized teeth. The motor spins the gear and moves the timing belt. If the base of a part is fixed to one point of the timing belt, we can achieve linear motion. The lead screw design consists of a threaded rod and a flanged nut. The threaded rod is attached to the motor using a coupler and the flanged nut is fixed to the moving part. As the rod spins, the part attached to the flanged nut moves linearly.

Belt driven systems operate at higher speeds compared to the lead screw. The disadvantages of using a belt drive are a slight decrease in accuracy and they are typically more expensive. The benefits of using a lead screw are increased accuracy, increased versatility, and lower cost. [2] Both methods use smooth rods to guide the moving part as it moves in a linear path. For the lead screw design, the guide rods are also used to stop the part from rotating with the flanged nut.

### 3.3.7 Motors

There are many different types of motors each used for different applications. The most common motors used are DC, Stepper and Servo motors. A standard DC motor is a common motor choice due to its simplicity in design and functionality. There are two types of DC motor designs: brushed and brushless. Brushed DC motors work by the brush contacting the rotor and changing the polarity of the center coils. The change in polarity, along with the stationary magnet surrounding the center coils, yields rotational motion. Brushed DC motors are voltage controlled which makes them relatively simple to operate. The motor speed is directly proportional to the voltage so increasing voltage will increase the speed. Due to the brushes making contact to a moving rotor, it creates friction and generates heat.

The construction of a brushless DC motor is opposite of a brushed DC motor. The coils are located on the inner walls of the motor and the rotor contains a permanent magnet. Brushless DC motors are slightly more complicated compared to brushed DC motors as proper timing is needed to control the state and polarity of the coils. Although they are more complex to control, brushless DC motors are a more durable and efficient motors to use compared to its brushed counterpart. Although brushed and brushless motors make great options for continuous motion or speed-controlled applications, position control is not possible.

Stepper motors and servo motors are great options when it is crucial to control exact positioning. Stepper motors work by changing the polarity of coils arranged close together. Stepper motors function very similar to brushless DC motors except the coils turn on and off in discrete steps. The coils and magnet are arranged in a way that each step results in short, incremental motion. The distance for each step is equal in distance therefore each step can be tracked. A stepper motor driver circuit is needed to operate a stepper motor. The driver circuit is input is a square wave and interprets the frequency of the rising edge. The frequency of the input is proportional to the speed of the motor. The driver then outputs the correct amount of voltage and current to each coil in increments. Stepper motors also offer high torque at low speeds Stepper motors are using in many motion-controlled applications such as CNC machines and 3D printers.

A servo motor is comprised of a DC motor and a gear box. The last gear in the servo motor is attached to an encoder. The encoder sends feedback to the microcontroller and the microcontroller adjusts accordingly. Unlike the stepper motor, servo motors do not need a driver circuit. Servo motors come in two packages: positional and continuous. Positional servo motors are limited to 180 degrees of motion. They are typically used in robotics applications. Continuous servo motors are not limited in motion, but they do not offer high torque at low speeds. [3]

## 3.3.8 Relevant Software

### 3.3.8.1 Rust

Rust is a systems level programming language with a strong emphasis on memory safety and reliability. Most languages are designed with a choice in mind between high level features (like garbage collection, and type inference), and low-level control (like pointers and memory management), but Rust takes a different approach to trying to unify the two; this leads to a language that allows for many high-level conveniences while maintaining the speed and efficiency one would expect of a low level systems language. One of the most unique features of the Rust language, and one which is integral in its ability to unify high level features with low level control, is the borrow checker. Rust's borrow checker is, in essence, a set of memory ownership policies enforced by the compiler which guarantee memory safety for both single threaded and concurrent tasks. Like a garbage collector, Rust's borrow checker will free memory which is no longer being referenced or in scope, but it also implements functionality similar to semaphores by ensuring that only one call can be made to a variable at any given time; specifically, functions can only use data in memory that they own, and when a variable is passed to another function its ownership is too which makes data races quite difficult (impossible) to produce.

Despite these high-level features, Rust implements all of these checks at compile time and then lets you run the code without any overhead from maintaining that safety leading to fast, low-level code with high throughput and reliability that would normally be reduced by a garbage collector running periodically in the background. Additionally, zero cost abstractions mean that as a programmer you can write and modify high level features and be confident that your safe and readable code will also be fast as if you had written and

implemented the algorithms yourself, taking the time to optimize them for minimal instruction usage and a low memory profile.

The Rust compiler, or rather the official Rust compilers for various systems, work in two parts, first compiling the Rust code into what is called an intermediate representation and then feeding that intermediate representation into the LLVM compiler to be converted into the appropriate binary. Because of this choice in compilation methods, Rust can support a very large and diverse portfolio of target systems despite being such a new language. Using a single 'phase one' compiler which takes in Rust code and produces a safe and reliable intermediate representation with all the features that make Rust what it is, and then feeding that intermediate representation into the appropriate LLVM backend to be compiled for any system it supports means that Rust gets the compatibility and stability of an old and well tested/used compiler to add to its repertoire of features that are uncharacteristic of such a young language. It also means that whenever LLVM adds a new supported system, or if one exists but has not been implemented for Rust, a new Rust compiler for that target can be made quickly and easily while also maintaining a high degree of certainty that things will work.

Being so versatile, not to mention fast and reliable means that Rust happens to be a great choice for embedded systems too, compilers exist for Rust to most embedded platforms and the language provides the low-level control needed to interact with such hardware, while also reducing the risk of memory issues which are a significant issue that embedded developers must deal with.

Rust is also a relatively new programming language that was first developed at Mozilla for use in their browser and released in 2011. Being so new, Rust includes many modern features that you wouldn't usually find in an older language like C++ or Java, like a standard package/dependency manager and build system. Build scripts have been around for quite some time, and are very helpful to insure that code is compiled properly in large projects that have complicated procedures, for example C++ has Make/CMake and java has Gradle, Ant, Maven, CMake, and others, but these are all 3rd party tools rather than being closely integrated with the language, and require setting up a new build system if you want to join a new project with a different one.

In addition to auxiliary features, Rust also has many modern components to the language itself like a more readable syntax and compile time type inference. Historically, low level languages like Rust could not have had features like that and still run smoothly, but as computing power has increased over time and better algorithms exist for compilers, we can now generate fast and efficient binaries in relatively short order while also allowing for more overhead in the syntax of the language. Like the language itself, Rust also has an advanced macro expansion system for the preprocessor. Macros are particularly important in Rust (for reasons related to the borrow checker described earlier) because they let you execute code on some data without changing scope, they also let you expand a single line of code into an arbitrarily large amount of code before compilation meaning you can save time rewriting your logic and also reduce the number of binary jump instructions used if that sort of optimization is needed.

Table 3.3-1 is a table of useful benchmarks performed on Rust code to help better display its strengths and weaknesses as well as to help compare it to other languages. These benchmarks are all standard(ish) measurements for language speed and can be performed locally to test any Turing complete language, however in order to save time and effort setting up a test environment these scores were obtained from [4].

**Table 3.3-1: Rust Benchmarks**

Benchmark	Time (S)	Memory (B)	Source Code Size (compressed) (B)	Busy (no type given)	CPU Load (core percentages)
reverse-complement	0.45	498,964	3040	0.77	25% 23% 100% 25%
binary-trees	1.09	198,728	765	3.9	87% 98% 88% 86%
k-nucleotide	2.7	159,048	1691	10.08	93% 92% 91% 99%
mandelbrot	0.93	32,676	763	3.7	100% 99% 100% 100%
regex-redux	0.77	147,524	2458	1.99	54% 59% 91% 54%
fasta	0.77	1,840	2529	1.59	66% 100% 4% 38%
fannkuch-redux	7.54	1,200	1020	29.78	97% 100% 99% 99%
pidigits	0.71	3,044	799	0.73	1% 0% 1% 100%
n-body	3.29	1,196	1753	3.33	0% 100% 1% 0%
spectral-norm	0.72	2,672	1055	2.86	100% 100% 100% 99%

### 3.3.8.2 Python

Python is a high-level interpreted scripting language that emphasizes readability and ease of access. Unlike Rust (which is known to have a rather steep learning curve), Python was designed with the goal of making it easy to not only use the language, but also making it more accessible to new programmers which is to say easier to learn. Python is an interpreted language and includes quite the smart interpreter which allows for all sorts of high-level features. From automatic memory management with a garbage collector, to runtime type inference (which is much slower than compile time but allows for variable types to be changed mid-run without redeclarations), and enforcing style guidelines as part

of the syntax, everything about python is designed to help you focus on the logic of your program while the interpreter handles all the nitty gritty implementation of that logic.

Python is also an object-oriented language like Java, and also requires that all code is encapsulated and written as an object like java but differs in that code can be typed in the 'global' scope and will automatically be considered as part of its file as a class/superclass when interpreted. Thanks to all of these ease-of-use features, Python has a large and dedicated user base which help to maintain its great support and expansive catalogue of open-source modules which are available to help with nearly any project. This ease of use, can even be seen as creating a self-perpetuating cycle of new programmers learning Python due to its accessibility, finding it useful and open sourcing their own libraries, and letting those libraries help make the language even more accessible and appealing to new users.

Python is not without its downsides though, as an interpreted language it can be painstakingly slow at times, and also sacrifices a great deal of control over the programs lower-level implementation in order to abstract away difficult computer science concepts and provide useful features in their place. One example of this is the lack of types, all data in Python is object oriented and inherits from the object class which makes type casting neigh on impossible. Of course, this can be substituted for high level alternatives like using class methods to return new objects of the desired type, but this is still an imperfect solution and does not allow for certain optimizations at all. One example being the famous fast inverse square root from Quake III Arena, which makes use of a series of type casts to reinterpret the bits from a floating-point number as an integer before implementing some clever (albeit non-trivial bordering on magical) math to find (an approximation of) the multiplicative inverse of the square root of a number in fewer cycles than it would have taken to do the floating-point division. Naturally, the level of control that is needed for this level of optimization is a blatant violation of Python's focus on readability and letting the interpreter handle the tedious stuff, technically it is considered undefined behavior in C as well, but the compiler will at least let you try, and assume you know what you are doing.

Another well known 'issue' with Python is the Global Interpreter Lock which prevents multithreading. In order to make things as easy as possible for programmers, Python tries (and succeeds) to prevent data races in concurrency by preventing true concurrency. Instead, the interpreter will only run one thread at a time, to ensure no other threads are accessing data, and will switch between threads (when there are more than one) while saving their state to go back later and continue in order to simulate concurrency in a software implementation of hyper-threading. Depending on how well Python's scheduler works with the code in a given program, this can actually lead to noticeable improvements over similar, non-concurrent code; for example, if a thread is sleeping for some period of time, or waiting for a response from some other source (like a web server or another thread), the Python interpreter might save that thread and work on a different one for a while and then come back later to check on the previous one, this would allow for much more work to get done on the single core being used instead of spending time idling while some condition is not yet being met. This may sound like a useful trade off, sacrificing "true concurrency" for safety while keeping some of its benefits anyway, but it also means that a time sensitive task like a keyboard interrupt can potentially trigger a significantly delayed



reaction from the program (or worse, go completely unnoticed if the program is polling or implements a buffer poorly).

A major benefit of python being so well used and well loved, is that it has a staggering level of online support available in many forms. Not only can you find answers to many common programming issues in python, but the list of available libraries is seemingly endless with such libraries available to help with tasks in nearly every conceivable category from graphics to parallelism to data science to biology and chemistry. There are also a myriad of community driven, unofficial, interpreters; some interpreters available for Python make trivial/useless changes like the ability to name variables with emojis, while others change the interpreter model entirely for a just-in-time compiler to help make python orders of magnitude faster; there are also interpreters for nearly every system a programmer may want to run python on including embedded systems, there are even some VHDL Python interpreters for creating a python interpreter at the hardware level. Basically, anything you can think to do with Python, someone else has probably already been insane enough to try.

Table 3.3-2 is a table of useful benchmarks performed on Python code to help better display its strengths and weaknesses as well as to help compare it to other languages. These benchmarks are all standard(ish) measurements for language speed and can be performed locally to test any Turing complete language, however in order to save time and effort setting up a test environment these scores were obtained from [4].

**Table 3.3-2: Python Benchmarks**

<b>Benchmark</b>	<b>Time (S)</b>	<b>Memory (B)</b>	<b>Source Code Size (compressed) (B)</b>	<b>Busy (no type given)</b>	<b>CPU Load (core percentages)</b>
reverse-complement	7.20	1,005,184	814	10.75	20% 53% 48% 29%
binary-trees	48.03	462,732	472	174.44	89% 97% 88% 89%
k-nucleotide	46.28	241,108	1967	176.42	94% 97% 95% 96%
mandelbrot	163.32	12,080	688	642.00	98% 98% 98% 98%
regex-redux	1.36	111,852	1403	2.64	32% 40% 33% 88%
fasta	37.32	846,264	1947	71.03	10% 67% 83% 30%
fannkuch-redux	352.29	12,232	950	1,392.10	97% 99% 100% 99%
pidigits	1.28	12,024	567	1.29	0% 1% 100% 0%
n-body	567.56	8,076	1196	570.95	0% 0% 0% 100%
spectral-norm	120.99	13,424	407	479.86	99% 99% 99% 99%

### 3.3.8.3 C

First released in 1972, C is one of the oldest programming languages still in use and comes with many benefits for having such a legacy. Perhaps the most obvious benefit of such an old language is its speed, unlike hardware which gets faster with time, C was developed to run fast with older and slower hardware that would likely be compared to a handheld calculator today, so when backed with the astonishing computing muscle available on a modern computer, programs written in C tend to run at blazing speeds compared to similar programs written in newer languages. The reason C can achieve this speed is actually a legacy feature, it was originally designed to be a portable assembly which could be written

once and ported to any system with minimal changes while still being written at nearly the assembly instruction level.

C has come a long way since then and has seen many changes with many new and high-level features added, but true to its roots it continues to maintain its close relationship with the assembly instructions it becomes. Despite its close ties with assembly, C was also an attempt at implementing a more human readable syntax, making it what we would call a “high level” language. C has many high-level features that make it easier for programmers to write their code without focusing on hardware level things like how memory is read on a specific system, but perhaps the most important is the concept of custom data types. In assembly all data is stored in specific types based on what the processor is designed to understand and must be manually coerced into such a type if it isn't already, but C allows you to create entirely new data types and define how the processor should interpret and manipulate them. Once a type is created, using structs or enums, it can be used on any machine regardless of whether the processor knows how to process it or not, because the C compiler will create the new type as a collection of types of the processor does understand and define its actions accordingly. This may seem like a trivial feature that can easily be foregone by simply making variables for the types included in a struct, but aside from saving time, it also allows for one of the most fundamental principles of computer science: abstraction. C's structs and enums allow programmers to define custom types along with their behavior and reuse that code later, not only with confidence that it will work (from tests that a good programmer would carry out) and work consistently across the codebase, but also without needing to understand how the code works (in the event that it was written by someone else, you need to know how to use the code and what it does, but not why it does what it does).

Another benefit of C being such an old high-level language, is that it is extremely prevalent to the point where nearly every computer runs C somewhere. This prevalence means that most languages that came after will have some method implemented for communicating with C code in what's called a foreign function interface (except C++ which takes a different approach and just says “C code does not exist here, that code that you call C is actually C++ too so just don't worry about it”). Additionally, because C came before most other languages used today, and is so prevalent, the designers of any given language most likely used C at some point first and were inspired to make a new language to fix something about it. That is to say a great many programming languages in use today are based on C and even try to mimic it to some extent while solving a specific problem (this isn't really a good or bad thing about C per se, but it does mean that C programmers often have a comparatively easy time learning new languages then those that don't know C).

Table 3.3-3 is a table of useful benchmarks performed on C code to help better display its strengths and weaknesses as well as to help compare it to other languages. These benchmarks are all standard(ish) measurements for language speed and can be performed locally to test any Turing complete language, however in order to save time and effort setting up a test environment these scores were obtained from [4].

**Table 3.3-3: C Benchmarks**

<b>Benchmark</b>	<b>Time (S)</b>	<b>Memory (B)</b>	<b>Source Code Size (compressed) (B)</b>	<b>Busy (no type given)</b>	<b>CPU Load (core percentages)</b>
reverse-complement	0.86	712,208	820	1.27	99% 28% 1% 19%
binary-trees	1.54	168,832	809	4.35	60% 67% 57% 100%
k-nucleotide	3.72	130,260	1506	12.07	100% 89% 78% 57%
mandelbrot	1.27	31,792	1135	5.08	100% 100% 99% 100%
regex-redux	0.80	152,172	1397	2.01	52% 99% 48% 53%
fasta	0.78	1,156	1463	0.78	0% 0% 0% 100%
fannkuch-redux	7.58	872	910	29.61	100% 98% 99% 93%
pidigits	0.59	2,444	1090	2.37	100% 100% 100% 98%
n-body	2.18	768	1633	2.19	0% 100% 0% 0%
spectral-norm	0.40	872	1197	1.58	100% 100% 100% 98%

### 3.3.8.4 C++

Similar to C, C++ is a fast and high-level language with instructions close to the assembly they are compiled into leading to fast, efficient code. C++ however aims to improve upon C with the addition of some useful high-level features. C++ is a superset of C meaning all C code can also run as C++ code, but the language gives additional support for such tools as classes, namespaces, and templates (which allow for creating type agnostic code).

C++ also includes a greatly expanded standard library with useful data structures like vectors which allow you to create arrays that can dynamically resize themselves automatically, and hash maps which let you use hashable data types as search keys in an

array like manner instead of creating a function to hash objects to indices or searching through an unsorted list. C++ also allows for operator overloading which lets you assign the behaviors of a new type to specific operators within the language in order to create much more concise and natural feeling code.

Table 3.3-4 is a table of useful benchmarks performed on C++ code to help better display its strengths and weaknesses as well as to help compare it to other languages. These benchmarks are all standard(ish) measurements for language speed and can be performed locally to test any Turing complete language, however in order to save time and effort setting up a test environment these scores were obtained from [4].

**Table 3.3-4: C++ Benchmarks**

Benchmark	Time (S)	Memory (B)	Source Code Size (compressed) (B)	Busy (no type given)	CPU Load (core percentages)
reverse-complement	0.52	1,788	1853	0.52	0% 98% 0% 2%
binary-trees	0.94	176,428	1122	3.39	86% 88% 100% 85%
k-nucleotide	1.93	156,548	1631	5.88	69% 93% 69% 75%
mandelbrot	0.84	34,780	3542	3.27	99% 98% 98% 96%
regex-redux	1.1	203,924	1315	3.43	63% 77% 71% 100%
fasta	0.77	2,504	2751	1.52	64% 0% 99% 37%
fannkuch-redux	3.29	1,892	1528	13.06	100% 100% 100% 97%
pidigits	0.66	5,152	986	2.63	100% 100% 100% 100%
n-body	2.12	764	1927	2.17	0% 2% 100% 0%
spectral-norm	0.72	1,192	1044	2.85	99% 99% 100% 100%

### 3.3.9 Displays

For our project we will also need to have a display. This is important because we will want to provide the user with information as printing status, print time, print settings, cancel option, etc. This means that we need a display of a decently large size able to provide plenty of customizable and dynamic information. This leaves us with two of the most

common display technology options, LCD and OLED. Both of them are perfectly capable of meeting our specifications. They are the dominant technology for consumer screens such as smartphones, TVs, and monitors. However, OLED tends to be considered a premium screen. This is due to its much-improved contrast compared to LCD. OLED screen's pixels are self-emissive, meaning that they emit their own light instead of relying on a backlight like LCD screens do. Because of this when displaying black on the screen, OLED's pixels can just be turned off, therefore providing no light at all in the desired segment. This is not the case for LCDs. When displaying black, the pixels will try to block the light provided by the backlight, yet some light will still get through giving us a lower contrast. However, this premium contrast from OLED screens come at a cost. OLED is a newer technology and therefore it remains a more expensive screen.

Another component about screens relevant to this project is touch-screen capabilities. There are three options regarding touch screen technologies: Non touchscreen, resistive touchscreen, and lastly capacitive touchscreen. Capacitive touchscreen is the preferred technology for handheld devices given their high sensitivity and ability to handle multiple contact points at the same time. However, they are also the most expensive type. Then, we have resistive touchscreens. They are less sensitive, do not handle multiple contact points and have decreased contrast due to more layers needed. They might also need a stylus or a specific material as the input source in order to work. The main benefit coming from resistive screens is that they are significantly cheaper than capacitive screens. Lastly, we have the option to not use a touchscreen at all. However, we still need to have some sort of user input interface. Thus, using a non-touchscreen display would add another hardware component to our design, like buttons or knobs. The extra input components would eliminate the price benefit of using a non-touchscreen display and give our product an outdated feel.

In order to make our user interface more competitive with current market standards we will be using a touch screen. However, given that our main goal is to produce a low cost SLS printer we will pick the more economic technology: resistive LCD touchscreen. A specific screen part will be selected in the component selection section

### 3.3.10 Laser Source

Selective laser sintering products available on the market have a wide range of lasers, as was discussed in the preceding section. These laser sources can be diode-based, gas-based, fiber-based, or even diode pumped solid-state lasers. The selection of laser type is crucial to the quality of the beam, which in turn greatly affects the print quality. The beam should ideally be circular, with good quality, i.e., able to be focused to a small spot. The laser should also be of sufficient power. Laser type also determines the material capabilities of the printer itself. If the powder to be used has very low absorption for a particular wavelength, the powder will not sinter, and the laser-material combination will be useless. To describe proper selection of a laser source, a laser's operating principles will first be described, and then the different laser types will be compared.

Lasers work by way of stimulated emission. When electrons in a material become excited by outside energy, they move to a higher energy level. This is usually unstable, and the electron will seek to move to a lower energy level. While doing so, the electron will lose energy equal to the energy between the two energy levels, given by  $h\nu = E_2 - E_1$ . Depending on the medium, the energy lost can be in the form of vibrations or in the form of photons. This is called spontaneous emission and is the operational principle behind light emitting diodes.

If the medium is pumped with energy in the form of light, stimulated emission can occur. The electrons at a lower energy level are forced into higher energy levels, where they exist for some time before descending to lower energy levels. However, with stimulated emission, the photons that are emitted have the same characteristics as the input photons, i.e., phase, direction, and polarization. This is what makes lasers popular energy choices for laser additive manufacturing.

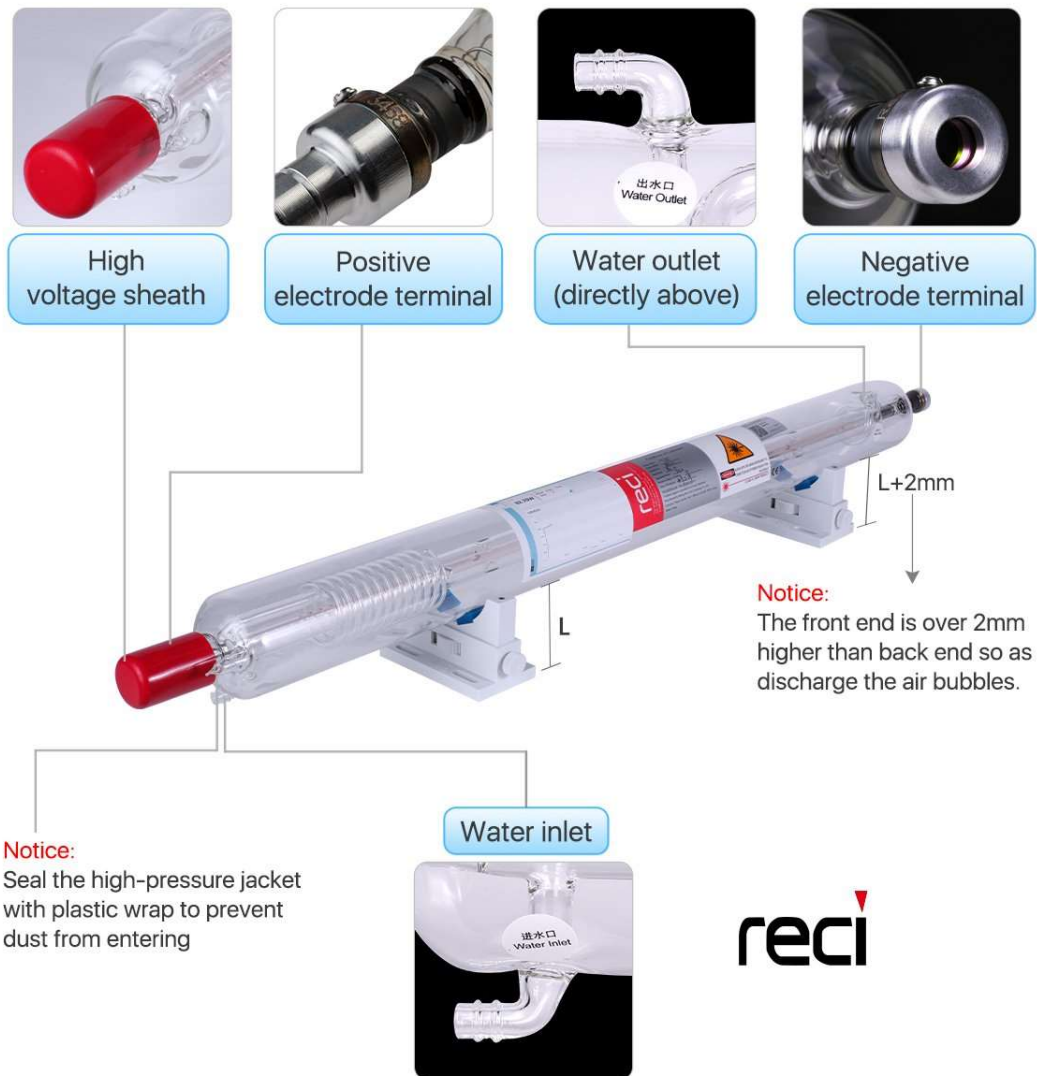
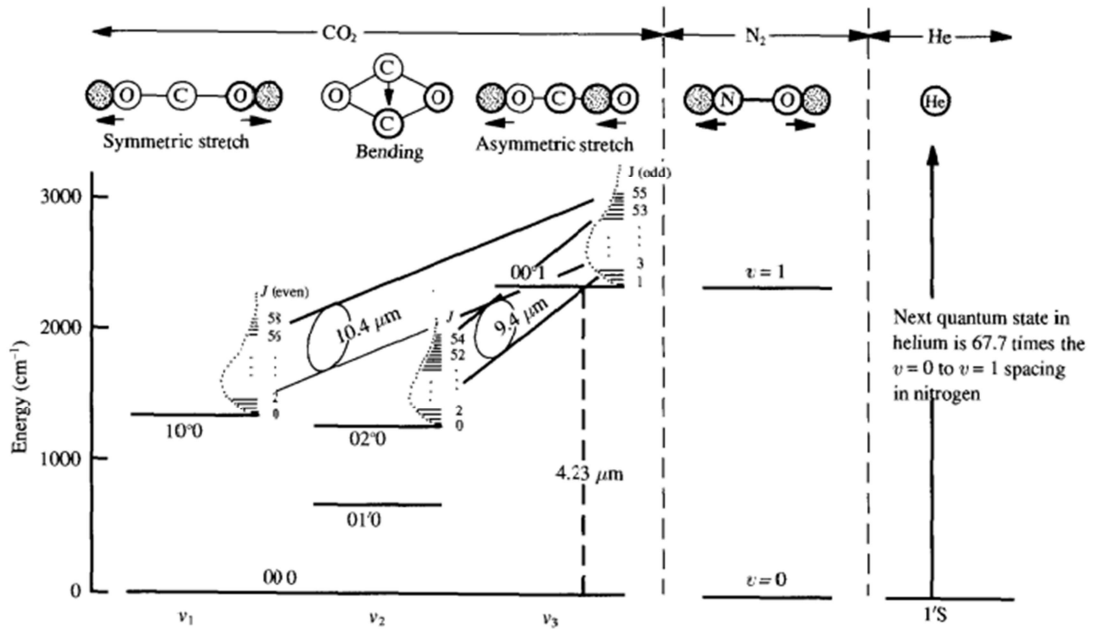


Figure 3.3-6 RECI Carbon Dioxide laser.

A common type of laser used on some SLS systems is the CO<sub>2</sub> laser. As the name suggests, the gaseous gain medium is carbon dioxide. However, it is not composed of entirely CO<sub>2</sub> – CO<sub>2</sub> is the emission material. The other gasses inside the laser are an approximately equal part nitrogen and a large amount of helium. When the gas tube becomes electrically charged, energy is transferred to the electrons within the gas. The energetic electrons can then transfer their energy to the gas inside. This can be done a number of ways. One way is through gas heating; the elastic collisions between electrons and neutral gas atoms raise the ambient temperature of the gas. Another way is by exciting the vibrational states of the gas atoms. Conveniently, an upper level N<sub>2</sub> state matches the CO<sub>2</sub> upper excited state. The N<sub>2</sub> upper state is also metastable, which means the N<sub>2</sub> molecule will emit a photon at this state only after a long time. Because the upper N<sub>2</sub> state matches the CO<sub>2</sub> upper excited state, their collision results in N<sub>2</sub> losing energy and CO<sub>2</sub> becoming excited. CO<sub>2</sub> can also become excited by gaining the energy directly from the energized electrons. Eventually, the excited CO<sub>2</sub> will lose energy in a few possible pathways, with the 10.6 μm transition dominating others, including spontaneous emission of other gasses. In this way, the gaseous medium can begin stimulated emission. Lasing conditions can be met by mirroring the ends of the gas tube, with a partially transmissive mirror as the output coupler. Helium aids the lasing process by providing a route for thermal conductivity, as well as optimizing the kinetic energy of the N<sub>2</sub> molecules (Kuhn, *Laser Engineering*).



**Figure 3.3-7 Energy-level diagram in a CO<sub>2</sub> laser.**

Because of the ease of exciting CO<sub>2</sub> to a higher vibrational state, it has higher efficiency than most other gas lasers, with typical gas lasers operating at less than 1% efficiency, and CO<sub>2</sub> lasers operating anywhere from 20% to 40%. Due to this efficiency, CO<sub>2</sub> lasers are



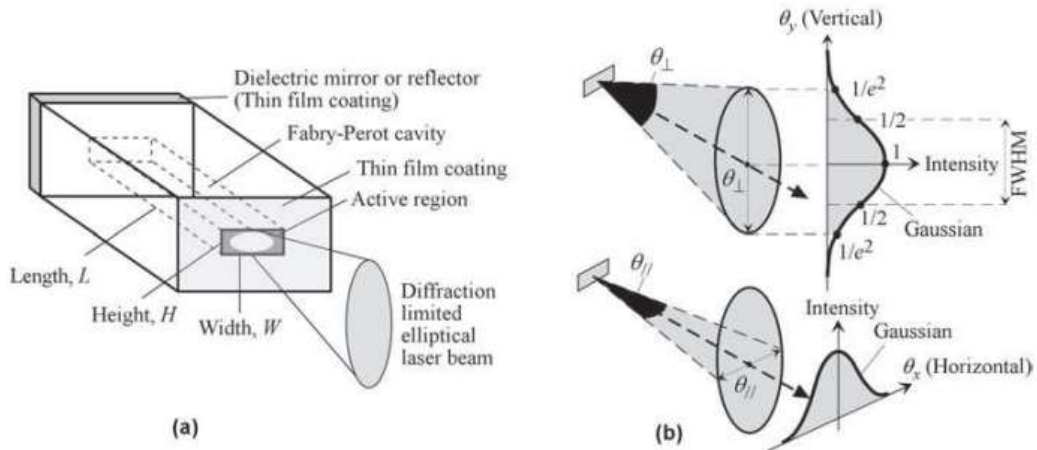
some of the most powerful lasers, and are the laser of choice on high-power laser processing machines, such as laser cutting, patterning, and fusion. For SLS, the output of the CO<sub>2</sub> laser at 10.6 μm is ideal, since most polymers have significantly high absorption in that range.

Gas lasers also have the best beam quality (nearly perfectly Gaussian), so CO<sub>2</sub> lasers would be an ideal choice when optimizing the best beam shape at the powder bed. However, there are drawbacks of CO<sub>2</sub> lasers. One is the size. CO<sub>2</sub> laser tubes for small, sub 100 mW powers, can be nearly a foot long. Approaching higher powers such as 10 or 20 watts means elongating the tube or increasing its gas volume in some manner. Further, CO<sub>2</sub> laser setups are expensive. The laser itself is expensive, and the optics used to control the invisible, high-power beam must be specially coated. This means the CO<sub>2</sub> laser may not be the best choice with some scanning methods, such as x-y tracking, but excels in others, such as galvanometric scanning.

Another type of laser used in SLS systems is the fiber laser. Fiber lasers work by the same principle of other solid-state lasers, in that the core is doped with a material (specifically, a rare-earth element such as neodymium or ytterbium), that contains the transition states for lasing. Pumping of a fiber laser can be achieved by any traditional pumping mechanism. Because fibers are very small and flexible, and because optical gain is a product of the length of the gain medium, fiber lasers can attain considerable gain coefficients; for example, a particular fiber with a 0.8-mW power output from a 20-mW input at a length of two meters can be boosted to an output power of 5 mW if the length increases to ten meters (Thyagarajan, *Lasers*, 2010). As such, their pump threshold power is quite low, and their required minimum lengths are also low. This makes fiber lasers very versatile, as they are lightweight, easy to use, and can sustain extremely high power, similar to CO<sub>2</sub> lasers – Chen et al. (<https://doi.org/10.1364/OL.441789>) achieved 1110 W output power with a Yb-doped fiber laser pumped by 11 W seed light. Fiber lasers also feature excellent beam quality. One difficulty of fiber lasers is their proper alignment, and ensuring they are pumped correctly. Because they are often pumped with laser diodes, it may be more suitable to choose a laser diode directly as the laser source instead of manually coupling the diode with the fiber. Further, the cost of the fiber-coupling alignment assembly is expensive, and the cost of the rare-earth element fiber is also expensive per meter.

A laser source used in less expensive SLS systems is the semiconductor diode laser. Similar to the other two lasers described, these lasers operate by inducing population inversions in a particular medium. The medium for these diodes is perhaps unsurprisingly semiconductors, for example, indium gallium arsenide. These semiconductor devices emit light by electron-hole recombination which occur due to the voltage differences across the junction. Upon injection of carriers into the junction, electrons that were pushed into the higher energy state can fall back down to the lower energy state of the other material, emitting characteristic laser light in a stimulated emission process. As can be seen in Figure 3.3-8, the active region in the semiconductor device is quite large. This large cavity (specifically large for high powered devices) can support many modes, so these devices have decreased beam quality when compared to gas or fiber lasers. Further, because of the unequal proportions of the emitter cavity, the beam which exits the device is highly

astigmatic and has an elliptical beam shape. Laser diodes are also limited in power to the materials they are made of, and because the cavities are quite small. Regardless, laser diodes have many useful features. First, their size is extremely small. From the previously mentioned figure, it can be seen that the size of the device itself takes up less than  $1 \text{ mm}^3$ . This is excellent for implementing onto systems as weight is no longer a concern. Also, they are very efficiency devices, and due to their small size, they are easily cooled and maintained. They are very easy to modulate, since they can be directly modulated by changing the input current signal, and they have access to many different wavelengths of light just by changing the constituent materials.



**Figure 3.3-8 Laser Diode Characteristics**

Choosing which laser type to use on the SLS system can be guided by the specification requirements. The budget is primarily the biggest driver. Although gas lasers have optimal beam characteristics, they are expensive, inefficient, and the optics required to control the gas laser beam are expensive. Fiber lasers also seem attractive due to their ease of use and good beam quality, but they also are expensive and more difficult to align. This leaves the laser diode. It is very inexpensive, and its multimodal characteristics and astigmatism, while unfortunate, can be managed easily. Powers are capped for laser diodes, but something around 5W for a blue diode should leave plenty of room for process parameter optimization.

A blue diode will be the most versatile operating wavelength, specifically 447 nm. This wavelength is highly absorbing in most visible materials except clear and white. This means any thermoplastic material used should not be these colors; that is why CO<sub>2</sub> lasers are used, because the white/clear materials are very absorbent in the 10.6  $\mu\text{m}$  band. Colored thermoplastics can be used. Also, by using a blue laser diode, sugar of specifically darker color can be recrystallized using the emitted energy, allowing the SLS to print food-safe shapes and patterns.

## 3.4 Strategic Components and Part Selection

### 3.4.1 Mechanical Components

#### 3.4.1.1 Threaded Rod and Flanged Nut

Although lead screws are a type of threaded rod, they are strictly designed for linear motion. The differences between lead screws and the threaded rods one can purchase at a hardware store are pitch, lead, and number of starts. The pitch is the distance between screw threads, the lead is the linear displacement of a nut after one revolution, and the number of starts is the number of individual threads wrapped around a screw. The number of starts is very crucial for linear motion as it greatly increases the lead without changing the pitch. The equation for lead is the product of the pitch and the number of starts.

To see how the type of lead screw affects our scanning speed, we calculated the max scanning speed using a 5/16-inch threaded rod from a home improvement store, and an ACME threaded 8mm lead screw. The calculations and max scanning speed for the 5/16-inch threaded rod and 8mm lead screw are shown in Table 3.4-1. From our calculations, we found that the threaded rod is only capable of a max scanning speed of 22.69 mm/s. In our design specification, we must be able to scan at least 30 mm/s. We are not able use the threaded rod in our lead screw design since we would not be able to reach the required speed.

The lead screw provides a maximum scanning speed of 133.3 mm/s which 5.87 times faster than the threaded rod. The lead screw is able to achieve these speeds due to its large lead. The lead is the linear displacement after a nut completes one full rotation. The factors that affect lead are pitch and starts. The pitch is the thickness of the thread and starts are the number of individual threads on a rod. As we are physically limited on increasing the pitch, starts are used as a way to improve the lead. Lead screws come in single, double, or four start. The lead screw used in our analysis is a 4 start lead screw. Since we are able to achieve a high max scanning speed using the lead screw, our stepper motors will not have to work as hard while operating. We plan on using the 8mm ACME threaded lead screw referenced in our calculations. The T8 flanged copper nut with a 2mm pitch, 8mm lead pairs with the chosen lead screw.

**Table 3.4-1: Threaded Rod and Lead Screw Analysis**

<b>5/16 Threaded Rod</b>	<b>Measurement</b>		<b>Lead Screw</b>	<b>Measurement</b>
Diameter (inch)	0.3125		Diameter (mm)	8
Threads/Inch	18			
Threads/mm	0.7347			
Pitch (mm/Thread)	1.3611		Pitch (mm)	2
Start	1		Starts	4
Lead(mm)	1.3611		Lead (mm)	8
Max Rev/min for stepper motor	1000		Max Rev/min for stepper motor	1000
Max Rev/sec for stepper motor	16.667		Max Rev/sec for stepper motor	16.667
Max Scanning Speed (mm/s)	22.685		Max Scanning Speed (mm/s)	133.33

### 3.4.1.2 Smooth Guide Rod and Linear Bearing

Our initial plan for acquiring the hardware was to purchase our hardware at a home improvement store. As this seemed like a feasible option early on, it proved to be invalid. Since most of the parts used for lead screw actuators are measured in metric, it was difficult to find equivalent parts in imperial measurements. The closest conversion we could come up with is 5/16-inch and 8mm. A 5/16-inch rod is 25 mils shorter in diameter than an 8mm rod. A 5/16-inch linear bearing does not exist therefore an 8mm linear bearing must be used. The difference in measurement doesn't seem like much but when using rods and linear bearings it does. Linear bearings need to be properly loaded in order to function properly or else they could jam or break. Another issue with using an undersized rod with linear bearings is the amount of play between them. A large amount of play could ultimately lead into inaccurate motion control. Due to these issues, we chose to use an 8mm smooth rod paired with LM8UU, 8mm, linear bearings.

