Department of Biomedical Engineering
BME706 Master's Design Report
Spring 2016

Design of an Open-Source Bioprinter

Brad Bradshaw

Mentor: Dr. Ramon Montero

Institutions: University of Miami

Date: 4/26/2016

Table of Contents

1. Abstract

This report details the development of a low-cost and highly flexible bioprinter suitable for research use. By leveraging an array of open-source technologies, this machine manages the tradeoffs between print resolution, overall cost, and material flexibility to enable fabrication of a wide variety of biologic constructs. The basis for the design centers on a RepRap-based FDM printer for the control of motion in three dimensions to an accuracy of less than 300 microns. Biomaterial extrusion occurs using an open-source syringe pump powered by a stepper motor, which is able to handle viscous biomaterials to an accuracy of less than 3%. Distributed heating is provided along the length of the syringe pump and fluid handling elements, providing thermal control of the chosen biomaterial during printing. Efforts to design a cooling system for the print bed to encourage rapid biomaterial gelation were eventually unsuccessful due to excessive cost. Upon integration of these systems, calibration of fluidic components, and optimization of software parameters, fully three-dimensional constructs were printed in a representative hydrogel with an accuracy less than 300 microns.

2. Objectives

2.1 General Objective

The purpose of this design document is to outline the steps taken toward the development of a fully open-source cell-compatible bioprinting system. Briefly, this report will detail the methodology behind hardware selection, in combining systems, in optimizing software and firmware parameters, and in testing under different relevant conditions. Bioprinting is a key platform technology in the biofabrication workflow, complementing other rapid prototyping, electrospinning, and bioreactor systems. However, currently available commercial products that fill this need cost more than $200,000USD. This work seeks to provide a lower-cost solution by leveraging open-source techniques such that bioprinting technology becomes widely attainable.

2.1.1 *Observation*

The precise localization of biomaterials, often described as hydrogels, is essential to the formation of a biologically relevant cellular system. For the induction of paracrine signaling, integrin-mediated communication, vasculogenesis, and defined mechanotransduction, high-resolution patterning of cells and extracellular matrix is needed in three dimensions. Such a

technique to place material at the requisite scale is often impossible manually; even when possible, it lacks the precision and reproducibility needed for scientific replication. Emerging fields that utilize cell-biomaterial systems for therapy (regenerative medicine)[1], modeling (stem cell organogenesis)[2], or *in vitro* diagnostics (organ-on-a-chip technologies)[3] absolutely need a high degree of precision and reproducibility when producing cell-matrix constructs.

2.1.2 *Problem*

The problems inherent to manual patterning of biomaterials, resolution and reproducibility, are best addressed through flexible automation. A programmable robotic system that can be tailored for different cell-material combinations would provide the highest utility. Such solutions currently exist commercially, but they cost more than $200,000USD.[4,5] Thus, a large gap exists at the lower end of the market, making these technologies impossible for the average-sized laboratory or biotechnology startup to access.

2.1.2 *Needs*

Briefly, the problems inherent to manual biomaterial construct fabrication are the lack of resolution and reproducibility along with a high labor cost. Currently available automated systems offer robust features but come with a very high price.

3. Background

Before delving into the specifics of bioprinting, it is first important to understand how this technology fits within the larger field of biofabrication. Biofabrication can be defined as "*the production of complex living and non-living biological products from raw materials such as living cells, molecules, extracellular matrices, and biomaterials.*"[6] While most people think about artificial organs, these biological products have applications spanning basic scientific research[7], pharmaceuticals development[8], energy[9], and agriculture.[10]

> *The broad spectrum of potential applications and rapidly growing arsenal of biofabrication methods strongly suggests that biofabrication can become a dominant technological platform and new paradigm for 21st century manufacturing.*[6]

From an even broader perspective, biofabrication techniques seek to turn traditional manufacturing strategies on their head. Devices that are intended to interface with living organisms must be designed with biology in mind. That is, everything from medical devices to biofuel plants to agricultural equipment must respect the constraints imposed by cells, tissues, organs, and organisms. The tools, techniques, and materials used to make these products are severely limited by their intended end use. Thus, design and manufacturing traditionally bow to biology. In contrast, biofabrication aims to turn that equation around, to leverage biology for design and manufacturing. By manipulating the synthetic capabilities of cellular systems, a new breed of products can be designed for a wide variety of applications.

Cellular bioprinting is just one element within the larger biofabrication workflow, as depicted in Figure A below. Technologies within this space differ with respect to the length scale of physical architecture they can manipulate and their capacity to handle sensitive biological specimens.

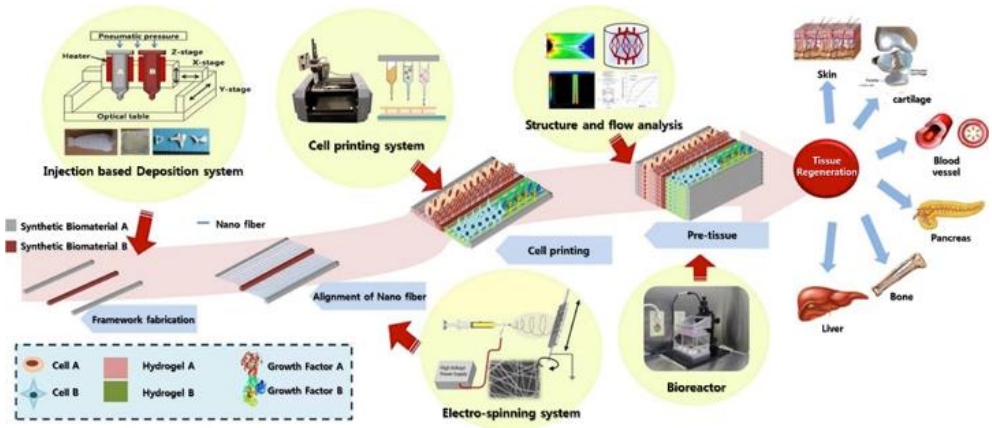


Figure A. Schematic depicting a biofabrication workflow. Multiple different technologies that act on various length scales are leveraged to produce a biological construct. The process consists of fabricating acellular materials with defined chemical and architectural features, the addition of living cellular elements in a highly controlled manner, and the conditioning of the neo-construct to reconstitute the form and function of an intended biological tissue.[11]

On the nano- end of the spectrum, technologies originally developed by the semiconductor industry, such as spin coating, photolithography, and etching, can be used to create physical architectures with features on a subcellular scale. However, these techniques require highly caustic reagents, expensive machinery, and cleanroom facilities, making them unsuitable for direct incorporation of biological entities. Often, these tools are used to create molds or stamps with defined nanotopography out of poly(dimethyl) siloxane (PDMS), a silicone-based elastomer, in a process known as soft lithography (Figure B).[12] Vapor deposition and electrospinning also manipulate materials at the nanoscale, but they offer a much lower degree of control on the resulting architecture.

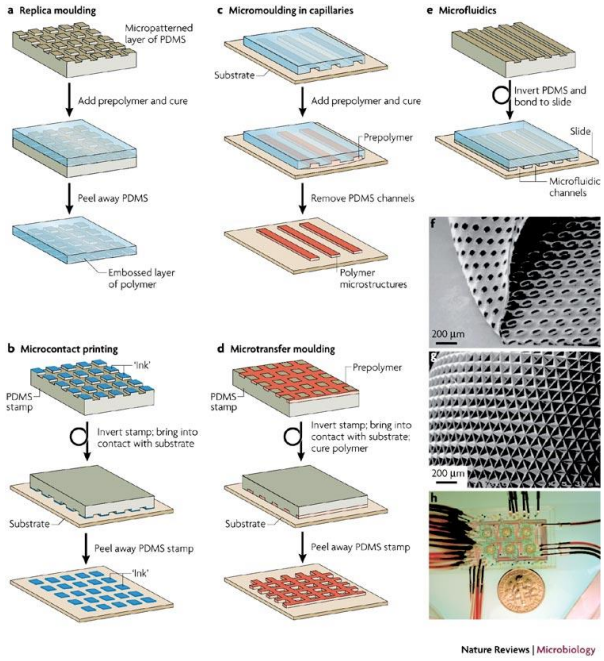


Figure B. PDMS-based microtechnology for biofabrication. Nanoscale features designed into a silicon wafer can be reconstituted in PDMS by replica molding. These features can be used to create open channels for microfluidic applications, to mold other polymers onto a surface, or to deposit "bio-inks" in a structured manner.[13]

Within the meso- range, a larger variety of techniques can be used to manipulate both biomaterials and cellular material. Traditional scaffold fabrication techniques, such as salt leaching and gas foaming, use sacrificial elements to build porous, acellular structures that can be subsequently loaded with biological material. When cast in a mold, these constructs can be precisely shaped; however, the stochastic nature of foaming or solute leaching does not permit significant control over pore formation. Modern meso-scale biofabrication techniques have evolved out of the rapid prototyping and printing industries. In most iterations, structures are built in a layer-by-layer fashion, with each two dimensional slice subsequently fused together. While thermoplastics are the most common substrates for typical 3D printers, their core technology has been increasingly repurposed to handle other materials, including viscous fluids and slurries.

To be realistically effective, biofabrication technologies must be flexible towards what they can produce. This flexibility can become manifest in several ways: machines may enable extremely high-throughput through automation, others may offer capabilities to use a wide swath of different biomaterials and chemistries, and still others may permit tuning of different parameters that result in a completely different kind of construct. As such, biofabrication machines should be thought of as "platform" technologies, upon which specific applications can be built. Tissue engineering provides a convenient example. The role of a tissue engineer is to design biological systems and then analyze their performance, often with the goal of developing models or mimics of real organs. This goal could never be realized without tools and techniques to physically construct biological systems and then to adjust that design following testing. Thus, biofabrication platform technologies speed the design-build-test cycle of tissue engineering, and are likely critical to the field's success.

One of the key barriers to the proliferation of biofabrication technologies throughout the research community has been their excessive cost. As detailed previously, much of the machinery involved has been co-opted from other industries, many of which are far more mature than biofabrication or tissue engineering. As such, the commercial forms of these technologies lie out of reach for the average academic research lab. There exists a great need for the development of low-cost solutions for the research community.

4. Design specifications

The MoSCoW method was used to identify the critical parameters for this design based on the specifications typical of the technologies to be leveraged (Table 1). Broadly speaking, these parameters describe a bioprinting device that is compatible with a semi-sterile, cell culture environment. They call for highly flexible design that will enable a variety of customizable bioprinting strategies, with room for as-yet-unknown modifications. Parameters that will limit all bioprinting workflows on a machine level (i.e. 3-axis resolution, build volume) are strictly defined to maximize utility. They exist as tradeoffs with total cost, which serves as the optimization criterion in this design.

MoSCoW Analysis

	"Acceptable"	"Desired"		
Parameter	**Must Have**	**Should Have**	**Could Have**	**Would like, Won't have**
Total Cost	$4,000 USD	$2,000 USD	$1,500 USD	$1,000 USD
X-Resolution	1 mm	500 um	200 um	100 um
Y-Resolution	1 mm	500 um	200 um	100 um
Z-Resolution	2 mm	1 mm	500 um	250 um
Footprint	1000x1000x2000 mm	1000x750x2000 mm	750x750x1500 mm	600x600x1000 mm
Multi-Resin Capability	1 resin at a time; interchangeable reservoirs	1 resin at a time; two powered pumps	2 simultaneous resins	3+ simultaneous resins
Build Volume	50 x 50 x 50 mm	100 x 100 x 50 mm	200 x 200 x 200 mm	400 x 400 x 200 mm
Resin Temp Range	RT - 37C	RT - 60C	RT - 100C	0C - 100C
Resin Temp Variance	< 15 C	< 10 C	< 5 C	< 2.5 C
Platform Temp Range	RT	4C - 35C	4C - 110C	-4C - 110C
Platform Temp Variance	N/A	< 15 C	< 10 C	< 5 C
Potential Chemistries	Temperature-responsive	Temp, ionic, pH, enzyme	Temp, ionic, pH, enzyme, sacrificial	Temp, ionic, pH, enzyme, light/laser, sacrificial
Cell Viability	>25%	>50%	>70%	>90%
Sterilization-capable	fluid handling, print head	fluid handling, print head, build platform	All except electronics	All including electronics

Table 1. MoSCoW Analysis of critical design features and desired parameters

5. Design constraints

The proposed device is not intended for clinical use; rather, it seeks to maximize utility in the laboratory or research and development (R&D) space. Therefore, the design will not meet current Good Manufacturing Practices (cGMP) for cell/combination product fabrication. In contrast, the primary constrain in this design is total cost, with the goal of maximizing access to bioprinting technology, especially for R&D groups without engineering expertise. In pursuit of developing a low-cost, high-flexibility device, the use of open-source technology and software is a key objective.

6. Design Development

The following sections detail the evolution of the bioprinter design.

6.1 *Information Gathering*

One of the benefits of basing a design on open-source technologies is the wealth of information available from a number of sources. For this bioprinter, the resources provided by the RepRap community project[14] and by the bioCurious Hackerspace[15] offered guidance toward selection and modification of fused deposition modeling (FDM) printers. More information on printer modification and hydrogel "ink" formulation is provided, open-source, by the Feinberg group at Carnegie Mellon[16,17] and the Angelini team at the University of Florida.[18,19] Details regarding projection photolithography were graciously provided by the Fang group at MIT[20,21] and Dr. Jordan Miller at Rice.[22]

6.2 *Generation of Alternatives*

Two alternatives were generated to meet the need for a low-cost, open-source bioprinting technology: projection stereolithography and extrusion printing.
In projection stereolithography (pSLA), patterned light is beamed onto a pool of liquid resin, causing it to selectively solidify (Figure C). The constituents of these resins vary substantially, but typically contain some kind of photoinitiator to induce polymerization and some kind of photoabsorber, such as a dye, to either quench polymerization and/or shift the adsorption spectrum peaks to maximize initiation. The polymer systems leveraged in these resins can either be of the synthetic variety, commonly (poly) ethylene glycol-diacrylate (PEGdA), or natural, such as acrylated hyaluronic acid (HA). Synthetic polymers have the advantage that their properties can be finely tuned for a given application. Additionally, functional moieties, such as adhesive peptide sequences, can be readily incorporated in a selective manner. Natural polymers offer less flexibility toward customization, but by their nature present both microarchitecture and biologically-active regions that are recognized by incorporated cells.

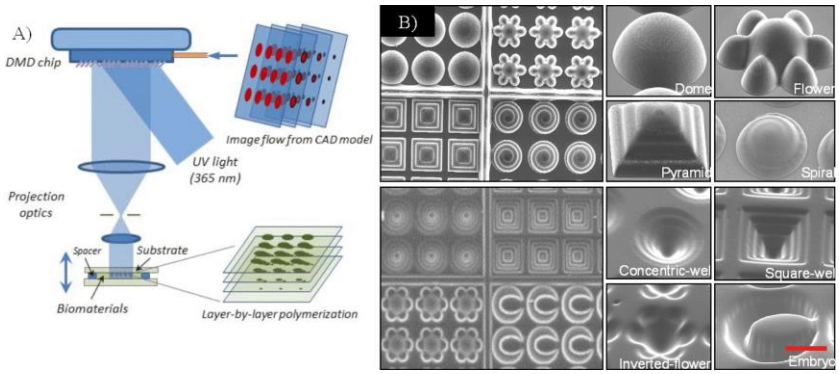


Figure C. Projection stereolithography involves directing light in the pattern of a photomask onto a pool of resin. Regions exposed to light undergo curing and selectively solidify. Complex structures are cured layer-by-layer with exquisite resolution.[23]

The resolution of pSLA depends ultimately on how precisely incident light can be directed. Classical SLA designs, also known as direct laser-writing systems, trace a finely focused laser across the surface of the resin to cure each layer. While highly precise, these systems are very slow; build time depends on laser raster speed, which is fundamentally limited by laser power. The projection component of pSLA refers to the use of a dynamic photomask that permits an entire layer to be cured at the same time. Most commonly, this photomask takes the shape of a particular microelectromechanical system (MEMS) called a digital micromirror device (DMD) (Figure D). A DMD chip consists of an array of microscale mirrors that are held in one of two states: $+12^{o}$ ("On") or -12^{o} ("Off"). Each mirror corresponds to a pixel in the projected image, where the "On" state will reflect incident light onto the resin, while the "Off" state will direct it away. In this manner, the image of an entire layer can be projected and cured simultaneously, dramatically increasing fabrication speed.

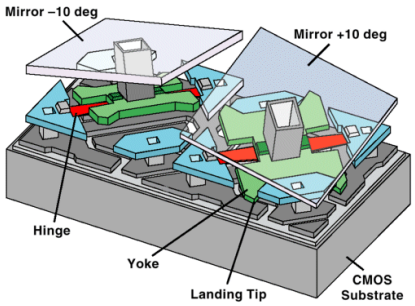


Figure D. The Digital Micromirror Device acts as a dynamic photomask in pSLA systems. Micromirrors representing individual pixels can be angled either toward or away from the resin bath, creating a binary map that corresponds with the image for each layer of the construct. This switch between "On" and "Off" states occurs within microseconds, enabling precise control over light exposure time.[24]

The second alternative investigated was based on an extrusion printing platform, commonly referred to as Fused Deposition Modeling (FDM). FDM printers exploit temperature-regulated changes in thermoplastic viscosity to build objects layer by layer (Figure E). A thermoplastic, most commonly (poly) lactic acid (PLA) or acrylonitrile butadiene styrene (ABS), is supplied to the printer as a thin, cylindrical filament. The filament is pinched by an extruder gear, which pulls the plastic at a controlled rate into a heated chamber called the hot end. Upon heating, the thermoplastic melts into a viscous liquid. As pressure builds up in the hot end due to the extruder gear forcing plastic into the chamber, the material will begin to press through a very small hole in the hot end, referred to as the nozzle. Outside the nozzle, the plastic rapidly cools and solidifies. This entire heating and extrusion assembly, known as the print head, is mounted above a three-axis stage. Together, the print head and the stage move to form each subsequent layer of the printed object.

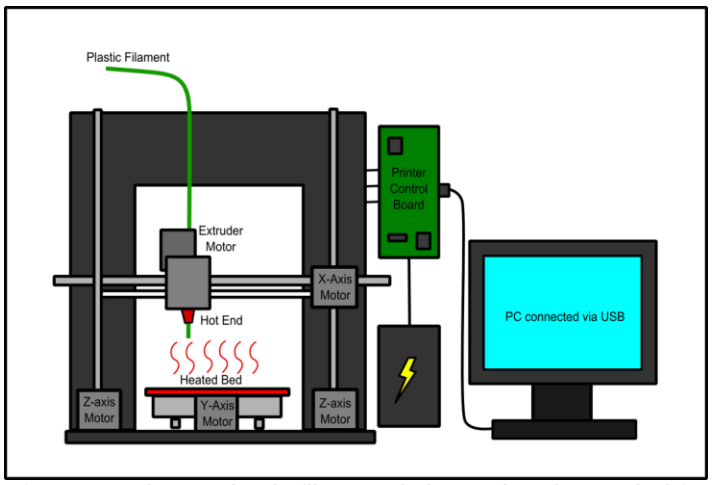


Figure E. A schematic of a FDM printer. Plastic filament is heated and extruded from the print head onto a heated bed. Precise control over plastic temperature ensures reliable extrusion and adhesion between build layers. The stage and print head share a 3-axis motion system, typically regulated with stepper motors. All functions, extrusion, heating, and motion, are controlled through a single board, which interfaces with a computer.[25]

FDM printers are not capable of bioprinting in their native form. While many synthetic and natural biopolymers have thermoresponsive properties, they do not exhibit the high viscosities needed for accurate extrusion. These polymers do liquefy at higher temperatures and gel upon cooling, however. Thus, only the print head assembly needs to be modified for bioprinting. This can be readily accomplished by replacing the extruder gear with an extrusion pump, by replacing the hot end nozzle with a dispensing needle, and by providing a distributed heating element for entire assembly.

6.3 *Alternative Selection*

Projection SLA and extrusion printing were compared according to three competing design factors: feature resolution, cytocompatibility, and overall cost. Feature resolution refers to the smallest element that can be reliably printed. Cytocompatibility refers to the capacity for the technology to preserve cellular viability during printing. Whereas incorporating cells post-fabrication nullifies this constraint, such an approach limits where the cells can be placed within the construct. Finally, the overall cost parameter takes into account the open-source goal of this design project. Results are summarized in Table 2 below. Based on its acceptable feature size, higher and more predictable cytocompatibility, and low cost, extrusion printing was selected for further development.

Alternative	**Feature Resolution**	**Cytocompatibility**	**Overall Cost**
pSLA	<100 um	Low	Estimated ~$5,500
Extrusion	<1000 um	High	Estimated ~$1,250

Table 2. Comparison of pSLA- and extrusion-based bioprinting technology

6.3.1 *Feature Resolution*

Feature resolution in pSLA is a complex function directly related to both projected DMD pixel size and polymerization kinetics. By projecting an image from the DMD chip through an objective lens, each of the micromirror pixels can illuminate an area of less than one square micron. Whereas light may beam on one square micron of the resin surface, the polymerization induced may spread outward. There does not yet exist a satisfactory computational model to account for this spreading; arduous trial-and-error testing is needed to validate any system. However, polymer spread is a function of initiator efficiency, exposure dose, quenching efficiency, and monomer diffusivity in solution. A conservative estimate for feature resolution in a low-cost pSLA system is less than 100 microns.

Feature resolution in extrusion printing is wholly defined by the bead of polymer formed at the print head. For thermoplastics, the length-width spread of this bead is practically limited to 110% of the hot end nozzle. That is, for a nozzle that is 0.4 microns in diameter, the smallest feature in the X-Y plane would be 0.44 microns. This spreading occurs because the hot end assembly pushes the thermoplastic bead into the layer of plastic directly beneath it. This is necessary to ensure reliable layer adhesion. When modified with an extrusion pump, printed feature resolution becomes significantly more challenging. First, the extruded biomaterial must maintain its shape, which means it must immediately gel. In the case of thermoresponsive biopolymers, this mandates highly precise temperature control at the print head. Secondly, the biomaterial must extrude in a laminar fashion, rather than form droplets at the print head. Droplets form when cohesive forces exceed adhesive forces between the biomaterial and the printing surface. Droplet formation is inversely related to viscosity for most biomaterials in this scenario. A conservative estimate for feature resolution in a low-cost extrusion printing system is less than 1000 microns.

6.3.2 *Cytocompatibility*

pSLA technology is significantly more cytotoxic than extrusion printing. The active reagents in pSLA resin, photoinitiators and quenchers, are typically toxic at high concentrations. Thus high-efficiency species at low concentrations must be used. High-efficiency photoinitiators, such as Irgacure 2959, are maximally activated with incident light within the ultraviolet (UV) range. These wavelengths are highly cytotoxic, inducing severe DNA damage by thymine dimerization. Secondly, the mechanism of SLA polymerization can chemically damage cell membranes. Briefly, photoinitiators induce polymerization by splitting into radical species, which donate their extra electrons to neighboring monomers. These monomers, typically acrylates, then bind each other until the reaction is quenched by a scavenging molecule or by the addition of two radicalized monomers. During this polymerization, these reactive species can bind to and damage the phospholipid bilayer that forms cell membranes, particularly along transmembrane receptors.

Cytotoxicity in extrusion printing arises only due to temperature and shear forces. Temperature only becomes an issue at the print head, when the biopolymer is

heated to its glass transition temperature. Most thermoresponsive biopolymers liquefy at very moderate temperatures (<40°C), which is tolerated by cells for a short period of time. Shear is introduced by the flow of viscous polymer through the print head. Shear can be minimized by ensuring that the fluidic path between the biopolymer reservoir and print head is significantly larger than a cell diameter.

6.3.3 *Overall Cost*

Initial design investigation into pSLA revealed the need for a dynamic photomask to cure UV-responsive biopolymers, which provides better flexibility from a cost and production time perspective than laser writing-based techniques. The Texas Instruments DLP® LightCrafter™ evaluation module provides the only accessible means to construct such a dynamic UV photomask by using their proprietary DMD chipset. However, the cost of this device (>$4,000), along with the optical apparatus needed for its integration, exceeded the bounds of this design project's budget. In contrast, the equipment needed to build an extrusion platform are significantly less costly. Many of the components are available open-source already, aligning with the goals of this design project.

6.4 *Conceptual Design*

Design of the bioprinter can be broadly divided into four complementary systems: the 3D printer, the software and firmware, the extrusion pump, and thermoregulatory components. The conceptual details of each will be explored in turn and are depicted in Figure F below. As a global overview, the extrusion pump will be modified to respond to extruder motor commands from the embedded printer firmware. The printer software and firmware will be modified to control the new constraints presented by the extrusion system. Finally, two parallel thermoregulatory systems will control the temperature of both the extruder and the build platform, which offers significant flexibility toward biopolymer choice.

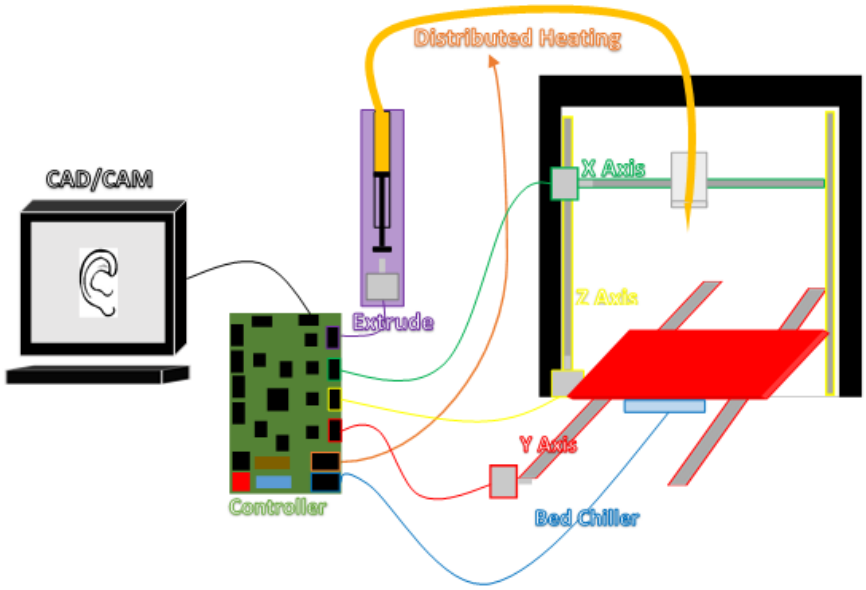


Figure F. Conceptual diagram of extrusion bioprinter, including three-axis stage, extrusion pump, thermoregulatory systems, control board with embedded firmware, and master computer feeding CAD models sliced into G-code.

6.4.1 3D Printer

The printer constructed in this design is based on the Prusa i3 model (Figure G) originally designed by Josef Prusa, a RepRap core developer. This iteration was selected due to its very low component cost (less that $600 USD) and its flexibility, as evidenced by the numerous variant designs documented on the RepRap Wiki.[26]

Figure G. The Prusa i3 3D printer, originally designed by Josef Prusa.[26]

The frame of the printer is constructed from milled aluminum and supported by a series of M8 rods to form the x- and y- axes. Two M5 threaded rods actuate the z-axis. All joints and connectors are 3D printed in poly(lactic) acid (PLA) (Figure H).

Figure H. 3D printed parts for the Prusa i3 model. Source files can be found on Thingiverse.[27]

4 NEMA 17 stepper motors, one each for the x- and y-axes and two for the z-axis, regulate positioning in three dimensions. GT2 timing belts and pulleys link motor control to axis motion. Three mechanical endstops are fitted and calibrated to the "home" position of the three axes (Figure I). These microswitches enable the device to auto-calibrate before each printing operation and remain accurate to within its tolerances.

Figure I. (Left) NEMA 17 stepper motors are used to actuate the 3 axes and operate the extruder. (Middle) GT2 timing belts and pulleys link motor action to axis motion. (Right) mechanical microswitches define the lower boundary of axis motion and define the "home" position. The upper boundary is defined in firmware.[26]

The extruder assembly is powered by a fifth NEMA17 motor, which pulls plastic filament down by pinching with an idler. The extruder body is composed of a hot and cold end, thermally separated from one another with a poly(ether) ether ketone (PEEK) plastic insulator and a poly(tetrafluoroethylene) (PTFE) non-stick liner. A diagram of a typical extruder assembly is provided in Figure J below. This Prusai3 uses a J-Head Mk VI nozzle, which comes with an attached 6 Ohm (Ω) power resistor for heating and a 100 kΩ thermistor for temperature measurement.

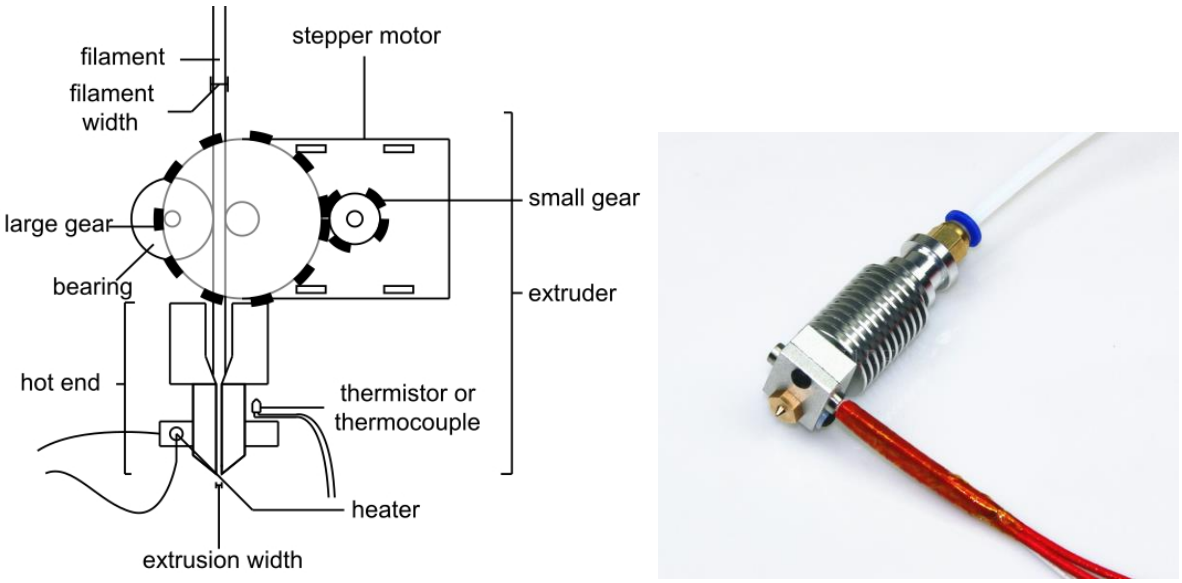


Figure J. (Left) Diagram of a typical extruder assembly. A stepper motor-idler system pulls solid plastic filament down toward the hot end of the extruder. A power resister-thermistor temperature regulation system maintains a constant melting temperature to extrude plastic through the nozzle tip. (Right) The J-Head Mk VI extruder nozzle with attached thermistor and power resistor. PEEK insulator (blue) and PTFE liner (white) separate the two sides of the extruder and prevent clogging.[26]

This Prusai3 also features a heated printing bed, enabling the printing of a wider variety of plastic types (Figure K). Much the same as the nozzle, the bed features an evenly distributed power resistor to provide Joule heating and an attached thermistor for temperature measurement.

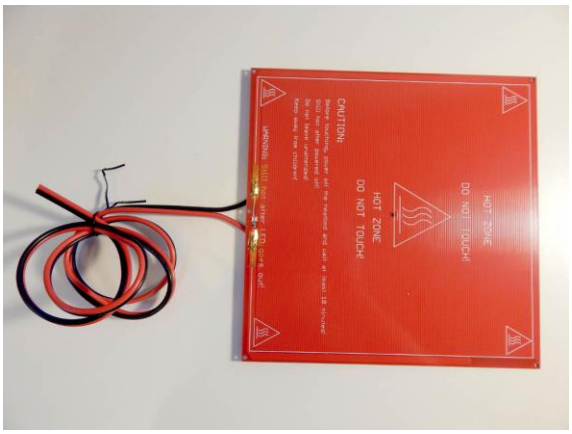


Figure K. Resistive heating build platform maintains plasticity of extruded filamentum. Under proper temperature conditions, each printed layer will adhere to the last, forming a monolithic construct.[26]

All of these components are powered and managed by a single board, the Mini-RAMBo from Ultimachine (Figure L). This controller integrates A4982 stepper drivers to control the motors, Atmega2560 and 32u2 processers for logic, jacks for the thermistors and switches, and pulse width modulation (PWM) MOSFET outputs for high voltage heating. Most importantly, this board, and its accompanying firmware, was designed open-source for easy modification.

Figure L. The Mini-RAMBo board by Ultimachine.[28,29]

6.4.2 Software/Firmware

While the hardware side of many 3D printers is readily amenable to modification and improvement, it is the supported and embedded code that truly enables design flexibility. Four general classes of code are needed to operate a typical 3D printer: computer-assisted design (CAD) software, G-code generators, G-code senders, and firmware.

6.4.2.1 CAD Software

CAD software, such as SolidWorks™, AutoCAD™ Inventor, and OpenSCAD (open-source)[30], enable the creation of three-dimensional constructs as parametric STL models. This standardized file system enables communication between systems and is the standard by which models are shared among designers.

