Closed-Loop Control of a 3D Printer Gantry

Benjamin McKittrick Weiss

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science

University of Washington

2014

Committee:

Duane Storti

Mark Ganter

Brian Fabien

Program Authorized to Offer Degree:

Mechanical Engineering

University of Washington

Abstract

Closed-Loop Control of a 3D Printer Gantry

Benjamin McKittrick Weiss

Chair of the Supervisory Committee:
Associate Professor Duane Storti
Mechanical Engineering

The use of closed-loop control to improve performance in gantry robots is a well-established technology, but adding the necessary sensors and computational hardware to low-cost 3D printer gantries has been generally thought to either be too expensive due to the cost of the hardware or ineffective in offering real practical performance improvements. This thesis develops and evaluates low-cost closed-loop controllers for the X and Y axes of a 3D printer gantry, for some trajectories demonstrating path-following precision improvements greater than 40%. The ability to detect and correct for skipped steps increases reliability and allows for more aggressive tuning of motion parameters; time savings of up to 25% are seen by doubling acceleration rate. The platform developed seeks to increase awareness of the potential for the integration of closed-loop control into existing open source designs.

Dedication

This thesis is dedicated to my Lord and Savior, Jesus Christ, without whom none of this would be possible.

Praise be to the name of God for ever and ever;
wisdom and power are his.
He changes times and seasons;
he deposes kings and raises up others.
He gives wisdom to the wise
and knowledge to the discerning.
He reveals deep and hidden things;
he knows what lies in darkness,
and light dwells with him.
Daniel 2:20-22[1]

[1] Scripture taken from the HOLY BIBLE, NEW INTERNATINOAL VERSION ®. Copyright © 1973, 1978, 1984 by International Bible Society. Used by permission of Zondervan. All rights reserved.

Acknowledgement

The author gratefully acknowledges the support of so many who have made this work possible. Thank you to my wife, for wholeheartedly supporting this endeavor. My thanks go also to Professors Ganter and Storti of the Solheim Lab at the University of Washington, without whose guidance and teaching the author would still be mostly ignorant regarding 3D printing. Thanks also to Matthew D Sorensen, whose insightful ideas, timely advice, and significant contributions to the project were a great blessing.

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1256082.

Table of Contents

1. Introduction

Controlling our environment and the things around us has been one of the core aims and one of the greatest achievements of the human endeavor. Manipulating matter and energy in order to make it behave in a way we deem desirable is at the center of the engineering endeavor, and the rewards of our inventive pursuits have made a profound mark on humankind for hundreds of years.

In the quest to manipulate matter and shape it into objects for our use or pleasure, the last 30 years have seen great growth in a variety of new technologies for manufacturing. Among the most important in potential to transform society because of its incredible flexibility is additive manufacturing, a new group of techniques for making objects layer by layer. In contrast to subtractive machining, where a block of material is cut down to the shape desired by its designer, additive manufacturing allows a minimization of waste and a complexity of form that makes it advantageous for a wide variety of applications [1]. It is loosening the bounds on achievable material forms and in the process creating new markets, new products, and new opportunities for engineers.

Manipulating energy has also seen tremendous growth in the last century, bringing together engineers, scientists, and mathematicians from a variety of fields and creating new disciplines including computation and control theory. In control theory, closed-loop control has been one of the most effective tools developed, being used to stabilize everything from satellite attitude to oven temperature. Closed-loop control theory now includes a vast literature and countless applications.

Due to their low cost, the vast majority of the consumer-level additive manufacturing machines presently produced rely on relatively simple open-loop control schemes that trust the device to go where instructed without actually checking to make sure the motion occurred. Technical and cost barriers have prevented the application of closed-loop control to low-cost additive manufacturing machines (more commonly known as 3D printers). Speed and accuracy improvements are theoretically possible with the application of closed-loop control, as errors can be detected and corrected in real time.

This thesis details the modification and testing of a 3D printer reengineered to use closed-loop control of the X and Y axes of the gantry. Section 2 surveys related works, both in industry and academia, giving context for the system developed here which is designed to fulfill the requirements laid out in Section 3. Section 4 details the modifications involved, including a description of the design decisions made at each stage of the process. The performance of the finished system is reported in Section 5, followed in subsequent sections by comments on lessons learned, future project developments, and final conclusions.

While neither intended nor ready for integration into a commercial product, this platform demonstrates all of the technologies needed for a viable, low-cost, consumer-level printer that uses closed-loop control and paves the way for future developers to further improve system performance with more complex control algorithms. Although the resulting printers will be slightly more expensive, the added investment in sensors and processing power will improve the performance of the machine in speed, accuracy, and reliability. Manipulating matter using 3D printers will become easier and more cost-effective.

2. Background

3D printing technology is a rapidly-evolving field, one which has seen an explosion of interest in the last decade due in great degree to the Maker Movement and the popular RepRap project [2]. This popularity in broader culture has spurred a large group of enthusiasts to begin experimenting with 3D printers, which has in turn created demand for printers in the consumer or pro-sumer markets. Section 2.1 surveys the current spectrum of commercially-available low cost 3D printers, including a discussion of the motion parameters relevant to the results described in Section 5. Subsequently, closed-loop control of stepper motors in the academic literature is briefly treated, as well as the analysis and control of 2D stages driven by stepper motors. Finally, efforts to apply closed-loop control to the 3D printing process are reviewed.

2.1. Commercial 3D Printers

This thesis focuses on low-cost 3D printers, generally with a price tag under 2000 USD. In this market space are well over 100 distinct printer models produced by dozens of companies across the globe. They come in all shapes, sizes, and colors, but with only a few exceptions use very similar motors, motor drivers, and controllers.

With the exception of the Rappy printer, which will be discussed later in this section, all of the roughly 30 printers surveyed used NEMA-17 [3] stepper motors to power their stages. Although information about the motor control electronics is somewhat less available, every indication is that these motors are driven by integrated-circuit motor drivers with peak currents in the range of 1 A per phase, step-direction interfaces to the host controller, and 8X or 16X microstepping capabilities.

In most cases, the processor sending step/direction commands to the motor driver is a microcontroller, which serves as the brains for the entire printer, directing all the motors, heaters, and fans, as well as performing path planning and I/O operations. Generally, one of the Atmel ATMega microcontrollers [4] fills this role. A few 3D printer manufacturers are moving away from this de facto standard towards a network of microcontrollers, either two working together or one for each motor, which are sent move commands from a master controller that might or might not be running real-time code. The 5th Generation MakerBot Replicators, for instance, have adopted this model of networked microcontrollers [5].

Entry-level 3D printer manufacturers compete in the marketplace on three performance criteria: layer height (Z), X-Y precision, and speed. Layer height is driven by the diameter of the print nozzle and has little relevance to the present discussion. X-Y precision tries to quantify the smallest steps the stage can take in the X and Y axes, and is primarily a function of the microstepping setting on the motor (generally 8X or 16X) and the mechanical reduction in the drive train. Reported values are generally in the range of 15-50 µm (0.0006 in to 0.002in), and seem to represent the specified capabilities of the motor, driver and stage and not real-world measurements. Theoretically, a 200 step/rev motor with 16X microstepping driving a 0.5 in pulley can drive a belt in steps of 12.4 µm, but motor step error, belt slop, and the dynamics of the support structure frequently introduce large error terms on this quantity in real life, sometimes increasing it by an order of magnitude or more.

Advertised speed is of somewhat more importance in this thesis. Printer manufacturers frequently report “travel speed”, which refers to the maximum rate of motion of the X and Y motors, and “print speed”, which describes the speed at which printing can take place. Since printing requires contour-following capabilities and is additionally limited by the extruder motor torque, print speed is generally lower than travel speed. Print speeds on the high end run around 150-200 mm/s [6], [7], with one printer (one of the two that uses closed-loop control) advertising print speed of 300 mm/s [8]. By comparison, the 4-year-old MakerBot Replicator I (on which the printer used for this project is based) has a stock print speed of 90 mm/s and a travel speed of 150 mm/s, as described by its slicing and control software, MakerWare [9]. The fastest printers all use techniques to reduce the amount of mass being moved around the build space, increasing the acceleration achievable with a given motor torque.

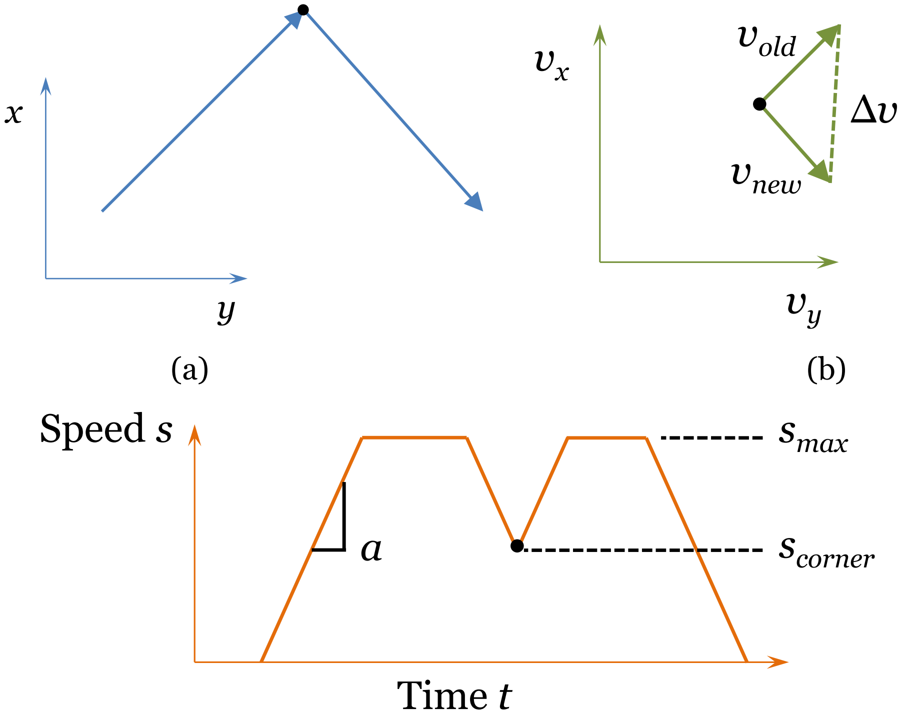
Only two printers surveyed deviate from the open-loop stepper motor approach. The Rappy printer, by Stellamove, Inc [10], uses optical rotary encoders attached to DC motors with gear boxes to control the belt-driven parallel gantry. Marketed for increased reliability and resistance to shock, the use of brushed DC motors simplifies the motor driver electronics significantly and may reduce cost as well, at the expense of processing time spent reading the controllers and computing control laws. The Chinese Panowin F3CL printer [8] uses stepper motors with closed-loop motion control to reduce sensitivity to mechanical disturbance and improve build quality. The company has a video [11] correcting for external disturbances on the X and Y axes that demonstrates closed-loop control capabilities, but details on the sensors used or control scheme employed are not available in the company's documentation. The printer also has a significantly higher price tag, reported to be almost 4000 USD.

Taken together, the low-end 3D printing market is filled with printers with similar characteristics. Almost all of them utilize an open-loop stepper motor and belt stage design, which is low-cost, simple to design and implement, and at least theoretically has precision sufficient for most needs. The disadvantages of these designs include the inability to compensate for belt slop, motor inaccuracies, and other system dynamics, as well as the possibility of skipped steps, which could cause prints of long duration to fail. Most printer manufacturers report only travel speed and print speed, but this single number does not fully capture the dynamic capabilities of the printer, nor directly predict to the time needed to build models.

2.2. Motion Parameters

The actual time required to perform a print is a function not just of the maximum speed of the printer but also of the rate of acceleration the X, Y, and Extruder (E) axes are capable of, as well as the degree to which the printer must slow down around corners. Since these quantities will be important in the developments of this thesis later on, a more detailed explanation of their definition is given here.

The behavior is similar to that experienced when driving a car. Just because the speed limit is 25 mph does not mean all cars can reach their destination in the same time. Some will have better cornering, so they don't have to decelerate as much when turning, and others will have better acceleration, enabling them to return to the posted speed more quickly.



(a) (b)

(c)

Figure 1. Anatomy of a Corner: (a) Movement path (b) Velocity vectors at the corner (c) Speed profile

Figure 1 illustrates these concepts, as implemented in the RAMPS firmware [12], using a simple two-move motion profile. Initially, the robot starts at rest. The motors accelerate at a finite, constant[2], firmware- or user- specified velocity a until either the "cruising speed" s_{max} is reached or the robot must start decelerating for a corner. Corner velocity is selected so as to ensure the magnitude of the velocity change (magnitude of the vector difference between the command velocity just before and just after the corner) does not exceed the specified speed change limit, see Figure 1 (c).

The MakerBot-endorsed open-source firmware for the Replicator I, Cupcake, and Thing-O-Matic printers, called Sailfish, includes default acceleration and speed change settings [13]. The Marlin firmware, which is used on a variety of printer designs, also specifies defaults. Anyone building their own printer and using Marlin eventually winds up tweaking these settings, but the values used by these two developers tend to be good starting points and representative of the broader spectrum of printers. Acceleration specified by Sailfish is 1000 mm/s^2, while Marlin uses a much higher 3000 mm/s^2. Sailfish's speed change default is 15 mm/s (though the documentation suggests "draft quality" prints can support speed change values up to 40 mm/s), and Marlin's is about the same at 20 mm/s.

Generally, these parameters are tuned by manufacturers or users to minimize print time and avoid skipped steps while maintaining smooth extrusion and finished part surface quality. Because the motors are run open-loop, any steps skipped are undetectable and therefore unrecoverable, resulting in the entire print shifting in the X or Y direction from that layer onwards.

[2] Marlin and Sailfish both support varying acceleration and include code that could dynamically adjust the acceleration rate so that diagonal moves accelerate faster (both X and Y motors can share the torque load when accelerating diagonally), but the way the code is written this dynamic acceleration is only available if $a_{axis} < a_{max} < \sqrt{2}a_{axis}$, where a_{axis} is the axis maximum acceleration and a_{max} is the system maximum acceleration. This kind of configuration is suggested neither in the documentation nor in Sailfish or Marlin firmware defaults. The accelerations reported in this section are $a = \min(a_{axis}, a_{max})$.

2.3. Closed-Loop Stepper Motor Control

Closed-loop servomechanisms of the kind developed in this thesis are not a new study, having been present in the literature for at least 60 years in a broad range of manufacturing and engineering applications [14]. This section presents a brief survey of this vast body of literature as it applies to the present project.

Although stepper motors are designed primarily to run open-loop, significant performance improvements in speed, acceleration, torque, and reliability can be obtained using a closed-loop controller [15]. This occurs mostly because open-loop operation requires margins to ensure behavior stays within the local equilibrium positions at each step. Closed-loop control allows these margins to be reduced or eliminated, with a proportional improvement in performance. Since stepper motors involve nonlinear electromagnetic relationships (see [16] for an excellent, approachable stepper motor model derivation), their control is a nonlinear control problem that has received much attention in the academic literature. Control techniques such as feedback linearization ([17], [18]), sliding mode control [19], robust control [20], and backstepping control [21] have been successfully applied to this problem, with control inputs to the system chosen either as motor voltage or motor current.

The control problem requires measurement or estimation of system states consisting of the current in each phase of the motor, the rotor position, and the rotor velocity. Nonlinear observers can be used to estimate states not directly measured, but generally the more states estimated the more motor parameters need to be known up-front. At least some of the motor parameters are frequently estimated using online least squares adaptation algorithms, allowing small manufacturing variations and motor wear to be compensated for in real time. In all these configurations, position and speed measurements are made directly at the rotor, and Bodson, et Al point out that the granularity of the encoder is a major source of noise [17]. Industry has also recognized the value in utilizing closed-loop control of stepper motors, but frequently relies on simpler models and control algorithms that can be run on lower-cost processors, including the use of PID and lead-lag compensating controllers, sometimes in specialized integrated circuits, such as the one described in [22].

2.4. Control of Stepper Motor Driven X-Y Stages

Using stepper motors to control X-Y stages is a common practice in the industry and research communities, and efforts to control and synchronize this larger system are also prevalent. Generally, these systems consist of a gantry arrangement, where two Y motors drive parallel linear slides, between which rests a third motor and slide, the X axis, connecting the Y axis to the load. Because the Y axis has two motors, it is possible to control θ to a small degree (normally the control target for θ is 0 to avoid binding). Extensive research on modelling and controlling these axes together is available, including cross-coupling terms, referred to as “contouring control” or “coordinated control” (see, for example, [23] and [24]).

The CNC industry has seen extensive use of servo motors and stepper motors in NC machines of all types, running in open or closed-loop mode. There is a vast literature describing a myriad of controllers and methods to reduce contouring error in coupled axes, primarily in the case of XYZ motion for contour machining. Ramesh, et Al. gives a lengthy overview of some of these techniques [25].

The IcePAP project, [26], utilizes multi-axis motor control for positioning of various mechanisms in scientific research labs. Its platform is roughly similar to that utilized for the current work, albeit driving much larger stepper motors and at a much higher cost per axis (500 Euro). It utilizes a network of motor controllers, one for each axis, which can be configured and synchronized using a master node, allowing systems with many degrees of freedom to be controlled synchronously. Closed-loop control of stepper motors is implemented using encoder feedback only.

2.5. Closed-Loop 3D Printing

Specifically applied to 3D printing, the author is aware of no scholarly work that has addressed the use of closed-loop stepper motor driven stage designs, although there has been some discussion and preliminary development among the enthusiast communities (see [27] and [28]). These discussions indicate interest in the technology, but unwillingness to invest until its benefits are demonstrated or costs fall.

Controlling the stage, however, only addresses one level of the quality of the parts resulting from the additive manufacturing process. Several efforts have sought to define controllers based not just on instantaneous X and Y position, but on the actual characteristics of the previous layer of deposited material [29], or on measurements of the global or local deviation of the part from the design file [30]. Although the measurement and computation requirements of such approaches are significantly greater, they possess the greatest ability in improving part quality.

Widely used to great benefit in other areas of industry, closed-loop motor control could allow for similar advantages in the control of low-cost fused filament fabrication machines. While the industry has experimented with a variety of mechanical frameworks, the selection of hybrid stepper motors and single-chip microcontrollers driving them in open-loop mode remains nearly universal. An opportunity exists, then, to merge the knowledge of other application domains with the low-cost 3D printing market, improving performance without undue increase in cost.

3. Problem Statement

This research seeks to explore this opportunity to improve printer performance in the low-cost market sector, in which printers are widely used by consumers and increasingly by groups in developing nations where the cost of professional-grade printers are prohibitive. A large collaborative community of enthusiasts and engineers in the open source and Maker Movements also increase the likelihood that the project will be further developed and increase its potential for impact. Because the goal is to produce a product that can be integrated into a 1000 USD printer, each axis should not cost more than 30 USD to convert to closed-loop control.

