Design of a Desktop Milling Machine for Fabrication in an Introductory Machine Shop Class

by

Dan Lorenc

Sc.B. Mechanical Engineering
Massachusetts Institute of Technology, 2010

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

February 2010

Signature of Author...
[illegible]
Department of Mechanical Engineering
January 14, 2010

Certified by.......................................
[illegible]
Martin L. Culpepper
Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted by.......................................
[illegible]
John H. Lienhard V
Collins Professor of Mechanical Engineering
Chairman, Undergraduate Thesis Committee

Design of a Desktop Milling Machine for Fabrication in an Introductory Machine Shop Class

by

Dan Lorenc

Submitted to the Department of Mechanical Engineering
on January 14, 2010 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

The purpose of this research is to design, fabricate and test the electromechanical subsystem of a CNC milling machine kit. Unlike all other CNC kits on the market, the purpose of this kit is to teach students the principles of electro mechanics during its construction and to teach the principles of CNC machining during its operation. The design process and final designs of both the software and electronic subsystems are described and evaluated. The CNC kit is assembled using off-the-shelf, easy-to-use electronic elements. The software is programmed using Matlab, a common and easy to program software system purchased for all students by MIT. The outline for a course in which this kit could be taught is given, including sample code and circuit schematics.

Thesis Supervisor:	Martin L. Culpepper
Title:	Associate Professor of Mechanical Engineering

ACKNOWLEDGEMENTS

I would like to thank the entire Mechanical Engineering Faculty of MIT for the incredible opportunities I have been given. Specifically, I would like to thank Professor Brisson for teaching me that real engineers are just as comfortable with equations as they are with a set of calipers. I would like to thank Professor Sarma for teaching me that engineering is only half the battle in making a difference. Finally, I would like to thank Professor Culpepper for reminding me that a working prototype is worth all the modeling and physics in the world.

I would also like to thank my family. Without their support I wouldn't be anywhere near where I am today.

CONTENTS

FIGURES

TABLES

CHAPTER

1

INTRODUCTION

The objective of this research is to design and develop the electronic components of a desktop milling machine kit. This kit will enable students to construct a three-axis CNC milling machine capable of machining plastic and aluminum parts with a working volume of 8 cubic inches. The machine will be controlled using a cross-platform graphical user interface that is able to interpret standard G-Code.

The construction of this kit will be used to teach the principles of electronics and electro mechanics to junior and senior level mechanical engineering students. While the Mechanical Engineering Department recommends students take an introductory electronics class as early in their careers as possible[1], no such class taught in the department. Interested students must elect to take classes in the Electrical Engineering Department. While these classes are no doubt useful to electrical engineers, the subject matter is of limited relevance to students majoring in other fields. An introduction to electronics within the Mechanical Engineering Department could be tailored to the applications mechanical engineers are more likely to encounter in their careers.

Once constructed, this kit will be used to teach the principles of machining and manufacturing to mechanical engineering students. These students will benefit from having a safe and lower cost alternative to learning the operation of CNC machining. Currently students learn the CNC process in a professionally staffed production-level machine shop on machines ranging in cost from thousands to tens of thousands of dollars. Putting a low-cost safer alternative in the hands of students will provide them a safe place to learn to program CNC machines without risking thousands of dollars of damage.

This research is important because electromechanical and computer control skills are becoming increasingly important to mechanical engineering students. No longer is a purely mechanical skill-set sufficient in academia or the workplace. Students are expected to not only design mechanical systems, but to control and actuate these systems using computers and electronics. The construction and operation of this kit would better prepare students for both academia and the workplace.

1.1 Background on Elements of Machine Design

MIT Course 2.72, Elements of Machine Design, is a course taught to mechanical engineering students in their junior and senior years. Focusing on advanced modeling, design and integration of mechanical systems, students have traditionally been tasked with the design and construction of a desktop lathe, shown in Figure 1-1, over the course of a semester. The scope of 2.72 has been selected to include all major components of mechanical systems from fluid mechanics to computer simulation.

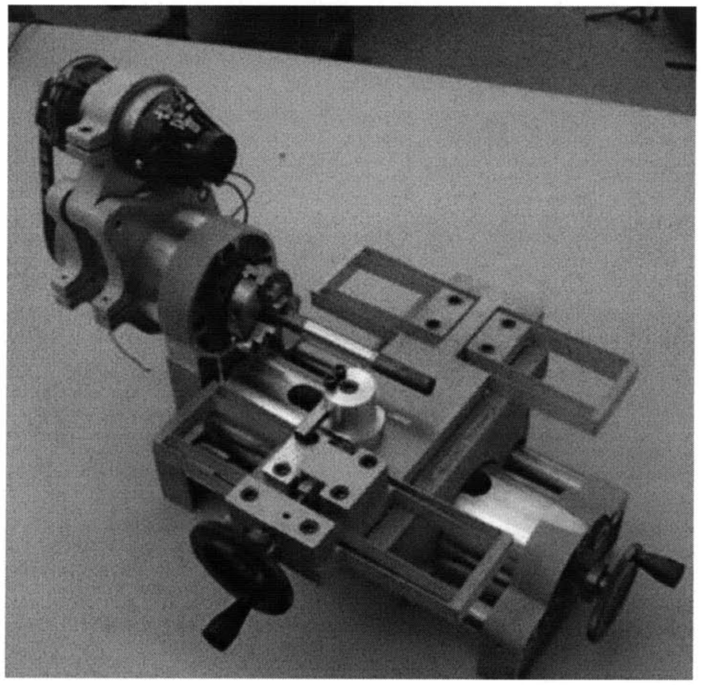


Figure 1-1: Spring 2009 2.72 Group 5's finished lathe prototype

2.72 is taught with the philosophy that both hands-on and analytical abilities are required to be a successful engineer. 2.72 was designed to expose students to the strengths and weaknesses of the principles and models taught in their earlier classes; enabling them to make better engineering decisions later in their careers.

The desktop lathe was chosen as the 2.72 project because it is able to both serve as a learning platform integrated into laboratory and design activities and to allow students to create other mechanical elements for later projects[2]. In order to keep cost down and meet a required precision of <10um, flexural elements are used as linear bearings. In addition to the required precision, the lathes must be resilient enough to fail safely in a worst-case situation: a swift plunge of the tool into a ¾" diameter steel stock.

1.2 CNC Desktop Milling Machine

While the manual lathe designed and fabricated in 2.72 has served the class well, mechanical engineering design projects are increasingly requiring electrical and electronic components. The students of 2.72 in coming years would be better served by a design project that incorporates both mechanical and electronic design. Keeping with the tradition of a project resulting in a useful end product, a CNC desktop milling machine was selected. A computer will control the milling machine so students can produce parts with paths too complex for hand operation.

The milling machine kit will be capable of machining aluminum and plastic parts using standard machine tools and G-Code. Operation will be similar to the EZ-Trak machines found in the Laboratory for Manufacturing and Productivity. By removing the actuators and replacing them with hand wheels, the machine will be able to perform both CNC milling and manual milling. The spindle speed and tool changing will be performed manually, so a less expensive version of the kit can still function with no electronics.

---

[1]MIT Course Catalog. http://web.mit.edu/catalog/degre.engin.ch2.html

[2]MIT Course 2.72 Class Website. http://pcsl.mit.edu/2_72/index.html

CHAPTER 2

DESIGN REQUIREMENTS

2.1 Educational Requirements

Since the primary goal of this kit is to serve as a learning platform, ease of use and educational value are of highest priority. Off the shelf components are only used where they can be useful elsewhere. For example, stepper motor drivers can be used off the shelf because it is likely students will have to drive stepper motors in other pursuits, but a 3-axis controller wouldn't be useful off the shelf because students aren't likely to need to fabricate another milling machine.

Open-source hardware is used where possible because it is generally supported much better across platforms and has a much lower barrier to entry. For example, Texas Instruments makes very good microcontrollers, but their compilers only work on Windows machines and the system costs several hundred dollars to get started. The Arduino microcontroller project is an open-source project with a compiler that runs on Linux, Mac and PC.

Finally, the components should be as inexpensive as possible, without sacrificing functionality and ease of use. Students will be much more likely to use $15 motors on later projects than $150 motors, all else equal. The goal of this kit is not only place electromechanical skills into the "hypothetical" toolbox for each student, but to also place these components into their "physical" toolbox.

