SENIOR DESIGN I GROUP 03

Magic Chess

Haley Amason
Joshua Burbridge
Brittany Nottingham
Thong Tran

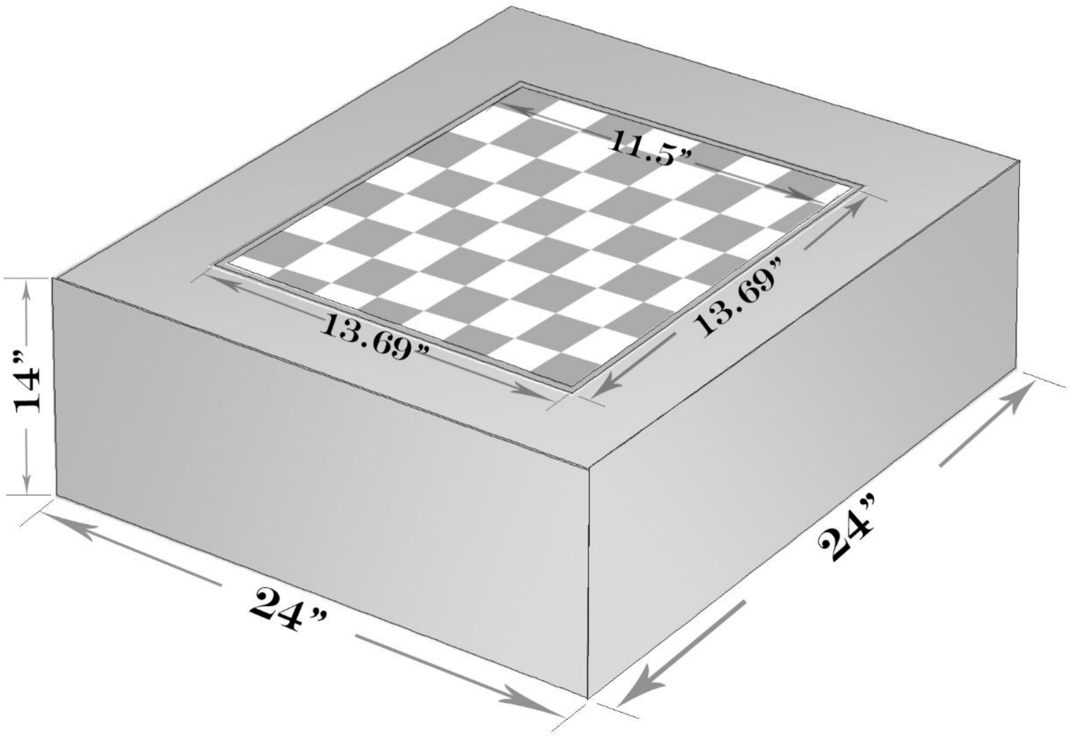


Table of Contents

Table of Figures

Table of Tables

Acknowledgements

We would like to acknowledge Dr Samuel Richie for his helpful guidance throughout the year, and for the speakers and resources he has provided for the Senior Design course.

We would like to acknowledge Soartech for their sponsorship, their interest in our project, and the team in the Oviedo office for all of their words of wisdom.

We would like to acknowledge Joshua Haley for his expertise, continuing guidance, for personally meeting with us for advising, and for donating essential hardware components to Magic Chess.

We would like to acknowledge Tom Kerrigan for his excellent open source chess engine, which drives the core of this project.

We would like to acknowledge the team at Carnegie Mellon University for their creation of PocketSphinx, the software used for voice recognition.

We would like to acknowledge Carl S for the mechanics of the XY stage.

We would like to acknowledge the Accreditation Board for Engineering and Technology for maintaining high standards in engineering education.

Finally, we would like to acknowledge the University of Central Florida for providing an environment conducive to research and design, and for providing helpful on-campus resources to aid us in our project.

Chapter 1 : Executive Summary

Magic Chess is a voice-activated, hands-free chess board with both player versus player (PvP) and player versus computer (PvC) capability. The primary features of this board are the ability to communicate with it via voice commands, the absence of a player's need to physically move the chess pieces, and the inclusion of an intelligent chess engine that allows someone to play alone.

The structure of each game follows a simple series of events. When starting a new game, a player will be able to access a basic, easy-to-use interface via an LCD screen. This initial screen will present the player with a few simple options, such as the number of players and, if applicable, the difficulty level of the computer. A human player can make a move by clearly vocalizing a command (e.g. Knight to E5). The board will then interpret this command via voice recognition software and utilize the electromagnet controller to automatically carry out this move without any additional action by the player. Move legality is checked after every command. If a player attempts an illegal move, the board will let them know via a message on the LCD screen, and will wait for them to attempt another move. In PvC mode, the board will track the state of the chess board and each move the player makes, and then utilize decision-making algorithms such as Minimax and Iterative Deepening to determine the best move. Again, the electromagnet in the board will complete the corresponding move on behalf of the computer, and then wait for the player to make their next move. After every move, the board will check for endgame conditions (i.e. Check, Checkmate, or Stalemate). If any of those is detected, the board will take notice and operate accordingly.

Magic Chess is intended to be a fun game that can be enjoyed almost anywhere. To make the game more appealing, we designed it to be as portable as possible. All of the necessary components to play a full game are included in the board itself. In addition, we had to provide some shielding to protect the electronics from the magnetic component of the game. This makes the board somewhat heavy, but we felt that this was the best for both portability and keeping everything safe. The only technical requirement for operation is that it must be plugged into a standard 15 volt AC socket. The diagram on the next page shows the overall block diagram for the project, illustrating the general flow each game should take.

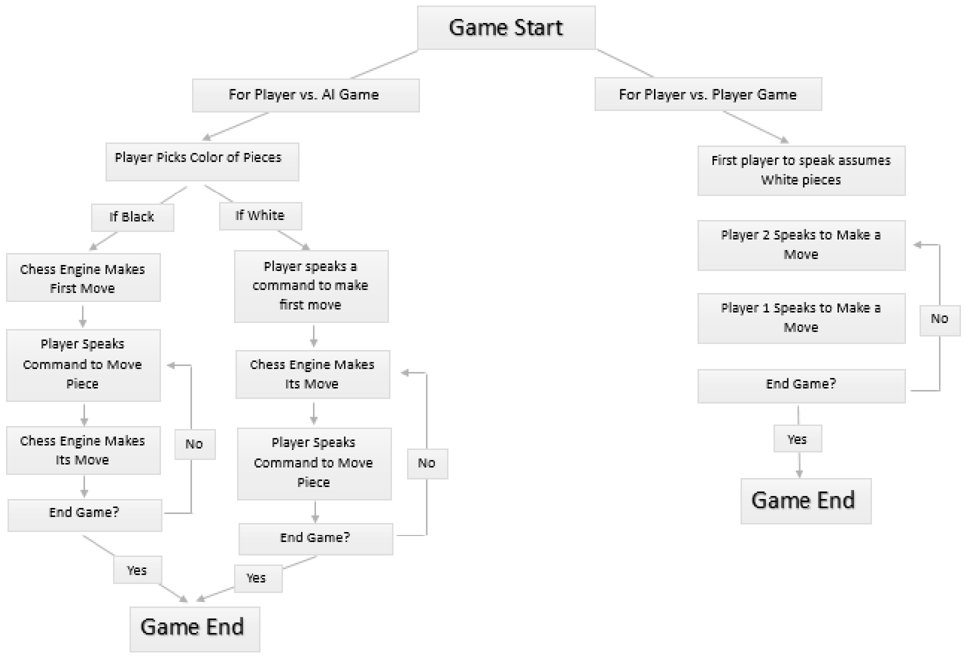

Game Start
For Player vs. AI Game
For Player vs. Player Game
Player Picks Color of Pieces
If Black
If White
Chess Engine Makes First Move
Player Speaks Command to Move Piece
Chess Engine Makes Its Move
No
End Game?
Yes
Player speaks a command to make first move
Chess Engine Makes Its Move
Player Speaks Command to Move Piece
No
End Game?
Yes
Game End
First player to speak assumes White pieces
Player 2 Speaks to Make a Move
Player 1 Speaks to Make a Move
No
End Game?
Yes
Game End

Figure 1-1 Overall Block Diagram

Chapter 2 : Project Description

Section 2.1: Project Motivation and Goals

The motivation for this project comes from wanting a fun, stimulating project that is challenging and purposeful at the same time. The purpose of this project is to eventually donate the finished product to a children's hospital as a toy. All group members like the concept of a "no touch" (or "wizards") chess, since it contains both significant hardware and AI components.

The concept for "no touch" chess comes from the J.K. Rowling book and Warner Brothers' movie, Harry Potter and the Sorcerer's Stone. In the movie, the wizards tell the pieces where they want them to move and the pieces move themselves to that location without any physical interaction from a human. The pieces also capture other pieces in the game, and the captured pieces are then removed from the board. Also in "wizards" chess (as it is called in the book), a player is allowed to either play against another player, or against an AI of some kind (in the movie, the pieces are somewhat sentient, so they can play themselves, but for our purposes, they would be computer controlled). We wanted to recreate a board of this type, remaining as faithful as possible to the imaginary version. To do this, we had to make sure to emphasize a few key factors.

The first was portability. We wanted people to be able to easily move the board to almost any location with minimal effort. This meant that all the electronic components that make the board work had to be encased together in a space-efficient manner. The primary difficulties that this challenge posed were making sure the electromagnet that moves the pieces had enough room to operate, providing some sort of shielding for the electronic components to protect them from magnetic fields, and allowing enough room to prevent overheating of the parts. There was an obvious trade-off between allowing enough space in the board for the entire system to operate and maximizing portability.

The second factor was responsiveness. In the fictional version of Wizards' Chess, voice commands are understood immediately, and the pieces are moved at once. To have a truly faithful reproduction, we could not allow the board to take 60 seconds to interpret a voice command, and an additional 60 seconds to respond and make its move. To capture the magical element of the game responsiveness had to be strongly emphasized. This meant that the software we used had to be as computationally inexpensive and fast as possible. To do this, we performed extensive testing of different open-source chess engines, ultimately using our own benchmarks to decide which software to include.

The third factor was intuitiveness. We did not want the user(s) to have to do any troubleshooting, nor should it take them more than a few minutes to learn how to set up the game and utilize the basic controls. The idea for this project was inspired by a series of children's novels, so we assumed many of the people interested in playing this game would be children. For this reason, we decided to include an LCD screen on the board. This is intended to give the player a familiar user interface as a starting platform, and as they become more comfortable with the board, they may begin to utilize voice commands. If ever a user does not understand something, they can find helpful information on the LCD screen or in the included instruction manual.

The fourth factor was reliability. Because the board includes moving parts, and because we were using a variety of open-source software, we had to do quite a bit of testing to ensure that all of the subcomponents could integrate well with one another. It would be unacceptable if, for example, the computer began making illegal moves, or the electromagnet began to move the wrong pieces. We found that the best way ensure our board was reliable was to start with a core subsystem, test it extensively, integrate another subsystem into it, test it extensively again, and repeat this process, continuing to add each piece one-by-one.

The fifth and final factor was minimalism. Specifically, minimalism meant never over-engineering or using more parts than strictly necessary, minimizing the budget, and focusing on making the system as simple as possible overall. We wanted to avoid the problem of escalating complexity that occurs when systems become bloated or redundant.

By devoting ourselves to all of these factors and considering them heavily in every major decision, we hope to design and build a truly remarkable product.

Section 2.2: Objectives

- Design and build a self-contained system on which a user can play chess
- Self-moving pieces
- Pieces controllable via voice command
- One player and two player modes
- Varying difficulty level
- Stay within budget of approximately 500 USD

Section 2.3: Project Requirements and Specifications

The project requirements listed below were agreed upon by the whole group. They are divided into hardware, software and miscellaneous specifications to cover as many areas as possible.

Section 2.3.1: Hardware Specifications

- Board will be 20” x 20” x5” or less
- Board material must allow for significant enough flow of EM waves for pieces to move
- LCD Screen depicting current play field
- Micro ATX form factor motherboard for computer to run AI component of project

- Secondary custom PCB including microcontroller and stepper motor control hardware
- Case for the final product will be made out of either aluminum or plexi-glass to preserve a professional and put-together final product.
- Stepper motors will move fast enough to be able to complete piece movement in 3 seconds or less.

Section 2.3.2: Software Specifications

In Hands-Free Chess, software must be included to perform the following general tasks: maintain a representation of the chess board, recognize input and send the appropriate output, play a full game of chess against a human opponent, recognize voice commands, operate the electromagnet, and organize and control all of the above tasks. Due to the complexity of all of these things, open-source software with some modifications will be used when possible. Other tasks, such as the magnetic controller, will require specialized software developed by the team.

The first two tasks can be accomplished by what might be called a "gamekeeping" module. This is the software necessary to keep track of what is happening on the board, as well as accept input from both players, check for move legality, check for endgame conditions, and keep track of whose turn it is. This part of the software is responsible for the flow of the game between two players, be them human or computer.

The third task, being able to play a full game of chess against an opponent, is daunting. Developing an intelligent chess program is a problem that is still being researched extensively, although there are now many programs in existence that can beat most human opponents. These programs use complex strategies and algorithms such as minimax and iterative deepening, which are described in later sections. Luckily, most open-source chess engines include searching functions and play books, so it is likely that, for the purposes of the project, this problem has already been solved.

The fourth task is also crucial to the operation of Magic Chess. Without the ability to order chess pieces to various places on the board, the project would not achieve the aesthetic standards outlined above. Like with the chess engine, there exists free and open-source software to accomplish this task. Due to the hardware constraints, however, it would be best if the voice-recognition software were easily adaptable. This is because it is not a requirement that the software parse and understand anything the users say. What it needs to be able to understand are the chess commands, which come from a dramatically smaller library than what the voice-recognition software will likely be designed to search through. This will reduce the time needed to parse voice commands, and will also place fewer constraints on the hardware in the board itself.

The fifth task is the operation of the electromagnet that moves the pieces. There will almost certainly not be an open-source program that accomplishes this, so it will be necessary to write some code to operate our own embedded system.

The final task is the lower-level software that all of the above software will run on, which is essentially an operating system. It is likely that this will be some distribution of Linux. Any open-source software that was originally written for Windows will then have to run in Wine or some virtual machine.

Finally, it will be necessary to interface all this software together to achieve the flow of calculation outlined in the software diagrams to follow. Specifically, a voice command must be parsed, translated, and fed into the chess engine, which must then update the board. After that, the chess engine must send the move to the magnetic controller to physically move the chess piece. The magnetic controller will then relinquish control to the voice recognition unit or the chess engine, depending on the mode in which the game is being played. Because most of the software was not designed to be used together, the most difficult task involving the software will likely be making sure everything interfaces properly, to provide as smooth a flow to the game as possible.

Additional specifications:

- Chess Engine must have reasonable response time; most moves by the AI should take less than 5 seconds to compute.
- Speech recognition module must be able to interpret commands quickly. This should take 3 seconds or less to avoid confusion and/or repeated commands.
    - The software must have efficient algorithm design. It should run in at least polynomial time, with search algorithms running in O(nlogn) or better.
- The AI unit will have 3 difficulty levels: Easy, Medium, and Hard, with additional look-ahead capabilities with increasing difficulty level.

Section 2.3.3: Miscellaneous Specifications

- At least two modes of play including:
    - Player VS. Player
    - Player VS. AI
- Voice Recognition for at least two different human voices
- Ability to do route management for the pieces as each piece moves differently
- Ability to remember routes for 6 different pieces and identify incorrect moves
- Use of electromagnets and permanent magnets to get pieces to move without human interaction – pieces may be as much as 0.25 pounds.

- The AI response for a Player VS. AI game should take no more than 3 seconds.
- Spaces must be large enough for pieces to move among each other without colliding
- Final project must be sturdy for repeated use by children.

Section 2.4: Project Management

To keep the project manageable, we have divided the project into sections and assigned each section to an individual group member. Each group member is responsible for presenting evidence of going through the six phases of product development: research, design, material acquisition, prototyping, testing and integration. The table below (Table 2-1) lists each section and the group member assigned to it.

Group Member	Section
Haley Amason	Power Supply
Brittany Nottingham	Microcontroller
Brittany Nottingham	LED Screen and Controller
Haley Amason	Electromagnets and Permanent Magnets
Brittany Nottingham	Stepper Motor and XY-Stage
Thong Tran, Joshua Burbridge	Motor Control Programming
Thong Tran	Voice Interpretation
Thong Tran, Joshua Burbridge	Artificial Intelligence
Joshua Burbridge	Chess Engine Control

Table 2-1 Table of Group Member Responsibilities

The purpose of assigning tasks is to divide and conquer the project and to provide group members with a way of keeping each other accountable and on task when working towards project deadlines. Weekly meetings have been held to discuss progress and talk about project direction.

Chapter 3 : Research

Section 3.1: Existing Similar Projects and Products

This section outlines existing products and senior design projects that we could find to help further define and explain the “No Touch” chess project. It includes a Wireless Arduino Powered Chess set that allows players to play through the internet as if they were playing on the same board and a magnetic child’s play set and a commercially available magnetic chess set for children and adults.

Section 3.1.1: Wireless Arduino Powered Chess

Wireless Arduino Powered Chess is an online chess game built by another senior design student here in the United States. It features two chess boards connected wirelessly to the internet at different locations that interact with a single user for each board. Gameplay is started by the white piece user moving his/her piece on his/her personal board. The board registers the move and the new location of the piece that was moved, and transmits the data wirelessly through the internet to the other board. The other board then uses an XY-stage powered by stepper motors to move an electromagnet under the corresponding piece and move it as if the other player had moved it themselves. This project includes illegal move handling, a GUI for the online interface, and other features. This project is similar to ours in the fact that it controls pieces using an XY-stage powered by stepper motors and moves the pieces without human interaction.