### 3.4.1.3 Self -Manufactured Components

Since SLS printers are not as popular as some other additive manufacturing methods, SLS printer mechanical parts and assemblies are not as accessible on the market. Therefore, we will be designing and building the mechanical system. The enclosure will be manufacture out of wood. Wood is a widely accessible material and acts as a good insulator compared to other materials. Using a ¼ inch piece of wood, we can use a laser cutter to cut out the frame of the enclosure. A laser cutter is accessible to the team free of charge, so constructing the enclosure using this method is cost effective.

The brackets and mounts needed to join the linear actuator subsystems for the SLS printer will be 3D printed. Specially designed brackets are needed for our SLS printer design since there are no standard brackets we can utilize. The brackets and mounts are designed using SolidWorks and are 3D printed in Polylactic Acid, also known as PLA. PLA is an excellent filament to use for prototyping as it is easy to print, non-toxic, and inexpensive. Thermoplastic 3D printers are available to the team free of charge. 3D printing these components will greatly reduce the cost of the SLS printer construction.

### 3.4.1.4 Stepper Motor

Picking the appropriate motor to use for 3D printing is a crucial design decision. 3D printing requires precise motor control in both speed and position. We plan on using a stepper motor to drive the linear actuator system. Stepper motors offer high torque and precise motion. Referring to the requirements specification, the motor holding torque needed is 40-45 N · cm. Stepper motors are standardized to specific sizes. The sizes we will be considering are NEMA 14, NEMA 17, NEMA 23. Depending on the manufacturer, holding torque could vary a small range within the same size stepper motor. Table 3.4-2 lists the stepper motors we considered.

**Table 3.4-2: NEMA Stepper Motor Comparison**

Stepper Motor	NEMA 14	NEMA 17	NEMA 23
Part Number	14HS13-0804S	17HS4401S	23HS22-2804S
Connection	Bipolar	Bipolar	Bipolar
Step Angle	1.8	1.8	1.8
Weight (Kg)	0.17	0.28	0.7
Holding Torque (N · cm)	18	42	126

From the table, we find that the NEMA 14 stepper motor does not have a large enough holding torque to satisfy the requirement. NEMA 17 and NEMA 23 both have the necessary holding torque except they are vastly different in weight and size. Since NEMA 23 would be too large and powerful for our application, the motor we plan on utilizing is a NEMA 17 stepper motor. NEMA 17 stepper motors are commonly used in CNC machines, laser engravers, and 3D printers. The stepper motor will drive the lead screw to achieve linear motion, so a coupler is needed to join the shaft of the motor to the lead screw. The shaft of the stepper motor is 5mm. Since our lead screw is 8mm, we will need a 5mm to 8mm coupler.

## 3.4.2 Electrical Components

### 3.4.2.1 Stepper Motor Driver

Since we plan on using a stepper motor for our linear actuator system, we will need a stepper motor driver to control the stepper motor. When looking for a stepper motor driver integrated circuit (IC), we investigated ICs that have plenty of documentation and are implemented on modules. It was crucial for us that they were implemented on modules because we wanted to be able to test before our final PCB design. The selection came down to four drivers: DRV8825, A4988, DRV8436, DRV8811. The stepper motor driver characteristics are listed in Table 3.4-3.

**Table 3.4-3: Stepper Motor Driver IC Comparison**

Part #	DRV8825	A4988	DRV8436	DRV8424
Manufacturer	Texas Instrument	Allegro	Texas Instrument	Texas Instrument
Operating Voltage	8.2 V – 45 V	8 V – 35 V	4.5 V – 50 V	4.5 V – 35 V
Full-Scale Current	2.5 A	2 A	1.5 A	2.5 A
Micro step Resolution	Up to 1/32	Up to 1/16	Up to 1/256	Up to 1/256
Inventory (1: Low – 4: High)	4	1	3	2
Available as a Module	Yes	Yes	No	No
Cost	\$4.98	\$3.36	\$3.96	\$4.00

From the table, we find that the operating voltage for all the chosen ICs are within the same range with A4988 being the lowest at 35 V. This is not an issue as we plan on operating voltage being 12 volts at most. Having a high operating voltage max does help in the case of voltage spikes. The best micro step resolution is 1/256 of a step. A higher micro step leads to a smoother rotation, so we expect the motors to run the smoothest with the DRV8436 or the DRV8424. Inventory is the biggest determinant on which chip we will be using in our design. The DRV8825 will have the largest inventory in November and December 2021. The A4988 is currently not available as an IC but they are available as a module. DRV8436 and DRV8424 have limited inventory as their next order dates are late 2022. The DRV8825 is also the most expensive chip at \$4.98. Although it is the most expensive, they are also available as modules.

Our first choice is the DRV8825 since it is available as an IC and as a module. The DRV8825 stepper motor modules will be used to test the stepper motors and linear actuator system as mentioned later in 7.1.1 Stepper Motor Testing and 7.1.2 Linear Actuator Testing.

Our second choice is the DRV8436. The DRV8436 had higher micro stepping and is more affordable compared to the DRV8825. If it is in inventory by the time of our final PCB fabrication, we will design our motor driver circuit around the DRV8436. In the event that the DRV8436 is out of stock, we will design our final PCB to include the DRV8825.

### 3.4.2.2 Laser Driver Operational Amplifier

The laser driver design requires an operational amplifier (op-amp). The op-amp's role in the design is to regulate the voltage at the sense resistor. For ease of design, the op-amp should be able to operate with a single supply voltage. The single supply voltage configuration is where the +Vcc pin is connected to the supply voltage and the -Vcc pin is connected to ground. The current of the driver is affected by the voltage across the sense resistor. The voltage and resistance used in the design is relatively low so a small voltage change could greatly affect the current output. Therefore, the input offset voltage should be as low as possible. The final op-amp selection consists of three op-amps that operate with a single supply and have a low input offset voltage. The data for each op-amp is shown in Table 3.4-4.

**Table 3.4-4: Op-Amp Comparison**

Part #	TL072AC	LM358A	LM324
Manufacturer	Texas Instrument	Texas Instrument	Texas Instrument
Supply Voltage	4.5 V – 40 V	3 V – 32 V	3 V – 32 V
Typical Input Offset Voltage	3 mV	2 mV	3 mV
Maximum Input Offset Voltage	6 mV	3 mV	7 mV
Inventory (1: Low – 3: High)	3	1	2
Cost	\$1.30	\$0.54	\$0.45

The final three op-amp candidates are the TL072AC, LM358A, and LM324. The TL072AC offers an input offset voltage of 3 mV and a max input offset voltage of 6 mV. The TL072AC has the most available inventory but it is the most expensive op-amp coming in at \$1.30. The minimum supply voltage is also the highest among the final selection. The LM358A offers the lowest typical and max input offset voltage. The biggest concern with this op-amp is the inventory. The LM324 offers the same typical input offset voltage as the

TL072AC but has the highest maximum input offset voltage. It is also the most affordable option, and they are also high in inventory. The chosen op-amp is the LM358A due to its low cost and low input offset voltage. The second option is the LM324, and it will be used if the LM358A is out of stock.

### 3.4.2.3 Power MOSFET

For the laser driver design, the MOSFET plays a crucial role in delivering the necessary current to the laser diode. There are many types of MOSFETs but the one needed for this application is a power MOSFET. Power MOSFETs are used to control high power electronics with logic level voltage. The laser diode has an operating current of 3 A so we must use a MOSFET that can support the current demand. When using power MOSFETs, it is important to note that heat dissipation is an important factor to consider. Therefore, the final selection of MOSFETs will have a large maximum power dissipation. Another factor that affects the design is the drain-source resistance. When the gate-source threshold voltage is met, the MOSFET turns on. When the MOSFET is on, there is a resistance between the drain and the source. Since the current of the laser driver is determined by the input voltage and a sense resistor with a low resistance, the drain-source resistance will affect the current. After considering these factors, we selected three possible candidates: IRFZ44NPBF, IRL7833PBF, and RQ3E130BNTB. The data for each power MOSFET is shown in Table 3.4-5.

**Table 3.4-5: Power MOSFET Comparison**

Part #	IRFZ44NPBF	IRL7833PBF	RQ3E130BNTB
Manufacturer	Infineon Technologies	Infineon Technologies	ROHM Semiconductor
Drain-Source Voltage	55 V	30 V	30 V
Drain Current	49 A	150 A	39 A
Gate Threshold Voltage	2-4 V	1.4-2.3 V	1-2.5 V
Maximum Power Dissipation	94 W	140 W	16 W
$R_{DS(on)}$	17.5 m $\Omega$	3.8 m $\Omega$	6 m $\Omega$
Cost	\$1.43	\$ 1.84	\$0.56

From the table, we find that all of the power MOSFETs meet the stated requirements. IRFZ44NPBF offers a high drain to source voltage and a good maximum power



dissipation, however, the drain-source resistance is relatively high. Although the resistance is low compared to other MOSFETs, it might be too high for our application. The IRL7833PBF offers a high drain current, high maximum power dissipation, and a low drain-source resistance. It is the most expensive MOSFET, but cost is not as great of a concern since only one will be used. The RQ3E130BNTB is the most inexpensive MOSFET on the list but offers the lowest maximum power dissipation.

After careful consideration, we ended up choosing the IRL7833PBF. Along with having the best performance in all of the considered factors, it has the lowest gate threshold voltage. The drain source voltage and the drain current are sufficient for the laser driver design. The maximum power dissipation of 140 W ensures that the temperature of the MOSFET will not be an issue. A heat sink will still be used to promote optimal temperature control.

### 3.4.2.4 AC/DC Converter

Given that our printer will be stationary and with a substantial volume, the issues of size and weight of the transformer system to convert AC into DC will not significantly affect us. Additionally, we need to keep in mind that our main objective is to make SLS printing more affordable. Thus, it would benefit us to use the transformer system to reduce the cost of the rectifier circuit and the capacitor. With the aforementioned advantages, we have decided to go ahead and select the transformer system for our design. This means that we need to select 3 components for this AC/DC converter.

#### 3.4.2.4.1 Transformer

When choosing a transformer, we need to make sure it is capable of transmitting enough power. We have pre-selected 3 different transformers listed in Table 3.4-6. and a final selection will be done in this section. From initial estimates even the F-96U, a 60W transformer would be able to supply enough power to all our modules. However, we could end up being limited by this transformer and this might prevent us from adding additional features if time allows. Thus, even when our main goal is to make an affordable SLS printer, going with the cheaper option could be a risk in terms of sufficient power. The next option would be the 166S11. This transformer provides us with close to double the power. But since the extra desired power is just a precautionary measure, it would not be worth spending more than double for a lot of power that might go unused. This leaves us with the 185E16. This transformer falls in the middle in both power and price. Therefore, it would be a good balanced choice. However, we will wait until other modules power necessities are further defined to make a final selection\*.

\*After checking our local hardware store we were able to obtain the 185E16 transformer at \$17.50 making it our best option.

**Table 3.4-6: Step Down Transformer Comparison**

	<b>F-96U</b>	<b>166S11</b>	<b>185E16</b>
Manufacturer	Triad Magnetics	Hammond Manufacturing	
Voltage Primary	115V	115V	115V - 230V
Voltage Secondary	10V	11V	Parallel 8V - Series 16V
Power	60W	110W	80W
Price	\$22.17	\$57.48	\$34.65
Dimensions	6.35cm L 6.985cm W 7.62cm H	11.43cm L 6.35cm W 7.62 H	6.35cm L 6.629cm W 7.62cm H

### 3.4.2.4.2 Full Bridge Rectifier

Given that we selected the 185E16 transformer, we will have the option to step down to either 16V or 8V, that means that we will have roughly 5A or 10A of current going through this rectifier. Thus, we need to make sure the component we pick is rated for such current. Three different options are discussed in this section and their specifications listed in Table 3.4-7. Starting with the option by Vishay, the VS-KBPC810PBF (810 for short) has a maximum forward current of 8A. This would limit us to pick the 16V configuration of the transformer which might be undesirable in case that it makes us design an extra DC/DC regulator. As for the bridge from Onsemi, the 12A gives us the choice to operate the transformer at either 16V or 8V configuration. A cheaper option is the BR1005 however, its max forward current is exactly 10A. Operating the transformer at 8V 10A over long periods of time might damage the bridge. Since our product might have long print times then the BR1005 would not be desirable for our design. Yet, having found the BR1005 at our local hardware store and considering its low price, it will be a good part for testing. As for the final design we will select the Onsemi 512-GBPC1210W.

**Table 3.4-7: Rectifier IC Comparison**

	<b>VS- KBPC810PBF</b>	<b>512- GBPC1210W</b>	<b>583- BR1005</b>
Manufacturer	Vishay	Onsemi	Rectron
Forward Voltage	1V	1.1V	1.1V
Forward Current	8A	12A	10A
Price	\$4.10	\$4.99	\$1.20
Dimensions	1.575cm L	2.9cm L	1.96cm L
	1.575cm W	2.9cm W	1.96cm W
	0.533cm H	1.123cm H	0.75cm H
Mounting Type	Through Hole	Through Hole	Through Hole

### 3.4.2.4.3 Capacitor

After running simulations to test different capacitance values, we selected 4700uF as a good capacitance to have a low ripple. At this value, electrolytic capacitors are the only type of capacitors produced. Considering that from the 185E16 transformer we might have a voltage of 16V and that this is actually RMS voltage, we need to have the capacitor rated for at least 23V. From our local hardware store, we were able to find a 4700uF capacitor rated for 25V at \$1. This capacitor could be good enough for our design if we end up using the transformer in 8V configuration. However, if we use the 16V configuration, just like mentioned before we would need a voltage rating of 23V which is too close to the 25V rating of the capacitor. In which case we would use the capacitor ALC80A562CC063 that has a voltage rating of 63V and a price of \$8.59.

### 3.4.2.4.4 Power Cord

Another key component needed is the cord to connect the wall outlet to our transformer. As per the standards discussed later in the document, we have decided to get a NAME 5 - 15 cord. This way we ensure we have wide compatibility with wall outlets. There are a few more NAME cords that are widely compatible. However, we pick NAME 5 since it has a ground pin. The presence of the ground pin is a user safety feature. For devices with metal cases or lots of exposed metal parts, the ground pin is connected to this metal. That way metal pieces that are not supposed to interact with electricity do not get charged and become a shock hazard for the user. In our case, some possible exposed metal that might need to be grounded would be the external metal of the power transformer, metal structure

support and plates, and any metal in the enclosure in the case that it is not fully designed on wood.

### 3.4.2.5 DC/DC Regulator

Given that we will be limited by the power delivered by the transformer in the AC/DC converter we need to make sure that we use components with as high efficiencies as possible. This makes switching regulators much more practical for our design. In this section we will propose a few switching regulators that we could use. However, a decision will not be made until other modules fully define their power needs. Also pending on microcontroller power calculations there exists the possibility of using a linear regulator for the processing module. This is because the power that goes into processing hardware is very low, and thus, low efficiencies would not result in big power losses when compared with the total power use by SLS printer. Both regulators presented in Table 3.4-8 are good options. However, LM2576HV is used in the electronics II lab here in UCF therefore we are more familiar with it. It also gives us the option to test before choosing one for our design. Both of these regulators need external components. The values and selection of these components will be calculated and discussed further in the design section of this report. The LM2576 comes in multiple levels, including an adjustable option. Thus, we can use it for our 9V regulators and the 5V regulators. However, we do need to find another regulator for the 6V since the laser driver needs a soft-start to protect the diode.

Ultimately, we made our design around the LM2576. This is because chip supply is scarce and most other regulators were out of stock. For the 6V laser driver, we were limited to the LM22679 due to availability. However, this chip still meets all of our requirements. Which are capable of 6V output with 3A of supply.

**Table 3.4-8: Voltage Regulator ICs**

	<b>TPS56339</b>	<b>LM2576HV</b>
Manufacturer	Texas Instruments	Texas Instruments
Input Voltage	4.5V - 24V	7V - 40V
Output Voltage	0.8V - 16V	3.3V - 15V
Max Current	3A	3A
Price	\$1.38	\$3.69

**Table 3.4-9 Soft-Start Voltage Regulator IC**

	<b>LM22679</b>
Manufacturer	Texas Instruments
Input Voltage	4.5V - 42V
Output Voltage	1.285V - 40V
Max Current	5A
Price	\$4.274

### 3.4.2.6 Display

From section 3.3.9 within 3.3 Relevant Technologies, we defined that we would be using a LCD touchscreen of resistive type. In order to simplify design, we decided to look for screen modules that are ready to be connected to a processor and be used. After some research we selected a 3.5 inch screen by Treedix, this screen module available on Amazon has a resolution of 320x480 and can display color. It also has a microSD card module in which images can be stored. The cost of this display module is \$15.99

The module has 20 pins, 4 for SD card SPI communication, 3 power pins, and the rest are LCD pins. All of those pins except for the power pins will need to be connected to GPIO pins of the power module.

### 3.4.2.7 Microcontroller

#### 3.4.2.7.1 Hardware Class

To start, the computing needs to be done somewhere, a microcontroller, an FPGA, or a full desktop/mobile processor with an operating system. We already decided that we'll be using programming languages like C++ and Rust, which basically requires that we don't use an FPGA, but let's take a look at that too, we may decide that it is worth changing our mind on the software stack or using it for some auxiliary computing needs.

FPGA's encode instructions as physical logic gate connections in an array of gates in order to execute those instructions on what is essentially custom hardware designed for the task. When using the same algorithm and with other needed resources being comparable, there is simply nothing faster than custom hardware (an ASIC or Application Specific Integrated Circuit would be faster, but it is the same technology just not reprogrammable, and also

would cost millions of dollars for our production size). The main downside of using an FPGA is that you have to define the connections of each and every logic gate which is an arduous task to say the least and for our application, is most definitely not worth it unless we need to perform complex optical calculations in real-time; as such we would not be using FPGA's unless the complicated calculations mentioned earlier are needed.

Microcontrollers are a great option as well, they provide the standard computational functionality you'd expect from an ordinary CPU but generally with a stripped-down instruction set, along with lower clock speeds and lower power consumption than what you'd see in a desktop or mobile processor. Microprocessors also generally provide more easily accessible physical IO pins that are of great importance for any computer system interacting with the outside world, this can of course be substituted with an expansion card for the specific application on a normal computer or in the case of hobbyist all in single board computers like the raspberry or jetson nano such hardware is already part of the board. Still, having an interface to the outside world as a standard feature is a boon for microcontrollers which more powerful CPUs tend to lack.

The final option is of course the classic and you're fully featured desktop or mobile CPU. These CPUs tend to have a larger instruction set and a higher clock frequency which lets them do a lot more work much faster, multi-core CPUs are also quite common and resources like ram storage tend to be much more abundant. The downside of these more powerful CPUs is that they require more power as well as more cooling and tend to cost much more too. Given the microprocessors standard ability to interface with the real world using its physical IO pins, and its significantly lower price (we can easily use five or ten microcontrollers to control all the different subsystems for less money than a Raspberry Pi), using a microprocessor seems to be the way to go.

### 3.4.2.7.2 Instruction Set Architecture

So now that we have our processor 'type' figured out, we need to determine the instruction set we'll be using. Strictly speaking, this doesn't affect our project too much as we will be using compiled languages and a real time operating system to abstract away any hardware specific, tasks like simulating floating point numbers on an architecture that doesn't support them, but it is still an important decision that needs to be made and will help narrow down our choices for part selection later. Two of the most popular Instruction sets for embedded systems are AVR, and ARM. While there are pros and cons to both, ARM tends to be more computationally powerful than AVR while also maintaining a comparable price on most chips, making it a much better option for its cost to performance ratio.

### 3.4.2.7.3 Microcontroller

There are many microcontrollers/microprocessors to choose from, but luckily there are two more deciding factors to take into account: price and availability. We could go with a Texas Instruments board which tend to use cheaper microprocessors (because they make them themselves, not because they are worse parts), but even though they usually have good availability the development boards are often significantly more expensive than

something like an Arduino, and the standard libraries available with their compiler tend to be less user friendly as well. We could also go with an Arduino (not necessarily the uno, they have more powerful “professional” boards), but their higher end boards are currently out of stock at the most common distributors and the chips aren’t much easier to come by. It is important to note, that while microprocessor chips will eventually be used in our project and be soldered to our PCB, we still need development boards for testing and prototyping. One option that seems pretty good given the current chip crisis, is the new Raspberry Pi Pico; it is powered by the RP2040 chip which is a powerful dual core ARM cortex m0+ processor, both the board and chip are readily available with no factory lead time, and they’re fairly cheap too (\$4 for the board and \$1 for the chip). Additionally, I happen to have 6 Pi Pico boards which we can use for testing and also desolder the chips if need be (they aren’t that expensive so I don’t mind) making it a no brainer to use parts we already have right now and don’t need to spend more money on instead of waiting around for new parts.

### 3.4.2.7.4 Final Microcontroller Design Decision

Despite the massive advantages in processing power and availability that arm boards like the rp2040 pi pico hold over similar AVR based boards, we found that AVR did have 2 distinct advantages that ultimately made it the better choice. First is compatibility, the AVR instruction set has been a mainstay in embedded systems for a long time, and therefore has much better compatibility with different open source software options that would be quite helpful. Additionally, the IO on the pi pico is simply too limited, and didn’t provide enough pins to send all of the signals needed for our project. While both problems could have been solved with careful design, like using bit shift registers to drive more pins, or modifying the available software to work as needed with ARM, we found that the simplest solution was to simply use a chip with better compatibility across hardware and software. We ultimately chose the AVR based Atmega 2560 which was supported by the Repetier open source 3D printer firmware, and had dozens of extra pins to be used for our extra functionality.

## 3.4.3 Optical Components

This section details the choice criteria for parts within the optical subsystem. Vendor products will be analyzed to find the best options for laser source, lens setups, and other miscellaneous parts that may be needed.

### 3.4.3.1 Laser Selection

The laser sources available on the market will be analyzed to see which ones best fit the project’s specification requirements. The laser should be relatively low cost, within the budget, and have ideal wavelength characteristics. The wavelength should be either far IR (10.6 microns), near-IR (1060 nm), or visible (400 nm – 600 nm). These are the most desirable laser wavelengths for thermoplastics.

**Table 3.4-10 Types of Lasers Used in SLS Printers**

Laser Type	Active Medium	Power	Wavelength	Manufacturer	Model Number	Cost
Gas	CO <sub>2</sub>	80W	10640 nm	Omtech Laser	EFR 80W Laser Tube	\$399.99
Gas	CO <sub>2</sub>	100W	10640 nm	Omtech Laser	EFR 100W Laser Tube	\$699.99
Fiber	Ytterbium	20W	1060 nm	mks Spectra-Physics	VGEN-SP-NL-25-20	\$2400
Fiber	Ytterbium	30W	976-1075 nm	IPG Photonics	YLM-30	\$1199.80
DPSS	Nd:YAG	10W	1064 nm	Endurance Lasers	Invincible	\$495
DPSS	Nd:YAG	50W	808 nm	Endurance Lasers	FAP800 Coherent	\$1495

A quick purview of the available high-power laser types available makes it clear that most of these laser types are far too expensive for this project, as shown in Table 3.4-10. Gas lasers such as CO<sub>2</sub> cost hundreds of dollars, and that’s only for the tube – no power or optics. Fiber lasers are much easier to use and have less hassle when setting up, yet high power fiber lasers are thousands of dollars. Other companies like Endurance Lasers provide diode-pumped solid-state lasers. For the power, they are not that expensive, but they still fall out of the scope of this project’s budget.

However, diode lasers have yet to be explored. These are small and inexpensive. Table 3.4-11 shows a comparison of available high-power diodes on the market that could be used for SLS printing. The target wavelength here was 445 nm due to its high absorption in various materials.

High power diodes are certainly the way to go for a budget SLS printer. While some diodes are still very expensive, the diodes offered by Osram are an excellent price. The highlighted diode is the final chosen laser source for the SLS printer. This diode features very high output power with a 9-degree slow axis divergence and 49-degree fast axis divergence. Thorlabs diodes have much less divergence compared to Osram, but they are also significantly more costly per output power



**Table 3.4-11 Comparison of Laser Diodes**

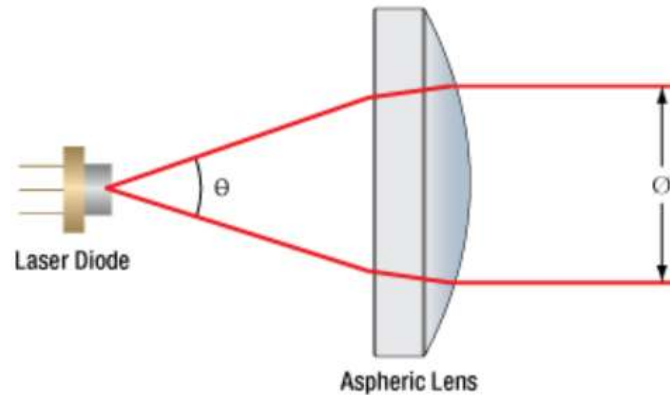
Laser Type	Power	Wavelength	Manufacturer	Model Number	Cost
Diode	6W	445 nm	Nichia	NUBM44	\$216.70
Diode	5W	447 nm	Osram	PLPT9 450LB E	\$61.03
Diode	4W	445 nm	OptLasers	RLS/B445-4000CM	\$1335.00
Diode	3.5W	448 nm	Lasertack	BLD-448-3500	\$493.50
Diode	3.5W	447 nm	Osram	PLPT9 450D E	\$74.25
Diode	3W	447 nm	ThorLabs	L450G1	\$176.39
Diode	1.6W	445 nm	Osram	PLPT5 447KA	\$33.97
Diode	1.6W	450 nm	ThorLabs	L450P1600MM	\$86.30
Diode	1.4W	462 nm	Lasertack	RLS/NDB7675	\$343.50
Diode	1W	405 nm	ThorLabs	L405G1	\$710.80

### 3.4.3.2 Lens Selection

Because the SLS system will not use a galvanometric scanning assembly, the lenses involved will be used for controlling the laser beam out of the diode and onto the powder bed. As mentioned previously, diode lasers have severe astigmatism which results in a highly elliptical beam. The lenses used in the optical assembly will achieve three functions: collimation, beam shaping, and focusing.

Collimation of a laser beam is rather straightforward, but different lens types can create a beam with different characteristics. The most simple collimating lens is the spherical lens type. These can be further classified into the concavity of their faces. Plano-convex lenses have one face flat while the other is convex. These lenses have a positive focal length and works best in situations where one conjugate distance is much larger than the other. For example, collimated light has an (ideally) infinite conjugate distance, so a plano-convex lens would be great for focusing collimated light, or collimating light from a point source. A bi-convex lens has two convex faces. This is similar to the plano-convex lens, except the bi-convex lens really works best in situations where the conjugate distance is only different in a multiple range of 0.2 to 5. These lenses also work best when the object and image distance is the same. Plano-concave and bi-concave lenses exhibit negative focal lengths and can collimate converging light or diverge collimated light.

These spherical lenses are inexpensive and easy to make, but they come with their own problems. Spherical lenses or singlets induce spherical aberrations within the optical system. The change in distance at the outer edge of the lens versus along the optical axis can be quite significant in an optical system. For example, a single plano-convex lens which focuses collimated light with a 10 cm focal length can have spherical aberrations as large as 2.2 mm, which is the distance between the marginal focus and paraxial focus. The marginal focus is the focal point from the rays at the outer edge of the spherical lens while the paraxial focus is the focal point from the rays at the center of the spherical lens.



**Figure 3.4-1 Laser diode collimation by an aspheric lens.**

There are two main routes to combating spherical aberration. The first is to use a multi-lens system. This can work very well for only a minor addition of complexity. For example, the same system as above can have its spherical aberration reduced to 0.8 mm by simply adding another plano-convex lens. However, it is important to note that this also reduces the focal length. Using three lenses can decrease the spherical aberration even more, again at the cost of reduced focal length. Another method to reduce spherical aberration is to use a different type of lens that corrects for the aberration specifically. These are called aspheric lenses and are very common for diode collimation applications. Aspheric lenses get their name from the shape of their faces; they are not truly spherical, but still curved. The curvature of the lens faces eliminates spherical aberrations by correcting the optical path length of wide-angle light through the lens. When using an aspheric lens for laser diode collimation, there is no longer a need to use multiple lenses just for collimating. However, aspheric lenses induce chromatic aberration, and they are more costly to produce. Thankfully, the laser light within the SLS system will be monochromatic so chromatic aberrations are not a problem. The cost, however, certainly will be a deciding factor.

Focusing the laser light after it has been collimated can be accomplished with a simple spherical lens as was described before. To reduce spherical aberrations with a plano-convex lens for focusing collimated light, the curved face of the lens should be the face upon which the infinite conjugate light is incident.

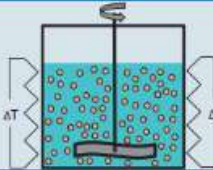
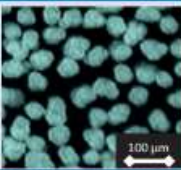

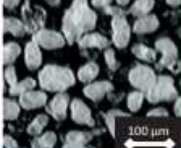


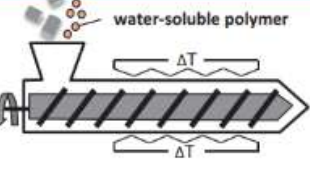
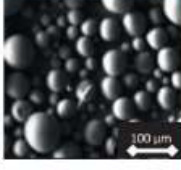
The lens selection now focuses on three types: spherical, plano-convex, aspheric, and cylindrical. These are compared and contrasted with different suppliers in the table below.

**Table 3.4-12 Comparison of Lens Types**

Lens Type	Focal Length (mm)	Diameter (mm)		Manufacturer	Model Number	Cost
Plano-Convex	75 mm	25.4		ThorLabs	LA1608-A	\$34.36
Plano-Convex	75 mm	25		ThorLabs	LA1257-A	\$34.08

### 3.4.3.3 Material

Most SLS printers available on the market are finely tuned for a specific type of material. This material is usually polyamide, such as Nylon 6, Nylon 11, or Nylon 12, which is a type of thermoplastic. In thermoplastics, bonds between polymer chains can be broken by sufficient outside energy. In the case of SLS, this outside energy is supplied by the focused laser beam. Once the bonds are broken, the thermoplastic undergoes reflow, and reshaping of the material occurs. Under a certain energy, bonds are reformed, and the plastic hardens. In this manner, thermoplastics can be used for SLS.

Process	Process diagram	Advantages/Disadvantages	Typical particle
Suspension polymerization		<b>Positive:</b> <ul style="list-style-type: none"> <li>- very good spherical particles</li> <li>- narrow unimodal distribution</li> </ul> <b>Negative:</b> <ul style="list-style-type: none"> <li>- time-consuming procedures require specific process know-how</li> </ul>	
Precipitate		<b>Positive:</b> <ul style="list-style-type: none"> <li>- potato-shaped particles with good free-flowing properties</li> </ul> <b>Negative:</b> <ul style="list-style-type: none"> <li>- time-consuming procedures require specific process know-how</li> </ul>	
Cryogenic grinding		<b>Positive:</b> <ul style="list-style-type: none"> <li>- simple process – widely used</li> <li>- available for almost all polymers</li> </ul> <b>Negative:</b> <ul style="list-style-type: none"> <li>- severely damaged particle surfaces and geometries (bad free-flowing properties)</li> </ul>	
Coextrusion		<b>Positive:</b> <ul style="list-style-type: none"> <li>- almost perfectly spherical particles</li> <li>- continuous process</li> </ul> <b>Negative:</b> <ul style="list-style-type: none"> <li>- complex procedures and time-consuming subsequent processing steps</li> <li>- recycling of process aids</li> </ul>	

**Figure 3.4-2 Comparison of Powder Creation Technologies**

Thermoplastics can also be added with different materials to enhance their final structure. Some of these materials include ceramics, carbon fiber, and other metals such as aluminum. While the sintering does not directly sinter the individual bits of metal or carbon fiber, the reflow of polymer around the nanoscale additions allow the final resulting products to behave as if they were evenly spread throughout. Some companies such as EOS also create their own powders. In particular, they have one called PA 2200 which has additives, specifically titanium dioxide, to increase absorption.

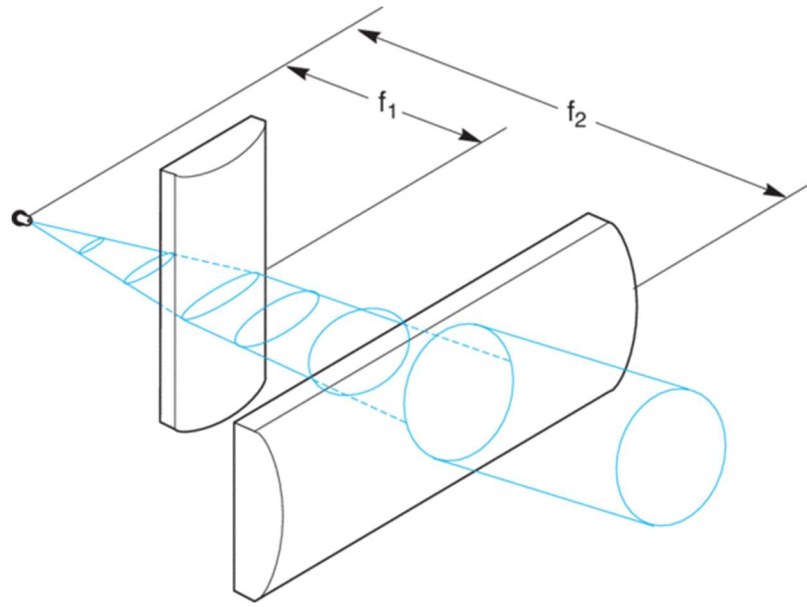
Unfortunately, thermoplastics are very expensive. One kilogram of Nylon 12 costs as much as the entire project budget. Because of this, the group will seek to make its own thermoplastic powder. One method to do this is by suspension polymerization (see Figure

3.4-2) and precipitate. These are both chemical reactions which seek to produce the polymer microparticles in solution. The chemical solvents for these reactions are usually quite toxic. Also, creating large enough batches for SLS involve complicated machinery which the group does not have access to. This leaves cryogenic grinding and coextrusion. Coextrusion, like the other two processes, is complex, and requires expensive machinery. All that is left to try is manually grinding bits of thermoplastic to achieve smaller particle sizes. In cryogenic grinding, the powder particles are inserted into a drum or tube of some length and kept at very cold temperatures, usually with liquid nitrogen. By keeping the thermoplastic particles at cold temperatures, the likelihood for cracking increases due to the increased brittle nature of the particles. Then, they are ground until a certain particle size distribution is met. The group will attempt their own version of cryogenic grinding by grinding acrylonitrile butadiene styrene (ABS) pellets kept cold with dry ice in a coffee grinder. ABS is a typical thermoplastic found in extruder-based 3D printers and creating a powder from it could be a viable source of thermoplastic powder for the SLS system. Preliminary tests show a powder can be created in this manner with particle sizes approaching sub 100 microns in diameter.

Other materials used in the SLS system will need to be tested thoroughly to achieve the correct process parameters. Because the group does not have a mechanical engineer or thermal modelling capabilities, the simulation events involving sintering of polymers cannot be predicted to a good degree. This means, with any material, the testing of the material under many conditions will be necessary to establish whether or not the material can be sintered in the first place.

### 3.4.3.4 Beam Shaping

Laser diodes have highly elliptical beam shapes. In SLS systems, the pixel dimensions on the powder bed are defined by the spot size and shape of the laser beam. Ideally, the spot size will be equal in both the x and y directions, which means the spot should be ideally gaussian. To fix the ellipticity of the laser diode, beam shaping methods can be used.

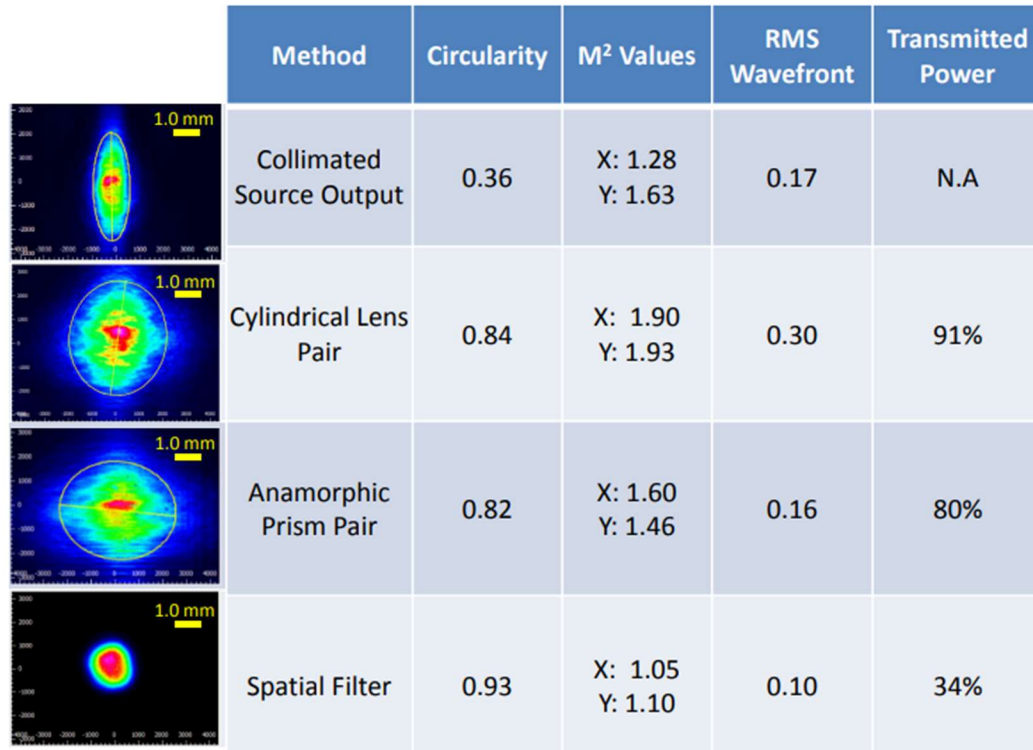


**Figure 3.4-3 A Cylindrical Lens Pair Circularizes and Collimates a Beam**

One method of beam shaping involves using a cylindrical lens pair. This method of beam shaping is shown in Figure 3.4-3. A cylindrical lens differs from a spherical lens by focusing only along one axis. In this case, the axis is either the slow or the fast axis of the laser diode. By using different focal lengths, the axes of the elliptical beam can be both circularized and collimated. This cylindrical lens pair would effectively replace the collimating lens in the optical layout.

Figure 3.4-3 shows a cylindrical lens pair. The ratio of  $f_1$  to  $f_2$  is the same as the ratio of the slow diverging axis to the fast-diverging axis. Beam shaping with cylindrical lenses results in a circularity of 0.84, which is much better than the uncircularized value of 0.36.

Another method of beam shaping is by using an anamorphic prism pair. In this method, prisms are used to resize the beam along one axis. This is similar to the cylindrical lens method and results in a similar circularity of around 0.82. However, the anamorphic prism pair is much more sensitive to misalignment. The prism pair also displaces the optical axis, which would be a considerable challenge to incorporate into the SLS printer. If a prism pair were used, the optical axis would shift within the laser module. For this reason, the anamorphic prism pair will not be considered.



**Figure 3.4-4 Comparison of Beam Shaping Methods**

The last option for beam shaping is by using a spatial filter. A spatial filter is just a small hole that the beam waist passes through such that the excess lobes are cut off. Of course, this results in a tremendous loss in power, but the highest circularity, at 34% input and 0.93 respectively. This is also the cheapest option.

