Spring 11

Computer Numerical Control Machine Project Proposal

Anh Luong	luong@eng.utah.edu
Willis Lutz	lbzinuse@gmail.com
Jared Pringle	jaredpringle86@gmail.com
Ashton Snelgrove	snelgrov@eng.utah.edu

Website:
http://www.eng.utah.edu/~luong/Anh_Luong/Team_Teal/Team_Teal.html

Scrum:
https://teamteal.scrumd.com/

Table of Contents

1. Introduction

This proposal outlines the planned construction of a three-axis Computer Numeric Control (CNC) machine, for the purpose of rendering two dimensional vector graphics. The CNC machine and control software will take vector image input in the form of Scalable Vector Graphics (SVG) files, and render the image onto a medium. The medium will be a flat surface, such as conventional paper, white board, or light reactive surface. The machine will be able to move on three axes and will (if design time permits) be capable of drawing with multiple instruments, i.e. pencil, laser, etc. The machine should meet the goals of balancing high precision and speed, use-limited resources and as many recycled parts as possible, and be reproducible by a hobbyist.

2. Project Description

2.1 Software

2.1.1 SVG

Scalable Vector Graphics (SVG) is a World Wide Web Consortium (W3C) standard for describing two dimensional vector image files. Vector images consist of shapes, line vectors and style information instead of the arrays of pixels available in raster images like JPEG or PNG. SVG files are ASCII text documents in XML format, and can be manipulated with a drawing program (such as the open source editor Inkscape) or with a text editor as plain text. The open standard and XML format allows the files to be read in as text and then parsed into a usable data structure by our image conversion software.

2.1.2 G-code

G-code is a industry standard for a machine control instruction set, specified in several international standards including RS274D and ISO 6983. G-code files are ASCII text files, consisting of a sequence of command codes. Each command code is, in general, a single alphabetical character followed by numeric parameters. While standards exist, many proprietary extensions and modifications are introduced by manufacturers for their specific machines. The G-code produced by our machine will conform to the standard expected by EMC2.

2.1.3 Algorithm

The image conversion software will take SVG image file on standard input and output G-code on standard input. The Python programming language has been chosen to write the software, due to ease of use, ubiquity, and integration as a scripting language in third party applications such as Inkscape and EMC. Our algorithm for conversion will consist of several conversion stages, each stage providing a transformation towards G-code.

The first step will be to de-sugar the SVG file. SVG is a complex format, allowing advanced constructs like embedded bitmap images, text and font effects, and stroke and fill styles. To make the rendering

feasible, all the extraneous elements will be removed or simplified to basic paths. Some of these simplifications, such as text to path conversion of text or polygon to path conversion, may be performed as scripted actions inside the Inkscape editor. This step may consist of several independent reductions.

Once the SVG file has been simplified, the resulting XML will be parsed into a data structure consisting of a set of paths. A series of object classes will be defined to provide a convenient storage medium for the variety of paths descriptions used (e.g. Bezier curves, simple Cartesian lines).

Next, the paths will be ordered into a render priority queue. One simple algorithm would be to define a starting location for the write head, locate the path closest in distance to this location, and add it to the render queue. Calculate the location of the write head at the end of this path, and add the next nearest path from that location. Repeat this process until there are no line segments. This greedy algorithm should provide good performance by limiting movement between paths. Drawing closely located elements would also provide an aesthetically pleasing rendering process, as the write head will not simply scan down the page, but instead render the drawing in a more varied way.

Finally, the priority queue will be used to generate the actual G-code paths. Each path will generate a pen down command and a sequence of movement instructions, followed by a pen up command a move instruction to the starting location of the next path. G-code supports a variety of control codes, and will be better understood once we have written a series of test G-code scripts. The completed G-code instruction file will be output on standard input.

2.1.4 Enhanced Machine Control

The G-code instructions would then be sent to Enhanced Machine Control (EMC), an open-source machine control suite. EMC provides a G-code interpreter and a low level hardware abstraction layer. The hardware abstraction layer emits simple controls signals for motor steps and direction, and allows the configuration of signal layout to be sent the CNC machine over a serial or parallel connection.

2.1.5 GUI

We plan on implementing a GUI interface for the user of the CNC machine which will interface with the EMC2 open source software. This GUI will enable the user to upload an SVG image from file into our algorithm where it will be converted to G-code and sent into the EMC2 software which will send commands to the microcontroller. The GUI will feature an easy interface that will allow the user to view data retrieved from the EMC2 software such as current X, Y, and Z positions. If we have time and are able to, we would also like to implement an interface to Inkscape or some similar SVG image drawing program so that the user could draw an image in Inkscape and then push a button which would send their drawing into our GUI so that the CNC machine could reproduce it.

2.2 Hardware

2.2.1 Microcontroller

The Atmel AVR family Atmega328p has been selected as the micro-controller for this project. This micro-controller provides a variety of features needed for the project, while being cheap and readily available. The controller has 1Kb of RAM and 32Kb of program flash, 23 general purpose IO lines, as well as a variety of hardware features such as in-system programming and RS-232 and SPI serial communications. The AVR family is well supported by the gcc-avr project and the Eclipse C development environment.

The hardware abstraction layer of the EMC software will send signals over either serial or parallel connection from the host computer to the micro-controller. Serial communication over RS-232 is built into the controller, while parallel communications are supported over general IO pins.

The controller will interpret the control signals from the computer and send control signals to each of the three motor drivers. Each motor will require three output control lines: enable, drive, and direction. Each position feedback system requires two input control lines for the quadrature encoded signal. The total number of control lines is fifteen, leaving eight remaining IO pins available for parallel or serial communications.

The EMC software supports positional feedback from the controlled machine, so the option exists to pass the signals directly to the control software and allow the control software to make adjustments. The other option also exists to do any positional compensation internally on the micro-controller.