Section 3.1.2: Magnetic Chess Game for Children and Adults

In the early stages of the research, it was necessary to estimate the optimal amount of hardware resources necessary to run the software. It was noted that many retail stores such as Toys ‘R Us and Wal-Mart carry electronic chess boards as well as handheld chess games. While these products do not run all of the software necessary to implement the hands-free chess board, it was clear that some sort of intelligent chess-playing algorithm was being implemented. These primitive chess products set a minimum standard for the hardware necessary for the project. Unfortunately, it was not practical to purchase many different retail electronic chess games and break them open to examine their hardware.

Magnetic chess boards are common in the board game industry because they are convenient for travel and table play. Magnetic chess boards are more user-friendly than a nonmagnetic board because pieces cannot be easily moved by a small disturbance through the table or surface the board is on because the pieces are held in place by the magnetic attraction of the small magnet on the chess piece and ferromagnetic board until they are forced to move by a force greater than the magnetic attraction or pull between the magnet on the chess piece and the board.

There are various versions of magnetic chess available. For example Phantom Force Chess uses magnets to move its chess pieces across the board [1]. The game comes with two modes of play; you can play against the computer or watch the computer play itself. If the computer plays on its own both sides of chess pieces will move. If the user plays against the computer, the user is responsible for moving its own chess pieces while the computer’s chess pieces move by the magnets in the board. Our chess game will be different from the Phantom Chess game on many levels because it can be played with or without the AI (1 or 2 player) and the chess pieces will be move without any physical interaction from the user.

Differences between Phantom Chess and No Touch Chess

Specifications	Phantom	No Touch
AI vs AI Mode	✓	
2 player capability		✓
User vs. AI	✓	✓
Battery power	✓	
AC power	✓	✓
Voice command capability		✓
All chess pieces move without physical interaction from user		✓

Table 3-1 Phantom Chess vs. "No Touch" Chess

Section 3.2: Research of Technologies and Algorithms

This section outlines all research that has been done on technologies to make the project function as described by the specifications and executive summary. It starts with the problem of linear motion and different hardware technologies, then moves on to AI algorithms for the chess engine, voice interpretation and motor control.

Section 3.2.1: Linear Motion

Linear motion using motors is accomplished by using the motor to turn a drive system consisting of a screw, worm drive or belt. Using a screw requires additional math to convert number of pulses, timing or on-time of a motor into traveled linear distance. Screws also have either a non-captive shaft style or an external shaft style [2].Instead of a screw, a belt drive may be used to facilitate linear motion. Part of the belt is fastened to the stage, and a motor is at one end with a gear meshed with teeth in the belt. Rails are also used to guide the stage along a straight path.

Section 3.2.1.1: Motors

This section outlines research done on different types of motors that could be used to facilitate the XY-Plane motion of the stage that will hold the electromagnet.

Section 3.2.1.1.1: Stepper Motors

Stepper motors come in three basic forms: variable reluctant, permanent magnet, and hybrid [3], but all function in basically the same way. Each set of windings is energized one after the other. When the windings are energized, the rotor aligns its own magnetic field with the one being produced by the windings. The more rotor teeth and motor poles a stepper motor has, the more precise the step and control of the position of the rotor.

The advantages to a stepper motor are fairly simple controls, the eradication of a closed-loop control system to determine location, and relatively easy acquisition of the motors themselves.

Section 3.2.1.1.2: DC Motor

A basic DC motor is composed of two electromagnets welded to a shaft. The electromagnets consist of windings of copper on a ferromagnetic core. Bearings are placed at either end of the shaft to hold it in place while rotating. Two permanent magnets surround the electromagnets on the shaft and provide the primary magnetic field. Two brushes touch conductors on the shaft to power the coils and produce a secondary magnetic field in the coils. This field changes as the shaft rotates and continually tries to align itself with the primary magnetic field provided by the permanent magnets. The whole motor is surrounded by a non-conductive housing and powered by two opposite DC voltages. More complex and powerful DC motors are composed of more than two sets of electromagnets and permanent magnets [4]. DC motors require constant analog input to operate.

Advantages of DC motors include easy speed regulation, easy acquisition of spare parts, and absence of voltage peaks on the windings of the motor [5].

Section 3.2.1.1.3: Servo Motor

Servo motors are different from DC motors and stepper motors in the fact that they contain a controller and a set of reduction gears in their housings. Analog servos require a pulsed width modulated signal, power and a ground to operate. Digital servos also use a pulsed width modulated signal, but the difference between the two is in the fact that a digital servo has the control circuitry built into the unit. Most analog servos require a separate control unit. There are also two different types of servos: standard and continuous servos. Continuous servos go to a velocity and hold that velocity, whereas standard servos go to a position and hold it until the user specifies a new location. Servos also require a closed loop control system consisting of external sensors to give it a new position or velocity to match [6].

Advantages of servo motors include high intermittent torque, high speeds, working well for velocity control, and minimal noise production [7].

Section 3.2.1.1.4: Motor Control

Motor control methods vary based on the type of motor being used. The basics of each type of motor control are listed below.

Section 3.2.1.1.4.1: Stepper Motors

Stepper motors are controlled by a series of pulses. Each set of pulses is considered a step. Steps can consist of multiple pulses so an integrated circuit to control stepper motors is recommended [8]. Stepper motors can also be controlled using an H-Bridge configuration, but again it is normally cheaper and more space efficient to use an IC.

Section 3.2.1.1.4.2: DC Motors

DC motors are also capable of being controlled by an H-Bridge circuit of switches and a microcontroller. Speed is relative to the amount of voltage applied and current pushed through the rotor windings. There are also controller ICs available for DC motor applications [9].

Section 3.2.1.1.4.3: Servo Motor

As noted in the previous section, servo motors are controlled by pulsed width modulated (PWM) signals. A microcontroller normally is used to be an intermediary between a sensor which provides closed loop feedback, and the servo motor itself. The microcontroller will accept a signal from an external sensor, decode it, compare it to the current location or velocity of the servo, then send the servo a PWM signal containing the instructions on how to correct its current velocity or position.

Section 3.2.1.2: XY-Stage

With the XY-Stage being an integral part of the project, the group decided that a majority of the budget must go to make it as well made as it could be. Once research started, it was found that the team had two options for the XY-Stage: team assembly and a pre-assembled stage. Each type of stage must fit in the budget and fulfill the amount of travel needed by the project.

Section 3.2.1.2.1: Team Assembly

Team assembly of the XY-Stage could be time consuming and complicated, but the most important factors are whether or not the XY-Stage meets the requirements and specifications laid out in the section for requirements above and also remains in the budget set forth for the project.

There were instructions and parts lists found that the team decided were the best to use as the base for the XY-Stage and the parts total was below the amount designated in the budget. These instructions were also adaptable to fit the amount of travel the XY-Stage for this project required.

Section 3.2.1.2.2: Pre-Assembled Stage

A pre-assembled stage that comes with a motor controller and the proper range of movement would be ideal for this project as the group members are not mechanically inclined. The pre-assembled stage would have to meet our specifications and not go over the budget limits set forth in the following pages of this document.

When doing research on the pre-assembled XY-Stage, there was no available equipment that met the requirements for stage travel or that met the budget set forth by the team in the beginning of the project.

Section 3.2.2: Electromagnets and Permanent Magnets

This section will detail all primary research about electromagnets and permanent magnets their construction and types.

Section 3.2.2.1: Electromagnets

Electromagnets are magnetized by the current running through the coil wrapped around their ferromagnetic core. The current can either be alternating (AC) or direct (DC).Depending on the type of current the electromagnet accepts determines the strength of the electromagnet. Along with the type of current the electromagnet accepts other characteristics were considered in researching the electromagnets. Other characteristics to consider are size, strength and interaction with the material of the board.

Section 3.2.2.1.1: Types of Electromagnets

Section 3.2.2.1.1.1 AC vs. DC Electromagnets

There are various kinds of electromagnets but in the most basic sense there are AC powered electromagnets and DC powered electromagnets. Electromagnets are magnetized by the current running through the coil wrapped around their ferromagnetic core. If the current is alternating, the strength of the magnetic field will alternate in strength. However in DC the current is constant and unchanging which makes the magnetic field steady in its strength. The chess pieces should have as smooth movement as possible, therefore DC electromagnets should be used for the project because of their stable magnetism.

Section 3.2.2.1.1.2 Round vs. Square Electromagnets

Electromagnets also come in round and square form with different weight, size and power requirements. The chess pieces will have a round base and therefore the permanent magnet will also be round in form. To keep a steady attraction between the electromagnet and the permanent magnet they need to be similar in shape, size and strength (pull). Therefore, the electromagnet will need to be round to keep the chess pieces from dragging behind the electromagnet and being positioned slightly off the square designated by the player or AI’s command. By choosing an electromagnet that is round and with a similar diameter as the permanent magnet the chess pieces movement can be better controlled and placement of the chess pieces will be more accurate.

Section 3.2.2.2: Permanent Magnets

The permanent magnets and electromagnets need to be similar in design to increase their magnetic attraction and overall performance in the project. Size, shape and interaction with the material board and the electromagnet needs to be considered. Permanent magnets also come in different variety of composition, size and strength. The varieties of permanent magnets available for use for the project are explained in further detail in section 3.2.2.1.2.

Section 3.2.2.1.2: Types of Permanent Magnets

Section 3.2.2.1.2.1: Composition

Permanent magnets can be broken into three types based on their composition. Rare earth magnets (NdFeB or NIB) are the strongest permanent magnets available because they are very difficult to demagnetize. Rare earth magnets are susceptible to corrosion and therefore or usually sold in a protective casing. Another type of permanent magnet available is Alnico magnets. Alnico (Al, NI and CO) magnets are not as strong as rare earth magnets but they are not easily affected by temperature. However Alnico magnets can demagnetize easily.

Finally, ferrite magnets (commonly found on refrigerators) are have a high coercive force but can still be affected by temperature. Earth magnets will be the best kind of magnet to use because they are difficult to demagnetize and will not be easily affected by the heat coming from the electrical components of the board. [10]

Types of Permanent Magnets

Specifications	Pros	Cons	Advantage
Earth Magnets	Difficult to demagnetize	Susceptible to corrosion	✓
Alnico	Lease affected by temperature	Easily demagnetized	
Ferrite	Readily available and cheap	Strength varies with temperature	

Table 3-2 Types of Permanent Magnets

Section 3.2.2.1.2.2: Shape

Permanent magnets can be easily found any shape desirable. Their shape does not affect the magnetic strength but should be similar to the electromagnet to increase the pull between both magnets. The permanent magnets considered for the design should be round because the base of the chess pieces are round. The magnet should take up as much space on the bottom of the chess piece as possible to increase the attraction between the base of the chess piece and the electromagnet under the game board.

Section 3.2.3: Voice Capture and Interpretation

This section will go over the different hardware and software research that has been done that relates to the voice capture and interpretation. There is one section detailing the hardware necessary and another section that details a couple of different APIs that may be used for interpretation.

In order to have the pieces be able to move through magnetic controllers, we need a way to send commands to the micro controller. In order to do so would require two major components: a microphone to record the speech and a voice recognition library in order to interpret the speech. The voice recognition is a major requirement as the chess engine requires an input in order to play the game.

Section 3.2.3.1: Microphone

For the microphone, the project needs to have a microphone that will interface with a Windows or Linux operating system. It also has to be accessible by any interpretation API that is used by the software developers to ensure that the project works as the specifications have detailed.

The microphone that will be using will simply a standard Logitech microphone. Since the chess program only needs to understand a few word or phrases, a simple Logitech microphone will suffice. As such the microphone will only be responsible for recording voice commands from the player and sending it into the speech recognition software to interpret and translate the voice command into a command for the chess engine to understand. In addition, the microphone should have an on and off switch as such it should only be turn on by the players when they are issuing their respective commands.

Section 3.2.3.2: Interpretation APIs

In order for the machine to understand speech a voice interpretation library would be required. The voice library will be used to interpret the phrase spoken by the player and translate that into commands the chess program is capable of understanding. There are a few voice library APIs that would serve the required specifications. Two voice libraries found to fit the required specifications are Google's Speech Recognition API and CMU Sphinx.

Section 3.2.3.2.1: Google's Speech Recognition API

The first voice library that was found to fit the specifications needed by the project was Google's Speech Recognition API. The reason for considering this was that it is a powerful speech recognition library and it runs on Google's servers. Internet access is required in order to utilize it. As none of the project's specifications had any requirement of connecting to the internet, this speech recognition API will not be used. Even with the internet connection requirement, this speech recognition engine is already a powerful one which is more than capable of understanding and translating the voice commands given to it by the players. However, the internet connection requirement would require the project include either an Ethernet port or Wi-Fi, which puts a hamper the idea that it should not be tied down to anything but an electrical outlet for power.

Section 3.2.3.2.2: CMU Sphinx

The other voice library considered was CMU Sphinx, a voice recognition library developed by Carnegie Mellon University. This voice recognition library is capable of running under both Linux and Windows Operating System. The board will run either Windows or Linux, so CMU Sphinx fits that requirement. In addition, the library is installed on the system and no internet connection will be required. This allows the project to function without the availability of internet.

CMU Sphinx has two main versions, a main version written in Java known as Sphinx4 and a version written in C for embedded systems known as PocketSphinx. Both versions of CMU Sphinx share the same basic code along with the same English acoustic model. As such, deciding on which version to use will simply be a matter of testing discussed in a later chapter.

Section 3.2.4: Computational AI Algorithms

In computer chess, there is an array of popular algorithms that are considered standard requirements by most chess programmers. Whether the project runs on a completely customized chess engine or some open source program, it is clear that these algorithms will have to be included, and knowledge of their operation will be essential to the design team. The ones explained here are Minimax, Iterative Deepening, Quiescence Searching, and Null-Move Pruning.

Section 3.2.4.1: Minimax

Minimax is a popular algorithm used primarily in zero-sum games played against computers. A zero-sum game is one in which points awarded to one player come at the expense of the other player, therefore maintaining an overall balance between players. One example of a zero-sum game is, of course, chess. Minimax is essentially an operation on a tree that minimizes the worst-case scenario. Each level of the tree, which can have any integer as a branching factor, represents the options available to the player in a given turn. Because chess is a two-player game, each level of the tree represents the alternating turn order. Each branch in the tree represents a choice by the player.

The operation of the algorithm is as follows. The computer maps out all of the possible moves or game states up to the number of turns it wants to look ahead. Once the tree is generated, scores are assigned to the terminal nodes representing the relative value to the computer. For example, if each move in chess can be scored from 1 to 100, numbers closer to 100 might be better for the computer, and numbers closer to 1 might be better for the human. For a tree with a depth of 2, the computer would simply choose the node with the highest number, and take that branch (make its move). However, in games like chess, players must think more than one step ahead, or they are usually doomed to fail. Therefore, if the depth of the tree is 3, the branches to the terminal nodes represent the options available to the human player. The computer assumes the human will choose his or her own best move, and uses this fact to determine what the human will most likely do given each possible move by the computer. Yet again, if the computer wishes to look a further step ahead, the next level in the tree represents the moves the computer can make after the human as responded to its first move. This search can continue for as deep as the computer is instructed to go. It is now clear why the algorithm is named Minimax:

it is a series of alternating choices between a minimum number (best for human) and a maximum number (best for computer).

The following diagram shows the operation of Minimax on a game which is nearing its end. The numeric values represent the value of each move with higher numbers being better for the computer. A value of positive infinity represents a wining state for the computer, and negative infinity represents a winning state for the human.

The Structure of a Simple Minimax Search

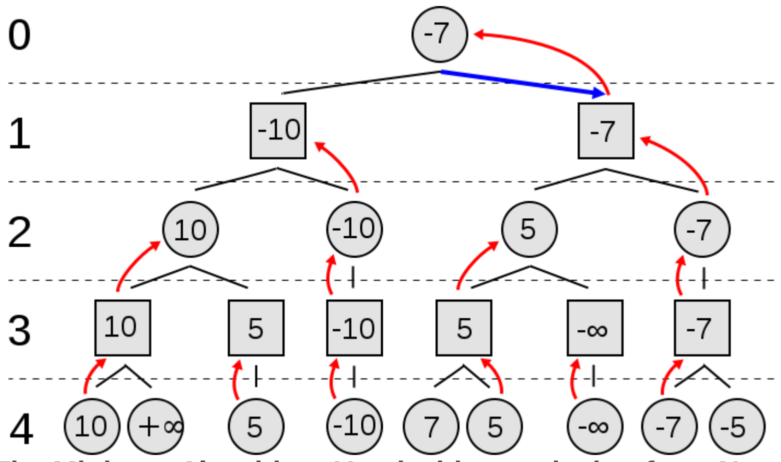


Figure 3-1 The Minimax Algorithm. Used with permission from Nuno Noguiera.

As shown in the diagram, the computer analyzes the tree starting with the terminal nodes. At level 3, it is the human's turn to move, and the human will always pick the move with the lowest value. Therefore, every node on level 3 is filled with the smallest value of the nodes below it. On level 2, it is the computer's turn, so the computer will choose all of the higher numbers. This same process occurs for level 1, and level 0 again represents the computer's turn. The computer chooses the move associated with the subtree with the greatest value because this number represents the worst scenario that could possibly occur. Note that a winning state for the human (negative infinity) is actually present in the subtree that the algorithm chooses, but the opportunity is only present on the computer's turn, so this can only occur if the computer intentionally makes that losing move.