Out of these options, if beam shaping will be incorporated into the final design, cylindrical lenses will be used. The benefits are clear: both circularization and collimation can be achieved using two cylindrical lenses. They are less expensive and easier to use than anamorphic prism pairs, and the power loss is negligible.

## 4. Standards and Design Constraints

### 4.1 Standards

The design of a newly engineered product should follow standards. This will help the product gain acceptance into the market, ensure safety of its developers and users, as well as help further iterations of the product in its future technological development. In this section, standards will be discussed and how they are applied to the creation of the SLS printer.

#### 4.1.1 NEMA ICS 16-2001

According to [nema.org](http://nema.org), the National Electrical Manufacturers Association (NEMA) is an ANSI-accredited Standards Developing Organization made up of business leaders, electrical experts, engineers, scientists, and technicians. [5] The NEMA ICS 16 is an NEMA standard for servo and stepper motors. The standard stepper motors are NEMA 14, NEMA 17, NEMA 23, and NEMA 34. The two-digit number refers to the tenths of an inch the faceplate of the stepper motor is. For instance, a NEMA 17 motor has a faceplate of 1.7 inches by 1.7 inches. The importance of the standard is interchangeability within our design. If we were to use the NEMA 23 standard stepper motor, the form, fit, and functionality will stay consistent no matter the supplier. This ultimately helps us out in our design as we do not have to design our motor system off of a single motor from a single supplier. We are free to choose the motor that best fits our budget without impacting the design.

#### 4.1.2 ASME B1.5-1997

The American Society of Mechanical Engineers (ASME) is responsible for implemented the ACME screw thread standard known as ASME B1.5-1997. The ACME screw thread was designed for motion transfer, so all ACME screws are used as lead screws. The standard states that the included flank angle, which is the angle between the surfaces of a thread, must be 29 degrees. [6] The height of each thread should also equate to the thickness of the thread, also known as the pitch.

#### 4.1.3 Optical Standards

There are many standards describing appropriate and safe use of optical elements in engineering design. ANSI Z136 is a series of standards aimed at protecting the people who will be using products as well as engineers who will be designing the products. Z136.1 is ANSI's "Safe Use of Lasers" and describes how to properly handle and label lasers and related equipment. These include the definitions for eyewear, engineering safety interlocks, signage, etc. Z136.9 is the "Safe Use of Lasers in Manufacturing Environments" which deals with technology that processes materials and creates new

components. This would be useful to abide by to ensure the SLS printer is as safe as possible.

ANSI B11.21 is another safety requirement, aimed also at processing equipment: “Safety Requirements for Machine Tools Using Lasers for Processing Materials.” This standard applies to machine tools which use lasers to process materials, much like SLS printers. It describes some of the hazards involved with using these machines and the protective measures that should be incorporated to protect its users.

A division of the FDA, the Center for Devices and Radiological Health (CDRH), has published a standard of which all lasers products that have been manufactured beyond 1976 must comply, called 21 CFR Part 1040. This is also known as the Federal Laser Product Performance Standard and is conveniently described in ANSI Z136.1.

The National Fire Protection Association (NFPA) provides the minimum fire protection requirements for products which use lasers, the installation of lasers, and the design of lasers. These are relevant for the SLS printer because the housing and design of the laser module and SLS printer itself must be built such that reflected light from the laser will not induce combustion of the surrounding environment within the printer. For example, the NFPA standards for laser use describe material reactants, which may also apply to the printer’s material selection.

The International Electrotechnical Commission (IEC) have developed many international standards regarding laser safety. There are a few in particular which could apply to the design of the SLS printer. IEC 60825-1, “Safety of Laser Products – Part 1,” details the safe use of lasers in the wavelength range of 180 nm to 1 mm. Another standard, IEC 60825-4, “Safety of Laser Products – Part 4,” describes the specification requirements for laser guard design, both temporary and permanent. This may be useful in designing a viewing window and for ensuring the enclosure of the SLS printer is safe.

The International Standards Organization (ISO) have a set of laser safety standards for the use of lasers in processing. Specifically, ISO 11553, “Safety of Machinery,” describes, in three parts, the requirements of laser processing machines and specifies the safety requirements relating to radiation hazards as well as hazards produced by materials and substances upon which radiation is incident. This would certainly apply to the SLS printer design.

#### 4.1.4 NEMA Power Cords

Power plugs and receptacles in the US follow NEMA established standards. There are multiple divisions of plugs that define electrical specifications such as voltage and current as well as physical characteristics of the plug like having a locking system. Most commonly used outlets support NEMA 1 - 15, NEMA 1 - 15-P (P represents the polarized pins), and NEMA 5 – 15, NEMA 5 – 20, or NEMA 5 – 30 in both polarized and non-polarized form. Where NEMA 1 is the typical two straight prong outlet, while NEMA 5 has the same two straight prongs as well as a ground pin. Both NEMA 1 and 5 handle a voltage of 125V. The second pair of digits represents the amps of current capable of



conducting. Lastly, the presence of a P at the end tells us that the two prongs are polarized. Given that these two are the only widely available plugs we are limited to using a power cord that follows one of these standards. However, they are more than capable of delivering all the power needed for our device. Thus, our product design will not be negatively affected by the use of any of these standards.

## 4.1.5 Software Standards

NPR 7123.1 is NASA's procedural requirement specification for the design and implementation of systems. In order to facilitate a well-structured design process built on sound engineering principles, and the creation of a robust and well-made software platform for our senior design project, the software design and creation will adhere to a subset of these principles based on their relevance to the project and importance (based on the criticality of the system). The standard can be broken up into 3 main phases, each with their own sub steps to insure well-made systems.

The first is the phase, the Identification Phase can be further divided into the following:

1. Requirements Definition Process
  - a. Stakeholder Expectations Definition
  - b. Technical Requirements Definition
2. Technical Solution Definition Process
  - a. Logical Decomposition
  - b. Design Solution Definition

Of these sub-phases, we will be using a modified version to better align with the nature of our project. Particularly, since our project is not funded and therefore has no stakeholders' certain modifications must be made. We will therefore replace the Requirements Definition Process, where stakeholders may define their requirements for the project to a Requirements Identification Phase where we determine our own requirements based on the project's goals. As such, the first sub-phase of the Identification Phase will be changed as follows while the second sub-phase will remain the same:

1. Requirements Identification
  - a. Define Project Goals
  - b. Define Project Constraints
  - c. Technical Requirements Definition
2. Technical Solution Definition Process
  - a. Logical Decomposition
  - b. Design Solution Definition

These processes represent a general algorithm (in somewhat plain language) used to generate the underlying requirements for a design, convert those requirements into some well-structured model (logical, behavioral, mathematical, etc), and from that model generate a solution that will satisfy the initial goals/requirements for that design. In accordance with NPR 7123.1 we should not only use this procedure for the overall design, but also use it recursively on the resulting designs until all designs are fully defined down

to components that can be “built, bought, or reused”. Because we are using this specifically for software however, we can further refine the algorithm by applying it until all designs produced are fully defined down to components found in the language we are using, this is a bit excessive however and can be relaxed a bit by including components found in the language’s standard library and publicly available open-source libraries. In this way we can define a set of steps that will take us from a basic idea of our design and its requirements all the way to a logically complete pseudocode that can easily be converted into the Turing complete language of our choice (or potentially something Turing incomplete if needed for runtime analysis reasons). Additionally, with each “level” of recursion, we can define a “layer” of the design that can be implemented as an abstraction layer whether in software or hardware so that layers don't need to worry about how other layers function, this can help simplify the definition of design requirements at each layer and also helps to organize things with good coding practices.

After the Identification Phase is the Product Realization Phase where the project is made, validated and integrated. Due to the nature of our project, we will have to make some changes here starting with the name, we will be calling this phase our Project Production Phase because it is where we will produce the final project. Like the previous phase, this next phase will be broken up into parts, there will be three sub-phases where the process will transition from design to development, evaluate the components, and implement a final product, they will be broken down as follows:

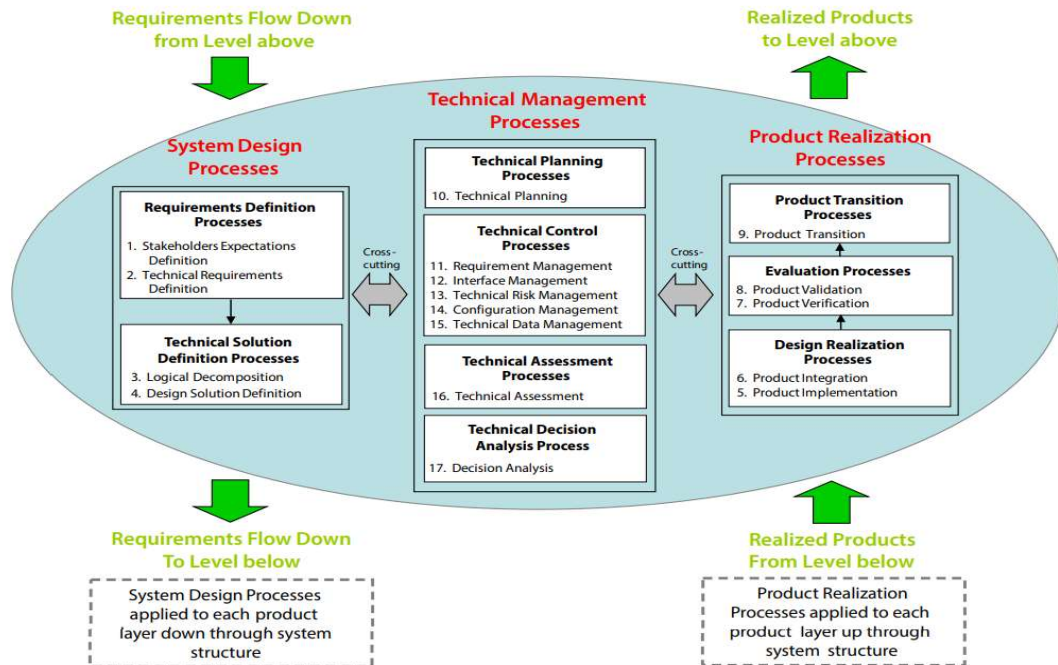
1. Realization Phase
  - a. Implementation
  - b. Integration
2. Evaluation Phase
  - a. Verification
  - b. Validation
3. Transition Phase
  - a. Transition up a Layer

This phase seems to work perfectly for our purposes and does not require any modification, so we will be using it as is. We start with the various components of the lowest abstraction layer (the last one made) and begin by implementing them or adding in premade versions. After each component is ready, we can integrate it with the other parts of its layer and begin testing; we verify that each component works on its own, and with any other components in the layer that it may need to work with, and then when all components are tested separately, we can also validate the layer as a whole. After a layer is fully validated, it is ready to be used in the next layer so we can begin making the components of the next higher layer that will be using the components from the previous one. Similar to the recursive process of designing simpler and simpler components, this process will also be used recursively on each layer but in the reverse order until we have built and tested all of the most complex components.

This last phase will be a little bit different in that it isn't completed sequentially with the others, but rather in parallel. While the identification phase is used to make the designs, and the production phase helps to realize them, and must therefore be executed after

(because the designs need to be made before they can be produced), the management phase helps to ensure the project progresses smoothly and must therefore be followed throughout the other processes. The Management Phase can break down into 4 sub-phases as follows:

1. Technical Planning Process
2. Technical Control Processes
  - a. Requirement Management
  - b. Interface Management
  - c. Technical Risk Management
  - d. Configuration Management
  - e. Technical Data Management
3. Technical Assessment Process
4. Technical Decision Analysis Process



**Figure 4.1-1 NASA NPR 7123.1 Overview [7]**

Figure 4.1-1 is a general overview of the design process in accordance with NASA’s NPR 7123.1 specification. As mentioned earlier, the Management Phase helps with the administrative tasks of ensuring proper progress in the other phases. Once our designs are made, we start by planning their implementation, we can then evaluate our design’s risks, requirements, and interfaces, as well as how they need to be configured in order to adjust our plans accordingly and as needed. After implementation, we can perform a technical assessment and analyze our decisions to further evolve our plans if needed or otherwise confirm the steps completion and function.

## 4.2 Design Constraints

Design of a marketable product must follow constraints. Some products feature political, social, and ethical constraints, while most products feature economic, time, and safety constraints. The relevant constraints will be described, as well as how they impact the design of the SLS printer.

### 4.2.1 Economic and Time Constraints

The SLS 3D printer project is fully funded by the group. Therefore, the biggest limitation when making design decisions is cost. The economic constraint affects the overall size of the built area. As SLS printers have a larger footprint to build area ratio than a conventional thermoplastic 3D printer, the build area will be smaller than a thermoplastic 3D printer of equal footprint. A large build area would greatly increase the size of the printer which increases total cost of the project. Therefore, the build area must be relatively small. The economic constraint also affects material selection and fabrication. Most 3D printers on the market use a metal enclosure to house the 3D printer. As metal fabrication is quite pricy, the best material to build the SLS printer is wood. Wood is a relatively inexpensive commodity, and easy to work with. Wood can be cut by using a saw for general cuts or a laser cutter for precise cuts. Access to a laser cutter is available to UCF students in the Innovation Lab free of charge. The final limitation due to the economic situation is part selection and design. We are not able to purchase kits for our mechanical system as they are too costly and hard to find. The mechanical system will be designed and constructed with the budget in mind. As we are not able to cut costs in all subsystems, building our own mechanical system will greatly reduce overall cost. In order to further cut cost, we will design, and 3D print the brackets and mounts used in the 3D printer assembly.

There is also an inherent time constraint placed on our project due to the nature of senior design. In order to comply with the class, we must meet documentation and presentation deadlines. Due to the many subsystems in your project, testing has started early on, and prototype fabrication is in mind. The time constraints also place a limitation on part selection. Most electrical components are placed on backorder, and they can have lead times up to two years. Since we have until the Spring 2022 semester, we must ensure that there will be inventory in the future for our part selection. This is accomplished by checking current inventory and the lead times for the selected parts.

### 4.2.2 Health and Safety constraints

With any marketable product, it is important to consider the impact on user health the product may have, as well as any safety requirements or training users may need before operating or using the product. There are several aspects of the SLS printing machine that, without proper safety features or guidelines, could end up causing temporal or permanent harm to the user. Here we will be splitting the different possible causes of harm as well as the corresponding feature or guideline that will be implemented to protect the user from any harm into three parts: electrical safety, laser safety, and safety with moving parts.

## 4.2.2.1 Electrical Safety

Since we have a product that is powered from a 120V wall outlet, it is essential to follow all precautions to avoid possible electric shock, fire, component explosion, short circuit, etc. The first precaution that we will take is the use of a NEMA standard power cord, more specifically NEMA 5 – 15. This means that we will have the presence of ground pin. By making sure all metal components in the printer that are not supposed to be electrically charged at any point have a connection to ground we are ensuring that electricity has a closed path to ground and therefore preventing these metals from becoming charged. If this is not done, we could have some of these metals be accidentally charged and be a shock hazard for the user.

The next safety concern is exposed electric wires. This is especially dangerous when dealing with the AC/DC converter. All electrical connections with the transformer must be properly isolated. This will be mostly important during the design, manufacturing, and testing phase as we might need to have contact with those connections to troubleshoot. Additionally, exposed wires at any module, even if it is low voltage, could be a hazard given that our printing material are powder particles of different materials. In the case that enough powder finds its way into exposed wire this could result in melted materials in undesired places that could affect the proper functioning of the device.

Another important aspect of safety to keep in mind for electrical design is selecting components that are properly rated for whatever voltage, current, or power that they are expected to handle. Using parts not designed to support the electrical characteristics placed on them might result in part failure. This could produce fire or explosions. To avoid any harm from such cause, every component is carefully selected and detailed in the design section of the corresponding module they belong to.

The last aspect of electrical safety to worry about is heat. All components dissipate some power in the form of heat. This thermal energy needs to be properly dissipated away from the device. Thus, we need to make sure all electronics are properly placed in a ventilated area and if provide heat sinks to components prompt to overheating.

## 4.2.2.2 Laser Safety

Safe use of lasers has long been identified by ANSI Z136.1. In any environment which uses lasers, care must be taken to avoid damage of the environment and its users by the laser. There are many aspects of an environment which must be considered when evaluating the required laser safety constraints.

Perhaps the most important aspect of laser safety is the output power of the laser beam itself. The laser used in this SLS system will have an output power ranging from somewhere around 1W to a maximum of 5W operating continuously at 447 nm. This makes the laser a Class 4 laser, which presents serious hazards. Direct view of this laser with the eye can result in permanent eye damage. Contact of the skin with this laser may result in immediate burns. Materials in contact with the focused laser beam may be able to ignite spontaneously. For these reasons, development of the laser system must occur

with considerable personal protective gear, such as laser safety goggles rated for Optical Density 4+ at 447 nm, as well as hand coverings. Because diffuse reflections can still cause significant eye and skin damage, it is crucial to ensure every person in the development environment wears the necessary PPE and is aware of the laser at all times. The minimum number of users within the laboratory environment should be enforced, and anyone outside the laboratory environment should be aware of high-power laser testing before entering. As for the user, the laser will be enclosed, and no interactions should occur. However, it is important to notify the user that dismantling the enclosure could increase the risk of exposure to Class 4 laser radiation.

The beam path of the laser must also be considered. Because the laser will be collimated and focused, reflections of the beam will occur between interfaces. These reflections must be considered in the design of the enclosure. Thankfully, the beam path is not bending or going around any corners, so reflections from mirrors will not be of worry. Further, the beam path will be quite short, so the chance of accidental exposure to the beam path will be minimized. However, the power density of the beam as it travels through the optical system will change dramatically. At the focal point, the power density is extremely high and could cause serious wounds needing immediate care. It will be important to ensure that access to the lasing environment be restricted when the laser is operational.

The interaction of the laser beam with the powder itself should be identified as potentially hazardous. Reflections of the beam from the powder can still result in serious eye injury. The point of sintering should never be examined with a naked eye while the laser is on. Suitably rated safety goggles must be worn at all times if examining the laser-powder environment while the laser is operational

Proper control measures should be considered when designing the SLS printer. These control measures include things such as: symbols, in the design of ANSI Z535 or IEC 60825-1, to alert unaware users of potentially hazardous laser light; interlocks, to ensure operation of the laser is completely intended, and that the powder environment cannot be accessible during laser operation; illuminated operation signs, alerting users in the area that the laser is on, and noncompliance with enclosure safety standards could result in unintentionally leaked laser radiation; emergency deactivation switch, in case the interlock fails and the laser must be turned off despite approved functioning; or an emergency beam block, to completely enclose hide the beam incase accidental opening of the enclosure, while ensuring no damage occurs to the laser module itself.

### 4.2.2.3 Safety of Moving Parts

In our printer there will be three main modules that will be made up of moving parts powered by motors. Those being the moving plates that will contain the print and the printing powder, the swiping mechanism that places more powder into our print to add more layers, and the X-Y scanning system that moves the laser around. Any of those moving modules could be a hazard for the user if is exposed and proper guidelines are not put in place. Thus, it is important that all of the printer is enclosed restricting all access to any moving parts while in operation. Additionally, an emergency stop program can be

added in order to shut off the power to the motors if the enclosure is opened during the printing process. Lastly, a manual shut off switch could work as the last line of defense against an emergency.

### 4.2.3 Environmental Constraints

Another key constraint that we have for this project is product waste and energy consumption. Because of our design that uses printing powder as the support structure and it is placed across the entire printing bed, the amount of material used for each print will be significantly larger than the material on the final product. In some cases, the product will consist of less than 10% of the entire material. Independent of the material being used, such low efficiencies would create a concerning amount of waste. Overlooking this factor would be unethical engineering practice. In order to avoid producing waste as such high rate, we would incorporate in our design a recycling mechanism. In which the powders not fused together to make the final print, would be recovered, and stored for future use. This way the only waste material would come from undesired powder sintered into our final print. This undesired used material would be minimized by carefully designing each module. For example, the smaller the laser beam then the more accuracy is achieved on the sintering process, and thus less neighboring material would be fused to our print. Additionally, different scanning techniques might be used to reduce unwanted sintering, as an example, it could be a better technique to do multiple short passes of the laser in an area rather than just one longer pass. In short, achieving a high material reutilization percentage will constantly be a priority on every aspect of the design.

### 4.2.4 Manufacturability Constraints

When building an engineering product, there may be constraints on manufacturability and sustainability. These constraints can pose questions such as, “Can this product be manufactured reliably? Is this product sustainable?” In the case of the SLS printer, its design would need to be overhauled to ensure sustainable production. The working prototype will feature inexpensive products that would deteriorate over time. The most obvious design replacement would be the wooden exterior. Wood is simply not a good design material for a manufactured product. It is cheap and lightweight, which is great for prototypes, but the final design would certainly feature a full metal enclosure.

Further, the tolerances of the design would need to be minimized in such a way that two SLS printers would print the same. This is actually an enormous task, because the print quality relies on a very large number of variables. In fact, some manufacturers create calibration testing procedures for customers and tell them that they will need to do their own calibration to get the best print quality. Indeed, manufacturing perfectly replicable SLS printers may not be possible, but its design should be a goal.

Other manufacturability constraints arise from the constraints on the products used within the SLS printer. For example, the laser diode had a lead time of 11 weeks, and most of the blue laser diodes from Osram become obsolete after some time. Agreement with other

manufacturers would need to be reached to ensure the products used within the SLS printer are constantly available.

In the same vein, it is important to make sure manufacturability of the SLS printer does not impose strict material requirements that are impossible to fulfill. For example, if the enclosure is made out of a particular type of metal, it should not be an alloy that is difficult to make or source naturally. Using materials that are difficult to source naturally only sets up the product for increased cost or, at worst, failure. Likewise, using materials that are, from the beginning, difficult to create, also suggests the product will see increased costs in the future.

Lastly, the durability of the product itself must be designed for. It would be pointless to manufacture a product that does not last longer than two months. To do this, the focus of design improvements would lie in the areas of moving parts. Specifically, the motors that run the x-y scanning system, the sweeper blade, and the print bed, and the powder delivery system would all need to be tested for durability and perhaps exchanged with different components to ensure a sufficient lifetime. Also, optical elements such as the laser diode and the lenses would need to be tested for lifetime at different operating temperatures and powers to further optimize the product's time of use.



## 5. Project Hardware and Software Design

### 5.1 Mechanical Design

#### 5.1.1 Enclosure

Unlike most traditional 3d printers which can rely on their frame for support, our design will make use of an external enclosure for stability and structure, like the exoskeleton of a beetle or crustacean. Our enclosure will have two primary parts, an electronics housing section, and a print section, the latter of which will house and support our print bed, powder reservoir, powder delivery system, and x-y scanner/laser sintering system.

The electronics section will take the form of a triangular or trapezoidal extrusion coming out from the base of the print section and will include the power supply, computing unit or units, and the screen for the user interface mounted on the outer panel for convenient use. The side of the electronics section will also be removable or on a hinge for easy access to the electronics during the build and testing phases, as well as to facilitate general repairability. Being separate from the print section also means that we can seal the sections to ensure that no particles from the print powder make it into contact with sensitive electronics and can safely add ventilation and even a fan if need be without interfering with the powder distribution.

The print housing section will be a large rectangular section made to hold the main part of the printer. It will not only hold the various print systems, but also provide support and structure for them as well. Most of this part of the housing will be fixed together, but one side along the face with the largest area and opposite the electronics section will be removable or hinged like in the electronics section. For the print section, this is of particular importance because it not only allows us to build and test the printer more easily, but it is also a needed feature in order to remove our finished print.

Despite being two separate sections, the electronics and print sections will be fixed together in order to form a single device. The enclosure will also provide stability for the printer by having a wider base on one axis due to the electronics section and being wider on the other axis due to the printer having a reservoir and excess powder reservoir on either side of the print bed.

#### 5.1.2 Reservoir

In order to achieve SLS 3D printing, the powder must be properly stored and contained. Due to the nature of laser sintering, the powder has to be contained in multiple areas. The first reservoir is the powder reservoir. The powder reservoir is where the powder is initially stored. The second reservoir is the build reservoir. The build reservoir holds the powder that is in the sintering process. In an ideal sense, both of the reservoirs will have equal volume and only these reservoirs would be needed to build an SLS printer. In practice, powder delivery will not yield a perfect transfer to the other reservoir. Some powder will

be pushed to the sides or will be built up in one area. Two design modifications will be used in order to accommodate for the imperfections.

Since the powder is transferred via pushing the powder over to the next reservoir, there is a possibility that powder can build up at the end of the build reservoir. This could be problematic since the excess powder can fall back into the build reservoir during sintering. If this occurs, the laser could sinter the excess powder onto the planned sintering area and ruin the print. The fix to this issue is adding a third reservoir for the excess powder. If there happens to be excess powder at the end of the powder delivery, the sweeper mechanism, mentioned in 5.1.4 Sweeper Subsystem, will continue to push the powder into the excess powder reservoir. Ensuring that powder is consistently even in the build area will lead to more reliable and accurate printing.

Since we are accounting for excess powder build up, more powder is needed to accommodate for the losses. To ensure that there is always enough powder to cover the print bed, we will need to make the powder reservoir larger than the build reservoir. Referring to the specifications, we need to have an overall build volume equivalent to a 4-inch cube. The upper reservoir needs to have a height greater than 4 inches since the build plate will take up some of the volume. Exact dimensions are discussed in 5.1.3 Plate Subsystem. To account for the loss in powder, the powder reservoir area will be 5 in by 4 in.

At the end of the print, the powder reservoir should be empty, and the build reservoir should be full. The reservoir must have a removable panel in order to access the finished print and clean the powder. The panel will be secured using latches on the side and a hinge at the bottom. When unlatched, the panel will pivot along the bottom hinge's axis of rotation. A model of the reservoir with the panel removed is shown in Figure 5.1-1.

### 5.1.3 Plate Subsystem

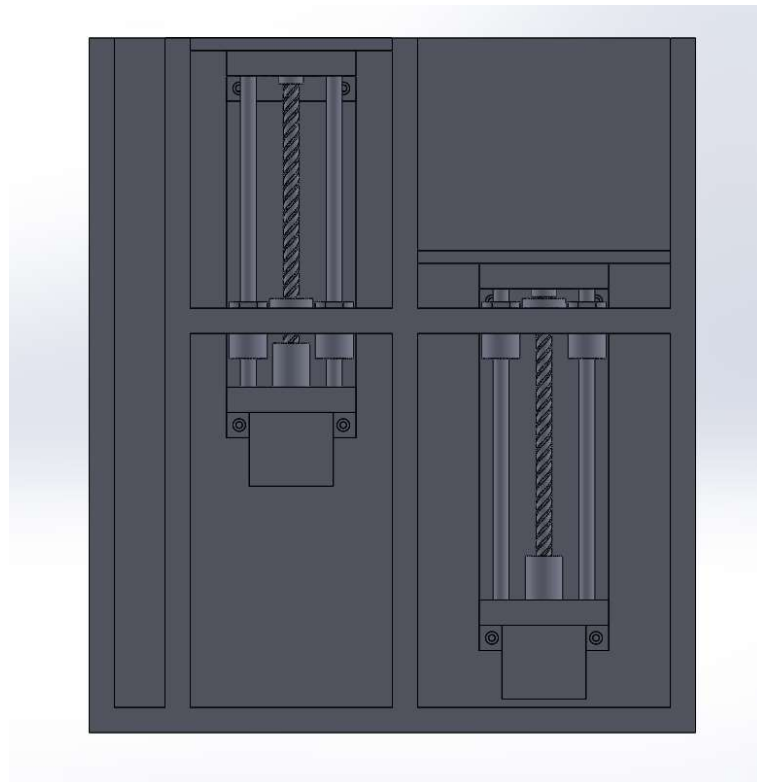
As mentioned in the 5.1.4 Sweeper Subsystem, the build reservoir and powder reservoir have their own plate system. The plate system is responsible for changing the volume of the reservoir. The build and powder reservoir are split into two sections: an upper cavity and a lower cavity. The upper cavity houses the powder and the plate. The lower cavity is primarily empty and is used to house the other end of the linear actuator. The plate system will use a modified lead screw design for the linear motion up and down. Instead of fixing the flanged nut onto the part we wish to move, we will fix it to the platform that splits the reservoir in two sections. Mounting the flanged nut allows the whole linear actuator assembly to move relative to the flange nut rather than the flanged nut moving relative to the assembly. The motor end of the linear actuator is housed in the lower cavity while the plate is housed in the upper cavity.

The motor end of the linear actuator contains a 3D printed bracket that fixes the motor and two smooth guide rods in place. The flanged nut for the lead screw is mounted to the middle platform. Two linear bearings will also be mounted to the middle platform. In order to mount the linear bearings, we design a flanged housing for the linear bearing. The linear bearings are used to guide the smooth rod in linear motion. The plate end of the

linear actuator contains another 3D printed bracket that fixes a pillow bearing and the other end of the guide rods. The bracket is then mounted to the bottom of the plate.

The plates for the build reservoir and the powder reservoir have different areas. The build plate is 4 in. by 4 in. while the powder plate is 5 in. by 4 in. Both plates are a 1/4 inch thick. Accounting for volume that the parts in the upper reservoir will occupy, the reservoir depth can be determined. For the build volume to be a 4 in. cube, the upper reservoirs for the build and powder platform must be 5.1 inches deep. The bottom reservoir is 7.4 inches deep so that the linear actuator can move the build plate through the full range of motion. The upper and lower powder reservoirs have the same height as the upper and lower build reservoirs, respectively. The total outer dimensions of the reservoirs with the plate subsystem are 4.5 in. x 11 in. x 13 in. A 3D model of the designed plate system and the reservoirs are shown in Figure 5.1-1.

Since the lead screw used is right-hand threaded, as the lead screw spins counterclockwise, the plate will move up. For plate calibration purposes, a bump switch will be fixed to the motor end of the linear actuator. Before powder is loaded into the printer, the plates will move up until the bump switch is activated. At this point, the code understands it is at the top position. The build plate will stay at the top position and the powder plate will move to the bottom position. The printer is then ready for the powder to be loaded.



**Figure 5.1-1: Plate System 3D Model**

## 5.1.4 Sweeper Subsystem

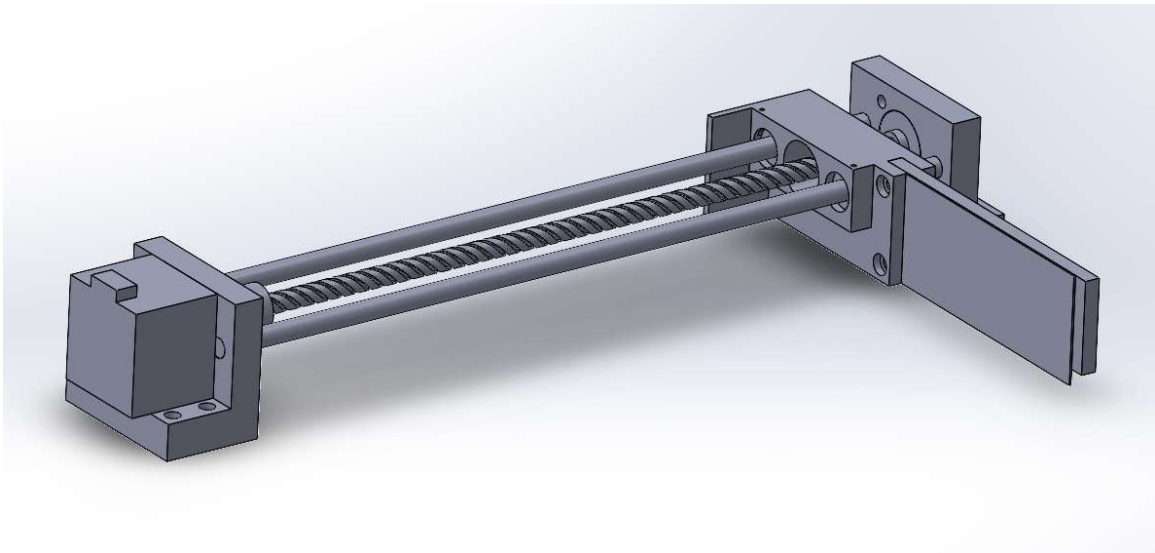
The SLS printer consists of three reservoirs: the powder reservoir, the build reservoir, and the excess-powder reservoir. The powder reservoir houses the powder for printing. The printer transfers the powder from the powder reservoir to the build reservoir using a sweeper mechanism. A good analogy for the sweeper mechanism is scooping flour with a measuring cup. When one scoops up flour with a measuring cup, there will be excess flour at the top. One would take their finger and run past it in order to level out the flour. The same goes with the sweeper mechanism. Both the build and print reservoir contain vertically moving plates. Before the print starts, the build plate will be located at the upmost position while the powder plate will be at the bottom most. When the print is initiated, the powder plate will move up a certain distance and the build plate will move down the same distance. The sweeper mechanism starts on the powder side and runs across the powder reservoir so it can “sweep” the powder to the build side. In the case that there is excess powder, the sweeper will continue to push powder across the built reservoir and into the excess-powder reservoir. Once all powder is delivered, the sweeper mechanism will return to its start position and wait until powder needs to be transmitted again.

The sweeper mechanism uses a lead screw linear actuator system. When designing the sweeper mechanism, we found that there are two possible ways to fix the guide rods. The first way is placing a guide rod on each side of the reservoir. The benefit of using the first way is that it ensures that each side is level with each other. Although this orientation was considered, there are some issues with it when implementing it in our design. If one motor is used, it will push the sweeper arm unevenly on one side which could possibly lead to future problems. A stepper motor and lead screw would need to be placed on each side of the reservoir to push each side of the sweeper arm evenly. Since more parts would be needed, the first method would increase the total cost of the project.

The second way is placing both guide rods next to each other with the lead screw running through the middle. The second way would have the same orientation as the plate subsystem shown in Figure 5.1-1. The concern with the second method is torque placed on the system. The sweeper arm will place static torque on the base bracket since it is mounted to only one side. Although it is a concern, it is not a major issue since the arm is not very long. It would become more of a problem if the reservoirs were larger. Using this method, only one motor is needed. The second design will also be easier to mount as it only need to be mounted to one side. Due to the lower cost and ease of design implementation, the second guide rod orientation will be used.

Using the second method, we began modeling the sweeper subsystem. The sweeper arm must be able to move from the end of the powder reservoir to the end of the excess-powder reservoir. The distance between both ends is 10.5 in. As mentioned previously, the start position is on the powder reservoir end. A bump switch will be fixed to the bracket on the powder reservoir side. The bump switch will be used to calibrate the sweeper subsystem and designate the start position. The sweeper bracket moves along the track system and is responsible for holding the sweeper arm. The bracket houses two linear bearing and the flanged nut for the lead screw is mounted to it.

The sweeper arm is made of a ¼ inch piece of wood that stretches across the 4 in. reservoir length, and it is mounted to the sweeper bracket. Attached to the sweeper arm is a strip of sheet metal. The strip of sheet metal is used to make up for the difference in high from the reservoir and the base of the sweeper. The sheet metal ensure that the sweeper is moving as much available powder in each run as possible. The sweeper is driven by one stepper motor as it should have sufficient torque to move the sweeper along with the powder. The sweeper mechanism will be directly mounted to a plate that sits on top of the reservoirs. A 3D model of the sweeper subsystem is shown in Figure 5.1-2.



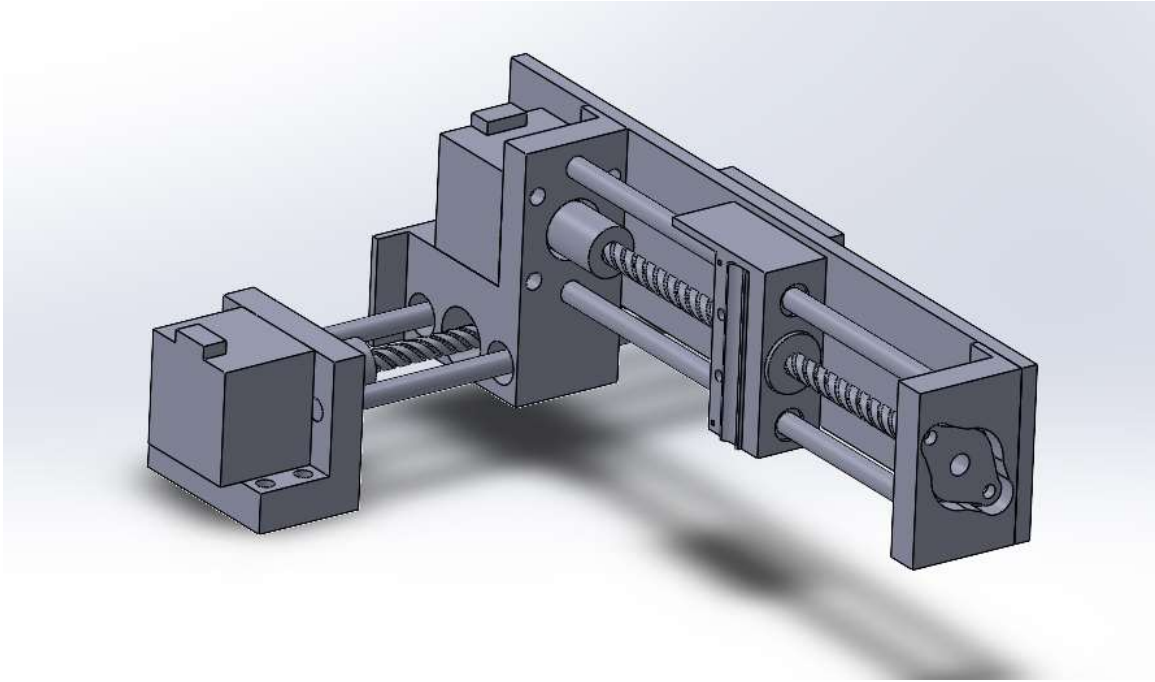
**Figure 5.1-2: Sweeper Subsystem**

### 5.1.5 X-Y Track System

The goal for the X-Y Track Subsystem is to move the head of the laser module in a cartesian plane. The track subsystem must be able to move the laser module around the whole build area. The basis of the system will use a lead screw linear actuator to control each axis. Therefore, two linear actuators will be used to move the laser module. The y-axis linear actuator is responsible for moving the laser module. The y-axis design is similar to the linear actuator design of the plate system except the flange nut is mounted to the base of the laser module. The 3D printed bracket mounted to the motor fixes the motor and two smooth guide rods in place. It also includes two linear bearings running perpendicular to the y-axis guide rods. This bracket will also be used to join the x-axis and the y-axis. Along with the flange nut, the base of the laser module will have two linear bearing running parallel to the flange nut. The bracket located on the other end of the linear actuator will act as an end cap to fix the two guide rods. A pillow bearing will be used to fix the lead screw to end cap as well. A bump switch is fixed to the motor end of the y-axis and will be used for y-axis calibration.

The x-axis linear actuator is responsible for moving the y-axis linear actuator in the x-direction. The x-axis linear actuator will use a similar system to the sweeping mechanism. Using the same design rationalization as the sweeper mechanism, the assembly will be

easier to mount if both guided rods are fixed on one side rather than fixed on both sides of the reservoir. Therefore, the guide rods are located on each side of the lead screw. The end bracket mounted to the motor fixes the guide rods to the motor. A bump switch will be mounted to the motor end bracket and will be used for x-axis calibration. The flanged nut of the x-axis will be fixed to the motor end bracket of the y-axis. The linear bearings housed in the motor end bracket of the y-axis linear actuator will ride on the x-axis guide rods. A 3D model of the X-Y track subsystem is shown in Figure 5.1-3.