The goal of this project was to install sensors to detect the actual motion along the X and Y axes of the gantry and to use this information as the input for a closed-loop controller that can modify the commanded position of the stepper motors to compensate for detected errors. It was believed by the author that such a system would improve the accuracy of the printed parts by correcting for the mechanical dynamics in the belt drive system and error in the motor step size, as well as catching and correcting missed steps, which enables the stepper motors to be tuned more aggressively and could increase the speed of the print process. In this way, it goes beyond the closed-loop control of the Rappy 3D Printer, which only controls motor angle but does not close the loop to actual carriage position.

4. System Developed

Building upon the MBot Cube printer and the Intelligent Motor Control hardware platform (IMC; [31]), this project involved creating a modified IMC Axis Controller capable of closed-loop control, then implementing this controller on the X and Y axes of an MBot Cube printer. A closed-loop controller requires a hardware sensor that gives feedback to the system. This sensor must be mechanically attached to the robot, and electrically connected to the IMC axis controller board. The IMC axis controller firmware needs to be extended in order to allow closed-loop, real-time control, and a control law must be developed to drive the system. The following subsections detail the components of the system developed.

4.1. Existing Platform

This project was built around a first-generation MBot CUBE printer [32]. Priced around 1000 USD on its initial release in 2013, the printer is a modified version of the MakerBot Replicator I [33] with a larger build volume. In this design, the build plate is moved vertically using a lead screw and a plywood frame supports an X-Y gantry-style carriage that moves the print head over the build plate.

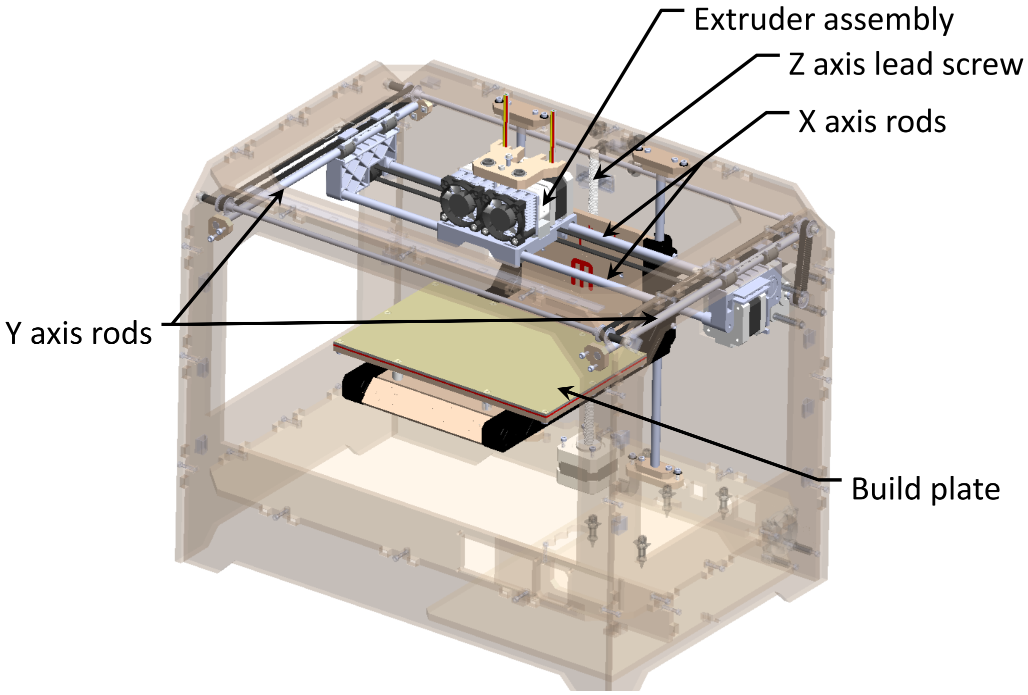


Figure 2. 3D printer high-level view. MakerBot CAD from [33][3]

[3] The MBot CUBE printer is based upon the MakerBot Replicator I design, for which CAD models are available in [33]. For the purposes of this project, the minor differences between the two printers enable the use of the MakerBot CAD models for mechanical design and illustration purposes.

As Figure 2 shows, the X-Y gantry carriage consists of two stages. The Y axis stage sits upon two rods, on the left and right sides of the robot, using linear bearings to allow free motion along the rods' axis. A stepper motor drives a shaft that is coupled to belts on each side of the Y axis, allowing it to move while keeping the two sets of linear bearings from binding. The X axis stage sits between the two parts of the Y axis carriage and consists of another pair of parallel rods, on which the print head assembly rides using linear bearings. Another stepper motor and belt drive assembly moves it along the X axis.

In order to handle the increased computational demands of a closed-loop controller on each axis, this project built on the Intelligent Motor Control printer control framework developed previously by Sorensen and Weiss [31]. The IMC framework uses a separate ARM-based processor to control each axis, creating the needed computational horsepower to perform the larger, more complex real-time computations used in closed-loop control (see Figure 3).

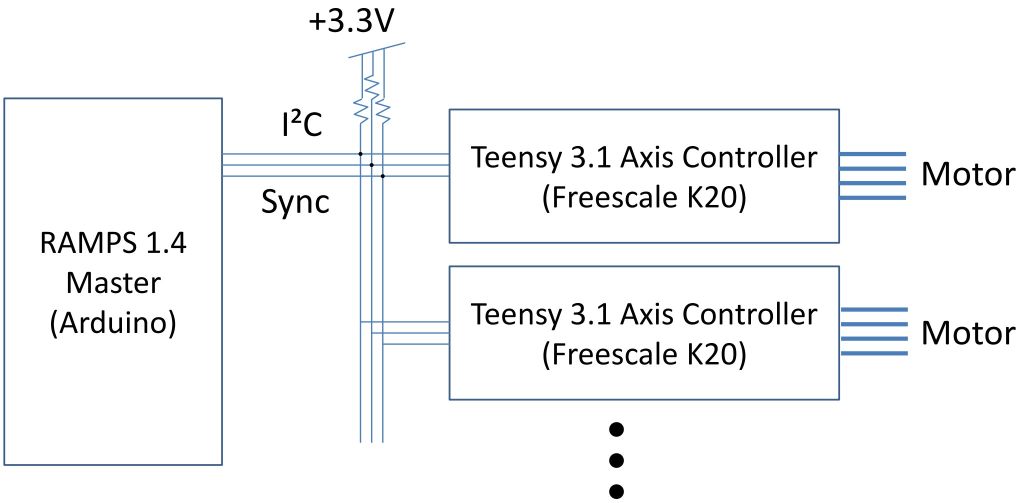


Figure 3. Intelligent Motor Controller Block Diagram. Modified from [31]

Using a single processor for path planning, driving stepper motors, USB and/or SD card communication, and thermal management creates bottlenecks. In addition to faster processors on each axis, splitting the work between processors allows each axis controller to do a better job of regulating the real-time aspects of the task, since fewer interrupts need to be serviced simultaneously. The result is more consistent pulses sent to the stepper driver, which increases the achievable speed of the system before motor resonance becomes a problem. In addition, this higher real-time accuracy helps ensure the closed-loop controller is updated regularly, making discrete-time models of the system accurate.

Beyond performance considerations, IMC is an effective electronics platform for this work because it is built around up-to-date firmware (the master runs a modified recent version of the Marlin firmware), which enables most open-source slicing engines to produce G-Code that will run on the new system. Since all the axis controllers communicate with the master using a well-defined packet interface, it is straightforward to control only one or two axes in closed-loop, leaving others to run in open-loop mode, depending on the performance and cost goals of the user. Finally, the cost of the Freescale K20DX256 microcontroller (about 5 USD in quantity 100) makes them an economical high-performance processor with good motion control capabilities. For this project, the K20 chips used were packaged on the 20 USD PJRC Teensy 3.1 development platform [34], but they could be converted to a surface-mount custom board, including the motor driver, for about 20 USD/axis.

4.2. Sensor Selection

Measuring the position of the print head relative to the gantry is the first step in producing a closed-loop control solution. Using a rotary encoder on the motor would detect the motor position, but not the effects of belt stretch in the stage itself. Because capturing and controlling for the dynamics of the belt drive is an important goal of the project, a rotary encoder connected to the motor itself is not sufficient. Any linear encoder used has to have a resolution at least as good as that of the stepper motor. The stepper motors used in the test system (and widely in the low-end 3D printing market) produce a linear step distance of 0.00049 in, or 12.4 µm. Measurements on the robot gantry using the developed encoder system give an actual step size of 10.61 µm for X and 10.60 µm for Y, likely caused by the effective bend radius of the belt being slightly smaller than the pulley's nominal radius.

The selection of a linear encoder is further restricted by cost. Each IMC axis hardware (motor driver, MCU, and PCB) costs roughly 20 USD, so an encoder should cost about 10 USD to stay under 30 USD. The encoders need to be able to measure over a distance of about 15 in. Finally, it is desirable to have the selected encoder measure with a resolution of at least the step size (~10 µm), ensuring that even the smallest actuator motion can be detected.

Linear encoder sensors are built around a variety of technologies. Generally a strip, called the scale, is manufactured with carefully-controlled, periodic (for an incremental encoder) or unique (for an absolute encoder) patterns that can be detected by the sensor. The sensor usually detects differences in the optical, magnetic, capacitive, or inductive characteristics of the scale and converts them into a measurement. While professional encoders in all of these categories can achieve resolutions on the order of 1 nm, low-cost linear encoders have significantly lower precision.

Low-cost optical encoders are predominantly produced by Agavo and used in inkjet printers. These come in resolutions 17.6 µm/tic, and can be purchased (in quantity 100) for about 10 USD/each. Companion scales are difficult to find, but also cost about 10 USD per axis. Unfortunately, 17.6 µm/tic is greater than the 10 µm axis step size, making control more difficult, especially since encoder granularity is was cited as a major source of noise in [17]. Capacitive linear encoders are available, primarily in the form of digital calipers, but resolution is limited to 0.0005 in. (12.7 µm), and total strip length is generally 8 in. or less (or the caliper becomes too expensive).

Magnetic encoders are also produced; Austrian Micro Systems (AMS) as well as several other manufacturers sell scales with precise magnetic dipoles every 2 mm. AMS produces a sensor (the AS5311, [35]) that interpolates the magnetic field of the dipole and creates a linear encoder with resolution of 0.49 µm/tic, which can be purchased for roughly 7 USD in quantity 100. Companion scales can be purchased in 30mm lengths from AMS for 6.75 USD (quantity 1) [36], or in larger quantity or length from other manufacturers at additional cost. The AS5311 chip requires a small carrier board, which can be either specially-designed for this application or a generic TSSOP breakout board, either available for about 5 USD. Although the AMS solution, at 18.75 USD, is slightly more expensive than the target price point, its performance is more than an order of magnitude better than any other low-cost encoders. As a result, it was selected as the sensor for use in the system.

Because the resolution produced by the sensor is much higher than any measurement devices available during development, only limited validation of the sensor performance was possible. A hysteresis sweep and repeatability measurement validated with a caliper was performed, but the error produced is more likely due to the measurement instrument and setup than the encoder being evaluated.

4.3. Mechanical Design

The sensor itself needs to be held in close proximity to the magnetic scale as each axis moves. In order to accomplish this, an assembly of 3D printed parts was designed and built to attach and position the scale to the fixed part of each of the X and Y axes and to attach and hold the sensor chip on its carrier board in the correct position relative to the scale to ensure an accurate reading.

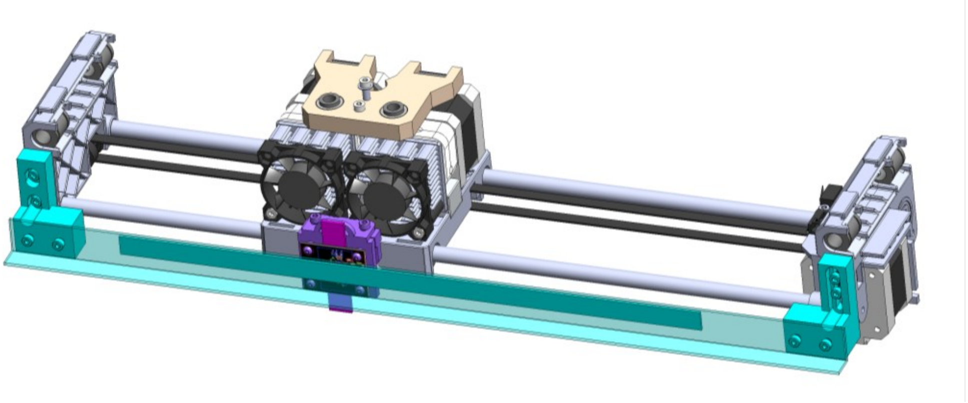


Figure 4. X Axis Sensor Mount Design. Scale support strip shown semitransparently in foreground (teal); sensor mounted to carriage (purple).

The design developed utilizes 3D printed parts for most of the components and bolts onto the frame and plastic carriages in various places. The magnetic scales are attached to stiff aluminum plates, which are attached to "fixed" structure for each axis. For the X axis, the scale is attached to the Y axis carriage and the sensor mounted on the extruder assembly, see Figure 4; for the Y axis, the scale is attached to the printer frame and the sensor is carried by the Y axis carriage, as shown in Figure 5. Adjustment slots built into the design allow the sensor to be tightly positioned with respect to the scale, keeping the two close enough to ensure accurate readings.

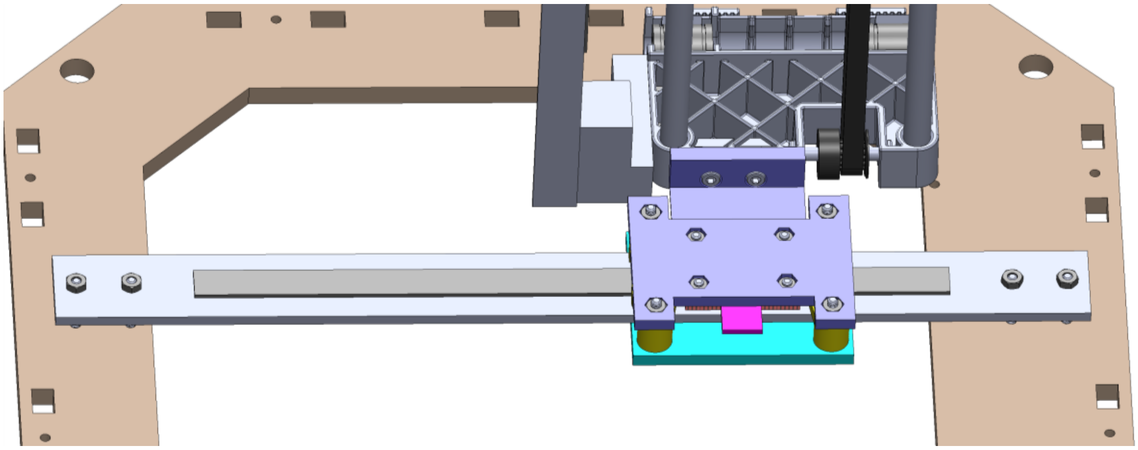


Figure 5. Y Axis Sensor Mount Design. View from under the carriage, inside printer. Sensor is supported by the assembly attached to the carriage, which moves along the scale, fixed to the frame.

When the X axis moves, the dynamic forces exerted on the Y axis carriage which supports it causes the distance between the Y axis scale and sensor to vary significantly. In tests, these vibrations consistently moved the sensor too far away from the scale and caused the encoder to lose track of position. To overcome this, the Y axis sensor mount was redesigned to utilize a flexure which allows the Y axis rails to move laterally without pushing the sensor out of alignment. After installation, no further sensor errors caused by misalignment have been observed in normal operation. Assembly diagrams and selected drawings for the mechanical system can be found in Appendix A.

4.4. Electronics

The electrical design for the closed-loop controller developed is patterned closely after the one used in the development of the first IMC platform [31]. Each controller consists of a Teensy 3.1 development board [34], connected to a Pololu-style stepper driver [37] via a step/direction two-wire interface. A modified I^{2}C interface connects the IMC axis to the master controller. End stop configuration also follows the stock IMC implementation. In addition, the Teensy is attached to the AS5311 encoder using either hardware-decoded A-B quadrature or a subset of the SPI communications protocol. SPI was used for this project due to higher resolution and lower latency on that interface. The interface runs at 1 MHz, and although no testing to failure was performed, the communications channel exhibited a negligible error rate when routed over 18 in. of twisted-pair wire, twisting each signal as well as the chip power with a separate ground. Because the project itself was just a prototype, the axis controller was implemented on a solderless breadboard.

The AS5311 chip resides on a carrier board which breaks out its communication pins and includes a few debasing capacitors. This carrier board can be a generic breakout board for TSSOP packages with a few components wired on. Alternately, Matthew Sorensen has developed a 2-layer commercially-fabricated board that does the breakout, places the passives, and includes pads for LEDs to indicate magnetic field strength [38].

4.5. Software Design

The software designed for the system primarily involved extensive modifications to the original code from the IMC project. New code was implemented to read the sensor value, compute target position and velocity at any given time based on RAMPS move data, and update the controller at precisely timed intervals. In addition, a USB communication interface for the controller was developed to aid in tuning and configuration of the control algorithm.

One of the key aspects of the code developed is careful use and prioritization of interrupts. Interrupts are used throughout both the original IMC codebase and the newly developed closed-loop controller for USB communications, timing of pins on the motor and I^{2}C/sync communications interfaces, and limit switch trip detection. Using so many interrupts requires special care to ensure that interrupts which are time-critical (IMC sync line and controller update timer) are serviced at a higher priority than low-importance ones (I^{2}C or USB communications, for instance). Compared with the constrained interrupt prioritization used on the AVR microcontrollers, the ARM Cortex M4 core used by the Teensy is very flexible, allowing interrupts to be arbitrarily grouped into 16 priority levels which can supersede one another mid-interrupt [39].

The time required to read the encoder, calculate the position and velocity targets for the controller, and compute the system inputs from the control law put an upper bound on the update speed of the controller. Using any of the control algorithms described in the next section, the control update takes less than 0.15 ms. This allows the controller to update at frequencies as high as 1 kHz without the update time itself producing undue jitter.

In practice, the system dynamics observed exhibit characteristic frequencies in the range of 100 Hz. In order to fully capture the system behavior, a controller update frequency of 1 kHz was selected, 10 times greater than the system dynamics, while keeping the time to perform each controller update to around 10% of the update period. The processing power of the Teensy's 32-bit ARM core makes this update frequency possible.