2.1.1 Electromechanical Requirements

From the general education requirements, the formal design requirements and parameters for the electromechanical system were selected. These are shown in Table 2-1.

The system is made easy to learn by selecting "Black box" hardware elements. These elements represent the most generic elements of their class. They can take a lot of abuse and don't suffer from any application specific "gotchas." For example, a stepper motor driver that is capable of driving almost any motor with little configuration is a better choice than an application specific driver, even with lower performance.

Safety is an issue anytime students are learning a new technology. For this reason, the components selected must be as safe as possible. Low voltage motors and sensors that are much more difficult for students to injure themselves with are required. High torque and powerful motors can also pose a problem to inexperienced students. Hair and loose clothing can be caught in motors and gearboxes, resulting in significant injuries.

The ultimate test for a system comes when it is placed in the hands of a student. The parts will be pushed to their limits and then some. Wires will be reversed, voltages doubled and motors stalled. Fragile components are not acceptable. Circuit boards should be robust enough to withstand multiple voltages. Motors should not overheat when stalled. Sensors should not fail with the wrong inputs. In addition to selecting the most reliable components as possible, failure can be reduced by making the system as modular as possible so small problems don't result in systemic failures.

With a learning system, things will go wrong. Students will lose parts, circuits will get fried and motors will get burnt out. The electromechanical components are especially susceptible to failure from misuse. In order to account for this loss of parts, the components must be as inexpensive as possible. The learning process is often fraught with failure. If the system is too expensive to allow students to fail, the educational value is reduced. Leveraging existing technologies can be an effective method of reducing cost. For example, off-the-shelf limit switches are often much cheaper than custom designed limit switches.

Table 2-1: Educational requirements of electromechanical system

Requirements	**Design Parameters**
Easy to learn	"Black box" hardware
Safe	Low voltage, low torque
Robust	Modular design, reliable components
Inexpensive	Leverage existing technologies

2.1.2 Software Requirements

The electromechanical system is only half of the milling machine kit. The expensive motors and CNC capability of the machine is useless without a method of programming and controlling the machine. Most CNC machines use a separate desktop or laptop computer to run control software. This software can be used to run preprogrammed machine code or to create and run new programs. Machine parameters such as positions, speeds and tools can be monitored and set through the software interface. The formal requirements and design parameters of the software system are shown in Table 2-2.

The requirements of the CNC control software for this mill kit are similar to the requirements for production CNC software, however the priorities are different. Production software is optimized to give the machinist the maximum amount of power over the machine code as possible. This power often comes at the expense of ease of use and learnability. The interfaces can be confusing to beginners and problems are hard to debug. The software for this CNC mill needs to be as easy to use as possible, even if it means some powerful features are missing.

While making the software easy to use, it must still be as realistic as possible. Because G-Code is run on nearly every CNC machine a student is likely to encounter, the mill kit should run G-Code as well, even though it is notoriously difficult to use. This software should be viewed as a set of "training wheels" for using real, production quality software.

As with the electromechanical system, the software should be as inexpensive as possible. Commercial G-Code interpreters and CNC software exist, but these can cost thousands of dollars and make modifications difficult. Using expensive third-party libraries like these should be avoided and open-source or self-written code should be used instead, if possible.

Finally, the software, like the electro mechanics, should be as robust as possible. Error handling should be intelligent enough to allow an inexperienced user to operate the system. The program should make it clear to the operator where errors are from, what causes them and how to fix them. In addition, no software project is ever complete. If the mill design changes at a later date, the software needs to be able to handle this. The software should be built in as easy and well-documented manner as possible so other students can modify it later.

Table 2-2: Educational requirements of software system

Requirements	Design Parameters
User Friendly	Intuitive, cross-platform, no installation
Realistic	Full g-code interpreter
Inexpensive	No expensive third-party libraries
Robust	Intelligent error handling, well-documented code

2.2 Functional Requirements

Although the primary goal of this kit is to serve as a learning platform, the kit must also meet several functional requirements in order to be useful. The standard parameters used when designing milling machines also apply here. The machine must be able to cut certain materials at certain speeds with certain resolutions in order to produce parts. This machine will need to cut aluminum with a maximum spindle speed of 2000RPM. A resolution of .1mm is required to produce anything of value with a cutting volume of only 131cc. Finally, the machine needs to finish parts in a finite amount of time. The spindle needs to move with a maximum speed of 1000mm/min to ensure parts are finished quickly enough.

These mechanical/manufacturing constraints translate to functional requirements in the electromechanical and software systems. The manufacturing requirements of the torque and RPM of the spindle translate directly to the electromechanical requirements. The resolution of the stage motors and the pitch of the lead screw determine the final cutting resolution. The pitch of the lead screws and the stage motors also determine the speed of the spindle and the cutting forces. These functional requirements are shown in Table 2-3.

2.2.1 Electromechanical Requirements

Table 2-3: Functional requirements of electromechanical system

Requirements	Design Parameters
Torque	20 oz-in torque spindle motor
RPM	10000 volt DC spindle motor
Resolution	Lead screw pitch under 20/inch
Speed	800 RPM stage motors

Cutting Forces	8 oz-in torque stage motors

Even the most advanced electromechanical system would be useless without a well-designed user interface. The software for this project needs to serve as both a front-end, allowing the user to operate the machine in an intuitive way and a back-end, translating these user commands to machine motion. While the majority of the requirements on the software are of an educational nature, there are several function requirements, shown in Table 2-4, that affect the design decisions.

There is no predominant operating system in use by students at MIT. Any given student is just as likely to run Linux on a homemade desktop machine as he/she is to run Windows 7 on a Macbook Pro. This presents a challenge while writing software used by students: anything written has to be cross-platform.

In addition to ranging in their choice of operating system, students also range in their level of computer savvy. Using a command-line driven program might be second nature to some students while others are lost the minute they have to type a single command. This software must be able to create parts using both a GUI and prewritten G-Code to accommodate all levels of students.

Finally, the end system must be as safe as possible. The software must incorporate emergency stops that prevent the machine from damaging itself and the user.

2.2.2 Software Requirements

Table 2-4: Functional requirements of software system

Requirements	**Design Parameters**
Platform	Mac, Windows, Linux
Interface	GUI and g-code
Safety	Emergency stops

CHAPTER

3 ELECTRONIC DESIGN AND IMPLEMENTATION

3.1 Architecture

The first step in designing any electro mechanical platform is designing the architecture. The final design of the architecture of the mill kit was arrived at by following the FRDPAARC^3 design process, taught by Alex Slocum. Starting with the functional requirements, all of the electromechanical components needed, shown in Figure 3-1, were identified.