Because the number of possible game states in chess is incredibly large, a method was derived to speed up this search. In some cases, it can be logically deduced that one subtree will always produce an outcome that is relatively worse than another subtree. Therefore, it would not change the outcome of the game to continue evaluating that subtree, meaning some unnecessary calculations are being performed. Specifically, if it is the computer's turn to play, and the algorithm finds a value in a subtree that is less than or equal to the value at the top of another subtree, the computer stops searching, and chooses the other subtree. This is because there exists a worse state in the subtree the algorithm is checking than the worst possible state in the other subtree already evaluated. Because the human will inevitably choose this state which is worse for the computer, the computer already knows that it should choose the opposite move. This process of pre-maturely halting an evaluation of a subtree is known as α/β Pruning, and it speeds up the computer's search dramatically.

Section 3.2.4.2: Iterative Deepening

Even with some alpha-beta pruning, the search space can still be massive. Iterative deepening is a technique to further reduce the search by rooting out decisions that the computer should not bother evaluating. For example, one possibility in a chess game might be to capture a defended pawn with a queen. Clearly, the opponent will then capture the queen, and the ultimate result is that the first capture was not worth the consequences. Even though a computer is unlikely to choose this idiotic move, it will nonetheless search the entire subtree generated by this move, and attempt to evaluate all of the possibilities, even if it clear at the start that the move will never be worth it. This sort of depth-first search is inefficient. A breadth-first search would not necessarily perform any better, as it requires substantial amounts of memory.

Essentially, iterative deepening retains the completeness of breadth-first searching, while taking advantages of the space-saving capabilities of depth-first searching. It does this by visiting every node in the tree up to a specified depth, then increasing the depth by one, and evaluating each subtree one level deeper than before. The space complexity is the branching factor of the tree multiplied by the depth of the shallowest goal (b*d), which is an improvement over the space complexity of a normal breadth-first search (b^d). The time complexity is the same as a depth-first search. Certain nodes will be visited multiple times, but this does not add much to the space and time used as the upper nodes of a tree are usually only a small portion of the total number of nodes in the tree, meaning that the overhead for this algorithm is relatively small.

Section 3.2.4.3: Quiescence Searching:

Quiescence searching is a strategy used to combat the Horizon Problem in many game engines. The Horizon Problem arises when the search space of computer moves is very large. In chess, for example, there are 20 possible opening moves, and about 30 possible second moves. So after both players have moved twice, there are about 360,000 possible game states. This number becomes incredibly massive after only a few turns, which makes it impossible for a computer to completely map out the structure of game using today's resources. Current chess engines can only search a depth of a few ply, where ply is defined as one move by a player or one level in the decision tree. Most chess engines only search about 5 to 10 ply. The Horizon Problem is that the engine cannot see beyond 5 to 10 moves because the search space explodes into a colossal number of nodes. The result is that the computer's greedy algorithm may end up choosing a move that looks attractive at the moment but may be detrimental in the future.

With quiescence searching, a heuristic can be implemented to favor certain types of moves over others. Moves that are not considered “interesting” are called “quiet”, and may not be searched to as great a depth as moves in a highly contested area on the board. This allows the computer to prioritize different subtrees based on an arbitrary heuristic as opposed to searching each one to an equal depth. This does not increase the efficiency of the search, but it helps the computer develop a better strategy.

Section 3.2.4.4: Null-Move Pruning

Null-Move Pruning is an extension of the alpha-beta cutoffs discussed earlier. Rather than searching through the entire tree to determine all of the cutoffs, the computer can guess if a cutoff is likely to occur. It does this by assuming that each move the computer makes should improve its position in the game. It compares a move that it is evaluating to the case in which the computer decided not to move at all, which is actually illegal in chess. In other words, if the current move under evaluation is worse than not moving at all, then it certainly would have produced a cutoff under normal search operations. This speeds up the search because only a shallow search is required to guess whether or not there is going to be a cutoff. These guesses are performed with acceptable accuracy.

Section 3.2.5: The Chess Engine

Creating a computer program that can play chess is a daunting task, even when trying to achieve only a rudimentary level of intelligence. Multiple chess engines abound on the internet, both closed and open source, and each one required either a team of professional developers working over a moderate period of time, or a single expert working over a long period of time to create. During research, different standards and abstract implementations of an original chess engine were discussed, but ultimately it was decided that Magic Chess should use an open source chess engine, due to time constraints and the complexity of the problem.

This section documents the benchmarking and evaluation process used to determine which open source chess engine would be most appropriate. Each page is a short profile of one of the engines that were investigated. Each profile contains a summary of the operation of the engine, a discussion of its positive and negative attributes, and benchmarking data, such as response time, amount of memory used, amount of the processor’s capacity used, etc. For consistency, each engine was tested with the same hardware. At the end of this section is a conclusion detailing what factors ultimately determined which engine to implement in the final product. All credit is given where due, and express permission will be obtained in writing from the creator(s) of the engine used in the final product.

In determining which chess engine to use, many key factors were considered. The first was **robustness**. Robustness refers to a collection of characteristics vital to the operation of the program. These include accuracy and error tolerance. In a more volatile engine, one mistake in data entry could have detrimental consequences on its operation. The second factor was **adaptability**. It was unlikely that a chess engine that perfectly matched the needs of the project was already in existence. Each engine was judged on its potential to be altered, how easily features could be removed, and how easily features could be added. The third factor was **availability and readability of documentation**. Engines with more extensive documentation were preferred over ones on which little information aside from the source code could be found. The fourth factor was **use of resources**. Magic Chess should be as portable and as cost-effective as possible, so it should use a minimal amount of hardware. Performance analysis was completed on each engine via a series of benchmarks as described above.

The engines tested were chosen from a list of open source programs on chessprogramming.wikispaces.com. The eight engines were selected based on summaries, the quantitative performance results reported, and a sample viewing of some source code. They are called, in the order presented hereafter: Micromax, TSCP, Faile, Gerbil, Crafty, Fruit, Glaurung, and Stockfish. These chess engines seemed the most likely to score well based on the above criteria.

When possible, each engine was tested in both command line form and in either the WinBoard or Arena 3.0 (UCI) GUIs. The chess engines were tested on a personal computer for preliminary evaluation. The processor used was an AMD Athlon II Dual-Core M300 with a clock rate of 2.00 GHz. 3.75 GB of memory were available for use, and the operating system was the 64-bit version of Windows 7. While these specifications slightly exceed those of the hardware intended to be used in the chess board, they are still within a ballpark range.

Section 3.2.5.1: MicroMax

MicroMax is an open-source chess engine developed by Tim Mann. Written in C, it is the smallest known chess engine in existence at only 120 short lines of code. Because of its small size, it was of great interest due to the need for low-cost software. Even though the code is very primitive, the engine implements many intelligent strategies. One is a recursive negamax search, which is a variation on the popular minimax algorithm in most chess engines. In addition, it implements an all-capture MVV/LVA quiescence search, an internal iterative deepening algorithm, a hash table scoring the score and best move for fast lookup, and null-move pruning. Naturally, it also checks the legality of each move and supports full Fédération Internationale des Échecs rules (also known as FIDE or the World Chess Federation). Given the minimalism of the code, it is impressive that MicroMax can do all of these things. However, just because it takes up a small amount of space on a disk does not necessarily mean it uses the smallest amount of RAM. Programs of only a few lines can usurp all the available memory on a system if they are not written efficiently. Thus, it was necessary to measure as accurately as possible the amount of memory MicroMax and the other engines use. When possible, this was done with Arena 3.0, which reports relevant data at the bottom of the window. Otherwise, the data was taken from the Windows Utility Monitor.

MicroMax has player versus player and player versus computer capability. At each turn, the player makes a move, or the player instructs the program to determine a move. This means that the computer and the player can frequently switch sides at the whim of the player. This is not a useful function in relation to the project, but it does make the engine more versatile. In earlier versions, MicroMax was not very robust. In testing, it was necessary to play a few opening moves manually before invoking the negamax algorithm. Otherwise, the computer would attempt illegal moves or else do nothing. In version 4.8, this problem seems to have been resolved, so the engine should have no problems with robustness.

Adaptability, on the other hand, is something this engine lacks. Even though the code appears simple, its compact nature means a single edit could have far-reaching consequences. To adapt the engine for the necessary purposes, it would require being able to understand obfuscated code whose author sacrificed readability for size. All variable names are single letters, and use of whitespace is minimal. However, each line has a short comment with an explanation of what it does, which could help.

Tim Mann has included a large amount of well-written documentation on his website. Each aspect of the intelligence part of the program (e.g. iterative deepening or quiescence searching) is explained in a straightforward manner, and the lines that accomplish each of these things are highlighted. This is a helpful resource for anyone seeking to understand how Micromax operates.

	CPU (approx %)	**Memory (MB)**	**Avg Response Time (sec)**
Player's Move	<1	199	<1
Computer's Move	48	199	19

Table 3-3 : Resources used by MicroMax

Section 3.2.5.2: TSCP

TSCP stands for Tom's Simple Chess Program. Written in C in 1997, the size of the source code greatly exceeds Micromax, but it is still relatively compact. In addition to all of the standard features listed in the description of Micromax, TSCP also includes an opening book, which is a list of standard opening moves and good responses to them.

TSCP scores well in the robustness category. During preliminary testing, the computer never attempted any illegal moves, nor was it confused by nonsensical data. When the move entered was illegal or nonsensical, the TSCP simply printed "illegal move" and waited for another command. The computer can be toggled on and off at any point via the commands “on” and “off”. The advantage over Micromax is that with Micromax a user must tell the computer to make a move each round. With TSCP, if one wishes to play against a computer, it only requires one command at the beginning of the game.

Adaptability is something else TSCP excels at. According to the Readme file included in the software, TSCP was written to be didactic. Everything is commented very well, and it seems that most experienced programmers would be able to read through the source code and modify it to fit their purposes. The iterative deepening algorithm seems somewhat primitive, so if it were necessary to implement a varying difficulty level, the depth of the search could be altered based on the desired difficulty level.

Unfortunately, beyond the included Readme file, not much additional documentation on TSCP could be found. Therefore, the comments in the code play the most important role in a user’s ability to understand how the engine works.

	CPU (approx %)	**Memory (MB)**	**Avg Response Time (sec)**
Player’s Move	<1	2.1	<1
Computer’s Move	5	2.1	<1

Table 3-4 Resources used by TSCP

Even without ample documentation, TSCP’s ability to perform everything necessary in a short amount of time and still retain its minimalist qualities makes it a very likely candidate for use in our final product.

Section 3.2.5.3: Faile

Faile (version 1.4) is a chess engine written in C in 2000 by Adrien Regimbald. It is published under an open MIT license, which means anyone can use the software for any purpose provided the original copyright agreement remains in the code and the author is duly credited. The engine consists of 8 .C files are 4 header files. It implements all the standard algorithms found in most chess engines, and also seems to generate its own book of moves. How it accomplishes this is difficult to understand without extensive study. Like similar engines, it supports WinBoard and XBoard. It also has a command line mode, with a well-readable ASCII representation of the board.

Faile scores well in the robustness category, as no incoherent data entered produces any errors. The engine checks to see if the notation is correct, and if it is, it then checks move legality. If the input does not pass these two criteria, the engine continues to wait for valid input. It has not been observed to produce any errors or attempt any illegal moves. When the computer moves, it takes a noticeable amount of time to perform the calculation, but not terribly long. If the computer opts to use the book, the move time is almost instant.

Faile does not include player versus player capability, which means adaptability is an important factor. It must be possible to edit the source code to allow the user to select a one or two player game. Based on a quick viewing of the main function, faile.c, it seems relatively difficult to do this. The program was written without consideration for this additional functionality, which could make this a daunting task. For this reason, it would be far simpler to go with an engine that includes a more dynamic implementation of this function, such as TSCP, which allows the computer to be toggled on and off at any time.

Like many popular open source chess engines, Faile was written to be didactic, and is easily digestible. Some documentation exists covering the hash tables and opening book concepts in general, but there is nothing resembling a true manual. The code has some comments, but not nearly as many as TSCP, making this a less attractive candidate in this regard.

	CPU (approx %)	**Memory (MB)**	**Avg Response Time (sec)**
Player's Move	<1	6.1	<1
Computer's Move	50	6.1	18.62

Table 3-5 Resources used by Faile 1.4

Section 3.2.5.4: Gerbil

Gerbil is a chess engine written by Bruce Moreland in 2001. It is written in C and more sizeable than the previous chess engines discussed. It consists of 15 .C files and 1 header file. It is published under the GNU General Public License, meaning it is not necessary to obtain permission to use Gerbil. It was intended to be run using WinBoard. It will still function in command line mode, but it does not provide any sort of representation of a chess board, which is why the author warns the user at the beginning to “find a chess board unless you like to play blindfolded.” It does not support player versus player mode, so this would have to be added manually.

Gerbil does not score as well in the robustness category as the other engines. It appears to work fine with WinBoard, but in our chess board, WinBoard will not be used for obvious reasons. Therefore, it is of paramount importance that the engine chosen operate smoothly in command line mode. When starting up Gerbil, the engine instructs the user to enter “protover 2” before doing anything such as invoking a new game. Why this is necessary is unknown, but the engine still appears to work whether or not one enters this command. The difference is the notation the engine provides when carrying out its move. This uncertainty in the program’s operation should give a user pause, as it makes the engine seem somewhat unstable. In addition, the engine is described by the documentation as having decent null move pruning and hash tables, but only a rudimentary evaluation, meaning the engine may have some difficulty determining which move is best for it in the long run, or the evaluation itself may use substantial resources. Moreover, many of the comments left by the authors imply that many functions were coded to be merely “good enough” and not necessary optimal.

Because Gerbil only supports one human player, it would be necessary to modify it to support two. Given its size, this may prove a daunting task. The primary function alone is nearly 1000 lines of code, and it is not obvious which part of the code controls switching between the user’s turn and the computer’s turn. Because of this, the project would have to rely heavily on availability of documentation. Additionally, because the author describes Gerbil as only having a “rudimentary” evaluation function, it would be necessary to improve upon this to implement multiple difficulty levels. This is far more difficult than starting with a good evaluation, and making easier settings from the more difficult one by limiting the depth of the search.

Luckily, Gerbil has some helpful documentation. Many topics are covered, such as customizing the compilation, how to customize its operation with an initialization file, tips for editing the main source code and the opening book, various commands and what they do, and an explanation of a version of Gerbil called “Loser’s Chess”, which appears to be of no use to our project.

	CPU (approx %)	**Memory (MB)**	**Avg Response Time (sec)**
Player’s Move	50	12.2	<1
Computer’s Move	50	12.2	5

Table 3-6 Resources used by Gerbil

Section 3.2.5.5: Crafty

Crafty (version 23.4) is a chess engine written by a team of programmers led by Dr Robert Hyatt at the University of Alabama at Birmingham from 1996 to the present. It is larger than any of the other engines previously discussed, consisting of 42 .C files, 1 .CPP file, and 10 header files. It includes such features as a principle variation search, null move pruning, late move reductions, and static exchange evaluation.

Robustness is a difficult factor to evaluate given Crafty’s feature of constantly analyzing moves. Upon running the executable included with the source code, the engine appears to ask for a move from the player (white). After the player enters this move in algebraic notation, the engine begins to evaluate its options to make its first move. Once it determines its move, it then predicts what move the player will make next, and starts a new evaluation based on this. Meanwhile, the engine is waiting for the player's next move. After making its move, Crafty Chess prints out the move it predicts the player will make. If the player chooses to make this move, the response time appears to decrease, as the computer has already spent time evaluating this situation. If the player does not make the predicted move, the response time increases for the opposite reason. The engine responds well to gibberish input by declaring it an illegal move. Unfortunately, it does not seem to be able to recognize that moves such as "a3a9" (i.e. moving an imaginary piece to an imaginary location) are not valid moves. The engine does not declare these moves illegal, and instead takes another turn analyzing this nonsensical input. The odd thing about this situation is that, on other occasions, the engine would recognize such a move as illegal. Regardless, this unpredictable and unexplained behavior would be detrimental to a project such as this, so it is unlikely that Crafty will be used on that basis alone. Additionally, Crafty does not provide a representation of the board in command line mode, so that made it more difficult to test.

Crafty, on the whole, does not seem very adaptable. Its primary purpose seems to be to offer a chess engine that is constantly analyzing the board, and predicting a user's move. Much work has gone into making the computer a formidable opponent. Indeed, its "main" function is structured around this objective, so it seems like it would be difficult to, for instance, add a two-player mode. In addition, based on the information Crafty prints out as it evaluates each move, the decision-making algorithm seems highly complex, and modifying to implement different difficulty levels would take a lot of work.

It was difficult to find a lot of online documentation for Crafty. There are at least three different "official" websites on which to download the source code. To run Crafty, it was necessary to hunt for the correct executable as the files were scattered across all of the websites. Additionally, each website lists many different versions as being the most recent version, which further complicated its initial testing. To its credit, Crafty includes much explanation in the "main" function. There are literally thousands of lines of comments explaining various aspects of the engine as well as documenting the different versions. It was this text that was used to determine which version of Crafty was most up to date.