The sensor readings, target location and velocity, and control decisions are all recorded when the controller is running, and can be streamed back to a connected host computer over USB. A graphical user interface (GUI) was developed in Python to display these quantities, as well as adjust parameters and execute actions locally on the device without using the IMC network. This mode of operation was used extensively in testing and development of the controller, and allows the controller to follow arbitrary trajectories, step inputs, and noise functions using each of the control laws. For more details, see Appendix C.

4.6. Control Design

The development of a suitable controller forms the final step in this project. In order to do this, a PID controller was used to help guide the structure of the control flow and debug the control module. After that more complex controllers were explored and evaluated. The development of a suitable control law is by no means complete with this project; rather, this project has put in place the framework needed for others to fine-tune the controller and improve system performance still more.

At its most basic, the system and controller are assumed to take the form shown in Figure 6. The command input, $\hat{x}$, dictates the target position, in encoder tics. The framework developed allows for the command velocity to be an input as well, but none of the controllers explored here actually uses this capability. The controller, C, takes the command input and the system output as reported by the encoder, x_e, and produces a control input. A separate, feedforward path takes the command input itself through an optional filter F and adds it to the output of the error-driven controller to produce the control input, $\tilde{x}$. This quantity is subtracted from current motor position, as reported by the stepper control code, to produce $\Delta\tilde{x}$, the distance that needs to be traveled between now and the next update. This quantity is clamped, in order to keep the motor from reaching resonance, and then passed to the stepper control software, which triggers step/direction pulses at appropriate intervals between control updates. These pulses are transformed into current by the motor driver chip, which turn the motor. The encoder detects the resulting motion and passes it back for use in the error signal sent to the controller.

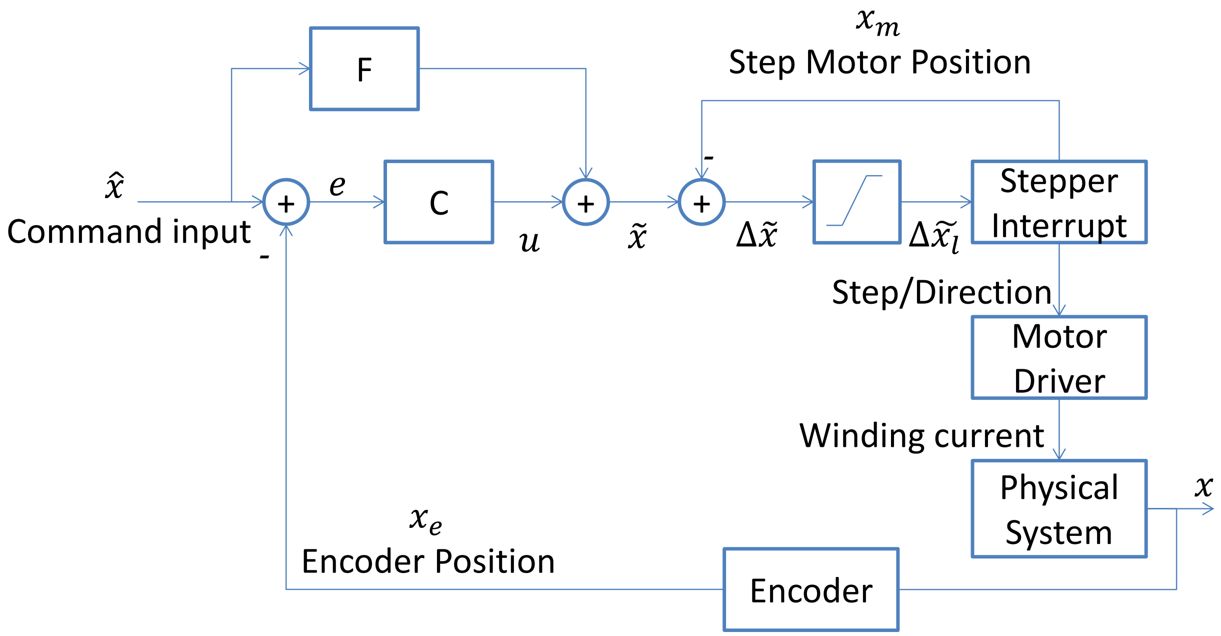


Figure 6. Controller Block Diagram

An open-loop controller that behaves almost identically to the RAMPS code can be created by setting $C = 0$ and $F = 1$, causing the control target to exactly equal the command input. For the PID controller implemented here, block C is used for the controller, while F is set to 1. In this way, the PID controller responds only to error, while the feedforward path handles the bulk of the motion. This is necessary because when the command input matches the encoder position (zero error state), the output of a PID controller is zero, while the input to the system must be the nonzero current location or motion will occur. An alternative solution would be to eliminate the *F* block and introduce an integrator after *C*.

Most commercial implementations of closed-loop control, especially in low-cost environments, utilize PI or PID control. One advantage of the Teensy's powerful processing core is that it enables the development of more computationally-intense control algorithms. In addition to implementing PID control, this project explored using three more complex controllers based on a system model: model following control, model-based control, and a compensating filter. For a detailed development of a dynamic model for the system, see Appendix D.

Model following control, implemented using the algorithms described in [40], uses input/output data from the system to create a linear model of the dynamics. The system inputs are calculated so that the output of the model follows exactly that of the control trajectory. If the model is accurate, the actual system output should also precisely follow the control trajectory.

In model-based control, the controller is designed so that the system follows the behavior of a template system model's outputs when presented with the command inputs [41]. This way, abrupt discontinuities can be included in the command trajectory without causing infinite velocity spikes in the system input signal.

Compensating filters are the broader class of filters to which PID belongs, and allow the definition of nearly arbitrary finite- or infinite- impulse response filters between the command input, error signal, and system input. Several compensating filters were explored using the Matlab Control Systems Toolbox.

Unfortunately, none of these more complex controllers were implemented successfully. It is unknown whether code errors, controller instability, system nonlinearities, modelling errors, or other causes resulted in this failure. As a result, the following section reports only the results using a PI control law, which is implemented as follows:

$$u[k] = K_P e[k] + K_I \sum_{m=1}^{k} e[m](t[m] - t[m-1])$$

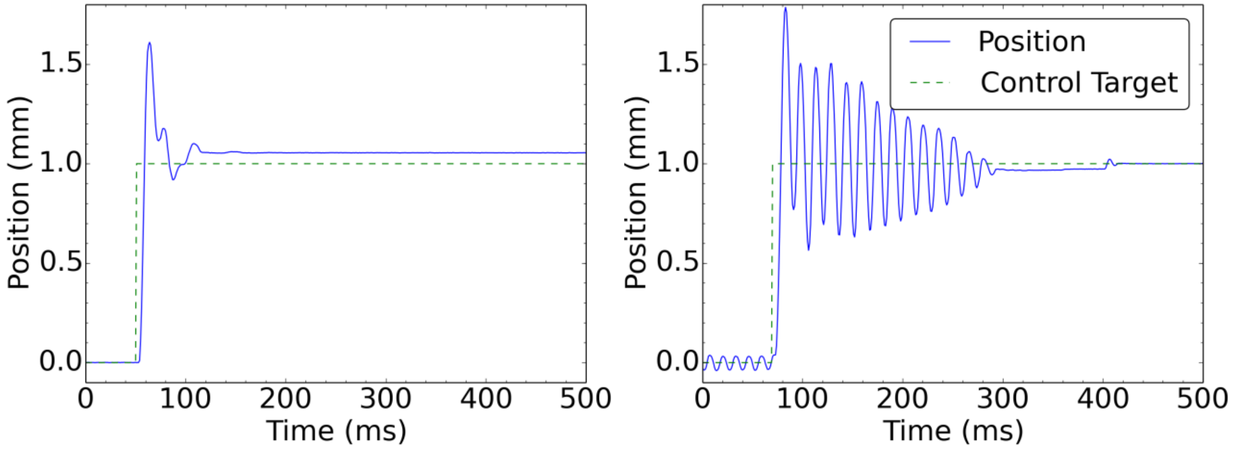
where K_P and K_I are the proportional and integral constants, $u[k]$ is the control output at time k, and $e[k]$ is the error signal. The derivative feedback term was not utilized because of large amounts of noise in the velocity calculation.

5. Results and Discussion

In order to evaluate the effectiveness of the PI controller in improving system performance, tests were run using the IMC platform, both with and without the controller enabled. Evaluation using the encoders has verified that the IMC platform without the controller enabled performs comparably with an unmodified Marlin/RAMPS setup, except for a delay of approximately 2 ms between moves, believed to be attributed to the delay caused by waiting for the IMC nodes to synchronize. The control tests shown below were performed with a PI controller designed experimentally for precise following of an input trajectory. Since the emphasis of this work was not on the controller tuning, the results displayed below should be improved substantially with more careful attention to these parameters, or the use of more complex controllers. This section summarizes some of the most interesting results obtained, beginning with plots of the approximate step response with and without the PI controller, trajectory following performance, and performance improvements under high acceleration and high speed change conditions.

5.1. Step Response

While step response performance is not the ultimate goal of the selected controller, it does provide insights into the effect the control law has on system performance. Step responses reported here are different from a traditional step response because of the saturation of the controller, meaning that the step occurs at maximum velocity over several update cycles instead of being truly instantaneous. Additionally, because actual printing involves a series of motions with trapezoidal velocity profile, the step response itself is never seen in the real world.



(a) (b)

Figure 7. X axis step response. (a) Open-loop (b) PI control (P=0.5, I=20)

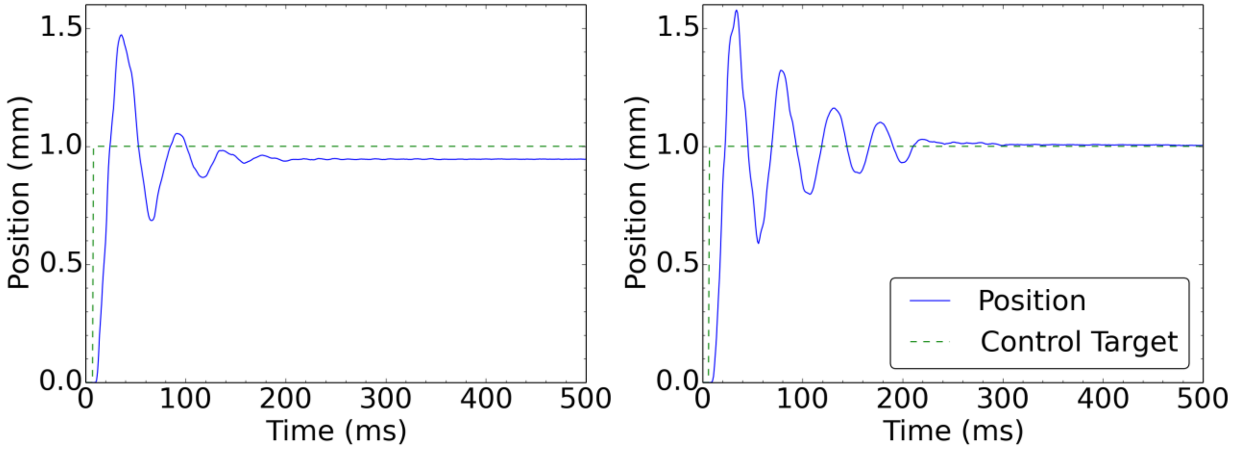


Figure 8. Y axis step response. (a) Open-loop (b) PI control (P=0.5, I=20)

The step responses shown in Figure 7 and Figure 8 reveal a system whose uncontrolled dynamics are underdamped with significant overshoot. It is important to note that this system is not linear; while the overshoot increases with increasing step size, it is not a linear increase. A step of 1.0 mm on the X axis produces an overshoot of roughly 0.5 mm (50%), while a step of 54 mm produces an overshoot of only about 1.0 mm (1.9%). The Y axis has similar performance characteristics.

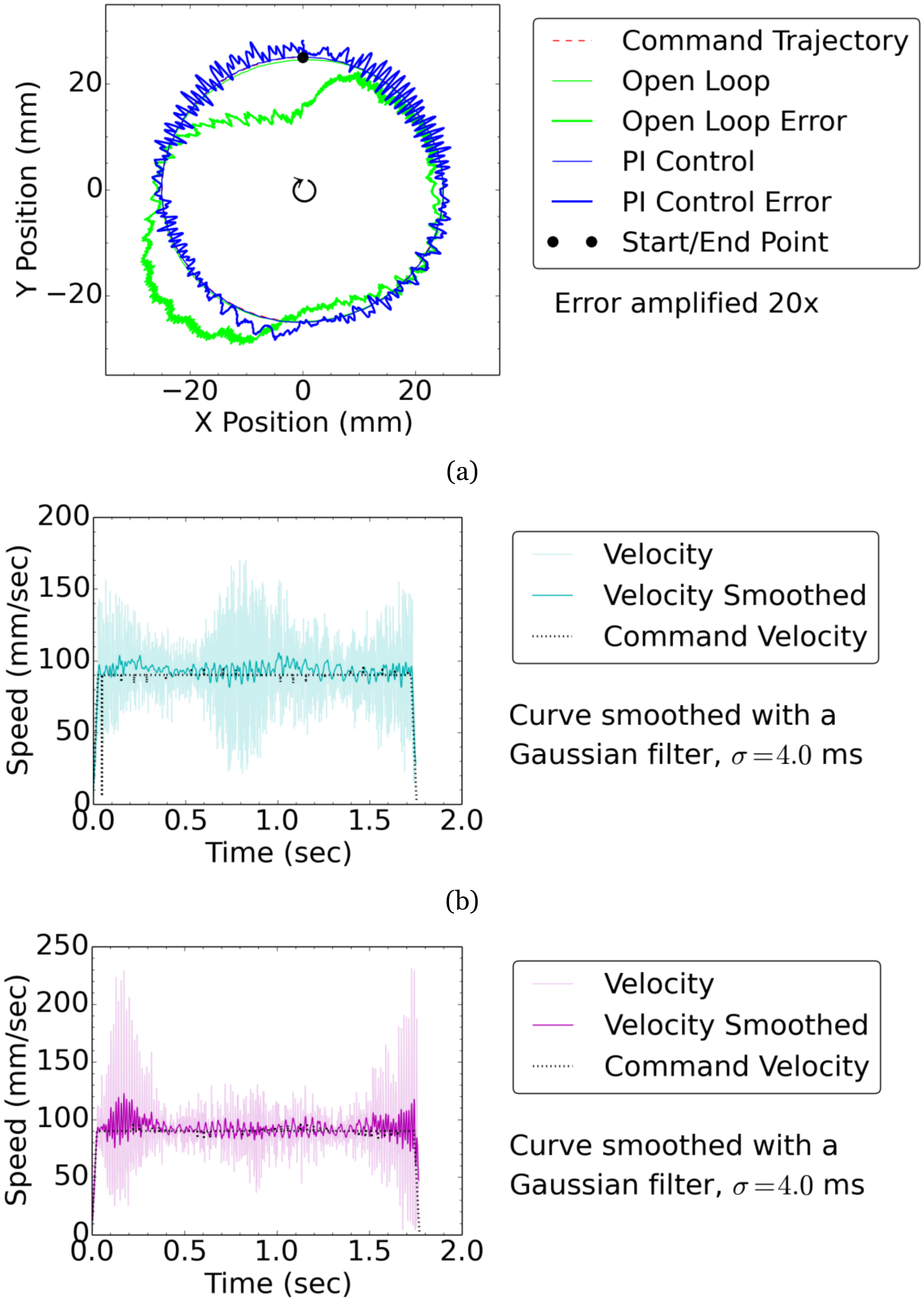
Under PI control, the X axis performance seemingly becomes worse. Not only does the overshoot increase, but the damping factor is reduced. This PI controller (with P = 0.5 and I = 20) was tuned for smooth tracking of a trajectories under trapezoidal velocity profiles, however, and not for step performance, and the integrator ensures that the position error is eliminated in about 400 ms.

5.2. Trajectory Following

Step response provides a characterization of the controller in a single axis manner. A better test of performance involves measuring the system's ability to follow a series of move commands. Formulated as G-Code, these commands replicate motions found in a real-world print. The first, found in Figure 9, shows the system's performance in following a circular trajectory composed of 50 linear segments, both in open-loop and closed-loop (PI control, P = 0.5, I = 20) modes, run at a nominal speed of 90 mm/s.

Figure 9 (a) shows representative result curves from both closed-loop and open-loop following of the circular trajectory, with error magnified radially 20-fold to better illustrate the differences in performance. It is visually obvious that the closed-loop control reduces the amount of radial error at each datapoint, which we can numerically quantify as,

$$e_r = \sqrt{\frac{1}{N}\sum_{i=0}^{N}\left(\sqrt{(x(i)-x_c)^2+(y(i)-y_c)^2}-r_c\right)^2}$$



(a)

(b)

(c)

Figure 9. Open-loop vs. PI closed-loop circle tracking. (a) Tracking error (amplified radially 20X) (b) Open-loop velocity profile (c) Closed-loop velocity profile.

where N is the number of datapoints, x_c and y_c is the center of the circle trajectory, r_c is the radius of the circle trajectory. The open-loop trace in the above figure has a radial mean squared error of 0.24 mm, whereas the PI controller performs with an error of 0.07 mm.

An alternative way to compute error is to find the distance between each datapoint and the command trajectory location at that point, then average these quantities. This captures both radial error from the target curve and temporal error (lead/lag) along the curve, and has the added advantage of being applicable to non-circular geometries. Defining the error in this way gives the expression for "mean trajectory error", e_t:

$$e_t = \frac{1}{N}\sum_{i=0}^{N}\sqrt{\left(x(i) - x_t(i)\right)^2 + \left(y(i) - y_t(i)\right)^2}.$$

The mean trajectory error for the open-loop control is 0.40 mm, with 0.22 mm standard deviation. Under PI control, this error is reduced to 0.09 mm, with 0.05 mm standard deviation, a reduction in error of more than 75%. Because of the way they work, the stepper motors will invariably track the commanded position to within one full step (equivalent to a linear movement of about 0.2 mm). Therefore, the error beyond this in the open loop case cannot have been caused by motor position error alone, and must result from gantry dynamics. The ability to correct for gantry dynamics makes this kind of closed-loop approach significantly better than the Rappy 3D printer, which can only control motor position.