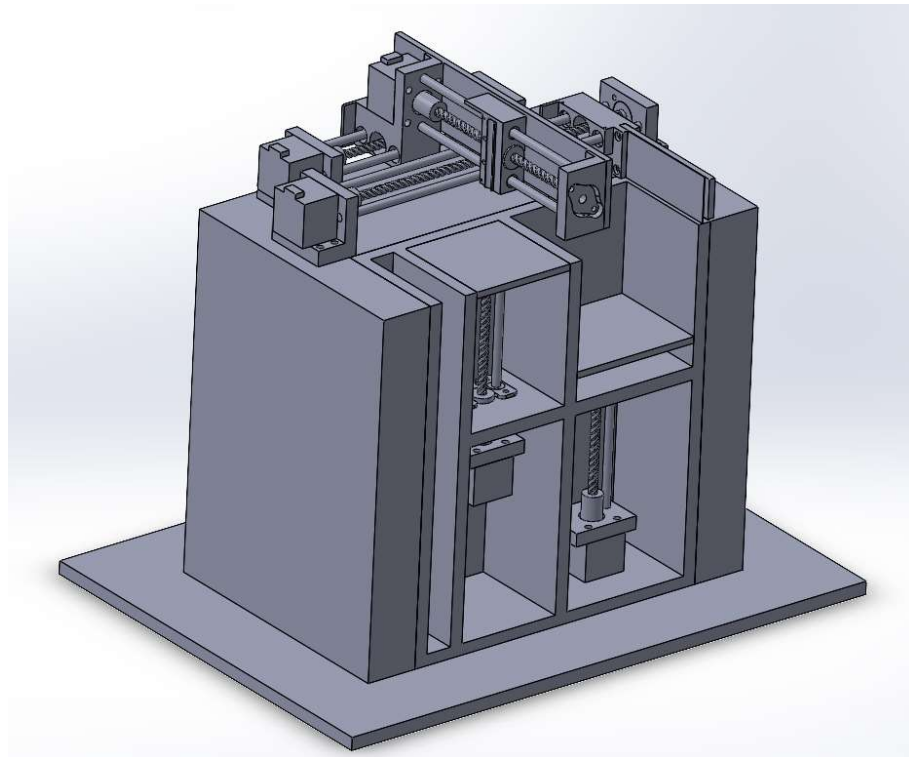


**Figure 5.1-3: X-Y Track Subsystem**

## 5.1.6 Final Assembly

The final mechanical design task is to join all of the subsystems into one assembly. The first step is to design a base that the subsystems can be mounted to. The longest subsystem is the sweeper mechanism therefore the mounting base must accommodate for this length. The sweeper mechanism measures 15.05 inches end to end. Therefore, the length of the base is 15.05 inches. The width of the base is determined by the reservoir width and the bases of the other subassemblies. The width of the reservoir is 4.25 inches, and another 6 inches can accommodate for the X-Y track system and the sweeper mechanism. The total width of the base is 10.25 inches. The reservoir sits centered on the base with the access-panel side flush with the length edge of the base. The X-Y track system and the sweeper mechanism needs to be mounted next to the top of the reservoir. The base is extruded with a height of 13 inches everywhere excluding the reservoir area. With the base and reservoir taken care of, the X-Y track system and the sweeper mechanism needs to be mounted to the top of the base.

When designing the final assembly, the greatest concern is the interference with the X-Y track system and the sweeper mechanism. Both systems were modified so that there was enough clearance between the moving parts. The sweeper assembly is mounted closest to the reservoirs as it is responsible for moving the powder. The base X-Y track system is mounted next to the sweeper assembly, and the y-axis arm will run over the sweeper assembly. The bottom of the laser module is 2 inches above the build reservoir. Therefore, the sweeper arm must be less than 2 inches tall. The total height of the sweeper arm was decided to be 1.75 inches. This decision ensures that the two mechanisms would not collide during operation. The model of the final assembly with the access panel removed is shown in Figure 5.1-4.



**Figure 5.1-4: Final Assembly**

## 5.2 Electrical Design

Designing an SLS machine involves designing multiple electronic subsystems. These include a power module, stepper motor driver module, laser driver module, and lastly a computer processing module. All of those will be designed by us. That includes component selection, schematic and footprint design, simulation, assembly, and testing. In this section we will be detailing all of these process except for assembly and testing, both of which will be discussed later in the report.

## 5.2.1 Stepper Motor Driver Circuit

Each linear actuator mentioned in the mechanical design uses a stepper motor, therefore, there are five stepper motors. Each stepper motor will have a driver to control the rotation of the stepper motor. The stepper motor driver works by alternating the polarity of the coils within the motor. The timing has to be just right in order to achieve a smooth rotation. The stepper motor driver IC we plan on using is controlled using a step-and-direction interface. The IC takes in a PWM signal and controls the speed of the stepper motor dependent on the PWM frequency. The IC also takes a direction signal which is a logic signal used to determine the spin direction. For instance, a logic high direction signal indicates clockwise.

The stepper motor driver IC chosen for this project is the DRV8825. The DRV8825 is a 28-pin stepper motor driver IC capable of 1/32 micro stepping and a maximum drive current of 2.5A. The DRV8825 contains two H-bridge drivers responsible for driving a bipolar stepper motor. Referring to the data sheet, we can find the pin configuration and start designing the schematic. The two pins used to power the H-bridge drivers are VMA and VMB. They are connected to the same supply voltage, each with a bypass capacitor. The supply voltage used will be between 9-12V. The pins connected to the stepper motor are AOUT1, AOUT2, BOUT1, BOUT2. Polarity of the connections do not matter, but one coil should be connected to the same letter output. For instance, AOUT1 and AOUT2 should be connected to the same coil. The driver connects to the stepper motor wires via male headers.

The NEMA 17 stepper motor we are using has a rated current of 1.5 A per phase. The data sheet provides an equation to set the full-scale current of the driver. The equation is shown in Figure 5.2-1. If the sense resistor is set to 0.1 ohms, the current is equal to twice the reference voltage. Using an input of 3.3 V, a voltage divider can be created to output 1.5 V to the reference voltage pins. The voltage divider is created using a trim potentiometer in order to precisely set the reference voltage after manufacturing. Since we set the sense resistor to 0.1 ohms in our calculation, 0.1-ohm sense resistors are connected to ISENA and ISENB. The voltage divider is connected to AVREF and BVREF. The input voltage for the voltage divider is connected to pin V3P3OUT, which outputs 3.3 V.

$$I_{FS} (A) = \frac{xVREF(V)}{A_v \times R_{SENSE} (\Omega)} = \frac{xVREF(V)}{5 \times R_{SENSE} (\Omega)}$$

**Figure 5.2-1: Full-Scale Current [8]**

In order to control the micro stepping, pins MODE2, MODE1, and MODE0 are used. The micro stepping mode is indexed by the input logic of the 3 pins. For instance, an input of 011, where MODE2 is the most significant bit, will result in 8 micro steps/step. The STEP pin is used to control the speed of the motor. The pin is connected to the microcontroller which outputs a square wave signal. The step input detects the rising edge of the square wave which drives the motor to move a step. The direction of the motor is determined by the logic level of DIR pin. The logic level is set by the microcontroller, so DIR is connected





**Table 5.2-1: Stepper Motor Driver Components**

Reference Designator	Value	Tolerance	Part Number	Cost
R3, R4	0.2 $\Omega$	$\pm 1\%$	CRL0805-JW-R100ELF	0.27
R5	10 k $\Omega$	$\pm 1\%$	MCWR08X1002FTL	0.064
R6	1 M $\Omega$	$\pm 5\%$	ERJ-6GEYJ105V	0.113
C1, C2, C3	0.1 $\mu$ F	$\pm 10\%$	C0805C104K5RACAUTO	0.363
C27	100 $\mu$ F	$\pm 20\%$	MCMHR25V107M8X7	0.301
C5	0.01 $\mu$ F	$\pm 10\%$	C0805J103K3RACAUTO	0.42
C6	0.47 $\mu$ F	$\pm 20\%$	MCGPR63V107M10X13	0.542
TRIM_POT	10 k $\Omega$	$\pm 10\%$	3296W-1-103	1.26
U1	-	-	DRV8825	Located on Module

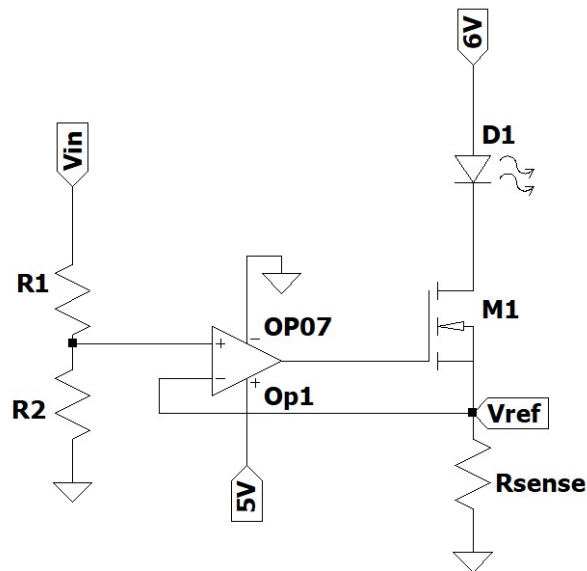
## 5.2.2 Laser Driver Circuit

The objective when designing a laser driver is delivering constant current to the diode. We will be designing the laser driver for the PLPT9 450LB\_E 5W laser diode. The operating forward voltage and operating current are located in the laser diode data sheet. The operating forward voltage for the laser diode is 4.3 V and the operating current is 3A. There are many design methods to achieve constant current. Laser drivers typically use a current driver IC. When researching for LED current driver ICs, there was a very limited set of ICs that met our output requirements. The ICs that met our output requirements have limited stock and long lead times. Due to the fact that LED current driver ICs were not easy to acquire, we did not consider them for our design. After doing more research, we found that the heart of the current driver ICs is a buck converter. A buck converter is a type of switching regulator which switches the input voltage at a set frequency in order to achieve a regulated voltage.

Our initial design used a buck converter to drive the laser diode. Texas Instruments offers a power design tool called WEBENCH that creates power converter circuits around the constraints set by the user. The input power of the laser driver will be the rectified DC signal from the AC to DC converter. The rectified DC voltage is around 11 to 13 V, therefore, I set the minimum and maximum input voltage accordingly. The output must be 4.3 V and 3 A. After running the design tool, we found that all designs contain a high efficiency buck converter. The selection of regulators was filtered by efficiency, BOM Cost, and BOM count. Since high power efficiency is optimal, we set the efficiency of the regulator to greater than 88%. As we are designing around affordability, the BOM cost is set to less than \$4.

After analyzing the circuits developed by the WEBENCH tool, we noticed some issues with using a buck converter design. Like the current driver ICs, the selection of buck converters was limited in stock. Buck converters also deliver constant current with a ripple. The output voltage and output current of switching regulators have a sawtooth waveform. An inductor and capacitor are connected to the output stage of the regulator in order to smooth the waveform. The inductor helps the current stay constant, and the capacitor helps the voltage stay constant. Even though the peak-to-peak ripple is lowered, the ripple still exists. Due to the shortage in stock and the output ripple, we continued our research for a design.

Our final design for the laser driver is based off of a voltage to current converter. A voltage-to-current converter consists of an op-amp, a power MOSFET, and a sense resistor. One benefit of using a voltage-to-current converter, rather than a buck converter IC, is the part accessibility. The components of a voltage-to-current converter are all common components that are always in stock. Another benefit of using a voltage-to-current converter is that it delivers a true constant current. The current through the diode is equal to the current across the sense resistor. The current through the sense resistor is dependent on the input voltage of the op-amp. The output of the op-amp is connected to the gate pin of a power MOSFET. The inverting input of the op-amp and the sense resistor are connected to the source pin of the power MOSFET. The anode of the laser diode is connected to the supply voltage. The cathode of the laser diode is connected to the drain of the power MOSFET. Figure 5.2-3 shows the basic schematic for the laser driver circuit.



**Figure 5.2-3: Laser Driver Fundamental Schematic**

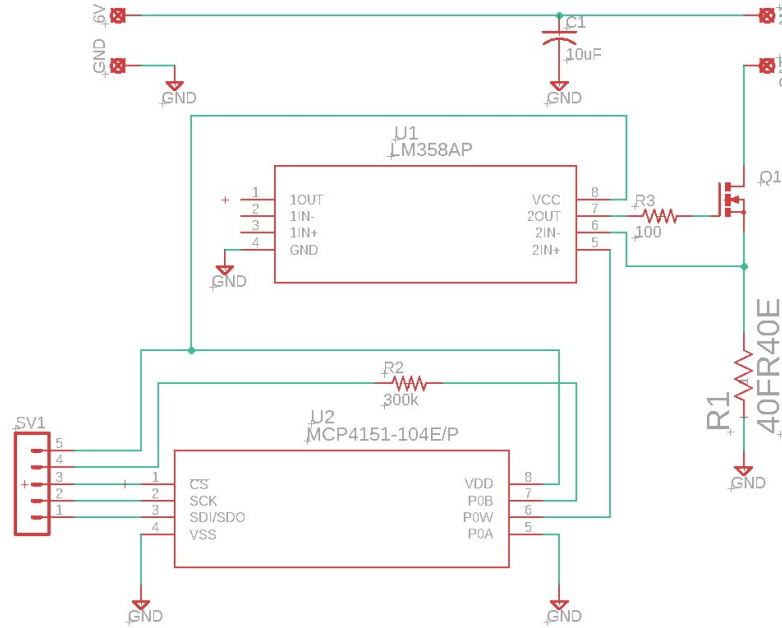
Since we plan on controlling the laser diode with the microcontroller, we will be using a logic pin that outputs 5V from the microcontroller, along with a voltage divider, as the op-amp input. Since the sense resistor will receive 3 A of current, the power dissipated from the sense resistor needs to be considered.

At first, we designed with 3.3V in mind and a single laser power. When considering a low input voltage, we investigated the optimal voltage division we can achieve with standard resistor values. Standard resistor values repeat by decade. With this fact, let's consider two resistors that have the same value, but one resistor is a decade larger than the other. If the smaller resistor value is used as R1, the ratio for this scenario will always be 1/11. Therefore, we could easily achieve 0.3V from a 3.3V input. The resistors used for the voltage divider are a 1 k $\Omega$  resistor and a 10 k $\Omega$  resistor. With an input voltage set to 0.3V, we can calculate the sense resistance needed and the power dissipated by the resistor. The sense resistor must be 0.1 ohms and the power dissipated is 0.9 W.

Using the calculated values, we researched if a 0.1-ohm sense resistor with a power rating over 0.9 W existed. Unfortunately, we could not find a sense resistor with such specifications. We did, however, find that 0.2-ohm sense resistors are more popular than 0.1-ohm resistors. By connecting two 0.2-ohm sense resistors in parallel, we can achieve an equivalent resistance of 0.1 ohms. The current across each resistor is now 1.5 A, instead of 3A, which reduces the power rating requirement for each resistor to 0.45 W. There are plenty of 0.2-ohm sense resistors with a power rating over 0.45 W.

The op-amp used in the laser driver design is a single supply op-amp. As mentioned in 3.4.2.2 Laser Driver Operational Amplifier, we will be using the LM358. The LM358 has an offset input voltage of 2mV therefore the true voltage seen by the sense resistor is approximately 0.302 V. Therefore, the current across the diode will truly be around 3.02 A. After finding all of the components and their values, we simulated the circuit using LT Spice. In order to ensure a proper simulation, the spice models for the power MOSFET, IRL7833PBF, and the op-amp, LM358, were used. The circuit simulation yielded a constant current of 3.019 A across the diode. The laser driver circuit design is functioning properly.

After careful consideration with this design, we found that it would be better to variably control the laser power instead only operating at max power. In addition, we needed to change the 3.3V reference voltage to 5V since the MCU design changed as well. The modifications made to the final design includes adding a digital potentiometer and changing the sense resistor. A digital potentiometer was used to adjust the input voltage reference using the MCU. By adjusting the input reference voltage, we can adjust the current seen by the diode. Since we want to have precise control over the laser power in our new design, we need to increase the reference voltage range. The sense resistor chosen was a 0.4-ohm resistor rated for 10 Watts. To achieve the same max current of 3 A, 1.2V is needed as the max input reference voltage. The max power dissipated by the resistor is 3.6 W which is within the 10 W rating. The new voltage reference source outputs 5V therefore, a resistor needs to be added in series with the potentiometer to drop the max voltage to 1.2V. The digital potentiometer chosen is 100k ohms therefore adding a 300k ohm resistor in series with the digital potentiometer would drop the voltage 1.25 V. The digital potentiometer is controlled via SPI with the microcontroller. The potentiometer has 256 taps; therefore, we can control the reference voltage with 0.005 V of accuracy. The circuit design was then captured in Eagle. The Eagle schematic capture is shown in Figure 5.2-4.



**Figure 5.2-4: Laser Driver Final Schematic**

Like the stepper motor driver, the BOM was generated using the Design Link tool in Eagle. The sense resistors and the voltage divider resistors need to have a smaller tolerance than the other passive components because they will determine the current. The mentioned resistors need a tolerance of 1% or less. The components used in the laser driver design are listed in Table 5.2-2. From the table, the total cost of components is around \$7.38.

**Table 5.2-2: Laser Driver Components**

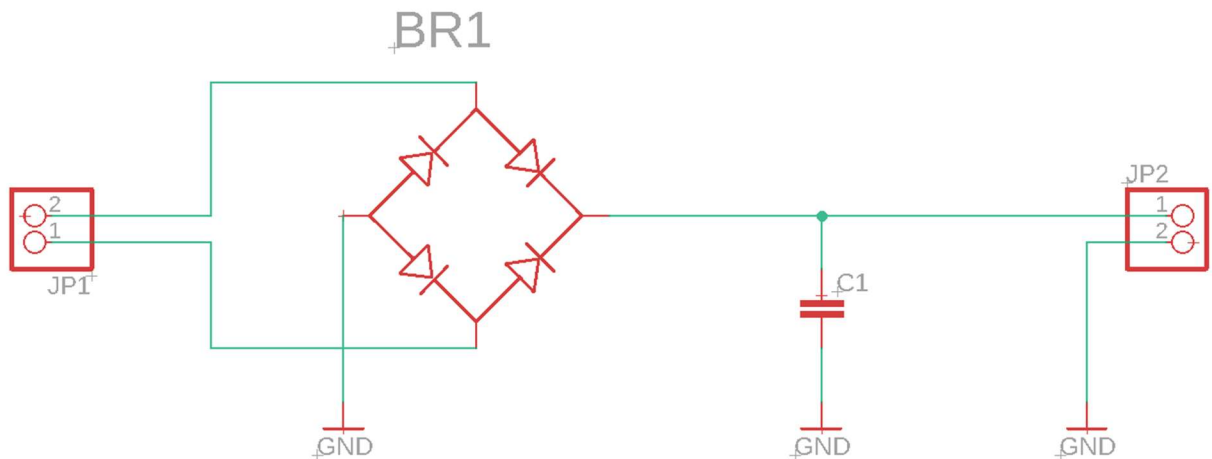
Reference Designator	Value	Tolerance	Part Number	Cost
R1	400 mΩ	± 1%	40FR40E	3.97
R2	300 kΩ	± 1%	ERJ-6ENF3003V	0.11
R3	100 Ω	± 5%	AC0805JR-07100RL	0.10
C1	10 uF	± 10%	CL21A106KOQNNNG	0.11
U1	-	-	LM358PSR	0.069
U2	100k	-	MCP4151-104E/P	1.18
Q1	-	-	IRL7833PBF	1.84

### 5.2.3 AC-DC Converter

As mentioned in section 3.3.3 AC – DC Converter component selection, we have based our design of the AC/DC converter on the transformer system. This means that our design will stay simple but then we also need to consider the encasing of the transformer. The

positioning of the transformer within our printer will be further described in future updates of this document once the enclosure has been fully defined. As of now, we can still take care of the schematic and printed circuit board (PCB) design for the rest of the circuit.

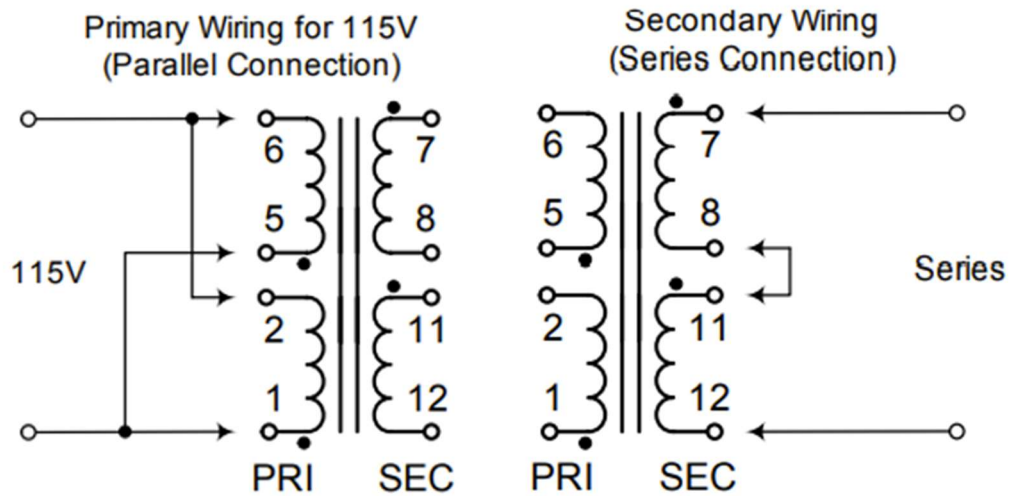
The rest of the circuit englobes the full bridge rectifier and capacitor. Therefore, our schematic will have two pairs of pin heads. One pair for the input voltage and one for the output. The input will receive the stepped down voltage from the transformer. These pins will be connected to the corresponding pins of the full bridge rectifier, and the positive and negative pins in the rectifier will be connected to the positive pins of the capacitor and ground respectively. It is important to have the capacitor connected with its negative pin to ground. This is because we are using an electrolytic capacitor which are polarized. On the second pair of pin heads, we will then connect one pin to the node that is in contact with the positive terminal of the rectifier and the positive terminal of the capacitor. This pin will then be our output unregulated DC voltage. The second pin of this pin head pair will then be connected to ground. This layout is shown in Figure 5.2-5: Eagle Schematic of AC/DC Converter. The rectifier and capacitor symbols used in this schematic are generic and not associated with the corresponding footprints of the real components used. Thus, we will need to import the footprints and match to the symbols in order to create our PCB.



**Figure 5.2-5: Eagle Schematic of AC/DC Converter**

Everything described until this point takes care of the connections after the voltage has been reduced from the 120V. We also need to configure the transformer ports correctly in order to properly step down the voltage. We mentioned before how the 185E16 transformer can be used to provide a 16V or 8V output from a 115V input depending on the configuration of its ports. This is because the transformer has two pair of windings in both the primary and secondary side. The wiring configurations are provided in the datasheet and shown in Figure 5.2-6. If the input is 115V, which is our case, we must connect the two windings of the primary side in parallel. Thus, one wire from the power cord is connected to ports 6 and 2 and the other to ports 5 and 1. As for the secondary side,

we are able to pick from either the parallel or series configuration. Due to the drop-off voltage of the 9V regulators we have chosen the 16V configuration of the transformer, so we must use the series configuration for the secondary wiring



**Figure 5.2-6: 185E16 Transformer Wiring from Datasheet**

At this point our AC/DC converter is done. Then, because we have multiple electric powered modules that operate at different voltage levels, the next stage would be to design the DC/DC converters to reduce the voltage levels. However, we would be making a mistake if we design our voltage regulators with an input of 16V in mind. This is because the output of the transformer does not have an amplitude of 16V. The real amplitude will differ due to two different reasons. First, is that the input voltage is given in RMS values and thus the 16V output is also an RMS value. To find the amplitude of the wave we need to multiply by the squared root of 2, giving us an amplitude of roughly 22.6V. The second reason why the final DC value of the converter will vary is that the voltage from the wall outlets tends to vary from 110V to 120V, then our final DC value will be between 0.956 to 1.043 of the 22.6V. Meaning that we might get an output DC anywhere from 21.6V to 23.6V.

## 5.2.4 DC-DC Converter

As discussed in the part selection section of this report, power efficiency is a key specification of the entirety of the power module. This is why it would be best if the amount of DC/DC conversion stages is minimized. The powered modules are the stepper motor drivers, laser driver, and microcontrollers. These require 9V, 6V and 5V correspondingly.

## 5.2.4.1 5V Regulator

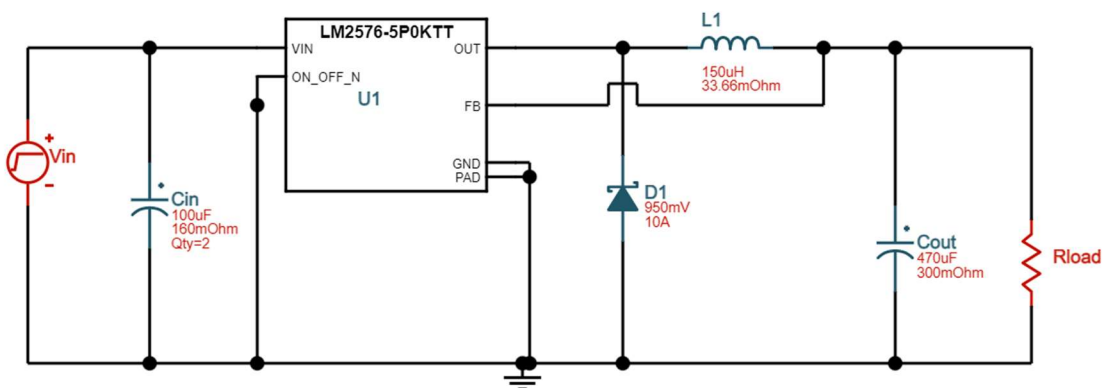
All computing processing chips be it microcontrollers or drivers will need 5V input. Thus, we will design a switching regulator with an input of  $22.6 \pm 1V$  and an output of 5V using the LM2576 regulator.

Since 5V is one of the voltages supported by the fixed configuration of the LM2576 we can follow the guidelines in section 8.2.1 of the datasheet to design this circuit. This design process includes the use of the TI WEBENCH power designer tool as well as information from the graphs provided in the datasheet.

After providing the specifications reflected in Table 5.2-3, WEBENCH gave us the design in Figure 5.2-7. According to WEBENCH, this circuit has an efficiency of roughly 84%. This is not as good as some other regulators can have efficiencies above 90%. However, as mentioned in the part selection section efficiency would not be much of an issue when we are powering the computing components given that 20% losses would account for less than a watt. Thus, we will go ahead and continue with the LM2576 for the 5V voltage regulator.

**Table 5.2-3: WEBENCH Design Input**

Specs	Value
Supply Type	D.C.
Vin Min	18V
Vin Max	26V
Vout	5V
Iout Max	1A



**Figure 5.2-7: WEBENCH Design for 5V Regulator**



With the circuit fully designed now is time to pick specific components. By looking at the datasheet we can see the specifications needed by each component apart from value. Starting with the inductor L1, the datasheet suggests using an inductor rated for the frequency of the IC (52kHz) and for 1.15 x maximum load current. The inductor used in the WEBENCH design meets these requirements with a resonance frequency of 4.3MHz and a maximum load current rating of 8.8A. However, this component is considerably expensive as it is around \$8 a piece. Another option given in the datasheet is the PE - 92108. The series of inductors to which this one belongs was designed specifically for 50kHz LM simple switcher regulator series. Unfortunately, the series of inductors have been discontinued and therefore are not widely available anymore. Thus, as of now, we will be designing around the WEBENCH proposed inductor, the Würth Elektronik 74437429203101. This inductor was later replaced by PCV-0-154-05L due to size.

The next component is the input capacitor. This capacitor ensures stable operation, and the datasheet recommends 100uF at a 25V rating for sufficient bypassing for a 5V regulator. From WEBENCH we have that for our 5V regulators the voltage rating could be as long as 16V but still maintaining 100uF. The Panasonic 16SVPC100M used for Cin in our schematic has a price of \$0.95. However, the datasheet recommends an electrolytic capacitor while this Panasonic capacitor is not electrolytic but polymer instead. Before acquiring it for our circuit we will first research the differences between an electrolytic capacitors and polymer capacitors to pick the most adequate for our design.

The other capacitor needed in our circuit is the output capacitor. This component takes care of improving our final output by reducing the ripple voltage down to an acceptable range (Usually within 1% of the final voltage). Both the datasheet and the WEBENCH tool give us a 680uF electrolytic capacitor. The Nichicon UUD0J681MNL1GS capacitor used in our design in Figure 5.2-7. has a voltage rating of 6.3V. This is consistent with the datasheet requirements that ask for the Cout voltage rating to be at least 1.5 times the regulating voltage. Since we are regulating 3.3V then this would make the capacitor have a minimum rating of 5V. However, it is better to go with even higher voltage ratings in order to reduce the equivalent series resistance of the capacitor. This way we end up with a capacitor that has a voltage rating significantly over 5V.

The last component needed in our design is the diode. There are two requirements for the diode to work correctly. First, it needs to have a current rating that is at least 1.2 times the maximum output current, thus we need a diode with a current rating greater than 3.6A. The second requirement is that its reverse voltage is greater than 1.25 times the maximum input voltage. Our design currently has 12V as the maximum input thus the reverse voltage needs to be at least 15V. The datasheet suggests the use of the SR302 Schottky diode or a 1N5823. Both have high enough reverse voltage ratings yet only 1N5823 with its 5A current rating are acceptable for our design. Unfortunately, it seems to not be available in common hardware suppliers. Therefore, we will go with the diode used in the WEBENCH produced circuit, the B540C-13-F. This diode has a voltage rating of 40V and a current rating of 5A. This diode has a price of \$1.18.

All components would give us the bill of materials in **Error! Reference source not found..** The total price is heavily affected by the inductor. Therefore, more research will be done to find an equivalent inductor that meets all required specifications for a lower price.

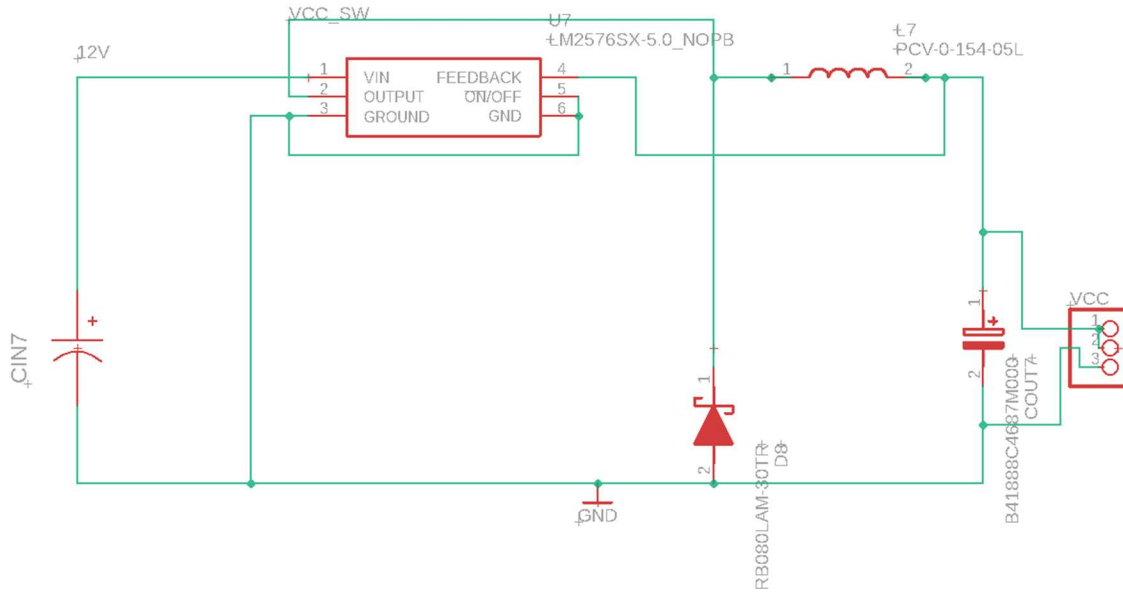
**Table 5.2-4: 5V Regulator BOM**

Reference Designator	Value	Tolerance	Part Number	Cost
U1	-	-	LM2576SX-5.0/NOPB	\$2.92
Cout	470u	±20%	B41888C4687M000	\$0.85
Cin	120uF	±20%	B41866C7127M000	\$0.71
D1	-	5A	RB080LAM-30CT-ND	\$0.43
L1	100uH	±20%	PM4344.104NLT	\$4.82
			Total	\$9.73

With all parts selected, the next step in the design process is to create the PCB layout so that a PCB can be ordered to assemble our final circuit. For this part we will be using Eagle software.

In order to avoid having to design the footprint for each component all parts selected were first searched on UltraLibrarian to make sure their footprints were available. After downloading the symbol and footprint files they were added into Eagle libraries. This makes the parts available for schematic design. On the schematic file all parts are added and wired as needed. The final schematic is seen in Figure 5.2-8.

After the schematic file is done, normally a board (.brd) file would be created from it. Then we would place the parts and wired them as needed. However, given that there are other modules that will be assembled on PCBs, this process will take place once all modules are designed. Then, we can place all circuits in a minimal number of PCBs. This process will be discussed in the PCB Vendor and Assembly section.



**Figure 5.2-8: 5V Regulator Eagle Schematic**

**Table 5.2-5: 5V Regulator BOM**

Reference Designator	Value	Tolerance	Part Number	Cost
U1	-	-	LM2576SX-5.0/NOPB	\$2.92
Cout	470u	±20%	B41888C4687M000	\$0.85
Cin	120uF	±20%	B41866C7127M000	\$0.71
D1	-	5A	RB080LAM-30CT-ND	\$0.43
L1	100uH	±20%	PCV-0-154-05L	\$4.57
			Total	\$9.48

## 5.2.4.2 9V Regulator

The 9V regulator is used to feed all the stepper motor drivers. They will draw a maximum of around 0.7A. Since we power each driver with an isolated 9V regulator, we only need each of them to be rated for about 1A to be safe. So after putting the same input values of 5V regulator into WEBENCH, we got the design in Figure 5.2-9.

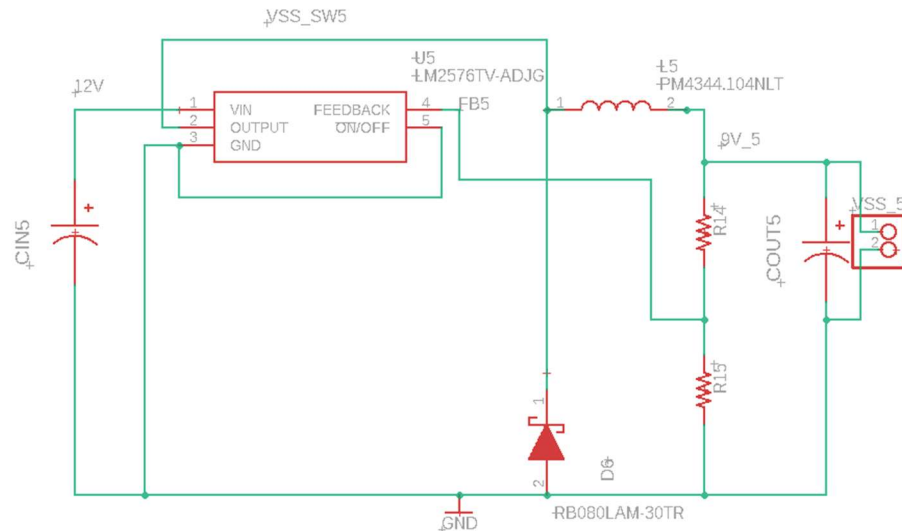


Figure 5.2-9: 9V Regulator Schematic

Table 5.2-6: 9V Regulator BOM

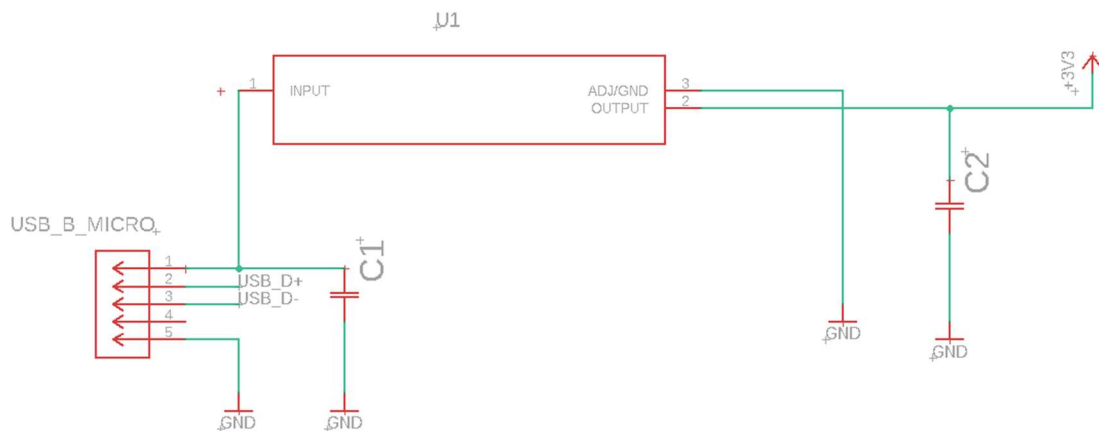
Reference Designator	Value	Tolerance	Part Number	Cost
U1	-	-	LM2576T-ADJ	\$2.83
Cout	330uF	±20%	EMVY250ADA331MHA0G	\$0.71
Cin	120uF	±20%	B41866C7127M000	\$0.71
D1	-	5A	RB080LAM-30CT-ND	\$0.43
L1	100uH	±10%	PM4344.104NLT	\$4.82
			Total	\$9.5

## 5.2.5 Microcontroller

For the start of the prototyping and testing process, we will be relying on already designed and built processor modules. Once the microcontrollers selected have been tested and proven capable of handling all processing needs, we will design a PCB with the corresponding ICs. As of now, the Raspberry Pi Pico has proven capable of being user friendly and driving the stepper motors. While there are still other tests to be ran in order to ensure proper functionality, we can almost certainly say that the IC processor in the Raspberry Pi Pico board will be the one used in our final product. This IC is the RP2040, and instead of using a Pi Pico board, we will design our own RP2040 board that meets all of our requirements. In this section we will carefully describe the board design that includes adding headers for GPIO, power supply, ground, as well as external components. All of our design will be guided by a hardware design guide provided by Raspberry Pi

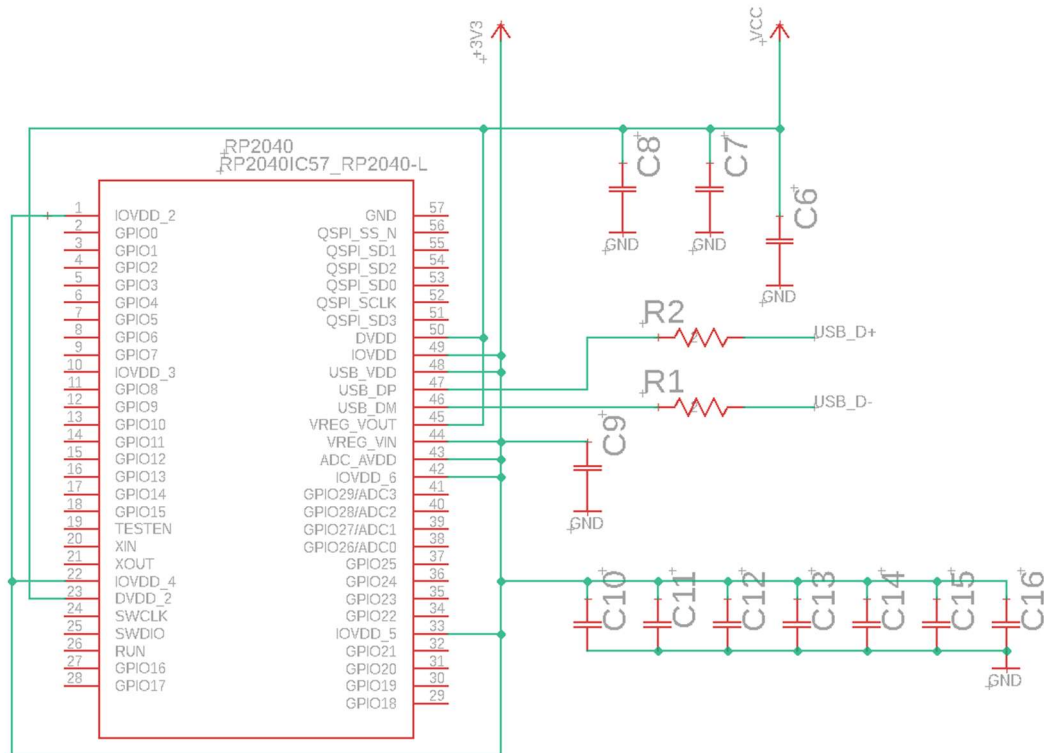
The first external component that will be added is 5 to 3.3V linear regulator. The RP2040 operates at two voltage levels, 3.3V and 1.1V. It also has an internal 1.1V regulator thus only the 5 to 3.3V is needed externally. From the previous section of the document a 3.3V regulator was already designed. However, when programming the boards, they will need to be plugged into a computer via an USB micro type B. Typically, USB ports from computers have an output voltage of 5V. which is why we still add the regulator to the microcontroller board. The other USB pins, USB\_D+ and USB\_D- will be connected to 27ohms resistors and then to the chip's USB port. These resistors are needed to meet the USB impedance requirements.