	CPU (approx %)	Memory (MB)	Avg Response Time (sec)
Player's Move	55	24.2	<1
Computer's Move	55	24.2	30.9

Table 3-7 Resources used by Crafty

Section 3.2.5.6: Fruit

Fruit (version 2.1) is a popular open source chess engine written in C++. It consists of 33 .CPP files and 33 header files with identical names. Originally written by Fabien Letouzey, it has since evolved into a closed-source project, with 2.1 being the final open release. Fruit 2.1 is a Universal Chess Interface engine only, which means to test it in WinBoard, it is necessary to use Polyglot, which is essentially a UCI-to-WinBoard adapter. Fruit does not seem to have a command line mode. Because this is not the only UCI-only engine under testing, it seems relevant to explain the differences between the UCI and WinBoard protocols, and the advantages and disadvantages they have in terms of design. WinBoard is a protocol designed by Tim Mann, the same programmer who created the MicroMax engine. Originally, it was simply a GUI used to interface with GNU Chess, but after its success, many people requested that it be extended to support other chess engines. Because it was not originally designed for this from the start, its support for other engines required ad-hoc functionality, resulting in an interface that was less optimal than it could have been. Its creator has said that it is in need of a massive overhaul, but it works pretty well in general. UCI is an alternative protocol that came later, and was written by Letouzey, mentioned above as having also created the Fruit engine. It was created to be an improvement over the design flaws that plagued WinBoard. However, UCI is controversial. Many programmers (such as Dr. Hyatt, creator of Crafty Chess) refuse to work with UCI, citing concerns that UCI takes control of certain functions and decisions that should be left to the chess engine, not the GUI. According to Dr. Hyatt, these decisions can be potentially game-altering. Regardless, Fruit was tested in WinBoard, as this protocol was deemed to be the easiest to work with.

With only the WinBoard version to test Fruit, it was difficult to assess the robustness of the engine. It was not possible to attempt illegal moves or input gibberish and gauge the engine's response. However, Fruit worked very well in WinBoard, and is a very popular engine, suggesting that it is indeed robust enough for common use.

Fruit is roughly as adaptable as most of the previous chess engines. It does not include a two player mode or varying difficulty levels.

Despite its popularity, documentation for Fruit 2.1 is not extensive. The author includes a basic readme file, and the code is somewhat commented, but there exists nothing resembling a proper manual.

	CPU (approx %)	**Memory (MB)**	**Avg Response Time (sec)**
Player's Move	2	68.2	<1
Computer's Move	40	68.2	15.4

Table 3-8 Resources used by Fruit

Section 3.2.5.7: Glaurung

Glaurung is a chess engine written in C++, first developed by Tord Romstad in 2004. It also uses the UCI protocol. Because Fruit was already configured in the distribution of WinBoard we used, we simply used WinBoard. For Glaurung and other UCI engines, it is necessary to use a UCI GUI. The one used here is Arena 3.0. Most chess engines are not stand-alone programs, so they usually fall into one of the two categories: WinBoard/XBoard or UCI. The version of Glaurung tested was 1.2.1. There are much newer versions in which the engine has undergone a complete underhaul, but after much searching, the only source code that turned up was for GlaurungServer 2.0 and beyond. GlaurungServer is a program that allows a user to play via an internet connection while the actual move calculation is performed on a remote computer. Adding networking capabilities to the chess board seemed infeasible, which was the rationale behind using the older version of Glaurung.

When used with the intended interface, Glaurung is a robust engine. Most of the moves it attempts are reasonable. However, the response times can be slow, and it appears to lack an overall strategy. Even for moves that are obviously beneficial, such as capturing a high-ranking piece with a much lower-ranking piece, the program can often take a while to finish its searching function. However, this would only slightly affect the operation of the hands-free chess board. The most important thing is that it can complete a game and maintain legality.

Glaurung looks as if it would be slightly more difficult to adapt than other chess engines. The “main” function invokes a large header file that seems very complex. Attempting to add player versus player mode or multiple difficulty levels could prove difficult. It was also noted that most of the above chess engines force a user to play white, while the computer plays black. This is a superficial feature, but it would allow the user to have more control if they could choose what color they wanted to play as. While examining the source code, it was difficult to locate the lines of code that assign the user the pieces and the computer the black pieces. For this reason, it was difficult to evaluate its adaptability with regards to this additional feature.

Like the other chess engines, Glaurung comes with a basic readme file, but most of the thousands upon thousands of lines of code are completely devoid of any comments. This would make it difficult to interface with the other components of the system, not to mention to adapt it to the appropriate purposes.

	CPU (approx %)	**Memory (MB)**	**Avg Response Time (sec)**
Player’s Move	<1	138	<1
Computer’s Move	55	138	14.73

Table 3-9 Resources used by Glaurung

Section 3.2.5.8: Stockfish

Stockfish is a UCI engine developed by Tord Romstad, Marco Costalba, and Joona Kiiski. It is actually a derivative of Glaurung 2.1, so the two engines share many similarities. Stockfish is known among chess programmers for its strength. It frequently does well in online tournaments and is often found at the top of most rating lists. Its difficulty level has been compared to Rybka, which is an internationally famous chess engine under a proprietary license. Like other UCI engines, Stockfish comes with its own initialization file for Polyglot, which means it can run in both WinBoard and UCI GUIs like Arena. Here, it is tested in Arena.

Stockfish is a solid engine. It is certainly comparable to strong engines like the one found at chess.com. It terms of pure robustness, it is on par with the other open source engines tested so far.

Adaptability is an important factor in this case because the Stockfish engine is quite difficult to play against. While most chess enthusiasts would likely have little trouble with it, it may be difficult for children, which is an important group in the target audience of this project. Therefore, in the final product, it may be necessary to decrease the difficulty level significantly for more enjoyable play. This was merely a desired feature in the other chess engines, but would be necessary in this one. Another interesting thing to note about the adaptability of Stockfish is that its creators designed it to be fully compatible with custom opening books. It is both possible and encouraged to swap out opening books by placing a .bin file in Stockfish's directory, provided the file is in the correct format.

Due to its popularity, Stockfish has, by far, the most documentation of any chess engine tested in this report. It has one official website that is updated frequently, and its documentation is separated into multiple articles that are so numerous that the website actually includes a search bar for the interested user or developer. This is certainly better than the decentralized distribution, a simple readme file, and a few sparse comments, which seem to be the standard for other engines such as Glaurung and Crafty. The articles cover vital topics such as how to compile the engine on Windows, OS X, and Linux, how to swap out opening books, and how to deal with high CPU usage or overheating. These articles prove especially helpful because, as shown in the table, Stockfish uses quite a bit of resources due to its difficulty and depth of search.

	CPU (approx %)	**Memory (MB)**	**Avg Response Time (sec)**
Player's Move	<1	138	<1
Computer's Move	100	138	27.4

Table 3-10 Resources used by Stockfish

Section 3.2.5.9: Conclusions

After a great deal of chess games, rooting through code and documentation, and benchmarking, it appears that TSCP or Tom’s Simple Chess Program is the engine that would be most appropriate for use in this project. This is due mostly to its incredibly small size and the fact that player versus player mode is already included, reducing the need to modify the code. The only other engine that did this was Micromax, and while Micromax was even smaller than TSCP, its long response times (usually greater than 60 seconds), its questionable operation, such as occasionally attempting illegal moves, its high level of memory consumption, and the need to prompt the computer to move every single turn were unattractive qualities.

Due to its smoother operation and abundance of documentation Stockfish was a close second choice. The main reason it was not chosen above TSCP was its size, which means it would not only use more resources than TSCP, but it would also be more difficult to adapt. Again, adaptability was critical in Stockfish due to its difficulty level. In the case that it is too difficult to adapt either chess engine, TSCP would be a better choice because it is easier for inexperienced players to beat. However, it was proposed that in the event that implementing a varying difficulty level in TSCP proves too difficult, it was always possible to run TSCP if a player wanted a easy match, and Stockfish if a player wanted a difficult match. For a match with medium difficulty, Faile would be a good third choice, as it performed up to all standards and used resources very sparingly. Either way, we feel that our research has produced many good options, with freedom in choosing between them.

The other chess engines were thrown out due to more glaring problems. Crafty, for instance, appears to be more of an experiment in human prediction than anything. It lacks essential documentation and its “official” website is rarely updated. Different versions of it can be found on different websites, which is in contrast to the extremely centralized distribution of an engine like Stockfish. Some engines such as Fruit and Faile ran fine, but did not have many positive characteristics in regards to our decision-making process.

One of TSCP’s drawbacks is that it is the only engine on the list that requires express permission from its creator to use. While Tom Kerrigan rarely disallows people to use his program as long as they stay within his guidelines, our use of this engine is still contingent upon his approval. In the event that permission is denied, Stockfish will still make a very good second choice, even though this will increase the response time and resource usage. Additionally, Stockfish is cross-platform and can run in Windows, OS X, and Linux, which allows much freedom when it comes to the software that will run the entire board.

Section 3.2.6: Motherboard and Accompanying Hardware Research

Section 3.2.6.1 Types of Motherboards

In order to run the AI component of the project a motherboard was required. The project needed a motherboard that mainly was affordable and small enough in size to fit comfortably in the chess board, and light as to not add any unnecessary weight to the chess board. Also the project would require enough RAM for the CPU to run the program quickly so there would be no additional delays. There are four main motherboard form factors considered for use in the project. The form factors are ATX, MicroATX, Mini-ATX and Mini- ITX these four were chosen generally by the general availability they are sold and by their common use in personal computers today. The standard ATX form factor motherboard is 12” x 9.6” in size and would be rather large to fit in the chess board, also the software of the project will not require a motherboard with that much space for a lot of RAM or a powerful CPU [12]. Another form factor available is the MircoATX which is slightly smaller than the standard ATX factor but still not as small as needed to make the chess board small and light [13]. The Mini-ATX is 70% smaller than the standard ATX at 5.9” x 5.9” and would fit comfortably inside a large chess board along with the other hardware components [14]. However, the Mini-ITX is the smallest of the four at 6.7” x 6.7” inches and would be the desirable motherboard form factor for the project [15]. The table below shows the form factors considered and the corresponding sizes.

Form Factor	Size [inches]	Advantage
ATX	12 X 9.6	
MicroATX	9.6 X 9.6	
Mini-ATX	5.9 x 5.9	
Mini-ITX	6.7 x 6.7	✓

Table 3-11 Motherboard Form Factors

Section 3.2.6.2 CPU and RAM

The AI component of the software of the project would need up to 2 GB of memory which would keep the cost of the RAM for the motherboard low. Other than the memory requirements of the software the only factors involved with the RAM is the type of PIN requirement the motherboard comes with. The CPU should be affordable but needs to have a good operating frequency to run the chess engine fast enough to not have the user waiting long for a response from the AI. Another factor to consider is the design of the CPU is determined by the socket type on the motherboard.

Section 3.2.6.3 Power Supply

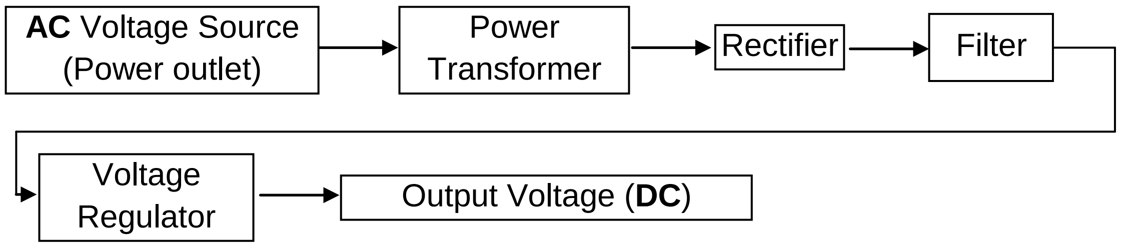
The power supply required for the motherboard is determined by the motherboards manufacturer. Mini-ITX motherboards typically require a ErP/EuP ready power supply. ErP stands for Energy-related Product(s) and EuP stands for Energy-using Product(s). For a power supply to be considered ErP/EuP ready it must be under 1.00W in off mode condition [16]. Other factors to consider when choosing the motherboard power supply will be the motherboard's power pin connector requirement. Most power supplies come with 20 (+4) pin connection. Also, a power supply that could power the motherboard and all the other electronic components in the chess board would be desirable. This would require a motherboard that came with multiple SATA power connectors.

Section 3.2.7: Power System

The chess board will come with multiple electronic components that will require a DC voltage supply. The power supply purchased for the motherboard may not have enough power pin connections or wattage to power all the components. Using the motherboard power supply is the desired result but in case the power supply chosen is found to be inadequate, AC-to-DC converter circuits will need to be researched and designed to ensure enough power is supplied to all components.

Section 3.2.7.1 AC to DC conversion

Below is a block diagram showing the basic circuit breakdown of converting the AC voltage from any basic wall outlet into the DC voltage needed to power the electronic components inside the board. The circuit will require an AC outlet plug, power transformer to step down the 120 voltage from the wall, a diode filter to limit the polarity of the signal coming from the outlet, another filter to begin turning the AC signal into an almost complete DC signal, and a voltage regulator to make the signal constant and possibly a resistive load for added protection before the DC reaches the component. [16]



Section 3.2.7.2 AC power cord

To begin the research on the AC to DC conversion circuit, the first thing to consider is the type of plug for the chess board to connect to any available power outlet. The power cord should come with a standard plug that fits into any basic American 15-A wall outlet and has exposed wires at the end of its cable. The power cord should be long so players of the chess board have a flexible radius to play from the outlet they have plugged the cable into.

Section 3.2.7.3 Power Transformer

The transformer is used in the converter circuit design because the input voltage from typical household wall sockets is about 120V AC, this voltage is comparatively higher than the voltage required to power the microcontroller, motor controller, and electomagnet of the chess board. Therefore a step down transformer is used to bring the voltage down to a desired and manageable level to be used to power the electrical components.

The number of turns on the primary and secondary windings of the transformer affects the output voltage from the transformer. The desired output voltage of the transformer chosen will need to be at least 12 volts because that is the highest voltage requirement out of the two electronic components in the chess board. The minimum requirement is set by the stepper motor controller which will need 12 volts to operate. Therefore, a transformer with a turns ratio of 10 to 1 should be used. [16]

The circuit design will either consist of a full wave rectifier circuit or a full wave bridge rectifier circuit. A basic full wave rectifier circuit requires a step down transformer with a center tap on its secondary windings. The center tap will half the output voltage from the transformer and would require two diodes. However, if the transformer is center-tapped the transformer will need to have twice as many turns as the bridge rectifier transformer to obtain the same output voltage. If the rectifier circuit is a bridge rectifier than the transformer no longer needs a center tap on its secondary windings. [17]

Section 3.2.7.4 Rectifier

The diode filter will need to be a full wave rectifier. Full wave rectifiers can consist of two standard diodes with a center-tapped step down transformer or can consist of four diodes in a layout specified as a bridge rectifier. A bridge rectifier is more desirable to use when converting an AC signal to DC signal because it does not require a center-tapped step down transform and will cost less to build. The rectifier in the circuit converts the alternating current signal polarity to a constant positive or negative polarity.

Section 3.2.7.5 Filter

The full wave bridge rectifier will change the polarity of the signal from alternating to a constant positive or negative charge. The filter is responsible for smoothing out the positive or negative sinusoidal wave from the rectifier. Usually the filter consists of a large electrolytic capacitor that will smooth out the “humps” of the positive or negative signal into small ripples.

Section 3.2.7.6 Voltage Regulator

Even though most of the signal is almost a steady DC signal, the variations in voltages could cause harm the electronic components they are powering, especially if the voltage goes under the minimum supply voltage for the microcontroller or motor controller. A voltage regulator can consist of a Zener diode and a resistor to limit the current though the Zener diode. The resistor also disposes of the unnecessary voltage from the rectifier and capacitor. The resistor and Zener diode take the ripple signal input from the rectifier and ideally smooth the signal out completely into the desired voltage output. [16]

Section 3.2.7.7 Expected Output Voltage (DC)

After the proper transformer, rectifier, capacitor and voltage regulator are calculated and properly built, the expected voltage should meet the needed voltage for the microcontroller, motor controller and electromagnet. However both components come with different power supply requirements. If necessary, two AC-to-DC circuits will be needed to built and tested if the motherboard power supply chosen is not enough to power the microcontroller, motor controller and electromagnet.

Section 3.2.7.8 Risk of building AC to DC converter

After extensive research it was realized that building an AC-to-DC converter circuit would not only be difficult but would take a lot of trial and error to obtain the desired results from the circuit. Available simulation such as Multisim would mimic ideal components and also does not have available commercially available components in the program to design with. Most of the component characteristics like the transformers, diodes, and resistors are not similar to what can actually be purchased online or in the store. Also, any simulation does not account for the environment in which the circuit will be performing and multiple electrical components (resistors, capacitors, diodes, and transformers) will need to be purchased once the testing of the power supply reveals any errors unforeseen by the simulation. The risks involved with building the projects AC adapter will be seriously considered as well as purchasing an already built AC adapter that comes with various output voltages.

Section 3.8: Operating Systems

A major requirement for the chess project is an operating system to run the chess engine on. Although a chess engine can be run on a microcontroller, it's much more efficient to run this on a CPU chip. As the requirement for the project requires using an Intel Atom chip, each operating system considered for the project must be able to run on the CPU without taking up too many resources. In addition, the computer will run with 4 GB RAM (expandable up to 8 GB), which is plenty of memory to run just about any operating system. The operating system's installation size must also be taken into consideration as the project will be utilizing a 60 GB SSD to store the operating system, the voice library, and the chess engine. The specifications of the computer system are as follows:

- Intel Dual-Core Atom Processor D525 (1.8 GHz)
- 2 x DDR3 DIMM slots
- Intel Graphics Media Accelerator 3150
- Pixel Shader 2.0, DirectX 9.0
- 5.1 CH HD Audio

The specifications listed above are the important specs of the computer system which will be compared with the requirements to run operating systems that was researched and considered to run on the computer system.

Section 3.8.1: Windows

One of the first operating systems that fits the requirements and specifications of the project that came to mind was the Windows Operating System.