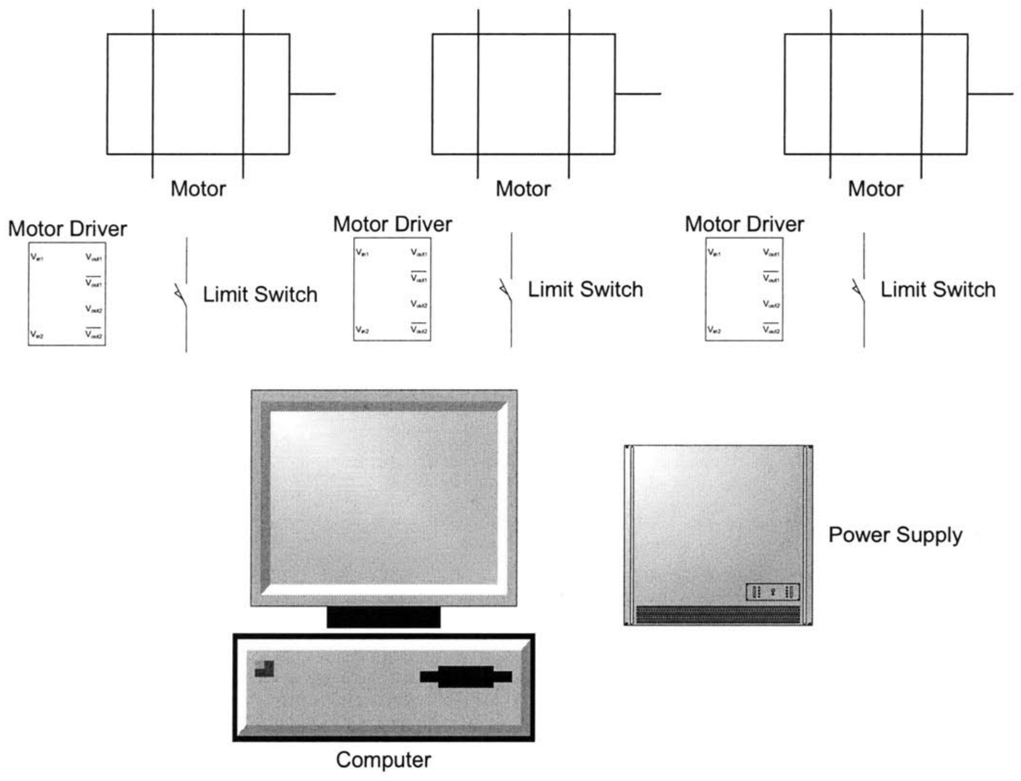


Figure 3-1: Required components for system architecture

Three motors are needed, one for each axis. Each motor needs to be driven and controlled. This can take place in one three-axis driver/controller or three separate driver/controllers. For safety and performance reasons, each axis needs a limit switch. The GUI software must run on a PC. Finally, everything must be powered by a power supply.

While these components could all be hooked directly to the computer, this would require at least 6 communication ports. Almost no laptops have six communication ports and very few desktop machines do. An ideal solution would only require one port on the host computer. This can be accomplished by adding another element to the architecture: a microcontroller. All of the electronics can plug into the microcontroller and the microcontroller can interface with the computer.

The final architecture of the mill kit, shown in Figure 3-2, supports all of the functional and educational requirements while still being as easy to understand as possible. The GUI software will run on the user's PC. The mill control software will run on a microcontroller which communicates back and forth with the PC. Each axis will be controlled by a separate motor driver/controller and limit switch. These drivers and switches will all be connected to the microcontroller and powered by a power supply.

Separating the control electronics into two halves: the microcontroller side and the PC side, forces the GUI software to be completely modular from the control software. This means multiple GUI packages can be used, all communicating with the same microcontroller, making cross-platform software easy to write. Additionally, the use of a microcontroller as well as a PC simplifies the electronics greatly. The final mill will need on the order of 30 input output pins, a target impossible to reach using a standard PC without complicated multiplexing circuitry but easily reached by modern microcontrollers.

Another benefit of this architecture is that each axis is driven independently of the rest. This makes adding or removing axes as easy as plugging in another set of limit switches and motor drivers. This also supports the educational requirement of using black-box hardware. Each axis is actuated by a standard motor driver/controller that can be ripped out of the mill and used to drive any other robot, regardless of the number of degrees of freedom.

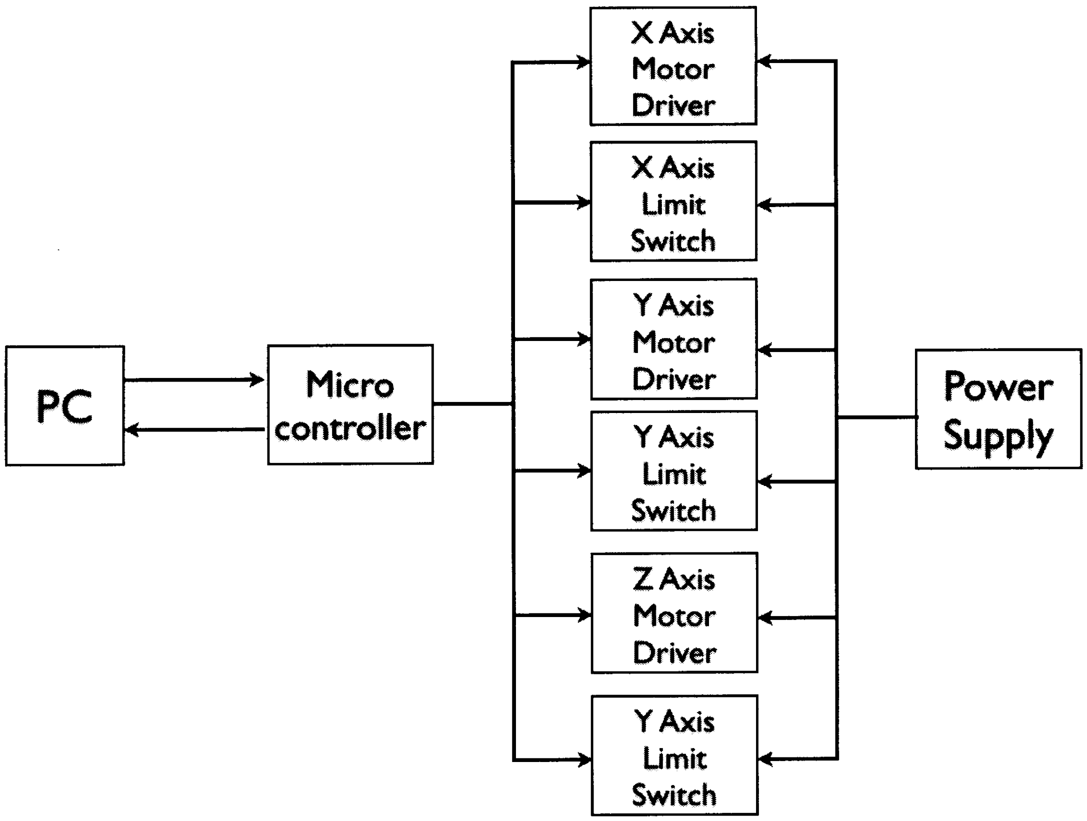


Figure 3-2: Schematic of electronic architecture

3.2 Component Selection

The selection of components for this kit represents the most important and difficult design decisions. A bad stepper driver or a great microcontroller could make or break the educational value of the entire project. With that said, special care was taken in the selection of each component. The reasoning behind each decision follows.

The most important component is probably the microcontroller. Responsible for communicating with the PC and controlling the actuators and sensors of the mill, there are thousands of possible microcontrollers that would all suit this project very well functionally. Unfortunately, there are very few microcontrollers suited for educational environments. The three main platforms used in academia are the Arduino[4], the Basic Stamp[5] and Texas Instruments[6]. The Arduino is an open-source microcontroller designed for ease of use and flexibility. The Basic Stamp, produced by Parallax, Inc., is an entry-level microcontroller targeted at first-time users. Finally, Texas Instruments makes the TI MS series of microcontroller with a focus on low cost and exact specifications. TI sells several orders of magnitude more microcontrollers than the other two models combined and their products are found in everything from coffee makers and alarm clocks to automobiles and the space shuttle.

The pros and cons of each microcontroller platform are shown in Table 3-1. The TI chip comes in lowest for price, with the Arduino in second. The Arduino has the most flash and RAM by far, while the Basic Stamp has a faster CPU. The number of I/O pins of the Arduino is the highest again, at 20. In addition to the performance specifications, the Arduino is the only microcontroller with a cross-platform integrated development environment, an educational requirement of the system. These reasons make the Arduino, shown in Figure 3-3, the natural choice for an entry-level microcontroller.