6.4.2.2 G-Code Generator

In order to actually print the designed model, two functions must occur. First, the construct must be sliced in a planar fashion along its z-axis. Then, these flat planes must be converted into files that describe the motion and extrusion of the printing heat, called G-code. Software packages that perform this function are known as G-code generators or slicers (Figure M). Most importantly, these programs make it simple to optimize parameters like print speed, extrusion rate, and infill pattern without having to modify any hardware. For this design, the open-source Slic3r program is used.[31]

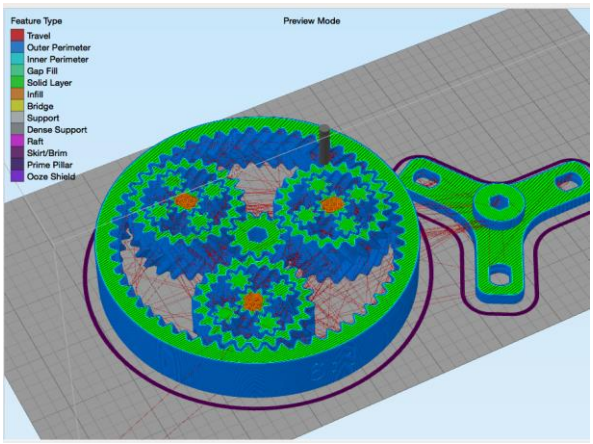


Figure M. G-code generators, or slicers, segment parametric STL files into individual layers. They then define the features of those layers as they correspond to the print: outer and inner perimeters are defined, followed by infill regions, bridges, and support. Slicers allow the user to control the flavor of G-code generated by manipulating the speed of the print head, the size of the nozzle or filament, the number of perimeters or borders, and more. Using these parameters, the slicer then outputs a set of tool path and extrusion instructions that can be interpreted by the 3D printers control board.[31]

6.4.2.3 G-Code Sender

Once G-code files have actually been generated for printing, they must then be communicated to the printer board. Software that serves this purpose is known as a G-code sender, but is more commonly referred to simply as "control software". Communication occurs either through a serial port on the board or via a SD card. Not only does the control software communicate G-code, but it is also used to turn on and off the different hardware components for testing and calibration. In this work, the open-source Pronterface software is used[32], whose interface is depicted in Figure N, below.

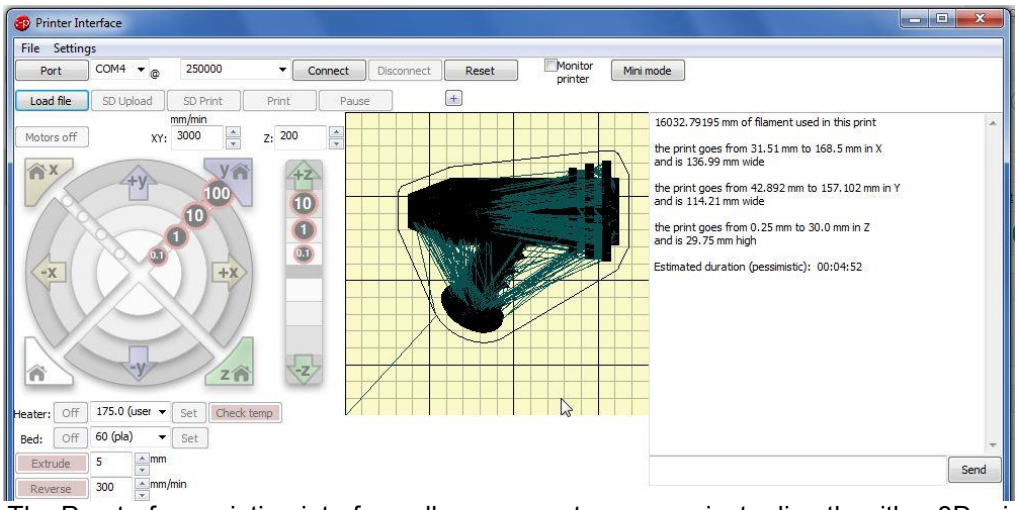


Figure N. The Pronterface printing interface allows a user to communicate directly with a 3D printer's control board. G-code can be communicated to the control board over a serial bus and the program will track the print progress dynamically. The Pronterface interface makes it simple to send basic movement and extrusion commands – which is very useful during testing. In addition, G-code can be written and sent directly over the serial monitor using the monitor panel on the right.[32]

6.4.2.4 Firmware

Now that G-code has been parsed and communicated to the board, the microprocessor must activate the stepper drivers in the proper order, while managing the power output through the MOSFETs. This is accomplished through firmware that is compiled and flashed onto the board via the Arduino™ integrated development environment (IDE). In this work, the open-source Marlin firmware is used.[33,34] The modified firmware developed in this work is presented in Appendix A10.

6.4.3 Extrusion Pump

The extrusion pump system is designed according to three primary criteria: overall cost, ability to integrate with 3D printer, and fluid dispensing accuracy. These parameters exist as tradeoffs and inform development. Integration with the 3D printer mandates operation using a stepper motor, controlled from the "extruder" A4982 stepper driver. A smaller device footprint will decrease the weight of the print head, which will speed printing and reduce inertia during axis movement. Two different pumping technologies were investigated for use in the bioprinter: peristaltic and syringe pumps (Figure O). Briefly, peristaltic pumps operate using a rotating cam that squeezes fluid through a conduit, while a syringe pump uses a plunger to press fluid through a needle tip. The pros and cons of each system are documented in Table 3 below.

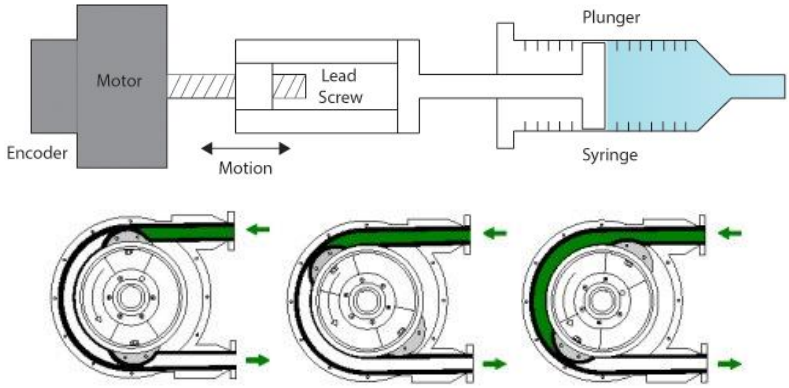


Figure O. (Top) Diagram of a simple syringe pump. A motor powers a linear actuator using a lead screw-capture nut carriage. Fluid is expelled by pushing a plunger through a syringe.[35] (Bottom) Diagram of a two-cam peristaltic pump. A motor powers a rotating cam that squeezes fluid through a conduit.[36]

Pump Comparison

	Syringe Pump	Peristaltic Pump
Pro	Highest precision delivery Pulse-free flow Low reservoir volume Well-developed open-source model Compatible with stepper motor	Limits fluid pressure Less expensive Compatible with stepper motor Smaller footprint
Con	High fluid pressure More expensive Larger footprint	Lower precision delivery Pulsatile flow High reservoir volume Nascent open-source model

Table 3. Comparison of pump technologies.

Based on the comparison above, the syringe pump system was selected due to its higher precision and pulse-free flow characteristics. Further, a recently published report from the Pearce group at Michigan Technological University[37] provides an open-source design for a compatible syringe pump (Figure P).[38]

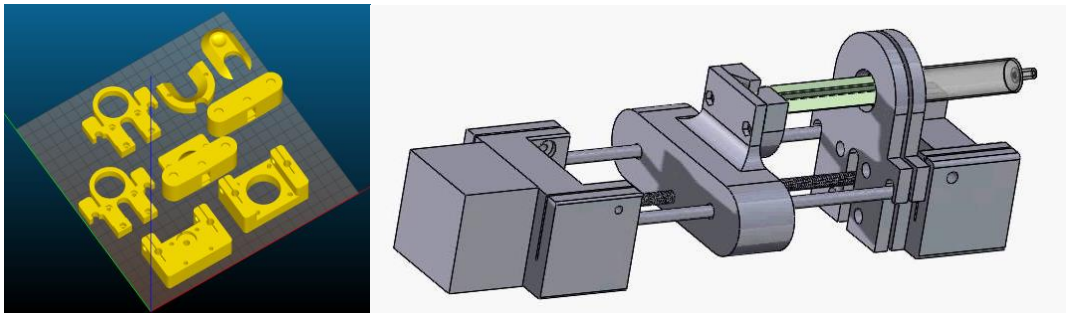


Figure P. Open-source syringe pump. (Left) All of the 3D printed parts needed for assembly (Right) Model of assembled syringe pump with loaded syringe.

For this syringe pump to be compatible with the 3D printer, it is important that the stepper motor characteristics match those of the drivers on the RAMBo board. A simple solution would be to extract the extruder motor from the extrusion assembly and repurpose it in the syringe; however, this design seeks to permit the printer to readily "switch back" into FDM operation with minimal hardware and software changes. Therefore, a compatible stepper motor must be sourced. The A4982 stepper driver can provide up to 2 amps (A) with a realistic voltage up to 12 V; therefore, a compatible motor must draw less than 2 A/phase at a resistance of 6 Ω/phase (Figure Q).

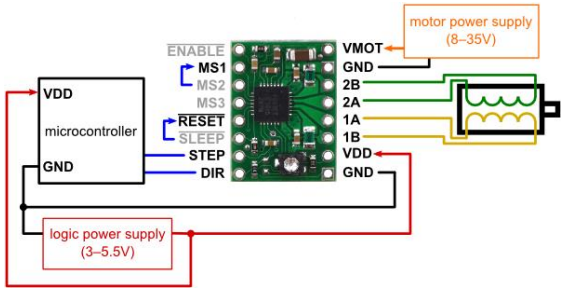


Figure Q. The A4982 stepper driver integrated onto the Mini-RAMBo board. The driver provides up to 2A at 12V, requiring a stepper that draws less than 2 A/phase at a resistance of 6 Ω/phase.[28]

6.4.4 Thermoregulation

Thermoregulation of the extrusion system and the print bed is needed to ensure that the biomaterial "ink" remains compliant upon printing, yet solidifies rapidly afterward. It is worth noting that many bioprinting material strategies do not rely on temperature gradients to induce gelation. Great examples from the Feinberg group include alginate-Ca^{2+}, collagen-pH, and fibrin-thrombin systems. CITATION However, when cells are incorporated into the mix, temperature management becomes essential to preserve viability.

Temperature control of the print bed is halfway solved in a normal 3D printer. In order for sequential layers of deposited plastic to solidify together, the temperature of the bed is heated to preserve malleability. While this feature is great for warming the bed, the resistive heating system employed offers no capability to cool the surface below room temperature. Many biomaterials, especially gelatin and other engineered thermoresponsive polymers, gel at below room temperature. Therefore, a second system is needed to act as a heat pump.

An efficient means to accomplish this is through the use of a Peltier element, which acts as a thermoelectric cooler (TEC). By applying a voltage across the element, heat can be removed from the print bed for dispersal through a heat exchanger. Figure R demonstrates an excellent example of a TEC assembly. Regulation of the voltage applied to the TEC enables tight temperature control, assuming that the ability of the heat exchanger to remove heat is adequate. By measuring the bed temperature with a second thermistor, a proportional-integral-derivative (PID) controller can be optimized to power the TEC through a high voltage relay. Figures R demonstrates these components.

Figure R. Printing bed cooling system components, supplied by Adafruit®. (Left) 12V, 5A TEC assembly with attached heatsink.[39] (Middle) 1%, 10kΩ thermistor for temperature measurement.[40] (Right) Powerswitch Tail 2 high relay for the control of high voltages with a tiny current draw.[41]

Temperature control of the syringe pump system cannot leverage the hot end from the extrusion assembly. Unlike the plastic filament, the entire volume of "bioink" needs to be maintained at a constant temperature before printing, so a much larger heating system is needed. While the design of a second thermoregulatory circuit using a thermistor and flexible resistive heating pad wrapped around the syringe itself would be sufficient, such a device already exists! New Era Pump Systems™ has already developed a syringe-heating system complete with properly-shaped resistive heating pads and a built in controller (Figure S).[42] This system does come with a hefty cost, however, and could be replaced with an open-source solution instead.

Figure S. New Era Pump Systems™ Thermo-Kinetic Heater for syringe pump.[42]

6.5 *Implementation Plan*

The schedule for design, sourcing, construction, and testing is detailed in the Gantt chart in Appendix A1. A budget is presented in Appendix A2, including details about component sourcing.

6.6 *Testing Plan*

Testing occurs according to a four step sequence. First, each of the individual components of the system is tested in isolation under ideal conditions. This serves as a quality check for the supplied components and allows the tester to work out any problems associated with assembly. For testing purposes, the components used in the design include a 3-axis stage, syringe pump, syringe heater, bed chiller, print head (nozzle) assembly, control board (firmware), and computer (software). Second, these components will be tested again in isolation, but under simulated real conditions. This serves as component qualification, setting the upper bounds for performance in verification testing. Third, the components of the system will be integrated piece by piece and tested under real conditions. This intermediate step is intended to uncover problems that may become obfuscated when more parts are in action and constitutes verification testing in this report. Finally, the entire integrated system is validated. A summary of the testing plan is provided below in Table 4.

Testing Matrix							
		System Components					
		3-Axis Stage	**Print Head / Nozzle Assembly**	**Syringe Pump**	**Control Board / Firmware**	**Syringe Heater**	**Bed Chiller**
Isolated Component Qualification	**Ideal Conditions**	Proof of Concept 1		Proof of Concept 3		Proof of Concept 4	Proof of Concept 6
	Real Conditions	Proof of Concept 2		Design Verification 2		Proof of Concept 5	Proof of Concept 7
Integrated Component Testing		Design Verification 1		Proof of Concept 8			
				Design Verification 3			
				Design Verification 4			
		Design Verification 5					
		Design Validation					

Table 4. Testing Matrix summarizes and nature, purpose, and order of the different tests.

6.6.1 Isolated Component Testing

Not all of the system components can or need to be tested in isolation. The print head (nozzle) assembly, microcontroller (firmware), and computer (software) have been excluded. Their qualification and verification will occur during integrated component testing.

6.6.1.1 *3-Axis Stage – Ideal Conditions*

The resolution of the three independent axes is determined by measuring the deviation from a set program of movements. Stepwise movement commands are sent via G-code using Pronterface. Measurements are performed manually using calipers by marking the movement path using a tracer. X- and Y-direction movements are tracked by binding a permanent marker to the print head. As the head tracks across the XY-plane, lines denoting its motion are drawn on a sheet of paper clipped onto the build plate. Similarly, Z-direction movements are tracked by binding a permanent marker to the print head in a direction parallel to the build plate. A sheet of paper is clipped to the printer frame, parallel to the Z-axis. As the print head completes each movement along the Z-axis, the marker can be pressed into the paper to mark its stopping point. For all three axes, 10 mm movements were repeated 50 times and measured manually. Accuracy is expressed both as the average absolute error and the average percent error in length:

$$|\ Avg\ Error\ | = \left|\ Measured\ Distance\ (mm) - 10\ mm\ \right|$$
$$\%\ Error = |\ Avg\ Error\ |\ /\ 10\ mm$$

Precision is expressed as the standard deviation of the error in mm.

6.6.1.2 3-Axis Stage – Simulated Real Conditions

For component qualification of the 3-axis stage, the thermoplastic extruder is used to print a representative model in PLA. The movement sequences required to accomplish this task are highly similar to those needed after modification and well represent the capability of the stage to meet specifications. The resolution of the stage is assessed by measuring the printed model using ImageJ software and comparing to the starting CAD file. The model selected for qualification testing is the "Prusa Logo" design provided by the printer's core developer. This design was selected because it contains features of the dimensions (0.5 mm - ~50 mm) and aspect ratios (> 2) relevant toward bioprinting. Resolution is determined as the error between the print dimensions and those specified by the CAD file. 50 measurements in all three dimensions are performed on 3 different printed models. Accuracy is expressed both as the average absolute error and the average percent error in length:

$$|\ Avg\ Error\ | = \left|\ Measured\ Distance\ (mm) - CAD\ Distance\ (mm)\ \right|$$
$$\%\ Error = |\ Avg\ Error\ |\ /\ CAD\ Distance\ (mm)$$

Precision is expressed as the standard deviation of the error in mm.

6.6.1.3 *Syringe Pump – Ideal Conditions*

The syringe pump is first tested using distilled water as a model bio-ink. No tubing is attached to the syringe. An Adafruit motor shield is used to control the pump's stepper motor under the control of custom Arduino code. Source code is documented in Appendix A11. The minimum and maximum pump velocities (steps/sec), calibration factor (steps/mm^3), and accuracy/precision are determined.

Velocities are determined by scaling the stepping rate up or down and visually determining when the motor fails. Stepping rates are communicated in real time from the microcontroller over a serial monitor and can be paused by using the Reset function.

The calibration factor, the number of steps required by the stepper motor to extrude 1 mm^3, is determined in a two-step process. First, an unknown quantity of fluid is extruded for 500 steps. The mass of the extruded fluid is measured with a balance. By using deionized water with a known density at room temperature, the extruded fluid volume is then calculated. The calibration factor in steps/mm^3 can then be extracted. This procedure is repeated 50 times for reliability.

A second experiment confirms the calibration factor and measures accuracy/precision. Again, an unknown quantity is extruded that corresponds to the number of steps equal to the calibration factor. The extruded volume should be exactly 1000 mm^3, which can be confirmed by weighing the extruded mass and calculating its volume using the room temperature density. Accuracy is expressed as both the average absolute error and average percent error in volumetric extrusion:

$$|\ Avg\ Error\ | = \left|\ Extruded\ Volume\ (mm^3) - 1000\ mm^3\right|$$
$$\%\ Error = |\ Avg\ Error\ |\ /\ 1000\ mm^3$$

Precision is expressed as the standard deviation of the error between target and extruded volumes in mm^3.

6.6.1.4 *Syringe Pump – Simulated Real Conditions*

Simulating real conditions for the syringe pump requires the use of a liquid biomaterial with moderate viscosity. Such testing is not possible without integrating the syringe pump with the syringe heater, since the model biomaterial used, gelatin, is only liquid at elevated temperature. Thus, the syringe pump will be qualified as part of Section 6.6.2.4.

6.6.1.5 *Syringe Heater – Ideal Conditions*

The capability of the syringe heater assembly to reach and maintain a set temperature is measured first. The syringe heater is managed by its own internal control, set using the manufacturer's GUI. A simple measurement device, consisting of a 100K thermistor connected to an Arduino

microcontroller loaded with custom code, is sufficient to determine temperature. A custom script written in Processing is used to graph temperature versus time, demonstrating thermal stability. The code used to measure temperature through the Arduino microcontroller and that used to plot temperature are documented in Appendices A12 and A13**.**

Two parameters are measured in this test: temperature stability and time to stability. Temperature stability is first assessed visually by ensuring that the plotted temperature remains within the confines of an acceptable error, +/- 5^{o}C of the set point. Next, the time to stability is defined as the time until the temperature plot last crosses either the upper or lower error threshold. A lower time to threshold thus describes a better tuned or more responsive system. To quantify stability, the average temperature error is calculated for a period of time 20 times the established time to stability.

6.6.1.6 *Syringe Heater – Simulated Real Conditions*

Qualification of the syringe heater assembly is accomplished by measuring the thermal stability of a syringe of gelatin with the heating pads wrapped around it. The measurement device described in Section 6.6.1.5 is used, with the thermistor submerged in the syringe to determine internal temperature. A custom script written in Processing is used to graph temperature versus time (Appendix A13).

Two parameters are measured in this test: temperature stability and time to stability. Temperature stability is first assessed visually by ensuring that the plotted temperature remains within the confines of an acceptable error, +/- 5^{o}C of the set point. Next, the time to stability is defined as the time until the temperature plot last crosses either the upper or lower error threshold. A lower time to threshold thus describes a better tuned or more responsive system. To quantify stability, the average temperature error is calculated for a period of time 20 times the established time to stability.

6.6.1.7 *Bed Chiller – Ideal Conditions*

The capability of the bed chiller to reach and maintain a set temperature is measured in much the same way as the syringe heater in Section 6.6.1.5. The Peltier element is powered through a relay under the control of an Arduino microprocessor. Two types of control code are used: a binary (on-off) system to determine the range of possible temperatures possible and a PID control algorithm to maintain dynamic thermal stability (Appendix A12). Temperature is measured using the same device described above, with the thermistor placed directly on the Peltier surface. A custom script written in Processing is used to graph temperature versus time (Appendix A13).

In order to delineate the contributions of the hardware and the control software toward temperature stability, a more robust testing protocol is needed. First, the capability and stability of the control algorithm is probed, then the power of the Peltier is tested. To quantify the efficiency of the software, a very moderate set point is chosen, 15^{o}C, which is well within the capability of the Peltier to maintain. Upon the success of this test, a more

realistic set point is chosen to measure the performance of the Peltier element.

Two parameters are measured in each iteration of this test: temperature stability and time to stability. Temperature stability is first assessed visually by ensuring that the plotted temperature remains within the confines of an acceptable error, +/- 5^{o}C of the set point. Next, the time to stability is defined as the time until the temperature plot last crosses either the upper or lower error threshold. A lower time to threshold thus describes a better tuned or more responsive system. To quantify stability, the average temperature error is calculated for a period of time 20 times the established time to stability.

6.6.1.8 *Bed Chiller – Simulated Real Conditions*

Qualification of the bed chiller assembly occurs in exactly the same manner as Section 6.6.1.7; however, the Peltier element is thermally bound to the print bed and the measurement/control thermistor is placed on the boundary of the print bed. The capability of the system to maintain a moderate temperature drop is assessed first to prevent thermal burn-out of the Peltier element while any remaining bugs in control software are ironed out. Finally, the feasibility of the entire assembly to maintain a full temperature drop is assessed.

Two parameters are measured in each iteration of this test: temperature stability and time to stability. Temperature stability is first assessed visually by ensuring that the plotted temperature remains within the confines of an acceptable error, +/- 5^{o}C of the set point. Next, the time to stability is defined as the time until the temperature plot last crosses either the upper or lower error threshold. A lower time to threshold thus describes a better tuned or more responsive system. To quantify stability, the average temperature error is calculated for a period of time 20 times the established time to stability.

6.6.2 *Integrated Component Testing*

6.6.2.1 *3-Axis Stage & Nozzle Assembly*

This test assesses the resolution of the stage after incorporation of the nozzle assembly, including tubing. The protocol is identical to that of Section 6.6.1.1.

6.6.2.2 *Syringe Pump & Control Board*

The capability of the extruder stepper motor driver on the control board to power the syringe pump is assessed here. The determination of minimum and maximum extrusion velocities of phosphate-buffered saline (PBS) is sufficient to demonstrate driver compatibility. Extrusion G-code commands of increasing or decreasing rates are sent to the control board via Pronterface.

6.6.2.3 *Syringe Pump, Syringe Heater, & Control Board*

Since the capability of the syringe heater to maintain a volume of gelatin equal to that used during printing at a target temperature was established in Section 6.6.1.6, this integration test serves to measure the minimum and maximum rates that gelatin can be extruded without occluding the tubing or stalling the stepper motor. The failure mode for rapid extrusion is viscous resistance in the tubing, while that for slow printing is temperature-induced gelation and occlusion. Extrusion G-code commands will be sent via Pronterface using various rates, with the laminar flow of gelatin within the tubing assessed visually.

6.6.2.4 *Syringe Pump, Syringe Heater, Nozzle Assembly, & Control Board: Calibration*

Having verified the capability of the syringe heater in Section 6.6.2.2, the syringe pump is now ready for qualification, as indicated in Section 6.6.1.4. The syringe pump is initially qualified by extruding PBS. PBS is used to prevent occlusion in the tubing; secondary qualification and verification (Section 6.6.2.5) will use a viscous biomaterial. The additional fluidic resistance imposed by the nozzle assembly motivates this preliminary assessment. Extrusion G-code commands are sent via Pronterface through the control board to pump PBS through a representative length of tubing to the nozzle assembly. The calibration factor (steps/mm) is calculated, uploaded into the Marlin firmware, and then verified.

A more detailed description for the calculation of this calibration factor in steps/mm and its use in adjusting the printer's firmware is detailed in Section 8. Suffice it to say for now that the mm referred to in the factor actually corresponds to an extruded volume. Initial calculation of the calibration factor is performed in the same manner as Section 6.6.1.3.: an unknown volume of water is extruded during a fixed number of steps and weighed. The volume is then calculated based on room temperature density, allowing determination of the calibration factor. This procedure is repeated 50 times for reliability. The printer firmware is then altered with this new calibration factor. To verify, water is extruded for a number of steps corresponding to 1000 mm according to the factor. The volume printed is compared to the target, permitting evaluation of accuracy and precision. Again, accuracy is presented as average absolute error and percent error, with precision as the standard deviation of error:

$$|\,Avg\ Error\,| = \left|\,Extruded\ Volume\ (mm\text{^}3) - 1000\ mm\text{^}3\,\right|$$
$$\%\ Error = |\,Avg\ Error\,| / 1000\ mm\text{^}3$$

6.6.2.5 *Verification Testing*

Verification of the entire integrated system is completed by assessing extrusion of a model viscous biomaterial, gelatin. Extrusion must be continuous and even as measured through extended straight lines and around corners for one layer. Resolution is assessed by optical measurement

using ImageJ in comparison to the original CAD model. Accuracy is measured as the average absolute error and percent error for 50 measurements made in all three dimensions:

$$| \textit{Avg Error} | = \left| \textit{Measured Distance (mm)} - \textit{CAD Distance (mm)} \right|$$
$$\textit{\% Error} = | \textit{Avg Error} | / \textit{CAD Distance (mm)}$$

Precision is represented as the standard deviation of the error.

6.6.3 *Validation Testing*

6.6.3.1 Gelatin Bioprinting Resolution

Whereas Section 6.6.2.5 measured print resolution in a single-layer model, this validation test assesses the capability of the bioprinter to construct a representative 3D object out of gelatin. Given that most biomaterials lack the stiffness necessary to produce tall or narrow structures, a model with a height:width aspect ratio of 1:3 was designed to be representative. Resolution is assessed by optical measurement using ImageJ in comparison to the original CAD model. Accuracy is measured as the average absolute error and percent error for 50 measurements made in all three dimensions:

$$| \textit{Avg Error} | = \left| \textit{Measured Distance (mm)} - \textit{CAD Distance (mm)} \right|$$
$$\textit{\% Error} = | \textit{Avg Error} | / \textit{CAD Distance (mm)}$$

Precision is represented as the standard deviation of the error.

6.6.3.2 Cell-Matrix Bioprinting Test

A cell-gelatin solution will be printed to assess the maximal cell viability attainable by the bioprinter. Live/dead staining and basic light microscopy will determine viability both immediately after printer and one day post-printing with incubation in a standard biocabinet.

7. Proof of concept

7.1 *Proof of concept requirements*

A proof of concept model is generated to demonstrate the critical design elements of the bioprinter. Using the Pareto analysis (Table 5), key features contribute to a cumulative 80% of the design effort. The functions to be tested include x-, y-, and z- resolution of the print head, syringe pump stepper motor – RAMBo board driver compatibility, TEC and extruder wrap thermoregulation capabilities, and system calibration. These factors inform the design of the proof of concept model and outline the testing protocol designed for it.

Quality Tools

Category	Total
X Resolution	100
Y Resolution	100
Z Resolution	90
Stepper Motor-Driver Compatib	100
Calibration	50
Bed Temp Mgmt.	50
Syringe Temp Mgmt	50
TEC Temp Reg.	80
Extruder Wrap Temp Reg.	80
Needle Enclosure	25
Slicer Parameters	50
Firmware Parameters	30

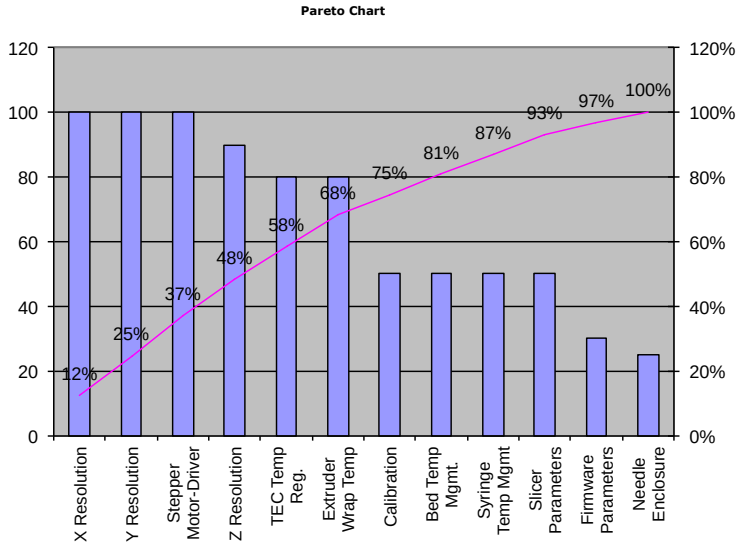


Table 5. Pareto analysis identifying the critical design elements for inclusion in proof of concept model.

7.2 *Proof of concept implementation*

The largest portion of implementation, the construction of the 3D printer, was completed in whole first. Digital photographs of the Prusa i3 printer with PLA filament built as part of this work are presented in Figure T below. Detailed instructions for constructing a Prusa device can be found on the RepRap wiki[26] or on the Prusa homepage.[43] Initial testing revealed difficulty in calibrating the axes. To establish a useful coordinate system, all three axes must return to a "home" or "zero" position from which the controller can calculate permissible bounds for movement. These homing positions are determined by three limit switches, which, when contacted, close a circuit that is then detected by the control board. Both the X- and Y-axis homed accurately, but the Z-axis presented some difficulties, particularly when the extruder nozzle extended further down than did the rail carriages on the Z-axis. To address this problem, a small component was designed, printed, and affixed to the rail carriage that would permit precise adjustment of the homing position (Figure T). Tests 6.6.1.1 and 6.6.1.2 were then performed as described in Section 6.6.

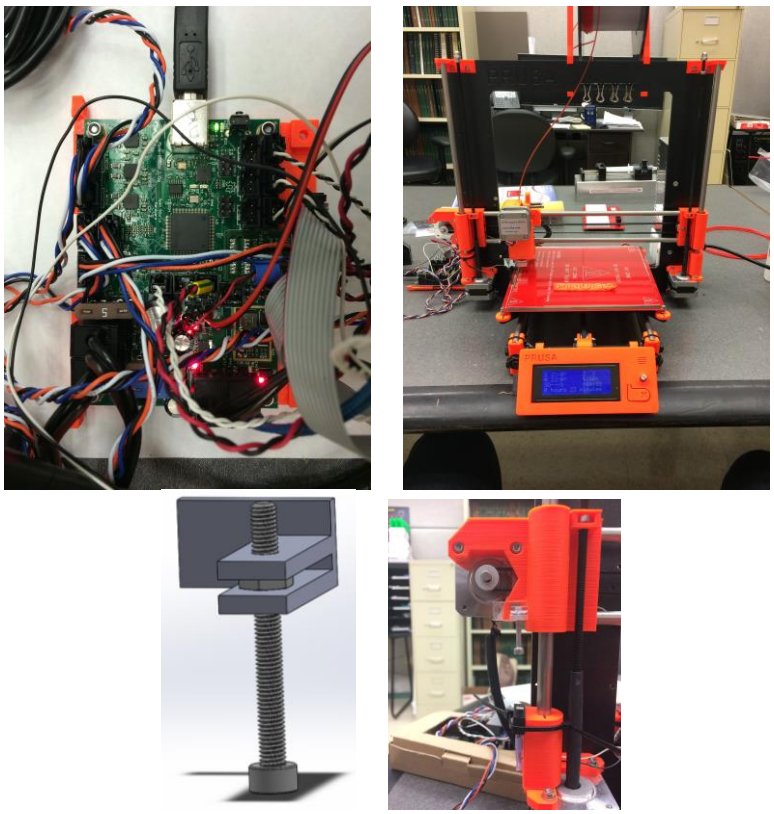


Figure T. (Top-Left) Assembled Mini-RAMBo board with all stepper motors, power supplies, microswitches, thermistors, and heating elements plugged in. Notice that all of the ports on the board, save one thermistor and three microswitch inputs, are utilized. The addition of multiple independent extruders operating synchronously would require a motherboard with additional stepper driver capacity. (Top-Right) The completed Prusa i3 printer with attached LCD screen. The panel reads a SD card, providing basic control of the hardware through a simple GUI and a means to send G-code to the microprocessor during offline printing. (Bottom-Left) Z-calibration component CAD model. (Bottom-Right) Z-calibration component bonded to X-axis, just above the limit switch.

The initial design for the extrusion assembly called for modification of an existing syringe pump. In an effort to minimize costs, while preserving the integrity of the pump, the least expensive model on the current market was purchased, a New Era™ NE-500. Figure Ua below demonstrates the commercial pump assembled with a syringe and needle. Upon testing, it was discovered that the stock stepper motor was not compatible with the drivers on the RAMBo board (Figure Ub). Therefore, two options were considered: replacing the motor on the prefabricated pump assembly with a compatible NEMA 17 model or opting for a completely new, open-source pump design with a suitable stepper motor. Since the driver board and large, heavy assembly of the stock pump would be superfluous for the design of the bioprinter, the open-source stepper design was selected (Figure Uc). This device was assembled using an SX17-1003LQEF NEMA 17 stepper motor from Microcon Technologies, which is identical to that used in the filament extruder assembly. The assembly of the syringe pump is documented in Figures Ud,e, below. Excellent instructions for constructing this piece equipment can be found online.[44] Test 6.6.1.3 was then performed as described in Section 6.6.