	32 bit (x86)	**64 bit (x64)**
Processor	1 GHz or faster	1 Ghz or faster
RAM	1 GB	2 GB
Hard Disk Space	16 GB of free space	20 GB of free space
Graphics	Directx 9 graphics device with WDDM 1.0 or higher driver	

Table 3-12 Minimum Requirements for Windows 7 [22]

The specifications of the computer unit that will be running the chess AI are more than sufficient to run the Windows 7 operating system. Although compared to the other operating systems, it has a higher requirement in order to operate. The minimum requirements are listed as the minimum amount of resources necessary to run the operating system, but nothing is mentioned about how efficient and capable the system is running Windows 7. So other operating systems are taken into consideration. In addition, the cost of a license for Windows 7 makes it much more attractive to consider other operating systems that have a much lower cost of license. However, one advantage of running Windows 7 on the project is that there are multiple solutions to running the CMU Sphinx voice library and a chess engine.

Section 3.8.2: Linux

Another operating system considered for the project is the GNU/Linux operating system. The reason for considering this operating system was that Linux is flexible and is open source software. This reduces the cost of the project as there is no license to buy and the open source license allows the system to be modified according to the specifications of the project. The slight disadvantage in running a GNU/Linux distribution is that since it’s not as popular as the Windows operating systems, some solutions may be more difficult to find and solve. Other than the slight negative, the primary consideration is determining which of the GNU/Linux distributions to use.

Six major GNU/Linux distributions were researched for usage in the project. The Linux distributions were chosen based on popularity and familiarity with the system. The six Linux distributions chosen are lists as follows:

- Debian
- Ubuntu
- Fedora
- Arch Linux
- Linux Mint
- Xubuntu

The reasons for considering each of the six Linux distributions will be discussed in the following subsections.

Section 3.8.2.1: Debian

The first GNU/Linux distribution considered according to the requirements of the project was Debian. The first reason for the consideration of this Linux distribution is the cost of licensing is absolutely zero as with most Linux distributions in general. Another major reason to consider this Linux distribution is the extensive documentation involving the deb package system, allowing files and program to be easily installed. In addition, one of the project members is already familiar with the Debian packaging system and already has experience in working with Linux distribution that uses the deb package management system. The major reason as to the consideration of this operating system is simply because of its lower hardware requirements compared to Windows.

The minimum requirements for running Debian on a computer system are as follows [23]:

- 1 Ghz Processor
- 128 MB RAM (minimum)
- 1 GB Hard Drive Disk free space

The recommended requirements for running Debian are as follows [23]:

- 1 Ghz Processor
- 512 MB RAM
- 5 GB Hard Drive Disk free space

The computer system has more than enough resources to run the Debian Linux operating system. In addition, the Debian Linux is supported on multiple hardwares, often just only needing to be able to run the Linux kernel [24]. As such this allows the computer hardware to focus more on the chess engine and voice library instead of running the operating system, which would produce a much better response time.

Section 3.8.2.2: Ubuntu

Another GNU/Linux distribution that was researched was Ubuntu Linux. There are a few reasons as to why this GNU/Linux distribution was considered. As with all other GNU/Linux distributions, it shares the main purpose of being free and open source software, which would be there is no cost in licensing. In addition, a member of the project team is familiar with this Linux distribution as it is based on Debian and uses the same deb package management system. Another reason is that it has lower requirement than Windows in order to run the system. The requirements for running Ubuntu Linux are as follows:

	Minimum	**Recommended**
RAM	64 MB	512 MB
Hard Drive Disk free space	1 GB	5 GB

Table 3-13 Requirements for running Ubuntu [25]

Although, there was not an exact listing of minimum processor speed listed, the CPU in the computer system is more than enough to run Ubuntu, as it is capable of running Windows 7. The other remaining reason why this distribution is considered is because it is advertised to simply work out of the box. This will mean there is less time spent working on configuring the system drivers, making sure it works with our hardware, and more time on compiling and implementing the voice library and chess engine. The final reason was that this distribution is based off Debian Linux. As a result, it makes this distribution extremely attractive due to the fact it is just like Debian, but without the most of the hassle of setting up a GNU/Linux operating system.

Section 3.8.2.3: Fedora

The third GNU/Linux distribution considered was Fedora and the reason is different from the previous two Linux distributions. As a GNU/Linux distribution, it shares many of the advantages of the previously mentioned distributions in terms of cost and licensing. In addition, it considers not only using completely free and open source software but also the latest software possible [26]. As considered, the requirements for running Fedora Linux are as follows [27]:

- 400 MHz or faster processor
- 768 MB RAM, 1GB RAM (Recommended)
- 10 GB hard disk drive free space

This brings a nice advantage to the project as this distribution has a very low processor requirement. Although, this really is not an issue as the processor the project uses is more than capable of running all operating systems mentioned so far. Even though the rest of the requirements are no issue for our computer system to run, the disadvantage to our project running this Linux distribution is that as Fedora often runs the latest software, it is possible that there are some bugs for which patches have not yet been developed. This could cause issues to arise involving integration of our systems. However, the project member that is familiar with the previous two Linux distributions is also familiar with this particular Linux distribution and its RPM package management system. As such, it is among the operating systems considered and researched on running the computer component of the hardware.

Section 3.8.2.4: Arch Linux

The fourth GNU/Linux distribution considered was Arch Linux. The main reason why Arch Linux was considered as an operating system to run on the computer system is because Arch Linux allows the user to set up the system any way the user wants. The major advantage this provides is that the operating system will not come with many unnecessary programs that most Linux distributions and other operating systems often load into the install. This allows the operating system to only be loaded with the programs and software needed to run the Sphinx voice library and chess engine without the need to worry about conflicts and bugs with other software that are not needed and would not be used in the project. As such, the requirements to run Arch Linux vary based on how the operating system is set up. The only major requirement is that 64 MB of RAM is needed for a basic installation [28].

With the specifications of the computer system, the computer system should be more than capable of running Arch Linux as it has proven to be more than capable of running all previously mentioned operating systems and Linux distributions. The only major disadvantage that comes with using this operating system is that this system is installed from scratch. Everything in the operating system must be loaded and installed manually. As such, this can become time consuming considering the amount of packages that must be loaded along with extra time based on errors or user mistakes during the installation and setup. The major advantage is that the system is set up exactly as required for usage of the project and nothing more. This allows a very efficient use of resources, allowing the computer hardware to put more resources towards the other components of the project rather than having a large overhead just from running the operating system.

Section 3.8.2.5 Linux Mint

Linux Mint was another GNU/Linux distribution that was considered and researched for use as an operating system for the computer component. The first reason as to why this Linux distribution was considered is the cost of licensing is completely free as with most other Linux distributions. In addition, this Linux distribution is based off Ubuntu, sharing the same exact deb package system as Ubuntu. As such binaries designed for Ubuntu also work in Linux Mint.

The minimum requirements for running Linux Mint are as follows [29]:

- A 32 bit PAE-enabled x86 or a 64 bit x86 processor
- 512 MB RAM (1 GB Recommended)
- 5 GB Hard Disk space
- Graphics card capable of 800x600 resolution

The requirements for running Linux Mint are comparable with all other Linux distributions researched and compared so far. Although there is no mention of processor speed, it can be assumed that whatever works on Ubuntu should work on this Linux distribution, due to Linux Mint being based off Ubuntu.

Section 3.8.2.6 Xubuntu

Xubuntu was the final GNU/Linux distribution that was considered as the operating system to run on the computer system. Xubuntu itself is a spinoff of the Ubuntu Linux operating system. The difference is that it is designed to run on computer hardware with lower specifications. As the only minimum requirement to run Xubuntu is that it only needs 256 MB of RAM, although it is recommended to have at least 512 MB of RAM [30]. Since Xubuntu is a spinoff of Ubuntu, all of Ubuntu binaries work perfectly in Xubuntu. In addition, since it’s more lightweight than Ubuntu, this allows the computer to dedicate more resources to focus on running the chess engine and voice library and less focus on keeping operating system running. This would allow the both the chess engine and voice recognition software to achieve optimal performance. This makes Xubuntu a highly attractive operating system to run on the computer system.

Chapter 4 : Design Summary of Hardware and Software

Section 4.1: Power System Design

The project needs a power supply large enough to power not only the motherboard but the PCB as well that was controlling the stepper motors, microcontroller, and electromagnet. AC/DC converter circuits were researched but found to be extremely dangerous to build in the fact that if built incorrectly could damage all the electrical components. Much trial and error would need to be done to design a suitable power supply. Wall converters were then considered because they are already built and tested by the manufacturer. Ideally a motherboard power supply would be found that could power both the motherboard and PCB. The motherboard chosen for the project is explained further below.

Section 4.1.1: Motherboard Power Supply

The motherboard power supply is manufactured by FSP Technology Incorporated also known as FSP Group. The model number of the power supply is FSP220-60LE(80). Below is a table showing the important specifications of the power supply, for the entire specifications please the specs sheet attached in the appendix. The table is followed by a schematic of the power supply reprinted with permission from FSP Group.

FSP220-60LE(80) Power Supply Specifications

Spec	Description
Type	Mini ITX / Flex ATX
Max Power	220 W
Main Connector	20(+4) Pin
+12 V Rails	2
SATA Power Connector	2
Input Voltage	115 / 230 V
Output Voltage	+3.3V @ 14A, +5V @ 16A, +12V1 @ 16A, -12V @ 0.8A, +5VSB @ 2.5 A

Table 4-1 FSP220-60LE(80) Power Supply Specifications

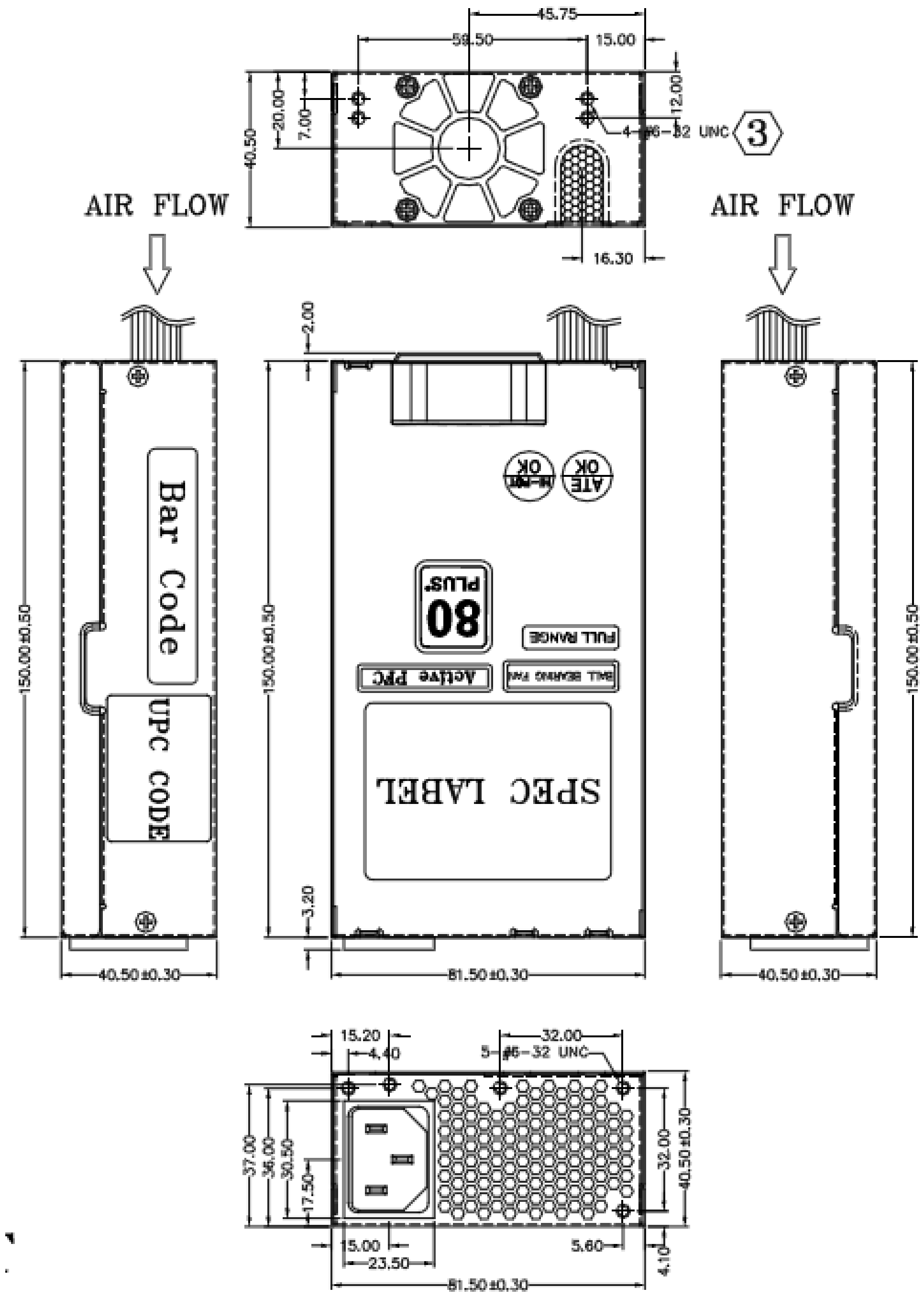


Figure 4-1 FSP60LE(80) Power Supply Layout, permission pending from FSP Group

The FSP220-60LE(80) power supply was chosen because it featured a 1-watt standby mode which is complaint with the Erp/EuP standard. Also the power supply came with 2 SATA connectors to use to power the microcontroller and motor controller. Later, the group realized that the SATA connector would not be enough to power the PCB board and that the Molex connectors PC and PD could be used instead. The 20 (+4) pin main connectors are important to power the mother board. The tables below lists all the pin connections and their features. The main connector pins have a length of about 200 millimeters and the two Serial ATA 15 pin connectors have a length of about 310 and 150 millimeters. The disk driver connectors vary in size; PA (see figure 5) is about 250 millimeters, PB is about 310 millimeters, PC is about 150 millimeters, PD is about 150 millimeters and PE is about 310 millimeters. For the full list of specifications the diagram see the appendix.

Main Connector 20(+4) Pin P1 / 24

Pin No.	Signal [V]	**Pin No.**	Signal[V]
1	+3.3	13	+3.3/+3.3VS
2	+3.3	14	-12
3	COM	15	COM
4	+5	16	PS – ON
5	COM	17	COM
6	+5	18	COM
7	COM	19	COM
8	PW – OK	20	-
9	+5Vsb	21	+5
10	+12V2	22	+5
11	+12V2	23	+5
12	+3.3V	24	COM

Table 4-2 Main Connector Pin Specifications

Serial ATA 15 Pin PF AND PG

PF		**PG**	
Pin No.	Signal [V]	**Pin No.**	Signal[V]
1	+12V2	1	+12V2
2	COM	2	COM
3	+5	3	+5
4	COM	4	COM
5	+3.3	5	+3.3

Table 4-3 SATA Pin Specifications

Disk Drivers

PA		**PB**		**PC**		**PD**		**PE**	
Pin No.	Signal [V]	**Pin No.**	Signal [V]	**Pin No.**	Signal [V]	**Pin No.**	Signal [V]	**Pin No.**	Signal [V]
1	+12V2	**1**	+12V2	**1**	+12V2	**1**	+12V2	**1**	COM
2	COM	**2**	COM	**2**	COM	**2**	COM	**2**	COM
3	COM	**3**	COM	**3**	COM	**3**	COM	**3**	+12V1
4	+5	**4**	+5	**4**	+5	**4**	+5	**4**	+12V1

Table 4-5 Disk Drivers Pin Specifications

The Molex connectors will be used to power the motor controller and microprocessor. The PCB requires a +12V and +5V input (see section 4.3.2 for details). This will be supplied by pins 1 and 2 of the Molex connector. The figure below shows the breakdown of the pin connectors from the power supply and how they are organized.