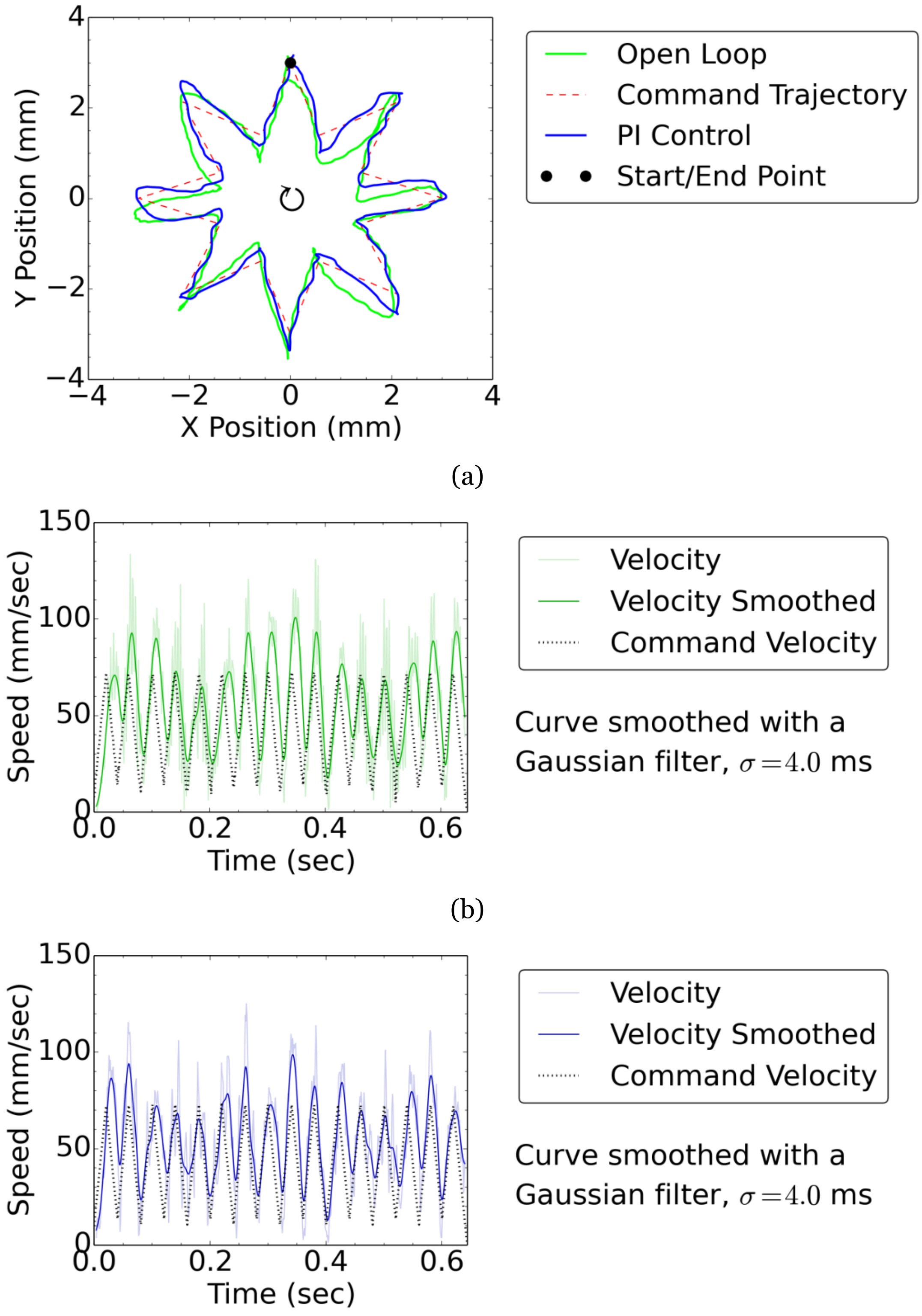
Figure 9 (b) and (c) plot the velocity magnitude at each time step for open-loop and closed-loop controllers, respectively. The velocity is computed by convolving the position information by a 9-wide Stirling Formula derivative kernel, in an effort to reduce the noise produced by the quantization of the encoder [42]. Additional Gaussian filtering is performed to produce the "smoothed" traces. Both open and closed-loop control have significant oscillations in the velocity profile over the flat part of the commanded velocity.

Of special note is the increased magnitude of the oscillations in the closed-loop case, a consequence of the proportional term in the PI controller as it tries to more closely match the command trajectory. This behavior is not unexpected after the step response shown in the previous subsection. The practical result is an increase in noise while the printer is operating and possibly a rougher surface finish.

A smaller, more intricate path is tested in Figure 10. Again, the trajectory was generated as G-Code and run at 90 mm/s. This time, the error path is plainly visible and no amplification has been applied, although it is somewhat less clear that the PI controller follows the path more closely than the open-loop system. Computing the mean trajectory error reveals an improvement: The open-loop error is 0.37 mm with a standard deviation of 0.22 mm, while the closed-loop controller has a mean error of 0.21 mm with a standard deviation of 0.11 mm, an improvement of 43%.

Figure 10 (b) and (c) again plot the velocity of the carriage. Because the corners are sharp, the printer decelerates to reach them, and since the features are so small, the printer never reaches full speed between moves, resulting in a saw-tooth-like command velocity. This time the measured velocity is less noisy, and the PI controller shows slightly less lag in matching the command velocity.

Trajectory following performance with the PI controller is not always better than the open-loop case, however. Removing slight oscillations just after corners have been a consistent problem addressed by tuning acceleration settings in open-loop firmwares, and square corners where one axis comes to a complete stop and the other continues continue to be a struggle. Sometimes the PI controller's high P gain creates even worse oscillations. Errors of up to 0.4mm have been observed and would need to be addressed with more careful tuning of the controller.



(a)

(b)

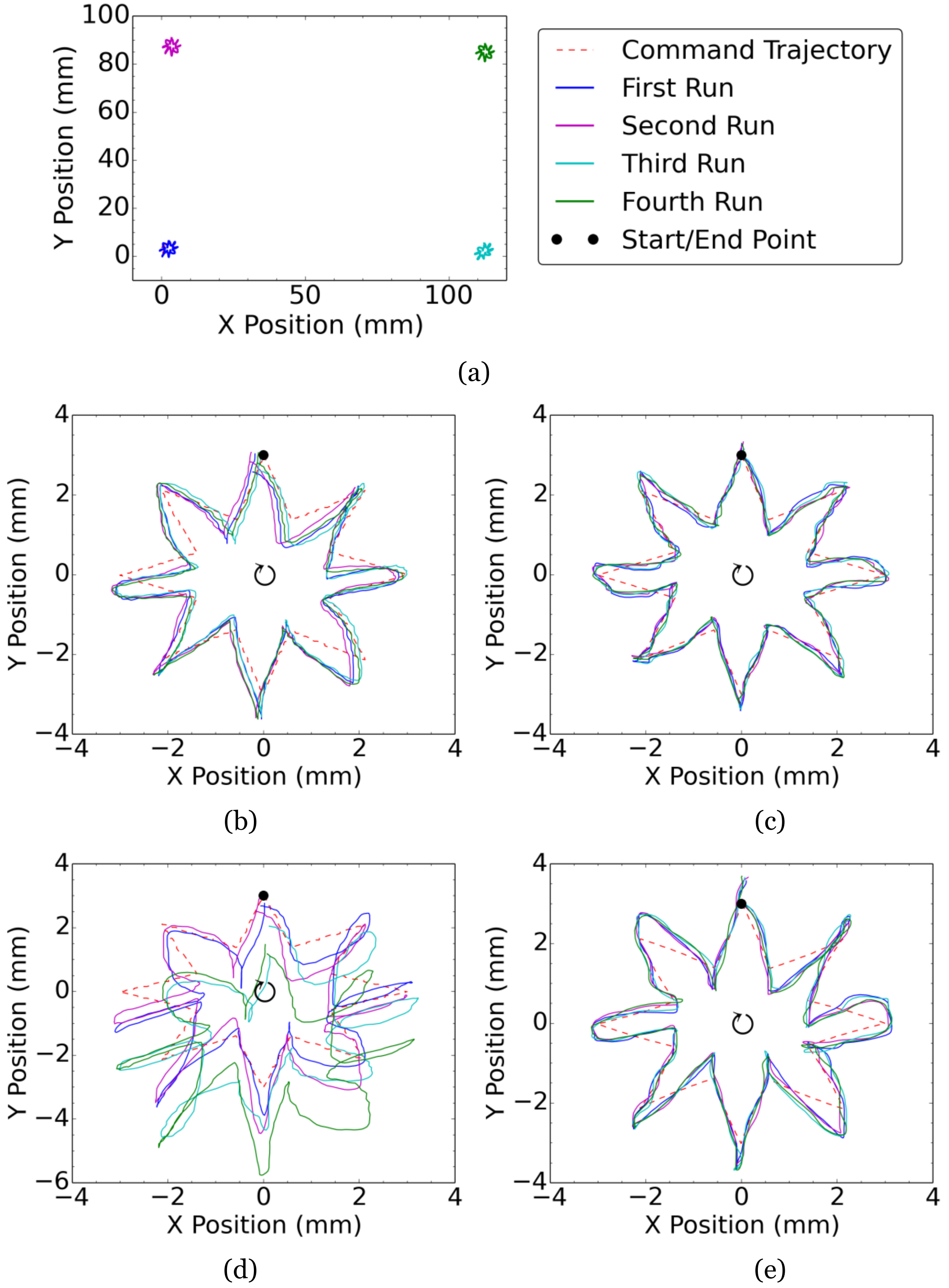
(c)

Figure 10. Open-loop vs. PI closed-loop star tracking. (a) Tracking error (b) Open-loop velocity profile (c) Closed-loop velocity profile.

5.3. Acceleration Performance

Beyond simply following trajectories more accurately, another advantage of closed-loop control is its ability to better compensate for errors produced when the system's motion parameters are set closer to the maximum capability of the hardware. To demonstrate this, two parameters are adjusted: acceleration rate and maximum speed change. Acceleration is discussed in this subsection, while cornering speed is handled in the following one.

Figure 11 shows the result of a test in which the small star pattern is drawn in four sections of the motion space (see Figure 11 (a)). The trajectory is identical to that used in the previous section, but run at increased maximum speed of 150 mm/s. Figure 11 (b) shows the four traces from the open-loop controller overlaid (such that the command trajectories all exactly coincide). Figure 11 (c) shows the same G-Code executed under closed-loop PI control. The differences between Figure 11 (b) and (c) are rooted in the integral component of the PI controller. Slight mis-calibration of the constants used by the printer to convert between mm and motor steps, along with irregularities in the belt and stepper motor geometries result in slight differences between the trajectories in different parts of the build plate (roughly 0.2 mm in this case). These differences were observed in a wide variety of geometries, even when operating at slow speeds, and are not indications of stochastic behavior in repeated measurements at the same place (repeatedly executing identical G-Code has consistently shown an envelope of less than 0.1 mm). This ability to tune out calibration error and stage irregularities is another advantage of a closed-loop controller.



(a)

(b)

(c)

(d)

(e)

Figure 11. High acceleration performance. (a) Location on print bed of the test runs. The same G-Code was executed under normal acceleration (a = 3000 mm/s²) for both open-loop (b) and closed-loop (c), then again under higher acceleration (*a* = 6000 mm/s²) for open-loop (d) and PI control (e).

Figure 11 (c) and (e) show the same trajectory, but with acceleration set to double its normal value, again for the open and closed-loop modes, respectively. The normal acceleration of 3000 mm/s^2 is increased to 6000 mm/s^2, well above the recommended operating settings for this kind of printer (Sailfish default acceleration is 1000 mm/s^2). As a result, the open-loop controller quickly starts skipping steps, as indicated by the traces drifting as the runs progress. The PI controller, however, can adjust for skipped steps and handles the increased acceleration much better; the tracking is not as good as it was before, but appears almost as good as that of the open-loop controller running at normal acceleration settings from Figure 11 (b).

These qualitative observations are reflected in the trajectory errors reported in Table 1. Although closed-loop normal acceleration has the lowest overall error, the closed-loop, high acceleration error is slightly less than that from the open-loop, normal acceleration trial. The increase in acceleration has a significant effect on speed: doubling the acceleration results in an average speedup of 0.168 s (26%). The speed gains seen in actual prints will vary, however, depending on how much of the print is spent accelerating or decelerating.

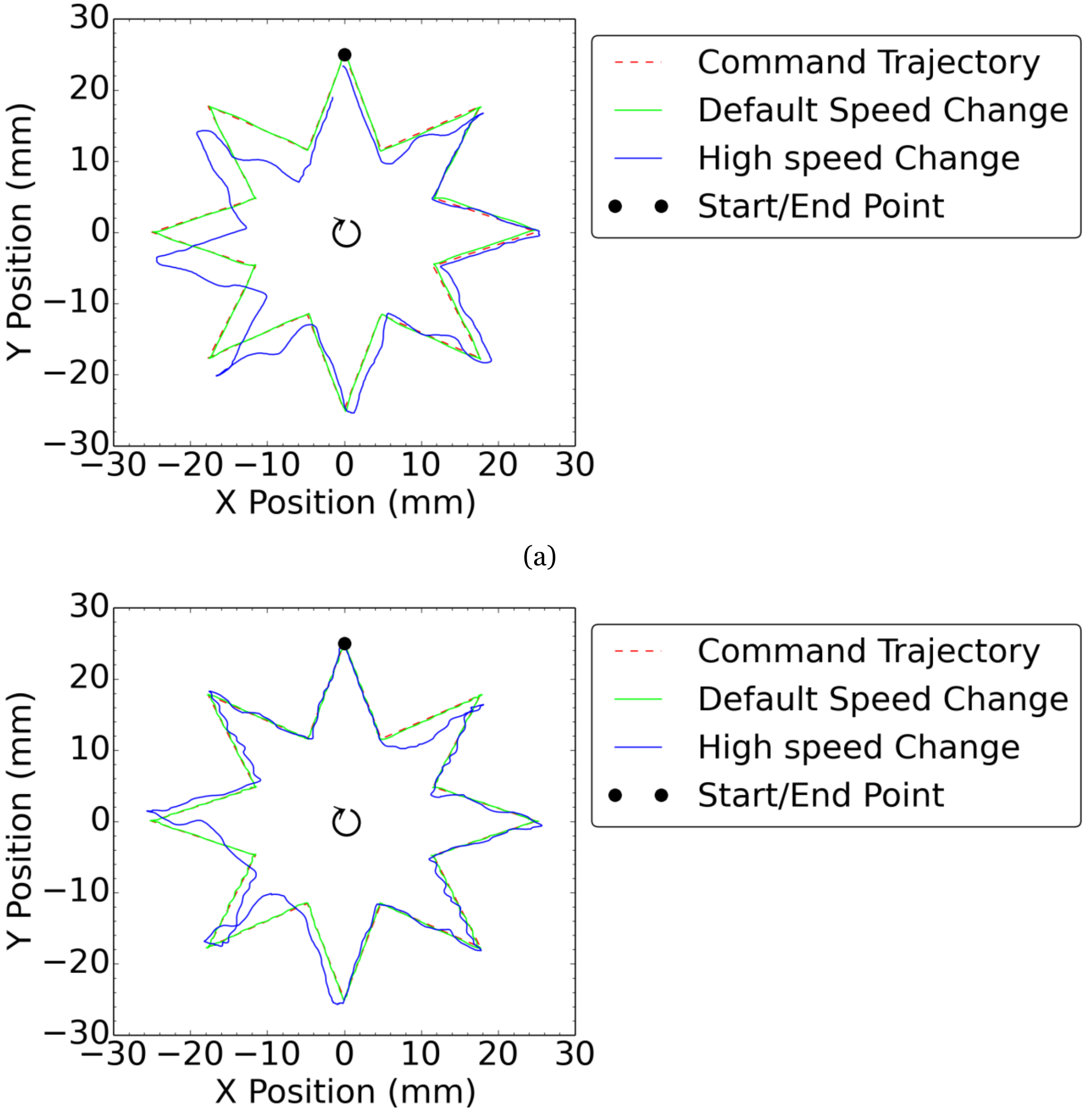
Table 1. Trajectory error of controllers with normal and high acceleration.

	Figure 8	Run Mean (Standard Deviation) - mm				Average (mm)	Average Time (s)
		1	2	3	4		
Open-loop, a= 3000 mm/s	(b)	0.413 (0.227)	0.419 (0.221)	0.366 (0.216)	0.364 (0.218)	0.391	0.646
Closed-loop, a = 3000 mm/s	(d)	0.227 (0.111)	0.207 (0.107)	0.214 (0.113)	0.192 (0.105)	0.21	0.644
Open-loop, a = 6000 mm/s	(c)	0.717 (0.45)	0.7 (0.444)	1.369 (0.847)	1.915 (0.592)	1.175	0.473
Closed-loop, a = 6000 mm/s	(e)	0.374 (0.194)	0.345 (0.185)	0.378 (0.203)	0.352 (0.186)	0.362	0.482

5.4. Cornering Speed

In addition to acceleration, another parameter that influences print speed is the cornering speed, the maximum magnitude of the difference between velocity before and velocity after a corner. Figure 12 shows the effect this parameter has on a star-shaped command trajectory much larger than that used in previous experiments and run at 150 mm/s. Figure 12 (a) shows two responses, one with the speed change set to the default, 15 mm/s, and the other at a much higher 300 mm/s, effectively disabling all deceleration around corners. Figure 12 (b) shows the same comparison for the closed-loop controlled system. When the speed change is set so high, the motor torque is unable to accelerate the carriage fast enough and step skipping occurs. The open-loop controller is ignorant of the error and continues on with the move, while the closed-loop controller detects and corrects this offset, albeit with a significant amount of oscillation. The result is a reduction in trajectory error (from 5.19 mm in the open-loop, high speed change case to 1.19 mm in the closed-loop case), although this extreme example is beyond the error tolerance of any real-world application. The time required for even this simple geometry is also reduced, from 2.2 s to 1.7 s, a savings of 23%.

This capability could be especially useful in reducing the need for the extruder to slow down and then speed back up when the gantry is executing corners, improving the consistency and surface finish of the printed part. In addition, detecting and correcting skipped steps improves print reliability. No longer will a single skipped step cause a shift in all geometry following it, but rather a quick bump on the surface of the part. What is currently a cause of print failure can be reduced to a surface defect easily removed in post processing.



(a)

(b)

Figure 12. High speed change performance. (a) Open-loop control at normal speed change (15 mm/s max) and high speed change (300 mm/s max). (b) PI control performance under normal and high speed change.

6. Lessons Learned

One of the primary purposes of a master's thesis is to teach the student real-world lessons or skills. By that definition, this thesis has been a marvelous success, creating opportunity for me to learn and grow in a wide variety of areas along each stage of this project. The result is a better-equipped, more knowledgeable, and (hopefully) wiser graduate student, scholar, and engineer.