The datasheet suggests a NCP1117 linear regulator with a fixed 3.3V output. This means that minimal components need to be added to the regulator. The only components needed are input and output capacitors. As per the regulator datasheet they will be 10uF capacitors. This circuit is seen in Figure 5.2-10.



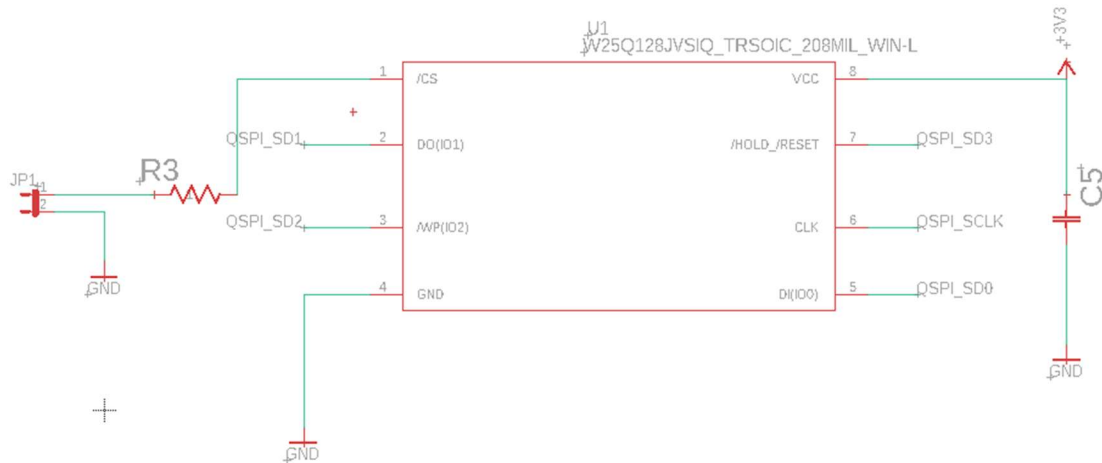
**Figure 5.2-10: 3.3V Regulator for RP2040 Power Input**

Then to finish the power need requirements of the chip, decoupling capacitors must be added to the RP2040 power pins as shown in Figure 5.2-11. They provide noise isolation as well as protection from sudden current surges. The datasheet recommends using 100nF capacitors for each power pin. Additionally, the internal 1.1V regulator needs a pair of external 1uF capacitors. They will be connected to VREG\_IN and VREG\_OUT.



**Figure 5.2-11: RP2040 with Decoupling Capacitors**

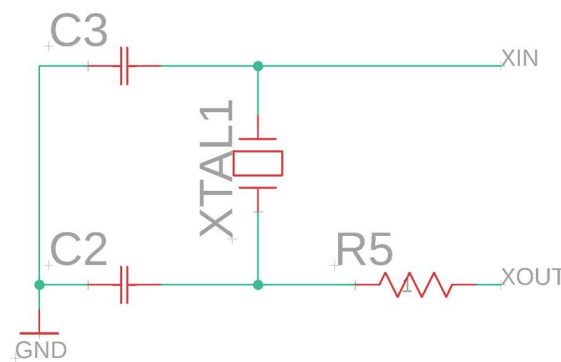
Given that we need to program the IC in order for it to perform the computations we desire, we need to be able to store the code in memory accessible to the chip. This is why a flash memory is added to our IC board design. From the RP2040 datasheet, the W25Q128JVS is suggested. This flash memory has a capacity of 16MB, which is also the largest capacity supported by our chip. Our current design shown in Figure 5.2-12, will use this same memory, however, other memories with less capacity might be sufficient for this project.



**Figure 5.2-12: Flash Memory Design for RP2040**

Another key component needed by the microprocessor, is an external crystal oscillator. This crystal is not absolutely necessary for the chip to run given that the RP2040 has an internal silicon oscillator. However, the frequency of this oscillator is not very precise, nor it can be controlled. Since our project is one that requires high degree of precision, not having a crystal oscillator could have a disastrous impact on the SLS prints.

In the datasheet, the ABL5-12.000MHZ-B4-T crystal is recommended. It is cheap and precise. The crystal needs a specific load capacitance. From its datasheet it is given that we need 18pF. Using a pair of capacitors in series with the crystal and accounting for the parasitic capacitance, we use a pair of 27pF capacitors. This means that we must place the crystal and capacitors in such a way that the parasitic capacitance is about 5pF. This will be later detailed in section 6.1 PCB Vendor and Assembly. If a different parasitic capacitance is obtained, we will have to revisit our capacitor selection. Lastly, a 1kohm resistor is added to protect the crystal from being overrun.



**Figure 5.2-13: RP2040 External Crystal Circuit**

This is where we need to consider the modules that will be using the microcontroller. We will have multiple motor drivers, laser driver, and a display module. However, the RP2040 has only 25 GPIO pins. Thus, we will need to have more than one RP2040 to control

everything and this will result in a different layout for each chip. Since, we will have multiple RP2040, we need them to communicate with each other. This will take 4 of our GPIO pins leaving every RP2040 chip we use with 21 GPIOs left to use. Given that laser, print bed, powder swiper, and x-y scanning systems need to work in perfect synchronization, it would be best that the motor drivers and laser driver are controlled by the same RP2040 chip. Independent of the number of chips used, they will each have unique external components listed until now, this includes a flash memory, micro-USB receptacle, and voltage regulator. A single crystal oscillator might be shared by all of them.

This first chip would then have a different pin distribution and different external components needed compared to the chip that will contain the display module. As explained in 5.2.1 Stepper Motor Driver Circuit, each motor driver needs 3 pins, the STEP, DIR, and nFAULT. We have 2 motors for x-y scanning, 1 motor for powder swiping, and 1 motor for each bed, print bed, and powder reservoir. Additionally, the motor drivers need 3 pins to specify the micro stepping. We will use the same micro step configuration for all motors. This will allow us to share the same 3 pins, and thus reducing the amount of pins needed. On top of this, we add a pin to each that resets the motors positioning on start up. All of those pins would add to a total of 23 pins. This is over the 21 pins we have left after using 4 for inter-chip comm. One possible solution is to move the nFAULT pins to another chip. The nFAULT only raises a flag when an error has occurred so is not directly related to the operation of the motors and thus, we can afford to have a slight delay on the flag. This would bring us to 18 motor pins, and we can now add the laser driver pin that turns the laser diode on and off in the same chip as well as the laser temperature sensor.

The other components we still need to assign to GPIO pins are the LCD screen, microSD card module, and lastly the nFAULT pins we excluded from the previous chip. Since our board will be controlling motor and laser drivers, G-code will be needed to move the motors as needed for each print. Given that the G-code is unique for every different print, we must be able to load G-code into the microcontroller easily. Thus, we will be using micro-SD to store the G-code. We have chosen to buy the SD card module that includes a memory IC and a voltage regulator. This memory IC communicates with our microcontroller via serial peripheral interface (SPI) communication protocol. This means, that apart from the voltage and ground pins, we will need to set apart 4 different GPIO pins on the microcontroller for the microSD card module. Those 4 pins will take the function of CS, MISO, MOSI, and CLK. Then we have our LCD screen module. The LCD has 17 pins that need to be connected to GPIO pins. Thus, we have already maxed out the second RP2040 chip.

On a third RP2040 IC we will add our 5 nFAULT pins from the motor drivers as well as any other functionality we might want to add in the future, as a camera module to watch our print live.



**Table 5.2-7: RP2040 Microcontroller Bill of Materials**

Reference Designator	Value	Tolerance	Part Number	Amount	Cost
RP2040	-	-	RP2040	3	\$1
U1	-	-	TLV1117-50IDCYRG3	3	\$0.61
Flash	16MB	-	W25Q128JVSQ_TR	3	\$1.95
USB_B_Micro	-	-	10118193-0001LF	3	\$0.48
XTAL1	12MHz	-	ABLS-12.000MHZ-B2-T	1	\$0.43
C1	10uF	10%	CC1210KKX7R8BB106	6	\$0.60
C2, C3	27pF	1%	C0603C270F3HACAUTO	2	\$0.22
C6,C9	1uF	10%	CC0805KKX5R7BB105	12	\$0.21
C5,C7, C10-16	0.1uF	10%	CL05A104KA5NNNC	27	\$0.10
R1, R2	27	5%	CRCW121027R0JNEA	6	\$0.22
MicroSD	-	-	AiTrip MicroSD Module	1	\$6.29
R3,R4	1k	1%	CRCW08051K00FKTC	4	\$0.10
			<b>Total</b>		\$29.82

After more research was done on the software side, an existing 3D printing firmware was found. This firmware is compatible with several microcontrollers, but it does not include the RP2040. Thus, we decided to change our chip to the ATMEGA2560. This does not only aid the software design, but it also simplifies our design of the microcontroller. The ATMEGA2560 has significantly more pins and we would have enough with using one. Thus, there is no need for multiple processors. Additionally, the ATMEGA2560 does not need an external flash memory, or 3.3V regulator. The last benefit of using this firmware and chip is that a GUI is available and can be used on a computer connected to the chip via USB. This eliminates the need for a display on our module. However, the new chip does not have an integrated USB-to-UART module, and we need to add it ourselves. For testing purposes, we used a HiLetgo USB-UART module that used a CP2102 chip. Due to shortage of chip supply, we decided to do our USB-to-UART design using the CP2102 so that we could desolder it from the already bought module. The full microcontroller module schematic is shown in Figure 5.2-15. From Table 2.3-1, we can also see that despite of the ATMEGA being 7 times the price as the RP2040, the total cost of the module ends up at less than half of the original RP2040 design.



**Table 5.2-8: Microcontroller Bill of Materials**

Reference Designator	Value	Tolerance	Part Number	Cost	Total
IC2	-	-	ATMEGA2560	7.62	7.62
IC1	-	-	CP2102	2.27	2.27
CR1	-	-	1N5818-T	.16	.16
LED1&2	-	-	155060GS75300	.4	.8
Y1	16MHz	-	CSTNE16M0V53L000R0	.27	.27
J1	-	-	USB3075-30-A	.7	.7
R1	1 kΩ	± 1%	RC1206FR-071KL	.1	.1
R2&3	10 kΩ	± 1%	RC1206FR-0710KL	.1	.2
C2,3,4,6,7	.1uF	± 10%	C1206C104M5RACTU	.1	.5
R4&5	27kΩ	± 1%	CRCW120627K0FKEAC	.1	.2
C5	1uF		C1206C105K3RACTU	.25	.25
					13.07

## 5.3 Optical Design

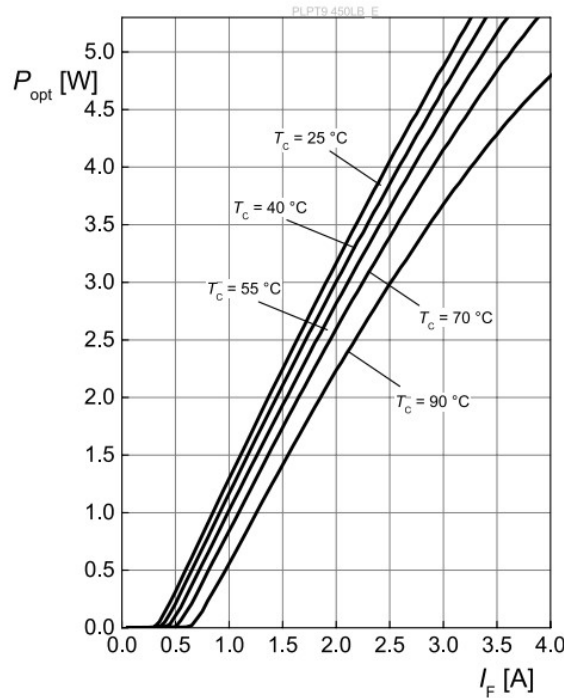
This section will detail the optical design of the SLS printer. The optical module consists of a laser diode, a collimating lens, a focusing lens, and possible beam correction methods, which are included as additional architectures.

### 5.3.1 Laser Power

The laser power is one of the most important variables for a successful SLS system. It is considered a variable process parameter that can change. However, it is coupled to two other variables in a relationship to delivered energy density, which is shown in the equation below.

$$E_A = \frac{P}{v_s H}, \quad E_V = \frac{P}{v_s H d_l}$$

This is the area-related energy density,  $E_A$ , and the volumetric energy density,  $E_V$ . Here,  $P$  is the laser power,  $v_s$  is the speed of the scanning spot,  $H$  is the hatch size, and  $d_l$  is the layer depth. The hatch size is the width of the line to be scanned. This is directly determined by the spot size of the system, which is defined to be  $175 \pm 25$  microns. The layer depth will also be determined by the powder delivery system, which should be around 150 microns. This leaves only two variables left to change to alter the delivered energy to the powder: the laser power and the scanning speed.



**Figure 5.3-1 L-I Curve of the PLPT9 450 LB-E Diode**

One of the goals of this project is to deliver a low-cost product with competitive features. Specifically, the cost sacrifice should be balanced by not sacrificing in other areas. This means the throughput of the SLS system should remain high. Because of this, the power will be set to as high as possible. This ensures that energy delivery will be optimized for maximum magnitude, and the scanning speed can, as a result, increase. Lower powers with slower scan speeds for the laser source have been shown to increase the young's modulus of SLS-printed parts. However, creating extremely durable and resilient prints is not a goal of this project, so the need for more dense parts will not be pursued.

The selected laser diode has a maximum output power of 5W, as seen by its L-I curve below in Figure 5.3-1. As long as the temperature does not increase too much, the power output should be relatively stable. To keep the diode at an output power of around 5W, the input current should stay around 3.5A. The housing will be created to ensure the module does not approach the upper operating temperature limits of the diode, which is near 90 degrees C.

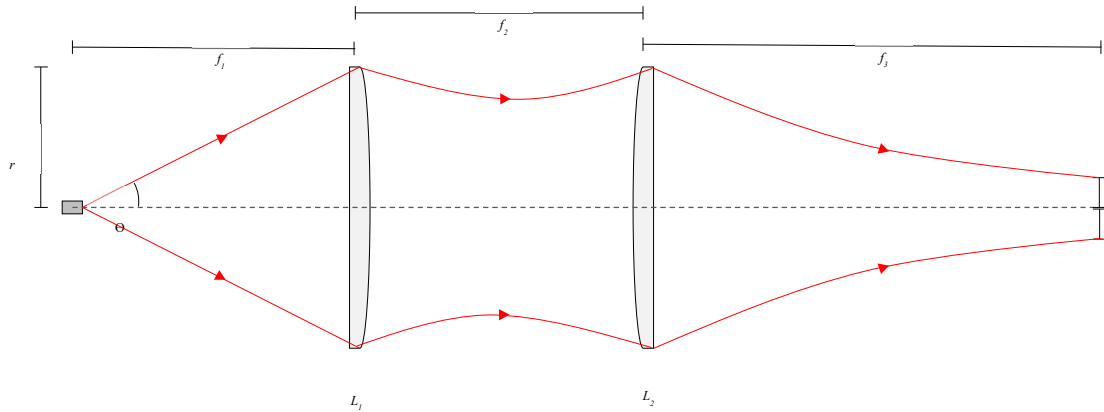
## 5.3.2 Collimating Lens

The laser diode has a high beam divergence. The slow axis diverges at 9 degrees while the fast axis diverges at 49 degrees. Because focusing ability depends on the size of the focused beam, the collimated beam should be as large as possible, while operating within the limits of the selected lenses. In other words, the aperture should be the lens itself. However, it is important to try to stay within the paraxial limit to ensure spherical aberrations are not too strong.

A single collimating lens can be used such as a plano-convex. If a half-inch lens is used, the beam radius can at most be 6 mm. This occurs when

$$f_{0.5\perp} = \frac{6 \text{ mm}}{\tan(49)} = 5.21 \text{ mm} \rightarrow r_{0.5\parallel} = \tan(9) f_{0.5\perp} = \tan(9) (5.21 \text{ mm}) = 0.825 \text{ mm}$$

Here,  $f_{0.5\perp}$  is the focal length of the fast axis. This is the limiting factor in the focal length of a single collimating lens. The slow axis radius at the collimating lens is  $r_{0.5\parallel}$ . This means the fast axis would be about 6.31 times the length of the slow axis which is extremely elliptical. The circularization is 0.16. Instead of using a half-inch lens, a one-inch lens can be used to make the beam size larger. In this case:



**Figure 5.3-2 Schematic of the Optical Layout**

$$f_{1\perp} = \frac{12.7 \text{ mm}}{\tan(49)} = 11.04 \text{ mm} \rightarrow r_{1\parallel} = \tan(9) f_{1\perp} = \tan(9) (11.04 \text{ mm}) = 1.749 \text{ mm}$$

The ellipticity and circularization stay constant. Even larger collimating lenses can be used. The ultimate focusing power is dependent on the collimated beam size, so the larger the better. There are limits to the size of the lenses due to the laser module design. Anything above 2 inches will make the laser module too large to mount on the x-y track system.

### 5.3.3 Focusing Lens

The focusing lens should be able to focus the beam to a waist of 175 microns at a distance of 75 mm. This is much higher than the smallest possible beam waist with this system, so the focal spot will actually be defocused to a sufficiently large waist. To have the best focusing abilities, an aspheric lens could be used due to the high NA. This would also eliminate any spherical aberrations in the system.

Realistically, the focusing lens will be a Thorlabs planoconvex 1-inch diameter lens with a focal length of 125 mm or 100 mm. With these, a spot size of 175 microns occurs below 2 cm away from the lens, for a working distance of 8 – 10 cm. However, the laser module design will incorporate some space from the focusing lens to the end of the module to leave room for a thin protective layer of plastic or other material, to be determined in testing.

### 5.3.4 Beam Shaping

Instead of using a single plano-convex lens for collimating, a cylindrical lens pair will be used. The cylindrical lens pair would act as a collimating lens system, as well as a beam shaping system. When using the cylindrical lens pair to collimate and beam shape, the ratio of the focal lengths of the cylindrical lens should equal the ratio of the laser beam divergence along the fast and slow axes. The ratio for the laser diode is

$$\frac{f_2}{f_1} = \frac{\theta_{fast}}{\theta_{slow}} = \frac{49}{9} = 5.44$$

The cylindrical lenses can be chosen from the part selection mentioned in chapter 3. The best cylindrical lens combination was found to be a selection of LJ1598L1 and LJ1638L1. The focal length ratio of the two lenses is:

$$\Delta_f = \frac{22.19}{3.91} = 5.675$$

The resulting beam shape parameters:

$$\begin{aligned}d_{fast} &= 2(3.91) \tan(24.5) = 3.564 \text{ mm} \\d_{slow} &= 2(22.19) \tan(4.5) = 3.493 \text{ mm}\end{aligned}$$

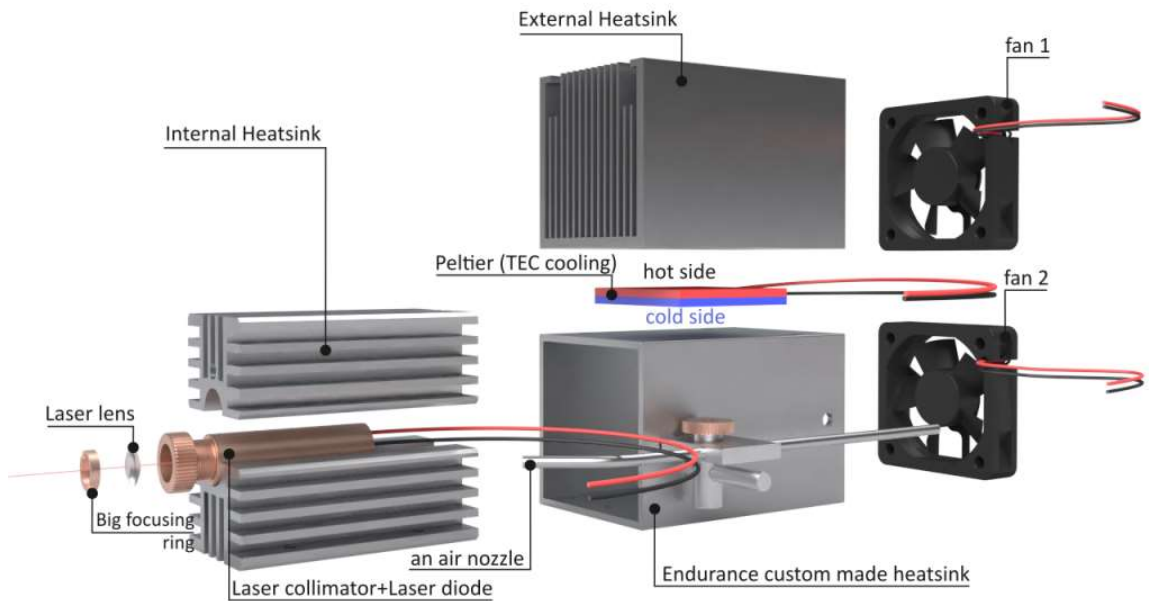
This results in a circularization of 0.98 which is excellent.

### 5.3.5 Laser Module

All the components of the optical assembly must be assembled into a laser module. The laser module should be able to handle a few functions. One function is that it should be able to dissipate heat. Heat dissipation is crucial for the success of this experiment. The duty cycle of the laser could potentially be hours. This means heat will accumulate over

long periods of time. Figure 5.3-3 shows a blown-up view of an Endurance laser module. This is a good example of proper heat control in the laser module.

Within this module design, the laser diode is housed inside a copper cylinder. The copper cylinder has excellent thermal dissipation capabilities. A thermal-electric cooler will not be used due to the inefficiency. Fans will be used, as long as their air currents do not affect the powder bed below the laser module. This will need to be tested to ensure the powder bed remains undisturbed after distribution while the laser module moves around.



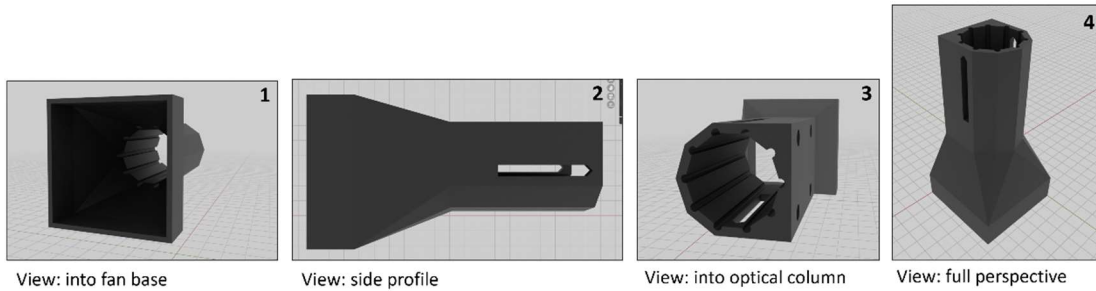
**Figure 5.3-3 Exploded View of Endurance’s Laser Module**

Within this module design, the laser diode is housed inside a copper cylinder. The copper cylinder has excellent thermal dissipation capabilities. A thermal-electric cooler will not be used due to the inefficiency. Fans will be used, as long as their air currents do not affect the powder bed below the laser module. This will need to be tested to ensure the powder bed remains undisturbed after distribution while the laser module moves around.

The laser module will also need to house the lenses. The choice of cylindrical lens pair with the focusing plano-convex lens makes the module design between the laser diode and exit aperture nontrivial. Mounts for the lenses will need to be integrated into the design of the module by 3D printing.

Figure 5.3-4 below shows the designed housing in Blender. The module was designed with a few aspects in mind. Panel 1 shows a view into the top of the module. This is where the fan sits. The fan pulls air from the surrounding environment and forces it into the module. Panel 2 shows a side profile of the module. Where the tapered section stops is where the copper module sits which holds the laser diode. Panel 3 shows the bottom of the module. Visible are the ten channels surrounding the optical column. These channels serve two purposes; one, they direct airflow down the optical column to cool the laser diode, and

two, they serve as mounting connections to hold the lenses. The slit in the side allows air to exit the module as to not disturb the powder below, as well as act as an access port to align the lenses once they are placed inside the module.

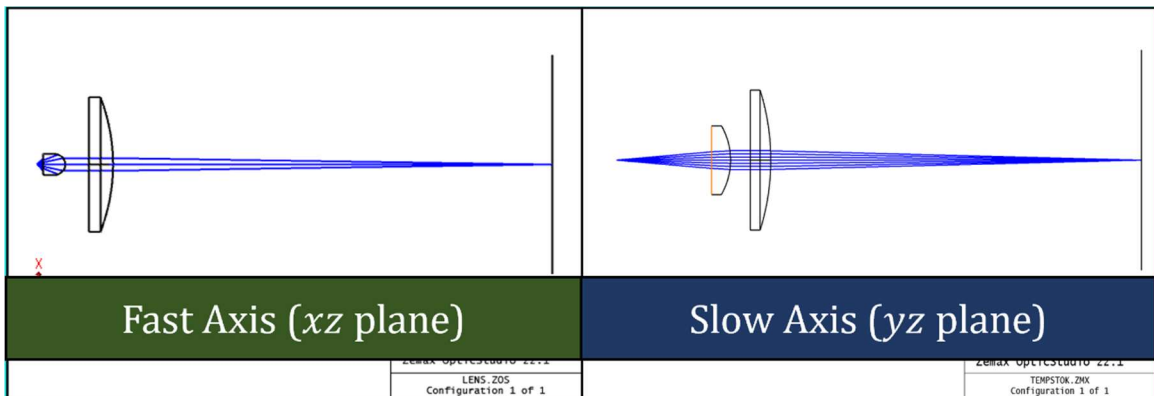


**Figure 5.3-4 Laser Module Housing**

### 5.3.6 Simulation Optimization

Currently, access to simulation software such as Zemax is being investigated. Once access is granted, optimization of the optical layout will commence. The input parameters will be the calculated parameters from the preceding sections. Variables to optimize include: the distances between lenses; the size of lenses; the distance to the powder bed; focal length of focusing lens; cylindrical lens layout to maximize circularization.

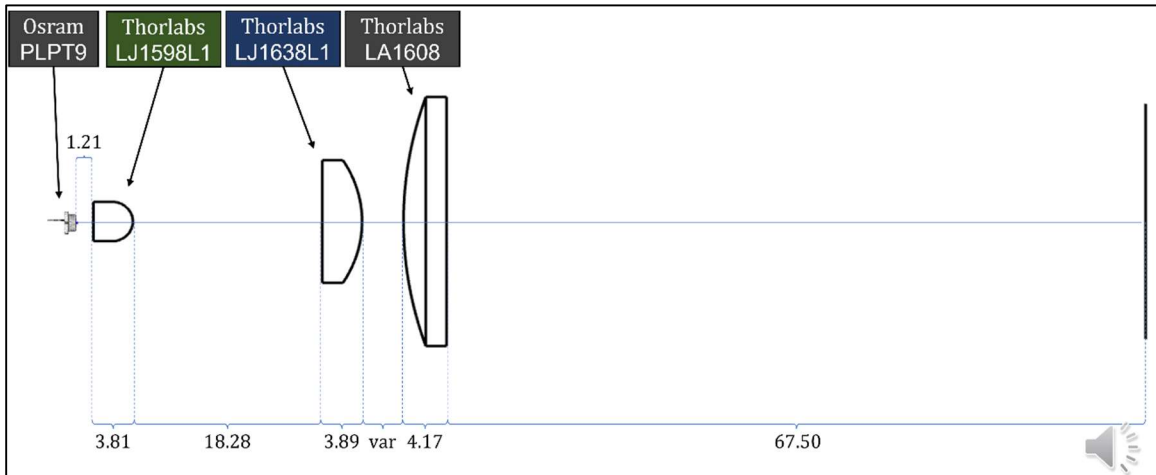
After receiving access to Zemax, the optical simulations were attempted. The results are shown below in Figure 5.3-5. The slow and fast axes had to be simulated separately, as defining an astigmatic gaussian beam in Zemax is complex and time-consuming. The fast axis ( $xz$  plane) shows the first cylindrical lens collimating the light from the diode and then focusing it at the image plane. In this picture, only a few beams were used, but more beams showed the significance of spherical aberration. The slow axis figure shows the second cylindrical lens collimating and focusing the light from the diode onto the image plane. The slow axis has much better focusing and very little aberrations present.



**Figure 5.3-5 Zemax Simulation Results**



The results of the simulation were used to optimize the distances between lenses. These optimized distances are shown in Figure 5.3-6 below. All optical elements are listed, and all distances are in units of millimeters. The distance from the diode to the fast axis correcting lens was optimized to be 1.21 mm. The distance from the diode to the slow axis correcting lens was optimized to be 23.22 mm. Finally, the focusing lens has a variable distance from the correcting lenses due to the collimated nature of the light. However, this scheme results only from a geometric optics interpretation. If physical optics were considered, the focusing lens would most likely have an optimal position. This distance was found empirically through testing.



**Figure 5.3-6 Optimized Optical Layout**

## 5.4 Software Design

### 5.4.1 Operating System

One of the basic parts of a software stack is the system it will run on. We can use an operating system like Linux which has all sorts of useful features as well as good security and support, but this will require a more powerful processor to run (with more volatile and non-volatile memory as well), and create more overhead too; additionally Linux is not a real time operating system meaning processes would not necessarily be carried out when we expect them to be, this isn't a bad thing in most cases but it might be for us.

If a Linux Operating System creates too much overhead then we can also use just the kernel, it provides lots of useful abstractions for memory allocation and process scheduling and is light enough to be used in the control systems for some battery chargers, but like the operating systems that use the Linux kernel it still is not real time.

The next option is to use a real time operating system, like the aptly named Real Time Operating System (RTOS), to provide helpful abstractions while maintaining a task

schedule defined by the program being run. Real time operating systems are commonly used in embedded systems and can be very helpful when you need to maintain some specific timing using the processor instead of interrupts or auxiliary hardware, they also provide a lot of the same abstractions as a kernel would and are even lighter on embedded systems because they are compiled with the firmware and only include the parts that the firmware makes use of.

The last option would be to write our program to run directly on the CPU with no abstractions, this will also provide a well-defined scheduling order and the least overhead of any option but requires us to spend more time on things like memory allocation and garbage collection when writing our programs. With all of this in mind, a real time operating system is probably the best choice as it will give us the abstractions, we need without adding unnecessary bloat, and if we don't end up using any of the available features then it will be like we didn't use the operating system at all.

## 5.4.2 Programming Language

Now that we know that we'll be using a real time operating system, we should probably decide how we are going to write our program inside this system. It should be easy to understand why writing in binary is out, even if we had the time, it would still be very difficult to link such a program to our operating system; Assembly on the other hand is supported as an inline compiler directive of sorts in many compiled programming languages and therefore may be an option for some parts of our program if speed is of vital importance.

Generally, when working with embedded systems C is the language of choice, it runs fast and gives the programmer a lot of control over how the program interacts with the hardware. Programs compiled from C also tend to use minimal memory (as long as the programmer doesn't do their job too terribly) because many instructions are so close to the machine language that there is almost no overhead compared to similar instructions in more modern languages (im looking at you python). The biggest downside of C is probably that in exchange for that close interaction with the hardware, the language is not very feature rich meaning many functions need to be implemented by the programmer and may not be optimal.

Another option with many of the same benefits as C is C++, C++ is a superset of the C programming language so any valid C code will also be valid C++ code, but the extra features it does include are very helpful. C++ is object-oriented meaning in addition to structs and enums, you can also create classes and namespaces, and features like inheritance become possible as well. C++ also has a more feature rich standard library that includes types like vectors which allow you to create arrays that can resize themselves, and templates which allow functions, classes or just about anything else to work with any type instead of the type they were explicitly defined for (templates are a bit of a pain to use though). Features like this make programming a lot easier if you take the time to learn them, and even if you don't then any C programs are still valid making C++ a generally better choice unless specific optimizations from a C compiler are needed.

In addition to C and C++, Rust is a modern compiled language (with an LLVM compiler meaning it will work just fine for embedded systems too) that aims to replace C/C++ by offering comparable speed and low-level control with a modern syntax to make programming an easier and more pleasant experience (after you learn the language, it has one of the steepest learning curves of any language I've ever used). Rust also provides some built-in memory safety using what it calls a borrow checker that enforces a memory ownership policy at compile time, this makes multithreading a breeze, and means that once the program compiles many errors related to memory (like segfaults) simply would not happen. With such great features Rust is a really great choice for low level code projects like an embedded system, unfortunately being such a new language means that there is not nearly as much support for when things go wrong and fewer open-source libraries to help with common tasks.

While there are hundreds of programming languages to choose from, we'll keep this list short by only looking at the few languages that are specifically good options, and one unusual language that provides some unexpected benefits. That last language is, unexpectedly, Python. Despite the language being well known for its poor memory and runtime efficiency when compared to compiled languages like C++ or Rust, Python is an easy to learn and simple to use language that has developed a large user base over the years for exactly those features. It is an interpreted language that prioritizes human readability over speed and efficiency, and because of that (and the fact that you can interface with compiled libraries in order to run programs at reasonable speeds) it is used in a variety of applications like scripting, image processing, data science, and server hosting (some operating systems even use it for parts of their user interfaces). Recently with the rise of hobbyist microcontrollers, a growing number of people have wanted to use python in their embedded systems, and so MicroPython and others like it were made. MicroPython is a stripped-down Python interpreter compiled for certain microcontrollers to allow the Python community to take part in the world of embedded systems without needing to learn a new language; being an interpreter, MicroPython still suffers many of the same issues as the standard Python interpreter but can alleviate that to some extent because its not as feature rich meaning many python programs can actually be used at a reasonable speed. With its ease of use and gentle learning curve, Python can be a great language for many new programmers who want to get into the world of embedded systems, which also makes it a great choice for our senior project (at least for some of the non-vital systems) as some group members with little coding experience have expressed interest in being involved in the programming aspects of the projects.

All that being the case, this project will likely involve the use of several languages either as libraries linked against one another or compiled to run as their own independent processes that can communicate as needed. The "backbone" of the project will be built in Rust, if possible, to help alleviate memory issues and facilitate parallelism where needed, or C++ if it is found to have better relevant support and open-source libraries. C, or inline Assembly will be used in places where precise hardware control is needed, or speed is vital (like timing systems), and Python will be used for less important programs to be written faster or by people with less experience.

### 5.4.3 Final Software Design

The code base for this project was ultimately built on top of the Repetier open source 3D printing firmware, with a number of modifications made to best suit the specific requirements of the project. The z-axis was inverted to match the mechanics of an SLS printer, and also had a custom homing sequence made to accommodate the motor and sensor placement. Additionally the function callouts, along with the appropriate g-code commands were created for the reservoir and sweeper motors to be homed and used. Finally, similar functionality was also made for the laser module which replaced a traditional FDM extruder head. This firmware was developed in mostly C++, with some traditional C features, as well as inline assembly for certain high speed functions that needed specific timing. Using open source software significantly speed up the development time for the software side of this project, and also reduced the needed testing because it was already known to work on thousands of other printers, and only needed to be verified to work properly on our system.

## 5.5 Possible Future Capabilities

The goal of the project is to create an affordable SLS 3D printer before the prototype due date. There are some capabilities that had to be eliminated due to cost and time. These capabilities do not affect the overall functionality of the printer but offers useful information for the user. The first capability would be visibility of the print area. The laser diode is dangerous to view with the unprotected eye. When testing the prototype, we will be utilizing safety glasses to observe printer operation. Once the printer prototype is finalized, the printer will be covered during operation. The cover we plan on using offers no visibility to the user. When printing, visibility is useful to ensure that the printer is functioning correctly. There are two different methods to add visibility. The first method is adding a camera inside the enclosure. The video feed would output on the user interface display. The second method is using laser safe glass to the top of the enclosure. Laser safe glass is fairly expensive and would increase the total cost of the project. The camera would not increase total cost by much but requires more time to implement it to the design.

The second capability is a larger material selection. For this project, we plan on printing with sugar and affordable thermoplastics. Other materials that could be used in SLS printing is metals, ceramics, and polymers. Laser sintering offers high tensile prints and accessibility to more materials compared to conventional 3D printing. The implementation of these materials requires bed heating and higher wattage lasers. The cost of higher end material for sintering is very high. Some powders in a 1-kilogram container costs more than the total cost of the affordable SLS printer. Printing with these higher end materials is more expensive and time consuming to implement.

## 6. Prototype Construction and Coding

### 6.1 PCB Vendor and Assembly

There are four different modules that will require a printed circuit board (PCB). Those are the power, motor driver, laser driver, and lastly the computing hardware modules. Each of those will probably have independent boards in order to simplify design and avoid all the complications of multilayered boards. The PCB will be designed using Eagle by Autodesk. This means that we will need to obtain all component's footprints and if not available then design them ourselves. As it is the case with the components used in our AC/DC converter. For that specific board, there are only two footprints that would have to be designed and both are simple enough. Thus, we should not have many complications with it. It is important to make sure the design is accurate with the dimensions of the parts, given that any errors could result in issues with our final PCB after being ordered.

When designing the PCB layout, there are other aspects to be cautious about. Placing capacitors near corresponding chip's pinouts, avoiding parasitic capacitance, reducing trace's lengths to minimize characteristic impedance. Every layout design consideration will be detailed in this section in future updates of the report. Our PCBs independent of quantity, will have to include all the circuits mentioned in 5.2 Electrical Design.

JLCPCB was the PCB manufacturer of choice for our project. The university's IEEE chapter has worked with this company to produce PCBs for workshops and thus the members that got in contact with the company could be of help when placing orders for our project's PCBs. The manufacturer offers competitive prices for both PCBs and stencils. Stencils are crucial since most components in our design are surface mount and are difficult to solder by hand. The hot air reflow stations provided in the senior design lab, along with solder paste, were heavily used in the assembly of all boards. All in all, assembly and testing for all PCBs was successful. Figure 6.1-1 shows the final prototype PCBs mounted in their final configuration.