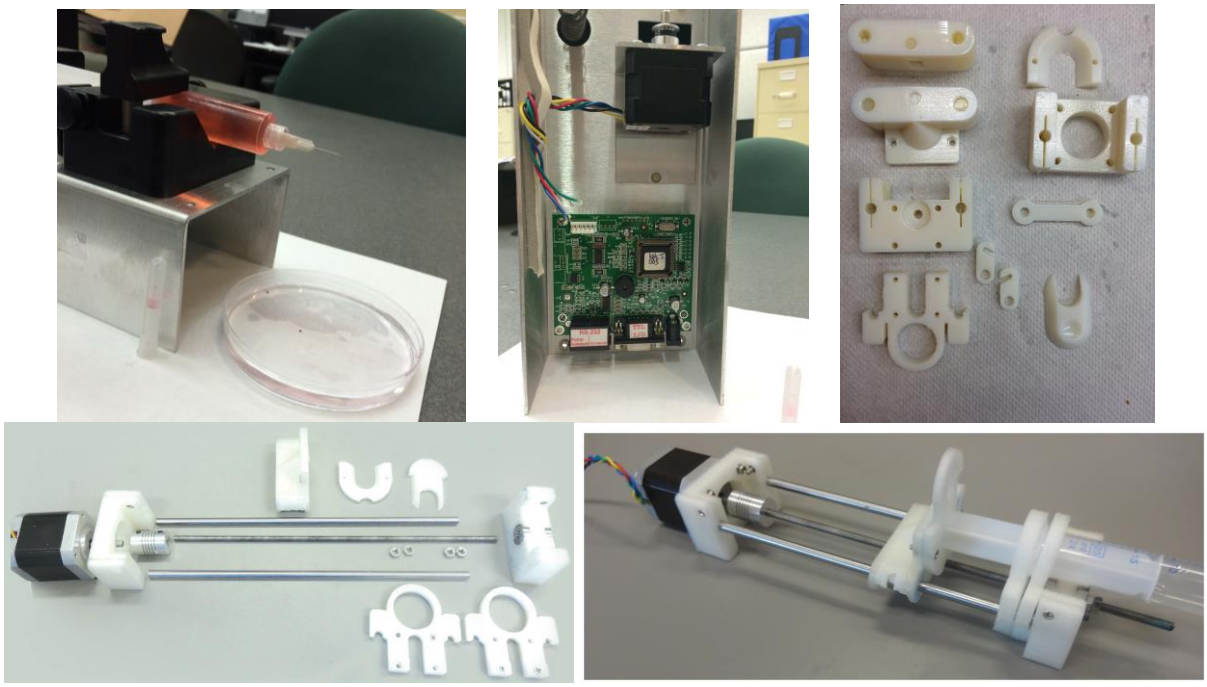


Figure U. (Top-Left) New Era™ NE-500 syrimge pump assembled with syringe and needle for testing. (Top-Middle) The stepper motor included with the stock pump is incompatible with the RAMBo board stepper motor drivers. (Top-Right) 3D printed parts based on the open-source syringe pump design by the Pearce group at Michigan Technical University. Source files can be found online.[45] (Bottom-Left) Laying out all of the components needed for the syringe pump. (Bottom-Right) The fully-constructed pump loaded with a syringe.

The two thermoregulatory subsystems were assembled next. The syringe heater from New Era Pump Systems™ needed only to be connected to its controller, as described by the manufacturer. In order to determine the capability of the heater to maintain a set point, a temperature measurement system was developed using a 100KΩ NTC thermistor and an Adafruit Uno R3. The system will be described in the context of its use with the bed chiller assembly, but a simple diagram is provided in Figure V, below. Tests 6.6.1.5 and 6.6.1.6 were then performed using this system for measurement and the graphing code developed in Processing (Appendix A13).

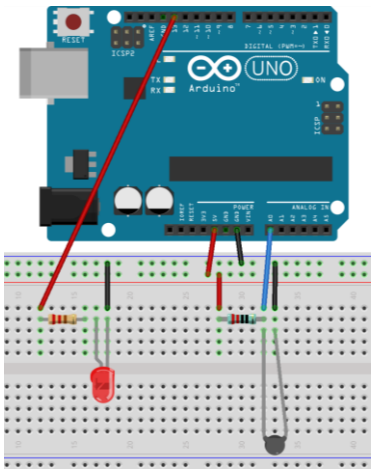


Figure V. Simple temperature measuring device utilizing a thermistor set up in a voltage divider circuit. Source code for the microcontroller is provided in Appendix A12.

The bed chiller required a substantial design effort; both a heat sink and feedback control system were developed. Peltier elements pump heat from one face to the other, proportional to the amount of current supplied to the element. The direction of heat flow depends on the direction of current flow. As with most heat pumps, efficiency is lost as the temperature of the heat sink, or hot face, increases. The TEC assembly purchased from Adafruit® features an aluminum heat sink bonded to one face and cooled by a large fan. By convecting heat from the sink, the efficiency of the TEC element can be maintained. Thus, to maintain the efficiency of the Peltier element, the cooling fan was powered continually by an external power supply.

Next, a feedback control system was developed. The design goal of the bed chiller is to maintain a desired, reduced temperature at the printing plane. As such, the system is intended to act much like a home thermostat – powering the air conditioning when the room temperature rises above a certain threshold, and turning the compressor off when the temperature falls back down. Doing so requires at minimum three essential elements: a temperature-measurement device, a microcontroller that manages the set point, and a heat pump. In the case of the bed chiller, these three components are a 100KΩ NTC thermistor, an Arduino Uno R3 microcontroller, and the Peltier assembly. Importantly, the Arduino cannot handle the current (5A) needed to power the Peltier element; therefore, a relay switch (PowerSwitch Tail 2) was placed between the two such that the microcontroller could throttle the heat pump at will. Since the Uno does not have the latent capacity to measure the changing resistance of the thermistor, the temperature measurement circuit was set up as a voltage divider. Upon integration with the 3-axis stage, the Peltier will be thermally pasted to the aluminum build plate and the thermistor will be glued to its border. The full system layout is depicted in Figure W, below.

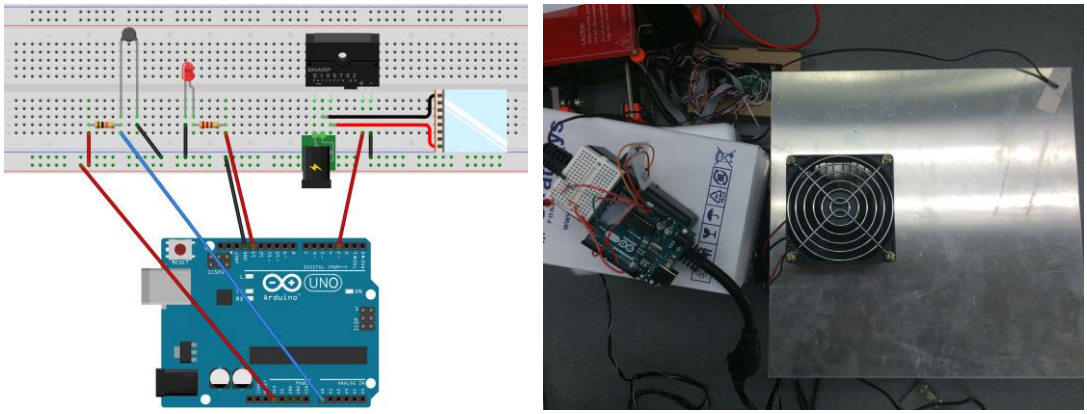


Figure W. (Left) Layout of the bed chiller system. An Arduino Uno microcontroller continuously measures the voltage at A0, which varies with temperature according to the thermistor set up in a voltage divider. With this information, decisions are made regarding cooling and the relay switch connected to pin 3 can be toggled on or off. The Peltier element is powered externally, under control of the relay. The LED circuit connected to pin 13 serves as a debugging tool and an indicator light when the Peltier is powered. (Right) Digital photograph of the Peltier assembly during testing.

With the hardware in place, two different control architectures were implemented using the Arduino. The full source code can be found in Appendix A12. First, a simple binary algorithm, the "bang-bang" model, was constructed to evaluate the capability of the system. This system simply works by comparing the current temperature to the set point at regular intervals and either opening or closing the relay if cooling is needed. Bang-bang systems work excellently when the parameter being measured is in the immediate proximity of the active element; that is, if the thermistor were directly connected to the Peltier element. However, the goal of this design is to manage the temperature of the entire build plate, so temperature must be measured away from the active element. In this scenario, bang-bang architectures suffer from systemic overshoot due to the latency in temperature change from the Peltier to the thermistor. In other words, by the time the thermistor has registered that the build plate has reached the set point, the region directly above the Peltier element has cooled far beyond target. The exact opposite is true when the build plate is warming back up. For this reason, most modern control systems utilize a Proportional-Integral-Derivative (PID) algorithm.

The PID controller attempts to control a system by modeling its current status, history, and future using an error function.[46,47] The error function, e(t), is just the deviation of a parameter from its target, in this case, the difference between the current temperature and the set point. By tracking the error function over time, the dynamics of a system can be modeled to best control the output, the Peltier element. The basic PID algorithm looks like this:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

in which e(t) represents the error function, u(t) the output to the active element, and K_p, K_i, and K_d are constants of proportionality that define the relative weights of the present, past, and future models. The error function is quite easy to calculate using the thermistor, but the output function presents some difficulties. Essentially, the output function is just a number which will scale in value with the need for the active element to perturb the system. This value can be most readily used to scale the amount of current (or voltage) the active element, the Peltier, receives, which will adjust how rapidly it pumps heat. Because the Peltier is powered through a relay, however, its status is limited to either full power or completely off. Thus, the PID code designed here defines a "Relay Window" that correlates the output function to a duty cycle. That is, the output function value is scaled to a certain proportion of the relay window for which the Peltier should be powered. Instead of toggling how much power the Peltier receives, this PID code varies how long the Peltier is powered. Upon completion of the bed chiller control code, Tests 6.6.1.7 and 6.6.1.8 were performed as described.

To verify that the syringe pump constructed previously could be adequately powered by the control board for the 3D printer, the first step of system integration was performed. The extruder stepper motor (E0) was removed from the board and replaced with the stepper powering the syringe pump (Figure X). Both devices used stepper motors with the same parameters, so no adjustments needed to be made to the stepper driver. However, the motor powering the filament extruder operates by spinning a geared wheel, while the one on the syringe pump turns a threaded rod. Therefore, each stepper motor must turn in opposite directions to extrude material. The following change was made in the Marlin firmware to invert the default extruder direction:

From: *#define INVERT_E0_DIR false*
To: *#define INVERT_E0_DIR true*

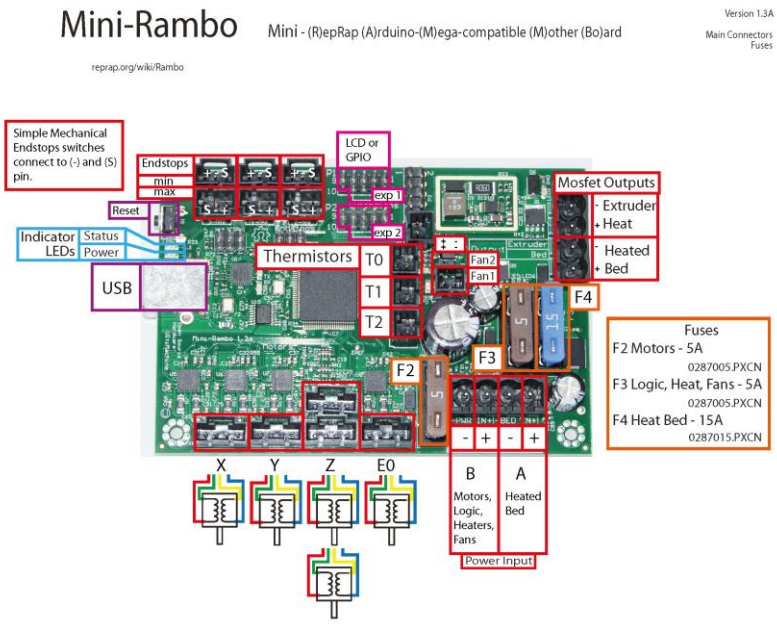


Figure X. Layout of the Mini-Rambo printer controller board. The syringe pump stepper motor snaps into the extruder connector labelled E0.[28]

Further, the Marlin firmware has a built-in safety feature that is intended to prevent the extruder motor from being powered if the nozzle is not yet hot. If filament were pushed into a cold nozzle, it would not melt causing enough force to build up that the print head would break apart. While this feature is still useful for bioprinting, the threshold for this safeguard needs to be lowered to a level well below the melting point of the intended biomaterial. To enable room-temperature extrusions, the threshold was lowered to 20^{o}C, with the following change:

From: *#define EXTRUDE_MINTEMP 190*
To: *#define EXTRUDE_MINTEMP 20*

With that edit to the firmware, the final test for the proof of concept model, Test 6.6.2.2 was performed to demonstrate the compatibility of the syringe pump to the extruder stepper driver on the printer's control board.

7.3 *Results*

7.1.1. 3-Axis Stage – Ideal Conditions

Raw sample data from the X-, Y-, and Z-Resolution tests is provided below (Figure Y). A larger, representative sample of data can be found in Appendix A3. All motion commands were for a target distance of 10 mm. X- and Y-direction movements were altered to expedite testing. A histogram of movement error for all data points is provided in Figure Z. An error of 0 indicates a highly accurate motion, while precision is inversely related to the spread of the histogram. Lastly, the accuracy expressed as the average absolute error and percent error and the precision as the standard deviation of error are summarized in Table 6.

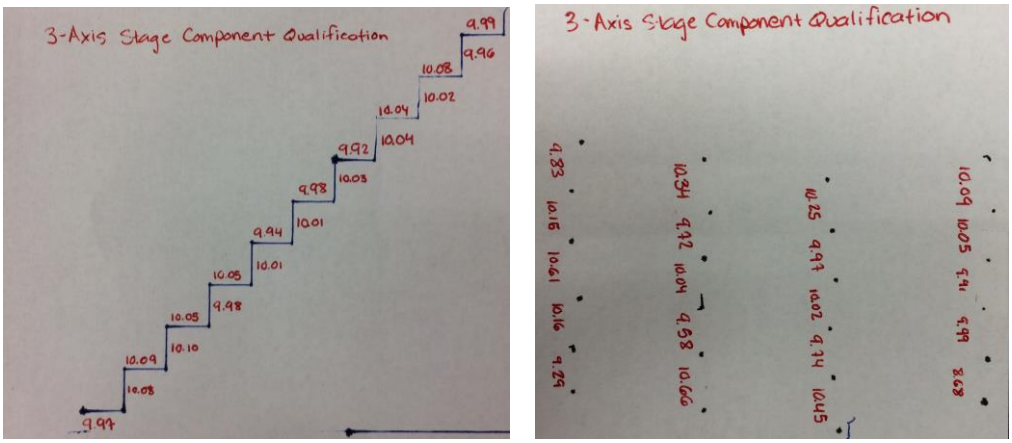


Figure Y. Sample data for X-, Y-, and Z- resolution.

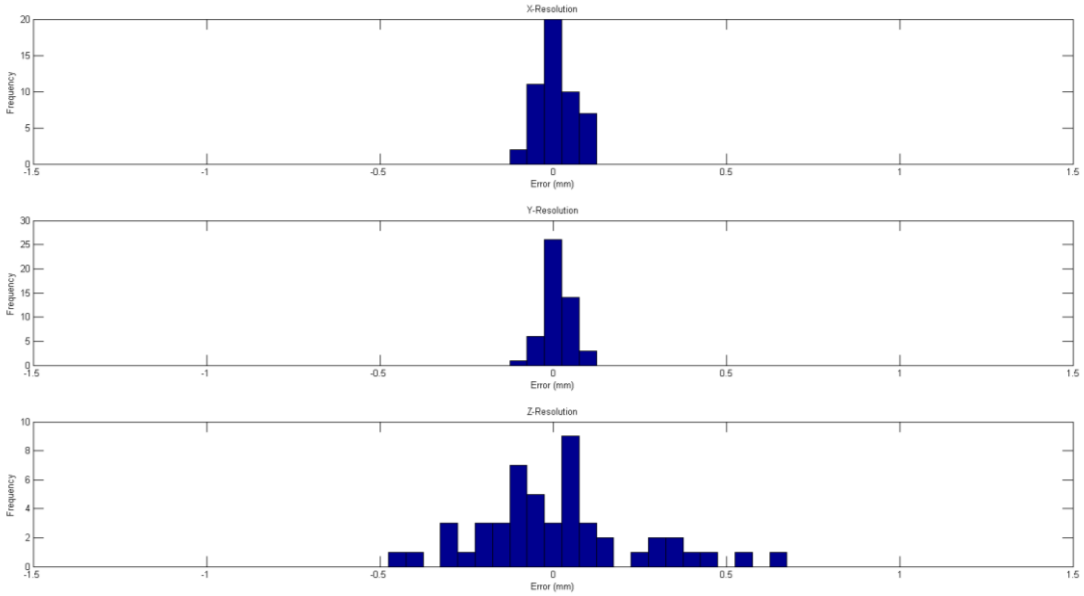


Figure Z. Histograms of X-, Y-, and Z-movement error.

Resolution Summary - 3-Axis Stage Qualification			
Dimension	**Accuracy: Avg Error (mm)**	**Accuracy: % Error**	**Precision (mm)**
X	0.05	0.51%	0.05
Y	0.03	0.35%	0.03
Z	0.28	2.94%	0.22

Table 6. Data summary for 3-axis stage component qualification under ideal conditions.

7.1.2. 3-Axis Stage – Simulated Real Conditions

To obtain 50 independent measurements for Test 6.6.1.2, three separate models were printed with the unmodified extrusion printer. An example of the optical measurement system with sample data from one model is depicted in Figure AA below. A larger representative sample of data can be found in Appendix A4. A histogram of dimensional error for all data points is provided in Figure AB. An error of 0 indicates a highly accurate print, while precision is inversely related to the spread of the histogram. Lastly, the accuracy expressed as the average absolute error and percent error and the precision as the standard deviation of error are summarized in Table 7.

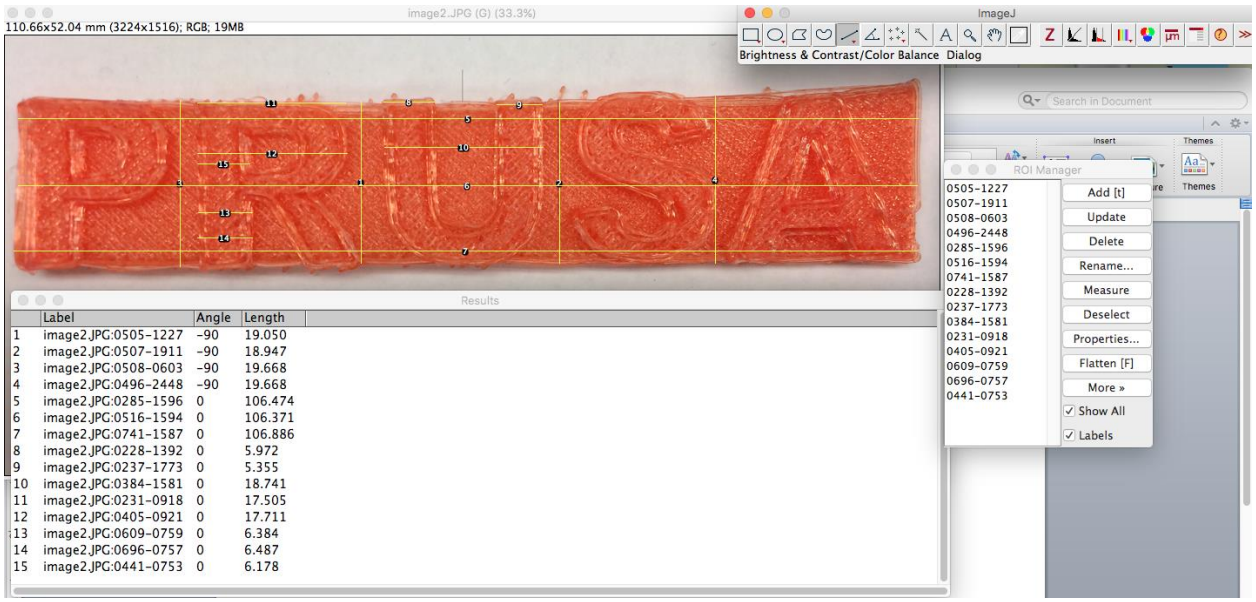


Figure AA. ImageJ software is used to make XY measurements on a printed model.

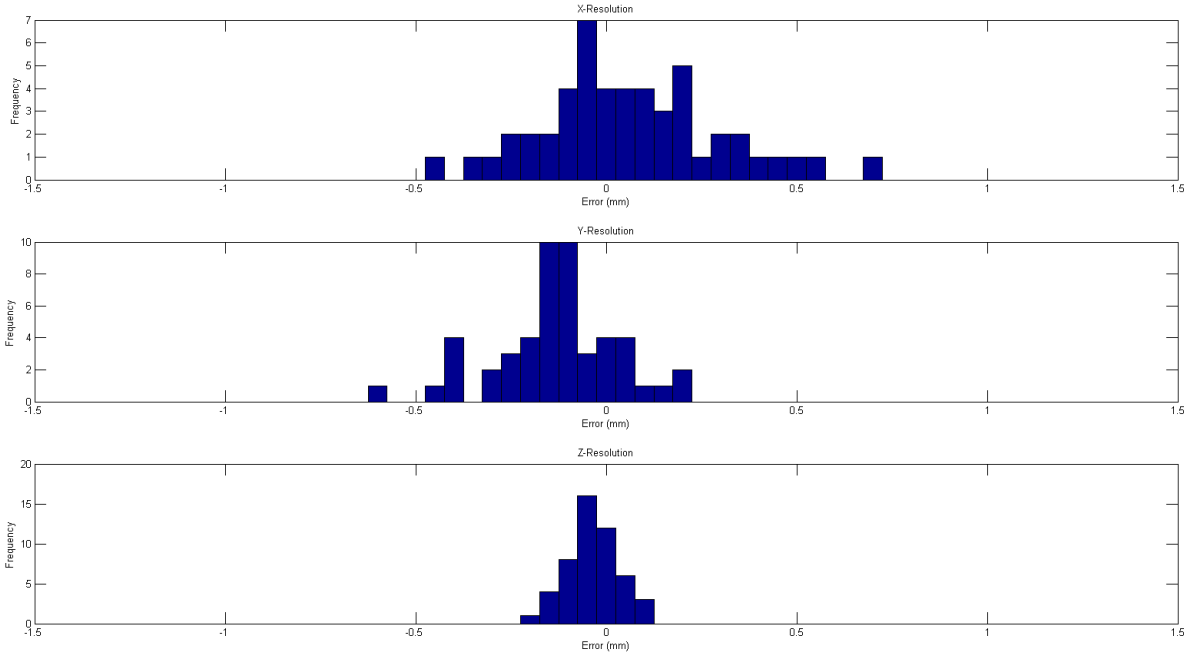


Figure AB. Histograms of X-, Y-, and Z-dimensional error.

"PRUSA" Logo Print Resolution: Data Summary			
Dimension	**Accuracy: Avg Error (mm)**	**Accuracy: % Error**	**Precision (mm)**
X	0.51	2.14%	0.24
Y	0.34	2.49%	0.21
Z	0.08	6.00%	0.07

Table 7. Data summary for 3-axis stage component qualification under simulated real conditions.

7.1.3. Syringe Pump – Ideal Conditions

Qualification of the syringe pump required a two-step process. First, the pump motor was actuated for 500 steps and the mass of extruded water determined with a balance. Using the density of water, the extruded volume was then calculated. The calibration factor in steps/mm^3 was then determined for each sample. A histogram of calibration factors is provided in Figure AC. In order to perform the second step of the process, the number of steps required to extrude 1000 mm^3 was calculated using the calibration factor, which came out to 1,270 steps. The error in extruded volume was calculated by again measuring the mass of water and dividing by its room temperature density. A histogram of volume error is provided in Figure AC. An error of 0 indicates a highly accurate extrusion, while precision is inversely related to the spread of the histogram. A larger representative sample of data can be found in Appendix A7. The calibration factor determined in the first step, the accuracy expressed as the average absolute error and percent error, and the precision as the standard deviation of error are summarized in Table 8. Lastly, the maximum and minimum extrusion velocities were determined by visual inspection while ramping the step rate up or down. Results, including the expected or hypothesized failure mode, are presented in Table 9.

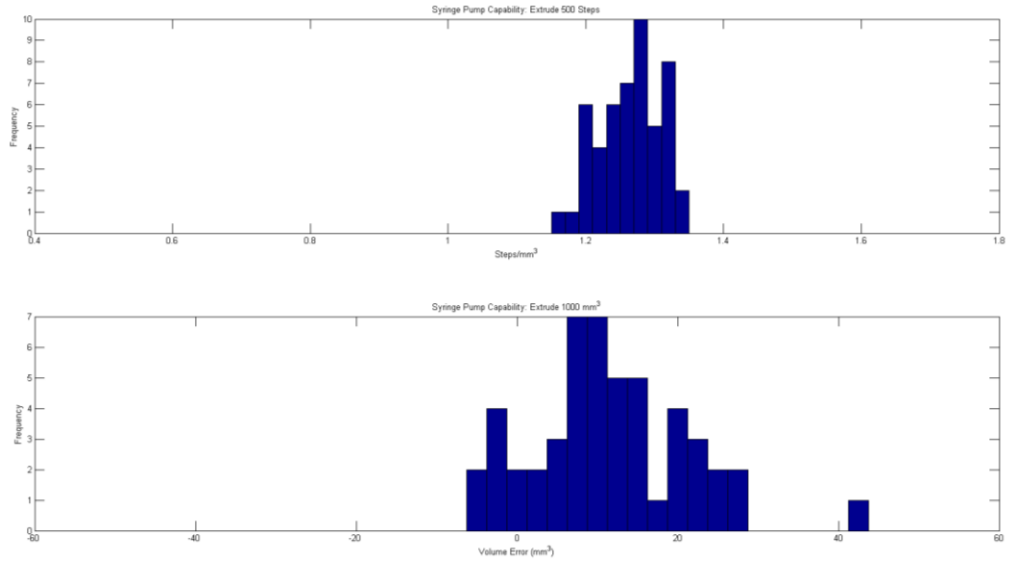


Figure AC. (Top) Histogram of calculated calibration factors based on 500-step extrusions. (Bottom) Histogram of error between intended and measured volume upon 1,270-step extrusion.

Calibration Summary - Syringe Pump Qualification	
Steps per mm^3	1.27
Accuracy: Average Error (mm^3)	14.34
Accuracy: % Error	1.91%
Precision (std) (mm^3)	10.79

Table 8. Data summary for syringe pump component qualification under ideal conditions.

Syringe Pump CQ: Speed Testing		
Speed	**Failure**	**Rate (steps/s)**
Max	Motor Stall	1800
Min	Viscous Stall	N/A

Table 9. Maximum and minimum syringe pump extrusion rates under ideal conditions.

7.1.4. Syringe Heater – Ideal Conditions

Upon assembly of the syringe heater according to the manufacturer's specifications, the capability of the system to maintain a set temperature was assessed. The temperature measurement circuit described in Figure V was constructed, with the thermistor bound directly to the heating wrap. Temperature was plotted in real time using Processing code documented in Appendix A13. A sample of the plotted data is depicted in Figure AD, below. Temperature stability was ascertained visually from these plotted data, along with time to stability. Average temperature error was then aggregated for a period 20 times that of the determined time to stability. These parameters are summarized in Table 10.

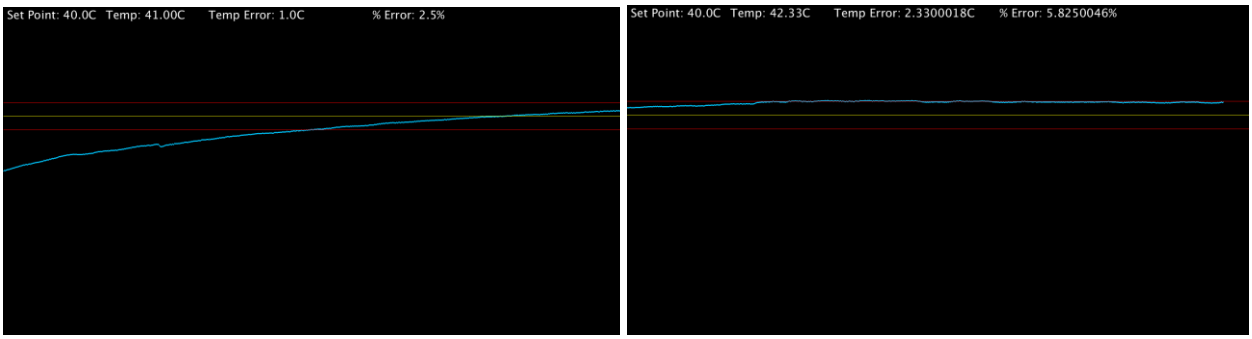


Figure AD. A plot of temperature in blue versus time demonstrates thermal stability. The set point temperature is plotted in yellow with +/- 2.5°C acceptable error in red.

Syringe Heater - Ideal Conditions		
Visual Stability	**Time to Stability (s)**	**Average Temperature Error (C)**
Yes	47	2.17

Table 10. Data summary for syringe heater component qualification under ideal conditions.

7.1.5. Syringe Heater – Simulated Real Conditions

Realistic syringe heater conditions were simulated by wrapping the heater around a syringe filled with 40 wt% gelatin and measuring the temperature inside of the slurry (Figure AE). Temperature was plotted in real time using Processing code documented in Appendix A13. A sample of the plotted data is depicted in Figure AF. Temperature stability was ascertained visually from these plotted data, along with time to stability. Average temperature error was then aggregated for a period 20 times that of the determined time to stability. These parameters are summarized in Table 11.

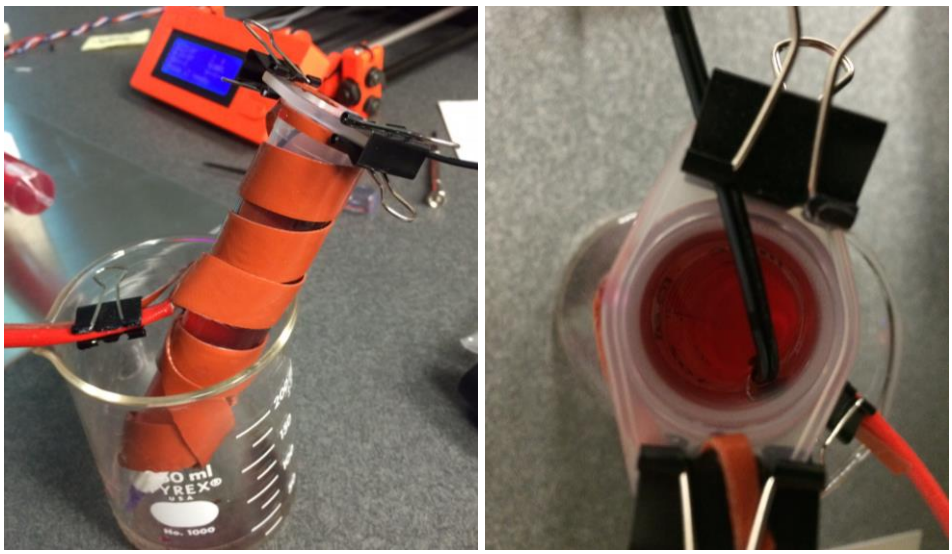


Figure AE. Digital photographs depicting the setup for syringe heater component qualification testing under simulated real conditions.

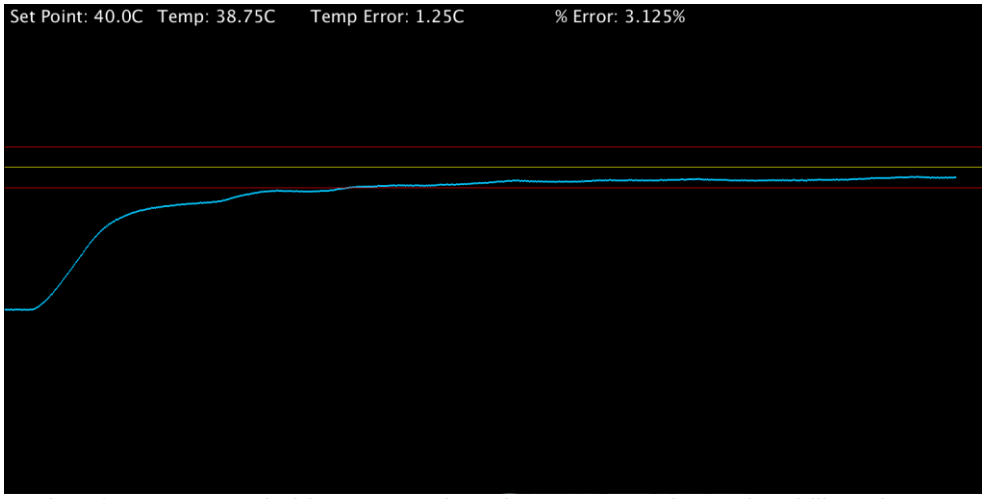


Figure AF. A plot of temperature in blue versus time demonstrates thermal stability. The set point temperature is plotted in yellow with +/- 2.5^{o}C acceptable error in red.

Syringe Heater - Simulated Real Conditions		
Visual Stability	**Time to Stability (s)**	**Average Temperature Error (C)**
Yes	252	-1.52

Table 11. Data summary for syringe heater component qualification under simulated real conditions.

7.1.6. Bed Chiller – Ideal Conditions

The bed chiller system was assembled according to the design depicted in Figure W. Briefly, the Peltier element is powered through a relay switch, under the control of an Arduino microprocessor. The Arduino calculates the temperature in real-time using a thermistor laid out in a voltage divider circuit. The power delivered to the TEC is determined by one of two control architectures: direct "bang-bang" or a PID system. For testing under ideal conditions, the thermistor was taped to the Peltier surface, allowing measurement of the absolute temperature capabilities of the element. The layout of this test is depicted in Figure AG. Using both flavors of control algorithm, the capability of the software to maintain a moderate temperature drop (15^{o}C) was first assessed, followed by a determination of the system's performance during full-range cooling (4^{o}C). Sample data for the bang-bang control algorithm is provided in Figure AH. Temperature stability was ascertained visually from these plotted data, along with time to stability. Average temperature error was then aggregated for a period 20 times that of the determined time to stability. These parameters are summarized for the "bang-bang" code in Table 12. The PID code was assessed next, with sample data provided in Figures AI and AJ and a data summary provided in Table 13.

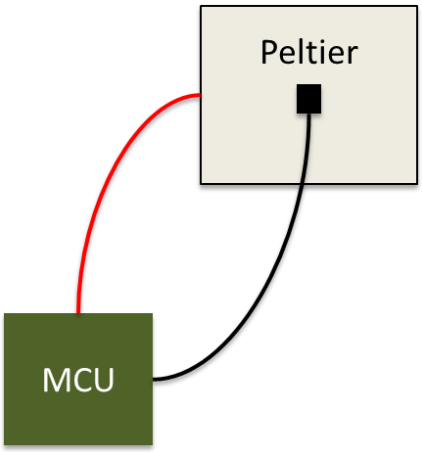


Figure AG. System hardware layout for ideal-case bed chiller component qualification.