I delved deeper into the design, selection, and characterization of sensor systems than I ever had before. Learning to choose a sensor from a range of technologies available, especially taking into account the needs and limitations of each sensor solution showed me the complexity of the “feedback” block my controls courses have frequently taken for granted. Dealing with quantization error, signal quality, and sensor bandwidth have all been important topics about which I knew little prior to this work. When designing the Y axis sensor mount, for example, I discovered motion on the X gantry created reaction forces sufficient to move the encoder away from the magnetic scale, causing it to lose track. The solution implemented involved a flexure to decouple normal displacements from the sensor, enabling it to keep its alignment.

Electrically, I grew in my understanding and ability to design and debug interface busses, both in this thesis and in the earlier work on the IMC system itself. Learning about bus bandwidth, signaling schemes, and writing my own bit-banged SPI interface helped me grow in my appreciation for how fast and easy electrical engineers have made USB, SATA, and a host of other data busses.

My software skills also grew as I delved deeper than ever before into the ARM and AVR instruction sets, writing extensive real-time C code for the ARM platform, debugging embedded systems, and making the leap away from emulated COM ports towards raw HID interfaces. My Python scripting skills grew as well as I designed a flexible front-end to control and interrogate the axis controller, a system I hope to reuse in future projects.

Finally, my growth in control systems theory and practice was significant. It is one thing to read about controllers and do practice problems in Matlab, and altogether something else to try to implement them on embedded hardware and make them work with a real, physical system. Even though none of the more complex controllers I developed wound up working, I still learned valuable lessons from the exercise, and appreciated the opportunity to experiment with all I have been learning in class. This area remains one of the most unfinished of the project (since it builds on all previous pieces), and is one of the biggest areas for future work.

7. Future Work

This thesis has just begun to scratch the surface of what closed-loop control can bring to the 3D printing world. More work has yet to be done, however, before this project is ready to be of practical use on a commercially-available robot.

The foremost area of future work centers on the development of more accurate, better-performing control algorithms. Getting the more complex controllers described here and elsewhere in the literature will be a major step towards realizing all that closed-loop control is capable of.

Further debugging and optimization must be performed on the IMC framework, bringing it up to the same level of reliability as RAMPS and other open-source 3D printing motherboards. Polishing and testing to both the IMC and closed-loop controller codebases will be required before they can move out of the development phase. Also, designing and fabrication of a PCB to hold the IMC axis electronics would significantly reduce system complexity and allow easy reproduction of the project to extend it to other axes or other printers.

Extending the project to include the extruder axis is a critically-important area of further study. Understanding and controlling the interplay between the gantry and the extruder will be key to improving print speed and quality. Because the extruder contains a more complex mechanical system, its controller will be significantly more interesting and complex than that for the X and Y gantry axes. Closing the loop on the Z axis would also be worthwhile, but is of lower priority because the Z axis is not generally involved in coordinated motion.

8. Conclusion

This thesis has demonstrated significant performance advantages in using closed-loop control on the X and Y axes of a 3D printer gantry. After building the required mechanical, electrical, software, and control components, improvements in accuracy, speed, and error recovery have been observed and quantified. The result is a system suitable for integration into a low-cost 3D printer which can increase performance without undue increase in cost, helping to meet progressively more stringent customer requirements in the competitive low end of the additive manufacturing space.

Even though attempts to implement complex controllers failed, a simple PI controller, roughly tuned, showed performance improvements that reduce path tracking error in circle and star trajectories by more than 40%, cut print time by up to 25% by increasing acceleration rate with only minimal loss of accuracy, and detected and recovered from skipped steps in a matter of millimeters. Further development and better tuning of the control algorithm should result in further improvements.

It is hoped that this project will spur interest and further research in closed-loop control of 3D printers by demonstrating the performance gains possible and providing a reference implementation that realizes these gains. Better, cheaper, faster printers are to the benefit of all.

9. Bibliography

[1] H. Lipson, *Fabricated: the new world of 3D printing*. Indianapolis, Indiana: John Wiley & Sons, 2013.

[2] "RepRap." [Online]. Available: http://reprap.org. [Accessed: 07-Jul-2014].

[3] "NEMA ICS 16 Industrial Control and Systems Motion/Position Control Motors, Controls, and Feedback Devices." National Electrical Manufacturers Association, 2001.

[4] "megaAVR Microcontrollers," *Atmel*. [Online]. Available: http://www.atmel.com/products/microcontrollers/avr/megaAVR.aspx. [Accessed: 19-Aug-2014].

[5] "MakerBot Replicator Z18 3D Printer." [Online]. Available: http://store.makerbot.com/replicator-z18. [Accessed: 28-Jul-2014].

[6] "ZMorph 3d printers Specifications," *Zmorph*. [Online]. Available: http://zmorph3d.com/specifications/. [Accessed: 28-Jul-2014].

[7] "LulzBot TAZ 4 3D Printer," *LulzBot*. [Online]. Available: http://www.lulzbot.com/products/lulzbot-taz-4-3d-printer#tab-body. [Accessed: 28-Jul-2014].

[8] "Panowin F3CL Desktop Printer," *Panowin*. [Online]. Available: http://www.panowin.com/2.product%20and%20service/3D%20printer/3.F3CL.html. [Accessed: 11-Aug-2047].

[9] "MakerBot MakerWare," *MakerBot*. [Online]. Available: http://www.makerbot.com/makerware. [Accessed: 12-Aug-2014].

[10] "rappy32-3d-printer." [Online]. Available: http://www.stellamove.com/#!rappy32eng/c13qm. [Accessed: 11-Aug-2014].

[11] "磐纹科技," *Panowin*. [Online]. Available: http://www.panowin.com/2.product%20and%20service/6.printing%20video.html. [Accessed: 11-Aug-2014].

[12] "RAMPS 1.4," *RepRap.org*. [Online]. Available: http://reprap.org/wiki/RAMPS_1.4. [Accessed: 30-Jul-2014].

[13] "Sailfish Usage Manual," *MakerBot*. [Online]. Available: http://www.makerbot.com/sailfish/. [Accessed: 11-Aug-2014].

[14] G. Brown and D. Campbell, *Principles of Servomechanisms; dynamics and synthesis of closed-loop control systems*. New York: J. Wiley, 1948.

[15] S. Ajdukovic, B. Kuzmanovic, and P. Crnosija, "Microcomputer implementation of optimal algorithms for closed-loop control of hybrid stepper motor drives," *IEEE Trans. Ind. Electron.*, vol. 47, no. 6, pp. 1319–1325, Dec. 2000.

[16] C. W. De Silva, *Sensors and actuators: control systems instrumentation*. Boca Raton, FL: CRC Press, 2007.

[17] M. Bodson, J. N. Chiasson, R. T. Novotnak, and R. B. Rekowski, "High-performance nonlinear feedback control of a permanent magnet stepper motor," *IEEE Trans. Control Syst. Technol.*, vol. 1, no. 1, pp. 5–14, Mar. 1993.

[18] M. Zribi and J. Chiasson, “Position control of a PM stepper motor by exact linearization,” *IEEE Trans. Autom. Control*, vol. 36, no. 5, pp. 620–625, May 1991.

[19] M. Zribi, H. Sira-Ramirez, and A. Ngai, “Static and dynamic sliding mode control schemes for a permanent magnet stepper motor,” *Int. J. Control*, vol. 74, no. 2, pp. 103–117, Jan. 2001.

[20] P. Krishnamurthy and F. Khorrami, “Robust adaptive voltage-fed permanent magnet step motor control without current measurements,” *IEEE Trans. Control Syst. Technol.*, vol. 11, no. 3, pp. 415–425, May 2003.

[21] Burg, Hu, Dawson, and Vedagarbha, “A global exponential position tracking controller for a permanent magnet stepper motor via output feedback,” 1994, pp. 213–220 vol.1.

[22] “General Purpose Motion Control ICs,” *Avago Technologies*. [Online]. Available: http://www.avagotech.com/pages/en/motion_control_encoder_products/integrated_circuits/controller_ic/hctl-1101/. [Accessed: 01-Aug-2014].

[23] C. S. Teo, K. K. Tan, S. Y. Lim, S. Huang, and E. B. Tay, “Dynamic modeling and adaptive control of a H-type gantry stage,” *Mechatronics*, vol. 17, no. 7, pp. 361–367, Sep. 2007.

[24] Chuxiong Hu, Bin Yao, and Qingfeng Wang, “Coordinated Adaptive Robust Contouring Controller Design for an Industrial Biaxial Precision Gantry,” *IEEEASME Trans. Mechatron.*, vol. 15, no. 5, pp. 728–735, Oct. 2010.

[25] R. Ramesh, M. A. Mannan, and A. N. Poo, “Tracking and contour error control in CNC servo systems,” *Int. J. Mach. Tools Manuf.*, vol. 45, no. 3, pp. 301–326, Mar. 2005.

[26] N. Janvier, J. Clement, P. Fajardo, and G. Cuní, “ICEPAP: An Advanced Motor Controller for Scientific Applications in Large User Facilities,” in *ICALEPCS2013: Proceedings of the 14th International Conference on Accelerator & Large Experimental Physics Control Systems*, San Francisco, CA, USA, 2013, pp. 766–769.

[27] “Closed-loop stepper controller progress,” *RepRap*, 06-Nov-2013. [Online]. Available: http://forums.reprap.org/read.php?1,263680. [Accessed: 01-Aug-2014].

[28] “Closed-loop control for low-cost 3D printers,” *Element14*, 21-Jan-2014. [Online]. Available: http://www.element14.com/community/thread/30891/l/closed-loop-control-for-low-cost-3d-printers. [Accessed: 01-Aug-2014].

[29] “US Patent # 5,303,141. Model generation system having closed-loop extrusion nozzle positioning - Patents.com.” [Online]. Available: http://www.patents.com/us-5303141.html. [Accessed: 28-Jul-2014].

[30] D. L. Cohen and H. Lipson, “Geometric feedback control of discrete-deposition SFF systems,” *Rapid Prototyp. J.*, vol. 16, no. 5, pp. 377–393, 2010.

[31] M. D. Sorensen and B. Weiss, “IMC-Axis-Simple.” 12-Aug-2014.

[32] “MBot Printer,” *MBOT 3D Printer Store.* [Online]. Available: http://www.mbot3d.com/products/mbot-cube-1-weeks-lead-time. [Accessed: 05-Jul-2014]. Archived at http://www.webcitation.org/6QqcYiOLm.

[33] MakerBot, “The MakerBot Replicator,” *MakerBot Thingiverse*, 12-Mar-2012. [Online]. Available: http://www.thingiverse.com/thing:18813. [Accessed: 05-Jul-2014].

[34] P. Stoffregen, “Teensy 3.1,” *PJRC.com*. [Online]. Available: http://pjrc.com/teensy/teensy31.html. [Accessed: 07-Jul-2014]. Archived at http://www.webcitation.org/6Qtm65M3C.

[35] “AS5311 Linear Sensor,” *AMS*. [Online]. Available: http://www.ams.com/eng/Products/Position-Sensors/Linear-Incremental-Magnetic-Position-Sensors/AS5311. [Accessed: 07-Jul-2014].

[36] “AS5000-MS10-300 Magnetic Strip,” *AMS*. [Online]. Available: http://www.ams.com/eng/Products/Position-Sensors/Magnets/AS5000-MS10-300. [Accessed: 07-Jul-2014].

[37] “Pololu - A4988 Stepper Motor Driver Carrier.” [Online]. Available: http://www.pololu.com/product/1182. [Accessed: 28-Jul-2014].

[38] M. Sorensen, “AS5311 Linear Encoder Boards,” *MDS*, 12-Aug-2014. [Online]. Available: http://www.sorens.in/posts/2014-8-12-linear-encoder. [Accessed: 12-Aug-2014]. Archived at http://www.webcitation.org/6Rme0WhVC.

[39] “K20 Sub-Family Reference Manual.” Freescale Semiconductor, Dec-2012.

[40] G. C. Goodwin and K. S. Sin, *Adaptive filtering prediction and control*. Englewood Cliffs, N.J.: Prentice-Hall, 1984.

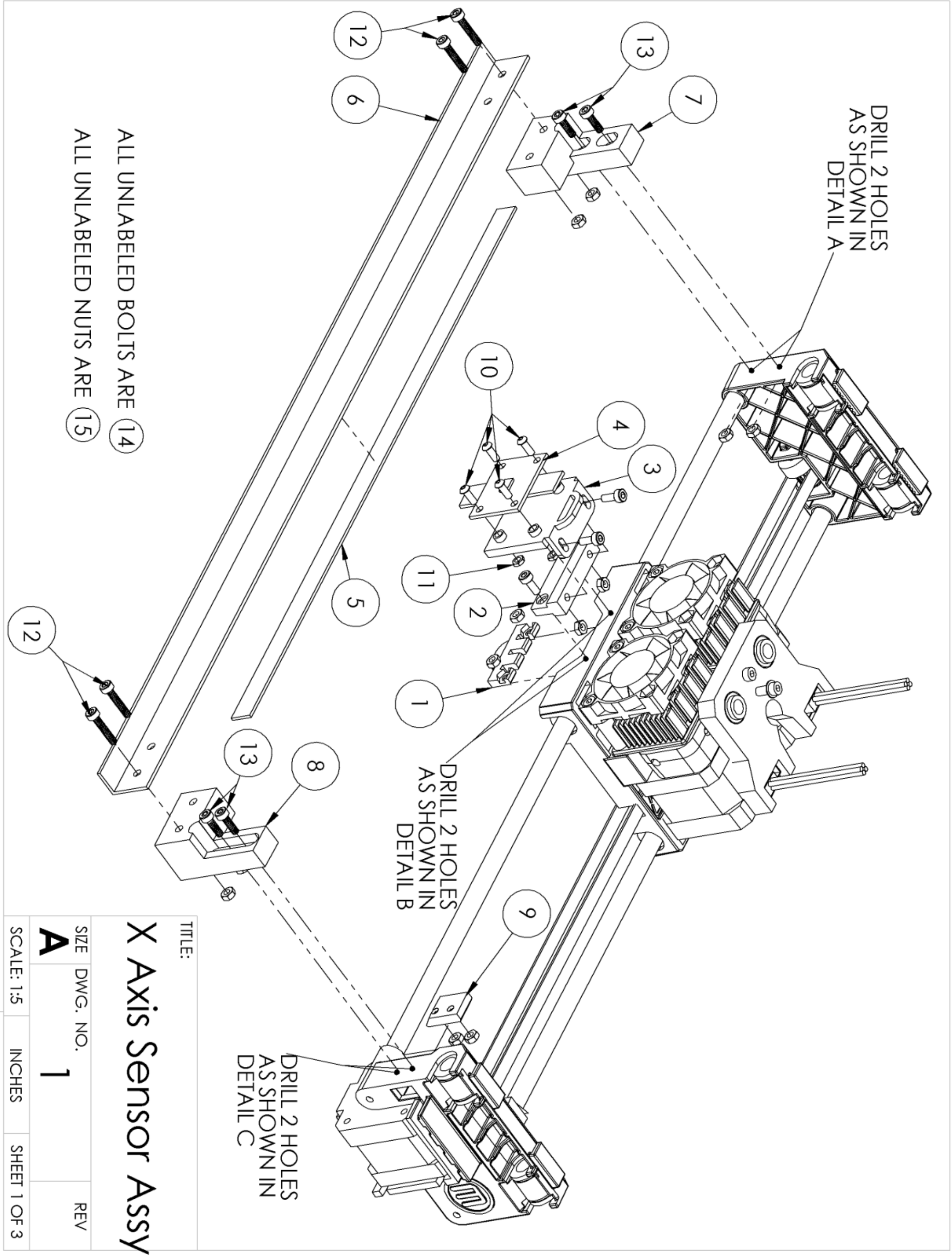
[41] K. J. Åström, *Adaptive control*, 2nd ed., Dover ed. Mineola, N.Y: Dover Publications, 2008.

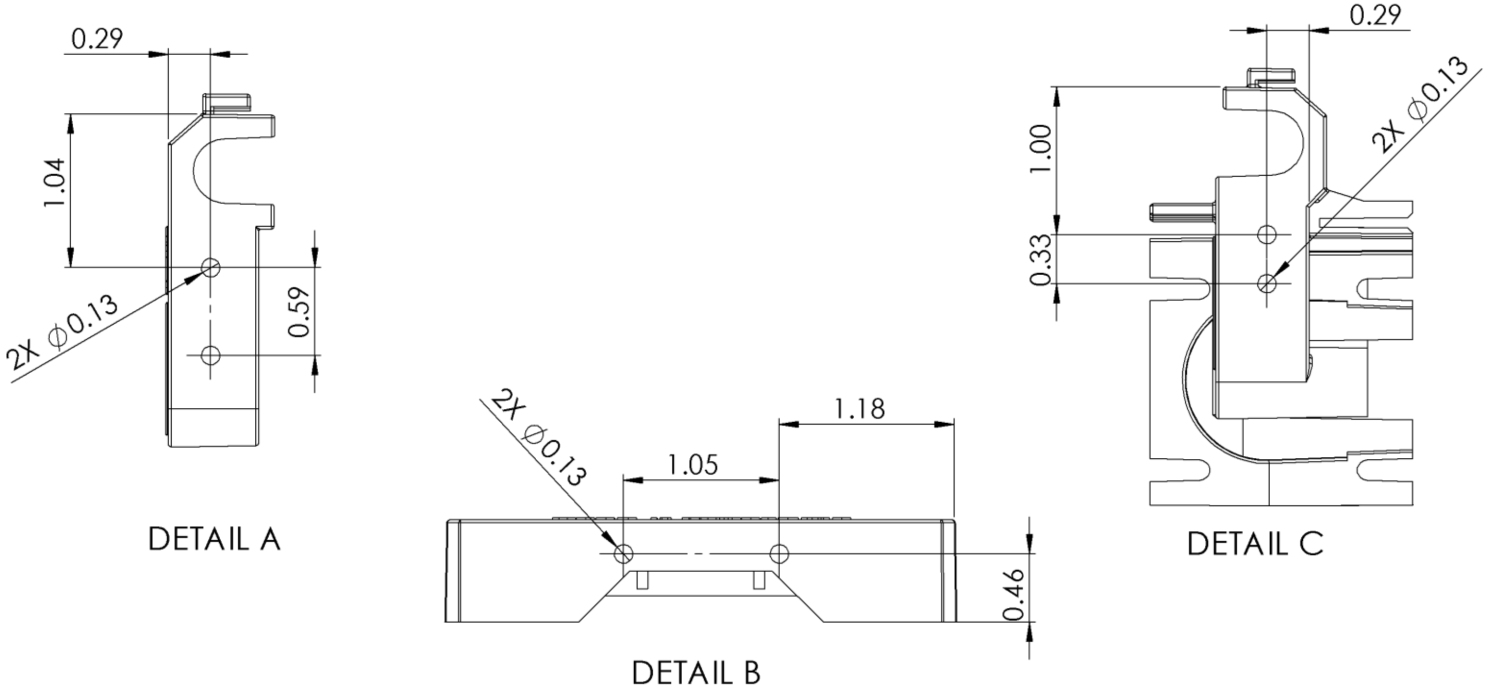
[42] G. Dahlquist, *Numerical methods*. Englewood Cliffs, N.J: Prentice-Hall, 1974.