2.2.2 Analog Circuit

An analog circuit allows for control of one stepper motor. The circuit is essentially the stepper motor driver. The circuitry that we will use will be based around the L297 and L298 stepper driver combo. The L297 takes the signals from the microcontroller and translate them into stepping signals to send to the L298. The L298 is the actual driver of the stepper motor. The L298 provides a capacity of 2A of current per coil. The L297 is also great for its ability to sense the amount of current flowing through the coils and will chop the signal to the L298 chip so that the average current flow is more desirable. This allows custom current to fix the motor. A couple of different circuit configurations are shown in Figures 5, 6 in Appendix A.

We also found another solution which is a SOIC package that would require surface soldering. Two chips that we found and could use are the Allegro A3967 or Allegro A3982, which are stepper motor drivers with translators all in one chip. The sample schematic of the Allegro A3982 is shown in Figure 7 in Appendix A. The advantage is having only one chip to solder instead of the two chips. It also has better DMOS technology, and no extra heat sink or cooling solution is required. We don't have to create a diode array because there are built-in diodes and synchronous rectification. L297 and L298 chips are rare to find, and the L297 is extremely expensive and low on sources, L298 even though replacement

chips are available, the chip is still extremely expensive (see Bill of Materials for pricing). Another benefit of the Allegro chips is that they are smaller in size. One down side is it is difficult to solder and chip replacement in case of burn out or defects would be expensive.

2.2.3 Stepper Motors

Stepper motors are more precise for a task like routing for CNC machines such as the one we are building for our project. Unlike DC motors, the stepper motors are brushless, synchronous electric motors that can divide a full rotation into a large number of steps. This allows for precise control without any feedback system depending on the size of the project and application. The stepper motors are constant power devices. As speed increases the torque decreases, so we'll have to try to find a happy medium for the need to drive our CNC machine. Stepper motors come in different types, unipolar which are easy to drive but have low torque and speed, and bipolar which are hard to drive but have high torque and high speed.

We are looking for a motor with high speed and high precision capabilities and could carry a reasonably sized object-so the bipolar would be a reasonable choice. Since we have an available transformer that allows 24V, we'll look for that with our motors. We will be using a 2A driver board so the motor should only take 2 amps of current per phase. Micro stepping with 1.8 degrees and 200 steps per revolution would be great for speed and also the precision that we wanted.

2.2.4 Positional Feedback System

The positional feedback system that we will use to track the position of all three stepper motors at any given time will come from a track ball mouse. The position of a track ball mouse is determined by spinning wheels that are attached to the actual tracking ball as the ball is moved, the wheel attached to the ball spins. This wheel is situated between a light emitting diode and a light sensor. The wheel itself has slots cut into it around the circumference of the wheel at regular intervals so that when the wheel spins, the led light is picked up by the light sensor. This information can then be used to determine how much the wheel has spun. We will attach this feedback system to our spinning stepper motor shaft via our motor housing. We can then use this data to calculate how many turns the screw attached to the motor housing has rotated. By knowing the threads per inch on the screw we can then calculate how far in a given direction the sled attached to the screw has moved.

2.2.5 Motor Housing

The motor housing will be very basic design with a plate that has holes for mounting a stepper motor to the plate. There will be a hole that will allow the shaft of the stepper motor to poke through to the other side of the plate and move freely. Beside the stepper motor, there will be a gear mounted to a screw on one side and a shaft on the other side. The shaft will be inserted into a hole in the plate next to the stepper motor shaft which we will also mount a gear onto. Attached to another shaft next to the screw gear will be our positional feedback system which will spin as the screw gear spins. We are also planning for the possibility of needing more intermediary gears between the stepper motor and the screw gear if necessary to reduce torque by gearing down to the screw gear, or gearing up to the screw gear to increase the speed at which the screw rotates. We will base our gearing around the abilities of

the stepper motor we decide to use. Each axis (X, Y, and Z) will have a motor housing unit.

2.2.6 Frame and Screws

The CNC machine is created around the ability to draw on a canvas. This canvas is 12 inches by 12 inches. We needed the machine to have tolerances that meet and exceed 12 inches on each side so it would be able to travel 15 inches in each direction, giving it 1.5 extra inches of room on each side.

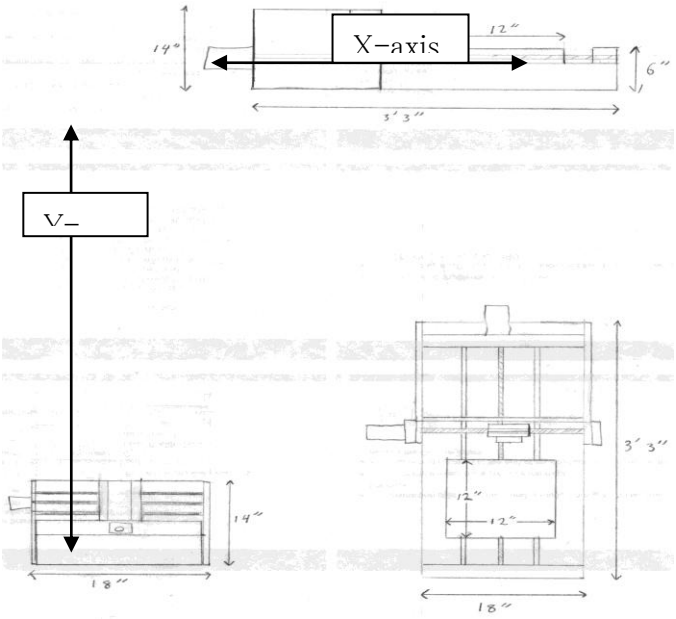


Figure 1: Top View

The sled is shown in Fig. 1 and is a 12 x 12 inch plate. It moves along the Y axis, secured in place with two rails. The rails are 36 inches long allowing the sled to travel smoothly. The sled needs to be light as possible so it will be constructed out of Plexiglas, allowing as little torque to turn the threaded rod that move the canvas up and down. A side profile shown in fig. 2 shows a better view of the sled movement in the Y-axis.