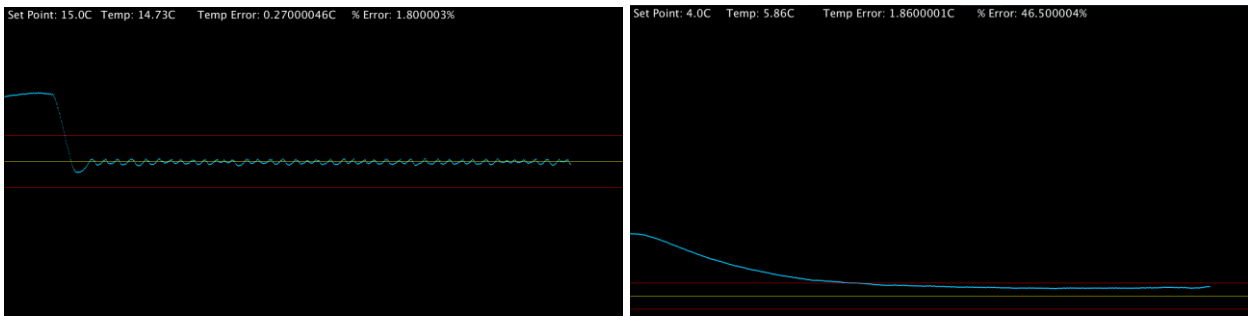


Figure AH. A plot of temperature in blue versus time demonstrates thermal stability. The set point temperature is plotted in yellow with +/- 2.5°C acceptable error in red. (Left) Assessment of capability for "bang-bang" control algorithm to maintain a moderate temperature drop. (Right) Assessment of capability for Peltier element to maintain a full temperature drop.

Bed Chiller - Direct Measurement		
"Bang-Bang" Control Algorithm		
	15C	**4C**
Visual Stability	Yes	Yes
Time to Stability (s)	12	29
Average Temperature Error (C)	0.06	1.24

Table 12. Data summary for "bang-bang" code-controlled bed chiller component qualification under ideal conditions.

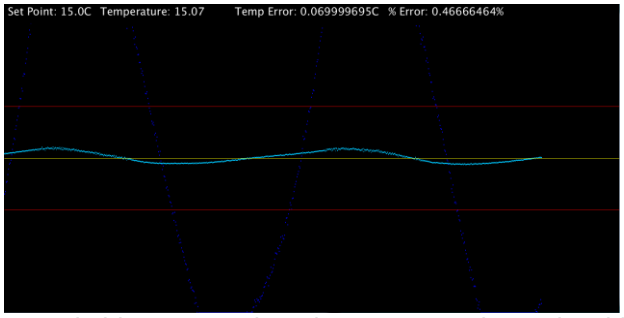


Figure AI. A plot of temperature in blue versus time demonstrates thermal stability of the PID-controlled bed chiller for a moderate temperature drop. The set point temperature is plotted in yellow with +/- 2.5^{o}C acceptable error in red.

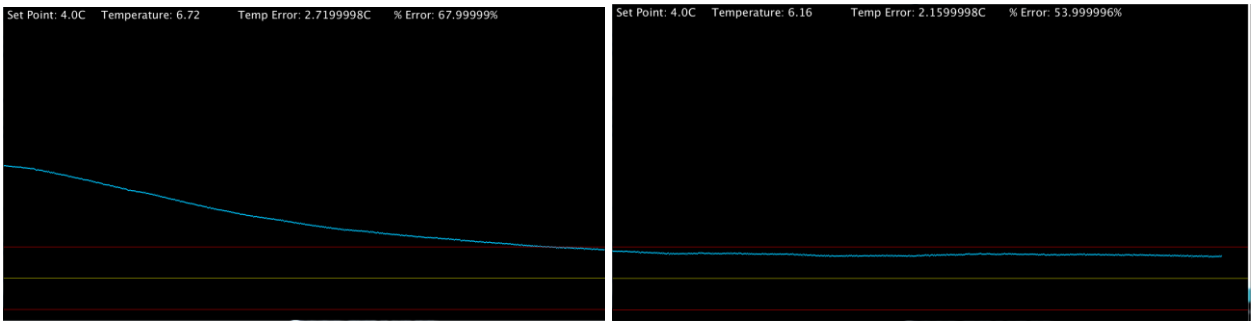


Figure AJ. A plot of temperature in blue versus time demonstrates thermal stability of the PID-controlled bed chiller for a full-range temperature drop. The set point temperature is plotted in yellow with +/- 2.5^{o}C acceptable error in red.

Bed Chiller - Direct Measurement		
PID Control Algorithm		
	15C	**4C**
Visual Stability	Yes	Yes
Time to Stability (s)	68	137
Average Temperature Error (C)	0.11	1.74

Table 13. Data summary for PID code-controlled bed chiller component qualification under ideal conditions.

7.1.7. Bed Chiller – Simulated Real Conditions

To simulate real conditions, the same testing was repeated in a new configuration (Figure AK). The Peltier element was mounted to a 200 mm x 200 mm 1/8" aluminum plate using thermal grease with the thermistor taped to the very border. The order of the tests remained the same: first the "bang-bang" code and then the PID at a moderate temperature drop, then a full one. The Peltier element assembly failed all tests, as depicted in Figure AL. The heat pump lacked the power to induce a measured temperature drop greater than 2.5^{o}C. This result is summarized in Table 14.

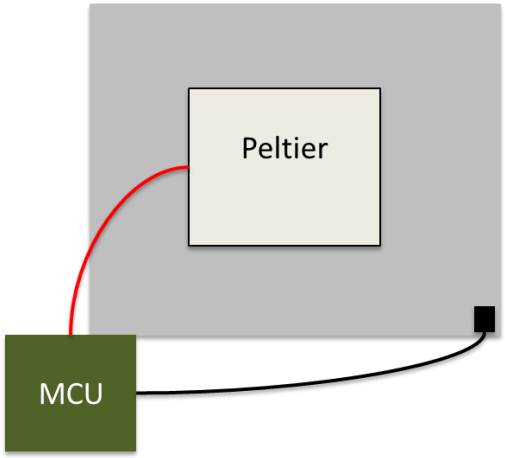


Figure AK. System hardware layout for simulated real-case bed chiller component qualification.

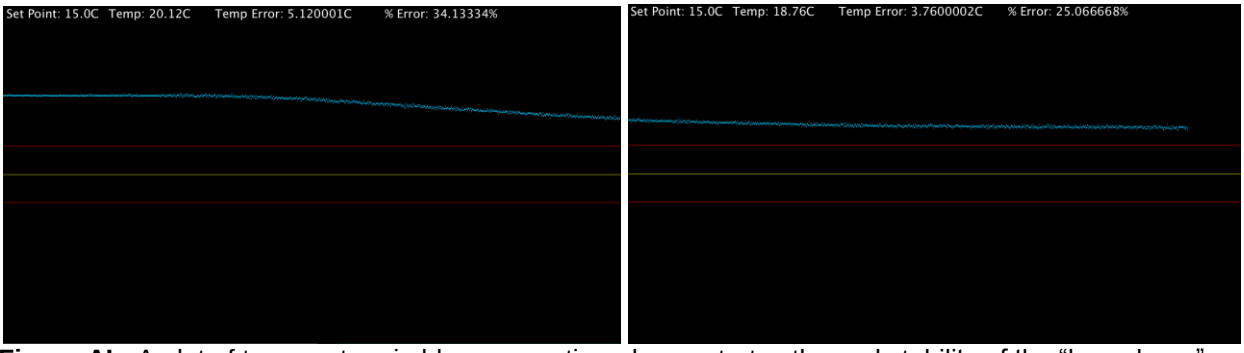


Figure AL. A plot of temperature in blue versus time demonstrates thermal stability of the "bang-bang" code-controlled bed chiller for a moderate temperature drop. The set point temperature is plotted in yellow with +/- 2.5^{o}C acceptable error in red. While thermally stable, the temperature never reached the set point.

Bed Chiller – Plate Measurement		
"Bang-Bang" Control Algorithm – 15C		
Visual Stability	**Time to Stability (s)**	**Average Temperature Error (C)**
No	240	3.57

Table 14. Data summary for the "bang-bang" code controlled bed chiller component qualification under simulated real conditions.

7.1.8. Integration: Syringe Pump & Control Board

The final test for the proof of concept is intended to demonstrate the compatibility between the stepper driver on the printer control board and the stepper motor powering the syringe pump. Assessment of the maximum and minimum extrusion rates set the upper and lower bounds for ideal-condition printing speeds. The A4982 stepper driver is current-limited at 2 A, while the Arduino motor shield R3 used during preliminary testing provides up to 2.5A; therefore, an approximately 20% lower maximum extrusion rate is expected, assuming a linear relationship between current and speed. However, the lower current will also reduce the torque produced by the motor, which may further limit extrusion

speed due to fluid pressure building up in the syringe. Thus, a speed reduction less than 30% will be acceptable. The testing setup is depicted in Figure AM, with the syringe linked to a prototype nozzle assembly. The test results are summarized in Table 15.

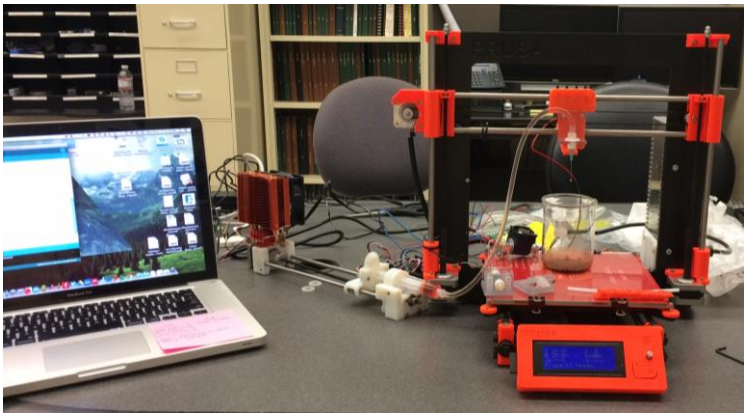


Figure AM. Test setup for the verification of syringe pump – stepper driver compatibility.

Syringe Pump – Stepper Driver Compatibility: Speed Testing		
Speed	**Failure**	**Rate (steps/s)**
Max	Motor Stall	1350
Min	Viscous Stall	N/A

Table 15. Data summary for the verification of syringe pump – stepper driver compatibility.

7.4 Discussion

Discussion of the results obtained by proof of concept testing is organized according to the order in which tests were performed. By focusing on the individual system components in isolation, their capability to meet the specification of the full bioprinting system are demonstrated first under idyllic conditions, then under simulated "real" settings. As an overall conclusion, the 3-axis stage, the syringe pump, and the syringe heater, but not the bed chiller, demonstrated characteristics suitable for integration into the bioprinter.

7.1.1. 3-Axis Stage

Initial resolution testing of the stage demonstrated very high accuracy, precision, and reliability. In the XY-plane, all motioned occurred with an error rate of less than 1% (max 0.05 mm error), well within the acceptance criteria of 1 mm error. In the Z-plane, error was slightly increased at just under 3% (max 0.28 mm error), which still sits within acceptable limits. Some of that error may be due to measurement bias during the test, as well. During testing, a marker was affixed to the right-side Z-axis lead screw carriage that was intended to trace a line along a sheet of paper affixed to the frame, parallel to the Z-plane. Instead of directly tracing the path of the carriage, the marker needed to be pressed into the page following each movement, potentially inducing error due to the slightly different

angles at which the marker was manually pressed. Regardless, the inherent system and measurement-induced error still fell well within acceptable limits.

Simulated real-condition resolution testing presented significantly more error in the 3-axis system. To summarize, X-axis error remained just under 2.15% (0.51 mm), Y-error under 2.5% (0.34 mm), and Z-error at 6% (0.08 mm). While all of these nominal error values fall within acceptable limits, it is worth delineating the different sources of error in this test versus the previous, ideal-case iteration. Firstly, the motions tested herein are far more representative of those that will be used during bioprinting, featuring many rapid, small movements along multiple axes simultaneously. Any laxity in the motor belts will produce significantly more error across a series of small movements then they will with longer, smoother ones, like in the ideal-case testing. Secondly, this real-condition test used measurements of the extruded filament as a surrogate for the accuracy of print-head movements. Disparities in filament adhesion to the print bed or slight inaccuracies in extrusion rate could create error that is not representative of print head motion. Further, the thermal contraction of the filament after printing and before measurement may further induce error. Finally, resolution was measured optically for this test, instead of manually with calipers. Optical measurement is a far more efficient process; however, any inaccuracies in scale calibration with the image are passed along to the measurements made.

With these additional, convoluting sources of error included in the mix, it is the opinion of the experimenter that the 3-axis stage is qualified for use in the full bioprinter implementation.

7.1.2. Syringe Pump

Initial testing of the syringe pump demonstrated surprisingly high accuracy, precision, and reliability, given its completely open-source design. Extrusion commands delivered a volume with an average error by volume of just under 2% (less than 15 mm^3). While extrusion accuracy is not specified directly in the acceptance criteria, it can be understood in the context of X/Y/Z-resolution. Over-extrusion will produce prints that exceed dimensions, while under-extrusion will result in gaps within the fabricated construct.

For the sake of explanation, assume that the 15 mm^3 error were distributed evenly across the entire 1000 mm^3 extrusion as an excess and that the entire extrusion took place along a 1000 mm line. Thus, the dimensions of the target construct would be a rounded rectangular prism 1000 mm long, 1 mm wide, and 1 mm tall. In a worst-case scenario, the evenly distributed 15 mm^3 error would also occur only along the leading edge, that is, along the 1000 mm long dimension, making it really 1015 mm long. If the error were evenly distributed along either of the other two dimensions, the 15 mm^3 error would change the width to 1.015 mm or the height to 1.015 mm, a hardly recognizable difference. This worst-case error would only induce an accuracy error of 1.5%, which, although exceeding he specified target of 1 mm error, would not likely reduce the resolution of the entire construct below target. Further, in a realistic case, this error would be distributed as both over- and under-extrusions in all three dimensions, reducing its effect on the overall resolution.

The measured extrusion error is complicated by the interfacial properties of the fluid used. As a highly polar fluid, water at the cubic millimeter scale experiences relatively large cohesive forces. These forces manifest as surface tension, which encourages water to form droplets. While high surface tension is certainly a useful property in layer-by-layer fabrication by improving the rigidity of each layer, it makes reliable extrusion challenging. Instead of extruding as a continuous cylinder, water forms droplets at the syringe or needle tip, where the cohesive forces exceed those induced by gravity. For the sake of these tests, the extruding needle was placed as close to the bottom of the fluid receptacle as possible, to encourage extruded droplets to adhere to a glass substrate. However, any hydrostatic pressure that resulted from the level of water in the receptacle rising above the tip of the needle would discourage appropriate extrusion. Thus, the cohesion of the fluid should be considered as a convoluting source of error in these measurements. It is hoped that the use of a more viscous and dense hydrogel during bioprinting will address these issues.

Speed testing of the syringe pump demonstrates its upper and lower bounds for extrusion velocity. The lower speed bound is expected to result from viscous resistance of the fluid contained in the syringe. Many hydrogel biomaterials exhibit shear thinning properties, typically of the thixotropic variety, in which the apparent viscosity decreases with both the magnitude and duration of applied shear stress. Therefore, the lower the extrusion rate, the higher the apparent viscosity and, therefore, fluidic resistance. The higher speed bound is expected to result from either the motor driver's step impulse generation limit or from torque overload within the motor. The former is a function of the microcontroller clock speed, which for the both RAMBo Mini's and the Arduino Uno's Atmega2560 is 16 MHz, well beyond any reasonable extrusion speed. Thus, it is the buildup of fluid pressure, created by shear along the fluid conduit and elastic recoil in the syringe, tubing, and nozzle assembly that will limit the motor speed.

From the initial syringe pump qualification testing, a maximum speed of 1,800 steps/s was attainable, with no minimum speed determined. The stepper motor takes 200 steps/revolution, so this figure correlates to 9 revolutions per second, or 540 RPM. This figure, which is well beyond the needs for printing, represents the maximum capability of the pump, since the fluid used was only water, which is highly non-viscous compared to a hydrogel. No minimal speed was determined, which is not surprising since a non-viscous fluid was used for testing. Both of these figures were re-tested using the A4982 stepper driver on the printer control board. Since the maximal current supplied by this driver is 20% less than that on the motor shield, a lower maximal speed was expected. The expected lower speed would likely be more than 20% less than the previous value determined, because the lower current will decrease both the stepping torque and the holding torque of the motor – that is, it should not be assumed that current and extrusion speed are directly related. Integration testing indicated a maximal speed of 1,350 steps/s, which corresponds to 405 RPM – well above that needed during printing. Again, no minimal speed was determined, as expected when using a non-viscous fluid.

Taken together, these results indicate that the open-source syringe pump designed herein provides the power, accuracy, and precision needed for integration into a bioprinter.

7.1.3. Syringe Heater

Initial testing of the syringe heater demonstrated highly accurate temperature regulation and the heating capacity to maintain a hydrogel in its liquid state. When measuring temperature directly on the heating wrap surface, temperature stabilized in under one minute with an average temperature error just over 2°C. It is worth noting that the magnitude and direction of this error varied over the length of the wrap, indicating that the density of the resistive heating wires may vary somewhat. However, the wrap was sufficiently able to keep 80mL of 40 wt% gelatin within about 1.5°C of a 40°C target. These values, a large hydrogel volume, a high concentration of polymer, and a supraphysiological temperature set point were chosen as a worst-case scenario to demonstrate the capability of the system. While the system did take almost 5 minute to thermally stabilize, it maintained a very precise temperature from then on.

The results of the simulated real-case testing do demonstrate a few drawbacks of the syringe heater system. The first is that the thermistor which controls the heater power is embedded in the silicone heater wrap. In an ideal case, the thermistor would be submerged in the hydrogel fluid, such that the heater's internal controls would be sensitive to the actual temperature of interest. In its current form, the wrap will heat to a target temperature, but the hydrogel fluid, after a sufficient time, will only reach a temperature approximately 1.5°C lower. This is assumedly due to the heat losses in the system, primarily by radiation. Therefore, in future tests, the syringe heater will be set to ~1.5°C above the target temperature for the extrusion fluid. The second drawback is the difficulty of securing the heating wrap. Since one of the design goals of the bioprinter is to be able to change materials, and therefore syringes, at will, the heating wrap cannot be permanently bonded to the syringe or tubing. Therefore, it is secured with zip-ties, which don't guarantee a perfect seal between the wrap and the syringe. Any air trapped between the heater and the syringe creates a strong heat barrier due to its very low thermal conductivity (0.024 W/(m*K)).

Taken together, these results indicate that the syringe heater provides the power and precision needed for integration into a bioprinter.

7.1.4. Bed Chiller

Initial testing of the bed chiller system demonstrated satisfactory accuracy, but insufficient cooling power for bioprinter implementation. When assessing the capability of the two control algorithms to maintain the actual Peltier element at a specified temperature, excellent results were obtained. In all iterations using either the "bang-bang" or PID code at a target temperature of 15°C or 4°C, thermal stability was attained within just over two minutes, with an average temperature error under 2°C. These values fall well within acceptable error ranges. However, upon simulated real-case testing using an aluminum build plate, temperature error exceeded all reasonable bounds.

Three factors contribute to the insufficiency of the Peltier assembly to maintain the model build plate at the desired temperature.[48] First, the assembly used is only rated at 72 W, which is below that needed to fully cool the plate. The heat

load experienced by the TEC, once the plate has reached a target temperature, can be modeled as the convective heat transfer of the aluminum plate in air.

$$Q = h_c * A * (T_p - T_a)$$

Where Q refers to the heat load that needs to be carried by the TEC in W, h_c is the convective heat transfer coefficient that is specific to this situation (~100 $W/(m^2K)$ in a worst-case scenario), A is the exposed surface area of the plate, T_p is the temperature of the plate in Kelvin (277.15 K), and T_a is the temperature of the air (23 K). Thus, a suitable TEC assembly should be able to move a 152 W thermal load at minimum. Peltier assemblies additionally generate heat due to their internal resistance during operation, therefore, the real power needed is far greater. A reasonable estimate can be determined from the following formula:

$$\Delta T = (1 - (Q_{load} / Q_{cooling})) * T_{max}$$

Where ΔT refers to the actual temperature difference between the hot and cold sides (19°C), Q_{load} is the heat load (152 W), $Q_{cooling}$ is the cooling power of the TEC, and T_{max} is the most efficient temperature difference between the hot and cold side under no load, a rating provided by the manufacturer and most commonly ~66°C. Therefore the $Q_{cooling}$ needed for this application is just under 215 W, which is nearly three times the rating of the module used here. Secondly, this formula assumes that the interface between the Peltier module and the load is 100% thermally efficient, which cannot be true even when using ample thermal paste. Thirdly, this formula assumes that the temperature of the Peltier's heat side can be maintained at exactly 23°C by the heat sink and fan. Both of these conditions induce inefficiencies requiring a higher-power TEC. Peltier elements that are capable of more than 200W of cooling cost on average over $1,000 – way outside the budget for this project.

Taken together, these results indicate that the bed chiller designed herein is unsuitable for incorporation into the bioprinter. Since bed cooling was previously established as a desirable, but optional feature, system integration will progress without this component.

8. Design Integration, Verification, and Validation

8.1 Implementation

Implementation of the final design prototype mainly involved integration of the systems designed during the proof of concept stage. The one exception, however, was the design of a new nozzle assembly suitable for liquid biomaterial printing. The goals of this component were to be lightweight to ease the mass burden on the Z-axis motors, to use disposable components that could be sterilized, and to ease the physical stress on cells induced by printing. To minimize weight, a simple printed L-bracket with a single bore was mounted on the X-axis carriage (Figure AN). Sterile and disposable plastic Luer fittings pinch and lock through the bore hole, providing a sturdy mount. Sterile surgical tubing (Tygon, 3/8" OD) connects the upper Luer connector to the syringe pump. A dispensing needle twists into the lower adapter and serves as the new print heat. Importantly, the diameter of the fluid path extending from the tip of the syringe pump,

through the tubing and adaptors, and out the dispensing needle never constricts below 500 microns, some 50 times the average diameter of the cell. This restricts the size of the dispensing needle to a minimum of 21 Gauge. This wide conduit should shield cells from the fluid shear stresses that occur along the walls, permitting them to stay within the relatively laminar flow region along the center.
Upon assembly of the nozzle assembly, Tests 6.6.2.1 and 6.6.2.3 were performed.

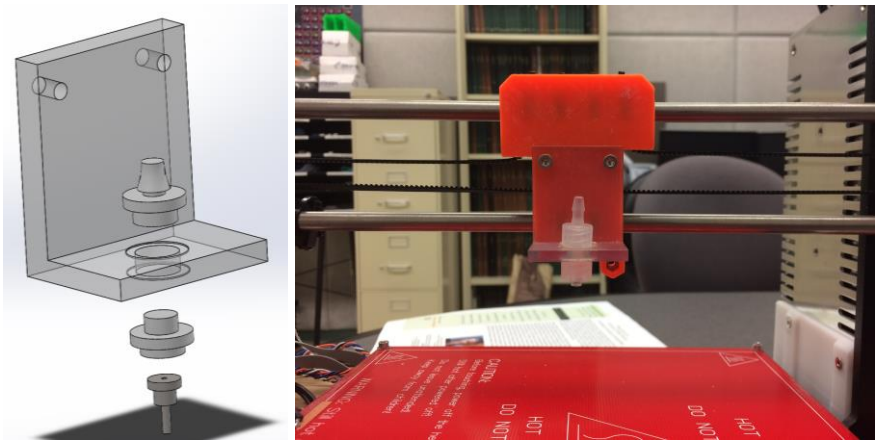


Figure AN. Nozzle assembly designed for bioprinting. Sterile, disposable Luer adapters connect through an L-bracket mounted to the X-axis carriage. Surgical tubing connects the upper adapter to the syringe pump, while a dispensing needle acting as the print head twists into the lower one.

With the syringe pump now functionally connected to the 3-Axis stage via the new nozzle assembly, the next step was to put together the syringe heater. The thermal wraps were first wound tightly along the syringe body, then evenly distributed along the tubing connecting to the nozzle. Zip ties held the wrap tightly in place against the plastic components (Figure AO). This step completes the integration of all the bioprinter hardware; however, significant worked remained to adjust the control board firmware and calibrate the system.

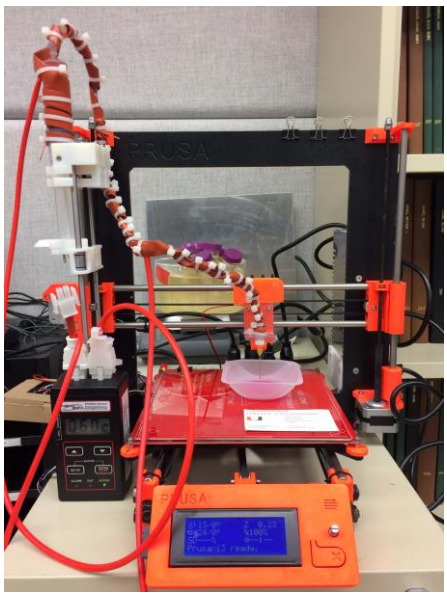
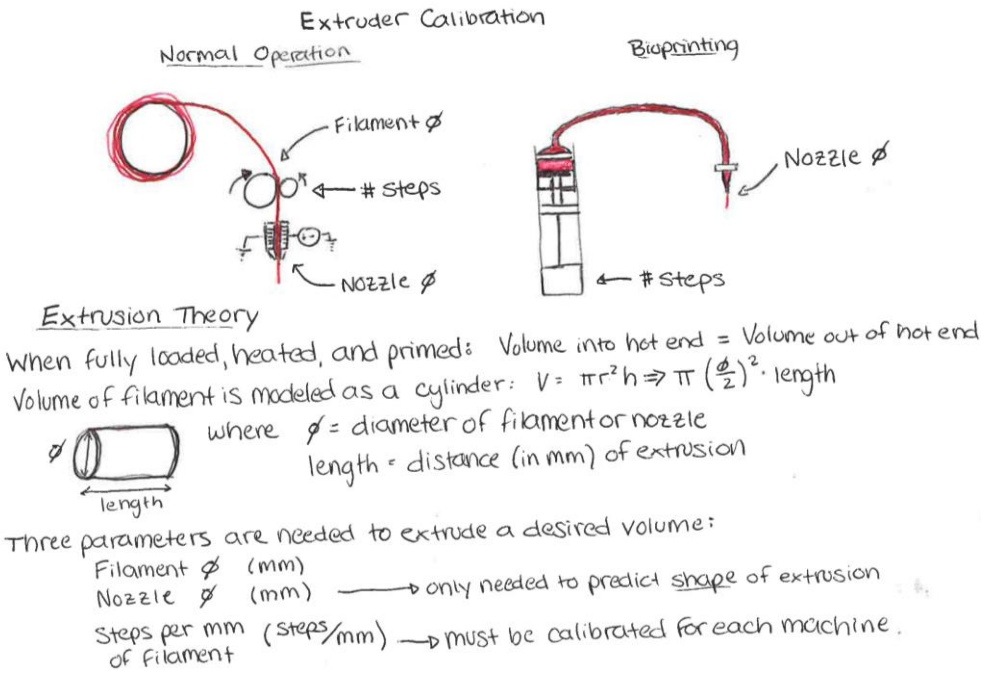


Figure AO. The syringe heating wrap was curls around the entire length of the syringe and tubing to provide even heating.

The Marlin firmware used to operate the control board and the slicing software used to generate G-code were designed specifically for extrusion of plastic filament, not a liquid material. Their architecture provides significant flexibility toward different printer layouts; however, and is amenable to modification. Doing so requires an understanding of how the slicer calculates extrusion length from an STL file and how the firmware translates that desired length into stepper motor commands.

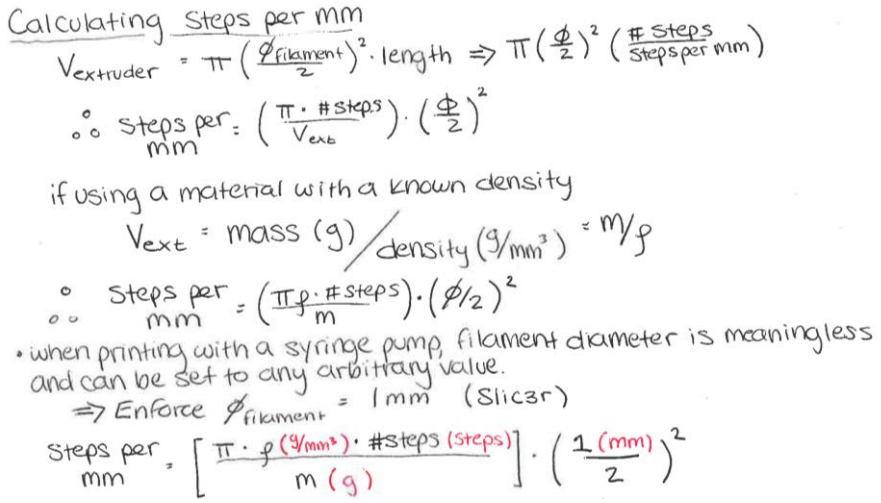
The G-code output from the slicer expresses plastic deposition in terms of an extrusion distance in mm and an extrusion rate in mm/s. These values should be thought of as surrogates for extruded volume (mm^3) and volumetric extrusion rate (mm^3/s). Extrusion distance simply refers to the length of solid plastic filament that needs to be fed into the hot end to produce the correct amount of liquid plastic at the nozzle. The rate parameter ensures that that volume exits the nozzle when the print head has moved to the correct location. Only two parameters are needed to calculate these values: the filament diameter and the nozzle diameter. Both volume and rate are determined using the continuity equation applied to fluid dynamics. Essentially, the volume of material into the hot end must equal the volume extruded, and the volumetric rate into the hot end must equal the volumetric rate out.



With the extrusion distance and rate determined by the slicer, the firmware must then translate these codes into stepper motor commands. This calculation requires a single parameter, the calibration factor in steps/mm. In this case, steps/mm refers to the number of steps the extruder stepper motor must take to push 1 mm of filament into the hot end.

With the modified bioprinting setup, filament diameter and steps/mm are meaningless since the diameter of fluid through the cylinder, tubing, and nozzle assembly is not consistent. However, if a filament diameter value is set, the calibration factor can be re-calculated to make up for the disparity. This constitutes the first part of Test 6.6.2.4.

First, a filament diameter of 1 mm was enforced in the slicer. Then, a volume of fluid was extruded with a set number of steps and measured using its mass and density. The extruded fluid was modeled as a cylinder with a diameter of 1 mm, equal to that used in the slicer. Therefore, the "extruded distance in mm," equivalent to the length of plastic filament pushed into the hot end was calculated. With this information, the calibration factor in steps/mm could be determined.



The calibration factor was then set in the firmware by making the following change:

From: *#define DEFAULT_AXIS_STEPS_PER_UNIT {100,100,3200/0.8,236}*
To: *#define DEFAULT_AXIS_STEPS_PER_UNIT {100,100,3200/0.8,9}*

It is worth noting that the nozzle diameter parameter used by the slicer is functionally equivalent to the needle inner diameter used in the bioprinter. This value is only needed to compute extrusion rate, since volume is determined solely by the extrusion distance and filament diameter.

The accuracy of the determined calibration factor was then assessed in the second portion of Test 6.6.2.4. With the maximum extrusion rate in steps/s having been evaluated previously, this value was then transformed into mm/s according to the calibration factor. It could then be defined in firmware to ensure that print speed would never exceed the capability of the syringe pump:

From: *#define DEFAULT_MAX_FEEDRATE {500, 500, 3, 150}*
To: *#define DEFAULT_MAX_FEEDRATE {500, 500, 3, 100}*

With these edits made and flashed onto the control board, the bioprinter was fully functional! However, a large number of additional slicer parameters, such as desired extrusion width and maximal print speeds, needed to be optimized by testing. Verification Test 6.6.2.5 serves to demonstrate the results of this optimization, while Validation Test 6.6.3.1 presents a real bioprinted construct.

8.2 Results

8.2.1 3-Axis Stage & Nozzle Assembly

Resolution is assessed again upon integration of the new nozzle assembly (Figure AP). Raw sample data from the X-, Y-, and Z-Resolution tests is provided below (Figure AQ). A larger, representative sample of data can be found in Appendix A6. All motion commands were for a target distance of 10 mm. X- and Y-direction movements were altered to expedite testing. A histogram of movement error for all data points is provided in Figure AR. An error of 0 indicates a highly accurate motion, while precision is inversely related to the spread of the histogram. Lastly, the accuracy expressed as the average absolute error and percent error and the precision as the standard deviation of error are summarized in Table 16.

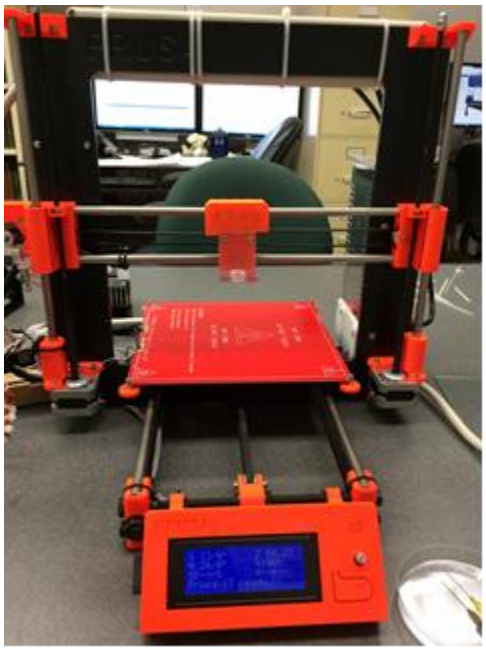


Figure AP. Experimental setup for motion resolution testing with the new nozzle assembly.