[43] “System Identification Toolbox – MATLAB,” *MathWorks*. [Online]. Available: http://www.mathworks.com/products/sysid/. [Accessed: 22-Aug-2014].

[44] “Products: HB stepping motor,” *Moons’*. [Online]. Available: http://www.moonsindustries.com/Products/Hybrid_Stepper_Motors/HB2P_17HD/. [Accessed: 23-Aug-2014].

10. Appendix A: Mechanical Drawings


DRILL 2 HOLES AS SHOWN IN DETAIL A
13
7
12
6
ALL UNLABELED BOLTS ARE 14
ALL UNLABELED NUTS ARE 15
10
4
3
11
5
2
1
12
13
8
DRILL 2 HOLES AS SHOWN IN DETAIL B
9
DRILL 2 HOLES AS SHOWN IN DETAIL C
TITLE:
X Axis Sensor Assy
SIZE
A
DWG. NO.
1
REV
SCALE: 1:5
INCHES
SHEET 1 OF 3



ADJUST MECHANISM SO TOP OF AS5311 CHIP IS NOT TOUCHING SCALE BUT IS WITHIN 0.5 mm (0.02 in)
AND AS5311 X-Z CENTERLINE IS ALIGNED WITH THE CENTER OF SCALE.

TITLE:		
X Axis Sensor Assy		
SIZE A	DWG. NO. 1	REV
SCALE: 1:1	INCHES	SHEET 2 OF 3

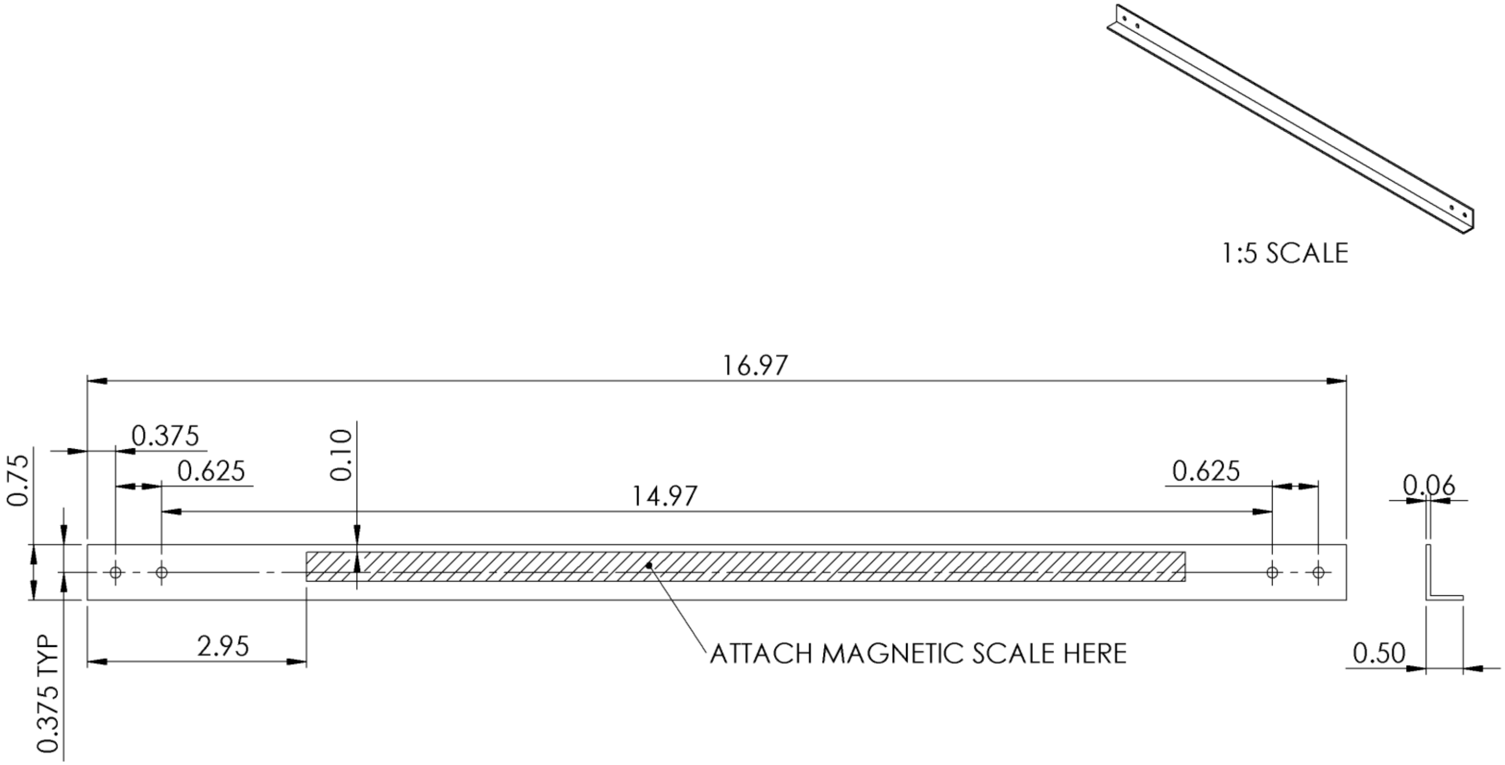
Callout	Qty	Part Name	Source
1	1	X Screw Holder	Printed
2	1	X Mount Plate	Printed
3	1	X Sensor Mount	Printed
4	1	Sensor PCB	Printed Circuit Board
5	1	AS5311-compatible Magnetic Scale	COTS part, 1x10x300mm
6	1	X Strip Mount	3/4x1/4x1/16 in Aluminum Angle Stock
7	1	X Axis Strip Mount Left	Printed
8	1	X Axis Strip Mount Right	Printed
9	1	X Axis Nut Holder Right	Printed
10	4	#4-40x0.5in Round Head Slot Screw	COTS part
11	4	#4-40 Small Pattern Nuts	COTS part
12	4	M3x16 mm SHCS	COTS part
13	4	M3x10 mm SHCS	COTS part
14	4	M3x8 mm SHCS	COTS part
15	12	M3 Nut	COTS part

TITLE:

X Axis Sensor Assy

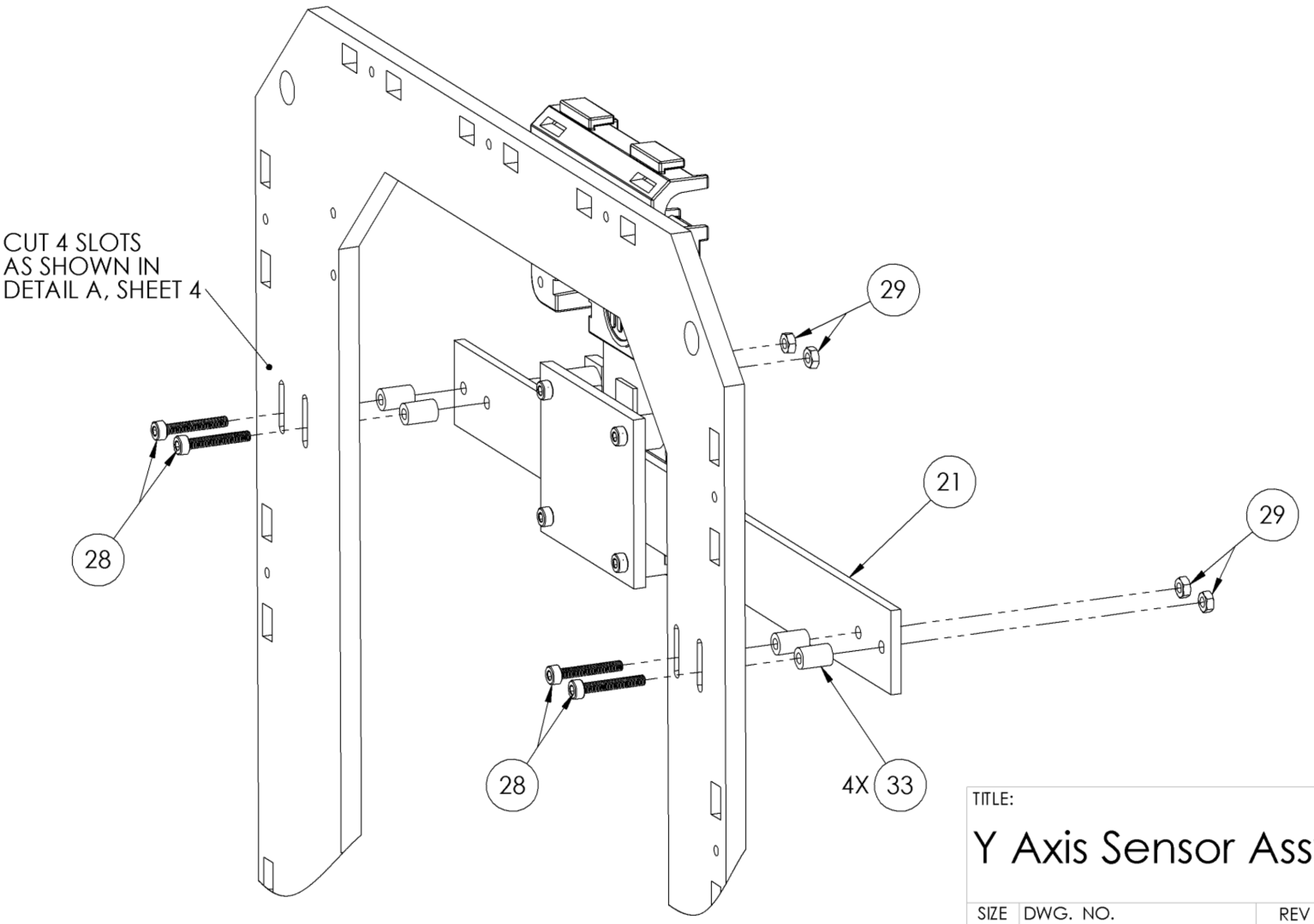
SIZE A DWG. NO. 1 REV

SCALE: 1:5 INCHES SHEET 3 OF 3



MADE FROM 0.5 X 0.75 X 1/16 IN ALUMINUM ANGLE STOCK

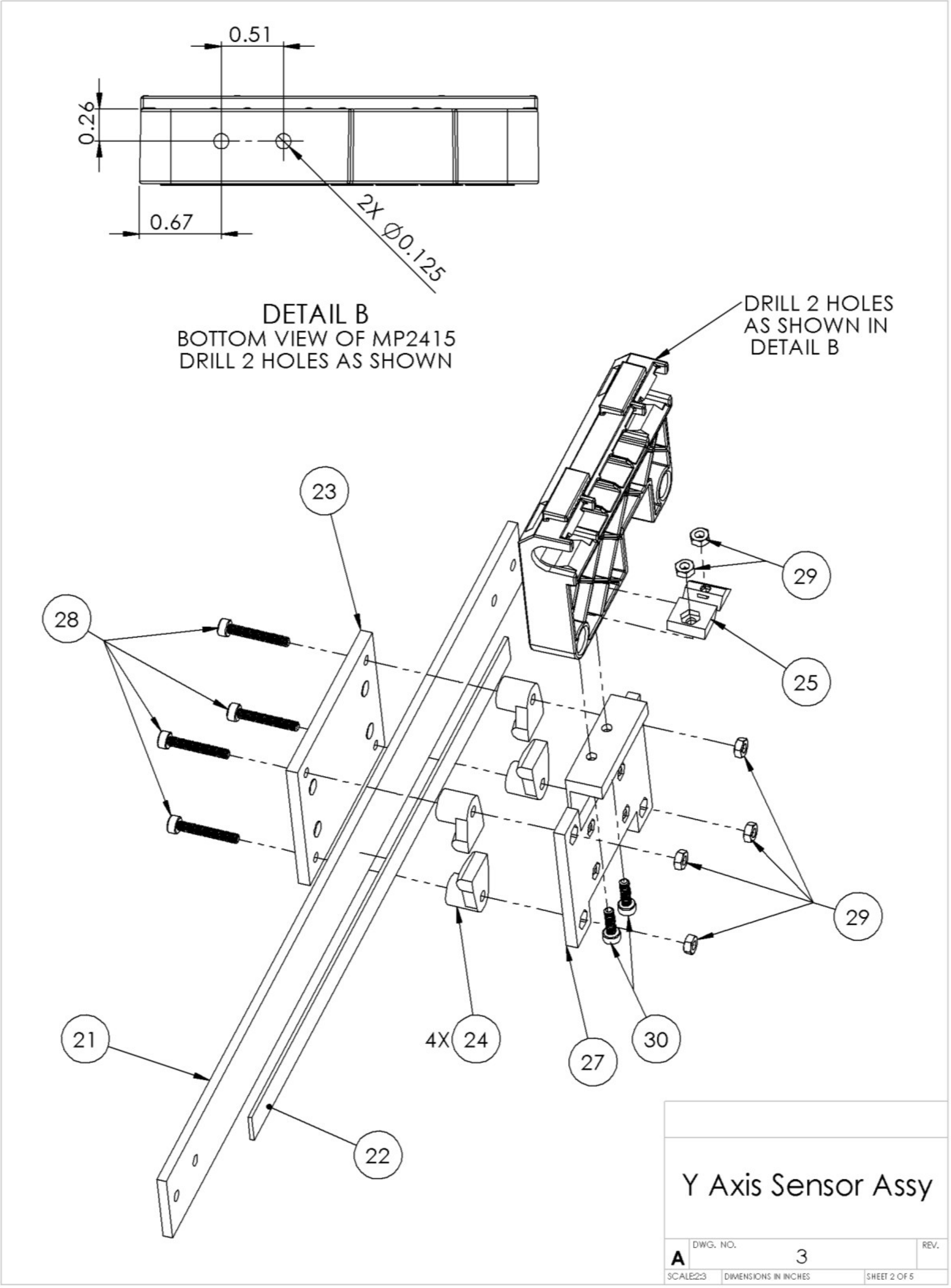
TITLE: X Strip Mount		
SIZE A	DWG. NO. 2	REV
SCALE: 1:2	INCHES	SHEET 1 OF 1

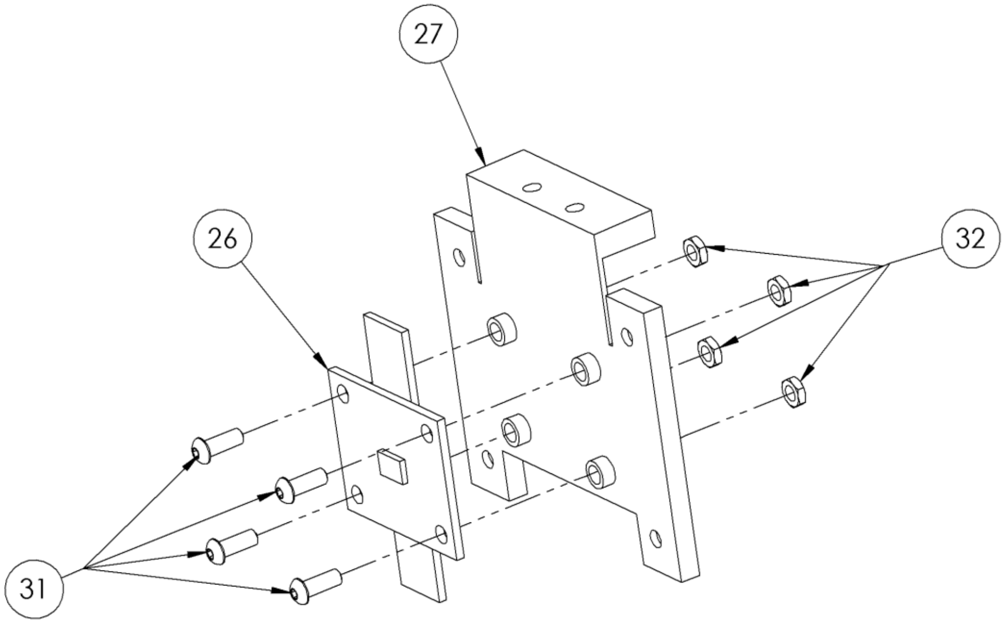


CUT 4 SLOTS
AS SHOWN IN
DETAIL A, SHEET 4

TITLE:		
Y Axis Sensor Assy		
SIZE A	DWG. NO. 3	REV
SCALE: 2:3	INCHES	SHEET 1 OF 5