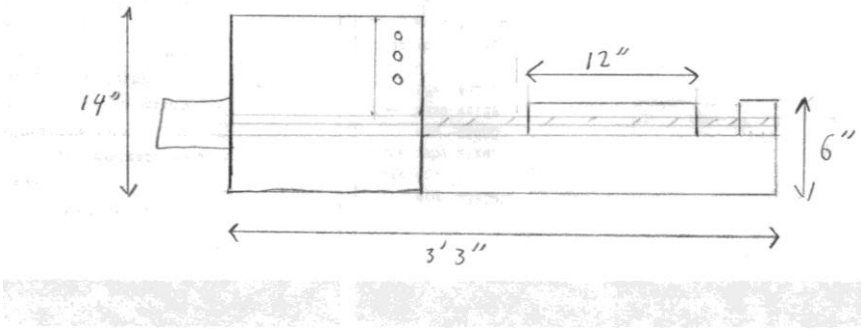


Figure 2: side view

The movement of the writing instrument will be attached to a block. This block will be able to move along the X-axis shown in fig. 1 by a threaded rod. This block is secured to two rails that allow it to glide from side to side approximately 16 inches. A frontal view of the frame, shown in figure 3, shows the block in the middle with the two rails and the threaded rod in the middle. Because our writing instrument is only a pencil or pen, we can bring the instrument close to the canvas and lift it only a few millimeters off of the canvas. This can be done with a small motor or actuator.

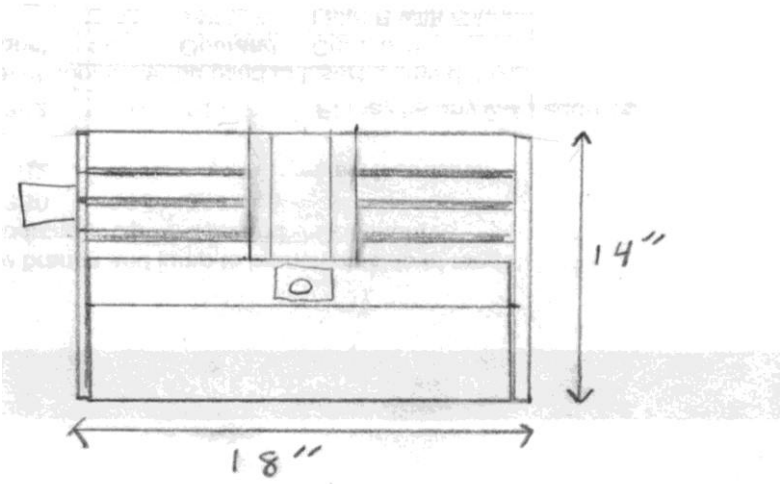


Fig.3: Front view

3. Risks

In the bill of materials, we have couple parts that are hard to find like L297 and L298. We could find some replacement like L297/1 for L297, for the L298 there are two options: L298N or L298P, these are a little easier to find than the L297 chip but still very rare.

We might have to find more, motors on top of the three we minimally need, in case of emergency and to have as replacements in case of failure. Salvaged motors tend to not have data sheets which will make it difficult for us to determine which motors will meet the needs of our project. We need to find at least two salvaged motors with the same specs for the X and Y directions of our frame.

The analog circuitry is quite confusing, and there are many different configurations to choose from. We just have to understand enough to make the circuit work for our own needs. Duplicating a working circuit is easy but we will have to figure out our analog circuit prototype first.

The structure tolerances in the final motor housing and the precision of the screw rods and motors may not be high enough forcing us to buy more expensive parts. This could up materials costs to less manageable levels than we are planning on spending.

The timings of motor controls might not be synchronized right between the 3 axes. Also, micro-controllers are quite powerful, but the complexity of the task may be too demanding of the limited resources that the micro-controller has available to it. Low level C programming also introduces significant challenges to programmers, from which mistakes could easily arise.

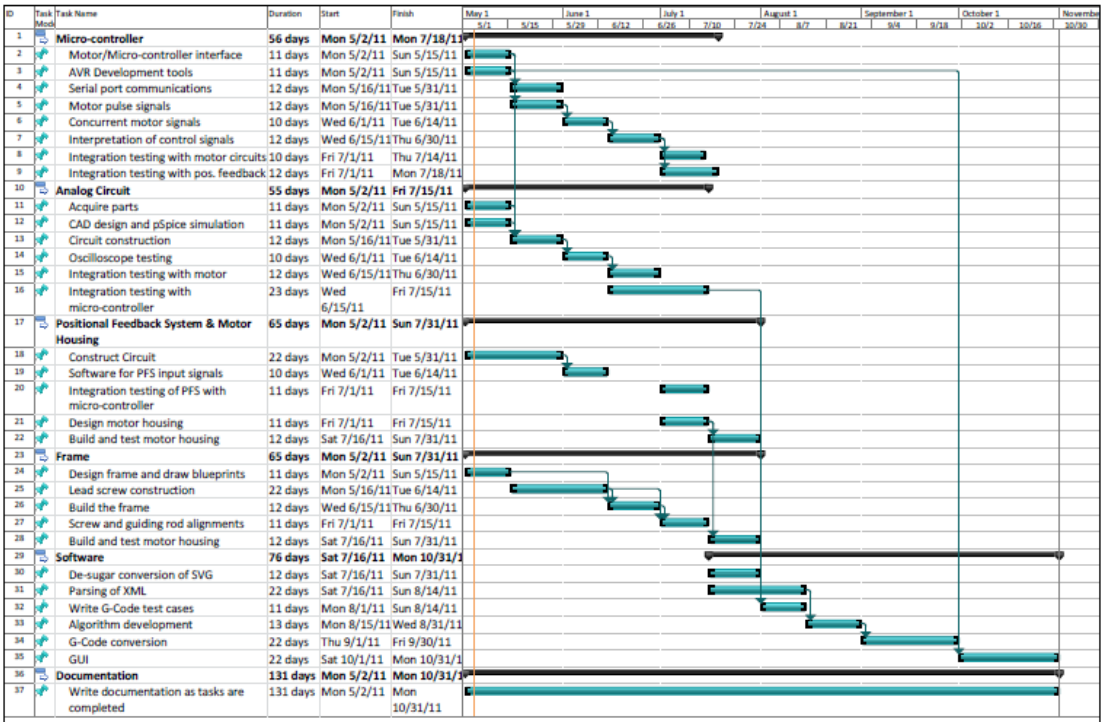
The algorithm for the image conversion software is complex, and we have to figure out how to create our own. Since we have coding experience, developing the algorithm will be the challenge at this step.