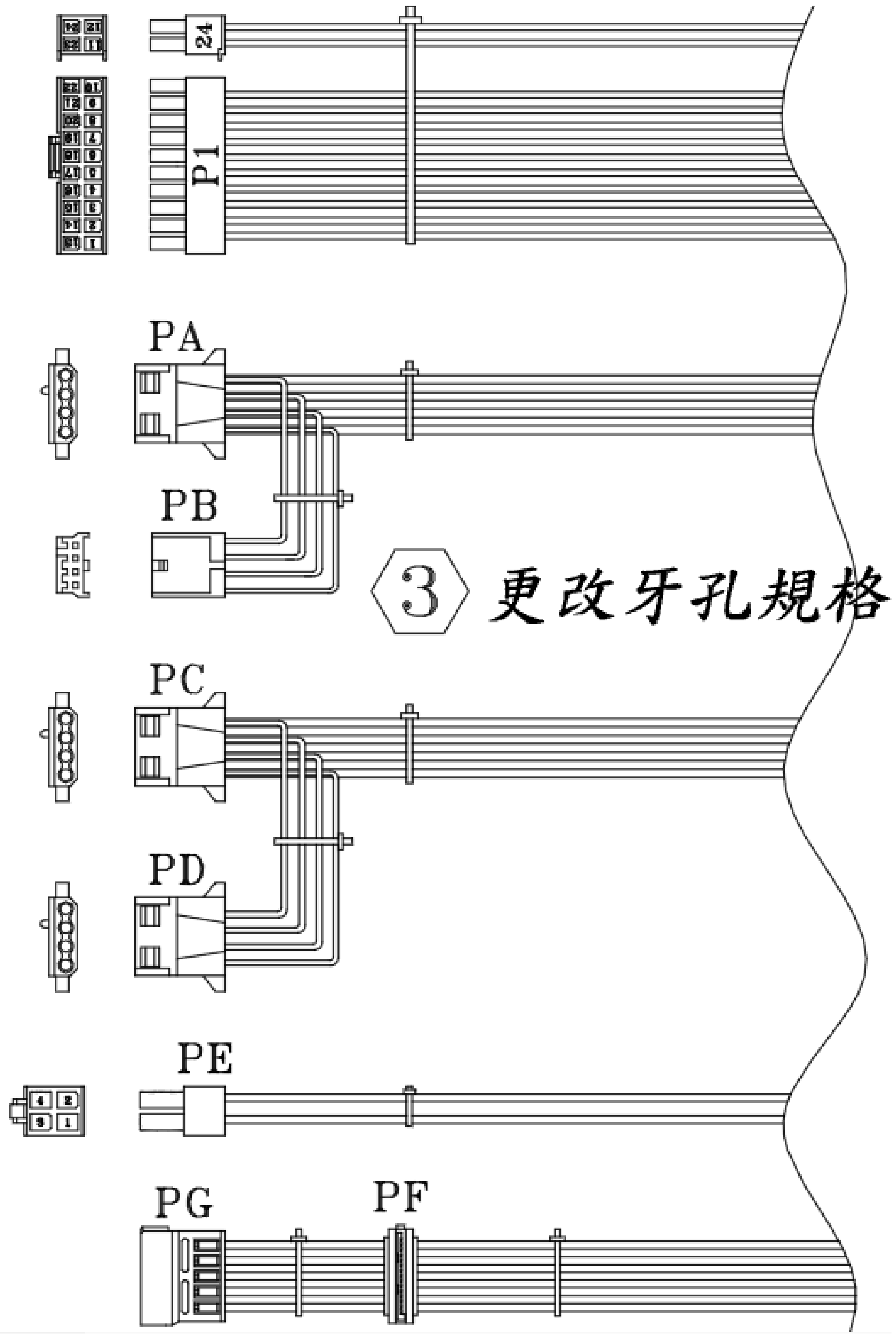


Figure 4-2 FSP220-60LE(80) Pin Configuration, permission pending from FSP Group

Section 4.2: Electromagnet and Permanent Magnet

In order to have the strongest attraction possible between the chess pieces and the electromagnet, only round permanent magnets were considered in the design and therefore only round electromagnets were considered in chapter 5. Also, to keep the movement of the chess pieces as smooth as possible, only DC electromagnets were considered. Also, DC electromagnets are more readily available than AC and consequently cheaper.

To better understand the interaction between the permanent magnets and the electromagnet, the group tested two permanent magnets on the game board, one acting as the electromagnet (See section 5.2.2.1). This test helped narrow down the selection of electromagnets to consider. The table below shows all the electromagnets considered (shown crossed out) as well as the specifications of the electromagnet chosen for the final design (Type G).

Round DC Electromagnets

Type	Size (diameter x height) [inchs]	Volts	Pull [lbs.]	Wt [oz]	Average cost	Overall Advantage
~~A~~	~~.75 x .62~~	~~12~~	~~4.5~~	~~.96~~	~~$34.09~~	
~~B~~	~~.75 x 1.250~~	~~12~~	~~10~~	~~1.7~~	~~$55~~	
~~C~~	~~1 x .719~~	~~12~~	~~10~~	~~1.9~~	~~$53~~	
~~D~~	~~.75 x .62~~	~~6~~	~~4.5~~	~~.96~~	~~$29.24~~	
~~E~~	~~.75 x .375~~	~~12~~	~~5.5~~	~~.8~~	~~$35.36~~	
~~F~~	~~.75 x .62~~	~~6~~	~~6~~	~~.96~~	~~$29.24~~	
G	.75 x .62	12	6	.96	$29.24	✓

Table 4-5 Comparison of Electromagnets

After considering the differences between all electromagnets in chapter 5, the last electromagnet will be purchased for the project. Magnets F and G both came with a pull that would be enough to attract and pull the chess pieces but required 6 and 12 volts of input to do so, respectively. It was decided that magnet G would be the best to use as the power supply supplied by the motherboard (FSP220-60LE) would give 12 volts off of its Molex power connection to the PCB board and it would be easier to power magnet G with the PCB board instead of having to lower the voltage for magnet F to use.

The electromagnet chosen for the project is manufactured by APW Company and has item number EM075-12-222. This electromagnet requires 1 watt of power and can pull 6 pounds with 12 volts. The dimensions of this electromagnet are .75 inches in diameter and .62 inches in height. The housing is finished in Zinc or electroless nickel plating. The magnet comes with 18 inch long leads. The electromagnet weighs .96 ounces and costs $29.24. A photo of the magnet being used is posted below.

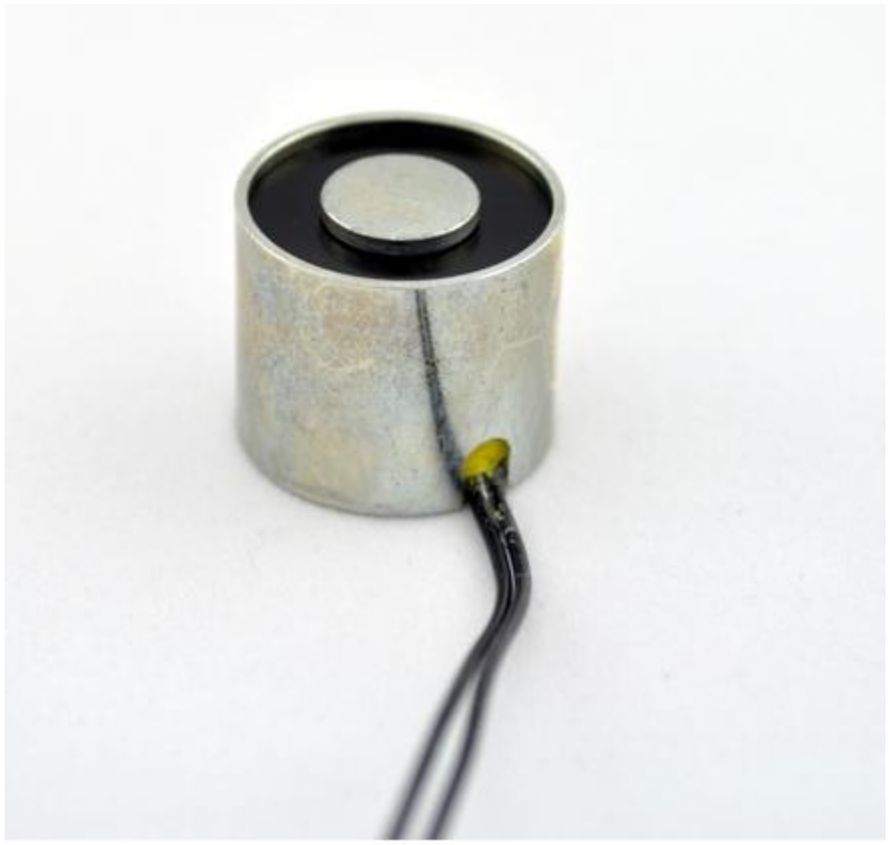


Figure 4-3 Picture of EM075-12-222 DC Electromagnet, Permission for use granted by APW Company [27]

The chosen permanent magnets for the project were the ones used for the electromagnet/permanent magnet test. The permanent magnets are neodymium or rare earth magnets. They were chosen because of the immediate discovery that they were not only stronger, but smaller in size than the ferrite magnets. 32 rare earth magnets were purchased at Skycraft for a total of $13.63.

Section 4.3: XY-Plane Linear Motion Stage

Section 4.3.1: Stepper Motors

The first stepper motor to be used in the design is a fairly heavy-weight stepper from SparkFun Electronics with an internal product number of ROB-10847. This particular motor is manufactured by Wantai Motor with a product number of 57BTGH420 [12]. This motor is a uni-polar hybrid stepper that is four-phase. It includes a 1/4" shaft that would make connecting this motor to the gear and belt system. This motor is also the most powerful in terms of torque of the four motors that were evaluated in chapter 5. This motor will drive up the current supplied by the power supply to the system. This motor was chosen because of its accuracy of 1.8 degrees per step, and the relative price of the motor. Also with the weight of the x-axis crossbar on the stage being around 2 pounds including the second stepper motor, the electromagnet, plus the piece of aluminum and the stage, the group also decided to go with this stepper motor because of its holding torque rating.

The second motor is also from SparkFun Electronics with an internal product number of ROB-09238. It is manufactured by Mercury Motor with an internal product number of 42BYG011 [14]. The step angle, voltage, and holding torque were similar to another stepper motor considered in chapter 5, but the price of this motor is less. It has a 5mm drive shaft, which would require parts measured in the metric system (or extremely close imperial system) measurements. For the x-direction movement of the stage, the group decided to go with this stepper motor since the weight being moved will not be as great as the motor for the x-axis crossbar. This motor is $14.95 and in stock at SparkFun. The table below lists out the two stepper motors that will be used in the design and their technical specifications.

Motor Number	Step Angle (deg/step)	Steps/ Rev	Length (in)	Rated Votlage (V)	Rated Current (A)	Holding Torque (lb-ft)	Weight (lb)
Stepper 1	1.8	200	2.2	3.6	2.0	0.651	1.54
Stepper 2	1.8	Not Incl.	1.3	12	.33	0.169	0.44

Table 4-3 Comparison of two stepper motors

Section 4.3.2: Mechanics of XY-Stage

Two XY-stages were considered for the project. One stage could be bought pre-built but would cost a significant amount of money to purchase. The other stage was much more affordable but would require team assembly. The assembly required stage was chosen to keep the projects budget at a minimum.

A how-to tutorial to build the mechanical portion of the XY-Stage was found on instructibles.com. The stage on this how-to was built in order to allow visitors to a website to draw a design in sand remotely. Acknowledgements go to Mr. Carl S. for designing the low-cost XY-table and sharing the instructions on how to build it with the public.

The whole stage was constructed out of t-slot extruded aluminum in 1”x1”, 1”x2” and 1”x3” dimensions. Sliding glass door acrylic rollers were used to stabilize the stage. Gears were attached to the output shafts of each stepper motor and combined with belts to move the x and y portions of the stage independently.

A brief description of the XY-stage construction will be covered in chapter 6, but as this is a mechanical design, and not an original design, an appendix with the full instruction set will be included at the end of this document. A photo courtesy of the original designer is provided in the figure below.

The basic parts for the mechanical portion of this design are:

- 1”x1”, 1”x2” T-Slot extruded Aluminum
- Miscellaneous hardware for stabilization and assembly
- Pulleys, Timing belts, shaft collars, shafts, and plates for translating the rotational motion of the electric motors to linear motion
- Motors, and their associated hardware.

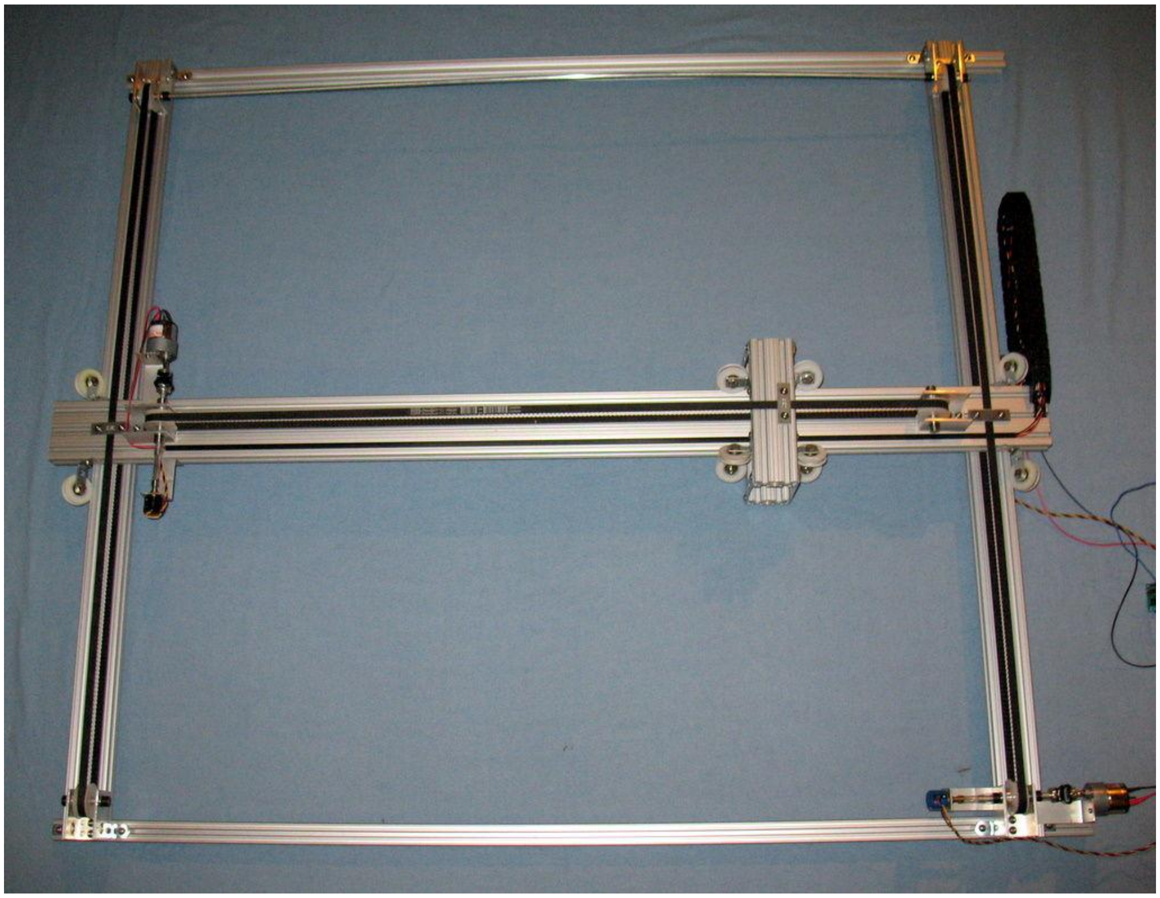


Figure 4-4 Photo of Finished XY-Stage Provided by Carl S. on instructibles.com Permission Pending [11]

Also, in the how-to, servo motors were used. This project will instead be using stepper motors for their ease of use and relative cheap price. The total cost of the XY-Stage and stepper motors is $452.73.

Section 4.3.3: Motor Control Board with Microcontroller

The motor control board will consist of two sets of microcontrollers and motor controllers. This is necessary because the motor control itself can only power one motor at a time, and uses several pins on the microcontroller, making it not possible to run two motor controllers off of the same microcontroller.

Section 4.3.3.1 Microcontroller

Three different microcontrollers were considered: the Stellaris M3 LM3S8962, MSP430G2553 and the Atmega 328. For reasons of price, ease of use, and ability to immediately acquire the part, the group has chosen to use the MSP430 series microcontroller to relay the stepper motor command to the stepper motor control IC. However, also for ease of use, and having already acquired the part, the group will use a Stellaris M3 evaluation board to provide the microcontroller hub and LCD screen. The evaluation board is further discussed in section 4.5.

The Texas Instruments MSP430 line of microcontrollers has a reputation for being low power consumption devices. The MSP430 line has the ability to go into hibernation mode, thereby cutting power consumption. It also has an extremely fast wakeup time when a command is received. The MSP430 is also an affordable microcontroller that supports USB, USCI and I^{2}C communications as well as has the ability to interface with a capacitive touch pad. It is capable of doing 32-bit operations even though it is only a 16 bit microcontroller.

The particular MSP430 microcontroller the group chose was a MSP430G2553. This particular microcontroller comes in a DIP package, which is easy to solder. This particular model also has Touch-Sense enabled IO Pins to use with a capacitive touch wheel. It has 24 General Purpose IO Pins, watchdog and brown out reset circuit protection, and is readily available at DigiKey [17]. The pin-out diagram and functional block diagram of the MSP430G2553 is located in the two figures below.