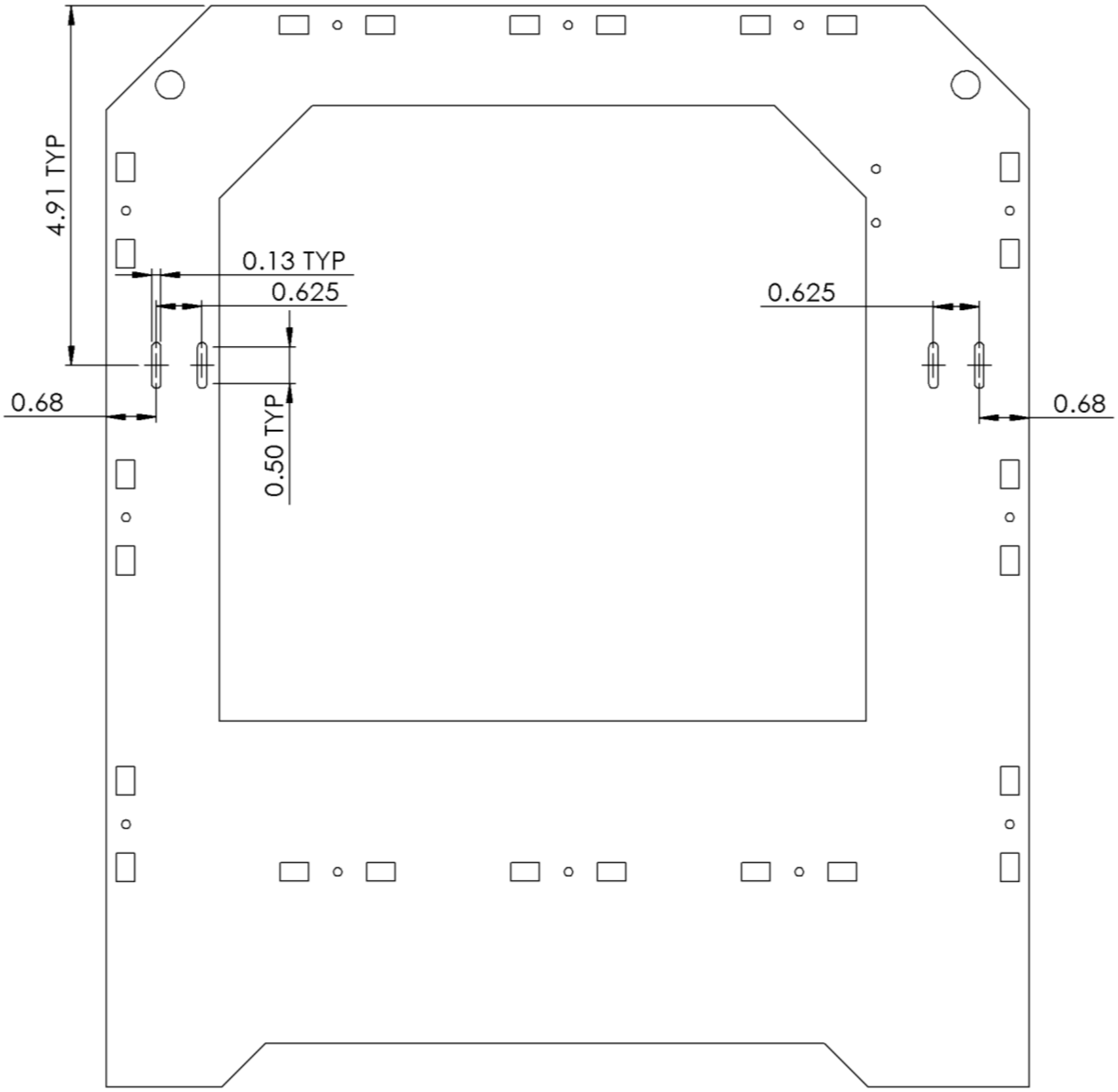
1


0.51
0.26
0.67
2X Ø0.125
DETAIL B
BOTTOM VIEW OF MP2415
DRILL 2 HOLES AS SHOWN
DRILL 2 HOLES
AS SHOWN IN
DETAIL B
23
28
29
25
29
21
4X 24
27
30
22
Y Axis Sensor Assy
A
DWG. NO.
3
REV.
SCALE:2:3
DIMENSIONS IN INCHES
SHEET 2 OF 5



TITLE:		
Y Axis Sensor Assy		
SIZE A	DWG. NO. 3	REV
SCALE: 1:1	INCHES	SHEET 3 OF 5

1



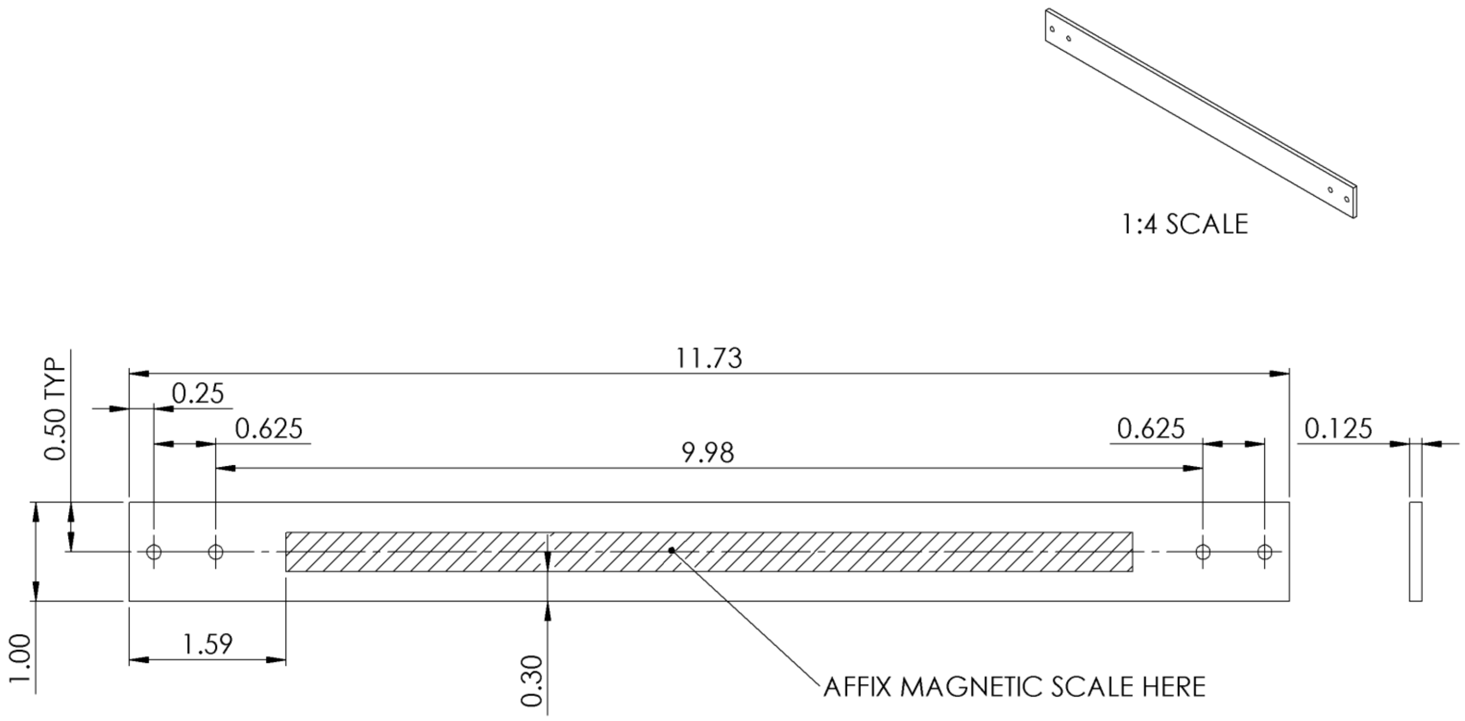
DETAIL A
FRONT VIEW OF MP2198
CUT 4 SLOTS AS SHOWN

Y Axis Sensor Assy

A	DWG. NO. 3	REV.
SCALE:1:2	DIMENSIONS IN INCHES	SHEET 4 OF 5

Callout	Qty	Part Name	Source
21	1	Y Strip Mount	1x1/8 in Aluminum. See Drawing 4.
22	1	AS5311-compatible Magnetic Scale	COTS Part; 1x10x217mm
23	1	Y Type 2 Sensor Carriage	Printed
24	4	Y Type 2 SC Insert	Printed
25	1	Y Nut Holder	Printed
26	1	Sensor PCB	Printed Circuit Board
27	1	Y Type 2 Sensor Mount	Printed
28	8	M3x20 mm SHCS	COTS Part
29	10	M3 Hex Nut	COTS Part
30	2	M3x10 mm SHCS	COTS Part
31	4	#4-40x0.5in Round Head Slot Screw	COTS Part
32	4	#4-40 Small Pattern Nut	COTS Part
33	4	0.25 OD 0.125 ID 0.372 in OAL Spacer	COTS Part, Trimmed to length

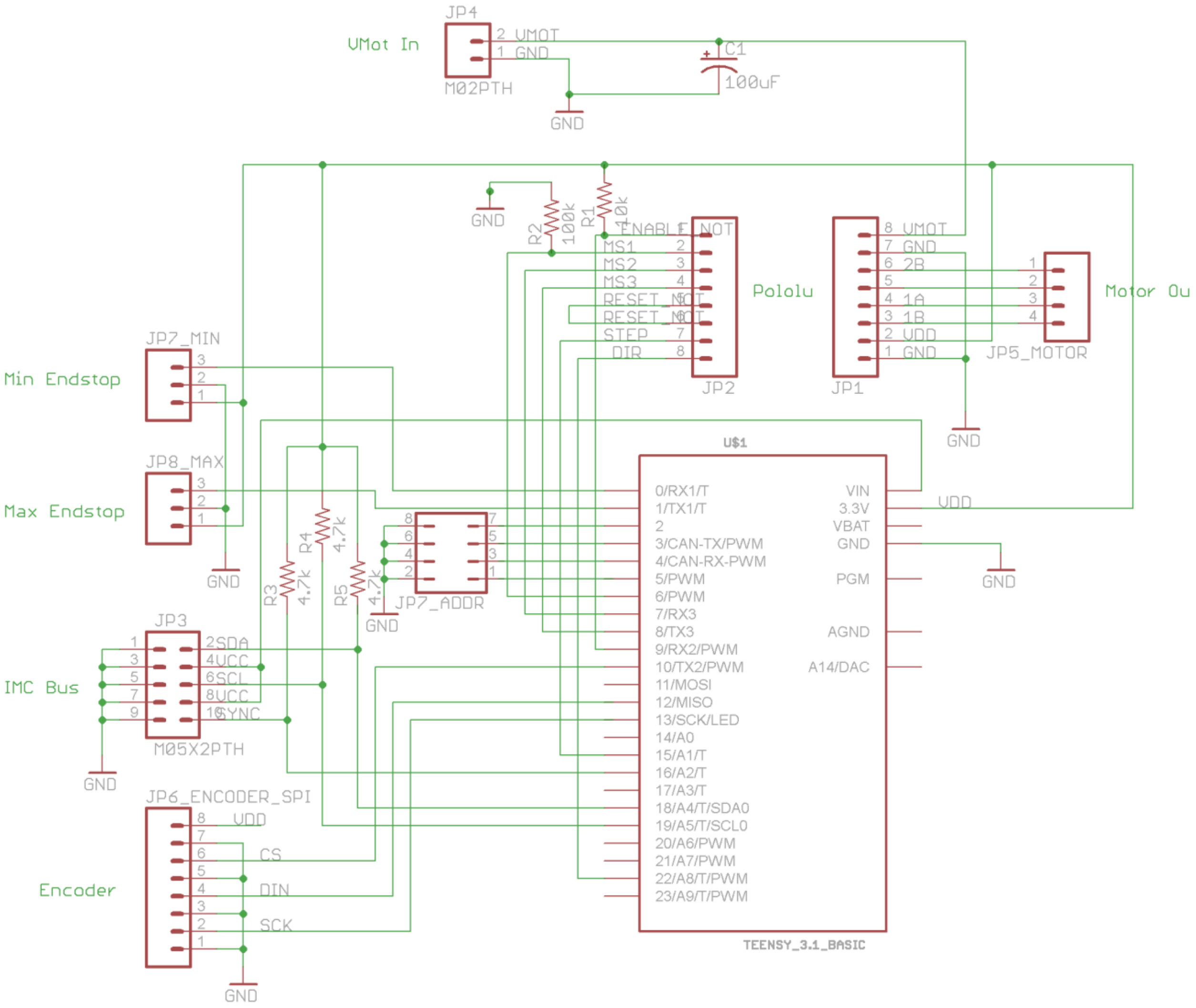
TITLE: Y Axis Sensor Assy		
SIZE A	DWG. NO. 3	REV
SCALE: 2:3	INCHES	SHEET 5 OF 5



TITLE: Y Strip Mount		
SIZE A	DWG. NO. 4	REV
SCALE: 2:3	INCHES	SHEET 1 OF 1

11. Appendix B: Electrical Drawings

The following figure shows the IMC Closed Loop Axis schematic, including connections to IMC Master, end stops, the motor, and the encoder.



For the Master Arduino, the only required connections are the IMC bus:

Signal	Pin
SCL	21
SDA	20
Sync	18

12. Appendix C: Source Files

The source code for the software developed for this thesis, the mechanical CAD, and all future updates and other errata have been made available at

https://sites.google.com/site/benweisspublic/projects/imc-closed-loop-control

13. Appendix D: Dynamic Model

Dynamic modelling is a central part of any control problem, enabling the design and evaluation of control approaches. In this project, the significant nonlinearities presented by the stepper motor and driver make any useful model likely to be inaccurate. As a result, the majority of the work on the thesis makes only minimal use of modelling. This appendix describes a nonlinear system model, the assumptions needed to linearize it, and the results obtained when sample data from the uncontrolled system is fit to the model.

13.1. Nonlinear Model

The model contains intrinsic nonlinearities in the form of motor electrodynamics and belt elasticity. Although far from capturing the full extent of these nonlinearities, the following derivation attempts to capture the basic structure of the system. All of the development is done for the X axis; the Y axis derivation follows identically except where noted.

Stepper motor models are widely available in the literature (see Section 2.4). For this derivation, we utilize the easily-understood model of a hybrid stepper motor given by de Silva in [16], Section 6.6. It assumes a constant-voltage supply and captures the mechanical and electrical dynamics only within a single step, centered at $\theta = 0$. The forces from Figure 13 (b) give the equation of motion for the motor,

$$T - T_L - T_b(\theta, \dot{\theta}) = J\ddot{\theta} \tag{1}$$

which is a second order differential equation in θ, the motor rotation away from the equilibrium position ($\dot{\theta}$ and $\ddot{\theta}$ here represent first and second derivatives of θ in time, respectively). T is the motor applied torque, T_L is the torque from the load, $T_b(\theta, \dot{\theta})$ is the damping term, and J is the inertia of the rotor.

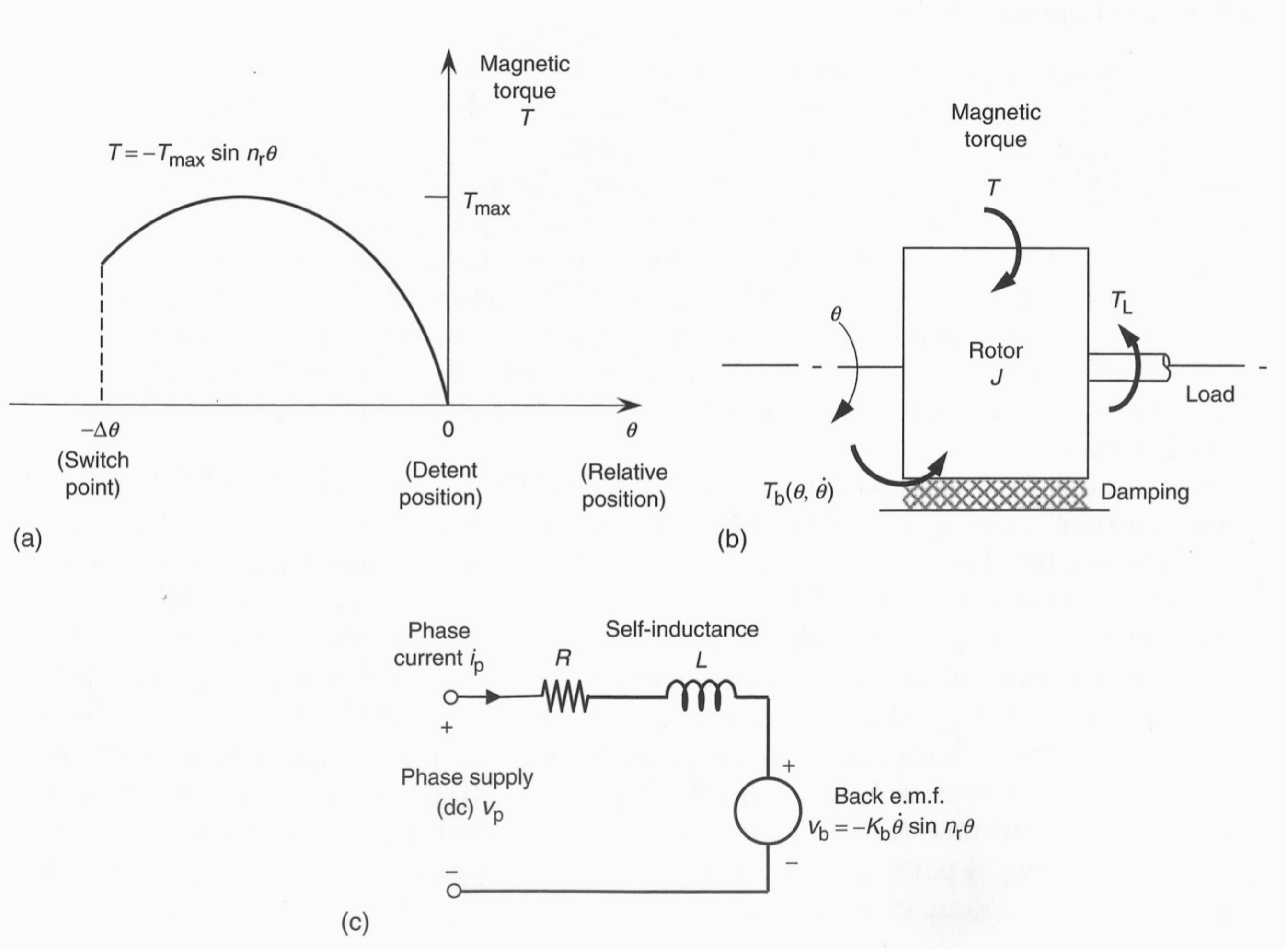


Figure 13. de Silva's stepper motor model. (a) Torque curve, (b) Mechanical dynamics, (c) Electrical dynamics. Figure from [16], p 447.

De Silva presents two approximations for motor torque. The simpler, which will be used when we linearize the model, is simply

$$T = -T_{max} \sin n_r\theta \tag{2}$$

where T_{max} is the holding torque of the motor and n_r is the number of rotor teeth (see Figure 13 (a)). The more complex model incorporates the electrical characteristics of the motor windings, and is itself simplified by truncating the Fourier expansion of the actual electromechanical dynamics. Torque is defined as,

$$T = -k_m i_p \sin n_r\theta \tag{3}$$

where k_m is the motor torque constant and i_p, the phase current, solves

$$v_p = Ri_p + L\frac{di_p}{dt} + v_b$$

where v_p is the constant supply voltage and R is the coil resistance. L and v_b are the coil inductance and back e.m.f., approximated respectively as

$$L = L_0 + L_a \cos n_r\theta$$

$$v_b = -k_b\dot{\theta} \sin n_r\theta$$

with L_o and L_a appropriately-chosen constants and k_b the motor back e.m.f. constant (see Figure 13 (c)). The experimental setup used for the thesis makes this model somewhat inaccurate, however, because the motor is driven not by a constant voltage supply, but by a switching-mode motor driver which implements its own closed-loop current controller.