Integration has also proved to be a challenge in past work and we have to make sure we have everything working perfectly together. Minor or major adjustments will cause delays and could hinder project progress.

Multiple communicating parts allow for good modularization, allowing testing in isolation but causing increased complexity during integration.

4. Milestones

4.1 Schedule



ID	Task Mode	Task Name	Duration	Start	Finish
1		**Micro-controller**	**56 days**	**Mon 5/2/11**	**Mon 7/18/11**
2		Motor/Micro-controller interface	11 days	Mon 5/2/11	Sun 5/15/11
3		AVR Development tools	11 days	Mon 5/2/11	Sun 5/15/11
4		Serial port communications	12 days	Mon 5/16/11	Tue 5/31/11
5		Motor pulse signals	12 days	Mon 5/16/11	Tue 5/31/11
6		Concurrent motor signals	10 days	Wed 6/1/11	Tue 6/14/11
7		Interpretation of control signals	12 days	Wed 6/15/11	Thu 6/30/11
8		Integration testing with motor circuits	10 days	Fri 7/1/11	Thu 7/14/11
9		Integration testing with pos. feedback	12 days	Fri 7/1/11	Mon 7/18/11
10		**Analog Circuit**	**55 days**	**Mon 5/2/11**	**Fri 7/15/11**
11		Acquire parts	11 days	Mon 5/2/11	Sun 5/15/11
12		CAD design and pSpice simulation	11 days	Mon 5/2/11	Sun 5/15/11
13		Circuit construction	12 days	Mon 5/16/11	Tue 5/31/11
14		Oscilloscope testing	10 days	Wed 6/1/11	Tue 6/14/11
15		Integration testing with motor	12 days	Wed 6/15/11	Thu 6/30/11
16		Integration testing with micro-controller	23 days	Wed 6/15/11	Fri 7/15/11
17		**Positional Feedback System & Motor Housing**	**65 days**	**Mon 5/2/11**	**Sun 7/31/11**
18		Construct Circuit	22 days	Mon 5/2/11	Tue 5/31/11
19		Software for PFS input signals	10 days	Wed 6/1/11	Tue 6/14/11
20		Integration testing of PFS with micro-controller	11 days	Fri 7/1/11	Fri 7/15/11
21		Design motor housing	11 days	Fri 7/1/11	Fri 7/15/11
22		Build and test motor housing	12 days	Sat 7/16/11	Sun 7/31/11
23		**Frame**	**65 days**	**Mon 5/2/11**	**Sun 7/31/11**
24		Design frame and draw blueprints	11 days	Mon 5/2/11	Sun 5/15/11
25		Lead screw construction	22 days	Mon 5/16/11	Tue 6/14/11
26		Build the frame	12 days	Wed 6/15/11	Thu 6/30/11
27		Screw and guiding rod alignments	11 days	Fri 7/1/11	Fri 7/15/11
28		Build and test motor housing	12 days	Sat 7/16/11	Sun 7/31/11
29		**Software**	**76 days**	**Sat 7/16/11**	**Mon 10/31/1**
30		De-sugar conversion of SVG	12 days	Sat 7/16/11	Sun 7/31/11
31		Parsing of XML	22 days	Sat 7/16/11	Sun 8/14/11
32		Write G-Code test cases	11 days	Mon 8/1/11	Sun 8/14/11
33		Algorithm development	13 days	Mon 8/15/11	Wed 8/31/11
34		G-Code conversion	22 days	Thu 9/1/11	Fri 9/30/11
35		GUI	22 days	Sat 10/1/11	Mon 10/31/1
36		**Documentation**	**131 days**	**Mon 5/2/11**	**Mon 10/31/1**
37		Write documentation as tasks are completed	131 days	Mon 5/2/11	Mon 10/31/11

Fig. 4: Project Gantt chart

A larger view of the project Gantt chart can be found in Appendix B.

4.2 Task Descriptions

4.2.1 Micro-controller

5/2 - 5/15 **Motor/Micro-controller interface**
Define the pin interface between analog motor driver circuit and micro-controller.
Produce a document describing the interface

5/2 - 5/15 **AVR Development tools**
IDE & SDK install for Atmega AVR development: Eclipse CDT, gcc-avr, avrdude.
Demonstrate programming of controller with demo code.

5/16 - 5/31 **Serial port communications**
Write software for transmit/receive over RS-232 and wire up level converter and controller.
Prereq: AVR Development tools
Demonstrate echo back of signals sent from a terminal.

5/16 - 5/31 **Motor pulse signals**
Write software to periodically generate motors signals
Demonstrate signals being correctly output on an oscilloscope.
Prereq: Motor/Micro-controller interface

6/1 - 6/14 **Concurrent motor signals**
Generate multiple motor driver signals concurrently.
Demonstrate signals correctly output on an oscilloscope.
Prereq: Motor pulse signals.

6/15 - 6/31 **Interpretation of control signals**
Install EMC on test machine.
Write software to interpret serial signals sent from EMC and produce motor driver signals.
Demonstrate signal output on oscilloscope for one and three motors using manual control signal input over terminal.
Prereq: Concurrent motor signals

7/1 - 7/14 **Integration testing with motor circuits**
Wire the motor driver circuit to the controller.
Test with digital control signals from micro-controller, drive the motor through analog circuit.
Demonstrate control signals from controller driving the motor.
Prereq: Interpretation of control signals, Motor Circuit Testing

7/15 - 7/31 **Integration testing with positional feedback**
Integrate the positional feedback into the motor control software.
Demonstrate the system attempting to compensate movement when PFS provides missing signals and normal operation.
Prereq: Software for PFS input signals, Interpretation of control signals

4.2.2 Analog Circuit

5/2 - 5/15 **Acquire parts**
Purchase the L297/L298 (get from DSL Lab)

5/2 - 5/15 **CAD design and pSpice simulation**
Demonstrate pSpice simulation and the circuit diagrams.