Table 3-1: Comparison of microcontrollers

	Arduino	**Basic Stamp**	**TI MS Series**
Model	Duemilanove	2pe Motherboard	MSP430
Price	$30	$69	$10
Flash	32KB	2KB	2KB
RAM	2KB	64B	128B
CPU	16MHz	20MHz	1MHz
Pins	20	13	14

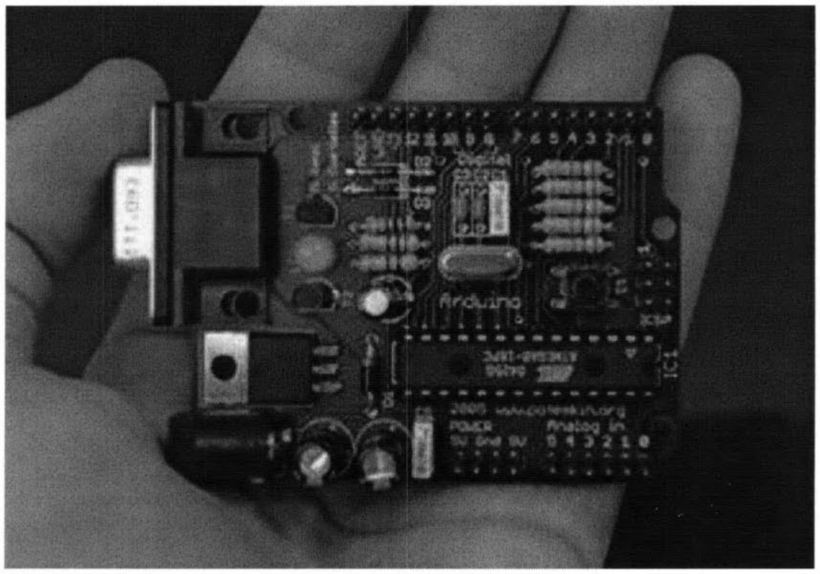


Figure 3-3: Arduino microprocessor

The next major component required is the motors. Motors come in three main flavors: DC, servo and stepper. DC motors are the most basic. A voltage is applied over the motor and a torque results. DC motors have no internal mechanism for feedback or control, so a circuit with an encoder is usually paired with the motor to allow position and velocity to be measured and controlled. Servo motors are DC motors with a potentiometer internally attached to the shaft. Servo motors use this potentiometer for position feedback and are capable of automatically moving to and holding any position within one rotation (+/-180 degrees). Stepper motors are internally wound to turn each rotation into an integer number of steps. Common step sizes are .9 degrees or 1.8 degrees, requiring 200 or 400 steps per revolution, respectively. Stepper motors have the benefit of not requiring complex feedback circuitry; the motors take one step every time they are told to by the controller.

The milling machine kit will require the motors to move more than one rotation in either direction, ruling out servo motors as a possibility. DC motors with encoders are cheaper than stepper motors, but when the supporting drivers and controllers are taken into account the costs are roughly the same. Since a major educational requirement is to favor simplicity, stepper motors were chosen as the actuation platform.

The specific stepper motors chosen are the W0-211-13-01 stepper motors from Lin Engineering[7]. These are the smallest motors available that meet the required torque and rpm values. These motors require a driver to amplify the controller signal. The RepRap Stepper Motor Driver V2.3[8] is a very inexpensive and easy to use stepper motor driver originally designed for home-built 3D printers. The driver accepts two inputs: a step and a direction. Whenever a step is received the motor is moved one step in the direction indicated. The stepper motor and driver are shown in Figure 3-4.

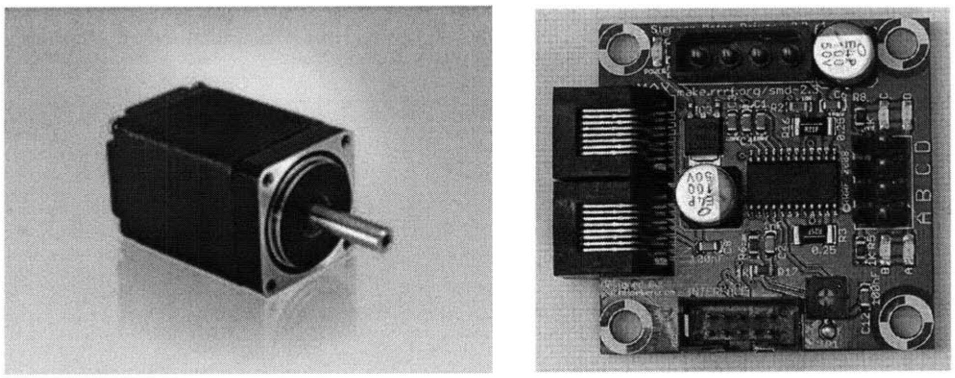


Figure 3-4: Stepper motors and control boards

The final element requiring selection is the power supply. The power supply must be capable of powering the three stepper motors, the three motor drivers and the microcontroller. An ideal power supply would also be inexpensive, readily available and provide several different power outputs and voltages. Commercial power supplies that meet the functional requirements are available, however they are very expensive. Thankfully, every personal computer has a power supply inside it which meets all of these requirements and is very inexpensive. A standard ATX power supply[9], shown in Figure 3-5, provides the necessary voltages and has at least five output plugs.

Figure 3-5: Power supply

3.3 Design

After the architecture is laid out and the components are selected, the next step is a standard wiring diagram. The ATX power supply is wired to each stepper driver using the standard IDE Drive power connectors. These drivers are then connected to their respective motor. The Arduino is wired to each motor driver using an I2C connector and to the PC using a USB cable. The black box circuit schematic is shown in Figure 3-6.

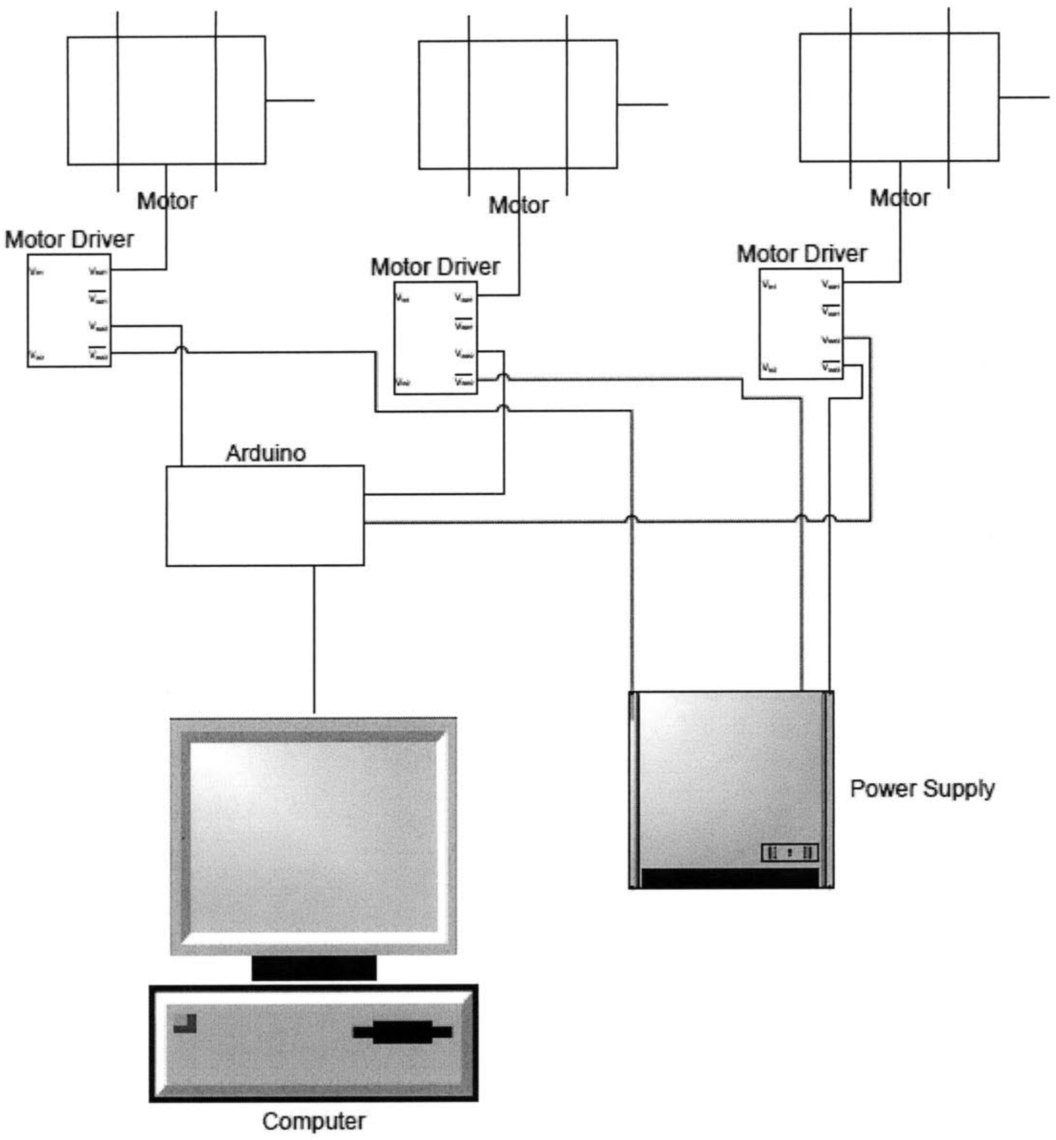


Figure 3-6: Black box circuit diagram

A picture of the Arduino wired to everything is shown in Figure 3-7. The motor driver connections and USB connection are shown. The stepper motor driver with wiring connections is shown in Figure 3-8. There are four leads to the motor, two from the Arduino and four from the power supply. The motor is shown in Figure 3-9. There are four leads from the motor driver that are connected using an IDC connector.