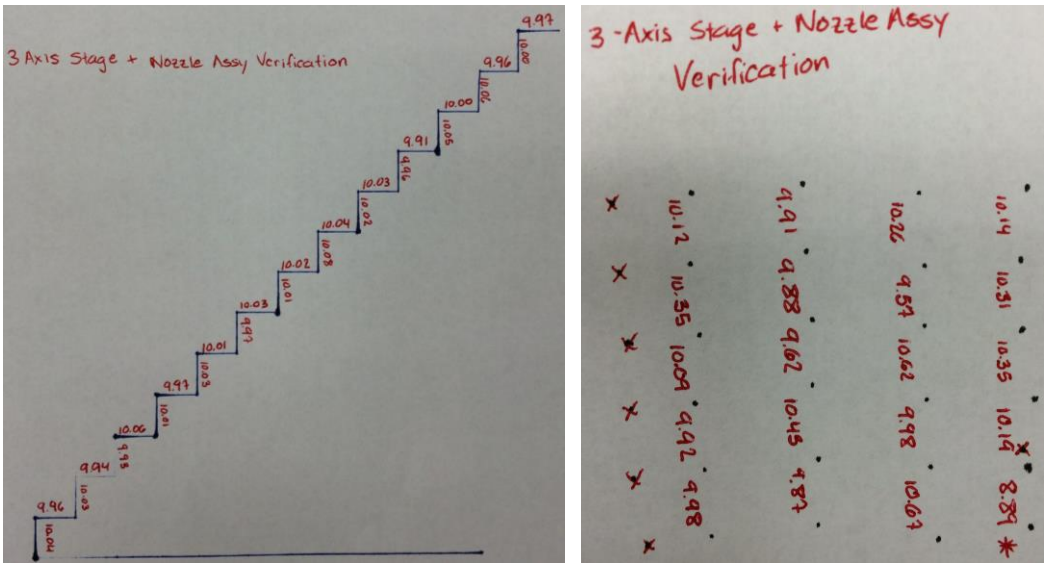


Figure AQ. Sample data for X-, Y-, and Z- resolution.

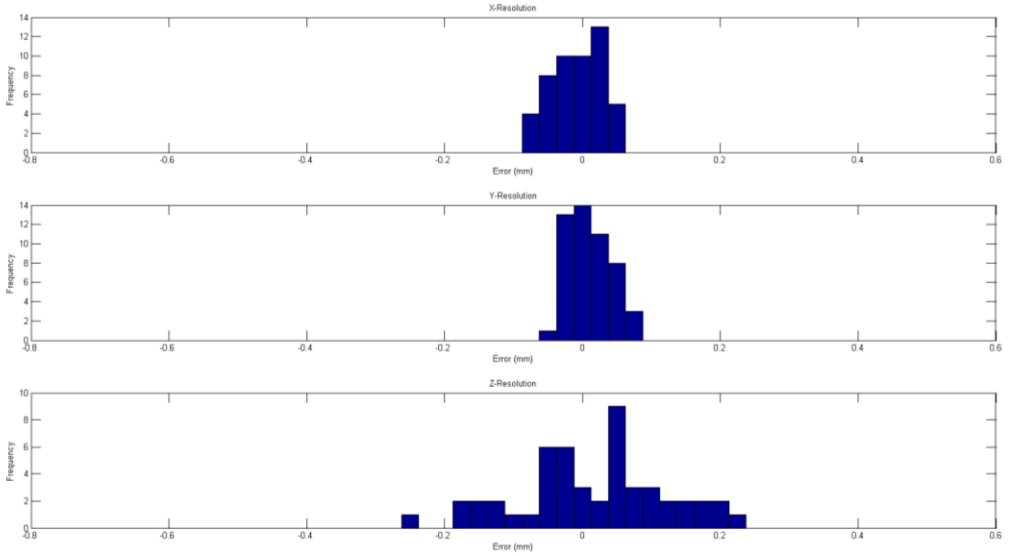


Figure AR. Histograms of X-, Y-, and Z-movement error.

Resolution Summary - Stage + Nozzle Verification			
Dimension	**Accuracy: Avg Error (mm)**	**Accuracy: % Error**	**Precision (mm)**
X	0.04	0.28%	0.04
Y	0.03	0.33%	0.04
Z	0.26	2.72%	0.13

Table 16. Data summary for 3-axis stage and nozzle assembly integrated component testing.

8.2.2 Syringe Pump, Syringe Heater, & Control Board

Maximal and minimal extrusion rate was again determined upon integration of the syringe heater to the pump controlled by the RAMBo board using 40 wt% gelatin as a model viscous biomaterial. Heating was set to 38.5^{o}C, with the intention that the gelatin would remain at 37^{o}C, warm enough to keep the gelatin liquid, but cool enough to be suitable for biological materials. The maximum and minimum extrusion velocities were determined by visual inspection while ramping the step rate up or down. Results, including the expected or hypothesized failure mode, are presented in Table 17.

Syringe Pump – Heater Compatibility: Speed Testing		
Speed	**Failure**	**Rate (steps/s)**
Max	Viscosity-induced motor stall	900
Min	Gelation in situ	N/A

Table 17. Data summary for the integrated syringe pump, control board, and syringe heater component testing.

8.2.3 Syringe Pump, Syringe Heater, Nozzle Assembly, & Control Board: System Calibration

Calibration of the entire integrated bioprinter required a two-step process, as described in detail in the Implementation section. First, the pump motor was actuated for a defined number of steps and the mass of extruded water determined with a balance. Using the density of water, the extruded volume was then calculated. The number of steps was varied to minimize some sources of measurement error identified during proof of concept testing. The calibration factor in steps/mm was then determined for each sample by modeling the extrusion as a cylinder with a diameter equal to the filament diameter parameter enforced in the slicer. A histogram of calibration factors is provided in Figure AS.

The calibration factor in steps/mm was then edited into the printer firmware and re-flashed onto the control board. To test accuracy, a volume equal to 1000 mm of filament (9000 steps) was extruded and measured. The error in extruded volume was calculated by again measuring the mass of water and dividing by its room temperature density. A histogram of volume error is provided in Figure AS. An error of 0 indicates a highly accurate extrusion, while precision is inversely related to the spread of the histogram. A larger representative sample of data can be found in Appendix A7. The calibration factor determined in the first step, the accuracy expressed as the average absolute error and percent error, and the precision as the standard deviation of error are summarized in Table 18.

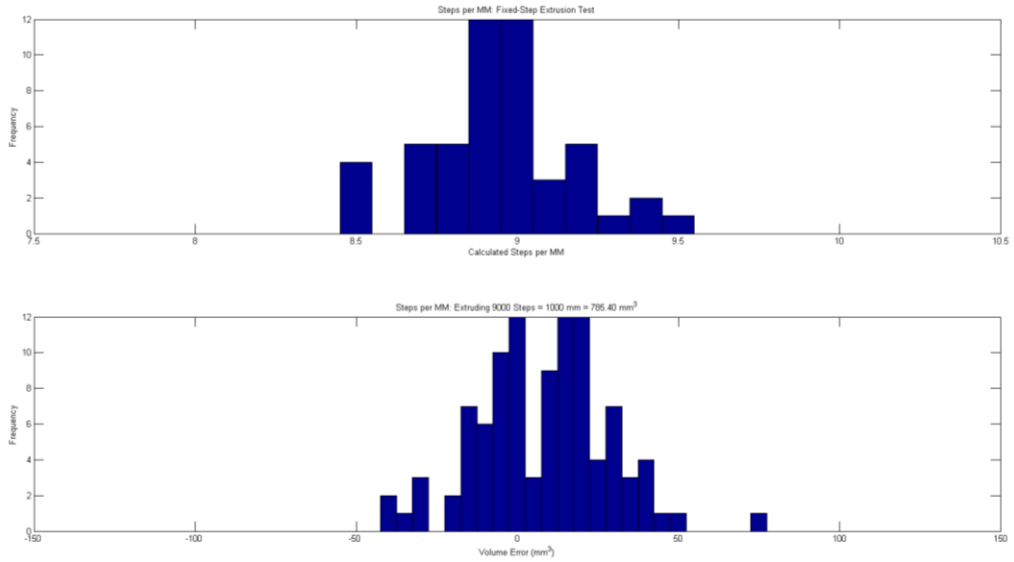


Figure AS. (Top) Histogram of calculated calibration factors based on fixed-step extrusions. (Bottom) Histogram of error between intended and measured volume upon 1,000 mm (9,000-step) extrusion.

System Calibration Summary	
Filament Diameter (Enforced-Slic3r) (mm)	1
Steps per mm	9
Accuracy: Average Error (mm^3)	25.64
Accuracy: % Error	2.89%
Precision (std) (mm^3)	21.36

Table 18. Data summary for system calibration.

8.2.4 Verification Resolution Testing

Following calibration of the integrated bioprinter system, printing parameters contained within the slicer software were systematically varied to optimize resolution. To verify that the bioprinter met the resolution specifications delineated during design development, the model depicted in Figure AT was printed. This model contains simple, two-dimensional architecture of increasing dimensions with a high aspect ratio – ideal for both optical measurement and assessment of printing resolution. An example of the optical measurements taken of sample print is depicted in Figure AU. A larger representative sample of data is provided in Appendix A8. A histogram of dimensional error for all data points is provided in Figure AV. An error of 0 indicates a highly accurate print, while precision is inversely related to the spread of the histogram. Lastly, the accuracy expressed as the average absolute error and percent error and the precision as the standard deviation of error are summarized in Table 19.

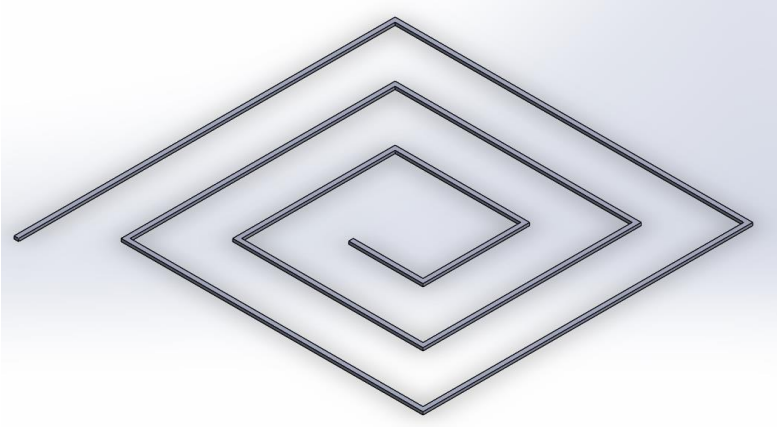


Figure AT. CAD model used for verification testing.

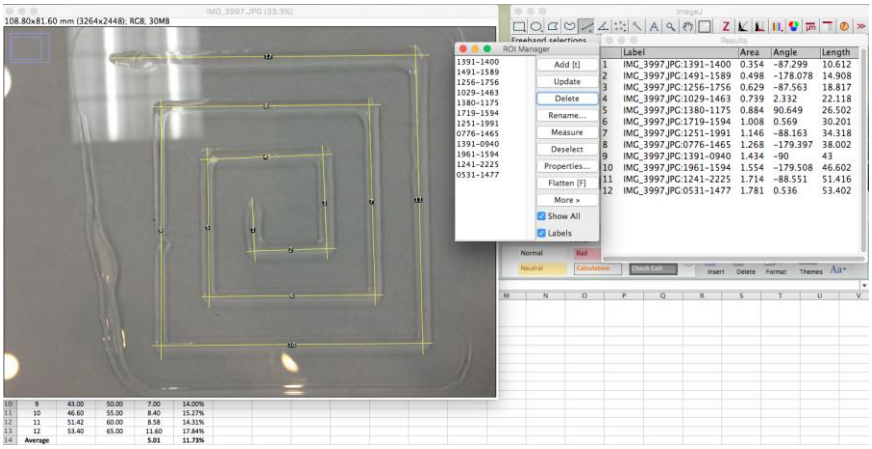


Figure AU. ImageJ software is used to make measurements of the printed construct. Dimensional error is calculated with respect to the input CAD model.

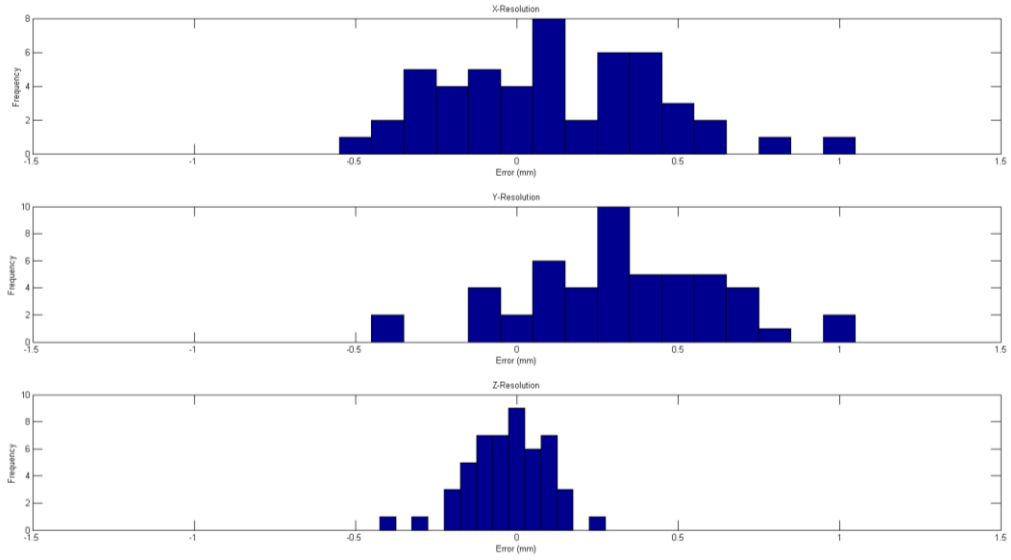


Figure AV. Histograms of X-, Y-, and Z-dimensional error.

Verification Resolution Testing: Data Summary			
Dimension	**Accuracy: Avg Error (mm)**	**Accuracy: % Error**	**Precision (mm)**
X	0.13	1.69%	0.30
Y	0.33	2.05%	0.28
Z	0.01	12.05%	0.06

Table 19. Data summary for verification resolution testing.

8.2.5 Gelatin Bioprinting Resolution

Validation testing was conducted by printing a fully three-dimensional model with gelatin, which contained features of differing sizes in all dimensions (Figure AW). Assessment of resolution was determined optically, as before, demonstrated in Figure AX. A larger representative sample of data is provided in Appendix A9. A histogram of dimensional error for all data points is provided in Figure AY. An error of 0 indicates a highly accurate print, while precision is inversely related to the spread of the histogram. Lastly, the accuracy expressed as the average absolute error and percent error and the precision as the standard deviation of error are summarized in Table 20.

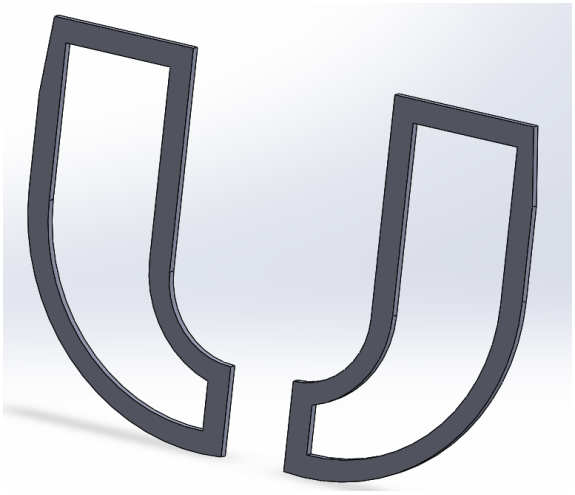


Figure AW. CAD model used for validation testing.

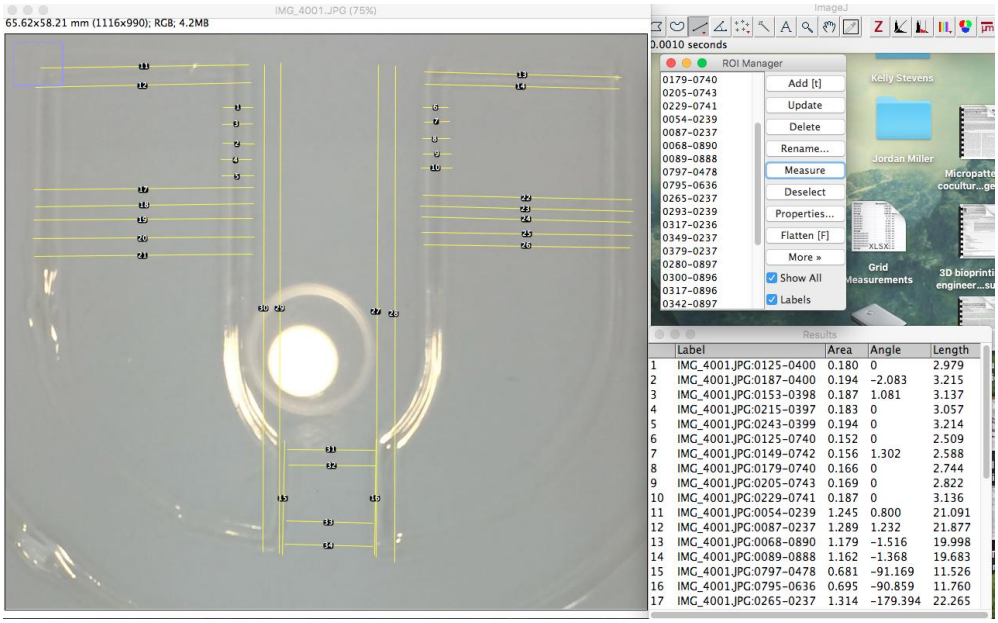


Figure AX. ImageJ software is used to make measurements of the printed construct. Dimensional error is calculated with respect to the input CAD model

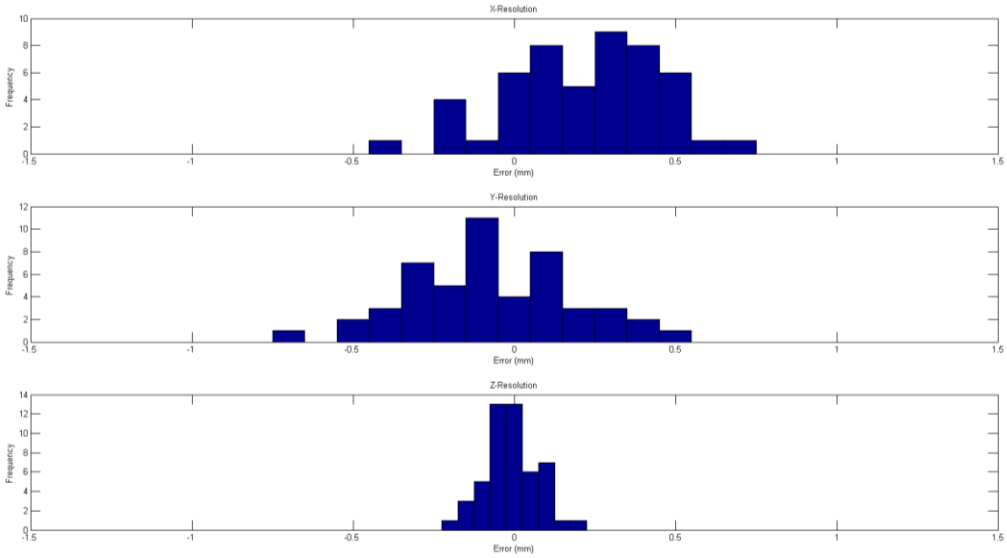


Figure AY. Histograms of X-, Y-, and Z-dimensional error.

Validation Resolution Testing: Data Summary			
Dimension	**Accuracy: Avg Error (mm)**	**Accuracy: % Error**	**Precision (mm)**
X	0.30	4.42%	0.31
Y	-0.06	0.85%	0.31
Z	-0.01	9.43%	0.10

Table 20. Data summary for validation resolution testing.

9. Discussion

The results presented in the preceding section demonstrate the feasibility of this design for printing thermoresponsive biomaterials in three dimensions at extremely low cost. For the purposes of an organized analysis, the results described in Section 8 will be analyzed sequentially first, followed by a discussion of the performance of the entire system. Shortcomings will then be discussed, along with recommendations for future improvements.

The first order of work before assembling the different bioprinter subsystems was the design of a fluid-handling print head. First, the print head needed to be relatively lightweight, to avoid adding any extra mass to the X-axis. Any extra mass on the X-axis not only adds work for the Z-axis motors, but also, and more importantly, it adds to the inertia to the X-axis carriage, which makes rapid back-and-forth movements difficult due to increased momentum. Secondly, the print head needed to use components that were sterile, for future cell-printing purposes. Metal fittings that could be sterilized in an autoclave could have been used, but disposable Luer adaptors provided a much simpler and more flexible option. Any variety of different size needles or tubing could be interchanged at will, permitting a wider variety of materials to be printed. Finally, the nozzle assembly design needed to minimize the shear stresses that would be experienced by cells during the printing process. This was accomplished by ensuring that the fluid conduit from the syringe tip to the dispensing needle never constricted below 500 microns in diameter, about 50 times the average cell diameter. Thus, the smallest dispensing needle, and therefore nozzle size, that can be used is 21 Gauge. The nozzle diameter defines the smallest feature that the printer is capable of producing and puts a limit to resolution. Models intended for printing must take this limit into consideration. Upon assembly, the spatial resolution of the 3-axis stage was re-assessed, to ensure that new and lighter print head wouldn't cause any problems. Accuracy in all three dimensions actually increased with this new setup, qualifying the bioprinter for full assembly.

The capability of the stepper motor driver on the control board to power the syringe pump was initially characterized during proof of concept testing; however, that test had to use a highly non-viscous fluid for pumping because the syringe heater had not been put together. Since the capability of the heater to maintain a real biomaterial in its liquid state was then established, this capability test was repeated using 40 wt% gelatin. The increased viscosity of this fluid, along with any elastic recoil in the tubing or syringe, took their toll on the stepper motor, limiting maximum speed to 900 steps/s, down from 1,350 steps/s previously. However, the motor can still turn at 270 RPM, well above any speed needed during printing. Upon determination of the appropriate calibration factor (steps/mm) for the printer, this maximum extrusion speed was converted to mm/s (900 steps/s = 100 mm/s) and set as the upper limit in the firmware.

With all of the bioprinter subsystems now fully qualified, the entire system could then be put together in its final form. Before anything useful could be printed, however, the discrepancies between a system intended to extrude plastic filament and one pumping a liquid needed to be ironed out.

Both the slicing software and the control board firmware were designed to operate almost any kind of FDM printer with minimal modification. Slic3r generates G-code that can be interpreted by almost any printer, regardless of the mechanics of the axes or extruder. Instead of writing commands specifying how many steps to turn each motor, the code calls

for motion in a certain direction for a certain distance, leaving the conversion into steps to the firmware. This conversion is dependent on the mechanics of the printer, such as the step size of the motor, the type of timing belt used, and any gear ratios that may be implemented, so this makes the G-code applicable universally. Additionally, Slic3r only requires the user to input parameters that are easy to measure on his or her printer: shape and dimensions of the print bed, nozzle diameter, and filament diameter. As described in detail in the implementation section, only the filament and nozzle diameters are needed to calculate the amount and rate of filament needed to produce a desired structure. This is based on the steady-state continuity equation as applied to fluid dynamics:

$$V_{in\ extruder} = V_{out\ extruder} \rightarrow A_{cross\text{-}section\ in} * L_{in} = A_{cross\text{-}section\ out} * L_{out}$$

$$\text{Therefore, } \pi * L_{filament} * (D_{filament}/2)^2 = \pi * L_{nozzle} * (D_{nozzle}/2)^2 = V_{extruded}$$

$$\text{Or, } V_{extruded} = \alpha * L_{filament} \quad \text{for } \alpha = \pi * (D_{filament}/2)^2$$

The length (L in mm) of filament, and hence volume (V in mm^3), pressed into the hot end must equal the volume of plastic extruded out of the nozzle, where volume is modeled as a cylinder defined by a diameter (D in mm) and length. Thus, extruded volume can be modeled as a constant (α in mm^2) times the length of filament. Taking the derivative of the first equation permits calculation of the volumetric flow rate:

$$d(A_{in}L_{in})/dt = d(A_{out}L_{out})/dt \rightarrow A_{in}v_{in} = A_{out}v_{out}$$

$$\text{So, } \pi * (D_{filament}/2)^2 * d(L_{filament})/dt = \pi * (D_{nozzle}/2)^2 * d(L_{nozzle})/dt$$

$$\text{Becomes: } \pi * (D_{filament}/2)^2 * v_{filament} = \pi * (D_{nozzle}/2)^2 * v_{nozzle}$$

$$\text{Rearranged: } v_{nozzle} = \{(D_{filament}/2)^2 / (D_{nozzle}/2)^2\} * v_{filament}$$

$$\text{Or, } v_{nozzle} = \beta * v_{filament} \quad \text{for } \beta = (D_{filament}/2)^2 / (D_{nozzle}/2)^2$$

The extrusion rate (v in mm/s) out of the nozzle depends directly on the introduction rate of the filament, scaled by the ratio of cross sectional areas (β, unitless). With these equations, Slic3r can generate commands to introduce the proper length of filament at the ideal rate into the hot end to extrude plastic in the correct locations.

Marlin firmware contains all of the conversion factors needed to convert this G-code into motor commands. Commands for the three axes and the extruder specify a direction (forward or backward), a distance (mm), and a speed (mm/s). These are converted into a motor rotation direction (clockwise or counterclockwise), a discrete number of steps, and a step rate (steps/s), according to printer-specific parameter in steps/mm. Importantly, extrusion commands are always communicated with reference to the distance of filament needed, not regarding the length of plastic coming out of the nozzle.

The conversion factors established in firmware for the three axes did not need to be altered, as the accuracy of the stage had been verified previously. However, the factor calibrating extrusion needed to be modified such that the proper volume of material would be deposited. Filament diameter is meaningless when using a syringe pump, so a value of 1 mm was enforced in Slic3r. As such, the extruded volume becomes:

$$V_{extruded} = (\pi/4) * L_{filament}$$

$$\text{Rearranged: } L_{filament} = (4/\pi) * V_{extruded}\ L_{filament}$$

$$\text{Or: } L_{filament} = (4/\pi) * (m_{extruded} / \rho_{fluid})$$

Therefore, the length of "filament" extruded can be calculated by determining the mass of the extruded fluid. By issuing extrusion commands of a known number of steps and

calculating the length of "filament" extruded, the conversion factor was determined for this system to be approximately 9 steps/mm. Editing this parameter into firmware ensures that the proper volume of fluid is extruded according to standard G-code commands. Fortunately, the rate of extrusion is still defined by the ratio of the "filament diameter" to the nozzle diameter, which is equal to the inner diameter of the dispensing needle.

With the new calibration factor flashed into the firmware, the accuracy of the extrusion pump was then assessed. G-codes calling for a 1,000 mm extrusion, which corresponds to 9000 steps or ~785 mm^3, were issued and the resulting volume was calculated. An error of less than 3% was determined, which was deemed acceptable given the sources of measurement error mentioned during previous pump calibration testing.

With the core printing parameters now dialed in, the system was now ready for verification testing. To doing so, a number of additional Slic3r parameters needed to be systematically dialed in, including extrusion width, infill, layer height, and print speed. These variables allow Slic3r to generate G-code suitable for printing in any pump-able material. For example, increasing extrusion width increases the volume of material deposited during each printing move. This can be helpful to overcome the cohesive properties of the fluid that prevent it from adhering directly to the print bed. Decreasing the print speed allows more time for the biomaterial to solidify, or in this place to cool down and gel, which prevents it from spreading across the print bed. To systematically dial in these variables, simple 2D models were drafted in CAD containing features of various aspect ratios, converted into G-code, and printed with 40 wt% gelatin. Resolution in the XY-plane averaged 300 microns, meeting the specifications for this design.

Validation was performed by printing a fully three-dimensional model using 40 wt% gelatin. Some further dialing in of Slic3r parameters was necessary, especially in defining the optimal layer height when moving vertically. One of the major downsides in printing with thermoresponsive hydrogels is that rigidity of the polymer is a function of both temperature and monomer concentration. While decreasing the print speed permits adequate time for each layer to cool and gel before moving upward, each subsequent stack of liquid material slightly solubilizes the layer beneath it. This is beneficial in that each layer needs to adhere tightly to the one beneath it to form a solid construct; however, it became apparent that the currently polymerizing gelatin being printed tended to void much of its water into the layers below. This resulted in a gradual increase in the water content, or alternatively a decrease in the gelatin concentration, in the lower layers, which induced a loss of rigidity and spreading. Despite this concern, fully three-dimensional structures were printed with a resolution along all three axes on the order of 300 microns, which meets the target for an extrusion bioprinter!

In summary, the goal of building a highly flexible bioprinting platform for an extreme budget was a success. By leveraging existing open-source FDM printing technologies, excellent resolution was obtained without requiring prohibitively expensive components.

Before moving onward, it is necessary to admit the most glaring shortcoming of this work, that is, determination of the viability of cells after printing. Cell death during printing is hypothesized to occur due to three mechanisms, each of which was taken into consideration during the design.[49] First, shear forces can physically damage cell membranes. These forces will be greatest in the narrowest portion of the fluid conduit, the dispensing needle, and will be a function of the apparent viscosity of the biomaterial used. However, biomaterial viscosity improves the resolution of the final print by discouraging spreading. Therefore, care

was taken to ensure that the fluid path was 50 times greater than the average cell diameter to shield it from shear at the wall, and material viscosity was chosen at a moderate value to balance cytotoxicity concerns with resolution aims. Second, cells can be damaged due to thermal conditions. Both heat and cold shock induce a systemic response eventually leading to apoptosis. The flexibility of the syringe heating system permits temperature to be controlled with high precision all the way from the syringe to the nozzle; therefore, the only challenge is to ensure that the chosen biomaterial possesses a suitable viscosity at a cytocompatible temperature. Gelatin is an excellent example, since it liquefies at the moderate temperature of 40^{o}C, which is only slightly above the ideal cell culture temperature. Third, the amount of time that a cell remains either within the syringe or on the build plate can be considered a mode of toxicity. Any variety of insults, airborne pathogens, evaporation, or light exposure, could result in cellular death. Therefore, speeding the total time needed to print a construct would help to keep cells alive. In this design, print speed was fundamentally limited by the cooling kinetics of the gelatin – which likely could have been improved if the bed chilling subsystem had been upgraded. With these design considerations in mind, it is hypothesized that this bioprinter should be able to print cells with relatively high viability; however, this capacity was not directly verified.

Incorporation of a syringe pump presented additional challenges that were not anticipated before testing, resulting in a second shortcoming of this design. In particular, flow was often non-uniform during printing. The key mathematical assumption behind the generation of G-code is that the volume of filament pressed into the hot end must equal the volume extruded out of the nozzle. In the case of this bioprinter, this relationship requires the volume of fluid squeezed out of the syringe to be equivalent to the volume out of the dispensing needle. Additionally, calculation of extrusion rates assumes that the material flows without hysteresis. Therefore, extrusion rates should depend only on the instantaneous syringe pump speed; there should be no lag between actuating the pump and extruding the material.

Two factors complicate these assumptions in this design. First, both heated filament and liquid biomaterial experience "ooze." This term describes the non-intended flow of material out of the nozzle due to either thermal expansion or gravity. The former represents an insignificantly small contribution to ooze, given the relatively mild temperatures used in the bioprinter. The latter, however, seemed to have a large and non-uniform effect. Only viscous forces and shear at the boundaries of fluid conduits act to oppose the effects of gravity. The use of relatively large-diameter tubing to connect the syringe pump and nozzle did not provide sufficient shear to prevent ooze; rather, material could be seen running through portions of the tubing that were positioned more vertically. Additionally, the thermal effects on apparent biomaterial viscosity need to be considered in this context. The second major factor complicating extrusion was the compliance of the tubing. The assumption that volume pumped equals volume extruded depends implicitly upon the total volume of the fluid conduit connecting the inlet and outlet remaining constant. In a normal FDM printer, this conduit is very short and constructed of both stiff PEEK and aluminum to match that condition; however, the bioprinter uses relatively elastic Tygon which can dilate under pressure. Thus, during printing, the apparent diameter of the tubing likely increases as a function of extrusion rate, which would manifest as a time-delay in extrusion out of the nozzle. That is, when printing a thick solid line at a rapid speed, the volume of biomaterial in the tubing initially increases as it widens, instead of being extruded at the nozzle. Towards the end of the printing movement, this trapped volume is squeezed out due to elastic recoil of the tubing, resulting in over-extrusion.

Both of these concerns could be addressed by re-designing the nozzle assembly to eliminate the tubing connecting the syringe pump to the dispensing needle. Instead, the syringe pump should be mounted directly onto the X-axis carriage. By simply reducing the length of the fluid conduit in this way, the effects of compliance on flow rate should be minimized. The effects of ooze could be minimized by using a smaller Gauge needle to enhance shear. In addition, Slic3r contains a feature to retract a certain volume of fluid at regular intervals during printing – which can be calibrated to match the rate of ooze. Further, the temperature, and therefore viscosity, of the biomaterial would be much easier to manage over the much shorter fluid conduit distance. Upon further research, a few open-source conceptual designs for carriage-mounted syringe pumps were discovered (Figure AZ). While intended for extrusion of highly viscous fluids, called pastes, at room temperatures, incorporation of a syringe heater should be possible.

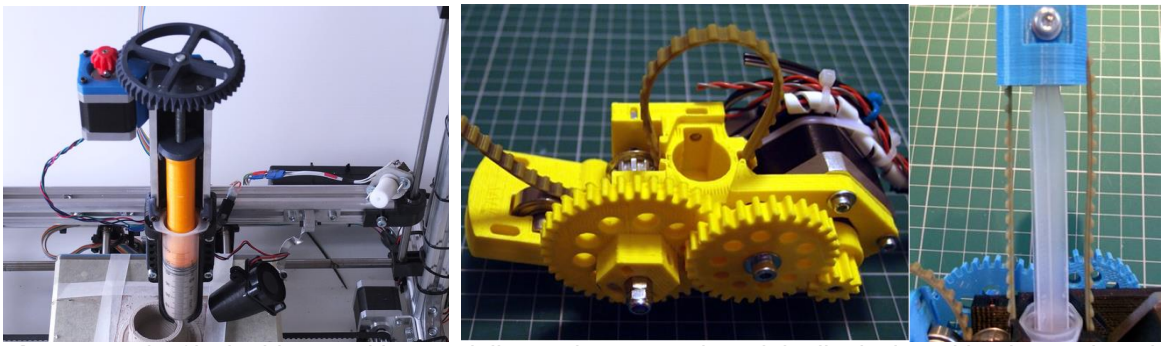


Figure AZ. (Left) Liquid Deposition Modeling syringe extruder originally designed by the +Lab at the Polytechnic University of Milan.[50] (Right) The Universal Paste Extruder originally designed by Richard Horne, a core RepRap developer in the UK.[51]

While redesigning the syringe pump to mount directly on the X-axis carriage may do much to address the identified fluid handling concerns, other pump options ought to be investigated as well. An ideal pump would exhibit purely volumetric positive displacement, where pump motor rotation can be directly correlated to extruded volume. To address oozing, an optimal pump would additionally prevent fluid from moving through the line under its own power, i.e. by eliminating any continuous fluid pathways from the reservoir to the nozzle. Two potential candidates that fit these criteria are the Moineau pump and the gear pump (Figure BA). The Moineau pump involves a helical rotor sealing against an appropriately-scaled stator, forming fixed-sized cavities that move toward the outlet. Similarly, a gear pump forms cavities between rotating spur gears and an appropriately-sized housing.