5/16 - 5/31 **Circuit construction**
Demonstrate a complete circuit board.
Prereq: Motor/Micro-controller interface
Prereq: Acquire parts, CAD design of circuit and pSpice simulation.

6/1 - 6/14 **Oscilloscope testing**
Provide input to the circuit manually using push-button switches.
Demonstrate control signal output on oscilloscope - output voltage and current should match simulation.
Prereq: Circuit construction.

6/15 - 6/30 **Integration testing with motor**
Provide input to the circuit using push-button switches.
Demonstrate motor movement
Prereq: Oscilloscope Testing

6/15 - 7/15 **Integration testing with micro-controller**
Duplicate milestone from Microcontroller section.

4.2.3 Positional Feedback System and Motor Housing

5/2 - 5/31 **Construct circuit**
Obtain sensing phototransistors and photodiodes.
Wire circuit, provide power and ground connections.
Demonstrate the signal generated from the 2 output lines using oscilloscope.

6/1 - 6/14 **Software for PFS input signals**
Write micro-controller software to interpret quadrature encoded input signals.
Prereq: Construct circuit, Serial port communications.

7/1 - 7/15 **Integration testing of PFS with micro-controller**
Duplicate task from micro-controller section

7/1 - 7/15 **Design motor housing**

Draw schematic for motor house layout: position of motor shaft, mounting screws, PFS gears/shaft, gear connection to screw, and screw connector.

7/16 - 7/31 **Build and test motor housing**

Demonstrate a working motor housing - smooth motion of gears, attach to shaft.
Prereq: Design motor housing.

4.2.4 Frame

5/2 - 5/15 **Design the frame and draw blueprints**

Provide a blueprint document, complete enough to allow construction

5/16 - 6/14 **Lead screw construction**

Demonstrate the finished screws.

6/15 - 6/30 **Build the frame**

Construct the frame components and assemble.
Demonstrate the finished frame.
Prereq: Design the frame and draw blueprints

7/1 - 7/15 **Screw and guiding rod alignments**

Demonstrate movement of the sled and canvas by manually manipulating the screws.

7/16 - 7/31 **Build and test motor housing**

Duplicate of task in PFS/Motor housing

4.2.5 Software

7/16 - 7/31 **De-sugar conversion of SVG**

Write script to control Inkscape to reduce SVG file into a SVG path-only subset
Demonstrate matching a series of test cases demonstrating correct behavior.

7/16 - 8/14 **Parsing of XML**

Define storage classes and object methods.
Determine library to be used for parsing.
Demonstrate a series of test cases proving correct object creation for XML object.

8/1 - 8/14 **Write G-Code test cases**

Write manual G-code to learn the commands.
Demonstrate working G-code interpreted correctly by EMC.
Prereq: working microcontroller/motor system

8/15 - 8/31 **Algorithm development**

Write code to prioritize render ordering.
Demonstrate a correct priority queue output of path selection algorithm.
Prereq: Parsing of XML

9/1 - 9/30 **G-Code conversion**

Addition unknown conversion steps between paths and G-code, create additional task checkpoints at this date as requirements are better understood.
Demonstrate test cases from path to G-code.
Prereq: Algorithm development.

10/1 - 10/31 **GUI**

Prereq: installed EMC

5. Point Person

Task	Lead
Analog Driver Circuit	Anh
Positional Feedback System	Jared
Frame	Will
Software	Ashton

6. Bill of Materials

Item	Vendor	Quantity	Price - USD	Total Price - USD
Stepper Motors	Ashton	3	Free	Free
Micro-Controllers	Sparkfun	1	7.99	7.99
Positional Feedback System – Track Mice	DigiKey	3	2.99	8.97
Angle Iron	Willis	1	Free	Free
Rails	Home Depot	6	1.00	6.00
Acme threaded rod	Machine Shop	3	20.00	60.00
Linear Bearings	Home Depot	1	20.00	20.00
Driver Nuts	Home Depot	1	5.00	5.00
Coupling	Home Depot	1	10.00	10.00
Transformer – 24 V 10 A	Ashton	1	Free	Free
L297	Jameco/Mouser	3	8.95/11.29	26.85/33.87
L298	Jameco/Mouser	3	3.25/4.67	9.75/13.98
Discrete Components	ECE Stock Room	1	30.00	30.00

7. References

Advertisement. *Home Improvement Made Easy with New Lower Prices | Improve & Repair with The Home Depot*. Web. 25 Apr. 2011. <http://www.homedepot.com/>.

"ATmega48A/48PA/88A/88PA/168A/168PA/328/328P Datasheet." Atmel Corporation, Aug. 2010. Web. 25 Apr. 2011. <http://www.atmel.com/dyn/resources/prod_documents/doc8271.pdf>.

Closed, Soldering Sj2. "Easy Driver Stepper Motor Driver." *SchmalzHaus.com Brian Schmalz Homepage*. Web. 08 Apr. 2011. <http://www.schmalzhaus.com/EasyDriver/>.

Cnc Machine Manufacturers. Photograph. *Review about Cnc Machine*. Web. 25 Apr. 2011. <http://www.cncmachine-details.info/cnc-machine-manufacturers.html>.

"Design Fundamentals for Phototransistor Circuits." Fairchild Semiconductor, 30 Apr. 2002. Web. 25 Apr. 2011. <http://www.fairchildsemi.com/an/AN/AN-3005.pdf>.

"EMC2 Documentation." Enhanced Machine Controller Project. Web. 25 Apr. 2011. <http://www.linuxcnc.org/docview/html/>.

"Inkscape User Documentation." *Inkscape. Draw Freely.* Web. 04 May 2011. <http://inkscape.org/doc/index.php?lang=en>.