**Figure 6.1-1: Final PCBs Mounted**

## 6.1.1 Laser Driver PCB

The laser driver did not undergo a revision from the initial design. Most part were through hole; therefore, soldering was not a big issue. The laser driver PCB was actually used as a first test to see how the hot air reflow station and the solder paste worked. The design has three 0805 passive components. The stencil was placed on top of the PCB and solder paste was applied precisely on the pads. Once verifying our capability of using the reflow station, we applied the same knowledge to the more complex and complicated PCBs. The PCB layout for the final laser driver design is shown in Figure 6.1-2. The final assembled laser driver PCB is shown in Figure 6.1-3. The final laser driver PCB was verified following the test procedure in the testing section. The test procedure and results for the laser driver PCB are stated in 7.1.3 Laser Driver Testing. The laser driver PCB worked as intended with the driver being to adjust and hold constant current.

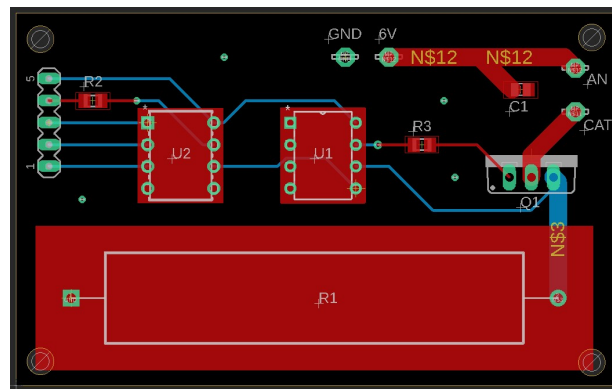


Figure 6.1-2: Laser Driver Module PCB Layout

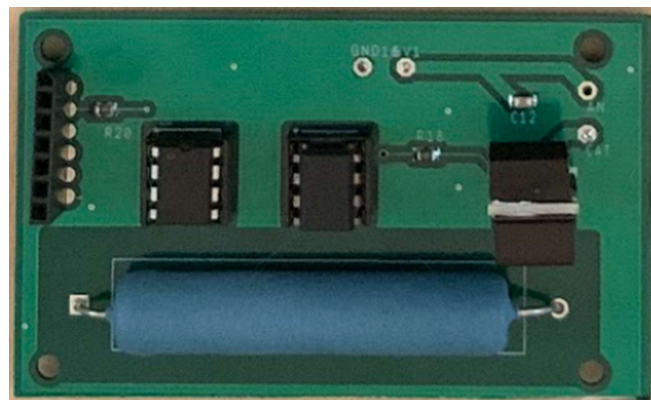


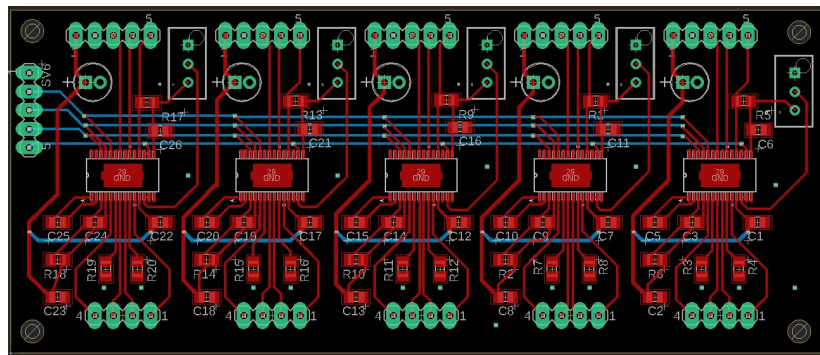
Figure 6.1-3: Final Laser Driver Module PCB

## 6.1.2 Stepper Motor Driver PCB

The stepper motor driver PCB underwent a revision. The first design had three design flaws. The first design flaw was incorrect usage of the chip enable pins. The chip enable pins for the DRV8825 are nSLEEP and nRESET. The initial design tied these two pins to the on-chip 3.3V regulator. Upon initial power-up, the stepper motor driver chips did not turn on. The reason was that the 3.3V regulator only turns on when the chip enable pins are logic high. To fix this for testing purposes, we severed the connection between the regulator and the enable pins and soldered an external wire to the enable pins. On the second revision of the PCB, an extra header pin was attached in order to externally turn on the enable pins.

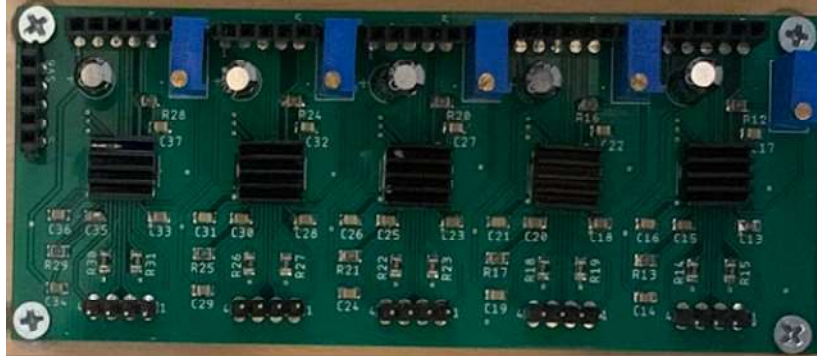
The second design flaw was that the reference voltage used to set the full-scale current was shared across all chips. The problem was that the 3.3V regulator on one chip must work properly in order to properly set the reference voltage for all chips. For troubleshooting purposes, we found that isolating the reference voltages would be best for the design. Furthermore, the initial voltage reference was created using a set voltage divider. We found that a variable voltage divider would work better in order to precisely set the reference voltage after manufacturing. On the second revision, we added trim potentiometers for every driver, so each driver had their own variable voltage reference.

The final design flaw was not having a way to prevent high in-rush current. The DRV8825 has built in circuit protection that shuts the chip down if in-rush current is detected. Therefore, the chip would only turn on if a 10 – 100 uF electrolytic capacitor was attached in parallel with the supply voltage. Therefore, a 10 uF electrolytic capacitor was added to the second revision. The PCB layout for the final stepper motor driver revision is shown in Figure 6.1-4. The final assembled stepper motor driver PCB is shown in Figure 6.1-5. The final stepper motor driver PCB was verified following the same testing done with the stepper motor driver modules. The test procedure is stated in 7.1.1 Stepper Motor Testing. All stepper motor drivers on the PCB were tested and they all passed PCB verification.



**Figure 6.1-4: Stepper Motor Driver Module PCB Layout**

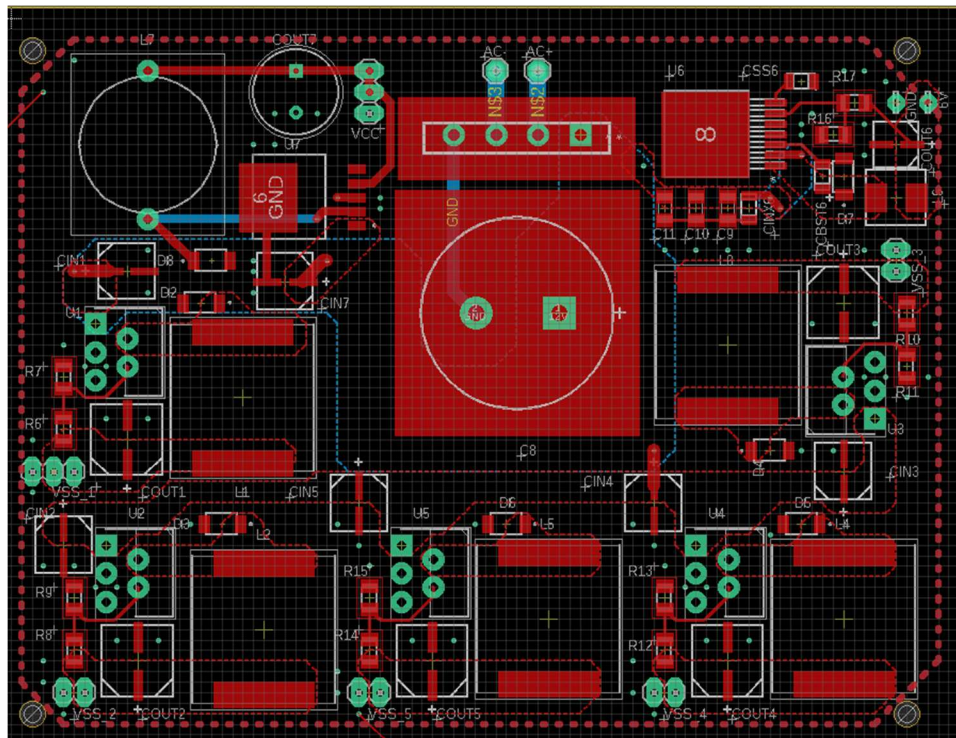




**Figure 6.1-5: Final Stepper Motor Driver Module PCB**

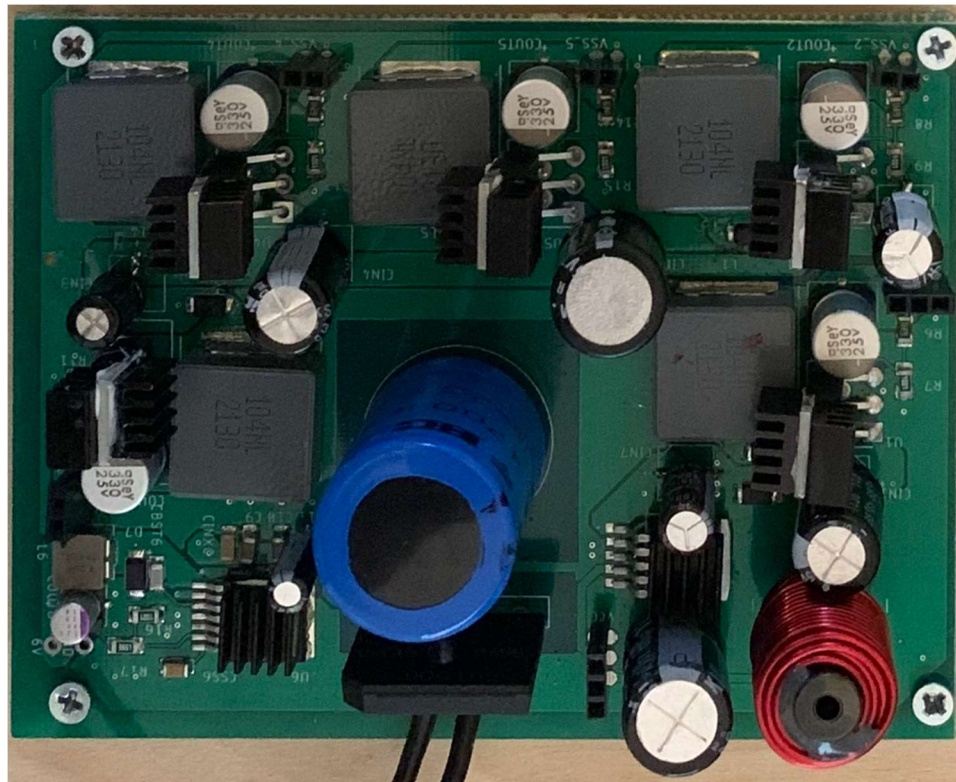
### 6.1.3 Power Module PCB

The power module PCB takes care of providing all the power required by the printer. Its proper functioning is essential to the normal operation of the system. Thus, we need to make sure it can handle high levels of current during long durations of time. To ensure this, we designed the traces to be wider than the minimum requirement to maintain temperature rise at or below 25°C. In most cases, instead of going with this minimum trace width we used polygons. This is the common practice for power PCBs.



**Figure 6.1-6: Power Supply Module PCB Layout**

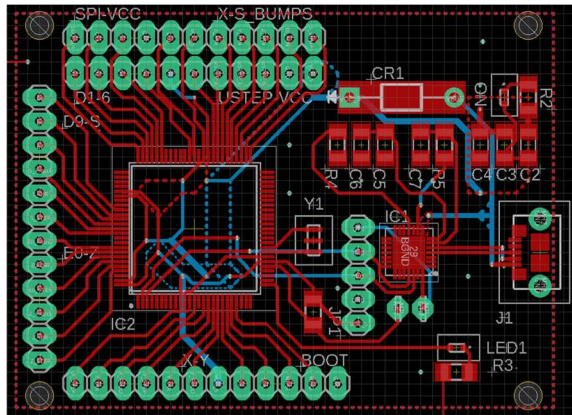




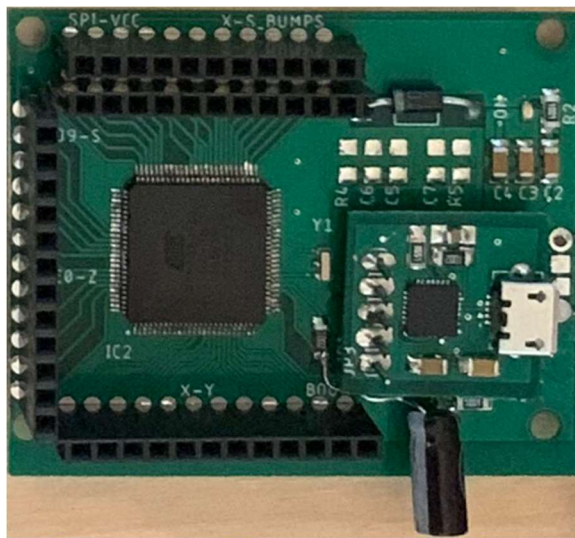
**Figure 6.1-7: Final Power Supply Module PCB**

## 6.1.4 MCU Module PCB

Designing the MCU PCB was rather straightforward. It was mostly about routing I/O pins to the pin headers and adding a few passive components. However, there was a critical design consideration that had to be accounted for. This is the USB-to-UART module. USB data are transmitted through the D+ and D- lines. There is a requirement for these two to be a differential pair with a  $90\Omega$  impedance. The easiest way to accomplish it is by placing the bridge chip, in our case the CP2102, as close as possible to the USB port. This was not done in our first revision thus, we design a separate USB-to-UART module that we could connect to the board in the same way we did for testing with a HiLetgo module.



**Figure 6.1-8: Microcontroller Module PCB Layout**



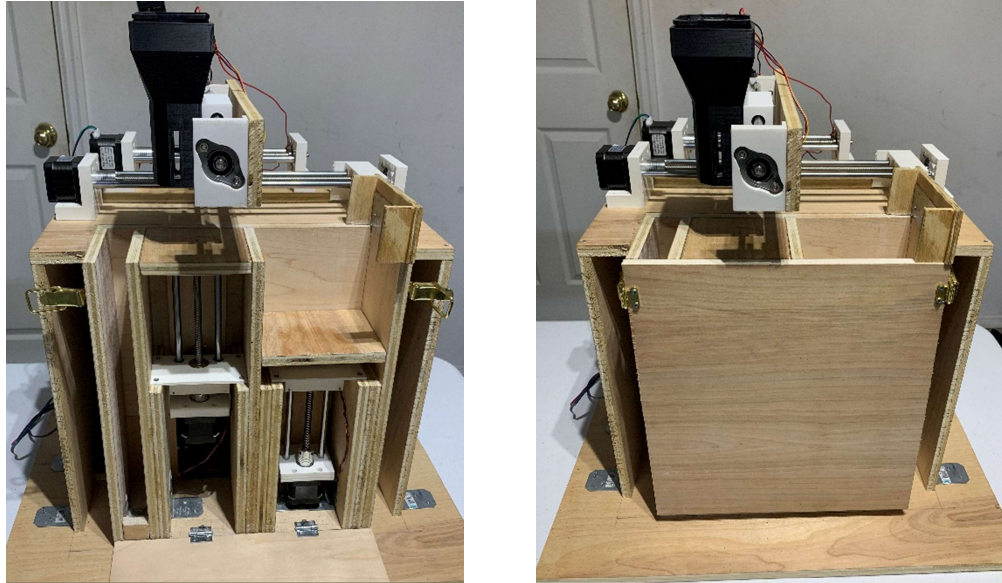
**Figure 6.1-9: Final Microcontroller Module PCB**

## 6.2 Mechanical System Construction

The mechanical structure is primarily made out of wood. Initially, the structure was supposed to be constructed  $\frac{1}{4}$  inch plywood and the pieces would be laser cut. We found that  $\frac{1}{4}$  inch plywood would be too weak to use as a supporting structure therefore we opted to using  $\frac{1}{2}$  inch plywood. The problem with  $\frac{1}{2}$  inch plywood is that it is too thick to cut using the laser cutter provided in the Innovation lab. Although it would have been easier to get precise cuts with a laser cutter, other methods had to be used to construct the printer. A Miter saw and jig saw, along with the free cuts provided with the purchase of plywood at Home Depot, we were able to cut all wooden panels. The assembly followed the design mentioned in 5.1 Mechanical Design.

Some additions were made to the final assembly of the mechanical design. The side panel uses two latches and a hinge system. During the print, the printer must contain all of the powder without it spilling out. The latches provided a tight seal between the panel and the

rest of the structure. Additionally, the sweeper mechanism has two side barriers. From early on powder testing, we noticed that the powder would move outward as the arm moved from one end to the other. The side barriers helped contain the powder as it moved across the printer. Figure 6.2-1 shows the final construction of the mechanical system. In the figure, notice the latches for the side panel and how they play a role in securing the side panel firmly.



**Figure 6.2-1: (Left) Final Mechanical Construction with Side Panel Open. (Right) Final Mechanical Construction with Side Panel Open.**

## 6.3 Final Coding Plan

### 6.3.1 General Code Structure

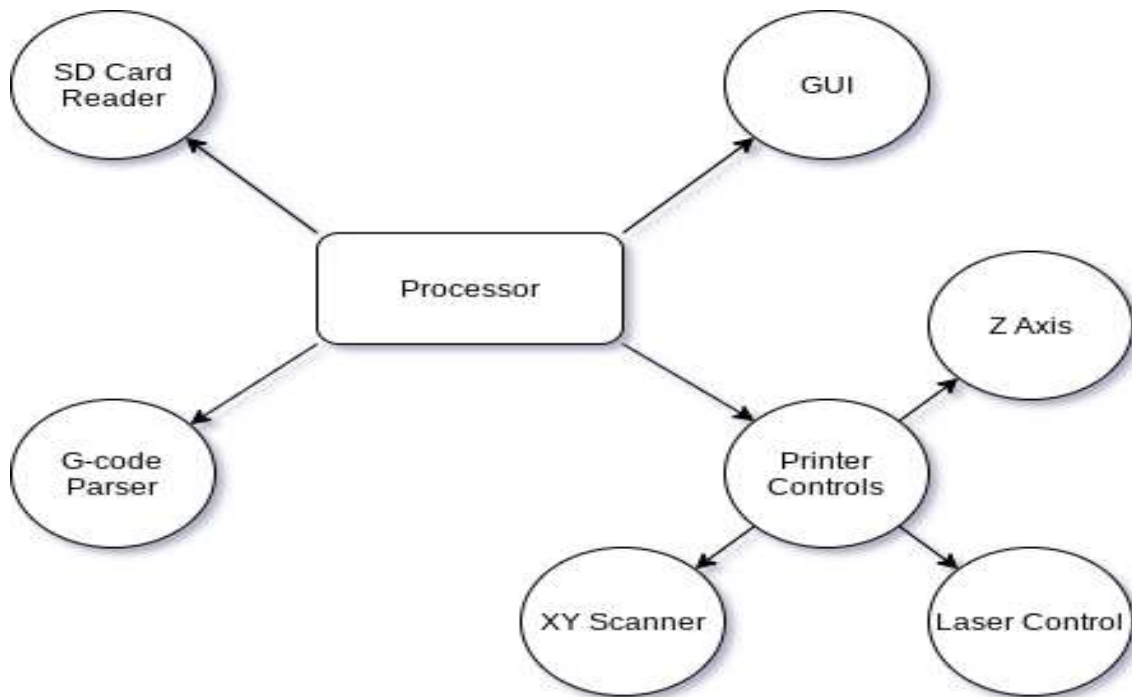
The overarching goal of the software design is to create a complete set of software needed to run an SLS 3D printer. Depending on how we define an SLS 3d printer, and the scope of software needed to run one, this design can be quite simple; it would only require enough to drive the motors to create a premade and hardcoded design for something simple like a cube. The goal for this project, however, is something more in line with modern consumer products which would naturally require a much more advanced feature set.

In order for our software to meet the needs of our goals, it needs to fulfill a few conditions as well as meeting some constraints. To start, it needs to be able to interact with the printer's hardware like the x, y, and z axes as well as the powder distribution system and the laser. The software also needs to be able to accept some arbitrary specification of what it will be printing; there are different formats available, but in this case G-code will be used as it is the current standard in the 3d printing community. The software must also be able to read

the G-code file from somewhere (it is not going to be hardcoded) like an SD card or a USB flash drive. We will be using an SD card for similar reasons to G-code, that is to say that it is something of an unofficial standard in the 3d printing community so using it will make our design more easily acceptable. A USB flash drive, however, would have the advantage of also accepting SD cards via an adapter, but that would likely require significant extra effort to support both formats. Beyond the G-code, the software should be able to accept further instruction from the user like laser power or scanning speed to set as values for the print, as well as choosing the desired file to print and starting/pausing/stopping the print. There are several ways we can do this like buttons, turn dials, a touch screen, or something else entirely like voice input or gesture control. We will be going with a touch screen in order to give the final product a more modern feel as well as simplifying the needed components and the code needed to run them. This last point may sound counterintuitive, but push buttons require either computationally expensive loops to detect consistently, or interrupts with signal debouncing which is a difficult task to implement well. A touch screen display, however, with its own display driver communicating with the microprocessor over I2C or SPI will handle all that itself and simply send signals indicating user inputs.

In order to implement our software to the previously defined standards there are many structures it can follow, but it should all turn out similarly as long as good programming practices like abstraction are applied. To meet the needs of a 3D printer to draw cross sections of a model layer by layer, we will construct a PlanerScan class that will allow us to control the x and y axes to draw any basic geometric component we may need with minimal code, and high reusability. There will also be a separate ZSystem responsible for moving the print bed and powder reservoir bed, as well as the powder distribution sweeper, and controlling the height of each layer. For the last of our printer's physical needs, we will have a LaserManager class to control anything we may need the laser to do like powering on and off, as well as adjusting the output power if we decide to implement such a feature later on. In addition to the components that interact with and affect the printer's physical parts, we also need some less visible software to do things like parsing g-code; we will be converting g-code into an intermediate representation of type PrintModel which will contain the printer's instructions in a format ready to be passed to hardware abstractions like PlanerScan and LaserManager. In order to get that g-code though, we will need to read it from the SD card, this can be done in its own class but that's probably unnecessary as libraries already exist for doing exactly that; we will start with one of those libraries and create a wrapper class if it becomes needed to manage code complexity. The last thing we need for the printer is a user interface, we will be making a graphical user interface to go along with the touch controls that will allow the user to give the printer certain commands as well as viewing some information like elapsed print time and expected remaining time. A block diagram of the general code structure is shown in Figure 6.3-1.





**Figure 6.3-1 General Code Structure**

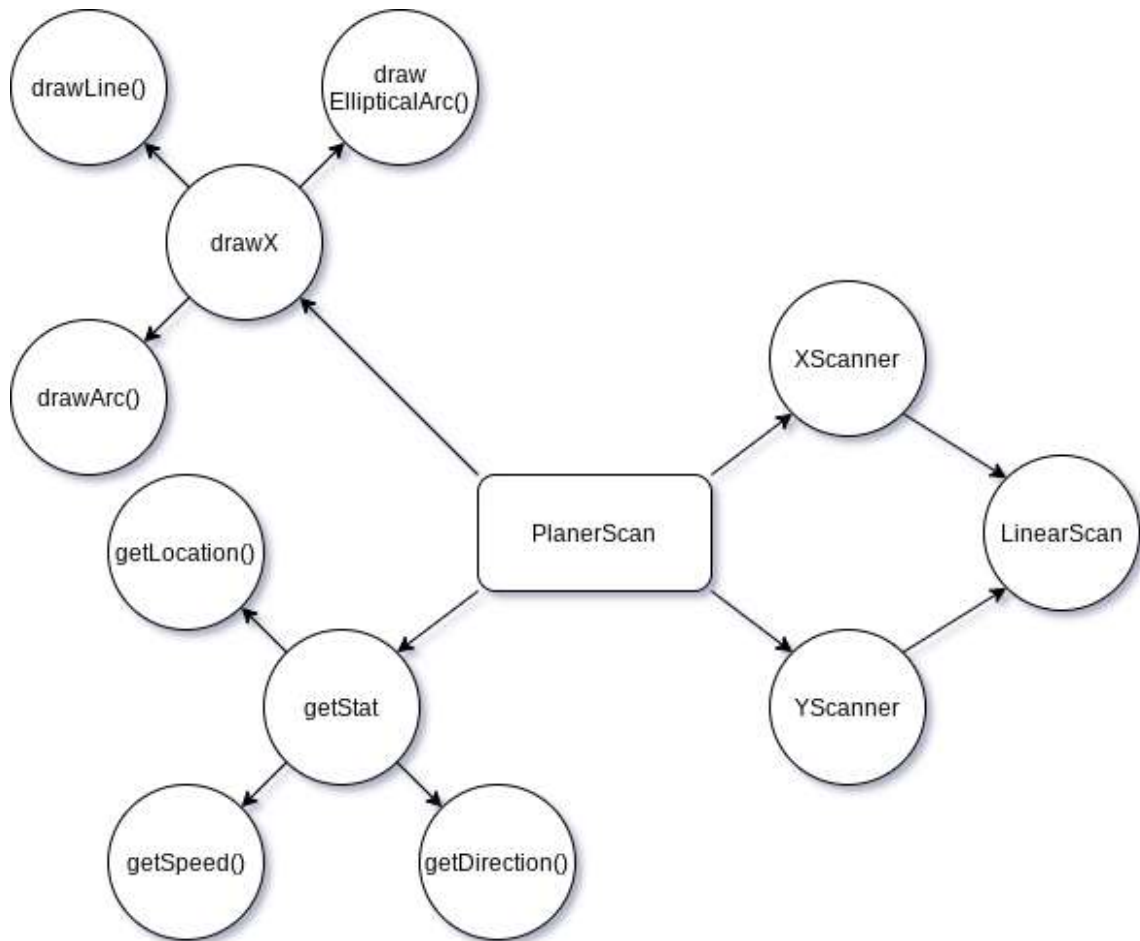
## 6.3.2 PlanerScan Coding Structure

The PlanerScan class is an abstraction between the higher-level print management functions and the hardware, as such there are a few goals it must meet in order to fulfill its role and intended purpose. First, is that the class should expose a simple and highly reusable API for controlling the xy-scanning system, it should allow for any and all needed movements on those axes without the need for supporting calculations on how those movements should be made to work. Additionally, the class should handle the timing needs of its movements to ensure that the print head (laser) is always moving at the correct speed without further monitoring or processing in the calling function. Finally, the scanning must be synced with the printer's other functions like the laser power, and the print bed movement; the PlanerScan class should provide a way to access any needed information for this synchronization or a method to synchronize automatically.

Along with the above goals, the PlanerScan class must follow some constraints to meet the software's needs and be considered well designed. First and foremost, the entire API must be exposed through a single class, interface, or other data structure; this is important because this program will not only be multithreaded, but also networked, so separating related data will create artificial traffic that will slow down the entire program including other subsystems. Additionally, all of the calculations needed to move the laser correctly should be handled internally including both speed and direction as well as acceleration (movement and timing). The API will only take geometric definitions of the path to follow in order to draw shapes (as well as how long the path segment should take to trace) so it should be able to define movements internally for basic shapes like line segments, arcs,

and elliptical arcs. In order to facilitate synchronization with other subsystems, the PlanerScan class will also expose a function or variable for live updates of its progress on a given task.

To best implement the above standards, the PlanerScan class will use the following structure. First, the API will rely on separate XScan and YScan classes to manage the X-axis and Y-axis movements separately. This will allow for simpler and more readable code defining different planer movements, as well as easier debugging for any issues that may come up with the movement of a given axis. The two classes will, however, still use a unified Scan class to control one dimensional movement. Regardless of how they are implemented however, the API will also need to expose functions for drawing including drawLine(), drawArc(), and drawEllipticalArc() which will accept parameters defining their respective shapes in a cartesian plane and use them to generate those movements and add them to a queue for use after syncing. Timing parameters can also be included to override the default movement speed and calculate a new one that will allow the given time at the given speed, these parameters will however be optional with null/nan defaults. When a new movement is added to the queue, it would not be carried out immediately when reached, instead the subroutine will halt and wait for a continue command when the other subroutines have caught up and been synchronized; this can be handled in a higher scope by a calling function, or internally via an event handler system. The class should also maintain and update a flag indicating the scanner's state; the state flag will indicate either `_inuse_`, `_preparing_`, or `_ready_` which will respectively indicate if the scanner is currently drawing, moving to position to draw, or ready to start drawing in order to help other classes know when an appropriate time is to do things like moving the print bed or queuing up a new shape to draw. Additionally, the class will provide several functions to read important information about the scanning as it is happening. These should include functions like `getX() -> float`, `getY() -> float`, `getXSpeed() -> float`, `getYSpeed() -> float`, `getSpeed() -> float`, and `getDirection() -> float` which will provide up to date information about the scanner with speeds using positive/negative to indicate direction, and direction returning the current movement angle in degrees above the X-axis. In addition to setting the desired speed on a movement-by-movement basis, the API will expose a `setSpeed()` function which will allow the default speed to be set; this speed will be calculated as the speed of the print head tangent to its direction. Along with speed however, the PlanerScan class will also manage acceleration and deceleration in order to reduce jitter and sync the laser speed with its power. Finally, the `Start()` function will set the scanner drawing a given shape in a new thread, and immediately return to the previous scope in order to process the next movement. A block diagram of the PlanerScan structure is shown in Figure 6.3-2.



**Figure 6.3-2 PlanerScan**

### 6.3.3 ZSystem Coding Structure

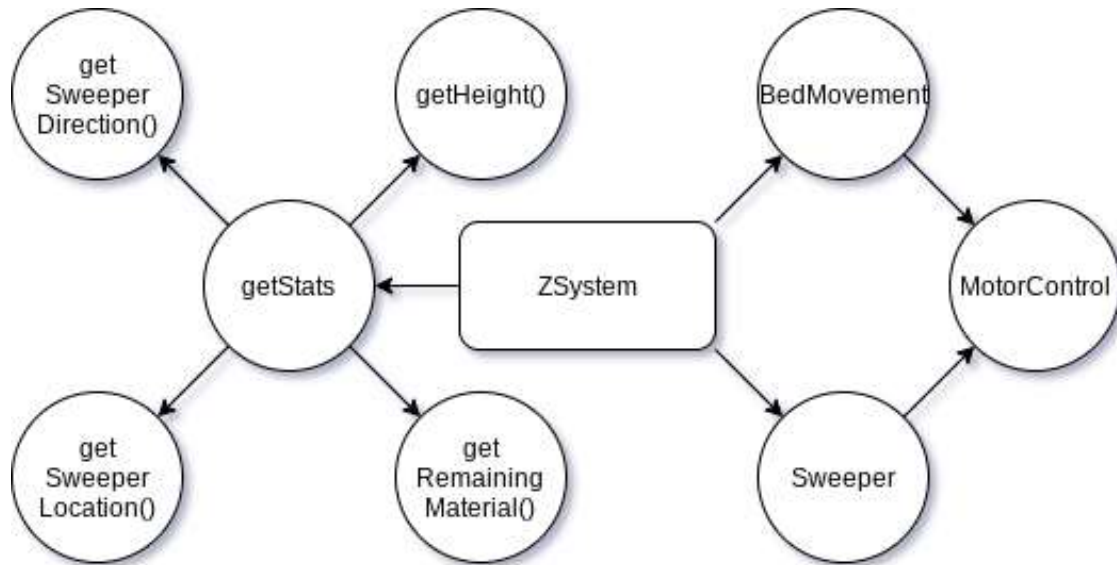
The ZSystem, like the PlanerScan class, will be a physical control class used to abstract away hardware instructions and calculations for simpler programming. The goal of the ZSystem is to create a reusable set of code to control any movements related to the z axis. It should allow for any arbitrary movements needed for a given print, and also handle timing and synchronization related to the z axis. Like other abstraction layers, the ZSystem will handle all lower-level calculations internally, requiring only the parameters of the movements to accomplish as arguments, and accordingly moving the build plate, the reservoir plate, and the powder delivery sweeper.

Along with meeting the above goals, the ZSystem must also adhere to some constraints. First of all, like any good abstraction in our system, this one will expose all of its needed functionality through a single class. Additionally, lower-level functions should never need to be carried out above this layer in order for it to function; with just a layer height, and a start new layer command, this class should handle all underlying processing needed to start the new layer. Also, any acceleration or deceleration determined to be needed (with respect

to the Z-axis) will be handled by this class as well and would not require any extra programming. Finally, the API must expose all information that may be needed for proper synchronization with other classes controlling other systems.

To adhere to the above goals and constraints as best as possible, the ZSystem class will be implemented as follows. First of all, the API will make use of a unified BedMovement class to control each Z-axis plate, the build plate, and the powder reservoir plate. This will be done because unlike the x and y axes, the two build plates need to move in the same fashion at the same time, so if a bug appears in one or the other, they can be caught and fixed faster without making the code more complicated. The ZSystem class will also use a Sweeper class to control the powder distribution sweeper, both the Sweeper class, and the BedMovement class will depend on the same MotorControl class to interface with the io pins and control the motors. The API will also expose functions to adjust the print height and distribute the needed powder material for each layer. When the print height is changed to the next layer height, the class should automatically move both the print bed as well as the powder reservoir bed to their respective new heights before sweeping the powder from the reservoir into the print area. As is common in 3d printers across the industry, the layer height will be adjustable here too, and accordingly, the ZSystem will provide a function to set the desired layer height for the print. Although the option would not be given due to reliability concerns, the code will also have the function to adjust the desired layer height during the print to combine coarse prints with fast times and fine prints with high detail in a hybrid approach. The API can take one of two approaches to use states, either locking by not returning the active scope to the calling function until the print movements are finished or updating a flag with the Z-axis state which can be in either `_heightadj_`, `_powderdelivery_`, or `_ready_` to indicate that the layer is changing, the powder is being distributed, or that the layer is ready for printing. Along with this flag, the API will also expose stats functions including `getPrintHeight() -> float`, `getReserviorHeight() -> float`, `getRemainingMaterial() -> float`, `getSweeperState() -> bool`, `getSweeperLocation() -> float`, and `getSweeperDirection() -> bool`. These functions can be used to help determine when to start printing a layer, as well as displaying stats about the current print. A block diagram of the ZSystem structure is shown in Figure 6.3-3.





**Figure 6.3-3 ZSystem**

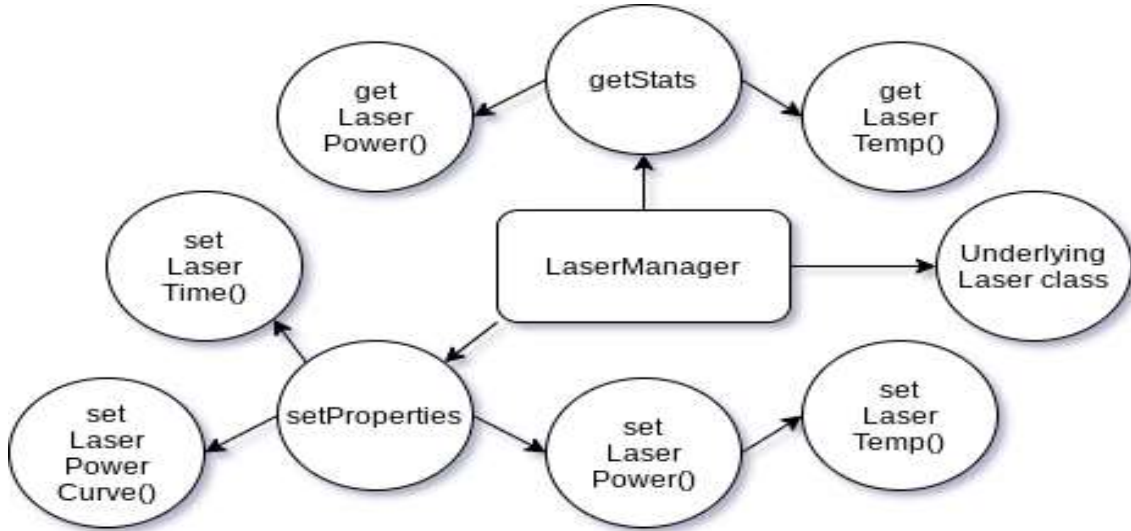
### 6.3.4 LaserManager Coding Structure

The last of the physical control systems, the LaserManager class will handle all things relating to the laser module. It should expose a simple API that can easily be reused throughout the codebase to manage the laser's power state, output power, and timing. The class will also monitor the laser temperature and control any active cooling mechanisms as needed to prevent overheating. Like other classes that deal with physical printing systems, the LaserManager will have a mechanism for syncing with other classes.

To meet the requirements of our design, the LaserManager should adhere to the following constraints, first the API should be exposed through a single class so as to simplify the interface. It should also include any calculations or processes needed to run the laser as internal components, and not require any special processing from its calling function except to specify the parameters of how the laser should be managed. The exception to this would of course be relying on lower level abstractions internally. Also, being a physical control system, the LaserManager will need to expose some sort of functionality for syncing with other classes.

To best adhere to the above goals and constraints, as well as following a consistent structure with the rest of the code base, the LaserManager will be designed with the following in mind. First, a lower level laser class can be used to interface with the laser directly and abstract away some of the functions for cleaner and more readable code, as well as easier testing and bug fixing. Additionally, the API will expose several functions to control the laser including startLaser(), stopLaser(), setLaserTemp(float c), and setLaserTime(int ms). Some other functions that may be included in future updates are setLaserPower(float power), and setLaserPowerCurve(float (\*curve)(int)), the latter of which accepts as input a function pointer which in turn accepts an integer for time and maps it to a corresponding float value for power at that time. The API will also include a flag to show the laser state

which can be set to `_fullpower_`, `_intermediate_`, or `_off_`. To help facilitate synchronization at higher layers, the `LaserManager` class will provide monitoring functions like `getLaserPower()` -> float, and `getLaserTemp()` -> float. A block diagram of the `LaserManager` structure is shown in Figure 6.3-4.