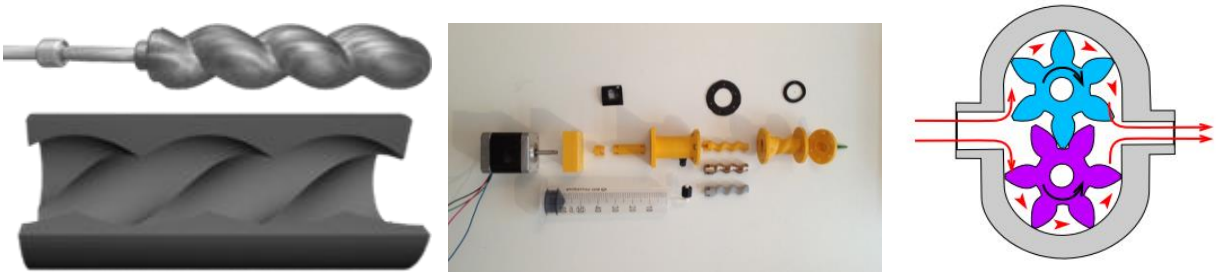


Figure BA. (Left) Schematic of a Moineau pump, demonstrating the helical rotor that fits inside its stator. Non-overlapping cavities form between the two during rotation.[52] (Middle) Basic design for a 3D printed Moineau pump driven with a stepper motor.[53] (Right) Schematic of an external gear pump, in which fluid is drawn in by low pressure on the left and is expelled under high pressure on the right.

One of the key advantages of this bioprinter design is that a huge variety of different materials could be used; however, only a single thermoresponsive polymer was demonstrated. Thermal gelation is a common feature among both natural and synthetic biomaterials, including collagen and decellularized extracellular matrices. However, other methods of gelation could be readily implemented, in particular enzyme-mediated or ionic polymerization.[54] For example, a fibrin hydrogel polymerizes in the presence of its enzyme, thrombin, and alginate solutions gel with divalent cations such as calcium.[55] Such a scheme would be trivial to implement using this printer by redesigning the build plate as an open box. By filling the box with a solution containing the polymerizing agent, viscous biomaterials could be printed and gelled *in situ* without the need for precise thermal control. Taken a step further, this polymerization solution could also serve to support intricate structures during printing that may lack the structural integrity to stand on their own. For example, the Feinberg group describes a method to print alginate into a calcium-rich support bath filled with shear-thinning gelatin microspheres.[17] This bath holds the printed construct in place, preventing the alginate from spreading as it cures and can then be melted away with gentle warming.

The device designed herein represents an excellent first step toward producing a truly flexible and modular bioprinter, capable of precision fluid handling at extreme low cost. With the identification of a few key hardware changes, mostly centered on the fluid handling system, it is expected that this design will continue to improve in the future.

10. Conclusion

In conclusion, this report details the development of a low-cost and highly flexible bioprinter suitable for research use. By leveraging an array of open-source technologies, this machine manages the tradeoffs between print resolution, overall cost, and material flexibility to enable fabrication of a wide variety of biologic constructs. The basis for the design centered on a RepRap-based FDM printer for the control of motion in three dimensions to an accuracy of less than 300 microns. Biomaterial extrusion occurred using an open-source syringe pump powered by a stepper motor, which was able to handle viscous biomaterials to an accuracy of less than 3%. Distributed heating was provided along the length of the syringe pump and fluid handling elements, providing thermal control of the chosen biomaterial during printing. Efforts to design a cooling system for the print bed to encourage rapid biomaterial gelation were eventually unsuccessful due to excessive cost. Upon integration of these systems, calibration of fluidic components, and optimization of software parameters, fully three-dimensional constructs were printed in a representative hydrogel with an accuracy less than 300 microns.

11. Acknowledgements

A big thanks goes out to Dr. Ramon Montero for lots of support throughout this design, especially with regard to troubleshooting the firmware. (Soon-to-be) Dr. Lukas Jaworski also deserves credit for some excellent and thought-provoking conversations, especially with regard to the design of control algorithms.

12. References

1. Murphy, S. V & Atala, A. 3D bioprinting of tissues and organs. *Nat. Biotechnol.* **32,** 773–785 (2014).
2. Ouyang, L. *et al.* Three-dimensional bioprinting of embryonic stem cells directs highly uniform embryoid body formation. *Biofabrication* **7,** 044101 (2015).
3. Johnson, B. N. *et al.* 3D printed nervous system on a chip. *Lab Chip* **16,** 1393–1400 (2016).
4. EnvisionTec 3D-Bioplotter. Available at: http://envisiontec.com/3d-printers/3d-bioplotter/. (Accessed: 1st January 2016)
5. Organovo NovoGen MMX Bioprinter. Available at: http://organovo.com/science-technology/bioprinting-process/. (Accessed: 1st January 2016)
6. Mironov, V. *et al.* Biofabrication: a 21st century manufacturing paradigm. *Biofabrication* **1,** 022001 (2009).
7. Wu, D., Fang, N., Sun, C. & Zhang, X. Adhesion force of polymeric three-dimensional microstructures fabricated by microstereolithography. *Appl. Phys. Lett.* **81,** 3963 (2002).
8. Kitson, P. J., Symes, M. D., Dragone, V. & Cronin, L. Combining 3D printing and liquid handling to produce user-friendly reactionware for chemical synthesis and purification. *Chem. Sci.* **4,** 3099–3103 (2013).
9. Jakus, A. E. *et al.* Three-Dimensional Printing of High-Content Graphene Scaffolds for Electronic and Biomedical Applications. *ACS Nano* **9,** 4636–4648 (2015).
10. Modern Meadow: Bioprinted Meat. Available at: http://www.modernmeadow.com/#food. (Accessed: 1st January 2016)
11. Cho, D.-W. Intelligent Manufacturing Systems Lab. Available at: http://ims.postech.ac.kr/bbs/board.php?bo_table=sub1_1. (Accessed: 1st January 2016)
12. Xia, Y. N. & Whitesides, G. M. Soft lithography. *Annu. Rev. Mater. Sci.* **28,** 153–184 (1998).
13. Weibel, D. B., DiLuzio, W. R. & Whitesides, G. M. Microfabrication meets microbiology. *Nat. Rev. Microbiol.* **5,** 209–218 (2007).
14. RepRap. Available at: http://reprap.org/. (Accessed: 1st January 2016)
15. BioCurious - BioPrinter Community Project. Available at: http://biocurious.org/projects/bioprinter/. (Accessed: 1st January 2016)
16. Feinberg, A. W. Regenerative Biomaterials & Therapeutics Group. Available at: http://regenerativebiomaterials.com/. (Accessed: 1st January 2016)
17. Hinton, T. J. *et al.* Three-dimensional printing of complex biological structures by freeform reversible embedding of suspended hydrogels. *Sci. Adv.* **1,** 1–10 (2015).
18. Angelini, T. E. Soft Matter Research Laboratory. Available at: http://plaza.ufl.edu/t.e.angelini/. (Accessed: 1st January 2016)
19. Bhattacharjee, T. *et al.* Writing in the granular gel medium. *Sci. Adv.* **1,** e1500655 (2015).
20. Fang, N. X. Nanophotonics and 3D Nanomanufacturing Laboratory. Available at: http://web.mit.edu/nanophotonics/index.htm. (Accessed: 1st January 2016)
21. Sun, C., Fang, N., Wu, D. M. & Zhang, X. Projection micro-stereolithography using digital micro-mirror dynamic mask. *Sensors Actuators, A Phys.* **121,** 113–120 (2005).
22. Miller, J. S. Physiologic Systems Engineering and Advanced Materials Laboratory. Available at: http://millerlab.rice.edu/. (Accessed: 1st January 2016)
23. Hribar, K. C., Soman, P., Warner, J., Chung, P. & Chen, S. Light-assisted direct-write of 3D functional biomaterials. *Lab Chip* **14,** 268–75 (2014).
24. Texas Instruments - Digital Micromirror Device. Available at: http://www.ti.com/lsds/ti/analog/dlp/how-dlp-works.page. (Accessed: 1st January 2016)
25. Tufts University - 3D Printing Handbook. Available at:

http://maker.tufts.edu/handbooks/3d-printing/about. (Accessed: 1st January 2016)
26. RepRap: Prusa i3. Available at: http://reprap.org/wiki/Prusa_i3. (Accessed: 1st January 2016)
27. Prusa i3 STL Files. Available at: http://www.thingiverse.com/thing:119616. (Accessed: 1st January 2016)
28. RepRap: Mini-RAMBo. Available at: http://reprap.org/wiki/MiniRambo. (Accessed: 1st January 2016)
29. UltiMachine: Mini-RAMBo. Available at: https://ultimachine.com/products/mini-rambo-1-3. (Accessed: 1st January 2016)
30. OpenSCAD. Available at: http://www.openscad.org/. (Accessed: 1st January 2016)
31. Slic3r. Available at: http://slic3r.org/. (Accessed: 1st January 2016)
32. Pronterface. Available at: http://www.pronterface.com/. (Accessed: 1st January 2016)
33. RepRap: Marlin. Available at: http://reprap.org/wiki/Marlin. (Accessed: 1st January 2016)
34. GitHub: Marlin. Available at: https://github.com/MarlinFirmware/Marlin. (Accessed: 1st January 2016)
35. Syringe Pump Diagram. Available at: http://www.pmdcorp.com/news/articles/html/precision_fluid_handling_deep_dive.html. (Accessed: 1st January 2016)
36. Peristaltic Pump Diagram. Available at: http://www.wmcpumps.com/index.php/support/sheet-1-peristaltic-pumps. (Accessed: 1st January 2016)
37. Pearce, J. M. Michigan Tech's Open Sustainability Technology Lab. Available at: http://www.mse.mtu.edu/~pearce/Index.html. (Accessed: 1st January 2016)
38. Wijnen, B., Hunt, E. J., Anzalone, G. C. & Pearce, J. M. Open-Source Syringe Pump Library. *PLoS One* **9,** e107216 (2014).
39. Adafruit Peltier Assembly. Available at: https://www.adafruit.com/products/1335. (Accessed: 1st January 2016)
40. Adafruit 10k Thermistor. Available at: https://www.adafruit.com/products/372. (Accessed: 1st January 2016)
41. PowerSwitch Tail II Relay. Available at: http://www.powerswitchtail.com/Pages/default.aspx. (Accessed: 1st January 2016)
42. NewEra Syringe Heater. Available at: http://www.syringepump.com/heater.php. (Accessed: 1st January 2016)
43. Prusa Build Guides. Available at: http://prusa3d.dozuki.com/c/English_manuals. (Accessed: 1st January 2016)
44. Open-Source Syringe Pump. Available at: http://www.appropedia.org/Open-source_syringe_pump. (Accessed: 1st January 2016)
45. Syringe Pump STL Files. Available at: https://www.youmagine.com/designs/syringe-pump. (Accessed: 1st January 2016)
46. Arduino PID Library. Available at: http://playground.arduino.cc/Code/PIDLibrary. (Accessed: 1st January 2016)
47. PID Tutorial. Available at: http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/. (Accessed: 1st January 2016)
48. TEC Guide. Available at: http://www.meerstetter.ch/compendium/tec-peltier-element-design-guide. (Accessed: 1st January 2016)
49. Nair, K. *et al.* Characterization of cell viability during bioprinting processes. *Biotechnol. J.* (2009). doi:10.1002/biot.200900004
50. Levi, M. +Lab at Polytechnic University of Milan. Available at: http://www.piulab.it/. (Accessed: 1st January 2016)
51. Horne, R. Universal Paste Extruder. Available at: http://www.thingiverse.com/thing:20733. (Accessed: 1st January 2016)

52. Moineau Pump. Available at: http://hackaday.com/2013/10/16/3d-printering-pastestruders/. (Accessed: 1st January 2016)
53. Salo, T. Moineau Extruder STL Files. Available at: http://www.thingiverse.com/thing:15538. (Accessed: 1st January 2016)
54. Nakamura, M., Iwanaga, S., Henmi, C., Arai, K. & Nishiyama, Y. Biomatrices and biomaterials for future developments of bioprinting and biofabrication. *Biofabrication* **2,** 014110 (2010).
55. Skardal, A. *et al.* A hydrogel bioink toolkit for mimicking native tissue biochemical and mechanical properties in bioprinted tissue constructs. *Acta Biomater.* **25,** 24–34 (2015).

Appendices

A1. Gantt Chart

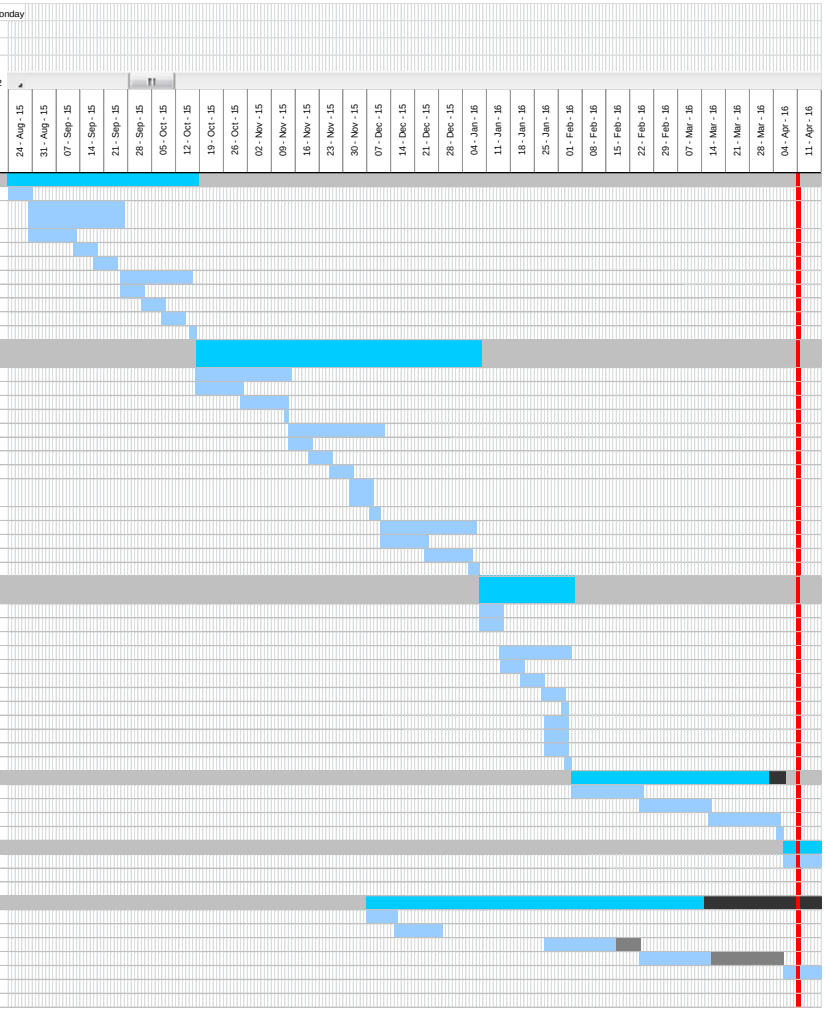
Dept. of BME; UM — Today's Date: 4/11/2016 Monday (vertical red line)

Project Lead: Brad

Start Date: 8/15/2015 Saturday

First Day of Week (Mon=2): 2

WBS	Tasks	Task Lead	Start	End	Duration (Days)	% Complete	Working Days	Days Complete	Days Remaining
1	**Concept Generation**	Brad	8/17/15	10/18/15	63	100%	45	63	0
1.1	Needs Screening		8/17/15	8/30/15	14	100%	10	14	0
1.2	Technology Investigation		8/30/15	9/26/15	28	100%	20	28	0
1.2.1	3D Printer		8/30/15	9/12/15	14	100%	10	14	0
1.2.2	Extrusion System		9/12/15	9/18/15	7	100%	5	7	0
1.2.3	Thermoregulation		9/18/15	9/24/15	7	100%	5	7	0
1.3	Concept Generation		9/26/15	10/16/15	21	100%	15	21	0
1.3.1	3D Printer		9/26/15	10/2/15	7	100%	5	7	0
1.3.2	Extrusion System		10/2/15	10/8/15	7	100%	5	7	0
1.3.3	Thermoregulation		10/8/15	10/14/15	7	100%	5	7	0
1.4	Buffer		10/16/15	10/17/15	2	100%	1	2	0
2	**Sourcing and Construction**	Brad	10/18/15	1/9/16	84	100%	60	84	0
2.1	3D Printer		10/18/15	11/14/15	28	100%	20	28	0
2.1.1	Sourcing		10/18/15	10/31/15	14	100%	10	14	0
2.1.2	Construction		10/31/15	11/13/15	14	100%	10	14	0
2.2	Buffer		11/13/15	11/13/15	1	100%	1	1	0
2.3	Extrusion System		11/14/15	12/11/15	28	100%	20	28	0
2.3.1	Primary Sourcing		11/14/15	11/20/15	7	100%	5	7	0
2.3.2	Primary Construction		11/20/15	11/26/15	7	100%	5	7	0
2.3.3	Secondary Sourcing		11/26/15	12/2/15	7	100%	5	7	0
2.3.4	Secondary Construction		12/2/15	12/8/15	7	100%	5	7	0
2.4	Buffer		12/8/15	12/10/15	3	100%	3	3	0
2.5	Thermoregulation		12/11/15	1/7/16	28	100%	20	28	0
2.5.1	Primary Sourcing		12/11/15	12/24/15	14	100%	10	14	0
2.5.2	Primary Construction		12/24/15	1/6/16	14	100%	10	14	0
2.6	Buffer		1/6/16	1/8/16	3	100%	3	3	0
3	**PoC/Verification Testing**	Brad	1/9/16	2/5/16	28	100%	20	28	0
3.1	3D Printer		1/9/16	1/15/16	7	100%	5	7	0
3.1.1	Printer resolution test		1/9/16	1/15/16	7	100%	5	7	0
3.2	Buffer		1/15/16	1/14/16	0	100%	-2	0	0
3.3	Syringe Pump		1/15/16	2/4/16	21	100%	15	21	0
3.3.1	Control test		1/15/16	1/21/16	7	100%	5	7	0
3.3.2	Features ID		1/21/16	1/27/16	7	100%	5	7	0
3.3.3	Soft/Firmware Opt.		1/27/16	2/2/16	7	100%	5	7	0
3.4	Buffer		2/2/16	2/3/16	2	100%	2	2	0
3.5	Thermoregulation		1/28/16	2/3/16	7	100%	5	7	0
3.5.1	Syringe heater test		1/28/16	2/3/16	7	100%	5	7	0
3.5.2	Bed chiller test		1/28/16	2/3/16	7	100%	5	7	0
3.6	Buffer		2/3/16	2/4/16	2	100%	2	2	0
4	**Design Improvement**	Brad	2/5/16	4/7/16	63	93%	45	58	5
4.1	3D Printer		2/5/16	2/25/16	21	100%	15	21	0
4.2	Syringe Pump		2/25/16	3/16/16	21	100%	15	21	0
4.3	Thermoregulation		3/16/16	4/5/16	21	100%	15	21	0
4.4	Buffer		4/5/16	4/6/16	2	100%	2	2	0
5	**Design Validation**	Brad	4/7/16	5/4/16	28	76%	20	21	7
5.1	Gelatin-only system		4/7/16	4/20/16	14	100%	10	14	0
5.2	Cells + gelatin		4/20/16	5/3/16	14	50%	10	7	7
5.3	Buffer		5/3/16	5/3/16	1	100%	1	1	0
6	**Writing**	Brad	12/7/15	5/11/16	156	64%	113	99	57
6.1	Midpoint Report		12/7/15	12/15/15	9	100%	7	9	0
6.2	Background		12/15/15	12/28/15	14	100%	10	14	0
6.3	Proof of Concept		1/28/16	2/24/16	28	75%	20	21	7
6.4	Update Development		2/25/16	4/6/16	42	50%	30	21	21
6.5	Design Validation		4/7/16	5/3/16	27	50%	19	13	14
6.6	Results & Discussion		5/3/16	5/10/16	8	50%	6	4	4
6.7	Abstract & Conclusion		5/3/16	5/10/16	8	50%	6	4	4



A2. BOM and Budget

System	Component	Qty	Unit Cost	Lot Cost	Source
Extrusion Printer	3D Printed Parts	26	$1.1538	$30.00	https://github.com/josefprusa/Prusa3/tree/master/old_single_plate/src
	Milled y-carriage	1	$60.0000	$60.00	https://github.com/josefprusa/Prusa3
	M8 smooth rod	2100	$0.0172	$36.10	http://www.mcmaster.com/#metal-rods/=108hufg
	M10 threaded rod	760	$0.0256	$19.46	http://www.mcmaster.com/#standard-threaded-rods/=108hv29
	M8 threaded rod	820	$0.0147	$12.07	http://www.mcmaster.com/#standard-threaded-rods/=108hvaq
	M5 threaded rod	600	$0.0107	$6.43	http://www.mcmaster.com/#standard-threaded-rods/=108hvgs
	GT2 belt, 6mm wide	1660	$0.0023	$3.88	http://www.amazon.com/FbscTech-Meters-timing-Rostock-GT2-6mm/dp/B00KWA06ZM/ref=sr_1_1?ie=UTF8&qid=1450105382&sr=8-1&keywords=GT2+timing+belt+2+meters
	GT2 pulley, 6mm wide	2	$2.7980	$5.60	http://www.amazon.com/E-accexpert-Synchronous-Pulley-wrench-Printer/dp/B00YQUX5H0/ref=sr_1_2?ie=UTF8&qid=1450105456&sr=8-2&keywords=GT2+pulley
	Hobbed bolt, M8x60mm	1	$6.5000	$6.50	http://www.ebay.com/itm/Hobbed-bolt-M8-for-Wade-reloaded-extruder-Prusa-Mendel-Reprap-/181075142521?hash=item2a28eb6779:m:msxgoyNDx4zFoS9JRi3_ENQ
	608zz bearings	3	$12.3300	$36.99	http://www.mcmaster.com/#608-ball-bearings/=108hzd4
	M4 springs	4	$2.8633	$11.45	http://www.mcmaster.com/#compression-springs/=108hz56
	J-Head V6 Hot End Assembly	1	$20.0000	$20.00	http://www.amazon.com/3D-CAM-Printer-Filament-Extruder/dp/B010MSTVZO/ref=pd_sbs_328_1?ie=UTF8&dpID=41jpf2ZI2EL&dpSrc=sims&preST=_AC_UL160_SR160%2C160_&refRID=118H8HK31HCV1DBJVY2Y
	Heated bed	1	$5.9800	$5.98	http://www.ebay.com/itm/1-RepRap-3D-Printer-PCB-Heatbed-MK2a-Heat-Bed-for-Prusa-Mendel-/171275126930?hash=item27e0cb0892:g:r4cAAOSwzrxUt4Jz
	RAMBo 1.3: 5x 1/16th microstep stepper drivers, 5 PWM mosfet outputs, 4 thermistor inputs, digital trimpot, 3 independent fuse-protected power rails, Atmega2560 and Atmega 32u2 processors, 2 channel SDRAMPS-compatible SPI breakout	1	$170.0000	$170.00	https://ultimachine.com/content/rambo-13
	Terminal blocks, mechanical endstops, and wiring	3	$0.0000	$0.00	Included with above
	LM8UU linear bearings	10	$1.3500	$13.50	http://www.amazon.com/6pcs-LM8UU-Linear-Bearing-Bushing/dp/B008RIKN7W
	5x5mm steel couplers	2	$4.2200	$8.44	http://www.ebay.com/itm/CNC-Motor-Shaft-Coupler-5mm-to-5mm-Flexible-Coupling-5x5mm-/321687248791?hash=item4ae60ddf97:g:rkkAAOSwHnFVlU2b
	M3x10mm bolt	18	$0.1998	$3.60	http://www.mcmaster.com/#socket-head-cap-screws/=108i7t7
	M3x18mm bolt	3	$0.2080	$0.62	http://www.mcmaster.com/#socket-head-cap-screws/=108i8so
	M3x30mm bolt	14	$0.2720	$3.81	http://www.mcmaster.com/#socket-head-cap-screws/=108i8y0
	M3 nuts	28	$0.0616	$1.72	http://www.mcmaster.com/#hex-nuts/=108iabh
	M3 washers	54	$0.0268	$1.45	http://www.mcmaster.com/#standard-washers/=108i9kt
	M4x20mm bolt	5	$0.2800	$1.40	http://www.mcmaster.com/#socket-head-cap-screws/=108i95o

M4x25mm bolt	2	$0.3000	0.60	http://www.mcmaster.com/#socket-head-cap-screws/=108i9ac
M4 nuts	7	$0.0684	0.48	http://www.mcmaster.com/#hex-nuts/=108iai9
M4 washers	10	$0.0337	0.34	http://www.mcmaster.com/#standard-washers/=108i9rc
M5 nuts	2	0.0824	0.16	http://www.mcmaster.com/#hex-nuts/=108iao5
M8 nuts	22	0.2676	5.89	http://www.mcmaster.com/#hex-nuts/=108iasf
M8 washers	22	0.109	2.40	http://www.mcmaster.com/#standard-washers/=108i9x5
M10 nuts	12	0.3924	4.71	http://www.mcmaster.com/#hex-nuts/=108ib2k
M10 washers	12	0.2234	2.68	http://www.mcmaster.com/#standard-washers/=108ia27
Zip Ties	40	0.0402	1.61	http://www.mcmaster.com/#zip-ties/=108ibok
Nema 17 bipolar stepper motors	5	13.99	69.95	http://www.ebay.com/itm/Lot-5-NEMA-17-Bipolar-Hybrid-Stepper-Motor-3D-Printer-RepRap-Mendel-Prusa-i3-/321373773968
System Total			**$547.82**	

Syringe Pump

3D Printed Parts	8	N/A	$40.00	https://www.youmagine.com/designs/syringe-pump
Nema 17 bipolar stepper motor	1	$14.9900	$14.99	http://www.amazon.com/Stepper-Motor-Bipolar-4-lead-Printer/dp/B00PNEQKC0/ref=sr_1_1?ie=UTF8&qid=1450107982&sr=8-1&keywords=Nema17+motor
5mmx5mm shaft coupling	1	$4.2200	$4.22	http://www.ebay.com/itm/CNC-Motor-Shaft-Coupler-5mm-to-5mm-Flexible-Coupling-5x5mm-/321687248791?hash=item4ae60ddf97:g:rkkAAOSwHnFVlU2b
625z ball bearing	2	$12.6000	$25.20	http://www.mcmaster.com/#standard-ball-and-roller-bearings/=108iha8
LM6UU linear bearing	2	$3.6150	$7.23	http://www.amazon.com/19mm-Rubber-Linear-Motion-Bearings/dp/B008LTYFK8/ref=sr_1_4?ie=UTF8&qid=1450108193&sr=8-4&keywords=LM6UU+linear+bearing
M3x10mm socket head cap screw	6	$0.0440	$0.26	http://www.mcmaster.com/#socket-head-cap-screws/=108iijy
M3x20mm socket head cap screw	4	$0.0600	$0.24	http://www.mcmaster.com/#socket-head-cap-screws/=108iir1
M3x40mm socket head cap screw	4	$0.1160	$0.46	http://www.mcmaster.com/#socket-head-cap-screws/=108iivo
M3 hex nut	13	$0.0555	$0.72	http://www.mcmaster.com/#hex-nuts/=108ij5b
M5 hex nut	5	$0.0555	$0.28	http://www.mcmaster.com/#hex-nuts/=108ij9r
M5 threaded rod	200	$0.0145	$2.90	http://www.mcmaster.com/#standard-threaded-rods/=108ijjn
M6 smooth rod	400	$0.0142	$5.70	http://www.mcmaster.com/#metal-rods/=108ik7w
System Total			**$102.20**	

Thermo- regulation

Thermo-Kinetic Heater Control Unit & Primary Heating Pad	1	$399.0000	$399.00	http://www.syringepump.com/heater.php
Secondary Heating Pad	1	$100.0000	$100.00	http://www.syringepump.com/heater.php
10K precision epoxy thermistor - 3950 NTC	1	$4.0000	$4.00	https://www.adafruit.com/products/372
Panel temperature meter	1	$9.9500	$9.95	https://www.adafruit.com/products/576
Peltier thermoelectric cooler/heatsink assy.	1	$34.9500	$34.95	https://www.adafruit.com/products/1335
Arduino Uno R3	1	$24.9500	$24.95	https://www.adafruit.com/products/50
Powerswitch tail 2	1	$25.9500	$25.95	https://www.adafruit.com/products/268
12V 5A switching power supply	1	$24.9500	$24.95	https://www.adafruit.com/products/352
Aluminum print bed	1	$16.9800	$16.98	http://www.mcmaster.com/#standard-aluminum-sheets/=108oyxw
System Total			**$640.73**	
Design Total			**$1,290.75**	

A3. 3-Axis Stage Resolution Testing Sample Data

3-Axis Stage Qualification Testing					
Dimension	**Measurement**	**Actual Length (mm)**	**Desired Length (mm)**	**|Length Error| (mm)**	**% Error**
X	1	9.97	10	0.03	0.30%
X	2	10.09	10	0.09	0.90%
X	3	10.05	10	0.05	0.50%
X	4	10.05	10	0.05	0.50%
X	5	9.94	10	0.06	0.60%
X	6	9.98	10	0.02	0.20%
X	7	9.92	10	0.08	0.80%
X	8	10.04	10	0.04	0.40%
X	9	10.08	10	0.08	0.80%
X	10	10.05	10	0.05	0.50%
Avg Error (mm)	**0.06**	**% Error**	**0.55%**	**Std Dev (mm)**	**0.06**
Y	1	10.08	10	0.08	0.80%
Y	2	10.10	10	0.10	1.00%
Y	3	9.98	10	0.02	0.20%
Y	4	10.01	10	0.01	0.10%
Y	5	10.01	10	0.01	0.10%
Y	6	10.03	10	0.03	0.30%
Y	7	10.04	10	0.04	0.40%
Y	8	10.02	10	0.02	0.20%
Y	9	9.96	10	0.04	0.40%
Y	10	10.03	10	0.03	0.30%
Avg Error (mm)	**0.04**	**% Error**	**0.38%**	**Std Dev (mm)**	**0.04**
Z	1	8.68	10	1.32	13.20%
Z	2	9.99	10	0.01	0.10%
Z	3	9.91	10	0.09	0.90%
Z	4	10.05	10	0.05	0.50%
Z	5	10.09	10	0.09	0.90%
Z	6	10.45	10	0.45	4.50%
Z	7	9.74	10	0.26	2.60%
Z	8	10.02	10	0.02	0.20%
Z	9	9.97	10	0.03	0.30%
Z	10	10.25	10	0.25	2.50%
Z	11	10.66	10	0.66	6.60%
Z	12	9.58	10	0.42	4.20%
Z	13	10.04	10	0.04	0.40%
Z	14	9.72	10	0.28	2.80%
Z	15	10.34	10	0.34	3.40%
Z	16	9.83	10	0.17	1.70%
Z	17	10.15	10	0.15	1.50%
Z	18	10.61	10	0.61	6.10%
Z	19	10.16	10	0.16	1.60%
Z	20	9.29	10	0.71	7.10%
Avg Error (mm)	**0.31**	**% Error**	**3.06%**	**Std Dev (mm)**	**0.44**

A4. "PRUSA" Logo Print Resolution Testing Sample Data

"PRUSA" Logo Print Resolution					
Dimension	**Measurement**	**Actual Length (mm)**	**Desired Length (mm)**	**|Length Error| (mm)**	**% Error**
X	5	106.47	105	1.47	1.40%
X	6	106.37	105	1.37	1.31%
X	7	106.89	105	1.89	1.80%
X	8	5.97	5.5	0.47	8.58%
X	9	5.36	5.5	0.15	2.64%
X	10	18.74	19	0.26	1.36%
X	11	17.51	17.5	0.00	0.03%
X	12	17.71	17.5	0.21	1.21%
X	13	6.38	6.25	0.13	2.14%
X	14	6.49	6.25	0.24	3.79%
X	15	6.18	6.25	0.07	1.15%
Avg Error (mm)	**0.57**	**% Error**	**2.31%**	**Std Dev (mm)**	**0.06**
Y	1	19.05	20	0.95	4.75%
Y	2	18.95	20	1.05	5.27%
Y	3	19.67	20	0.33	1.66%
Y	4	19.67	20	0.33	1.66%
Y	16	19.89	20	0.11	0.55%
Y	17	5.86	5.5	0.36	6.55%
Y	18	5.29	5.5	0.21	3.82%
Y	19	5.54	5.5	0.04	0.73%
Y	20	5.41	5.5	0.09	1.64%
Y	21	5.6	5.5	0.10	1.82%
Avg Error (mm)	**0.36**	**% Error**	**2.84%**	**Std Dev (mm)**	**0.04**
Z	22	1.05	1	0.05	5.00%
Z	23	0.97	1	0.03	3.00%
Z	24	0.86	1	0.14	14.00%
Z	25	1.13	1	0.13	13.00%
Z	26	0.92	1	0.08	8.00%
Z	27	1.95	2	0.05	2.50%
Z	28	1.87	2	0.13	6.50%
Z	29	1.89	2	0.11	5.50%
Z	30	1.96	2	0.04	2.00%
Z	31	1.99	2	0.01	0.50%
Avg Error (mm)	**0.08**	**% Error**	**6.00%**	**Std Dev (mm)**	**0.45**