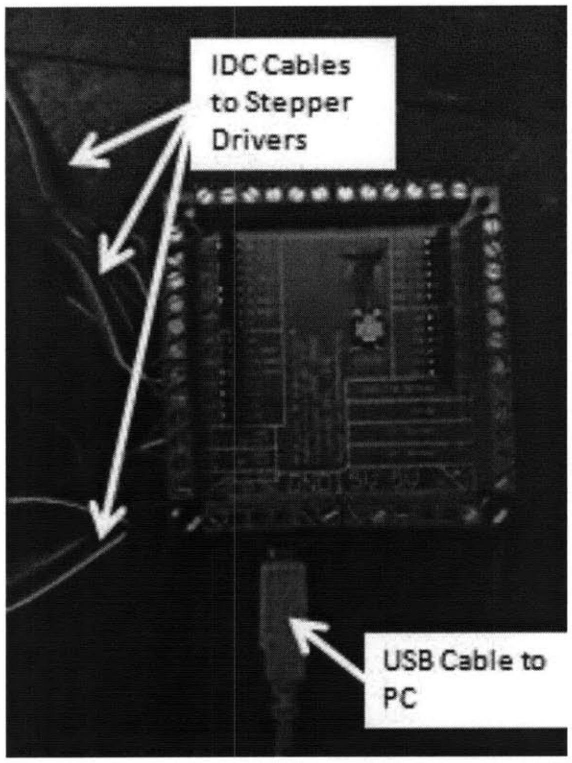


Figure 3-7 Arduino with annotated wiring connections

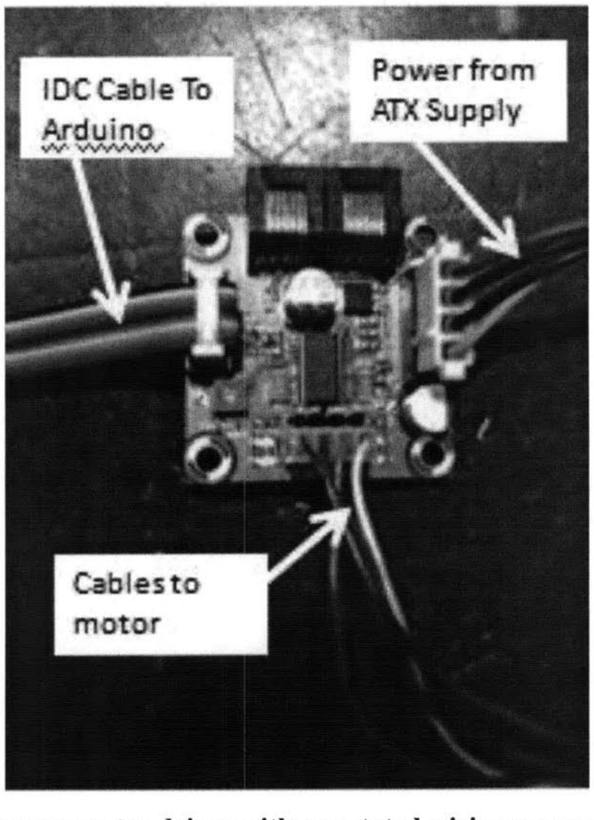


Figure 3-8 Stepper motor driver with annotated wiring connections

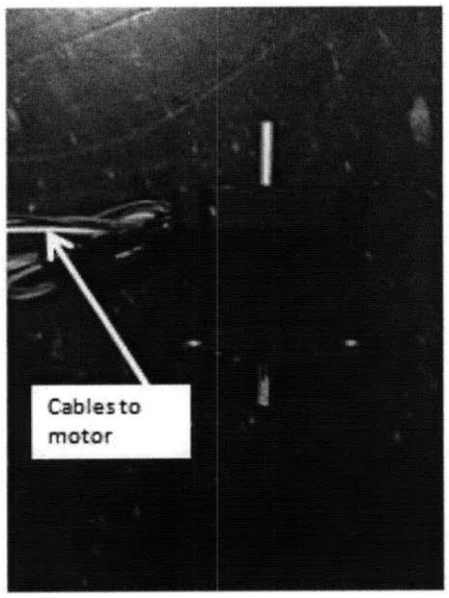


Figure 3-9 Stepper motor with annotated wiring connections

---

[3] FUNdaMENTALs of Design. Alex Slocum. http://pergatory.mit.edu/2.007/fundamentals

[4]Arduino Microcontroller Home Page. http://www.arduino.cc/

[5] Parallax, Inc. Home Page. http://www.parallax.com/

[6] Texas Instruments, Inc. Home Page. http://www.texasinstruments.com/

[7] Lin Engineering, Inc. Home Page. http://www.linengineering.com/

[8] MakerBot, Inc. Home Page. http://www.makerbot.com/

[9] ATX Power Supplies. Wikipedia. http://en.wikipedia.org/wiki/ATX

CHAPTER

4

SOFTWARE DESIGN AND IMPLEMENTATION

4.1 Software Selection

While almost any commercial software development package available today meets the requirements of this project, keeping the software beginner-friendly eliminates many options. The three biggest beginner development languages are Java from Sun Microsystems[10], Matlab from Mathworks, Inc.[11], and Visual Basic from Microsoft, Inc[12]. Java and Visual Basic are both free or come in free versions, but the learning curve is generally higher for these two than with Matlab. Matlab and Java are cross-platform while Visual Basic only runs on Windows machines. Additionally, Java is open-source and is taught in several Computer Science classes at MIT. Table 4-1 contains the summarized comparisons of these software platforms.

Although Matlab does require a costly user-license, MIT has already purchased Matlab seats for every student, making it essentially free. In addition, mechanical engineering students will all be taught Matlab as sophomores in course 2.086 in future years. The fact that Matlab is cross-platform, essentially free and students will already be familiar with it makes it the best choice of software platform for this project.

Table 4-1: Comparison chart of software platforms

	Java	**Matlab**	**Visual Basic**
Platforms	All	All	Windows
Free	Yes	No	Yes
Open Source	Yes	No	No
Easy Development	No	Yes	No
Student Familiarity	No	Yes	No

4.2 Software Architecture

After selecting Matlab as the primary development platform, the software architecture was designed. Starting from the functional requirements, the necessary components were decided then organized into the final architecture, shown in Figure 4-2.

The necessary software components, shown in Figure 4-1, start with the GUI. The user must interface with the software to tell the machine what to do. The standard format for interfacing with CNC machines is G-Code, so the software will allow the user to input G-Code. This G-Code is run through a G-Code processor in order to turn commands like G01 X3.5 Y7 S3500 into the necessary machine motions. The machine is then told what to do by the machine controller.

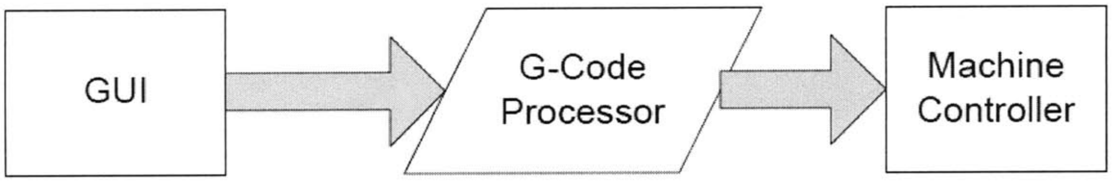


Figure 4-1: Necessary Software Requirements

Since the electronic architecture calls for two "intelligent" components: the PC and the Arduino, the work must be divided between these. The PC will be running the graphical user interface through Matlab, where the user will be able to edit and input G-Code. The Matlab GUI will process the user's entered G-Code and send this to the Arduino via a USB cable. The Arduino will interpret the G-Code and translate it into machine commands, making the axes move as desired.