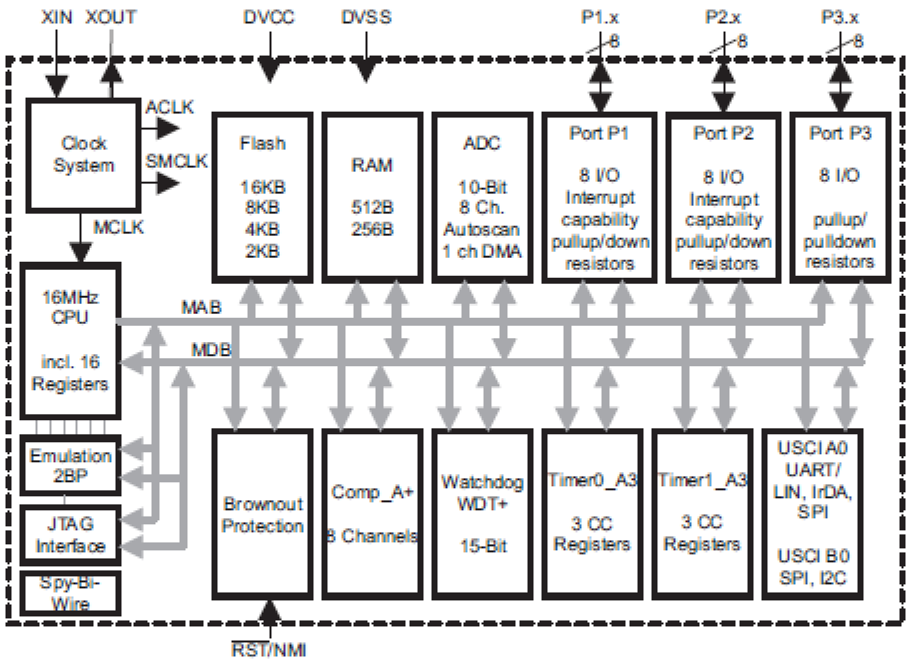


Figure 4-5 Functional Block Diagram for MSP430 - Permission from TI for Use [17]

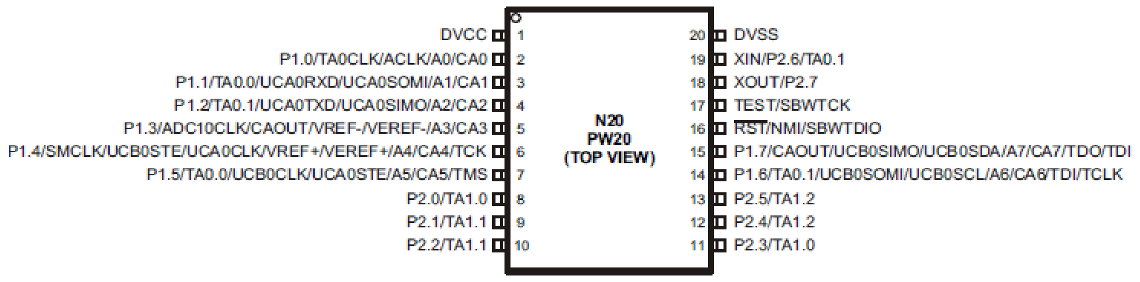

DVCC
P1.0/TA0CLK/ACLK/A0/CA0
P1.1/TA0.0/UCA0RXD/UCA0SOMI/A1/CA1
P1.2/TA0.1/UCA0TXD/UCA0SIMO/A2/CA2
P1.3/ADC10CLK/CAOUT/VREF-/VEREF-/A3/CA3
P1.4/SMCLK/UCB0STE/UCA0CLK/VREF+/VEREF+/A4/CA4/TCK
P1.5/TA0.0/UCB0CLK/UCA0STE/A5/CA5/TMS
P2.0/TA1.0
P2.1/TA1.1
P2.2/TA1.1
N20
PW20
(TOP VIEW)
DVSS
XIN/P2.6/TA0.1
XOUT/P2.7
TEST/SBWTCK
RST/NMI/SBWTDIO
P1.7/CAOUT/UCB0SIMO/UCB0SDA/A7/CA7/TDO/TDI
P1.6/TA0.1/UCB0SOMI/UCB0SCL/A6/CA6/TDI/TCLK
P2.5/TA1.2
P2.4/TA1.2
P2.3/TA1.0

Figure 4-6 Pin-out of MSP430G2553 - Permission from TI for Use [17]

The important aspects of the microcontroller are displayed in the table below.

Value	MSP430
Supply Voltage	3.3V
Bits	16-Bit
Language Supported	C, Assembly
Architecture	RISC
Communications	USB, USCI, I^2C
Output Voltage	3V
Package	20 Pin DIP
Cost	$2.79

Table 4-4 MSP430 Microcontroller Specifications [28]

Section 4.3.3.2 Motor Control IC

The three ICs that were compared were the Allegro Microsystems A3967, the Allegro Microsystems A4983, and the TI DRV8818. The advantage for the motor control chips goes to the Texas Instruments DVR8818 because it will help the group qualify for the Texas Instruments Design Competition and because of the numerous design examples provided by Texas Instruments. The design examples make an easier job of implementing the chip in the circuit and also provide a reference to make designing the motor control board with a microcontroller much simpler.

The motor controller IC is the Texas Instruments DRV 8818. This control IC is capable of supplying 2.5 A per winding and only requires a single pulse to a STEP input to move the motor forward one step. It, like the Allegro Microsystems stepper drivers (compared in chapter 5), also has available full, half, fourth, and eighth stepping modes available by combination of two logic inputs. The TI stepper motor controller also comes in a package that is able to be soldered by hand instead of machine and is capable of supplying the proper voltages and currents to the stepper motors the group has chosen. The table below summarizes the important aspects of the motor control chip.

Value	TI DRV8818
One Pulse Step	✓
Full, Half, Fourth and Eighth Modes	✓
Max Voltage Supply	35V
Max Current Supply	1.9A
Package Type	28-Pin HTSSOP
Step Frequency	500kHz
Documentation	✓✓✓
Cost	$6.88

Table 4-5 Motor Controller IC Specifications [21]

Section 4.3.3.3 Motor Controller PCB

This section will display the schematic and planned PCB for the two motor controllers and the one or possibly two microcontrollers necessary for their operation. The figure below displays the Schematic layout of the board. To ensure the PCB will be functional for the project, a proto board will be built to be tested with the hardware as explained in section 7.2.

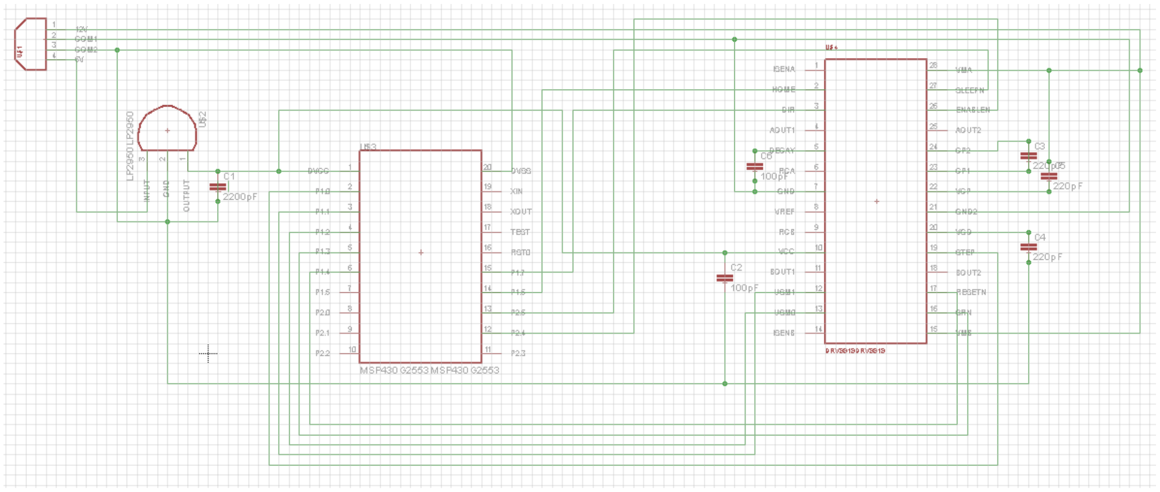


Figure 4-7 Schematic Diagram of Motor Control PCB

Section 4.4: Voice Capture Hardware and Software

The purpose of CMU Sphinx, the voice library, is to simply translate the recorded command into something that the chess engine will recognize and understand.

The voice library will be compiled and installed on the operating system. The correct binaries or source code will be used depending on the operating system that is installed into the computer component. The CMU Sphinx is a voice library that is coded in either Java or C, depending on whether the main version or the PocketSphinx version is used respectively. The choice of which version to use will be done by testing out both versions on the system. The version that has the most responsiveness and lowest consumption of resources will be the version that is deployed.

CMU Sphinx will receive the voice commands through the microphone and interpret them since it comes with an already existing language model. From there, CMU Sphinx will convert the voice command into a command which is understood by the chess engine, which would then perform the move commanded by the player. The response time from having CMU Sphinx interpret the command receive from the player into a command understood by the chess engine should take no longer than a few seconds. The best result should take under a second for the command to be interpreted and sent to the chess engine.

If required, training can be done on CMU Sphinx if it has trouble understanding the short command phrase players often used while playing chess. CMU Sphinx will simply understand and record commands, it will not concern itself which voice command came from which player. That will be for the chess engine to handle.

Section 4.4.1: Microphone

The usage of the microphone in the project is use to capture and record voice commands from the player. The microphone selected for the project will be a standard Logitech microphone. The microphone will need to be able to accurately record the commands spoken by the player. As such, a simple microphone will be more be than enough to fulfill this requirement.

Section 4.5: LCD Screen and Control Software

For the LCD screen, a team member already owned a Stellaris evaluation kit from Texas Instruments. The evaluation kit comes with a LCD screen built in, and has numerous communication methods that would send information not only to the voice recognition system, but with the MSP430s that comprise the motor control network as well. Some of the features of the Stellaris M3 evaluation kit include:

- 10/100 Ethernet Controller
- OLED graphics display 128x96 pixel resolution
- Navigation buttons
- Speaker
- MicroSD card slot
- USB interface for power supply

These and other features of the Stellaris EKS-LM3S8962 can be found in the user manual [16].

The block diagram of the Stellaris EKS is shown in the figure below:

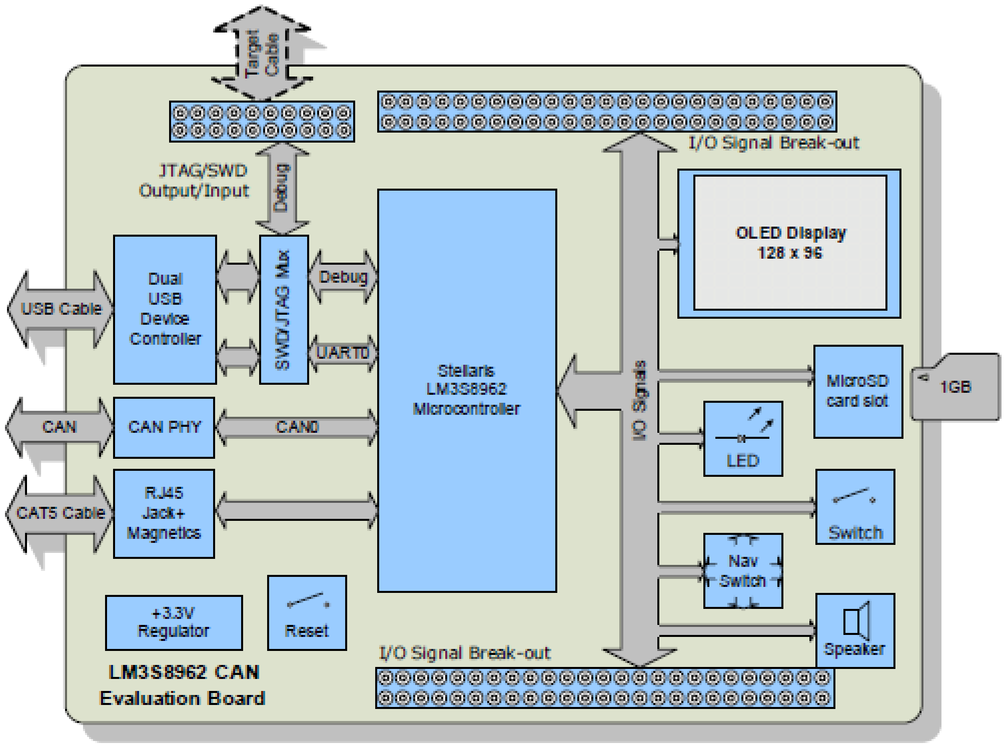


Figure 4-8 Stellaris EKS Block Diagram - Permission from TI for Use [16]

Section 4.6: Software Block Diagrams and Explanations

The diagram below shows the current Software Block Diagram at its most abstract level. It is displayed in the context of a user making a move. Explanation of each module follows the diagram.

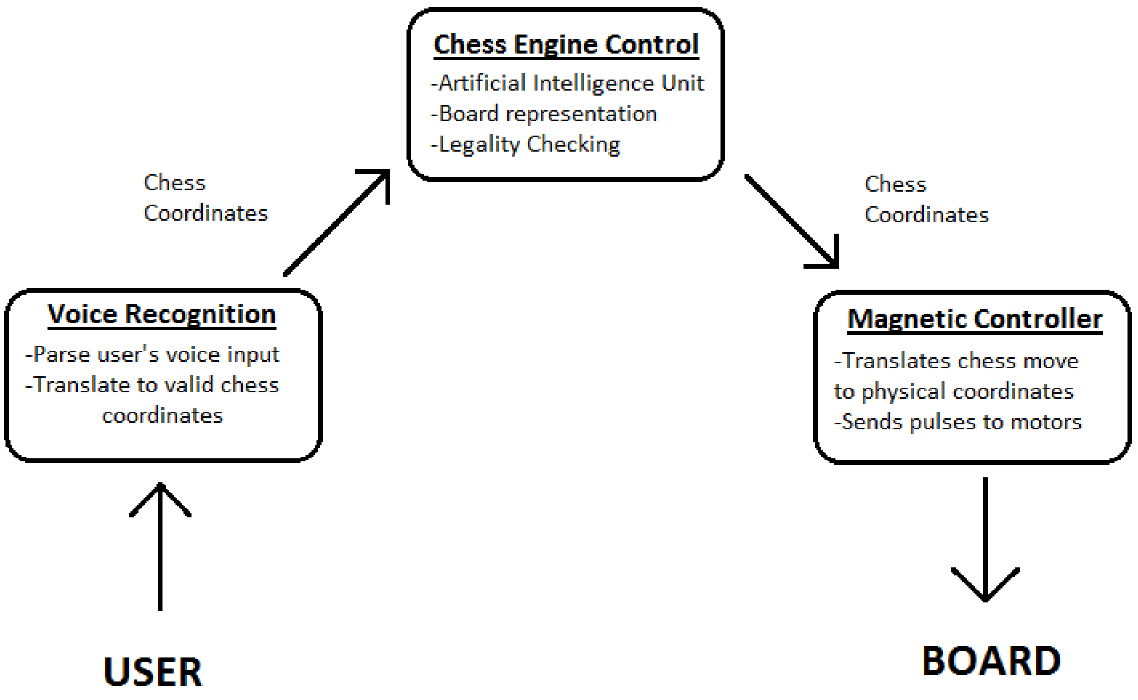


Figure 4-9 Revised Software Block Diagram for Player Move

The Chess Engine Control module remains the centerpiece of the software. In this case, the engine will be TSCP. Recall the benchmarks of TSCP measured with Arena and Windows Task Manager given earlier:

	CPU (approx %)	Memory (MB)	Avg Response Time (sec)
Player's Move	<1	2.1	<1
Computer's Move	5	2.1	<1

Table 4-9 Resources used by TSCP

The diagram shows how the software will be utilized during a normal turn by a user. The user will input a voice command, which will be parsed by the voice recognition software, which will send its (hopefully extremely accurate) interpretation of the command to the Chess Engine Controller in the form of a chess move in algebraic notation. The chess engine will process this and, if the move is legal, send the move to the Magnetic Controller. The Magnetic Controller will translate this move to physical coordinates and calculate the series of pulses that must be sent to the stepper motors to accomplish the move. This diagram holds for player versus player mode. When one player is playing against the computer, the diagram is very similar.

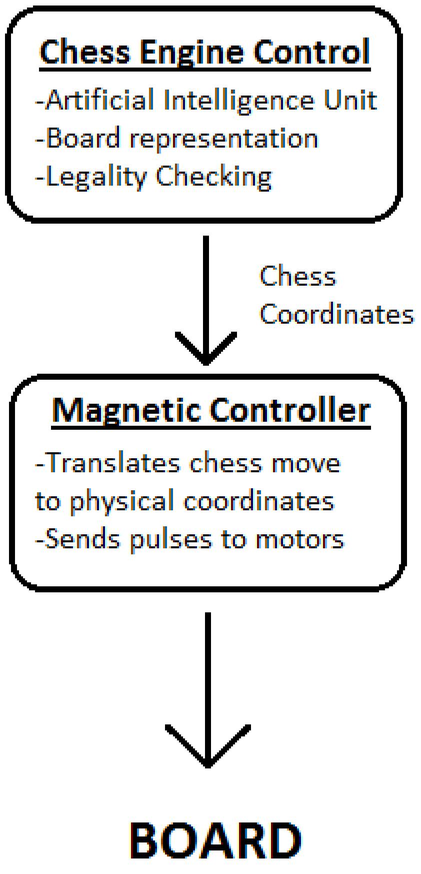


Figure 4-10 Revised Software Block Diagram for Computer Move

In this game state, the only difference is that the chess engine will determine when it is the computer's turn. Once the computer calculates its move, it will also send it to the Magnetic Controller to be physically implemented.

Chapter 5 : Project Hardware and Software Design Details

Section 5.1: Initial Design Architecture and Related Diagrams

This section contains initial block diagrams and block descriptions for each block. The hardware block diagram and block descriptions are located in section 5.1.1 and the software block diagram and block descriptions are located in section 5.1.2. This was included to show progression of thought and iterations of the design process.

Section 5.1.1: Hardware Block Diagram and Block Descriptions

This section contains the hardware block diagram (Figure 5-1) and its block descriptions. Each block has been assigned to a particular team member as noted in section 2.4. Further design details, including final designs and part decisions shall be located in subsequent sections of this chapter, with a summary of the design located in Chapter 4.