**Figure 6.3-4 LaserManager**

### 6.3.5 PrintModel Coding Structure

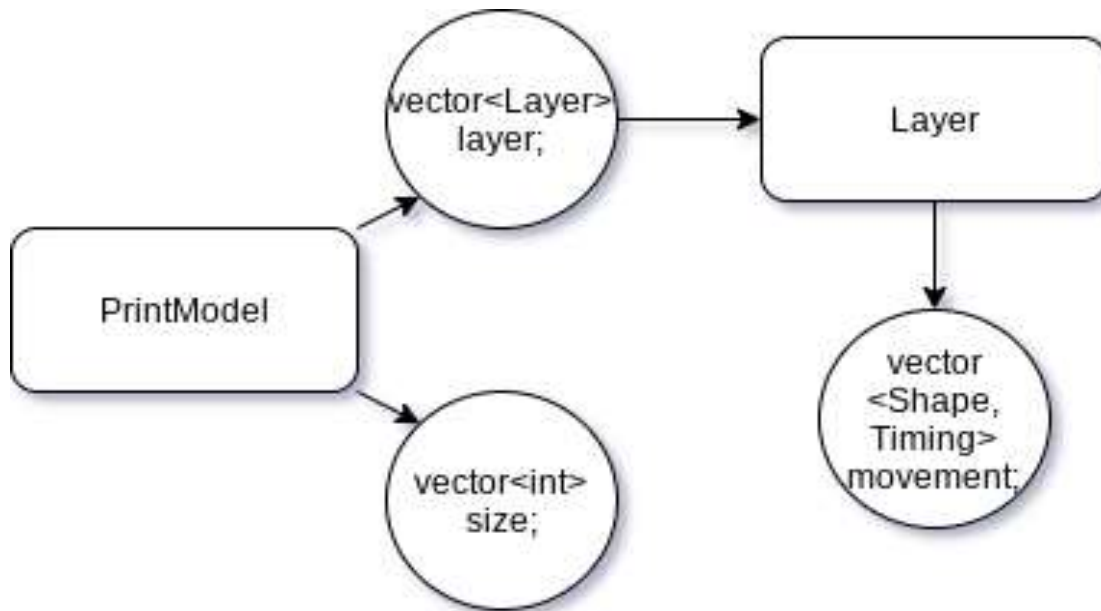
To render the G-code into a physical object, we can potentially use it directly as commands for our various processes, but an intermediate representation would simplify the process, separating the computational methods from the input and allowing the two to be tested and troubleshot separately. We will be making a `PrintModel` object to function as our intermediate representation with a few goals in mind, first that it contains the data in a format easily usable by the later functions in the code that will be using it. It should also contain all of the information for the print, including but not limited to the laser power, xy movements, and how the two are timed with one another. Lastly, we want to ensure that ours meets the standard we are reading from, and is transformed symmetrically, so the final goal will be that it is reversible back to the g-code it started from or something equivalent.

Like the various API's defined so far, the `PrintModel` data structure will be a single structure containing all the data needed for the print. While it can contain other structures, our code must be able to function with only the `PrintModel` defining the information for the print and now other variables being needed. Additionally, all of the information should be accessible in an easy and intuitive manner, it doesn't need to be stored that way, but it should be accessed and manipulated through a simple interface built using operator overloading. The existence of operator overloading means that we can (and will) also put effort into the efficiency of the structure without regard for its access patterns because they would not match the data one to one anyway. Ideally we can store the data in such a way that all components have  $O(1)$  access time and only the print data itself is stored without any auxiliary parts, however short of this the structure will be designed to achieve both of

these as closely as possible with a priority on memory use rather than access time. The PrintModel structure should also be fully reversible, back into its original g-code or something functionally equivalent, even after any optimization made for our printer's capabilities. Finally, the structure should allow for random access even when stored on nonvolatile storage media.

To meet the above constraints, as well as fulfilling the goals as much as possible, we will define our structure in a language that supports zero cost abstractions. Doing this means that we would not be adding any memory overhead for the format of our data structure and will only be storing the parts we are actually trying to store, along with access patterns implemented at compile time and then forgotten. We will probably use Rust for this due to its strong memory safety, this will be important for our program because we will be doing multithreading without the help of a kernel, and our PrintModel in particular will be referenced in a number of threads simultaneously. As for the object structure, it will contain several components including Layer objects to store layer data, as well as print parameters like layer height and laser power. Regardless of how the layers are stored, they should be accessible through an iterator from the object itself, that is to say that `object[i]` will return `object.layer[i]` if the PrintModel stores an iterable layer object, or if not then using brackets like that will return something of similar format as if the object had a layer iterable.

In order to facilitate random access from storage via memory mapping, all subcomponents should ideally have a predefined size; due to the expandable nature of this object however, predefining all subcomponent sizes will most likely prove impossible or extremely memory inefficient. Instead, subcomponent sizes can be stored as part of the file when it is saved so that the program can skip directly to the desired location when reading the file. The object can also associate relevant functions with itself, this would be done as a class if it were written in C++, but in Rust it would be done with traits instead. The object will also make heavy use of pointers for anything larger than the native pointer size of 32 bits in order to better segment the memory over the available storage. Of course, these subcomponents will have their values saved and not their references when being stored in nonvolatile memory. Additionally, the object will be fully reversible, and will have an associated function to generate G-code from itself; the G-code does not need to be identical to the original, but it must contain the same information to effectively produce an identical print. Finally, our representation of the print will be formatted as inputs for our printing functions. Each line or curve to draw and when, along with the laser power and other parameters will be stored, ready to be passed directly to the appropriate functions and executed by the printer. A block diagram of the PrintModel structure is shown in Figure 6.3-5.



**Figure 6.3-5 PrintModel**

### 6.3.6 GUI Coding Structure

In order for the user to interact with the printer, we will need some sort of interface. There are many ways this can be implemented like on a simple 12 section lcd, or in an android/iOS app over Bluetooth. We will make a GUI class to control the graphics being displayed on a 320x480 LCD display that will accept touch input in order to allow the user to enter commands for the printer. The GUI class will also interface with G-Code parser in order to update the user with a print time estimate and will also be able to send a stop print command to the relevant parts of the software if the user desires. The touch screen has several benefits for this project, including adding a more modern feel to the user interface, and also providing user inputs as a digital signal so processes like debouncing are not needed.

While there can be several classes controlling subcomponents of the interface, the overall system should be managed by a single class in order to provide a unified API for easier programming and more reliable testing. Among the features of the GUI class, it must have a way to read an SD card; even though all graphics components will be small enough to fit into the microprocessor's memory, there will be more than one "page" in the GUI so more components will be needed that will be stored externally. Another type of external communication is also needed, in order to do anything useful with the information from the user, the GUI system needs to be able to communicate with other systems to pass along its commands. Some commands the GUI may pass along include selecting the g-code file to print, starting the print, stopping the print, pausing the print, and setting parameters for the print like print speed. The GUI should also be able to interface with other systems in the other direction (receiving information) in order to facilitate processes like providing the user with the estimated time remaining, and the percentage of the print completed.

In order to meet the constraints listed above, and also fulfill our goals for the GUI, the software systems will be designed as follows. First, the API will depend on several lower-level classes including, but not limited to, an existing graphics library as well as a similar one for reading the SD card, a custom wrapper for the graphics library in order to handle specific parts of the GUI, a touch interface class, and a communication class to interact with other systems. The GUI class will unify all of these behind a single API that will provide all the functions needed at a high level to describe the appearance and function of the graphics to be displayed. A number of pages will exist in the GUI each displaying buttons for navigation, as well as other possible commands and relevant information. A home screen will act as a central navigation area where the user can access any (or most) other needed page for the task they want to perform. There should also be a files selection page the user can navigate to in order to choose the file to be printed, and a print settings page where the print parameters can be set or adjusted and the print started, stopped, or paused. Along with pages the user can interact with, information pages will also be available like a stats page where the user can see the elapsed print time, the estimated time remaining, the material used, etc. If there is enough time, a possible extra feature may include a 2d or 3d render of the print's progress so the user can see what's going on inside the powder.

## 7. Project Prototype Testing Plan

### 7.1 Hardware Testing

#### 7.1.1 Stepper Motor Testing

##### Test Description

The objective of stepper motor testing is to check if all purchased motors, along with the motor drivers, are working correctly. The test encompasses speed, and direction control using the microcontroller module. After completion of the test, we shall confirm whether or not the motors function properly.

##### Equipment and Components

The stepper motor testing will be conducted in the Senior Design Lab located in Engineering 1. The equipment and components used for motor testing are listed below:

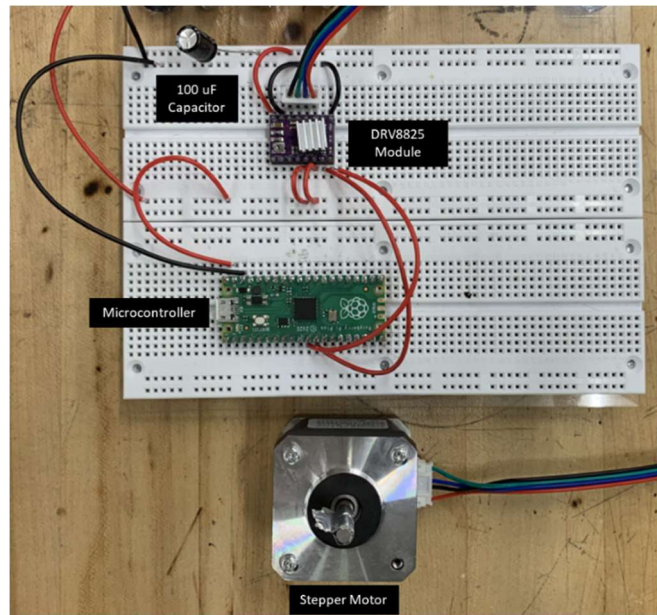
1. Multimeter
2. Power Supply
3. Breadboard
4. NEMA 17 Stepper Motors
5. DRV8825 Stepper Motor Modules
6. Raspberry Pi Pico Microcontroller Module
7. 100uF Capacitor
8. Aluminum Tape

##### Test Procedure

The first step is to determine the correct wiring of the stepper motor. Figure 9.2-1 below shows the output pins for the 17HS4401S stepper motor. The data sheet states that pins A and C are responsible for Coil A and pins B and D are responsible for Coil B.

Next step is to set up the stepper motor driver module. Figure 9.2-2 shows the pin layout of the DRV8825 stepper motor driver module. The motivation for stepper motor testing with the module comes from [9]. Before connecting the stepper motor to the driver module, it is important to set the current limit. From the stepper motor data sheet, we find that the operating current for the motor is 1.5 A. The potentiometer on the module is used to set the reference voltage. The current limit is set by adjusting the reference voltage to half the value of the current limit. Therefore, the reference voltage must be 0.75 V. Using the power supply, the board is powered by setting VMOT to the operating voltage (9V – 42V), RST and SLP to logic 3.3V, and the GND pins to ground. A 100uF capacitor is connected in parallel with the operating voltage input to protect the driver from voltage spikes. Once the power supply is turned on, measure the reference voltage by placing the probe on the top of the potentiometer. The potentiometer is adjusted until we read 0.75 V.

After setting the current limit, we connected the stepper motor and the microcontroller module to the driver module. A1 is connected to A, A2 is connected to C, B1 is connected to B, and B2 is connected to D. The STP pin takes a Pulse Width Modulation (PWM) signal. The motor speed is dependent on the frequency of the PWM signal. The microcontroller pin responsible for outputting the PWM signal is connected to STP. The DIR pin is responsible for direction. The motor spins clockwise when DIR is set to logic high and spins counterclockwise when DIR is set to logic low. DIR is connected to the microcontroller logic output pin. For this test, the micro stepping is set to full steps. Therefore, pins M0, M1, and M2 are not connected. A picture of the module wiring is shown in Figure 7.1-1.



**Figure 7.1-1: Stepper Motor Breadboard Testing**

After all of the modules are properly wired, a piece of tape is placed on the shaft of the motor so that it resembles a flag. The tape is used as a reference to observe the shaft movement. The stepper motor testing consists of running the motor at 4 different speeds with 5 second intervals. The microcontroller creates a square wave signal with increasing frequency. The test starts with 25 rpm and increments by 25 rpm every iteration. Once it reaches 100 rpm, it stops and runs the cycle again in the opposite direction. Once both cycles are complete, the test is complete.

#### Pass/Fail Criteria

The test is considered a success if the motor completes both cycles without missing a step. A failure is considered if the motor fails to complete the test sequence or misses a step. In order to complete the motor testing, every motor must be tested. After every motor is tested, each stepper motor driver module needs to be tested to ensure that all of the driver modules work. Once we ensure which motors and driver modules work properly, future linear motion tests can be conducted.

## Results

Once wiring was complete, we ran the test sequence for all motors and all motor driver modules. We ran the tests by setting the supply voltage to 9V. We measured the current draw from the module and found that the peak current draw is 0.63 A. With supply voltage and current, we can calculate the power demand. The motor driver draws 5.67 Watts. The motor driver power will be used for power supply design calculations. All of the purchased motors and motor drivers passed testing. With this conclusion, we can continue to linear actuator testing.

## 7.1.2 Linear Actuator Testing

### Test Description

The first objective for linear actuator testing is to build and test the linear actuator design that will be used for all linear motion. The purpose of the test is to see the effect of micro stepping and scanning speed on linear movement precision. The second objective is to build and test the x-y scanner design. Using the findings from the first test, we can tune the output so that speed and precision of x-y scanning is maximized

### Equipment and Components

The linear actuator testing will be conducted in the Senior Design Lab located in Engineering 1. The equipment and components used for linear actuator testing are listed below:

1. Equipment List from Stepper Motor Testing (excluding electrical tape)
2. Lead Screws
3. Guide Rods
4. Flanged Nuts
5. Pillow Bearings
6. Linear Bearings
7. Motor Couplers
8. 3D Printed Brackets
9. Bump Switches

### Test Procedure

The first step is to construct the linear actuator. The lead screw and guide rods will be cut according to the X-Y track system dimensions. There are 3 main 3D printed components used for testing. The first 3D printed part is a bracket used to mount the motor and join the left and right guide rods. The bracket will have screw holes to mount the assembly for testing. The second 3D printed part attaches to the flanged nut and houses two linear bearing for the guide rods. The second part will act as the base for the laser module. For the sake of test, we will attach a base that holds a pen. The pen will be used to draw out the path of the laser beam. The third 3D printed part is the end piece for the linear actuator. It joins the other ends of the two guide rods and holds the pillow bearing for the center lead



screw. It also houses a linear bearing that will be used in future testing. The bracket will also have screw holes to mount the assembly for testing.

The first aspect of the linear actuator we will test is scanning speed and its effect on accuracy. Each iteration of the test will move the pen from one end of the actuator to the midpoint at a constant speed. Once it reaches the midpoint, it will pause and move at the same speed back to the start position. The pen will draw the path onto a piece of printer paper. The test will start at a scanning speed of 30 mm/s and increase 5 mm/s every iteration. At the end of each iteration, we will measure the linear path the pen took and compare it to the desired length. At slow speeds, we expect to have little backlash and accurate movement. Therefore, the path measurement will be equal to the desired path length. As the speed increases, we will expect more backlash and possible overshoot. If this is the case, the path will be longer than the desired path length. Although the SLS printer scanning speed is primarily determined by the sintering characteristics between the laser and the powder, we want to quantify the acceptable max speed that the linear actuator can accurately move.

After quantifying the acceptable max speed, we want to test the affect micro stepping has on linear motion. From our research, we found that increasing the micro step is directly proportional to the smoothness of motor operation. We will set the scanning speed at a constant 30 mm/s, and it will follow the same path as the first test. Every iteration, we will increase the micro step. It will start at a full step, then continue to increase the micro step by a factor of 2 until it reaches 32. The test will not utilize the pen and it is purely observational. A camera will be used to record each micro stepping run.

Once we finish testing the single linear actuator, we will construct and test the full x-y track system. The linear actuator constructed for the first test will be modified and used as the y-axis track. The construction of the x-y track is described in 5.1.5 X-Y Track System. Once constructed, we will test the x-y track capabilities. Using the conclusions from the single actuator testing, we will set our motor conditions accordingly. The test involves having the x-y track draw simple two-dimensional shapes, exactly like a pen plotter. Three separate tests will be conducted. The first test is drawing a square, the second test is drawing a triangle, and the third test is drawing a circle. After all, three tests are complete, we will use the results as a reference and continue to tweak scanning speed and micro stepping until we reach our desired result.

After passing the initial tests, we will perform tests that will reflect future printer capability. The next test will be the line test. The line test consists of drawing 10 lines with equal length and measure the consistency of the printer. The line test will be performed at the top speed we find in the first test. The 10 lines will be drawn in a continuous fashion with the line drawing a 100 mm line moving up 10 mm and drawing another line in the opposite direction. This will continue until the tenth line is complete. All lines will then be measured and compared. The final test is using actual g code to draw the future laser path of the printer. For this test, we developed a model called “Smiley”. The Smiley model is shown in Figure 7.1-2. The test results will then be matched to the model to compare accuracy.



**Figure 7.1-2: “Smiley” Model**

### Pass/Fail Criteria

For the first test, a successful run consists of the pen tracing the exact length that was expected. A failure is considered if the pen traces a longer path than the programmed path. The speed right before the failure occurs will be noted as the maximum scanning speed. For the second test, a successful run consists of drawing the desired 2D object with correct dimensions. A failure is considered if the 2D shape begins to lose dimensional accuracy. Excessive jitter and other abnormalities in the drawing path will be considered as a failure. The speed previous to the failure will be noted as the new maximum scanning speed. For the third test, we will take the average of all measured values and a success is considered if the result is  $\pm 1\%$  of the intended value. For the fourth test, we shall compare the model and the final results. A success run is considered if the final results match the model in Figure 7.1-2.

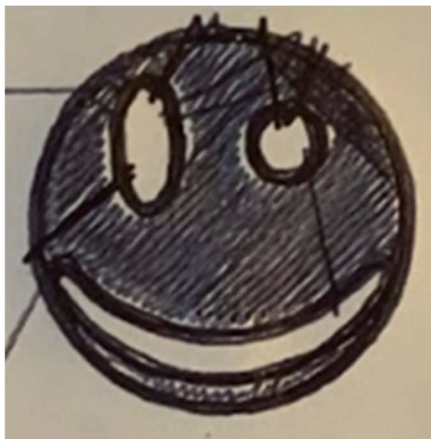
### Results

The first test was successful. We found that the linear actuator performs best between 30 mm/s and 50 mm/s. In terms of micro stepping, the higher the micro stepping, the smoother the motor operation. Therefore, we chose 1/32 micro steps for all further actuator testing and printer operation. The second testing was successful as well. The XY track system was capable drawing a square, triangle, and circle with the exact dimensions as intended. The third test was also a success. During the line test, we notice that the friction between the paper and pen caused the paper to bunch up in certain areas. This accounts for the measurements that were off of 100 mm. We do not see this as an issue since the laser module will move frictionless on the XY Track system. The average value for X was 99.54 mm and the average value for Y was 10 mm. Even with the friction between the pen and the paper, the line test measurements were successful. The actual measurements of the line test are shown in Table 7.1-1.

**Table 7.1-1: Line Test Measurements**

X	Y
100	10
100	10
100	10
98.9	10
100	10
98.5	10
100	10
99.1	10
100	10
98.9	10

The final test was the g code test. The g code used for the test was generated using a slicing software called Cura. For the test, we set the scanning speed to 50 mm/s. While the main purpose of the test was to compare the XY Track system results to the model, we also wanted to verify the actual scanning speed. We did this by recording the test and measuring the time it took to draw the outer diameter of Smiley using the recorded frames. The measured diameter of Smiley was 29 mm which means its circumference is 91.11 mm. The outer diameter took 66 frames complete. Using 30 frames per second, the duration of drawing the outer diameter was 2.2 s. Dividing the circumference by the time, we find that it was 41.41 mm/s. The actual results were lower than the expected 50 mm/s. This could be partly due to the fact that the g code slows down for some segments of the scan due to built-in formulas in the slicer software. All in all, we were able to achieve a successful drawing at a scanning speed greater than our required scanning speed of 30 mm/s. The results for Smiley are shown in Figure 7.1-3.



**Figure 7.1-3: XY Track System Results**

## 7.1.3 Laser Driver Testing

### Test Description

The objective of laser driver testing is to check if the designed circuit can properly deliver a constant current to the sense resistor. The first test consists of breadboard testing with the power MOSFET and using provided passive components and an op-amp. The second test is conducted after PCB fabrication to check if the constant current is equal to the desired value. Once both tests pass, the laser driver will be able to drive the laser diode.

### Equipment and Components

The laser driver testing will be conducted in the Senior Design Lab located in Engineering 1 building. The equipment and components used for laser driver testing are listed below:

1. Multimeter
2. Power Supply
3. Breadboard
4. IRL7833PBF Power MOSFET
5. Low Offset Voltage Op-Amp
6. Passive Components

### Test Procedure

The first test that is conducted is the laser driver breadboard test. To construct the breadboard test, we will follow the laser driver schematic shown in Figure 5.2-4. All of the resistors used in this test will be measured and recorded. A 10k-ohm resistor and a 1k-ohm resistor is used for the input voltage divider. The output voltage of the voltage divider is connected to the non-inverting pin of the op-amp. The gate of the power MOSFET and the output pin of the op-amp is connected via a 100-ohm resistor. The sense resistor is connected to the source of the power MOSFET and the inverting pin of the op-amp. The supply pins of the op-amp are connected to 10uF bypass capacitors. Since the op-amp used for this test is provided in the lab, the data sheet for the op-amp is used to set the proper supply voltage. The power supply is set to the supply voltage of the op amp, 5 V and 3.3V. To read the current, the diode shown in the schematic is replaced with the probes of the multimeter.

A total of five tests will be conducted, and the sense resistor value will vary for each test. The sense resistor values are 100-, 200-, 300-, 470-, 680-ohm resistors. These values were chosen because they are readily available in the Senior Design 1 lab. Using the offset voltage of the op-amp and the voltage at the non-inverting input, the expected current can be calculated. A tolerance of  $\pm 1\%$  will be given for the expected current. The expected current is calculated using the equation below. For each sense resistor, the input voltage and output current will be recorded. Once the laser driver passes the first test, we will fabricate the PCB.

$$I_{expected} = \frac{V_{in} + V_{os}}{R_{sense}}$$

The second test consists of testing the actual laser driver. The test must be conducted before connecting the laser diode to the driver. The test consists of powering the laser driver and reading the current between the pads where the laser driver will be connected. The expected current was analyzed in a LT Spice simulation. From the simulation, the expected current through the diode is 3.019 A. Once the current is verified, the driver can drive the diode.

### Pass/Fail Criteria

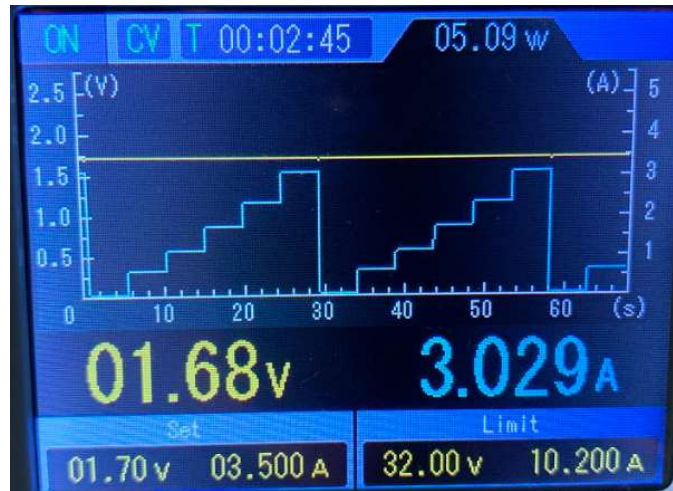
For the first test, a successful test result is considered if the output current is  $\pm 1\%$  of the expected value. If the current is not within the specified range, it is considered a failure and redesign must occur. For the second test, a successful test result is considered if the output current is  $\pm 1\%$  of 3.019 A. If the current is not within the specified range, the test run is considered a failure and redesign must occur.

### Results

The purpose of the first test was to test the design concept on a breadboard. All of the resistor values resulted in  $\pm 1\%$  of the expected output current. Once the proof of concept was tested, the PCB was manufactured, assembled, and ready for official testing. The first test was setting the digital potentiometer so that we can achieve an output current of 3.019 A  $\pm 1\%$ . In order to measure the output current, a wire was soldered to the cathode terminal of the laser driver which is directly connected to the drain of the MOSFET. The forward voltage of the diode is 4.3 V and the voltage applied to the diode is 6 V, therefore, the voltage seen by the drain of the MOSFET would be 1.7 V. A power supply was attached to the wire and set to 1.7V to simulate the conditions of the diode being connected to the circuit. The current can then be read off of the power supply. A simple Arduino script was written to set the digital potentiometer via SPI. When setting the digital potentiometer to 5, the measured output current was 3.029 which is within the successful range. Once we verified that max current was achievable, we tested other potentiometer values and measured the output current. The potentiometer values tested, and the measured output current is shown in Table 7.1-2. From the table, we can see that we are capable of changing the laser power successfully by changing the current seen by the diode from 0 – 3 A. Figure 7.1-4 shows a screenshot of the power supply when the digital potentiometer value was set to 5. The power supply used also displays a graph with respect to time. The graph shows the other current values listed in Table 7.1-2.

**Table 7.1-2: Laser Driver Measured Current**

Digital Potentiometer Value (0-255)	Measured Output Current (A)
5	3.029
51	2.259
102	1.671
153	1.095
204	0.579
255	0



**Figure 7.1-4: Max Laser Driver Current**

## 7.1.4 Kerf Testing

### Test Description

The kerf is the amount of material that is removed when laser cutting any material. In order to have exact parts, the kerf needs to be accounted for in the design. Since we will be using finger joints to join the pieces of wood, the gap between sections needs to be minimal. The objective of kerf testing is to find the kerf of the laser cutter that will be used for enclosure and reservoir fabrication.

### Equipment and Material

The kerf testing will be conducted in the Innovation Lab located in Engineering 2 building. The equipment and material used for the test are listed below:

1. Laser Cutter
2. ¼ inch plywood

## Test Procedure

The first step is creating a test design to find the optimal kerf value. The design chosen is a 90-degree corner using finger joints. The software used to develop the 3D model and the DXF file for the laser cutter is Onshape. Onshape automates the finger joint design process and accounts for the kerf when creating the finger joints. Five separate corner pieces with different kerf values will be cut out for the test. When researching kerf values for laser cutters, the most common kerf value using wood is 0.08 mm. Therefore, the kerf values that will be tested are 0.07, 0.075, 0.08, 0.085, and 0.09 mm. After designing the corners, the 3D module is converted to a DXF file. Once the pieces are cut, we will join the corner pieces for each kerf and deduce which kerf value offers the best fit.

## Pass/Fail Criteria

The pass criterion for the kerf testing is very subjective. The joint with the best fit is considered the passing kerf value. The joint should not be too loose or too tight. If the joint is too loose, the parts will not hold together properly, and total alignment will be off. If the joint is too tight, it could lead to cracking or improper fit when other joints are used. The passing kerf value will be used for reservoir and enclosure fabrication.

## Results

Kerf testing was not complete since ¼ inch plywood was no longer considered as the material for the mechanical base and enclosure. The wood used for the final design was ½ inch which is too thick for laser cutting. All wood was cut using personal machinery such as a Miter saw, and a jig saw.

# 7.1.5 Powder Delivery Testing

## Test Description

The objective of powder delivery testing is to ensure that the designed sweeper subsystem properly delivers powder from the powder reservoir to the build reservoir. The test is conducted after PCB fabrication and construction of mechanical assemblies.

## Equipment and Components

The powder delivery testing will be conducted at a group members house. The modules and parts used for the test are listed below:

1. Power Subsystem
2. User Interface
3. PCB with Stepper Motor Drivers and MCU
4. Reservoir Assembly
5. Plate Subsystem
6. Sweeper Subsystem
7. Confectioners' Sugar

## 8. Iced Tea Mix

### Test Procedure

The first part of the test is assembling the complete system without the laser module and the X-Y track subsystem. Once construction is complete, the system will complete a run without powder in the reservoirs. The dry run is done to ensure that all subsystems move properly. Once the powder plate is at the topmost position, the powder delivery sequence is complete. When the dry run test passes, a test using powder can be conducted. The powder used for the powder delivery test is confectioners' sugar. Once calibration is complete and the powder is loaded into the powder reservoir, the powder delivery can begin. Throughout the test, we will observe the build plate. The build plate should always have an even layer of powder after every swipe. The test will continue until all of the powder from the powder reservoir is delivered to the build reservoir and the excess-powder reservoir.

### Pass/Fail Criteria

The test is deemed successful if two conditions are met. The first condition is that all of the powder in the powder reservoir must be delivered without fault. If a system fails in executing the correct task, the test is considered a failure and the failed system must be redesigned. The second condition is that the powder delivered to the build reservoir must be even for every sweep. If there are divots in the build area, the test is considered a failure and the sweeper mechanism must be redesigned.

### Results

First, we tested the powder delivery system with no powder. The system was capable of running through a complete powder transfer with no issue. After the successful dry run, we loaded the confectioners' sugar and ran the powder delivery test. The powder delivery system was capable of moving powder from the powder reservoir to the build reservoir successfully. The issue we confronted with the confectioners' sugar was that the build area was not completely flat. There were little divots in the build area which would lead to unsuccessful prints if the printer were to sinter on top of an uneven area. The result of confectioners' sugar powder delivery is shown in Figure 7.1-5. We began testing with other powders to see if we could achieve better results. One of the powders we tested was iced tea mix. The tea mix was comprised of 92% sugar and was dark in color which is needed when sintering with a blue laser diode. Figure 7.1-6 show the results of powder delivery with the iced tea mix. Notice that it creates a much smoother surface compared to the confectioners' sugar. The iced tea mix ended up being the powder of choice after extensive testing.





**Figure 7.1-5: Confectioners' Sugar Test Results**



**Figure 7.1-6: Iced Tea Mix Results**

## 7.1.6 Microcontroller Board Testing

### Test Description

Our entire system's operation depends on the microcontroller module. Thus, we need to make sure that the board can communicate via USB, be programmed, and properly send all signals to our pin headers.

### Equipment and Components

Power Supply  
Multimeter  
Breadboard  
Oscilloscope

Computer

Every component that is part of the microcontroller design

### Test Procedure

The goal is to use the microcontroller designed as if we were using board sold with the chip. In the scenario that not all other modules are ready to be tested with the microcontroller, we would have to find other ways to test the same functionalities. A simple test for the micro-USB interface, and GPIO pins would be writing a program that turns LEDs on and off. This code needs to be saved to the memory by connecting the computer to the board using the micro-USB port. Then, LEDs are connected to corresponding pins to test functionality. To test the oscillating crystal we can compare the frequency at which the LEDs flash with the programmed frequency.

### Pass/Fail Criteria

The module must be seen by the computer when connected via USB.

The module must be successfully programmed by the Arduino IDE.

The pins will operate according to the program uploaded to the module.

### Results

The first iteration of the design did not pass the first test. However, via an acquired external USB module, we were able to upload a program through the Arduino IDE and see its functionally reflected in the test LED. In the revision of the design we were able to connect via USB using only our designs.

## 7.1.7 AC-DC Converter Testing

### Test Description

In this test, we aim to ensure that our design is capable of taking the 60Hz 120V<sub>RMS</sub> input from the wall and converting it into the desired constant 12V output. The first part of the test will consist of a prototype circuit built on a breadboard. A second test round will be needed once the PCB has been acquired. For the second test, it could be helpful to split it into 2 sub tests. One where the components are placed on the breadboard and wired to the PCB. This would ensure that the PCB traces are correct and also make it easier to swap components if needed. Once this has passed, the last test consists of soldering the components into our PCB and prove that the fully assembled board works as intended.

### Equipment and Components

Function Generator

Multimeter

Oscilloscope

185E16 50/60Hz Transformer  
BR1005 Full Bridge Rectifier  
4700uF 25V Electrolytic Capacitor  
Breadboard

### Test Procedure

For initial testing, we will use wire fittings to connect our cord wires to the corresponding transformer terminals. We then use wires to connect the secondary terminals of the transformer to our rectifier through a breadboard. We then use jumpers to connect the rectifier to the capacitor in the breadboard. Once the test circuit is built, we start by using the function generator to create a 20V peak-to-peak wave and feed it into our transformer. This way we would ensure that our circuit is wired correctly while still working with a relatively low voltage. From this initial test, we expect to have a  $1.4V_{\text{peak-to-peak}}$ . This comes from the parallel configuration of the transformer and turn ratio of 115:16.

Once the expected stepped down voltage has been achieved, we will pass it through our full bridge rectifier. Using the oscilloscope, we will ensure that that we have successfully rectified the wave. However, upon successful rectification, we will also see a drop in voltage. This is due to the forward voltage of the diodes used in our bridge. From the BR1005 datasheet we see that this forward voltage is of 0.55V per component. Given that both the positive and negative sides of the input wave pass through a pair of diodes in the rectification process, it means that the fully rectified wave should have its peak reduced by 1.1V. This leaves us with a final rectified amplitude of about 0.3V. The last step of this  $20V_{\text{peak-to-peak}}$  test is measuring the output when using the capacitor to maintain the voltage at the peak of the wave.

The next step in our testing process is to measure the output when we plug our cord into a regular wall outlet. When going through this stage of the testing, we need to guarantee safety. Unintentional contact with the primary side of the transformer might result in electric shock. This means that all exposed terminals on the primary side of the transformer need to be covered. Additionally, any change to the circuit must be done while the cord is disconnected from the wall outlet. Only when all the aforementioned safety guidelines have been followed can we then proceed to connect our circuit to the 120V outlet.

When using wall outlet power for our input, we will follow a similar testing process as when using the function generator. We will use the oscilloscope to look at the stepped down wave, then revise that the rectification works as expected, and lastly checking the DC output voltage from the capacitor. However, there are a few more items to pay attention to. First, we need to keep in mind that the power industry works with root-mean-squared (RMS) voltages. Thus, the real peak of these waves is  $\sqrt{2}$  times greater than the specified  $V_{\text{RMS}}$ . Thus, when calculating the expected value, we will add  $\sqrt{2}$  to our equation, leaving us with  $V_o = \frac{\text{Outlet Voltage}}{115} * 16 * \sqrt{2}$ . Considering the outlet voltage varies from  $110V_{\text{RMS}}$  to  $120V_{\text{RMS}}$  the final peak of the stepped down voltage would be in the range of 21.6V to 23.6V.

After considering the full bridge rectifier's diode forward voltages, we would be reducing this range to about 20.6V to 22.6V. When checking the DC output with the capacitor included in the test, we must not only look at the DC value but also at the output ripple. With all the previous process passing we would then test adding loads to ensure that we are providing sufficient power. During the load test we must also take temperature measurements. This is important because it will tell us if we need to provide a cooling system.

#### Pass/Fail Criteria

The transformer successfully steps down the voltage to  $16V_{RMS}$

The bridge rectifier and capacitor successfully provide a steady DC voltage

The AC/DC converter provides sufficient power to power all system components

#### Results

The module results were exactly as expected and we were able to run our system from this power supply.

## 7.1.8 DC/DC Regulators Testing

#### Test Description

When testing buck regulators, we are looking for our desired output voltage from the specified range of input voltages. Additionally, the main reason to use switching regulators instead of linear is the power efficiency, thus we want to make sure that we get low dissipated power. This will be measured by keeping track of input and output voltages and currents.

#### Equipment and Components

Power outlet

Multimeter

Power Supply PCB

#### Test Procedure

We will connect the power module to the wall power outlet and measure the voltage at each of the regulator's output. Once output levels have been verified, we will run our system from these regulators to ensure that sufficient power is transferred.

Additionally, we will measure the soft-start for the 6V regulator using the trigger feature of an oscilloscope.

#### Pass/Fail Criteria

The 9V regulators provide an output voltage no less than 8.9V and sufficient current to properly operate the stepper motors.

The 5V regulator provides a voltage no less than 4.5V and sufficient current for the microcontroller

The 6V regulator provides a voltage no less than 5.9V and sufficient current for the laser module (3A).

The 6V regulator has a soft-start of at least 25ms.

## Results

Every regulator met the specifications and were able to provide power for the system during long periods of time. Additionally, from Figure 7.1-7 we can see how the 6V regulator has a soft start of about 30ms.

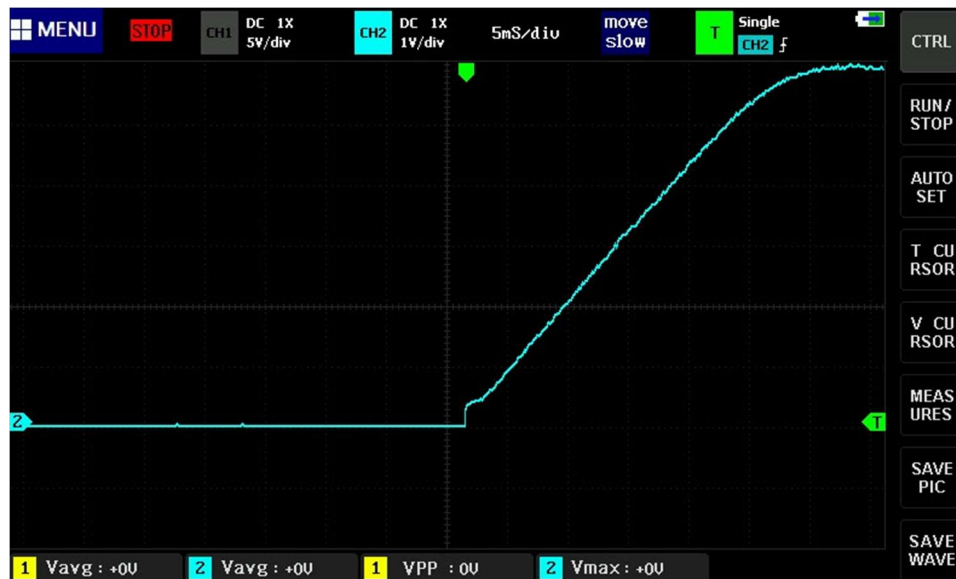


Figure 7.1-7: 6V Regulator 30ms Soft Start

## 7.1.9 Laser Diode Testing

Upon receiving the laser diode, testing must be carried out to characterize the diodes behavior and ensure it works as expected. I-V curves must be established to determine how much current must be supplied through the driver to deliver a particular output power. Steady state temperature control must be correct and function properly. The spectrum must be characterized at each temperature.

Before characterizing the diode, proper temperature control must be established. Laser diode behavior changes drastically with temperature; increasing temperatures lowers the output power, decreases the threshold current, and raises the operating wavelength. These

will all negatively affect the print quality of the SLS printer if left unmonitored and not kept constant. Once the temperature housing and assembly has been created, the diode will be placed inside and left to run for three hours at near maximum current and output power. Thermal readings of the diode and housing will be monitored consistently during this time, with adjustments made to the cooling variables to achieve a desirable steady-state temperature. Once a steady-state temperature has been reached after the three-hour period, the process will be repeated for different levels of input current to ensure the temperature does not fluctuate at any level of operation. Only after temperature control has been established can the other tests be performed.

The I-V characteristics of a laser diode are determined by changing the current input of the diode and measuring the output power at certain temperatures. The materials needed to create the I-V characteristics are listed below:

1. Laser diode.
2. Safety goggles (Optical Density 4+ for 447 nm)
3. Power meter/photo meter with high maximum optical input. Sensitivity should be in the visible region below 450 nm.
4. Temperature controlled setting.
5. Correctly tuned driver.
6. Condenser lens.
7. Lens mount with retaining ring for 2" diameter optics.
8. Lens posts.
9. Mounting base plate.

The procedure to determine the I-V curve is listed below.

1. Connect the optical components as shown in the diagram.
2. Turn on the temperature control and select a specific temperature for operation.
3. Turn on the diode driver and slowly increase current to below threshold of the diode.
4. While below threshold, measure the current, voltage, and power of the emitted light.
5. Increase the current steadily in equal intervals, taking measurements at each interval.
6. Once near the maximum current, return to step 2 to select a different operating temperature.
7. Plot P versus I.

The spectrum of the diode can also be recorded during the determination of the I-V characteristics. To do this, a spectrometer with a sufficient neutral density filter to ensure the photodiode does not get damaged can be placed in the beam path between optical power readings.

The divergence of the laser diode should also be determined. This will be necessary to accurately calculate the beam size upon entrance into the spherical or cylindrical lenses, as well as the beam waist and divergence upon exiting the focusing system. Finding the

divergence is basic trigonometry and does not require many components, so they will not be listed.