One main benefit of this architecture is error-handling. While processing the G-Code to be sent to the Arduino, errors such as malformed lines and tool crashes can be detected. This adds a degree of safety to the user interface so inexperienced machine programmers can be assured that their code will not cause expensive damage.

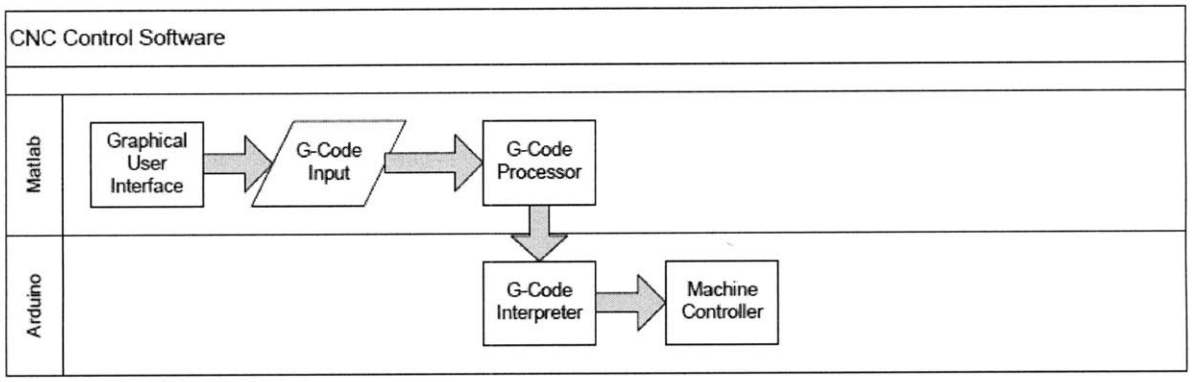


Figure 4-2: Software architecture diagram

4.3 GUI Development

The first step in developing a graphical user interface is "paper" prototyping. "Paper" prototyping is the act of sketching several iterations of a user interface with pencil and paper or a drawing program and showing the designs to potential users. Users comment and rank the designs they prefer and a final design is converged upon. The initial list of required GUI elements used in paper prototyping is shown below in Table 4-2.

G-Code Text Entry Field
Machine Start
Machine Pause
Machine Emergency Stop
Current Location Indicators
Move To Location
Set Location
Spindle Speed Indicator
Tool Indicator
X,Y,Z Speed Indicator

Table 4-2: GUI elements

The first iteration of the paper prototype is shown in Figure 4-3. The required GUI elements were simply placed around the screen without much thought. Upon first glance, the "Current Location", "Set Location" and "Move To Location" fields are all very similar, and could possibly be condensed into one field. Additionally, the Current Movement Speed is a user-modifiable value and would benefit from a slider of some sort to change it incrementally. Finally, a facility for managing the serial port the mill is connected to must be added.

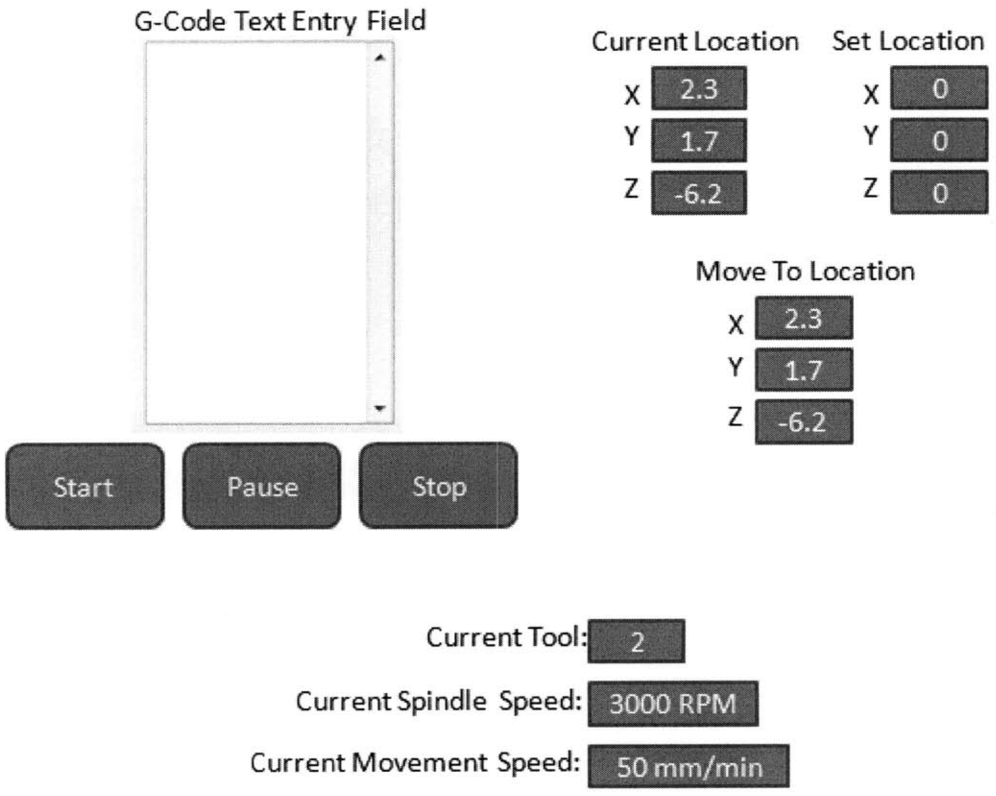


Figure 4-3: Initial paper prototype of software GUI

The second and final iteration of the software GUI is shown in Figure 4-4. The slider has been added to the Current Movement Speed section, the location tabs have been condensed into one set of locations and the serial port dialog has been added. This prototype, while far from perfect, represents a good balance of usability and ease of implementation. All of the GUI elements chosen are standard in the Matlab GUIDE toolbox.