The motor is coupled to the extruder by a timing belt and pulley system as shown in Figure 14. The belt is not tensioned with a tensioning device, and experiments indicate it is slightly elastic. For the X axis, the belt positions the extruder assembly; for the Y axis, the belt moves the whole X axis, but in this section we will maintain the X axis notation for clarity. The extruder is additionally supported by a pair of guide rods and four linear bearings.

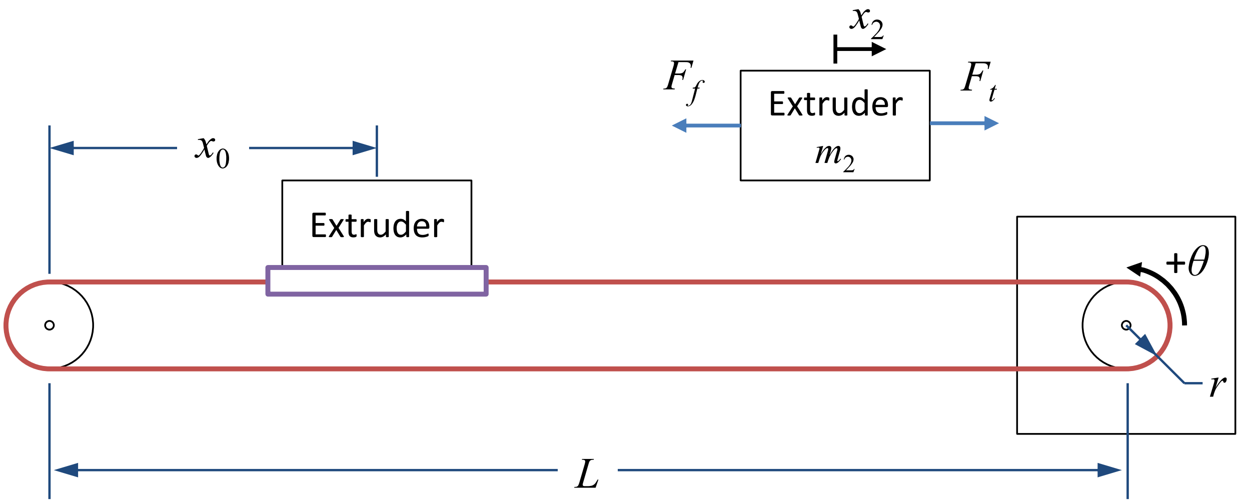


Figure 14. Schematic of the belt-driven carriage and free body diagram of the extruder.

The equation of motion for the extruder is simply,

$$m_2\ddot{x}_2 = -F_f(x_2, \dot{x}_2) + F_t(x_2, \theta) \tag{4}$$

Where m_2 is the mass of the extruder (or subsequent stage for the Y axis), $F_f(x_2, \dot{x}_2)$ is the damping force exerted by friction both in the linear bearings on the support rods and in the belt-pulley interface, and $F_t(x_2, \theta)$ is the force exerted by the belt. The damping force is some unknown function of extruder position and velocity. The tension force can be modelled as a spring according to,

$$F_t(x_2, \theta) = K_2(x_2, \theta)(-r\theta - x_2) \tag{5}$$

with nonlinear spring constant $K_2(x_2, \theta)$. Note that the spring force has the direction shown in the free body diagram whenever θ or x_2 are negative. The spring constant is a piecewise function with a different value depending on the direction in which the tensile force is applied. When it is pulling the extruder in the positive x_2 direction, the force is transmitted along the bottom of the belt, around the left idler pulley, and then across the top to the extruder. When the pull is in the opposite direction, force is transmitted to the extruder over the short section of belt on the top only. If the belt has a unit spring constant of $\hat{k}$ (Nm/m), we can write,

$$K_2(x_2, \theta) = \begin{cases} \dfrac{\hat{k}}{L - (x_0 + x_2)} & \text{for } -r\theta - x_2 > 0 \\ \dfrac{\hat{k}}{L + (x_0 + x_2)} & \text{for} - r\theta - x_2 < 0 \end{cases}$$

where x_0 represents the equilibrium position for the current step and L is the distance between the pulleys. This relation can be stated more succinctly using the signum function:

$$K_2(x_2, \theta) = \frac{\hat{k}}{L - (x_0 + x_2)\mathrm{sgn}(-r\theta - x_2)} \tag{6}$$

The definition of K_2 at the discontinuity is arbitrary; F_t will always have value 0 at that point. Also, the mechanics of the stage ensure the length of active belt cannot reach zero (which would result in an infinite value for K_2, equivalent to a rigid linkage).

Finally, the two models are linked by relating T_L, the load torque on the motor, to F_t, the tension force on the other end of the belt

$$F_t = -\frac{e}{r} T_L \tag{7}$$

where e is the load transmission efficiency. The result is a nonlinear system of differential equations, with nonlinearities caused by sinusoidal functions of θ, as well as belt dynamics.

13.2. Linearized Model

While the above model is complex with several nonlinearities, a series of simplifying assumptions reduce it to a more tractable linear model. The simplifications required involve significant assumptions which reduce the accuracy of the model; the following section points out some of the practical effects of this inaccuracy. The result is a simplified model, rephrased as a linear dynamic system, shown in Figure 15.

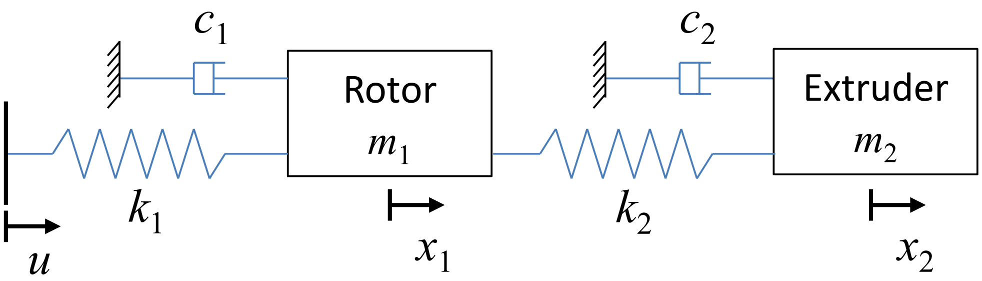


Figure 15. Linearized model, in linear coordinates.

The assumptions, briefly stated, are these:

- Motor torque is modelled as a spring, with constant k_1, between the rotor and the electromagnetic equilibrium point, u.
- The nonlinear spring, K_2, is modeled as a linear spring, k_2.
- All damping forces are assumed viscous.

The first assumption made is the modelling of the motor torque as a spring. Further approximate the simpler torque force definition given in Equation (2) by assuming $\sin\theta \approx \theta$. This assumption will hold so long as the actual rotor position stays relatively close to the magnetic equilibrium, which should be accurate during moves performed using trapezoidal velocity ramps. This will result in inaccuracies, however, in the step response, since the mechanical dynamics push the rotor far from the magnetic equilibrium, potentially even skipping steps. The simplified model replaces Equation (3), along with its dependent nonlinear differential equations, with

$$T = -T_{max} n_r \theta \qquad (8).$$

Instead of being fixed, the electromagnetic equilibrium can be changed in this model by varying the currents in the coils. Since microstepping makes the actual magnetic steps quite small, the stepping is smoothed into a continuous electromagnetic equilibrium position for the model, which can be manipulated as the system input. This introduces modelling error in the magnitude of the quantization steps averaged out. Error is also present because the torque curve may not be quite the same midway between two full step positions. The original model does not include microstepping, however, and the magnitude of this error is difficult to quantify.

Additionally, the nonlinear spring, K_2, is replaced with a constant value, k_2. Estimating a value to use for this spring constant is probably best done by experiment, but this was not attempted for this thesis. This approximation will best represent motion where the extruder assembly is farthest from the motor, where the size of the discontinuity is minimized because the equilibrium offset, x_0, is small.

Thirdly, the damping forces are all assumed to be viscous friction forces, i.e.

$$T_b(\theta, \dot{\theta}) \approx c_1' \dot{\theta} \tag{9}$$

$$F_f(x_2, \dot{x}_2) \approx c_2 \dot{x}_2 \tag{10}$$

with c_1' and c_2 appropriate constants. This assumption again reduces model accuracy. It is known that friction forces in systems like these are more complex, but any more complex friction model would make linearizing the system difficult.

Finally, for clarity, the system is converted into linear coordinates as shown in Figure 15, using the equivalence conditions

$$x_1 = -r\theta \tag{11}$$

$$F_* = -\frac{1}{r} T_* \qquad (12).$$

Applying Equations (11) and (12) to the system described in Equations (1), (2), (4), (5), (6), and (7), with the simplifications from Equations (8), (9), and (10), we obtain the following system:

$$\begin{gathered} m_1 \ddot{x}_1 = -k_1(x_1 - u) + \frac{k_2}{e}(x_2 - x_1) - c_1 \dot{x}_1 \\ m_2 \ddot{x}_2 = -k_2(x_2 - x_1) - c_2 \dot{x}_2 \end{gathered} \tag{13}$$

with these additional relationships resulting from the transformation to linear coordinates:

$$\begin{gathered} m_1 = \frac{J}{r^2} \\ k_1 = \frac{T_{max} n_r}{r^2} \\ c_1 = \frac{c_1'}{r^2} \end{gathered} \qquad (14).$$

The model can be stated in state space form, with state vector $\mathbf{q} = [x_1 \quad \dot{x}_1 \quad x_2 \quad \dot{x}_2]^T$, where the input, u, is the magnetic equilibrium position as commanded by the motor driver output current, and the output, y, is the position of the extruder, x_2:

$$\dot{\mathbf{q}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_1+k_2}{m_1} & -\frac{c_1}{m_1} & \frac{k_2}{m_1} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k_2}{m_2} & 0 & -\frac{k_2}{m_2} & -\frac{c_2}{m_2} \end{bmatrix} \mathbf{q} + \begin{bmatrix} 0 \\ \frac{k_1}{m_1} \\ 0 \\ 0 \end{bmatrix} u \tag{15}$$

$$y = [0 \quad 0 \quad 1 \quad 0]\mathbf{q}$$

13.3. Experimental Model

In addition to theoretical first-principles modelling, a model can be constructed using statistical techniques to fit experimental data to a model structure. In this approach, a model structure is selected by choosing a number of poles and zeros, and a least squares fit estimates the coefficients of the transfer function that best models the data. This functionality is implemented in the Matlab System Identification Toolbox [43], and was used to fit experimental data to models of various orders for use both in controller design and analytic model validation. This section details the results of best fits for models of the form described in the previous section.

The linearized analytical model, as shown in Equation (15), can be converted to a transfer function in the Laplace domain, assuming the belt efficiency, e, is unity:

$$G(s) = \frac{\frac{k_1 k_2}{m_1 m_2}}{s^4 + \frac{c_1 m_2 + c_2 m_1}{m_1 m_2} s^3 + \frac{c_1 c_2 + k_1 m_2 + k_2 m_1 + k_2 m_2}{m_1 m_2} s^2 + \frac{c_1 k_2 + c_2 k_1 + c_2 k_2}{m_1 m_2} s + \frac{k_1 k_2}{m_1 m_2}}$$

(16).

While we could attempt the difficult task of estimating appropriate values for each of the model parameters (m_1, m_2, k_1, k_2, c_1, and c_2) from knowledge of the physical system, we instead retain just the structure of the model and fit the coefficients of each s term using experimental data. Our model structure is thus,

$$G(s) = \frac{\phi_0}{s^4 + \psi_3 s^3 + \psi_2 s^2 + \psi_1 s + \psi_0} \qquad (17).$$

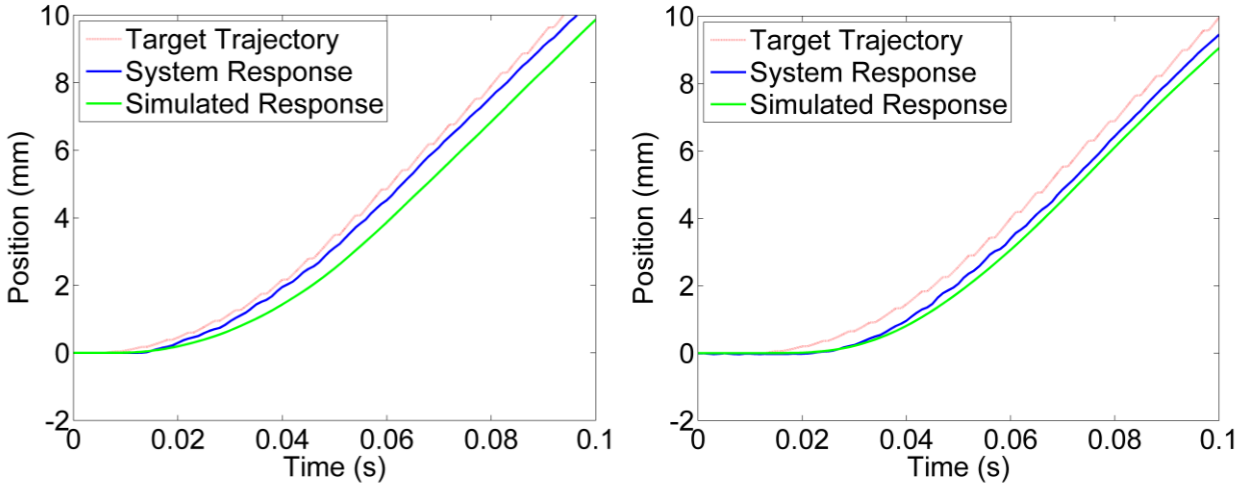
where the ϕ and ψ coefficients represent unknown quantities to be obtained. Note that our analytical model indicates ϕ_0 should be equal to ψ_0.

Data was gathered from the system's X and Y axes with the controllers running under open loop mode and the input path consisting of a series of small, random, instantaneous moves designed to stay well within the maximum velocity limits of the motor. This data was fit to a variety of model structures using the Matlab System Identification Toolbox. Although not exhaustive, a survey of possible models indicates that a no-zero, four-pole model like that shown in Equation (17) best fits the data. The transfer functions produced are

$$\begin{aligned} G_X(s) &= \frac{6.69158 \times 10^{11}}{s^4 + 16\,380.8\, s^3 + 7.82543 \times 10^6\, s^2 + 4.82342 \times 10^9\, s + 6.65380 \times 10^{11}} \\ G_Y(s) &= \frac{5.44776 \times 10^{10}}{s^4 + 3220.54\, s^3 + 3.38821 \times 10^6\, s^2 + 2.70238 \times 10^8\, s + 5.56597 \times 10^{10}} \qquad (18). \end{aligned}$$

Our model predicts that ϕ_0 should be equal to ψ_0, and it is clear from Equation (18) that the numerator and constant denominator terms are relatively close in both cases (X is the same to within 1% and Y varies by about 2%).

Because the simplification of the stepper motor model makes it unrealistic in responding to step responses, the model is evaluated against a target trajectory consisting of a smooth linear velocity ramp. Figure 16 shows a comparison of the actual and model-predicted system output for the X and Y axes. Both models have a delay greater than the actual system response, with the X axis lagging the actual system much more than the Y axis. The reason for this is likely bound up in the nonlinearities that were simplified to produce the model used for fitting. For example, the data used to generate the model was taken close to the $x_0 = 0$ end of each axis, while the test ramp data comes from the middle of each movement space. The test ramp pulls the carriage in the positive direction in both plots, which should place a higher spring constant between motor and load and result in a proportionate decrease in lag. Note that the jaggedness of the target trajectory is caused by float round off in the calculations and remains an unfixed bug.



(a) (b)

Figure 16. Response of model and system to the beginning of a trapezoidal velocity move on the X axis(a) and Y axis (b).

We can also use the fit model to estimate the unknown parameters of the analytic model. The analytic model shown in Equation (16) has four unique coefficients (ψ_0 thru ψ_4), and we have six unknown parameters. Fortunately, estimates of parameters for the motor can be made. The actual motors on the printer contain a special part number for which no information is publically available. However, a similar motor from Moons' Industries, the 17HD0013 [44], reports motor characteristics shown in Table 2.

Table 2. Known or measured system characteristics

Characteristic	Value	Notes
Number of Rotor Teeth, n_r(1)	50	Appropriate value for a 1.8° hybrid stepper, according to [16]
Rated Current(1)	0.4 A	
Holding Torque, T_{max}(1)	0.285 Nm	Further derated based on actual current used for motor drivers.
Rotor Inertia, J(1)	38 g-cm²	
X Carriage Mass, m_{2y}	1.07 kg	Measured
Extruder Mass, m_{2x}	0.45 kg	Measured
Pulley Radius, r	5.40 mm	Effective radius computed using linear encoder and stepper motor

(1) Quantity derived from Moons' Industries datasheet for the 17HD0013 [44], a similar motor to that used on the actual printer.

Using these estimates for the motor characteristics, we can compute three of our parameters, as shown in Table 3. Since the drivers were configured to run at roughly one quarter (Y axis) and one half (X axis) of the rated current, the T_{max} value used is proportionately reduced for each.

Table 3. Derived system parameters

Parameter	Formula	X Axis	Y Axis
Motor Inertia Mass Equivalent, m_1	$m_1 = \frac{J}{r^2}$	0.130 kg	0.130 kg
Load Mass, m_2	Measured	0.45 kg	1.07 kg
Spring Constant k_1	$k_1 = \frac{T_{max} n_r}{r^2}$	122 kN/m	244 kN/m

Of these, the masses are more likely to be accurate, since the maximum motor torque can change substantially between manufacturers and models, while motor mass is relatively constant. Using m_1 and m_2 from Table 3 allows us to match coefficients of the denominators of Equations (16) and (18) gives the results shown in Table 4. Comparing the derived and estimated values for k_1, it is encouraging to note that the values are of the same magnitude. The significant differences between the two axes in the belt spring constant, k_2, and in the motor damping, c_1, even though it is expected they should be similar, indicates the models do not fully and accurately reflect the system dynamics. The damping caused by the linear bearings, c_2, is roughly the same between the two axes, as it should be.

Table 4. Least-squares estimated system parameters

Parameter	Estimated Value X	Estimated Value Y
Motor Spring Constant, k_1	355 kN/m	392 kN/m
Belt Spring Constant, k_2	110 kN/m	19.7 kN/m
Motor Damping Constant, c_1	2100 Ns/m	411 Ns/m
Bearing Damping Constant, c_2	112 Ns/m	71.5 Ns/m

The model developed and evaluated in this appendix involves highly nonlinear phenomena, which require significant assumptions to reduce to a linear model. The linearized model was fit to data collected from the X and Y axes, and the results show reasonable correlations with the analytic model, although several discrepancies remain unresolved. Experimental testing of motor and belt parameters, as well as the use of a more complex system model, could help to reconcile these differences.