"Scalable Vector Graphics (SVG) 1.1 (Second Edition)." World Wide Web Consortium (W3C), 22 June 2010. Web. 25 Apr. 2011. <http://www.w3.org/TR/SVG11/>.

"Standard Cataloged Acme Inch Screw and Nut Quick Reference Chart - Nook Industries, Inc. PowerAc Acme Screws and Nuts." *Nook Industries : Linear Actuators for Motion Control Ball Screws, Screw Jacks, Lead Screws, Linear Slides, Acme Screw, Actuator*. 25 Apr. 2011. Web. 25 Apr. 2011. <http://www.nookindustries.com/acme/AcmeInchAvailability.cfm>.

Appendix A: Analog Circuit Designs

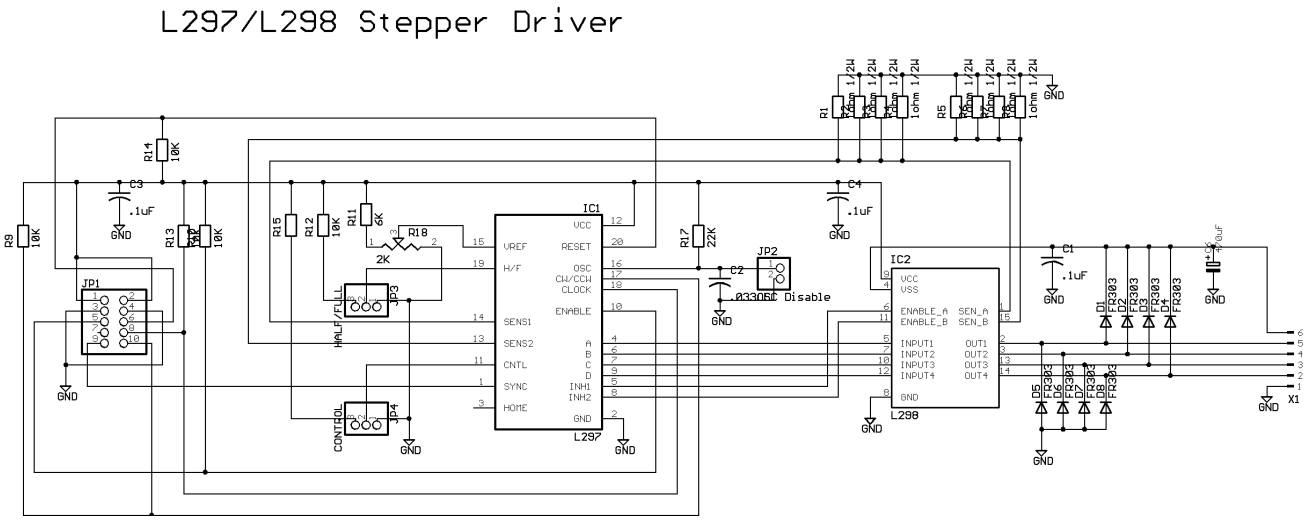


Fig. 5: Basic Stepper Motor Driver Circuit

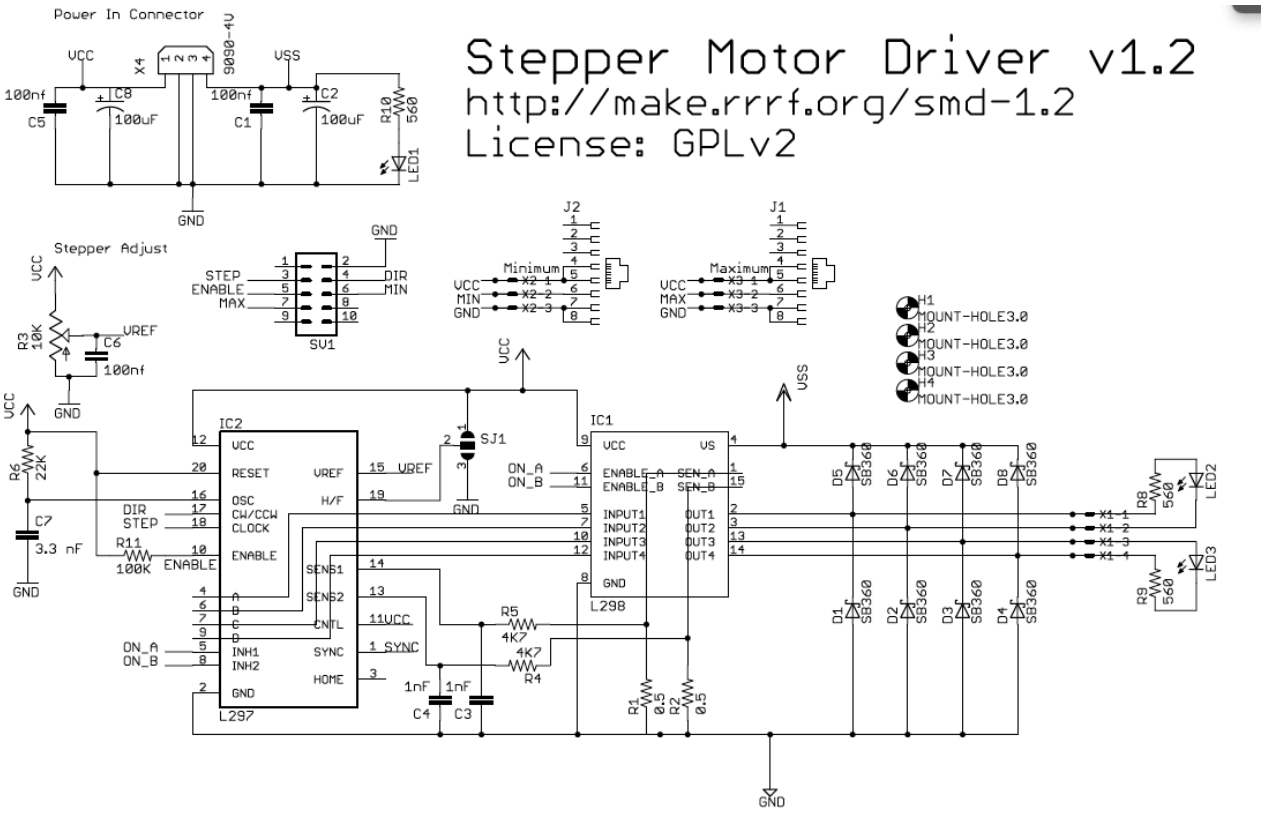


Fig. 6: Improved Stepper Motor Driver Circuit

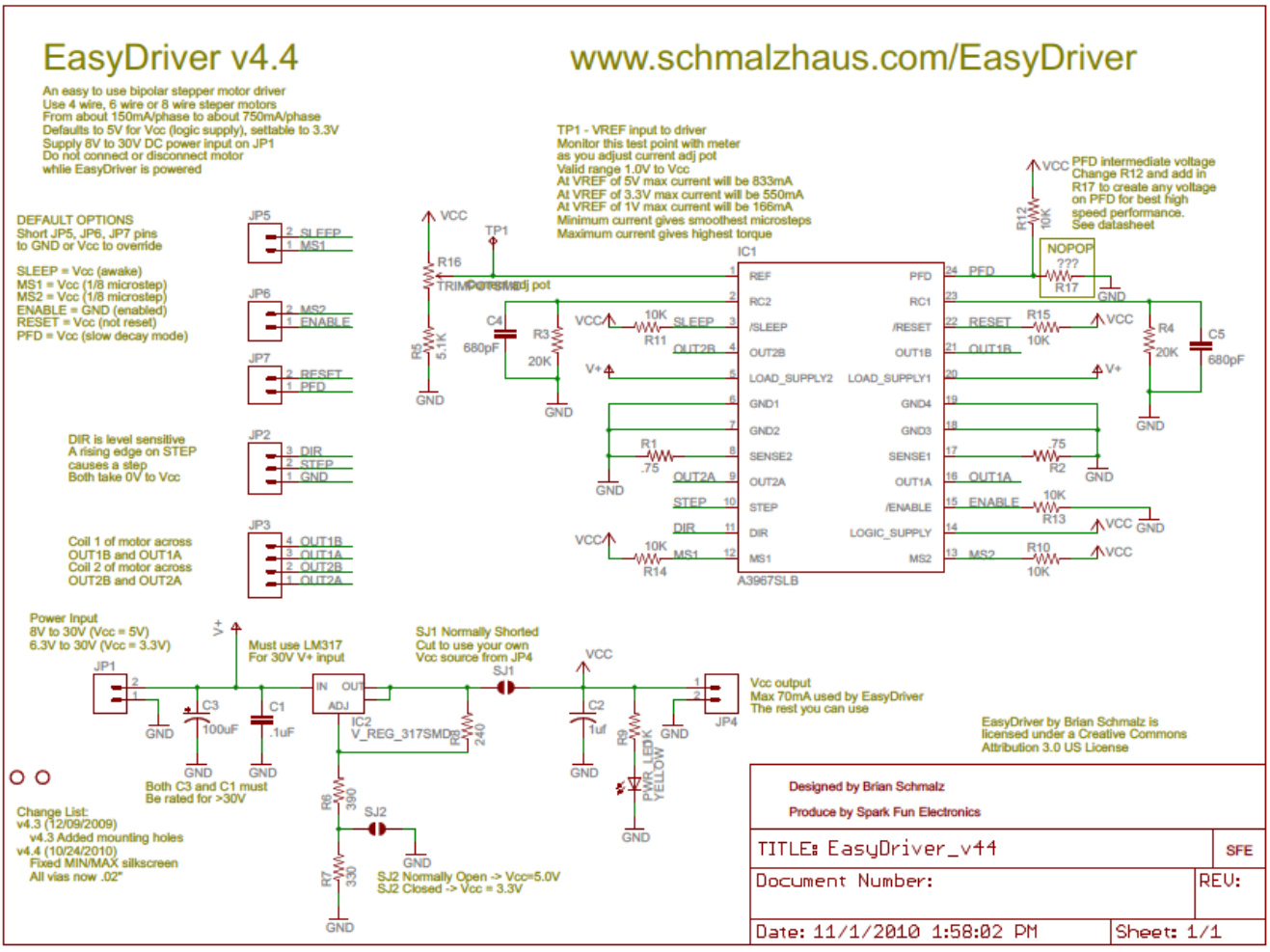
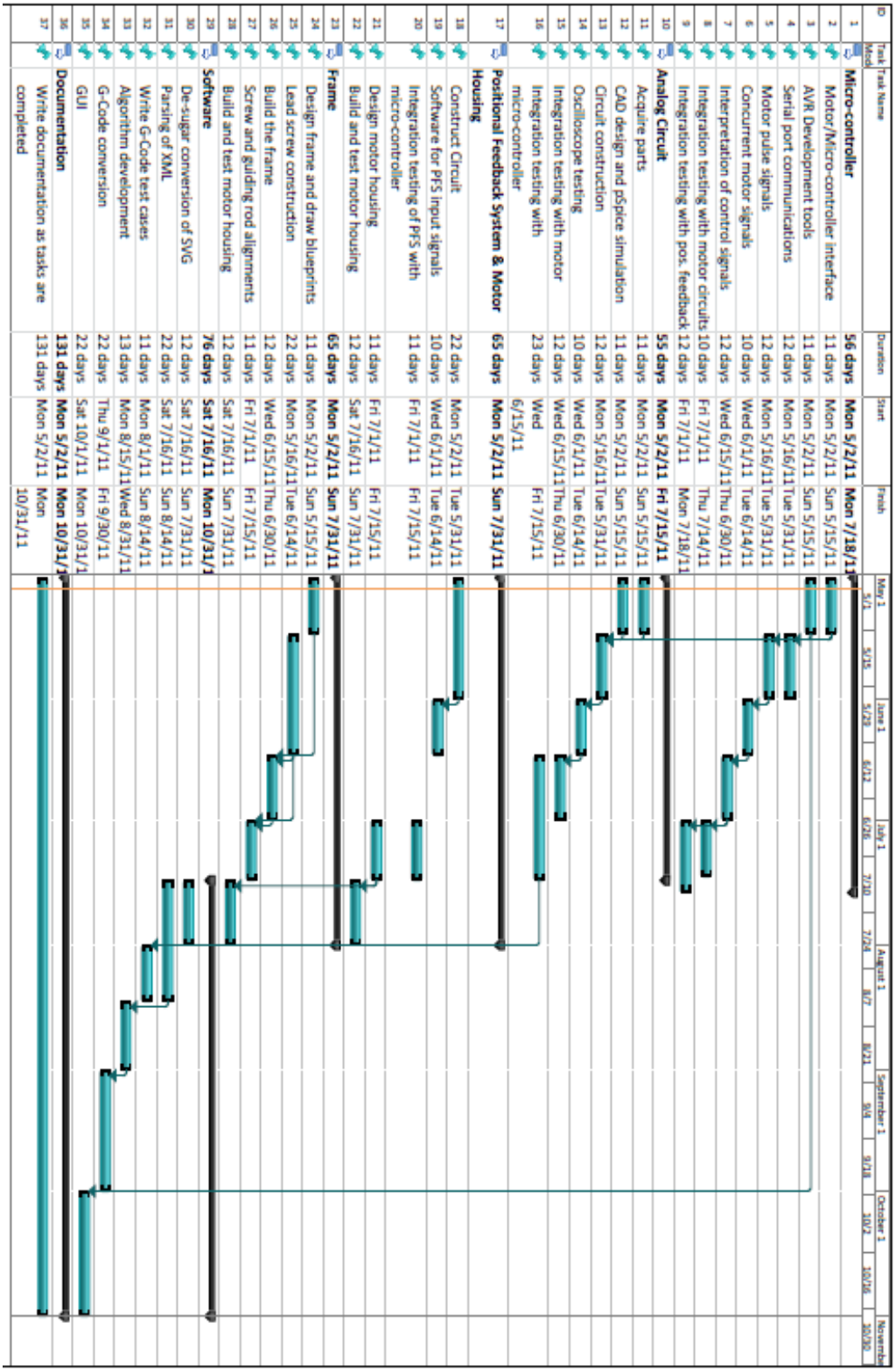