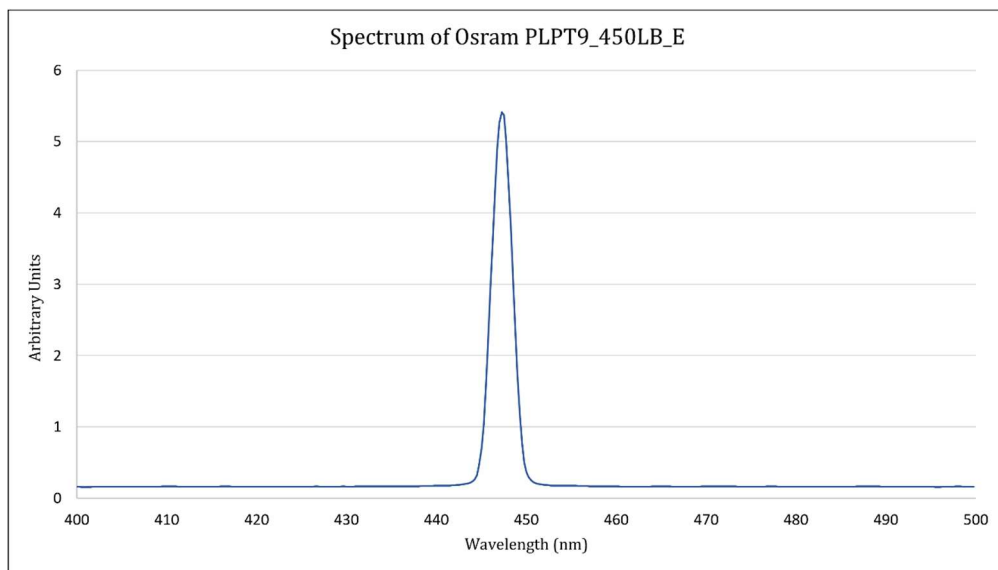
To begin the laser diode divergence measurements, the diode will be mounted and secured to a stable current source and temperature controller. A piece of graph paper with graph size known is secured to a beam stop. The distance from the diode to the beam stop is measured, and the size of the beam at the beam stop can be determined from the graph paper. Using this, trigonometric relationships can be used to find the divergence angle in both the fast and slow axis of the diode's beam.

The results of the tests were mixed. The laser power was unable to be directly measured, but instead indirect measurements were made. The delivered power was in line with other high-power diodes of similar strength. Because inaccurate power measurements were able to be conducted, the diode power was assumed to be accurate.

The divergence of the laser diode was higher than expected. By placing the laser diode in a copper module raised a substantial distance off the ground, angles could be found between the laser diode beam and the ground. Due to this larger divergence, significant spherical aberrations existed in the system and did not allow for best focusing.

Thermal characteristics of the laser diode within the copper module were also conducted. Without the fan, an input current of 300 mA heated the copper module in tens of minutes. However, with the fan, an input of 2.5A could be sustained without excessive heating of the module.

The spectrum of the laser diode is shown below in Figure 7.1-8. The output is extremely narrow and provides excellent control over sintering capabilities. The spectrum shows this diode is ideal for prototypical sintering printers with a peak wavelength of 447.39 nm.



**Figure 7.1-8 Spectrum of Laser Diode**

## 7.1.10 Lens Testing

Once the spherical lenses are received, testing must be done to ensure they follow the desired specifications and performance. This includes, most importantly, ensuring the focal length is correct. With the material of the lenses known, the refractive index is known. To perform the experiment, the following materials are needed:

1. Lens to be tested.
2. CMOS camera.
3. White light source (fiber bundle).
4. Post holders and mounts.
5. USAF resolution target.
6. Optical rail.

To begin the test, the post holders and mounts are placed along the optical rail at equal heights which defines the optical axis. The lens is inserted into a mount and placed an assumed focal distance away from the CMOS camera. The USAF resolution target is placed at the other side of the lens and illuminated by the white light fiber bundle. The focal length of the lens is defined when the highest quality image can be observed of the resolution target by the CMOS camera. This focal length can then be compared to the stated focal length on the data sheet.

The lenses performed as expected. The LA1608 lens had a focal length of 75 mm. The LJ1598L1 lens had a focal length of around 4 mm and the LJ1638L1 lens had a focal length of around 22 mm. Both measurements agree with the specifications.

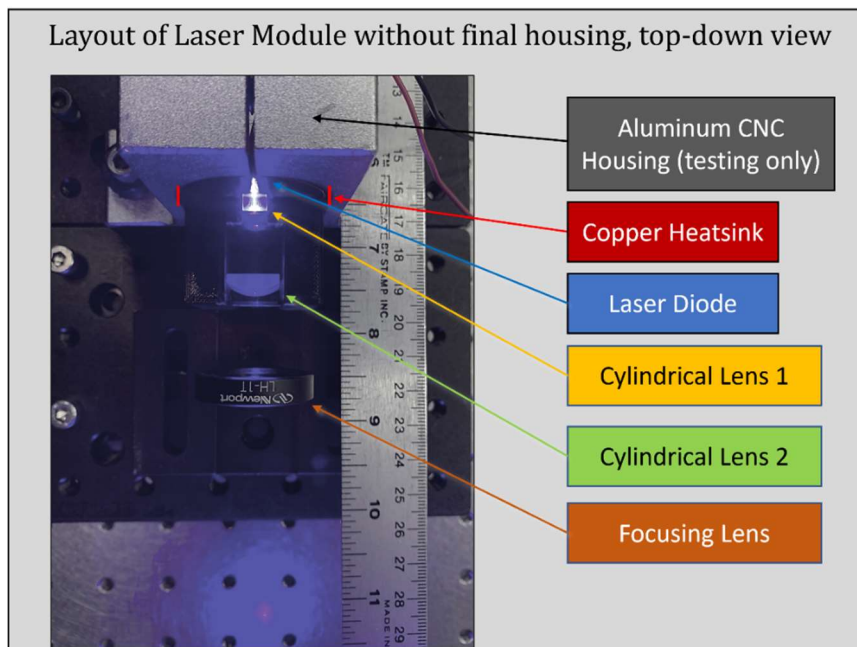
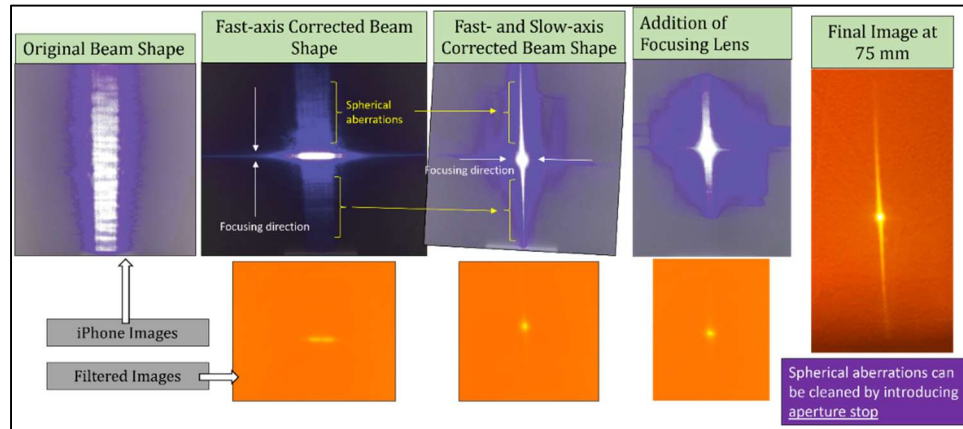


Figure 7.1-9 Layout of Laser Module Lens Tests



Figure 7.1-9 above shows the layout of the laser module without the final housing. The lenses were aligned as designed. This was the testing configuration to determine the focusing and beam shaping capabilities of the lenses.



**Figure 7.1-10 Lens Test Results**

The results of the lens testing are shown above in Figure 7.1-10. The original beam shape is highly divergent. After entering the first cylindrical lens, the beam is corrected along the fast axis, as shown in the image titled “Fast-axis Corrected Beam Shape.” The focusing direction is depicted by the white text and arrows, with resulting spherical aberration shown by the yellow text, curly brackets, and arrows. After the beam enters the second cylindrical lens, the slow axis is focused. Again, the focusing axis is depicted by white text and arrows, while the focused spherical aberrations are shown by the yellow arrows. The spherical aberration’s focused effect turns into tails along the fast axis direction. Finally, after the beam enters the focusing lens, the beam is focused to a small spot. The spot size of this system was not measured directly. Instead, the spot size was measured after introduction of the lenses into the 3D printed module.

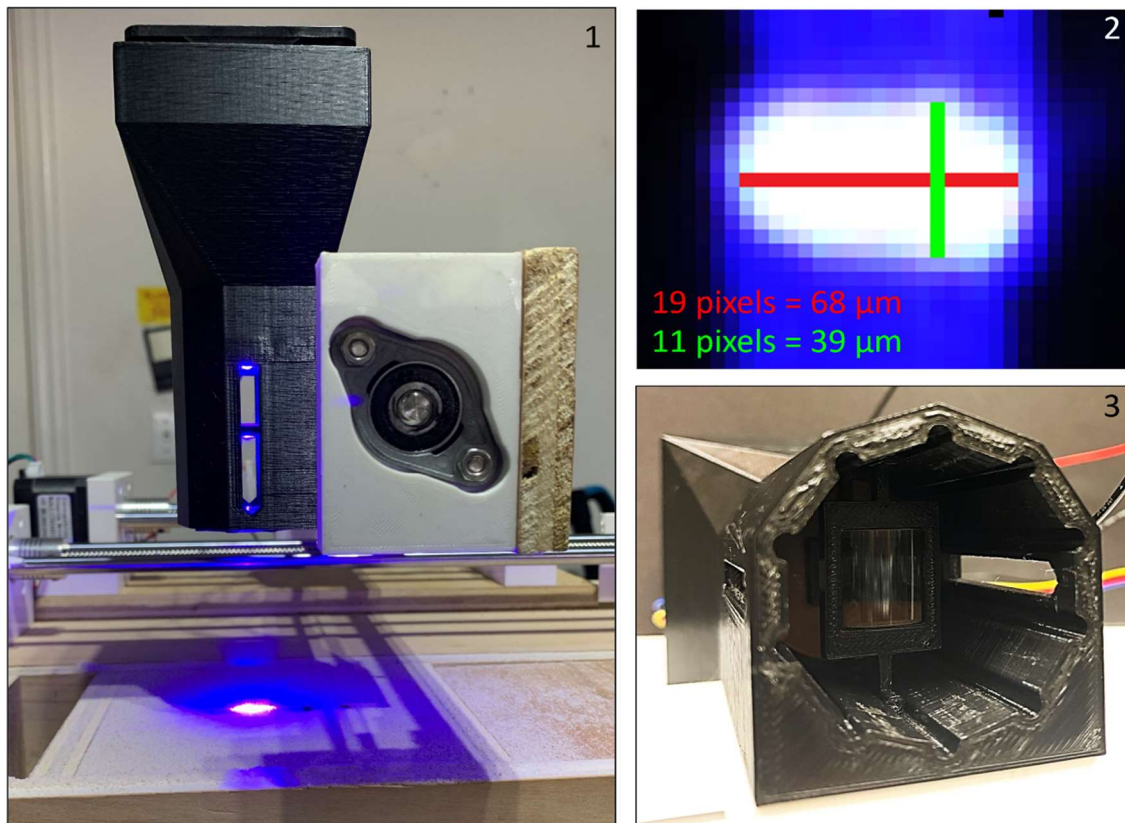
### 7.1.11 Laser Module Testing

The laser module was printed with a Prusa iMK3S printer. PLA was used as the material, and the layer height was set to 0.2 mm. Testing the laser module consisted of finding the appropriate distances between the lenses within the module as well as a spot size measurement. The lens mounts were printed in an array of different sizes as a parameter sweep of inner dimensions. The lenses were then fit in each mount to find the best mounting size. Figure 7.1-11 below shows the lenses in their best-fit mounts. Panel 1 shows the fast axis correcting cylindrical lens. The lens is mounted in a surrounding PLA enclosure with two protruding stems. The stems act as the positive component of the optical rail, with the channels being the negative component. Panel 2 shows the slow axis correcting cylindrical lens. The stems are in the same location as the fast axis lens mount. The spherical lens mount was designed with an aperture stop to diminish the effects of spherical aberration, as well as to enclose the end of the module such that air does not leave and disturb the powder below. All channels are closed with stems, and the lens acts as the barrier between the inside of the module and the outside.



**Figure 7.1-11 Printed Lens Mounts**

The constructed module and its results are shown in Figure 7.1-12 below. Panel 1 shows the entire module assembled and with the diode on. The three lenses can be seen from the side slit where blue light is refracted off of each lens. The top lens is the fast axis correcting lens, the middle lens is the slow axis correcting lens, and the last lens is the focusing lens. Panel 2 shows the focused spot size. The size of the spot was 68 microns by 39 microns, achieving 86% beam shape correction. Much higher beam correction merit could be reached with a finer layer height on the laser module and a better method of aligning the lasers once inside the module, as it was very difficult to align the lasers precisely with the given design. Panel 3 shows the slow axis correcting lens inside the module and provides insight as to how the mounts move along the optical column.



**Figure 7.1-12 Final Laser Module Results**

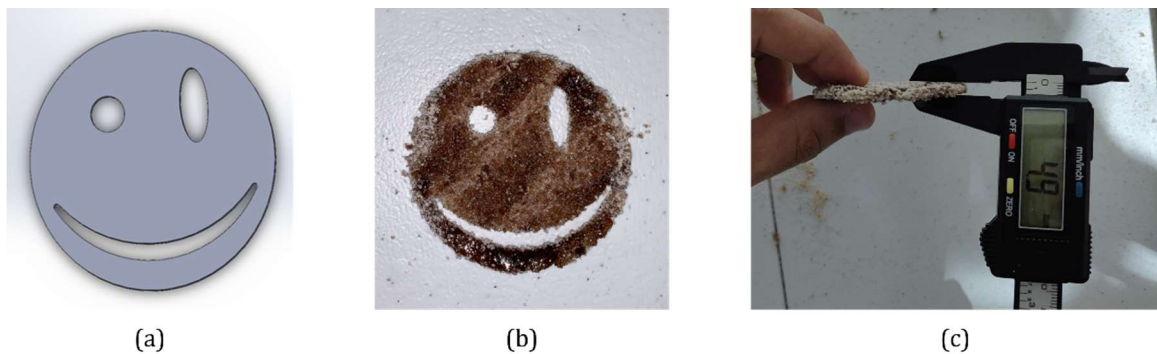
## 7.1.12 Laser Sintering Testing

### Test Description

The sintering capabilities of the printer will be the final test. The goal of this test is to determine the appropriate process parameters to have the highest quality sintering with a given powder material. This will be accomplished by printing lines with varying process parameters and seeing which lines have the best qualities, such as strength, porosity, surface roughness, and uniformity.

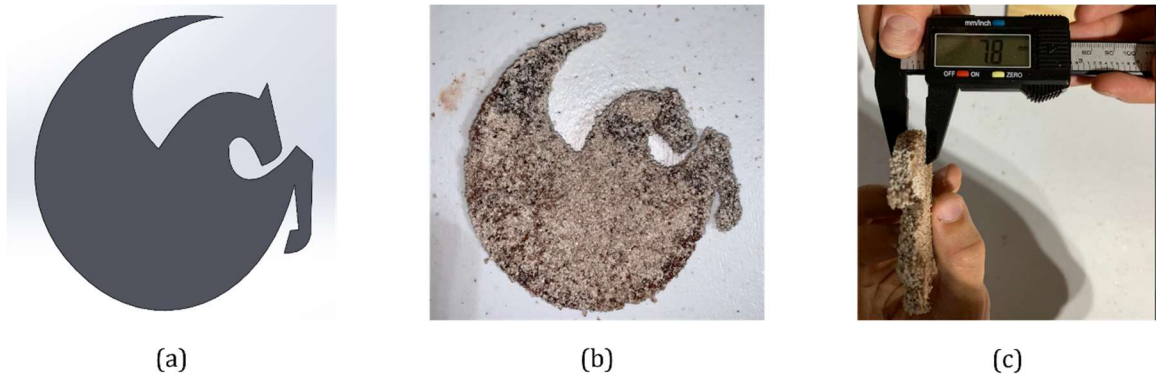
### Results

Sintering tests were performed with the raspberry tea powder, as this gave best results. The ‘Smiley’ model and the UCF Pegasus were both printed with great success. Stress tests of sintering at 50 mm/s were also conducted. Figure 7.1-13 below shows the ‘Smiley’ model test results. Panel (a) shows the model in the control software. Panel (b) shows the model after it has been removed from the print bed. Panel (c) shows the thickness of the ‘Smiley’ model to be 4.9 mm. This print was two layers thick. After many runs, successful process parameters were established, and the model was printed at 30 mm/s.



**Figure 7.1-13 ‘Smiley’ Sintering Test Results**

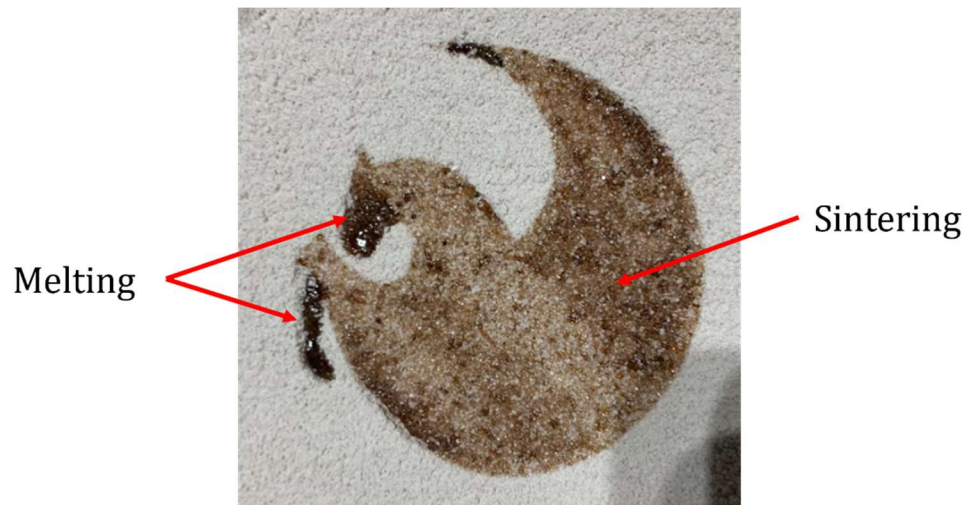
Next, the UCF Pegasus was printed at a similar speed of 30 mm/s. This test demonstrated the reproducibility of the printer. Figure 7.1-14 below shows the UCF Pegasus test results. Panel (a) shows the model in the control software. Panel (b) shows the printed model after it was removed from the print bed. Panel (c) shows the thickness of the print at 7.8 mm. This print was four layers thick. Again, the printer showed successful results as all systems were operating in unison and the model was fabricated successfully.



**Figure 7.1-14 UCF Pegasus Sintering Test Results**

Because the sintering process is extremely sensitive, variations in surface melting were seen. As shown in Figure 7.1-15, the UCF Pegasus logo has variations across the top surface. Some portions of the print are a very dark brown while others are much lighter colored. These differences can be attributed to the length of the scanned lines. While the printer sinters shorter lines, local thermal energy accumulates and does not have time to dissipate as the next line is scanned very quickly. This results in an overall higher threshold of heat due to exposure, and the powder experiences thermal degradation, indicative by the dark color. Lines with longer scan lengths exhibit lighter colors, indicating successful sintering without full melting.

The issues described above can be controlled in namely two ways. One way is to modulate the laser power in a linear fashion with the length of the scanned line. This would be easy to implement and would help distribute thermal energy equally to the print bed. Another way is to increase the hatch distance of the print. This also allows the thermal energy to be more equally distributed across the print bed.



**Figure 7.1-15 Differences in Sintering Quality**



Finally, high speed tests were conducted to determine upper limits of the printer. This test was performed at 50 mm/s. The object to be printed was a square with diagonal exposures. The result of the test print is shown below in Figure 7.1-16. Here, the print started off successfully, as seen in Panel (a). The bottom part of the square is where the print started, and it began well. However, as the print progressed, resonance between the two movement arms in the x-y track system increase in severity. As the square was half finished, the resonant motions caused the print to stall along an upper diagonal, causing severe melting. Panel (b) shows the print removed from the print bed. This resonance can be fixed by mechanically strengthening the x-y scanning system with further supports.



(a)



(b)

**Figure 7.1-16 High Speed Box Test Results**

## 7.2 Software Testing

The purpose of software testing is to ensure that all systems, as they relate to software, function properly, and to help ensure that software development proceeds smoothly throughout the duration of the project. As per Murphy's law, it can be assumed that issues will spring up within the software throughout the development process as the code base becomes more complex. In order to mitigate the effects of those issues, and prevent them from compounding one another, it is important to not only know about them quickly, but also know where they are and what they affect. In order to facilitate the continual development of software unmarred by such issues, a rigorous testing procedure will be applied throughout the process. First, as software is being developed and new features added, a parallel environment will be developed to ensure the proper functionality of those features. After a given software system becomes minimally feature complete, and again when it is fully feature complete, it will be tested with its associated hardware if any exist, and finally with any other systems it may interact with. Below are some examples of this process.

## 7.2.1 PlanerScan Testing

The testing for the PlanerScan class will take place in 3 phases: independent software testing, independent hardware testing, and unified hardware testing. The first phase, independent software testing, will consist of creating test cases to ensure the software performs as expected in different possible cases. After that, in the independent hardware testing phase, the code will be used on a prototype device with real hardware to test real world functionality. Finally, in the unified hardware testing phase, the PlanerScan class will be combined with the other physical control systems to test its ability to function in conjunction with the others.

For the independent software testing phase, a secondary class will be made in order to run the PlanerScan code in a simulated testing environment. This class will be able to call the various functions from the PlanerScan class and monitor the outputs to ensure they work as expected. The class will be used to implement a number of tests and run them far more efficiently than a human could, while also measuring the results far more accurately. The goal of this testing phase will be to test and verify the entirety of the PlanerScan classes functionality. This will include the basic ability to control stepper motors with a variable frequency square wave, as well as the more advanced control of that speed by precisely controlling the frequency of that square wave. The ability of the class to control motor acceleration will also be tested, along with higher level functions like controlling two motors in sync and using the two motors to draw simple shapes. In addition to checking proper motor control, the PlanerScan class will also be monitored to ensure it is raising the proper flags at the right times, and to measure its delay in coordinating with other classes. Once the PlanerScan class has been thoroughly validated in a digital space, physical testing can take place to ensure that the digital tests were accurate, as well as to help find further issues that may only come up in reality (and not an ideal simulated environment).

Once the independent software testing phase is complete, the data can be validated in the independent hardware testing phase. This phase will serve two purposes; because it isn't feasible to validate every aspect of the PlanerScan system with real hardware, we will instead validate the results of the independent software testing phase by checking a subset of those results and ensuring that they are correct within a certain margin of error, after that, knowing that the code works properly, we will move on to test how the software performs with real world limitations like motor power and torque requirements. In order to quickly validate the results of the independent software testing phase, we can set the PlanerScan class to draw some simple shape or shapes several times over and over again. This would allow us to measure how accurately the motors are being controlled based on how much variation there is between each layer of the drawing. If we time the drawing, we can also calculate and verify the drawing speed to check against the expected value from the previous testing phase. Once we know that the software works as expected, we can move on to testing how the hardware and general circumstances of the real world affect it. This would include testing the accuracy of the drawing system under different stresses like high resistance against the motor, voltage fluctuations, heat, and vibration. Once we know that the PlanerScan class works with its real-life components in the situations it may reasonably be subjected to, it can then move on to the unified hardware testing phase.

In the unified hardware testing phase, the fully validated PlanerScan class will be tested as it works with other classes like the ZSystem and LaserManager classes. At this point, not only the PlanerScan class, but also any other classes it is being tested with will have been tested to ensure proper functionality; this phase will be used to verify that all classes interact properly with one another and their respective hardware to perform the functions of a complete system. In order to test the PlanerScan class as a part of the overall system, we will complete a full or partial 3d print to determine that not only is the class moving the motors properly, but also that it is doing so in proper coordination with the laser and z axis as is needed for the project.

## 7.2.2 ZSystem Testing

Like the PlanerScan class, the testing for the ZSystem will also take place in 3 phases: independent software testing, independent hardware testing, and unified hardware testing. The first phase, independent software testing, will similarly consist of creating test cases to ensure the software performs as expected in different possible cases. After that, in the independent hardware testing phase, the code will be used on a prototype device with real hardware to test real world functionality. Finally, in the unified hardware testing phase, the ZSystem class will be combined with the other physical control systems to test its ability to function in conjunction with the others (namely, moving the print bed at the right time with respect to the sweeper mechanism).

For the independent software testing phase of the ZSystem, like with the PlanerScan class, a secondary class will be made in order to run the ZSystem code in a simulated testing environment. This class will be able to call the various functions from the ZSystem class and monitor the outputs to ensure they work as expected. The class will be used to implement a number of tests and run them much more efficiently than a human could, while also measuring the results far more accurately. The goal of this testing phase will be to test and verify the entirety of the ZSystem classes functionality. This will include the basic ability to control stepper motors with a variable frequency square wave, as well as the more advanced control of that speed by precisely controlling the frequency of that square wave. The ability of the ZSystem to move the build plates by a precise increment is of vital importance to the printer's function, so that will also be a large focus of this testing phase; not only will the frequency of the square wave be monitored, but also the precise number of wavelengths produced. In addition to checking proper motor control, the ZSystem class will also be monitored to ensure it is raising the proper flags at the right times, and to measure its delay in coordinating with other classes. Once the ZSystem class has been thoroughly vetted in a digital space, physical testing can take place to ensure that the digital tests were accurate, as well as to help find further issues that may only come up in reality (like making sure that the precise movements of the bed don't change with the weight on the bed).

Once the independent software testing phase is complete, the data can be validated in the independent hardware testing phase. Like with the PlanerScan class, this phase will also serve the same two purposes; because it isn't feasible to validate every aspect of the ZSystem system with real hardware, we will instead validate the results of the independent

software testing phase by checking a subset of those results and ensuring that they are correct within a certain margin of error, after that, knowing that the code works properly, we will move on to test how the software performs with real world limitations like motor power and torque requirements. In order to quickly validate the results of the independent software testing phase, we can set the ZSystem class to move the beds some distances in either direction several times (this test could also be repeated several times with different movement sizes). This would allow us to measure how accurately the motors are being controlled based on how much the final location of the beds differs from what it should be (we can program a pattern where the beds end up back where they started, or some known distance away). If we time the movements, we can also calculate and verify the bed speed to check against the expected value from the previous testing phase and compare with how fast we want the bed to move (if it goes too fast, it may disturb the powder and cause issues with the print). Once we know that the software works as expected, we can move on to testing how the hardware and general circumstances of the real world affect it. This would include testing the accuracy of the bed movements under different stresses like high resistance against the motor, voltage fluctuations, heat, and vibration, as well as changing loads on each motor as the powder moves from the powder reservoir to the print bed. Once we know that the ZSystem class works with its real-life components in the situations it may reasonably be subjected to, it can then move on to the unified hardware testing phase.

In the unified hardware testing phase, the fully validated ZSystem class will be tested as it works with other classes like the PlanerScan and LaserManager classes as well as the Sweeper class. At this point, not only the ZSystem class, but also any other classes it is being tested with will have been tested to ensure proper functionality; this phase will be used to verify that all classes interact properly with one another and their respective hardware to perform the functions of a complete system. Unlike the PlanerScan class, in order to test the ZSystem as a part of the overall system, we will only need to test it with the sweeper mechanism to ensure that it can consistently move the right amount and hold itself in place the rest of the time. However, because we will need to use the ZSystem when testing the PlanerScan class, it will be more efficient to test both at the same time by unifying the test and just printing something.

### 7.2.3 LaserManager Testing

The testing for the LaserManager class will take place in 3 phases just like the other physical control systems: independent software testing, independent hardware testing, and unified hardware testing. The first phase, independent software testing, will consist of creating test cases to ensure the software performs as expected in different possible cases; this is especially important for the laser manager class because if something goes wrong with the control systems for a high-powered laser then things can be damaged, or worse, people can get hurt so every step of testing must be thoroughly and properly performed. After that, in the independent hardware testing phase, the code will be used on a prototype device with real hardware to test real world functionality; this will of course be performed in a safe and controlled environment and will help us better understand any potential hazards that may present themselves in the future. Finally, in the unified hardware testing



phase, the LaserManager class will be combined with the other physical control systems to test its ability to function in conjunction with the others.

Like with all software testing for this project, for the independent software testing phase of the LaserManager class, a secondary class will be made in order to run the LaserManager code in a simulated testing environment. This class will be able to call the various functions from the LaserManager class and monitor the outputs to ensure they work as expected. The class will be used to implement a number of tests and run them far more efficiently than a human could, while also measuring the results far more accurately. The goal of this testing phase will be to test and verify the entirety of the LaserManager classes functionality, both to ensure that it will work properly for printing and to make sure that it is safe for use with a real laser. This will include the basic ability to control the laser with a digital signal of either high or low, as well as the more advanced control of that laser's power using pulse width modulation. The ability of the class to control the laser's state at precise intervals will also be tested. In addition to checking proper laser control, the LaserManager class will also be monitored to ensure it is raising the proper flags at the right times, and to measure its delay in coordinating with other classes and turning off the laser when given a kill command. Once the LaserManager class has been thoroughly validated in a digital space, physical testing can take place to ensure that the digital tests were accurate, as well as to help find further issues that may only come up in reality (like variation in laser power, and the beam reflecting unexpectedly).

Once the independent software testing phase is complete, the data can be validated in the independent hardware testing phase. Unlike the other physical control systems which have a much broader feature set, and lower safety requirements, the LaserManager class will be retested in its entirety during the independent hardware testing phase, or at least as entirely as is possible (we cannot test the code's internal functionality in a hardware test without an oscilloscope and a massive, scaled version of the processor). In order to quickly validate the results of the independent software testing phase, while also ensuring feature complete testing, we can set the LaserManager class to power the laser on and off for various durations and in various intervals at different power levels using all of the API's available functions. This would allow us to measure how accurately the laser is being controlled based on how much variation there is between each beam's duration (and power), and what we are expecting. Once we know that the software works as expected, we can move on to testing how the hardware and general circumstances of the real world affect it. This would include testing the accuracy of the laser system under different stresses like voltage fluctuations, heat, and vibration. Once we know that the LaserManager class works with its real-life components in a safe and reliable manner, it can then move on to the unified hardware testing phase.

In the unified hardware testing phase, the fully validated LaserManager class will be tested as it works with other classes like the ZSystem and PlanerScan classes. At this point, not only the LaserManager class, but also any other classes it is being tested with will have been tested to ensure proper functionality; this phase will be used to verify that all classes interact properly with one another and their respective hardware to perform the functions of a complete system. In order to test the LaserManager class as a part of the overall system, we will complete a full or partial 3d print to determine that not only is the class powering

the laser properly, but also that it is doing so in proper coordination with the x, y, and z axes as is needed for the project. This testing can be done separately from the ZSystem by simply using the PlanerScan and LaserManager classes as a laser engraver, however it would be more efficient to test the ZSystem at the same time.

## 7.2.4 Final Software Testing

Due to the change in software design, testing plans had to be changed as well. Luckily, one of the benefits of using open-source software is that it is generally well tested, and in this case we already knew that the firmware worked, and even that it worked with our hardware as we decided to use an officially supported chip set. That being the case our initial software testing consisted of connecting to the printer via the host server and moving motors through the user interface. This showed that our software was working properly, and could communicate with the computer, interpret g-code, and control physical components. After the initial testing and subsequent modifications, some intermediary tests took place involving controlling the various subsystems that were added to ensure they worked as expected, and that nothing else had broken in the process. Finally, with everything working we could do a full print to test the complete functionality of the printer.

## 8. Administrative Content

### 8.1 Project Milestone

The SLS printer project is a two-semester project. The first semester primarily consists of research, design, and documentation. The project timeline for Senior Design 1 is shown in Table 8.1-1. The second semester consists of prototype construction, redesign, and project submittal. The project timeline for Senior Design 2 is shown in Table 8.1-2. The dates for the second semester are still to be determined.

**Table 8.1-1: Senior Design 1 Milestones**

<b>Milestone Description</b>	<b>Type</b>	<b>Start Date</b>	<b>Due Date</b>
Project Idea	Action	8/27/2021	9/10/2021
Divide and Conquer Initial Documentation	Deliverable		9/17/2021
Research	Action	9/17/2021	10/1/2021
Divide and Conquer V2	Deliverable		10/1/2021
Design / Part Selection	Action	10/1/2021	11/5/2021
Senior Design I Documentation - 60 Page Draft	Deliverable		11/5/2021
Senior Design I Documentation - 100 Page Draft	Deliverable		11/19/2021
Design / Prototype	Action	11/19/2021	12/7/2021
Senior Design I Documentation - Final Documentation	Deliverable		12/7/2021

**Table 8.1-2: Senior Design 2 Milestones**

Milestone Description	Type	Start Date	Due Date
Prototype	Action	03/15	04/11
Test and Redesign	Action	04/11	04/15
Final Prototype	Action	04/15	04/17
Peer Presentation	Deliverable		04/19
Final Report	Deliverable		04/26
Final Presentation	Deliverable		04/19

## 8.2 Budget and Finance Discussion

The SLS 3D printer will be fully funded by the team. The team member who wishes to keep the project after completion will contribute the most. As stated in Table 8.2-1, our estimated budget for the SLS printer is around \$600. The total cost of the SLS 3D printer ended up being \$735.38. We went \$135 over budget due to unexpected costs that occurred during prototype manufacturing. Although the team went over budget, we still achieved a cost that is 4% the cost of the cheapest SLS 3D printer on the market.

**Table 8.2-1: SLS 3D Printer Budget and Total Cost**

Sub Assembly	Cost
Mechanical Assembly	\$182.67
Laser Module	\$214.74
Power Components	\$104.98
Computing Components	\$26.96
Motor Driver Components	\$64.35
Laser Driver Components	\$6.73
PCB Cost	\$68.97
Back-up Laser Module & Power Supply	\$65.98
<b>Actual Total</b>	<b>\$735.38</b>
<b>Projected Total</b>	<b>\$600</b>

## 8.3 User Instruction Manual

### 8.3.1 Software Setup

In order to use the 3D printer, the first step is to ensure all drivers are installed to communicate with the printer. The printer uses a CP2102 USB to UART bridge for programming and communication, the drivers for this chip can be found here: <https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers>, or by searching “cp2102 drivers” in the users favorite internet search engine. Once the correct drivers are installed, the printer will connect to the computer when plugged in via a virtual com port or VCP which can be found in the device manager, or any com port serial display. With the printer able to connect, the printer host/server software (available here: <https://www.repetier.com/download-now/>) can be used to interface with the printer for printing, experimentation, or other uses. In the server frontend, a new printer can be added by using the “Add new Printer” option in the dropdown from the plus button in the top right corner. By following the printer setup wizard, the user can name the printer, select the com port or VCP the printer is using, and adjust many other settings to their liking.

### 8.3.2 Print File Setup

Like with any 3D printer, in order to print, a 3D model must first be made and sliced into a series of 2D layers to be rendered in g-code. Once the model is made (using any 3D CAD software) or downloaded, any standard slicer can convert it into g-code; because most available slicers are designed for FDM printers instead of SLS, the g-code can be further processed with the available “Post\_Processing.py” script to make it more compatible and add in some customizations like a starting reservoir height to reduce excess material use. With the g-code made and processed, the file can be uploaded to the print server to be used later.

### 8.3.3 Printer Setup and Use

Before a print can start, the printer must be calibrated and loaded with material. Using the console, and the calibration sequence found in “Calibration\_Sequence.txt”, the printer can find the appropriate locations for all axes of movement and bring the reservoir to a desired height so as to not be filled with excess powder. With the printer calibrated, and the reservoir filled, the print can be started and left to run until completion when the part can be removed from the print bed.

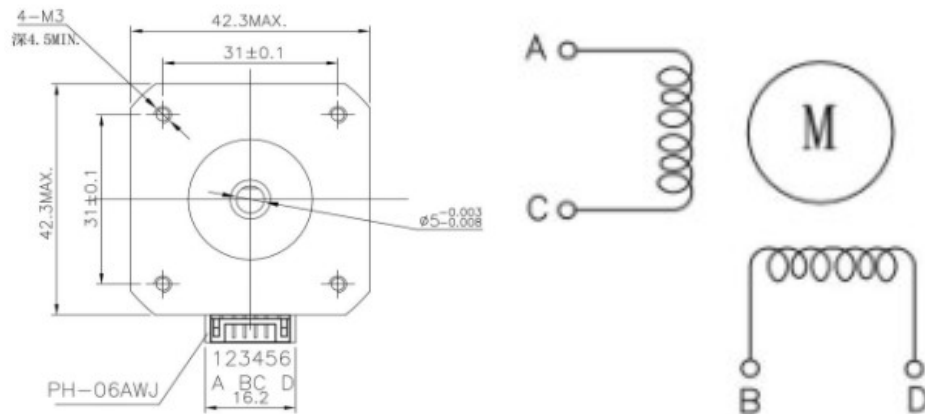
*This page intentionally left blank.*

## 9. Appendix

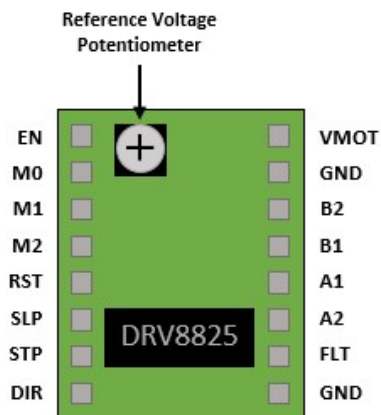
### 9.1 Appendix A – Copyright Permissions

Permissions for copyright content have been requested. Approval and/or denial will be detailed here upon receipt.

### 9.2 Appendix B – Miscellaneous



**Figure 9.2-1: Stepper Motor Schematic**



**Figure 9.2-2: DRV8825 Test Module**

## 9.3 Appendix C – References

- [1] M. Brandt, Laser Additive Manufacturing (no. 88). Woodhead Publishing, 2017, p. 475.
- [2] P. Linear. "Belt Drive or Lead Screw?" <https://www.pbcllinear.com/Blog/2020/February/Lead-Screw-or-Belt-Drives> (accessed October, 2021).
- [3] S. Studio. "Choosing the Right Motor For Your Project - DC vs Stepper vs Servo Motors." <https://www.seeedstudio.com/blog/2019/04/01/choosing-the-right-motor-for-your-project-dc-vs-stepper-vs-servo-motors/> (accessed June, 2021).
- [4] Benchmarks. "Which Programming Language Is Fastest?" <https://benchmarksgame-team.pages.debian.net/benchmarksgame/> (accessed October, 2021).
- [5] NEMA. <https://www.nema.org/> (accessed October, 2021).
- [6] ASME. "Acme Screw Threads." <https://www.asme.org/codes-standards/findcodes-standards/b1-5-acme-screw-threads> (accessed October, 2021).
- [7] NASA, NASA Systems Engineering Handbook. Createspace, 2017.
- [8] "DRV8825 Stepper Motor Controller IC Data Sheet," ed, 2021.
- [9] Makerguides. "Stepper Motor with DRV8825 and Arduino Tutorial (4 Examples)." <https://www.makerguides.com/drv8825-stepper-motor-driver-arduinotutorial/> (accessed September, 2021).