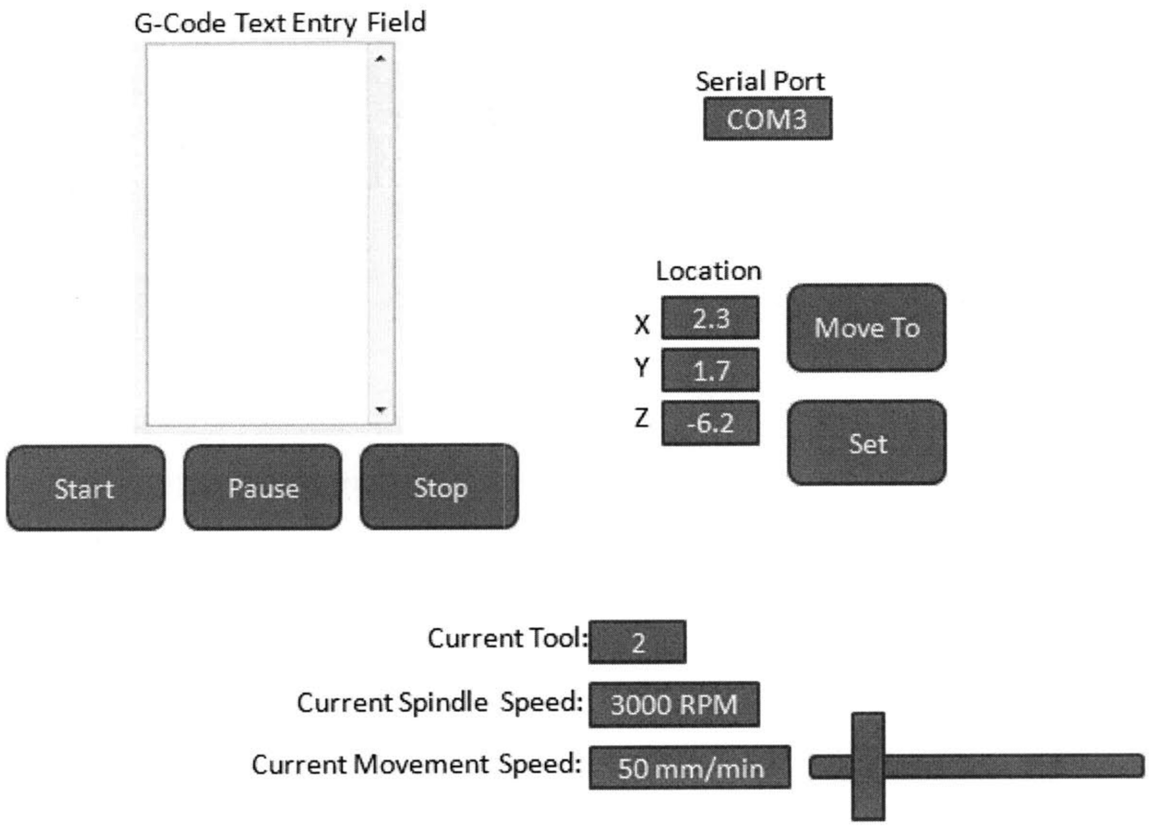
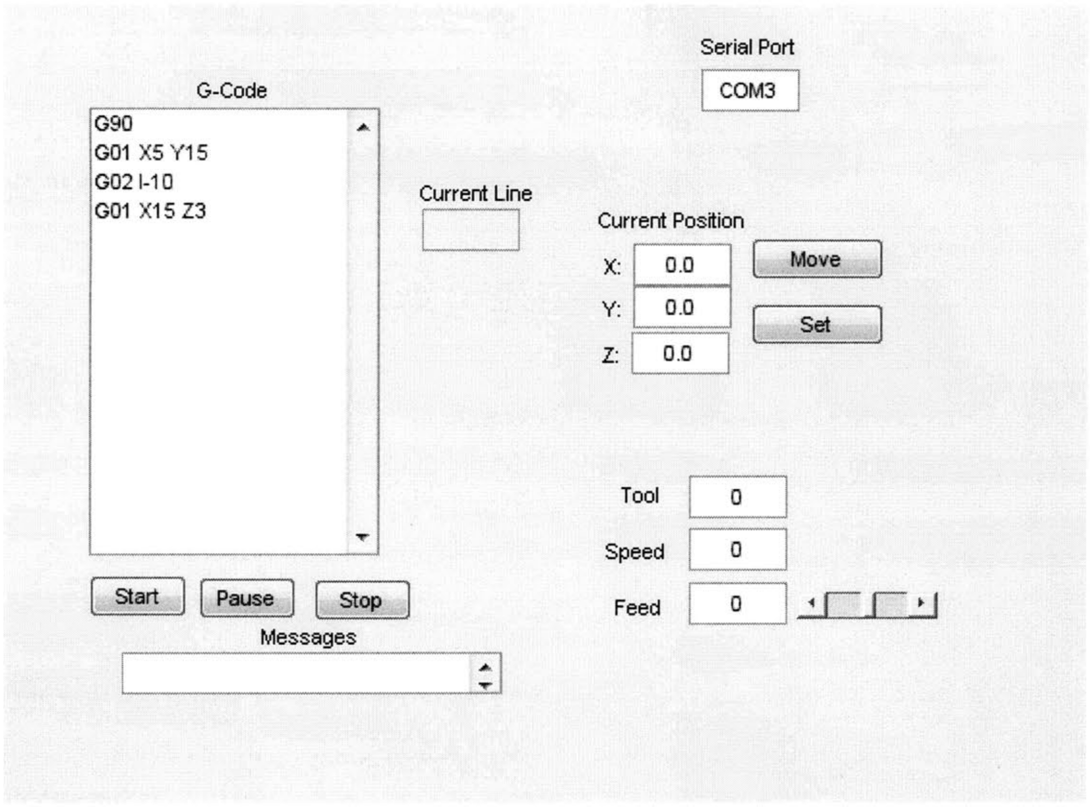


Figure 4-4: Second and final paper prototype of software GUI

Implementation of the GUI design was done in Matlab's GUIDE tool for automated GUI development. The elements are dragged and dropped onto a canvas from a toolbox, much like the paper prototyping. Values and properties can be set, and then Matlab generates skeleton code to operate the GUI. The user then fills in all the stubbed out functions to make the business logic operational. The final GUI design is shown in Figure 4-5.



Serial Port: COM3

G-Code:

```
G90
G01 X5 Y15
G02 I-10
G01 X15 Z3
```

X: 0.0
Y: 0.0
Z: 0.0

Tool: 0
Speed: 0
Feed: 0

Figure 4-5: Screenshot of Matlab GUI

---

[10]Sun Microsystems. Java Home Page. http://java.sun.com/

[11] Mathworks, Inc. Home Page. http://www.mathworks.com/

[12] Microsoft Developer Network. Visual Basic Home Page. http://msdn.microsoft.com/en-us/vbasic/default.aspx

CHAPTER

5

COURSE DESIGN AND IMPLEMENTATION

5.1 Course Requirements

It is not enough for students to merely build a functioning CNC mill. They must understand all of the aspects of its design and construction so they can use these concepts later. For this reason, the course would be best implemented if it forced the students to learn the electronics piece by piece.

This process would be enabled with a small series of objectives required of the students in 2.72. The first objective would be a simple "Hello World" program on the Arduino microcontroller, the details of which are shown in Table 5-1. This would require students got the required build environment configured and were comfortable with the basic syntax of the C language. A sample program, shown in Appendix B.1, requires the students to blink an LED at a regular interval.

Table 5-1: Requirements for "Hello World" task

Requirements for "Hello World" Task
Arduino Microcontroller
LED
USB Cable
PC

While this task seems trivial and pointless, accomplishing it forces the students to learn to use both the Arduino and more importantly the online documentation. Pin diagrams, code samples and tutorials are all available on the internet, for free. Students that are capable of leveraging existing knowledge will be much better suited for future work in the field of electromechanics.

The second task required of students should be to get a stepper motor spinning. This task exposes the students to wiring the Arduino to other circuit boards, using auxiliary power supplies and will probably be their first experience constructing bare-bones electromechanical circuits. Additionally, after completing this task students will now have the ability to actuate and control arbitrary mechanical designs: a main objective for the course. The hardware requirements for this task are shown in Table 5-2. The finished software is shown in Appendix B.2.

Table 5-2: Requirements for "Spinning" task

Requirements for "Spinning" Task
Arduino Microcontroller
Stepper Motor
Stepper Motor Driver
ATX Power Supply
USB Cable
IDC Cable
PC

The final task should, of course, be the construction and actuation of the milling machine kit. While the code base grows significantly from Task 2, the complexity does not. The system responds to inputs by moving the motors. There will be a temptation in teaching the course to create guides for the students to follow during these tasks, but this temptation should be avoided. All the information needed to complete these is available for free online and forcing the students to learn to access it there will help them immensely later in their careers.

CHAPTER

6

DISCUSSION AND RESULTS

6.1 Results

The test stage was fabricated roughly following the "McWire Cartesian Bot[13]" design. Intended to be an easy to fabricate platform for CNC mills, routers or 3D-printers, the "McWire Bot" served the purposes of this design very well. The finished prototype is shown in Figure 6-1.

The machine is connected to any laptop using a USB cable and the host software is run using Matlab. G-Code, produced from any standard CAD/CAM package, is entered into the GUI and run on the machine. The full list of supported G and M codes is given in A.2. The standard workflow for machining a part is as follows:

1. CAD of part in Solidworks or similar
2. CAM of part in Mastercam or similar
3. Post-processing of toolpaths in Mastercam or similar
4. Import of G-code into Matlab GUI
5. Machine zeroed using edgefinder and Matlab GUI
6. Part machined

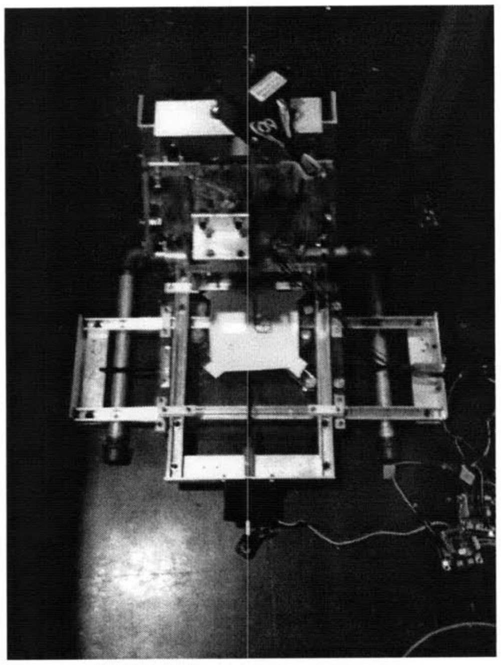


Figure 6-1: Finished prototype of CNC kit

The machine was verified using a testing apparatus shown in Figure 6-2. A marker was attached to the vertical stage of the test machine and a sheet of paper was placed on the horizontal stage. Several g-code commands were executed repeatedly and the path of the machine was recorded by the marker traveling across the paper. Specifically, the command "G02 I-10" was run five times, causing the pen to move in a clockwise circle over the same path each time. As expected, the resolution of the machine was far under the width of the marker used, 2mm. The marked-up sheet is shown in Figure 6-3.