A5. Syringe Pump Capability Testing Sample Data

Calculation Parameters	
Density (g/mL)	0.9977735
mm^3 / mL	1000

Syringe Pump Component Qualification: Step 1					
Extrude 500 steps, measure mass output, calculate volume, determine steps/mm^3					
	Density (g/mL):		0.9977735		
Sample	**Tare Mass (g)**	**Final Mass (g)**	**Extruded Mass (g)**	**Volume (mm^3)**	**Steps/mm^3**
1	14.62	15.02	0.4	400.89	1.25
2	14.56	14.93	0.37	370.83	1.35
3	14.36	14.76	0.4	400.89	1.25
4	14.34	14.7	0.36	360.80	1.39
5	14.4	14.78	0.38	380.85	1.31
6	14.4	14.78	0.38	380.85	1.31
7	15.03	15.42	0.39	390.87	1.28
8	14.92	15.32	0.4	400.89	1.25
9	14.77	15.16	0.39	390.87	1.28
10	14.71	15.1	0.39	390.87	1.28
11	14.79	15.2	0.41	410.91	1.22
12	14.78	15.19	0.41	410.91	1.22
13	15.4	15.82	0.42	420.94	1.19
14	16.06	16.46	0.4	400.89	1.25
15	15.14	15.52	0.38	380.85	1.31
16	15.1	15.5	0.4	400.89	1.25
17	15.2	15.59	0.39	390.87	1.28
18	15.18	15.58	0.4	400.89	1.25
19	15.82	16.2	0.38	380.85	1.31
20	15.52	15.91	0.39	390.87	1.28
21	15.5	15.91	0.41	410.91	1.22
22	15.59	15.99	0.4	400.89	1.25
23	15.58	15.97	0.39	390.87	1.28
24	16.21	16.6	0.39	390.87	1.28
25	16.46	16.85	0.39	390.87	1.28
26	15.92	16.35	0.43	430.96	1.16
Average Steps/mm^3:					1.27

Syringe Pump Component Calibration: Step 2						
Extrude 1267 steps (~1000 mm^3), measure mass output, calculate volume						
	Density (g/mL):		0.9977735			
Sample	**Tare Mass (g)**	**Final Mass (g)**	**Mass Change (g)**	**Volume (mm^3)**	**|Volume Error| (mm^3)**	**% Error**
1	16.6	17.59	0.99	992.21	7.79	0.78%
2	16.83	17.87	1.04	1042.32	42.32	4.23%
3	16.35	17.34	0.99	992.21	7.79	0.78%
4	16.09	17.13	1.04	1042.32	42.32	4.23%
5	15.99	17	1.01	1012.25	12.25	1.23%
6	15.95	16.96	1.01	1012.25	12.25	1.23%
7	17.59	18.6	1.01	1012.25	12.25	1.23%
8	17.86	18.85	0.99	992.21	7.79	0.78%
9	17.35	18.38	1.03	1032.30	32.30	3.23%
10	17.21	18.26	1.05	1052.34	52.34	5.23%
Avg Error (mm^3)	**22.94**	**% Error**	**2.29%**	**Std Dev (mm^3)**	**21.59**	

A6. Stage-Nozzle Assembly Resolution Testing Sample Data

Stage + Nozzle Verification					
Dimension	**Measurement**	**Actual Length (mm)**	**Desired Length (mm)**	**|Length Error| (mm)**	**% Error**
X	1	9.96	10	0.04	0.40%
X	2	9.94	10	0.06	0.60%
X	3	10.06	10	0.06	0.60%
X	4	9.97	10	0.03	0.30%
X	5	10.01	10	0.01	0.10%
X	6	10.03	10	0.03	0.30%
X	7	10.02	10	0.02	0.20%
X	8	10.04	10	0.04	0.40%
X	9	10.03	10	0.03	0.30%
X	10	9.91	10	0.09	0.90%
X	11	10.00	10	0.00	0.00%
X	12	9.96	10	0.04	0.40%
X	13	9.97	10	0.03	0.30%
Avg Error (mm)	**0.04**	**% Error**	**0.37%**	**Std Dev (mm)**	**0.04**
Y	1	10.04	10	0.04	0.40%
Y	2	10.03	10	0.03	0.30%
Y	3	9.93	10	0.07	0.70%
Y	4	10.01	10	0.01	0.10%
Y	5	10.03	10	0.03	0.30%
Y	6	9.97	10	0.03	0.30%
Y	7	10.01	10	0.01	0.10%
Y	8	10.08	10	0.08	0.80%
Y	9	10.02	10	0.02	0.20%
Y	10	9.96	10	0.04	0.40%
Y	11	10.05	10	0.05	0.50%
Y	12	10.06	10	0.06	0.60%
Y	13	10.00	10	0.00	0.00%
Avg Error (mm)	**0.04**	**% Error**	**0.36%**	**Std Dev (mm)**	**0.04**
Z	1	10.14	10	0.14	1.40%
Z	2	10.31	10	0.31	3.10%
Z	3	10.35	10	0.35	3.50%
Z	4	10.19	10	0.19	1.90%
Z	5	8.89	10	1.11	11.10%
Z	6	10.26	10	0.26	2.60%
Z	7	9.57	10	0.43	4.30%
Z	8	10.62	10	0.62	6.20%
Z	9	9.98	10	0.02	0.20%
Z	10	10.67	10	0.67	6.70%
Z	11	9.91	10	0.09	0.90%
Z	12	9.88	10	0.12	1.20%
Z	13	9.62	10	0.38	3.80%
Z	14	10.45	10	0.45	4.50%
Z	15	9.87	10	0.13	1.30%
Z	16	10.12	10	0.12	1.20%
Z	17	10.35	10	0.35	3.50%
Z	18	10.09	10	0.09	0.90%
Z	19	9.92	10	0.08	0.80%
Z	20	9.98	10	0.02	0.20%
Avg Error (mm)	**0.30**	**% Error**	**2.97%**	**Std Dev (mm)**	**0.39**

A7. Syringe Pump Calibration Testing Sample Data

Calculation Parameters	
Density (g/mL)	0.9977735
mm^3 / mL	1000
Diameter (mm)	1

System Calibration: Step 1				
Extrude X steps, measure mass output, convert to volume, calculate steps/mm				
	Using	236	steps/mm	
# Steps	**Extruded Mass (g)**	**Volume (mm^3)**	**Extruded Length (mm)**	**Steps/mm**
4720	0.4	400.89	510.43	9
4720	0.42	420.94	535.95	9
4720	0.39	390.87	497.67	9
4720	0.43	430.96	548.72	9
4720	0.38	380.85	484.91	10
4720	0.44	440.98	561.48	8
4720	0.43	430.96	548.72	9
4720	0.37	370.83	472.15	10
4720	0.44	440.98	561.48	8
4720	0.38	380.85	484.91	10
11800	1.05	1052.34	1339.89	9
11800	1.04	1042.32	1327.13	9
11800	1.02	1022.28	1301.60	9
11800	1.06	1062.37	1352.65	9
11800	1.03	1032.30	1314.36	9
11800	1.02	1022.28	1301.60	9
11800	1.05	1052.34	1339.89	9
11800	1.02	1022.28	1301.60	9
11800	1.03	1032.30	1314.36	9
11800	1.08	1082.41	1378.17	9
Average Steps/mm:				9

System Calibration: Step 2					
Extrude 9000 steps (1000 mm), measure mass output, convert to volume, calculate error					
	Using	9	steps/mm		
Iteration	**Extruded Mass (g)**	**Extruded Volume (mm^3)**	**Target Volume (mm^3)**	**|Volume Error| (mm^3)**	**% Error**
1	0.71	711.58	785.40	74	7.38%
2	0.78	781.74	785.40	4	0.37%
3	0.77	771.72	785.40	14	1.37%
4	0.82	821.83	785.40	36	3.64%
5	0.78	781.74	785.40	4	0.37%
6	0.69	691.54	785.40	94	9.39%
7	0.81	811.81	785.40	26	2.64%
8	0.83	831.85	785.40	46	4.65%
9	0.82	821.83	785.40	36	3.64%
10	0.81	811.81	785.40	26	2.64%
11	0.79	791.76	785.40	6	0.64%
12	0.76	761.70	785.40	24	2.37%
13	0.82	821.83	785.40	36	3.64%
14	0.84	841.87	785.40	56	5.65%
Avg Error (mm^3)	**31.54**	**% Error**	**3.15%**	**Std Dev (mm^3)**	**38.22**

A8. Verification Resolution Testing Sample Data

Verification Resolution Testing					
Dimension	**Measurement**	**Actual Length (mm)**	**Desired Length (mm)**	**|Length Error| (mm)**	**% Error**
X	2	14.91	14.00	0.91	6.49%
X	4	22.12	22.00	0.12	0.54%
X	6	30.20	30.00	0.20	0.67%
X	8	38.00	38.00	0.00	0.01%
X	10	46.60	46.00	0.60	1.31%
X	12	55.01	54.00	1.01	1.87%
X	14	14.45	14.00	0.45	3.21%
X	16	21.96	22.00	-0.04	0.18%
X	18	29.41	30.00	-0.59	1.97%
X	20	38.65	38.00	0.65	1.71%
X	22	45.49	46.00	-0.51	1.11%
X	24	52.90	54.00	-1.10	2.04%
Avg Error (mm)	**0.14**	**% Error**	**1.76%**	**Std Dev (mm)**	**0.61**
Y	1	10.61	10.00	0.61	6.12%
Y	3	18.82	18.00	0.82	4.54%
Y	5	26.50	26.00	0.50	1.93%
Y	7	34.32	34.00	0.32	0.94%
Y	9	43.00	42.00	1.00	2.38%
Y	11	51.42	50.00	1.42	2.83%
Y	13	9.87	10.00	-0.13	1.30%
Y	15	17.79	18.00	-0.21	1.17%
Y	17	26.24	26.00	0.24	0.92%
Y	19	33.68	34.00	-0.32	0.94%
Y	21	41.44	42.00	-0.56	1.33%
Y	23	50.49	50.00	0.49	0.98%
Avg Error (mm)	**0.35**	**% Error**	**2.12%**	**Std Dev (mm)**	**0.56**
Z	25	0.39	0.4	-0.01	2.50%
Z	26	0.36	0.4	-0.04	10.00%
Z	27	0.42	0.4	0.02	5.00%
Z	28	0.46	0.4	0.06	15.00%
Z	29	0.33	0.4	-0.07	17.50%
Z	30	0.36	0.4	-0.04	10.00%
Z	31	0.49	0.4	0.09	22.50%
Z	32	0.33	0.4	-0.07	17.50%
Z	33	0.45	0.4	0.05	12.50%
Z	34	0.49	0.4	0.09	22.50%
Avg Error (mm)	**0.01**	**% Error**	**13.50%**	**Std Dev (mm)**	**0.06**

A9. Validation Resolution Testing Sample Data

Validation Resolution Testing					
Dimension	**Measurement**	**Actual Length (mm)**	**Desired Length (mm)**	**|Length Error| (mm)**	**% Error**
X	1	2.98	3.00	-0.02	0.70%
X	2	3.22	3.00	0.22	7.17%
X	3	3.14	3.00	0.14	4.57%
X	4	3.06	3.00	0.06	1.90%
X	5	3.21	3.00	0.21	7.13%
X	6	2.51	3.00	-0.49	16.37%
X	7	2.59	3.00	-0.41	13.73%
X	8	2.74	3.00	-0.26	8.53%
X	9	2.82	3.00	-0.18	5.93%
X	10	3.14	3.00	0.14	4.53%
X	11	21.09	20.00	1.09	5.46%
X	12	21.88	20.00	1.88	9.38%
X	13	20.00	20.00	0.00	0.01%
X	14	19.68	20.00	-0.32	1.59%
X	17	22.27	21.00	1.27	6.02%
X	18	22.11	21.00	1.11	5.29%
X	19	22.19	21.00	1.19	5.65%
X	20	22.42	21.00	1.42	6.77%
X	21	22.11	21.00	1.11	5.28%
X	22	21.17	21.00	0.17	0.82%
X	23	21.57	21.00	0.57	2.70%
X	24	21.17	21.00	0.17	0.83%
X	25	21.33	21.00	0.33	1.55%
X	26	20.93	21.00	-0.07	0.31%
X	31	8.94	9.00	-0.06	0.70%
X	32	8.86	9.00	-0.14	1.57%
X	33	8.78	9.00	-0.22	2.41%
X	34	8.94	9.00	-0.06	0.67%
Avg Error (mm)	**0.32**	**% Error**	**4.56%**	**Std Dev (mm)**	**0.62**
Y	15	11.53	11.50	0.03	0.23%
Y	16	11.76	11.50	0.26	2.26%
Y	27	49.78	50.00	-0.22	0.44%
Y	28	50.25	50.00	0.25	0.50%
Y	29	49.62	50.00	-0.38	0.75%
Y	30	49.23	50.00	-0.77	1.54%
Y	35	11.54	11.50	0.04	0.35%
Y	36	11.13	11.50	-0.37	3.22%
Y	37	49.87	50.00	-0.13	0.26%
Y	38	50.21	50.00	0.21	0.42%
Y	39	50.43	50.00	0.43	0.86%
Y	40	49.89	50.00	-0.11	0.22%
Avg Error (mm)	**-0.06**	**% Error**	**0.92%**	**Std Dev (mm)**	**0.32**
Z	41	0.96	1	-0.04	4.00%
Z	42	1.12	1	0.12	12.00%
Z	43	1.13	1	0.13	13.00%
Z	44	0.87	1	-0.13	13.00%
Z	45	0.79	1	-0.21	21.00%
Z	46	1.12	1	0.12	12.00%
Z	47	0.91	1	-0.09	9.00%
Z	48	0.95	1	-0.05	5.00%
Z	49	1.08	1	0.08	8.00%
Z	50	0.97	1	-0.03	3.00%
Avg Error (mm)	**-0.01**	**% Error**	**10.00%**	**Std Dev (mm)**	**0.11**

A10. Marlin Firmware Code

```
#ifndef CONFIGURATION_H
#define CONFIGURATION_H

#include "boards.h"

// This configuration file contains the basic settings.
// Advanced settings can be found in Configuration_adv.h
// BASIC SETTINGS: select your board type, temperature sensor type, axis scaling, and endstop configuration

//===========================================================================
//============================= DELTA Printer ===============================
//===========================================================================
// For a Delta printer replace the configuration files with the files in the
// example_configurations/delta directory.
//

//===========================================================================
//============================= SCARA Printer ===============================
//===========================================================================
// For a Delta printer replace the configuration files with the files in the
// example_configurations/SCARA directory.
//

// User-specified version info of this build to display in [Pronterface, etc] terminal window during
// startup. Implementation of an idea by Prof Braino to inform user that any changes made to this
// build by the user have been successfully uploaded into firmware.

//#define STRING_VERSION "1.0.2"

#define STRING_VERSION_CONFIG_H __DATE__ "4/15/2016" __TIME__ // build date and time
#define STRING_CONFIG_H_AUTHOR "Brad Bradshaw" // Who made the changes.

// SERIAL_PORT selects which serial port should be used for communication with the host.
// This allows the connection of wireless adapters (for instance) to non-default port pins.
// Serial port 0 is still used by the Arduino bootloader regardless of this setting.
#define SERIAL_PORT 0

// This determines the communication speed of the printer
#define BAUDRATE 115200

// This enables the serial port associated to the Bluetooth interface
//#define BTENABLED              // Enable BT interface on AT90USB devices

// The following define selects which electronics board you have.
// Please choose the name from boards.h that matches your setup
#ifndef MOTHERBOARD
  #define MOTHERBOARD BOARD_RAMBO_MINI
#endif

// Define this to set a custom name for your generic Mendel,
#define CUSTOM_MENDEL_NAME "Prusa i3 Bioprinter"

// Define this to set a unique identifier for this printer, (Used by some programs to differentiate between machines)
// You can use an online service to generate a random UUID. (eg http://www.uuidgenerator.net/version4)
// #define MACHINE_UUID "00000000-0000-0000-0000-000000000000"

// This defines the number of extruders
#define EXTRUDERS 1

//// The following define selects which power supply you have. Please choose the one that matches your setup
// 1 = ATX
// 2 = X-Box 360 203Watts (the blue wire connected to PS_ON and the red wire to VCC)

#define POWER_SUPPLY 1

// Define this to have the electronics keep the power supply off on startup. If you don't know what this is leave it.
// #define PS_DEFAULT_OFF

//===========================================================================
```

```
//============================Thermal Settings  ============================
//==========================================================================
//
//--NORMAL IS 4.7kohm PULLUP!-- 1kohm pullup can be used on hotend sensor, using correct resistor and table
//
//// Temperature sensor settings:
// -2 is thermocouple with MAX6675 (only for sensor 0)
// -1 is thermocouple with AD595
// 0 is not used
// 1 is 100k thermistor - best choice for EPCOS 100k (4.7k pullup)
// 2 is 200k thermistor - ATC Semitec 204GT-2 (4.7k pullup)
// 3 is Mendel-parts thermistor (4.7k pullup)
// 4 is 10k thermistor !! do not use it for a hotend. It gives bad resolution at high temp. !!
// 5 is 100K thermistor - ATC Semitec 104GT-2 (Used in ParCan & J-Head) (4.7k pullup)
// 6 is 100k EPCOS - Not as accurate as table 1 (created using a fluke thermocouple) (4.7k pullup)
// 7 is 100k Honeywell thermistor 135-104LAG-J01 (4.7k pullup)
// 71 is 100k Honeywell thermistor 135-104LAF-J01 (4.7k pullup)
// 8 is 100k 0603 SMD Vishay NTCS0603E3104FXT (4.7k pullup)
// 9 is 100k GE Sensing AL03006-58.2K-97-G1 (4.7k pullup)
// 10 is 100k RS thermistor 198-961 (4.7k pullup)
// 11 is 100k beta 3950 1% thermistor (4.7k pullup)
// 12 is 100k 0603 SMD Vishay NTCS0603E3104FXT (4.7k pullup) (calibrated for Makibox hot bed)
// 13 is 100k Hisens 3950  1% up to 300°C for hotend "Simple ONE " & "Hotend "All In ONE"
// 20 is the PT100 circuit found in the Ultimainboard V2.x
// 60 is 100k Maker's Tool Works Kapton Bed Thermistor beta=3950
//
//    1k ohm pullup tables - This is not normal, you would have to have changed out your 4.7k for 1k
//                          (but gives greater accuracy and more stable PID)
// 51 is 100k thermistor - EPCOS (1k pullup)
// 52 is 200k thermistor - ATC Semitec 204GT-2 (1k pullup)
// 55 is 100k thermistor - ATC Semitec 104GT-2 (Used in ParCan & J-Head) (1k pullup)
//
// 1047 is Pt1000 with 4k7 pullup
// 1010 is Pt1000 with 1k pullup (non standard)
// 147 is Pt100 with 4k7 pullup
// 110 is Pt100 with 1k pullup (non standard)

#define TEMP_SENSOR_0 5
#define TEMP_SENSOR_1 0
#define TEMP_SENSOR_2 0
#define TEMP_SENSOR_BED 1

// This makes temp sensor 1 a redundant sensor for sensor 0. If the temperatures difference between these sensors is to high
the print will be aborted.
//#define TEMP_SENSOR_1_AS_REDUNDANT
#define MAX_REDUNDANT_TEMP_SENSOR_DIFF 10

// Actual temperature must be close to target for this long before M109 returns success
#define TEMP_RESIDENCY_TIME 3  // (seconds)
#define TEMP_HYSTERESIS 5       // (degC) range of +/- temperatures considered "close" to the target one
#define TEMP_WINDOW     1       // (degC) Window around target to start the residency timer x degC early.

// The minimal temperature defines the temperature below which the heater will not be enabled It is used
// to check that the wiring to the thermistor is not broken.
// Otherwise this would lead to the heater being powered on all the time.
#define HEATER_0_MINTEMP 5
#define HEATER_1_MINTEMP 5
#define HEATER_2_MINTEMP 5
#define BED_MINTEMP 5

// When temperature exceeds max temp, your heater will be switched off.
// This feature exists to protect your hotend from overheating accidentally, but *NOT* from thermistor short/failure!
// You should use MINTEMP for thermistor short/failure protection.
#define HEATER_0_MAXTEMP 300
#define HEATER_1_MAXTEMP 300
#define HEATER_2_MAXTEMP 300
#define BED_MAXTEMP 300

// If your bed has low resistance e.g. .6 ohm and throws the fuse you can duty cycle it to reduce the
// average current. The value should be an integer and the heat bed will be turned on for 1 interval of
```

```
// HEATER_BED_DUTY_CYCLE_DIVIDER intervals.
//#define HEATER_BED_DUTY_CYCLE_DIVIDER 4

// If you want the M105 heater power reported in watts, define the BED_WATTS, and (shared for all extruders)
EXTRUDER_WATTS
//#define EXTRUDER_WATTS (12.0*12.0/6.7) //  P=I^2/R
//#define BED_WATTS (12.0*12.0/1.1)      // P=I^2/R

// PID settings:
// Comment the following line to disable PID and enable bang-bang.
#define PIDTEMP
#define BANG_MAX 255 // limits current to nozzle while in bang-bang mode; 255=full current
#define PID_MAX BANG_MAX // limits current to nozzle while PID is active (see PID_FUNCTIONAL_RANGE below); 255=full
current
#ifdef PIDTEMP
  //#define PID_DEBUG // Sends debug data to the serial port.
  //#define PID_OPENLOOP 1 // Puts PID in open loop. M104/M140 sets the output power from 0 to PID_MAX
  //#define SLOW_PWM_HEATERS // PWM with very low frequency (roughly 0.125Hz=8s) and minimum state time of
approximately 1s useful for heaters driven by a relay
  #define PID_FUNCTIONAL_RANGE 10 // If the temperature difference between the target temperature and the actual
temperature
                          // is more then PID_FUNCTIONAL_RANGE then the PID will be shut off and the heater will be set to
min/max.
  #define PID_INTEGRAL_DRIVE_MAX PID_MAX  //limit for the integral term
  #define K1 0.95 //smoothing factor within the PID
  #define PID_dT ((OVERSAMPLENR * 10.0)/(F_CPU / 64.0 / 256.0)) //sampling period of the temperature routine

// If you are using a pre-configured hotend then you can use one of the value sets by uncommenting it
// Ultimaker
    #define  DEFAULT_Kp 40.925
    #define  DEFAULT_Ki 4.875
    #define  DEFAULT_Kd 86.085

// MakerGear
//    #define  DEFAULT_Kp 7.0
//    #define  DEFAULT_Ki 0.1
//    #define  DEFAULT_Kd 12

// Mendel Parts V9 on 12V
//    #define  DEFAULT_Kp 63.0
//    #define  DEFAULT_Ki 2.25
//    #define  DEFAULT_Kd 440
#endif // PIDTEMP

// Bed Temperature Control
// Select PID or bang-bang with PIDTEMPBED. If bang-bang, BED_LIMIT_SWITCHING will enable hysteresis
//
// Uncomment this to enable PID on the bed. It uses the same frequency PWM as the extruder.
// If your PID_dT above is the default, and correct for your hardware/configuration, that means 7.689Hz,
// which is fine for driving a square wave into a resistive load and does not significantly impact you FET heating.
// This also works fine on a Fotek SSR-10DA Solid State Relay into a 250W heater.
// If your configuration is significantly different than this and you don't understand the issues involved, you probably
// shouldn't use bed PID until someone else verifies your hardware works.
// If this is enabled, find your own PID constants below.
//#define PIDTEMPBED
//
//#define BED_LIMIT_SWITCHING

// This sets the max power delivered to the bed, and replaces the HEATER_BED_DUTY_CYCLE_DIVIDER option.
// all forms of bed control obey this (PID, bang-bang, bang-bang with hysteresis)
// setting this to anything other than 255 enables a form of PWM to the bed just like HEATER_BED_DUTY_CYCLE_DIVIDER
did,
// so you shouldn't use it unless you are OK with PWM on your bed.  (see the comment on enabling PIDTEMPBED)
#define MAX_BED_POWER 255 // limits duty cycle to bed; 255=full current

#ifdef PIDTEMPBED
//120v 250W silicone heater into 4mm borosilicate (MendelMax 1.5+)
//from FOPDT model - kp=.39 Tp=405 Tdead=66, Tc set to 79.2, aggressive factor of .15 (vs .1, 1, 10)
    #define  DEFAULT_bedKp 10.00
    #define  DEFAULT_bedKi .023
```

```
   #define  DEFAULT_bedKd 305.4

//120v 250W silicone heater into 4mm borosilicate (MendelMax 1.5+)
//from pidautotune
//    #define  DEFAULT_bedKp 97.1
//    #define  DEFAULT_bedKi 1.41
//    #define  DEFAULT_bedKd 1675.16

// FIND YOUR OWN: "M303 E-1 C8 S90" to run autotune on the bed at 90 degreesC for 8 cycles.
#endif // PIDTEMPBED



//this prevents dangerous Extruder moves, i.e. if the temperature is under the limit
//can be software-disabled for whatever purposes by
//#define PREVENT_DANGEROUS_EXTRUDE
//if PREVENT_DANGEROUS_EXTRUDE is on, you can still disable (uncomment) very long bits of extrusion separately.
#define PREVENT_LENGTHY_EXTRUDE

#define EXTRUDE_MINTEMP 20
#define EXTRUDE_MAXLENGTH (X_MAX_LENGTH+Y_MAX_LENGTH) //prevent extrusion of very large distances.

/*================== Thermal Runaway Protection ==============================
This is a feature to protect your printer from burn up in flames if it has
a thermistor coming off place (this happened to a friend of mine recently and
motivated me writing this feature).

The issue: If a thermistor come off, it will read a lower temperature than actual.
The system will turn the heater on forever, burning up the filament and anything
else around.

After the temperature reaches the target for the first time, this feature will
start measuring for how long the current temperature stays below the target
minus _HYSTERESIS (set_temperature - THERMAL_RUNAWAY_PROTECTION_HYSTERESIS).

If it stays longer than _PERIOD, it means the thermistor temperature
cannot catch up with the target, so something *may be* wrong. Then, to be on the
safe side, the system will he halt.

Bear in mind the count down will just start AFTER the first time the
thermistor temperature is over the target, so you will have no problem if
your extruder heater takes 2 minutes to hit the target on heating.

*/
// If you want to enable this feature for all your extruder heaters,
// uncomment the 2 defines below:

// Parameters for all extruder heaters
//#define THERMAL_RUNAWAY_PROTECTION_PERIOD 40 //in seconds
//#define THERMAL_RUNAWAY_PROTECTION_HYSTERESIS 4 // in degree Celsius

// If you want to enable this feature for your bed heater,
// uncomment the 2 defines below:

// Parameters for the bed heater
//#define THERMAL_RUNAWAY_PROTECTION_BED_PERIOD 20 //in seconds
//#define THERMAL_RUNAWAY_PROTECTION_BED_HYSTERESIS 2 // in degree Celsius
//===========================================================================



//===========================================================================
//=============================Mechanical Settings===========================
//===========================================================================

// Uncomment the following line to enable CoreXY kinematics
// #define COREXY

// coarse Endstop Settings
#define ENDSTOPPULLUPS // Comment this out (using // at the start of the line) to disable the endstop pullup resistors
```

```
#ifndef ENDSTOPPULLUPS
  // fine endstop settings: Individual pullups. will be ignored if ENDSTOPPULLUPS is defined
  // #define ENDSTOPPULLUP_XMAX
  // #define ENDSTOPPULLUP_YMAX
  // #define ENDSTOPPULLUP_ZMAX
  // #define ENDSTOPPULLUP_XMIN
  // #define ENDSTOPPULLUP_YMIN
  // #define ENDSTOPPULLUP_ZMIN
#endif

#ifdef ENDSTOPPULLUPS
  #define ENDSTOPPULLUP_XMAX
  #define ENDSTOPPULLUP_YMAX
  #define ENDSTOPPULLUP_ZMAX
  #define ENDSTOPPULLUP_XMIN
  #define ENDSTOPPULLUP_YMIN
  #define ENDSTOPPULLUP_ZMIN
#endif

// The pullups are needed if you directly connect a mechanical endswitch between the signal and ground pins.
const bool X_MIN_ENDSTOP_INVERTING = false; // set to true to invert the logic of the endstop.
const bool Y_MIN_ENDSTOP_INVERTING = false; // set to true to invert the logic of the endstop.
const bool Z_MIN_ENDSTOP_INVERTING = false; // set to true to invert the logic of the endstop.
const bool X_MAX_ENDSTOP_INVERTING = true; // set to true to invert the logic of the endstop.
const bool Y_MAX_ENDSTOP_INVERTING = true; // set to true to invert the logic of the endstop.
const bool Z_MAX_ENDSTOP_INVERTING = true; // set to true to invert the logic of the endstop.
//#define DISABLE_MAX_ENDSTOPS
//#define DISABLE_MIN_ENDSTOPS

// Disable max endstops for compatibility with endstop checking routine
#if defined(COREXY) && !defined(DISABLE_MAX_ENDSTOPS)
  #define DISABLE_MAX_ENDSTOPS
#endif

// For Inverting Stepper Enable Pins (Active Low) use 0, Non Inverting (Active High) use 1
#define X_ENABLE_ON 0
#define Y_ENABLE_ON 0
#define Z_ENABLE_ON 0
#define E_ENABLE_ON 0 // For all extruders

// Disables axis when it's not being used.
#define DISABLE_X false
#define DISABLE_Y false
#define DISABLE_Z true
#define DISABLE_E false // For all extruders
#define DISABLE_INACTIVE_EXTRUDER true //disable only inactive extruders and keep active extruder enabled

#define INVERT_X_DIR false    // for Mendel set to false, for Orca set to true
#define INVERT_Y_DIR false    // for Mendel set to true, for Orca set to false
#define INVERT_Z_DIR false     // for Mendel set to false, for Orca set to true
#define INVERT_E0_DIR true   // for direct drive extruder v9 set to true, for geared extruder set to false
#define INVERT_E1_DIR false    // for direct drive extruder v9 set to true, for geared extruder set to false
#define INVERT_E2_DIR false   // for direct drive extruder v9 set to true, for geared extruder set to false

// ENDSTOP SETTINGS:
// Sets direction of endstops when homing; 1=MAX, -1=MIN
#define X_HOME_DIR -1
#define Y_HOME_DIR -1
#define Z_HOME_DIR -1

#define min_software_endstops true // If true, axis won't move to coordinates less than HOME_POS.
#define max_software_endstops true  // If true, axis won't move to coordinates greater than the defined lengths below.

// Travel limits after homing
#define X_MAX_POS 190
#define X_MIN_POS 0
#define Y_MAX_POS 190
#define Y_MIN_POS 0
#define Z_MAX_POS 190
#define Z_MIN_POS 0.23
```

```
#define X_MAX_LENGTH (X_MAX_POS - X_MIN_POS)
#define Y_MAX_LENGTH (Y_MAX_POS - Y_MIN_POS)
#define Z_MAX_LENGTH (Z_MAX_POS - Z_MIN_POS)
//============================= Bed Auto Leveling ============================

//#define ENABLE_AUTO_BED_LEVELING // Delete the comment to enable (remove // at the start of the line)
#define Z_PROBE_REPEATABILITY_TEST  // If not commented out, Z-Probe Repeatability test will be included if Auto Bed Leveling is Enabled.

#ifdef ENABLE_AUTO_BED_LEVELING

// There are 2 different ways to pick the X and Y locations to probe:

//  - "grid" mode
//    Probe every point in a rectangular grid
//    You must specify the rectangle, and the density of sample points
//    This mode is preferred because there are more measurements.
//    It used to be called ACCURATE_BED_LEVELING but "grid" is more descriptive

//  - "3-point" mode
//    Probe 3 arbitrary points on the bed (that aren't collinear)
//    You must specify the X & Y coordinates of all 3 points

 #define AUTO_BED_LEVELING_GRID
 // with AUTO_BED_LEVELING_GRID, the bed is sampled in a
 // AUTO_BED_LEVELING_GRID_POINTSxAUTO_BED_LEVELING_GRID_POINTS grid
 // and least squares solution is calculated
 // Note: this feature occupies 10'206 byte
 #ifdef AUTO_BED_LEVELING_GRID

   // set the rectangle in which to probe
   #define LEFT_PROBE_BED_POSITION 15
   #define RIGHT_PROBE_BED_POSITION 170
   #define BACK_PROBE_BED_POSITION 180
   #define FRONT_PROBE_BED_POSITION 20

    // set the number of grid points per dimension
    // I wouldn't see a reason to go above 3 (=9 probing points on the bed)
   #define AUTO_BED_LEVELING_GRID_POINTS 2


 #else  // not AUTO_BED_LEVELING_GRID
   // with no grid, just probe 3 arbitrary points.  A simple cross-product
   // is used to estimate the plane of the print bed

     #define ABL_PROBE_PT_1_X 15
     #define ABL_PROBE_PT_1_Y 180
     #define ABL_PROBE_PT_2_X 15
     #define ABL_PROBE_PT_2_Y 20
     #define ABL_PROBE_PT_3_X 170
     #define ABL_PROBE_PT_3_Y 20

 #endif // AUTO_BED_LEVELING_GRID


 // these are the offsets to the probe relative to the extruder tip (Hotend - Probe)
 // X and Y offsets must be integers
 #define X_PROBE_OFFSET_FROM_EXTRUDER -25
 #define Y_PROBE_OFFSET_FROM_EXTRUDER -29
 #define Z_PROBE_OFFSET_FROM_EXTRUDER -12.35

 #define Z_RAISE_BEFORE_HOMING 4       // (in mm) Raise Z before homing (G28) for Probe Clearance.
                                       // Be sure you have this distance over your Z_MAX_POS in case

 #define XY_TRAVEL_SPEED 8000         // X and Y axis travel speed between probes, in mm/min

 #define Z_RAISE_BEFORE_PROBING 15    //How much the extruder will be raised before traveling to the first probing point.
 #define Z_RAISE_BETWEEN_PROBINGS 5  //How much the extruder will be raised when traveling from between next probing points
```

```
  //#define Z_PROBE_SLED // turn on if you have a z-probe mounted on a sled like those designed by Charles Bell
  //#define SLED_DOCKING_OFFSET 5 // the extra distance the X axis must travel to pickup the sled. 0 should be fine but you
can push it further if you'd like.

  //If defined, the Probe servo will be turned on only during movement and then turned off to avoid jerk
  //The value is the delay to turn the servo off after powered on - depends on the servo speed; 300ms is good value, but you
can try lower it.
  // You MUST HAVE the SERVO_ENDSTOPS defined to use here a value higher than zero otherwise your code will not
compile.

//  #define PROBE_SERVO_DEACTIVATION_DELAY 300


//If you have enabled the Bed Auto Leveling and are using the same Z Probe for Z Homing,
//it is highly recommended you let this Z_SAFE_HOMING enabled!!!

  #define Z_SAFE_HOMING   // This feature is meant to avoid Z homing with probe outside the bed area.
                          // When defined, it will:
                          // - Allow Z homing only after X and Y homing AND stepper drivers still enabled
                          // - If stepper drivers timeout, it will need X and Y homing again before Z homing
                          // - Position the probe in a defined XY point before Z Homing when homing all axis (G28)
                          // - Block Z homing only when the probe is outside bed area.

  #ifdef Z_SAFE_HOMING

    #define Z_SAFE_HOMING_X_POINT (X_MAX_LENGTH/2)    // X point for Z homing when homing all axis (G28)
    #define Z_SAFE_HOMING_Y_POINT (Y_MAX_LENGTH/2)    // Y point for Z homing when homing all axis (G28)

  #endif

  #ifdef AUTO_BED_LEVELING_GRID          // Check if Probe_Offset * Grid Points is greater than Probing Range
    #if X_PROBE_OFFSET_FROM_EXTRUDER < 0
      #if (-(X_PROBE_OFFSET_FROM_EXTRUDER * AUTO_BED_LEVELING_GRID_POINTS) >=
(RIGHT_PROBE_BED_POSITION - LEFT_PROBE_BED_POSITION))
          #error "The X axis probing range is not enough to fit all the points defined in AUTO_BED_LEVELING_GRID_POINTS"
        #endif
      #else
      #if ((X_PROBE_OFFSET_FROM_EXTRUDER * AUTO_BED_LEVELING_GRID_POINTS) >=
(RIGHT_PROBE_BED_POSITION - LEFT_PROBE_BED_POSITION))
          #error "The X axis probing range is not enough to fit all the points defined in AUTO_BED_LEVELING_GRID_POINTS"
        #endif
      #endif
    #if Y_PROBE_OFFSET_FROM_EXTRUDER < 0
      #if (-(Y_PROBE_OFFSET_FROM_EXTRUDER * AUTO_BED_LEVELING_GRID_POINTS) >=
(BACK_PROBE_BED_POSITION - FRONT_PROBE_BED_POSITION))
          #error "The Y axis probing range is not enough to fit all the points defined in AUTO_BED_LEVELING_GRID_POINTS"
        #endif
      #else
      #if ((Y_PROBE_OFFSET_FROM_EXTRUDER * AUTO_BED_LEVELING_GRID_POINTS) >=
(BACK_PROBE_BED_POSITION - FRONT_PROBE_BED_POSITION))
          #error "The Y axis probing range is not enough to fit all the points defined in AUTO_BED_LEVELING_GRID_POINTS"
        #endif
      #endif


  #endif

#endif // ENABLE_AUTO_BED_LEVELING


// The position of the homing switches
//#define MANUAL_HOME_POSITIONS  // If defined, MANUAL_*_HOME_POS below will be used
//#define BED_CENTER_AT_0_0  // If defined, the center of the bed is at (X=0, Y=0)

//Manual homing switch locations:
// For deltabots this means top and center of the Cartesian print volume.
#define MANUAL_X_HOME_POS 0
#define MANUAL_Y_HOME_POS 0
#define MANUAL_Z_HOME_POS 0.25
```

```
//#define MANUAL_Z_HOME_POS 402 // For delta: Distance between nozzle and print surface after homing.

//// MOVEMENT SETTINGS
#define NUM_AXIS 4 // The axis order in all axis related arrays is X, Y, Z, E
#define HOMING_FEEDRATE {50*60, 50*60, 4*60, 0}  // set the homing speeds (mm/min)

// default settings

#define DEFAULT_AXIS_STEPS_PER_UNIT   {100,100,3200/0.8,9}  // default steps per unit for Ultimaker
#define DEFAULT_MAX_FEEDRATE          {500, 500, 3, 100}    // (mm/sec)
#define DEFAULT_MAX_ACCELERATION      {9000,9000,30,10000}    // X, Y, Z, E maximum start speed for accelerated moves. E default values are good for Skeinforge 40+, for older versions raise them a lot.

#define DEFAULT_ACCELERATION          3000    // X, Y, Z and E max acceleration in mm/s^2 for printing moves
#define DEFAULT_RETRACT_ACCELERATION  3000   // X, Y, Z and E max acceleration in mm/s^2 for retracts

// Offset of the extruders (uncomment if using more than one and relying on firmware to position when changing).
// The offset has to be X=0, Y=0 for the extruder 0 hotend (default extruder).
// For the other hotends it is their distance from the extruder 0 hotend.
// #define EXTRUDER_OFFSET_X {0.0, 20.00} // (in mm) for each extruder, offset of the hotend on the X axis
// #define EXTRUDER_OFFSET_Y {0.0, 5.00}  // (in mm) for each extruder, offset of the hotend on the Y axis

// The speed change that does not require acceleration (i.e. the software might assume it can be done instantaneously)
#define DEFAULT_XYJERK                20.0    // (mm/sec)
#define DEFAULT_ZJERK                 0.4     // (mm/sec)
#define DEFAULT_EJERK                 5.0    // (mm/sec)

//===========================================================================
//=============================Additional Features===========================
//===========================================================================

// Custom M code points
#define CUSTOM_M_CODES
#ifdef CUSTOM_M_CODES
  #define CUSTOM_M_CODE_SET_Z_PROBE_OFFSET 851
  #define Z_PROBE_OFFSET_RANGE_MIN -15
  #define Z_PROBE_OFFSET_RANGE_MAX -5
#endif


// EEPROM
// The microcontroller can store settings in the EEPROM, e.g. max velocity...
// M500 - stores parameters in EEPROM
// M501 - reads parameters from EEPROM (if you need reset them after you changed them temporarily).
// M502 - reverts to the default "factory settings".  You still need to store them in EEPROM afterwards if you want to.
//define this to enable EEPROM support
//#define EEPROM_SETTINGS
//to disable EEPROM Serial responses and decrease program space by ~1700 byte: comment this out:
// please keep turned on if you can.
//#define EEPROM_CHITCHAT

// Preheat Constants
#define PLA_PREHEAT_HOTEND_TEMP 210
#define PLA_PREHEAT_HPB_TEMP 50
#define PLA_PREHEAT_FAN_SPEED 0   // Insert Value between 0 and 255

#define ABS_PREHEAT_HOTEND_TEMP 255
#define ABS_PREHEAT_HPB_TEMP 100
#define ABS_PREHEAT_FAN_SPEED 0   // Insert Value between 0 and 255

#define HIPS_PREHEAT_HOTEND_TEMP 220
#define HIPS_PREHEAT_HPB_TEMP 100
#define HIPS_PREHEAT_FAN_SPEED 0

#define PP_PREHEAT_HOTEND_TEMP 254
#define PP_PREHEAT_HPB_TEMP 100
#define PP_PREHEAT_FAN_SPEED 0

#define PET_PREHEAT_HOTEND_TEMP 240
#define PET_PREHEAT_HPB_TEMP 90
```

```
#define PET_PREHEAT_FAN_SPEED 0

#define FLEX_PREHEAT_HOTEND_TEMP 230
#define FLEX_PREHEAT_HPB_TEMP 50
#define FLEX_PREHEAT_FAN_SPEED 0

//LCD and SD support
#define ULTRA_LCD  //general LCD support, also 16x2
//#define DOGLCD  // Support for SPI LCD 128x64 (Controller ST7565R graphic Display Family)
#define SDSUPPORT // Enable SD Card Support in Hardware Console
//#define SDSLOW // Use slower SD transfer mode (not normally needed - uncomment if you're getting volume init error)
#define SD_CHECK_AND_RETRY // Use CRC checks and retries on the SD communication
//#define ENCODER_PULSES_PER_STEP 1 // Increase if you have a high resolution encoder
//#define ENCODER_STEPS_PER_MENU_ITEM 5 // Set according to ENCODER_PULSES_PER_STEP or your liking
//#define ULTIMAKERCONTROLLER //as available from the Ultimaker online store.
//#define ULTIPANEL  //the UltiPanel as on Thingiverse
//#define LCD_FEEDBACK_FREQUENCY_HZ 1000     // this is the tone frequency the buzzer plays when on UI feedback. i.e. Screen Click
//#define LCD_FEEDBACK_FREQUENCY_DURATION_MS 100 // the duration the buzzer plays the UI feedback sound. i.e. Screen Click

// The MaKr3d Makr-Panel with graphic controller and SD support
// http://reprap.org/wiki/MaKr3d_MaKrPanel
//#define MAKRPANEL

// The RepRapDiscount Smart Controller (white PCB)
// http://reprap.org/wiki/RepRapDiscount_Smart_Controller
#define REPRAP_DISCOUNT_SMART_CONTROLLER

// The GADGETS3D G3D LCD/SD Controller (blue PCB)
// http://reprap.org/wiki/RAMPS_1.3/1.4_GADGETS3D_Shield_with_Panel
//#define G3D_PANEL

// The RepRapDiscount FULL GRAPHIC Smart Controller (quadratic white PCB)
// http://reprap.org/wiki/RepRapDiscount_Full_Graphic_Smart_Controller
//
// ==> REMEMBER TO INSTALL U8glib to your ARDUINO library folder: http://code.google.com/p/u8glib/wiki/u8glib
//#define REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER

// The RepRapWorld REPRAPWORLD_KEYPAD v1.1
// http://reprapworld.com/?products_details&products_id=202&cPath=1591_1626
//#define REPRAPWORLD_KEYPAD
//#define REPRAPWORLD_KEYPAD_MOVE_STEP 10.0 // how much should be moved when a key is pressed, eg 10.0 means 10mm per click

// The Elefu RA Board Control Panel
// http://www.elefu.com/index.php?route=product/product&product_id=53
// REMEMBER TO INSTALL LiquidCrystal_I2C.h in your ARDUINO library folder: https://github.com/kiyoshigawa/LiquidCrystal_I2C
//#define RA_CONTROL_PANEL

//automatic expansion
#if defined (MAKRPANEL)
 #define DOGLCD
 #define SDSUPPORT
 #define ULTIPANEL
 #define NEWPANEL
 #define DEFAULT_LCD_CONTRAST 17
#endif

#if defined (REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER)
 #define DOGLCD
 #define U8GLIB_ST7920
 #define REPRAP_DISCOUNT_SMART_CONTROLLER
#endif

#if defined(ULTIMAKERCONTROLLER) || defined(REPRAP_DISCOUNT_SMART_CONTROLLER) || defined(G3D_PANEL)
 #define ULTIPANEL
 #define NEWPANEL
#endif
```

```
#if defined(REPRAPWORLD_KEYPAD)
  #define NEWPANEL
  #define ULTIPANEL
#endif
#if defined(RA_CONTROL_PANEL)
 #define ULTIPANEL
 #define NEWPANEL
 #define LCD_I2C_TYPE_PCA8574
 #define LCD_I2C_ADDRESS 0x27   // I2C Address of the port expander
#endif

//I2C PANELS

//#define LCD_I2C_SAINSMART_YWROBOT
#ifdef LCD_I2C_SAINSMART_YWROBOT
  // This uses the LiquidCrystal_I2C library ( https://bitbucket.org/fmalpartida/new-liquidcrystal/wiki/Home )
  // Make sure it is placed in the Arduino libraries directory.
  #define LCD_I2C_TYPE_PCF8575
  #define LCD_I2C_ADDRESS 0x27   // I2C Address of the port expander
  #define NEWPANEL
  #define ULTIPANEL
#endif

// PANELOLU2 LCD with status LEDs, separate encoder and click inputs
//#define LCD_I2C_PANELOLU2
#ifdef LCD_I2C_PANELOLU2
  // This uses the LiquidTWI2 library v1.2.3 or later ( https://github.com/lincomatic/LiquidTWI2 )
  // Make sure the LiquidTWI2 directory is placed in the Arduino or Sketchbook libraries subdirectory.
  // (v1.2.3 no longer requires you to define PANELOLU in the LiquidTWI2.h library header file)
  // Note: The PANELOLU2 encoder click input can either be directly connected to a pin
  //       (if BTN_ENC defined to != -1) or read through I2C (when BTN_ENC == -1).
  #define LCD_I2C_TYPE_MCP23017
  #define LCD_I2C_ADDRESS 0x20 // I2C Address of the port expander
  #define LCD_USE_I2C_BUZZER //comment out to disable buzzer on LCD
  #define NEWPANEL
  #define ULTIPANEL

  #ifndef ENCODER_PULSES_PER_STEP
      #define ENCODER_PULSES_PER_STEP 4
  #endif

  #ifndef ENCODER_STEPS_PER_MENU_ITEM
      #define ENCODER_STEPS_PER_MENU_ITEM 1
  #endif


  #ifdef LCD_USE_I2C_BUZZER
      #define LCD_FEEDBACK_FREQUENCY_HZ 1000
      #define LCD_FEEDBACK_FREQUENCY_DURATION_MS 100
  #endif

#endif

// Panucatt VIKI LCD with status LEDs, integrated click & L/R/U/P buttons, separate encoder inputs
//#define LCD_I2C_VIKI
#ifdef LCD_I2C_VIKI
  // This uses the LiquidTWI2 library v1.2.3 or later ( https://github.com/lincomatic/LiquidTWI2 )
  // Make sure the LiquidTWI2 directory is placed in the Arduino or Sketchbook libraries subdirectory.
  // Note: The pause/stop/resume LCD button pin should be connected to the Arduino
  //       BTN_ENC pin (or set BTN_ENC to -1 if not used)
  #define LCD_I2C_TYPE_MCP23017
  #define LCD_I2C_ADDRESS 0x20 // I2C Address of the port expander
  #define LCD_USE_I2C_BUZZER //comment out to disable buzzer on LCD (requires LiquidTWI2 v1.2.3 or later)
  #define NEWPANEL
  #define ULTIPANEL
#endif

// Shift register panels
// ----------------------
```

```
// 2 wire Non-latching LCD SR from:
// https://bitbucket.org/fmalpartida/new-liquidcrystal/wiki/schematics#!shiftregister-connection

//#define SAV_3DLCD
#ifdef SAV_3DLCD
   #define SR_LCD_2W_NL    // Non latching 2 wire shiftregister
   #define NEWPANEL
   #define ULTIPANEL
#endif


#ifdef ULTIPANEL
//  #define NEWPANEL  //enable this if you have a click-encoder panel
  #define SDSUPPORT
  #define ULTRA_LCD
  #ifdef DOGLCD // Change number of lines to match the DOG graphic display
    #define LCD_WIDTH 20
    #define LCD_HEIGHT 5
  #else
    #define LCD_WIDTH 20
    #define LCD_HEIGHT 4
  #endif
#else //no panel but just LCD
  #ifdef ULTRA_LCD
  #ifdef DOGLCD // Change number of lines to match the 128x64 graphics display
    #define LCD_WIDTH 20
    #define LCD_HEIGHT 5
  #else
    #define LCD_WIDTH 16
    #define LCD_HEIGHT 2
  #endif
  #endif
#endif

// default LCD contrast for dogm-like LCD displays
#ifdef DOGLCD
# ifndef DEFAULT_LCD_CONTRAST
#  define DEFAULT_LCD_CONTRAST 32
# endif
#endif

// Increase the FAN pwm frequency. Removes the PWM noise but increases heating in the FET/Arduino
//#define FAST_PWM_FAN

// Temperature status LEDs that display the hotend and bet temperature.
// If all hotends and bed temperature and temperature set point are < 54C then the BLUE led is on.
// Otherwise the RED led is on. There is 1C hysteresis.
//#define TEMP_STAT_LEDS

// Use software PWM to drive the fan, as for the heaters. This uses a very low frequency
// which is not ass annoying as with the hardware PWM. On the other hand, if this frequency
// is too low, you should also increment SOFT_PWM_SCALE.
//#define FAN_SOFT_PWM

// Incrementing this by 1 will double the software PWM frequency,
// affecting heaters, and the fan if FAN_SOFT_PWM is enabled.
// However, control resolution will be halved for each increment;
// at zero value, there are 128 effective control positions.
#define SOFT_PWM_SCALE 0

// M240  Triggers a camera by emulating a Canon RC-1 Remote
// Data from: http://www.doc-diy.net/photo/rc-1_hacked/
// #define PHOTOGRAPH_PIN     23

// SF send wrong arc g-codes when using Arc Point as fillet procedure
//#define SF_ARC_FIX

// Support for the BariCUDA Paste Extruder.
//#define BARICUDA
```

```
//define BlinkM/CyzRgb Support
//#define BLINKM

/**********************************************************************\
* R/C SERVO support
* Sponsored by TrinityLabs, Reworked by codexmas
**********************************************************************/

// Number of servos
//
// If you select a configuration below, this will receive a default value and does not need to be set manually
// set it manually if you have more servos than extruders and wish to manually control some
// leaving it undefined or defining as 0 will disable the servo subsystem
// If unsure, leave commented / disabled
//
//#define NUM_SERVOS 3 // Servo index starts with 0 for M280 command

// Servo Endstops
//
// This allows for servo actuated endstops, primary usage is for the Z Axis to eliminate calibration or bed height changes.
// Use M206 command to correct for switch height offset to actual nozzle height. Store that setting with M500.
//
//#define SERVO_ENDSTOPS {-1, -1, 0} // Servo index for X, Y, Z. Disable with -1
//#define SERVO_ENDSTOP_ANGLES {0,0, 0,0, 70,0} // X,Y,Z Axis Extend and Retract angles

/**********************************************************************\
 * Support for a filament diameter sensor
 * Also allows adjustment of diameter at print time (vs  at slicing)
 * Single extruder only at this point (extruder 0)
 *
 * Motherboards
 * 34 - RAMPS1.4 - uses Analog input 5 on the AUX2 connector
 * 81 - Printrboard - Uses Analog input 2 on the Exp1 connector (version B,C,D,E)
 * 301 - Rambo  - uses Analog input 3
 * Note may require analog pins to be defined for different motherboards
 **********************************************************************/
// Uncomment below to enable
//#define FILAMENT_SENSOR

#define FILAMENT_SENSOR_EXTRUDER_NUM        0  //The number of the extruder that has the filament sensor (0,1,2)
#define MEASUREMENT_DELAY_CM          14  //measurement delay in cm.  This is the distance from filament sensor to
middle of barrel

#define DEFAULT_NOMINAL_FILAMENT_DIA  1.0  //Enter the diameter (in mm) of the filament generally used (3.0 mm or
1.75 mm) - this is then used in the slicer software.  Used for sensor reading validation
#define MEASURED_UPPER_LIMIT          3.30  //upper limit factor used for sensor reading validation in mm
#define MEASURED_LOWER_LIMIT          1.90  //lower limit factor for sensor reading validation in mm
#define MAX_MEASUREMENT_DELAY         20  //delay buffer size in bytes (1 byte = 1cm)- limits maximum measurement
delay allowable (must be larger than MEASUREMENT_DELAY_CM  and lower number saves RAM)

//defines used in the code
#define DEFAULT_MEASURED_FILAMENT_DIA  DEFAULT_NOMINAL_FILAMENT_DIA  //set measured to nominal initially

//When using an LCD, uncomment the line below to display the Filament sensor data on the last line instead of status.  Status
will appear for 5 sec.
//#define FILAMENT_LCD_DISPLAY

#include "Configuration_adv.h"
#include "thermistortables.h"

#endif //__CONFIGURATION_H
```