Fig. 7: Surface Mount Driver Schematic

Appendix B: Project Gantt Chart



ID	Task Mode	Task Name	Duration	Start	Finish
1		**Micro-controller**	**56 days**	**Mon 5/2/11**	**Mon 7/18/11**
2		Motor/Micro-controller interface	11 days	Mon 5/2/11	Sun 5/15/11
3		AVR Development tools	11 days	Mon 5/2/11	Sun 5/15/11
4		Serial port communications	12 days	Mon 5/16/11	Tue 5/31/11
5		Motor pulse signals	12 days	Mon 5/16/11	Tue 5/31/11
6		Concurrent motor signals	10 days	Wed 6/1/11	Tue 6/14/11
7		Interpretation of control signals	12 days	Wed 6/15/11	Thu 6/30/11
8		Integration testing with motor circuits	10 days	Fri 7/1/11	Thu 7/14/11
9		Integration testing with pos. feedback	12 days	Fri 7/1/11	Mon 7/18/11
10		**Analog Circuit**	**55 days**	**Mon 5/2/11**	**Fri 7/15/11**
11		Acquire parts	11 days	Mon 5/2/11	Sun 5/15/11
12		CAD design and pSpice simulation	11 days	Mon 5/2/11	Sun 5/15/11
13		Circuit construction	12 days	Mon 5/16/11	Tue 5/31/11
14		Oscilloscope testing	10 days	Wed 6/1/11	Tue 6/14/11
15		Integration testing with motor	12 days	Wed 6/15/11	Thu 6/30/11
16		Integration testing with micro-controller	23 days	Wed 6/15/11	Fri 7/15/11
17		**Positional Feedback System & Motor Housing**	**65 days**	**Mon 5/2/11**	**Sun 7/31/11**
18		Construct Circuit	22 days	Mon 5/2/11	Tue 5/31/11
19		Software for PFS input signals	10 days	Wed 6/1/11	Tue 6/14/11
20		Integration testing of PFS with micro-controller	11 days	Fri 7/1/11	Fri 7/15/11
21		Design motor housing	11 days	Fri 7/1/11	Fri 7/15/11
22		Build and test motor housing	12 days	Sat 7/16/11	Sun 7/31/11
23		**Frame**	**65 days**	**Mon 5/2/11**	**Sun 7/31/11**
24		Design frame and draw blueprints	11 days	Mon 5/2/11	Sun 5/15/11
25		Lead screw construction	22 days	Mon 5/16/11	Tue 6/14/11
26		Build the frame	12 days	Wed 6/15/11	Thu 6/30/11
27		Screw and guiding rod alignments	11 days	Fri 7/1/11	Fri 7/15/11
28		Build and test motor housing	12 days	Sat 7/16/11	Sun 7/31/11
29		**Software**	**76 days**	**Sat 7/16/11**	**Mon 10/31/1**
30		De-sugar conversion of SVG	12 days	Sat 7/16/11	Sun 7/31/11
31		Parsing of XML	22 days	Sat 7/16/11	Sun 8/14/11
32		Write G-Code test cases	11 days	Mon 8/1/11	Sun 8/14/11
33		Algorithm development	13 days	Mon 8/15/11	Wed 8/31/11
34		G-Code conversion	22 days	Thu 9/1/11	Fri 9/30/11
35		GUI	22 days	Sat 10/1/11	Mon 10/31/1
36		**Documentation**	**131 days**	**Mon 5/2/11**	**Mon 10/31/1**
37		Write documentation as tasks are completed	131 days	Mon 5/2/11	Mon 10/31/11