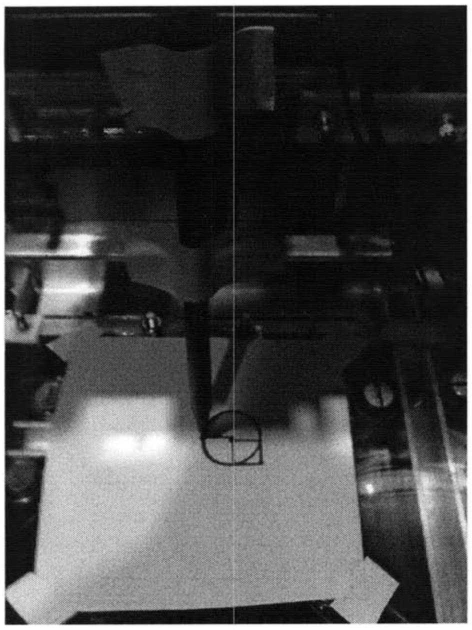


Figure 6-2: Testing apparatus using marker and paper

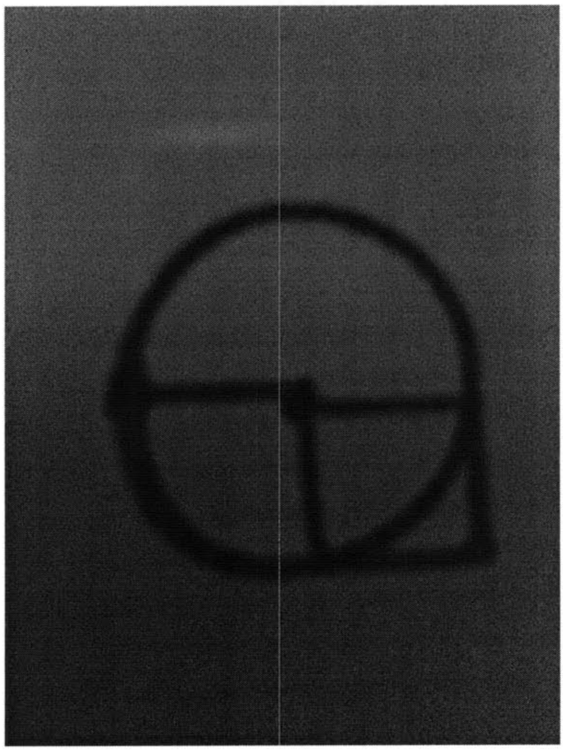


Figure 6-3: Paper used in testing apparatus

[13] RepRap Project. McWire Cartesian Bot Guide. http://reprap.org/bin/view/Main/McWire_Cartesian_Bot_1_2

CHAPTER 7
CONCLUSIONS

The scope of this thesis was to design, test and fabricate the electromechanical subsystem of the CNC milling machine kit. The mechanical design is ongoing at the time of this writing, requiring the electromechanical subsystem to be tested on a different 3-axis stage than that of the kit. Although the two systems will have to be tested at a later date, everything up to the motors is identical between the test stages and the final stages. The author sees no possible issues, performance or otherwise, in bolting the motors do a different lead-screw driven 3-axis stage.

While the design, fabrication and testing of the electromechanical susbsystem of the CNC mill kit was successful, there is still some work to be completed in the future before a course is taught using this platform.

The Arduino firmware supports limit switches, but the author has left these out of the initial prototype because the selection of limit switches is highly dependent on the mechanical design. Once the final mechanical design has been prototyped, limit switches that fit onto the stages can be selected and plugged into the Arduino.

Similarly, a rigid enclosure for the electronics has been purposefully left out of this initial prototype. This housing should be designed with the rest of the mechanical system in mind. Several possible layouts are possible. The entirety of the electronics can be housed in one body and wires run to all the motors and limit switches or the main microcontroller can be housed centrally and the motor drivers and limit switches can be placed near the motors. Both configurations have their advantages and the decision will need to be made when the final prototype is completed.

APPENDIX

A

OPERATIONAL SOFTWARE CODE

A.1 Code for Matlab GUI

The most recent version of the Matlab software GUI will be made available after the completion of this thesis at the URL: http://dl.dropbox.com/u/30118/matlab-code/gui.m, The code is several thousand lines long and is ill-fit for written publication.

A.2 Code for Arduino

Once again, the most recent version of the Arduino firmware will be made available at http://dl.dropbox.com/u/30118/arduino-firmware/stepper_control.pde. This code is also far too long for written-publication. The listing of supported g-codes and m-codes, as well as their syntax is shown in Table 7-1.

Table 7-1: Operational codes for Arduino firmware

G-Code	Description	Syntax
G00	Rapid Linear Interpolation	G00 X5 Y5 Z5
G01	Linear Interpolation	G01 X5 Y5 Z5
G02	CW Circular Interpolation	G02 I-10
G03	CCW Circular Interpolation	G03 I-10
G04	Dwell	G04 10
G20	Inches	G20
G21	Mm	G21
G28	Home	G28
G30	Home via intermediate point	G30 X5 Y5 Z5
G90	Abs mode	G90

G91	Inc mode	G91
G92	Set point	G92 X.5 Y0 Z0
M3	Spindle On	M3
M4	Spindle CCW	M4
M5	Spindle stop	M5
M6	Tool change	M6

APPENDIX B

EDUCATIONAL SOFTWARE CODE

B.1 "Hello World"

```
/*
   Blink

  Turns on an LED on for one second, then off for one second, repeatedly.

  The circuit:
  * LED connected from digital pin 13 to ground.

  * Note: On most Arduino boards, there is already an LED on the board
  connected to pin 13, so you don't need any extra components for this
example.


  Created 1 June 2005
  By David Cuartielles

  http://arduino.cc/en/Tutorial/Blink

  based on an orginal by H. Barragan for the Wiring i/o board

  */
 int ledPin =  13;    // LED connected to digital pin 13
 // The setup() method runs once, when the sketch starts
 void setup()   {
   // initialize the digital pin as an output:
   pinMode(ledPin, OUTPUT);
 }
 // the loop() method runs over and over again,
 // as long as the Arduino has power
 void loop()
 {
   digitalWrite(ledPin, HIGH);    // set the LED on
   delay(1000);                   // wait for a second
   digitalWrite(ledPin, LOW);     // set the LED off
   delay(1000);                   // wait for a second
 }
```

B.2 “Stepper Motor Spinning”

```
/*
   Spin

  Spins stepper motor

  The circuit:
  * Stepper motor hooked up to RepRap stepper motor driver.
  * Step Pin-4 Direction Pin-5 Enable Pin-6


  Created 10 January 2010
  By Dan Lorenc

  */
 int step_pin =  4;
 int dir_pin =  5;
 int enable_pin =  6;
// The setup() method runs once, when the sketch starts
 void setup()   {
   // initialize the digital pin as an output:
   pinMode(step_pin, OUTPUT);
   pinMode(dir_pin, OUTPUT);
   pinMode(enable_pin, OUTPUT);

   digitalWrite(enable_pin, LOW);      // inverted pin, low is on
   digitalWrite(dir_pin, HIGH);      // step forward

 }
 // the loop() method runs over and over again,
 // as long as the Arduino has power
 void loop()
 {
   digitalWrite(step_pin, HIGH);    // pulse pin on
   delay(5);                     // wait
   digitalWrite(ledPin, LOW);     // pulse pin off
   delay(5);                     // wait
 }
```