A11. Syringe Pump Testing Code

```
/*This is test code to calibrate a syringe pump powered by a stepper motor.
The pump stepper is controlled through an Adafruit motor shield v2.
Select desired test using booleans.

Steps/mL Calibration:
Load any fluid of known density into the syringe. Measure extruded mass w/ balance.
Calculate:  Steps/mL = (testSteps)/(extrudedVolume) | where extrudedVolume = extrudedMass(g) / density(g/mL)

Min/Max Velocity:
Load any fluid into syringe. Run test & open serial monitor.
Wait for motor to stall and record velocity.

Fluid Retraction:
Set to TRUE to withdraw the exact amount of fluid extruded during any of the above tests.
Increase testSteps to retract larger volume.*/

#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_MS_PWMServoDriver.h"

// Create the motor shield object with the default I2C address
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
//Connect a stepper motor with 200 steps per revolution (1.8 degree)to motor port #2 (M3 //and M4)
Adafruit_StepperMotor *myMotor = AFMS.getStepper(200, 2);

int stepCounter = 0;
int testSteps = 500;
int stepsPermL = 1279;                    //Determined during calibration

boolean calTest = TRUE;                   //set TRUE to run steps/mL calibration test
boolean maxSpeedTest = FALSE;  //set TRUE to run max speed test
boolean minSpeedTest = FALSE;   //set TRUE to run min speed test
boolean retract = FALSE;                  //set TRUE to refill syringe

void setup() {
  Serial.begin(9600);                     //set up Serial communication at 9600 bps
  Serial.println("Stepper test!");
  AFMS.begin();                           //create with the default frequency 1.6KHz
  myMotor->setSpeed(10);}           //10 rpm default

void loop() {
//Steps/mL calibration
  if (calTest == TRUE) {
    myMotor->step(testSteps, BACKWARD, MICROSTEP);          //Backward step = syringe
    myMotor->release();                                                    //extrude
    stepCounter = testSteps;
    calTest = FALSE;}*/

//Maximum speed test
  if (maxSpeedTest == TRUE) {
    for (int minSpeed = 0; speed < 10; speed += 0.1) {
      myMotor->setSpeed(minSpeed);
      myMotor->step(20, BACKWARD, MICROSTEP);
      stepCounter += 20;
      Serial.print("Speed: ");
      Serial.print(minSpeed);
      Serial.println(" revolutions per minute");
      delay(10); }
    myMotor->release();
    maxSpeedTest == FALSE;}

//Minimum speed test
  if (minSpeedTest == TRUE) {
    for (int minSpeed = 10; speed > 0; count -= 0.1) {
      myMotor->setSpeed(minSpeed);
      myMotor->step(20, BACKWARD, MICROSTEP);
      stepCounter += 20;
      Serial.print("Speed: ");
      Serial.print(minSpeed);
```

```
    Serial.println(" revolutions per minute");
    delay(10); }
  myMotor->release();
  minSpeedTest == FALSE; }

//Withdrawing fluid (load syringe)
 if (retract == TRUE) {
  if (stepCounter < 10) {
    stepCounter = testSteps; }
  myMotor->step(stepCounter, FORWARD, SINGLE);  //Forward steps = syringe withdraw
  myMotor->release();
  retract == FALSE; } }
```

A12. Peltier Element Control Code

```
/* This code controls the status of a relay switch that delivers power to a TEC.
Temperature is measured by way of a thermistor in a voltage divider circuit.
Control architecture can use a PID algorithm or be direct, depending on bool setting below.
Removing power from the relay enables measurement of temperature alone.
Temperature and PID output values (if applicable) are printed to the serial bus, for plotting
using Processing.*/

#include <PID_v1.h>

//Initialize temperature measurement parameters:
#define ThermistorPin A0            // identify analog pin to measure at
#define ThermistorResistance 10000  // resistance at 25 degrees C (nominal)
#define RoomTemperature 25          // Temp for nominal resistance
#define NumberSamples 5                     // #samples to average temp over. Tradeoff:
                                            // efficiency vs smoothness
#define BCoefficient 3950           // Beta coefficient of the thermistor
#define ResistorVal 10000           // Resistance of comparator resistor
int samples[NumberSamples];         // initialize sampling array

//Define pins for indicator LED and Relay
#define LEDPin 13
#define RelayPin 3

//Initialize PID parameters
boolean PIDon = TRUE;                       //Set TRUE to use PID, Set FALSE to use direct or //to measure
                                            temperature alone
                                            //Initialize PID parameters. Alter Setpoint, Kp, Ki, & //Kd to calibrate.
double Setpoint = 4, Input, Output, Kp=100, Ki=5, Kd=1;
int WindowSize = 500;                       //Size of relay window. PID will activate Relay for a //time (in ms)
                                            proportional to calculated output, //ranging from 0 to the WindowSize.
unsigned long windowStartTime;    //Initialize time tracker to shift relay window.

                                            //Create PID object, operating in REVERSE mode //(increase output to
                                            decrease input)
PID myPID(&Input, &Output, &Setpoint,Kp,Ki,Kd, REVERSE);

void setup()
{
  windowStartTime = millis();               //Establish starting point for relay window
  Serial.begin(9600);               //Begin serial communication
  analogReference(EXTERNAL);        //Use external voltage reference (see circuit
                                            //diagram) for improved temp measurement

  pinMode(LEDPin,OUTPUT);           //Set the I/O status of Arduino pins
  pinMode(RelayPin,OUTPUT);
  pinMode(ThermistorPin,INPUT);

  digitalWrite(RelayPin,HIGH);              //Open relay circuit to halt current flow

  myPID.SetOutputLimits(0, WindowSize);  //tell the PID to range between 0 and the full
                                            //window size
  myPID.SetMode(AUTOMATIC);         //turn the PID on

  blink(LEDPin);
}

void loop()
{
//Measure the current temperature and compute the PID output
  Input = getTemp(NumberSamples,ThermistorPin,ResistorVal,ThermistorResistance,RoomTemperature,BCoefficient);

  if(PIDon == TRUE)
  {
    myPID.Compute();                        //Output determined based on Input /using Kp, Ki, & //Kd

//Determine if the status of the relay based on PID Output
    unsigned long now = millis();           //Measure the current time

    if(now - windowStartTime>WindowSize){   //If needed, shift relay window
```

```
        windowStartTime += WindowSize; }
      if(Output > now - windowStartTime) {      //Determine relay status based on output
        digitalWrite(LEDPin,HIGH);                  //Turn on LED indicator
        digitalWrite(RelayPin,LOW); }               //Close relay circuit
      else {
        digitalWrite(LEDPin,LOW);                   //Turn off LED indicator
        digitalWrite(RelayPin,HIGH); }              //Open relay circuit
    //Pass temperature and output calculations
      Serial.print(Input);                      //Pass Input (temp) and Output (PID) to serial bus
      Serial.println(Output);           //for plotting in Processing
     }
     else {
      if (Input > Setpoint){
        digitalWrite(LEDPin,HIGH);              //Turn on LED indicator
        digitalWrite(RelayPin,LOW); }           //Close relay circuit
      else {
        digitalWrite(LEDPin,LOW);               //Turn off LED indicator
        digitalWrite(RelayPin,HIGH); }          //Open relay circuit
      Serial.prinln(Input); }
    }

    float getTemp(int numSamp, int TP, float SR, float ThermN, float TempN, float B)
    {
      uint8_t i;
      float average = 0;
//Read and average the voltage at the input pin, then convert to average thermistor resistance
      for (i=0; i< numSamp; i++) {              //Read in voltages at input pin with a small delay
       samples[i] = analogRead(TP);
       delay(10); }
      for (i=0; i< numSamp; i++) {              //Sum the input voltages
        average += samples[i]; }
      average /= numSamp;                       //Average the input voltages
      average = SR/((1023/average)-1);          //Convert input voltage to thermistor resistance
// Calculating thermistor resistance using voltage divider circuit:
// Vo = output voltage, Rt = thermistor resistance (TBD), Rr = comparitor resistor resistance
// Vcc = input voltage, ADC = analog-digital-converter measurement of voltage, equiv. to //average above
// ADC = Vo * 1023 / Vcc && Vo = (Rt/(Rt + Rr))*Vcc
// Therefore, ADC = ( (Rt/(Rt + Rr))*Vcc ) * 1023 / Vcc
// Which reduces to: ADC = (Rt/(Rt + Rr)*1023
// Solving for Rt: Rt = Rr / ((1023/ADC)-1)

//Convert thermistor resistance to temperature:
//Using simplified Beta parameter approximation of Steinhart-Hart equation:
//Long form: (1/T) = A + Bln(Rt) + C(ln(Rt))^3 --> too many variables
//Simplified: (1/T) = (1/To) + (1/B)*ln(Rt/Ro)
//where To and Ro are temp and res at nominal and T is in Kelvin
      float temperature = 0;
      temperature = (log( average / ThermN ))*(1.0/B) + (1.0 / (TempN + 273.15));
      temperature = (1.0 / temperature) - 273.15;  //Take reciprocal and convert to Celsius
      return temperature;
    }

    void blink(int pin)
    {
     uint8_t j;
     for (j=0; j<5; j++) {
      digitalWrite(pin,HIGH);
      delay(250);
      digitalWrite(pin,LOW);
      delay(250); }
    }
```

A13. Temperature Plotting Code

```
/*This code plots temperature and PID output via serial communication with a
microcontroller. Temperature and Output should be concatenated and passed together
before a single break for interpretation. The temperature setpoint and error margins
should be set in the initialization section. Scaling for presentation is best performed
by setting expected ranges for temperature input and PID output.*/
import processing.serial.*/

Serial myPort;                          //Create an instance of a serial object
int xPos = 1;                           //Initialize tracking variables: current x-position,
//float maxTemp = 0;                    //maximum temperature, and minimum temperature…
//float minTemp = 1000;
float inFloat = 0;                      //input, output, and setpoint values as floats…
float outFloat = 0;
float setFloat = 0;
float setUp = 0;
float setDown = 0;
float tempError = 0;                    //temperature error calculations…
float tempPercentError = 0;
float tempCounter = 0;
float tempAvgError = 0;

//float inputMultiplier = 20;           //Scaling factors can be adjusted to track temp and
//float outputMultiplier = 1;           //output on the same screen, if needed.

float minTempRange = 0;       //Establish expected ranges for input and output for scaling
float maxTempRange = 30;                    //Temp range should extend at least 5% below setpoint to //room temp
float minOutputRange = 0;                   //output range should extend from zero to the window size //used in PID
                                            controller
float maxOutputRange = 500;

float setPoint = 4;                     //Determine temperature setpoint from microcontroller code
float acceptableError = 5;              //Aggregate degC error

void setup() {
 size(1600,800);                        //Open a graphing window
 background(0);                         //Set black background
 println(Serial.list());
 myPort = new Serial(this, Serial.list()[0], 9600);                 //create a serial object to establish //communication
 myPort.bufferUntil('\n'); }            //fill the serial buffer until a break is encountered

void draw()
{ }

void serialEvent (Serial myPort)
{
  fill(0);
  stroke(0,0,0);
  rect(0,0,2000,50);

  String inString = myPort.readStringUntil('\n');       //Read entire serial buffer as a string
  String inputString = inString.substring(0,5);          //Select the temperature value (inputString)
                                                         //from inString
  inputString = trim(inputString);                       //remove extraneous values from input
                                                         //string
//To plot output from PID algorithm, pass output concatenated to temperature
  String outputString = inString.substring(5,inString.length()-1);          //Select the output value from
                                                                             //inString
  outputString = trim(outputString);                    //remove extraneous values from output
                                                         //string
  textSize(28);                                          //Print the current variables in the system for
  fill(255,255,255);                                     //tracking
  text("Set Point: " + setPoint + "C",10,30);
  text("Temperature: " + inputString,250,30);
  text("Output: " + outputString,150,30);

//Tracking Maximum and Minimum Temperature
  /*if (Float.parseFloat(inputString) > maxTemp) {
    maxTemp = Float.parseFloat(inputString);
    text("maxTemp: " + maxTemp,10,150); }
```

```
  else if (Float.parseFloat(inputString) < minTemp) {
    minTemp = Float.parseFloat(inputString);
    text("minTemp: " + minTemp,250,150); }*/

//Calculating Temperature Error
  tempError = abs(Float.parseFloat(inputString) - setPoint);
  tempPercentError = (tempError / setPoint) * 100;
  tempCounter++;    //Running average algorithm for aggregating temperatuer error
  tempAverageError = (tempAverageError*((tempCounter-1)/tempCounter) + tempError/tempCounter;
  text("Temp Error: " + tempError + "C",600,30);
  //text("% Error: " + tempPercentError + "%",1000,30);
  text("Avg Error: " + tempAverageError + "C",1000,30);

//Convert input and output strings into floats and scale for plotting
  inFloat = float(inputString);//*inputMultiplier;          //Uncomment multiplier to use scaling factor
  inFloat = map(inFloat,minTempRange,maxTempRange,0,height);    //Adjust expected range
  //text("iB: " + inFloat,600,30);                          //Track input value for debugging
  outFloat = float(outputString);//*outputMultiplier;     //Only use outFloat if reading in PID output
  outFloat = map(outFloat,minOutputRange,maxOutputRange,0,height);
  //text("oB: " + outFloat,800,30);

//Convert and scale the setpoint and error margins
  setFloat = setPoint;//* inputMultiplier;
  setFloat = map(setFloat,minTempRange,maxTempRange,0,height);
  setUp = map(setPoint + (acceptableError/2)),minTempRange,maxTempRange,0,height);
  setUp = setUp;//*inputMultiplier;
  setDown = map(setPoint - (acceptableError/2)),minTempRange,maxTempRange,0,height);
  setDown = setDownPoint;//*inputMultiplier;

//Plot setpoint, error marging, input (temperature), and output (PID)
   stroke(255,255,0);                                     //set line and fill color
   fill(255);
   line(1,height-setFloat,width,height-setFloat);        //Draw horizontal line at setPoint

   stroke(255,0,0);
   fill(255);
   line(1,height-setUp,width,height-setUp);               //Draw horizontal lines at error margins
   line(1,height-setDown,width,height-setDown);

   stroke(0,0,255);
   fill(255);
   line(xPos,height-(outFloat+1),xPos,height-(outFloat-1));      //Draw 3px vertical line centered on
                                                                  //output value
   stroke(0,200,255);
   fill(255);
   line(xPos,height-(inFloat+1),xPos,height-(inFloat-1));         //Draw 3px vertical line centered on
                                                                  //temperature value
//Reset the graph when the plot has reached full-screen width.
   if (xPos >= width) {
     xPos = 0;
     background(0);
     maxTemp = 0;
     minTemp = 1000; }
   else {
     xPos++; }
  }
```