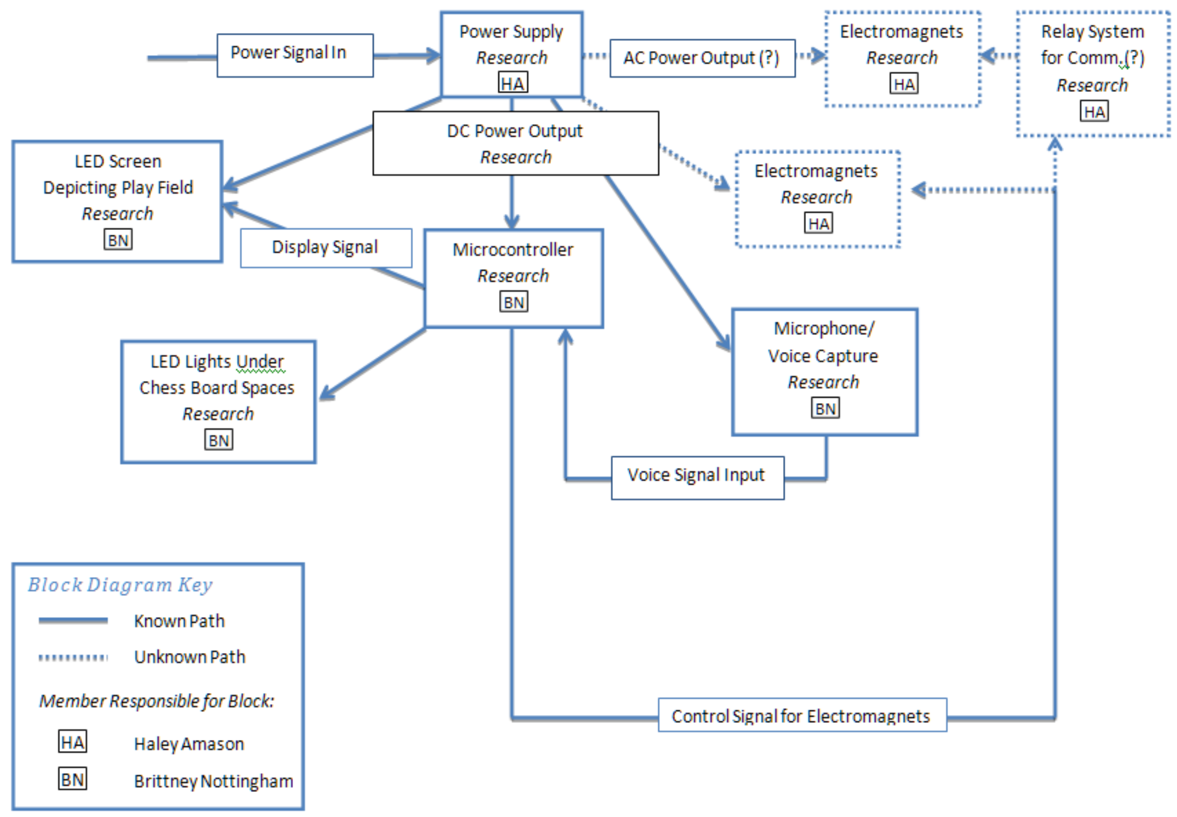

Power Signal In
Power Supply Research HA
AC Power Output (?)
Electromagnets Research HA
Relay System for Comm.(?) Research HA
DC Power Output Research
LED Screen Depicting Play Field Research BN
Electromagnets Research HA
Display Signal
Microcontroller Research BN
Microphone/ Voice Capture Research BN
LED Lights Under Chess Board Spaces Research BN
Voice Signal Input
Control Signal for Electromagnets
Block Diagram Key
Known Path
Unknown Path
Member Responsible for Block:
HA Haley Amason
BN Brittney Nottingham

Figure 5-1 Initial Hardware Block Diagram

Section 5.1.1.1: Explanation of Initial Hardware Block Diagram Blocks

This section includes explanation of the blocks included in the initial hardware block diagram and the reasoning behind keeping or eliminating the blocks from the final diagram.

Section 5.1.1.1.1: Power System Block Explanation

The power system block in this diagram was shown with dotted lines coming out from it as the group was not sure whether or not AC power would be required for the electromagnet portion of the project. Upon further research, it was discovered that the type of motors chosen (stepper motor) would not require AC power, nor would the electromagnet. Therefore, the relay communication system, as well as the AC power output blocks can be eliminated.

The power block will be renamed DC power and will, most likely, be drawn from the power supply that is giving power to the AI computer and distributed among the stepper motors, the motor control board, and any small breakout board used by the project.

Section 5.1.1.1.2: Electromagnet Block Explanation

The electromagnet block was intended to include the design for the electromagnetic component as well as the choices in permanent magnets that were to be included in the pieces. The piece magnets are permanent magnets, so they were always be “on”, but the electromagnet will have to ability to turn on and off when under the spaces they must access and pieces they need to move.

Section 5.1.1.1.3: Control Signal for Electromagnets Block Explanation

Originally this was going to be a control signal to turn the electromagnet on and off at the precise time it was going to be moving a piece. It has now morphed into the XY-Stage component of the project. This portion of the project will be controlling the location of the electromagnet that is to be placed on top of an XY-Plane Stage that moves around the underside of the board. The control signal itself will come from a microcontroller that will receive the distance and direction of the piece to be moved. A horizontal, vertical, and diagonal control scheme will have to be developed to ensure that the pieces will move amongst one another without colliding and still be able to complete their proper move.

Since this block has changed in not only its purpose, but the structure of the overall block diagram, it will be renamed to XY-Stage Control.

Section 5.1.1.1.4: Microcontroller Block Explanation

The microcontroller in this block diagram was meant to be not only the motor controller, but also the holder for the AI component of the chessboard. After researching the topic further, it was discovered that it would be better to contain the AI component and the voice interpretation units in a full-blown micro-ATX form factor computer. This would help the chess engine to run faster, and therefore meet our move requirement of taking less than 3 seconds to figure out the next move.

This decision also benefitted the motor control portion of our project. If kept as it was, the motor control portion would suffer from slow decision-making and therefore, slow the whole project down via a performance bottleneck. This would keep the project from meeting the part of the requirements that states that pieces must also be moved in less than 3 seconds to keep from confusing a player or repeated commands.

This block will therefore be renamed, split, and restructured to include blocks for the AI portion of the project as well as a microcontroller for (possibly) each stepper motor in the XY-Stage configuration.

Section 5.1.1.1.5: LCD Screen Depicting Play Field

The LCD screen depicting the play field was an original idea when brainstorming the project in the very beginning. The group thought it might become necessary to display the decision making done by the AI of the chess engine, current location of pieces, moves that are being considered, and any move that may be in progress. It might also become necessary if any player is mute or cannot speak to move the pieces. A touch screen would allow such players to also participate in a game of chess with friends.

Section 5.1.1.1.6: LED Lights Below Board

The lights below the chess board were originally being considered as part of making the game fun and interactive to use. The original idea was to have LED lights below the play field that would light up and show the current movement of the piece that was currently in play. The lights would turn off after the piece completed its move. Additional consideration then went into the design and the placement of the LED lights, and it was determined that it may be better to just have the LCD screen depicting piece movements. This was, in part, due to the fact that the electromagnet would have to closely skim the bottom of the board to make sure that enough of the EMF that it was producing would actually interact with the permanent magnet below the piece to make it move. Having LED lights in the way would limit the movement of the electromagnet and therefore make the game essentially unplayable unless a z-axis control was also added. In addition, another microcontroller would have to be added to control the lights themselves and make sure that the proper path was lit up as the piece moved.

This part of the diagram was eliminated due to the complications of actually adding the LED lights and the estimate of time that would have to be spent making it function properly.

Section 5.1.2: Software Block Diagram Explanation

This section contains the software block diagram (Figure 5-2) and its block descriptions. Each block has been assigned to a particular team member as noted in section 2.4. Further design details, including final designs and algorithm decisions shall be located in subsequent sections of this chapter, with a summary of the design located in chapter 4.

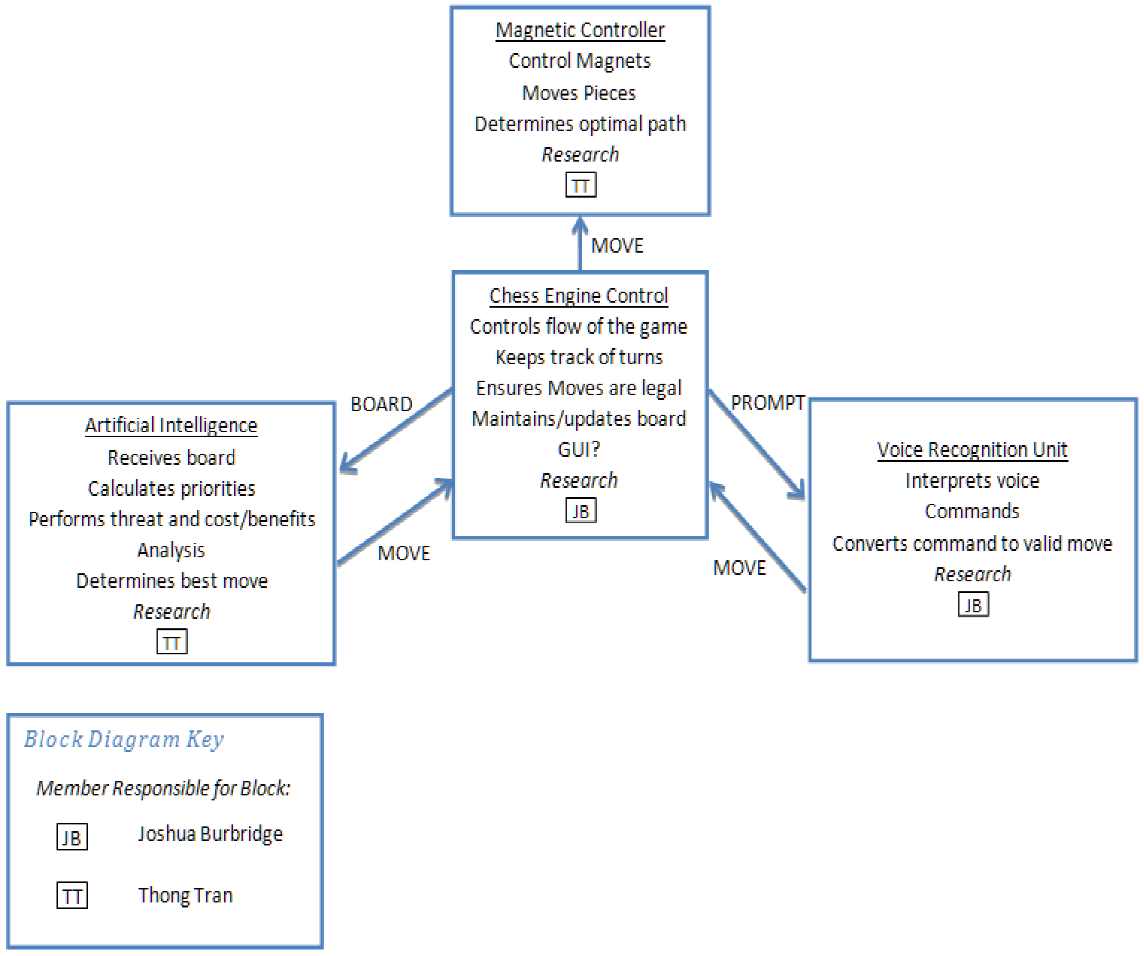


Figure 5-2 Initial Software Block Diagram

Section 5.1.2.1: Explanation of Initial Software Block Diagram Blocks

Over the period of time from the project's conception to its current design, the software diagram has undergone some important changes. Initially, the plan for the software was to only use open-source software for the voice recognition module, and to create an original chess engine from the ground up. The software diagram below is based on that assumption.

Section 5.1.2.1.1: Chess Engine Control Block

The centerpiece of this block diagram is the Chess Engine Control module. Because the ability to play a game of chess is the most important aspect of this project, Chess Engine Control was given top priority. After it was developed, everything else would be built around it. The primary function of this module is to control the flow of the game. This means it must be able to start a new game, alternate turns between players, and declare an end to the game by recognizing check, checkmate, or stalemate conditions. To do this, it must keep and maintain a representation of the chess board, updating it after every move.

To save on space, the possibility of creating a simple 8x8 matrix of bytes was investigated, where each numerical value in a byte (0-255) represented a condition. For example, a byte value of 0x00 might mean the square is empty. A value of 0xFA might signify that Black's rook currently occupies that space. Because this module was supposed to keep the game running smoothly, it was also delegated the task of checking move legality. Naturally, it would not be desirable if a player could begin making illegal moves, so it would be necessary to devise some way of storing a rulebook that the module could consult after every turn. In this way, it would act as the "referee" of each chess match. The final component of this module was optional. Some chess engines can be run via a command line interface, but most need a GUI such as WinBoard or Arena. The task of testing chess engine control would be greatly simplified by having a visual representation of the board available, whether this was through a GUI or some ASCII representation of the board in command line mode. After some research, it was discovered that WinBoard is a very popular open source GUI designed for the purpose of testing custom chess engines. However, it was initially unknown whether or not the module would need a custom GUI as well.

Section 5.1.2.1.2: Artificial Intelligence Block

In the original software block diagram, it was decided that the part of the chess engine that actually plays against a human would be a separate module from the chess engine controller itself. This was because it seemed more logical to design a foundation for the software and build up from that. The artificial intelligence unit would simply be viewed by the chess engine controller as another player, and would utilize the exact same interface as a human. A very abstract view of this interface simply shows the AI receiving the board from the chess engine controller, analyzing it, and responding with a move. In order to determine a best move, the AI would also have to have a representation of the rule book (so that it did not waste time searching illegal moves). After receiving the board it would perform a threat analysis, which would tell it which of its pieces were currently in jeopardy. It would also implement the minimax algorithm with alpha-beta pruning to perform a lookahead of a few moves, and perform cost/benefit analysis on that decision tree to ultimately decide which move to make. The cost/benefit analysis might be done by assigning a value to each piece to help the computer determine when to capture and when not to. If the end result of a capture was negative (i.e. capturing a defended pawn with a queen), the computer would know that this move was not worth making.

Because one of the features Magic Chess requires is a varying difficulty level, the initial software plan was to implement this by performing a lookahead of a greater number of moves with increasing difficulty level. For example, in easy mode, it might look 2 moves ahead, whereas in medium and hard modes it might look 3 and 4 moves ahead respectively.

Section 5.1.2.1.3: Voice Recognition Block

When first creating a broad design for this project, not much was known about voice recognition software. Instead of creating custom software, it would be necessary to use open-source software to interpret voice commands because the problem of analyzing signal frequency and statistically mapping it to possible English words was far beyond the scope of this project. It was known that Google had an open source API that used the Android operating system. This software is called RecognizerIntent. Initially, Magic Chess was going to use this software, but for reasons discussed in further detail in other sections, it was decided that Magic Chess should use CMU’s PocketSphinx, largely due to the fact that it was designed for use in embedded systems.

Because voice recognition libraries can often be quite sizeable, the possibilities of how to implement one on a smaller system were discussed. It was primarily noted that it would not be necessary to index a full dictionary in this case because the chess board would only be required to interpret a handful of very specific commands. If algebraic notation were used for moving pieces (where “e3e5” means to move the piece at e3 to e5), the board would only have to understand standard chess coordinates, which include the letters A through H and the numbers 1 through 8. In essence, it might be possible to have a dictionary with only 16 entries, so any input to the voice recognition software would be pidgeonholed into 16 possible “words”. To keep the board from constantly attempting to interpret input while the players are speaking normally, the possibility of including a button to signify an incoming command was discussed. This way, the board would not be constantly listening for voice commands and using additional resources to parse them.

Section 5.1.2.1.4: Magnetic Controller Block

The final module in the initial software block diagram is the magnetic controller. This is the software that will control the stepper motors that move the magnet that controls the pieces. Utilizing open-source software for this purpose was not a valid option, as it was unlikely that there existed code that already performs this function. The interface between the chess engine controller and this module is simple. The chess engine controller simply sends the coordinates of the most recent move to the magnetic controller, which translates this into the coordinates on the physical grid. The module then interprets these new coordinates to calculate the number of pulses that must be sent to the stepper motors to perform the current move. Once the magnet reaches the first coordinate, it engages and grabs the piece. It then moves to the second coordinate and disengages. To conserve power, the magnet will not move back to a “home” position by default, but will remain in place because it is reasonably likely that much of the attention in the game is being focused on the current area.

Originally, an additional function of the magnetic controller was going to be to calculate the most optimal path to perform a move. This was because some pieces, like the knights, can jump over other pieces in normal chess, but in Magic Chess, everything must operate on the same plane. It became obvious that it would be necessary to move other pieces out of the way before moving the knights. The problem was then how to minimize the number of moves necessary to clear a path for the knights. Multiple path-finding algorithms such as Dijkstra’s Algorithm were considered. Eventually, though, it became clear that the problem could be simplified by making the squares on the board large enough and the pieces small enough that they could be dragged along the lines in the grid to their destination, and therefore other pieces would not have to be moved at all.

Another important thing to consider is that in the case where the chess engine uses an opening book, it may calculate its move faster than the Magnetic Controller can complete its current process. Therefore, the Magnetic Controller will have to have a buffer and the ability to maintain a short list of tasks so that the game can continue without interruption or the system getting backed up. In a one-player game, the human cannot move immediately after the computer because the physical move on the board is the only way that the computer’s move is communicated to a human. Therefore, in this situation, the “to-do” queue would be a maximum of two moves long. This is not the case when two humans play together. It may be possible to overload the system if both players rapidly input voice commands over and over, but it is assumed the players will not behave like this. Therefore, it is safe to assume that the Magnetic Controller module would only need to store up to perhaps 4 moves at a time. This way, it can constantly listen for input as well as operate the magnet simultaneously.

Section 5.1.3: Revised Block Diagrams

In Sections 5.1.3.1 and 5.1.3.2 are revised block diagrams of the hardware and software portions of the project. These block diagrams were deemed necessary to redraw when research determined that the initial structure of the block diagrams were not conducive to a well working product or just simply not possible.

Section 5.1.3.1: Hardware Block Diagram

The following diagram is a revised block diagram for the hardware portion of the project. It was deemed necessary to revise the block diagram after it was discovered that some of the block names and functions would be changing and the structure of the diagram was greatly changed. As this is the final block diagram of the project, the block descriptions will be located in Section 5.2 as part of the detailed design descriptions of the project.

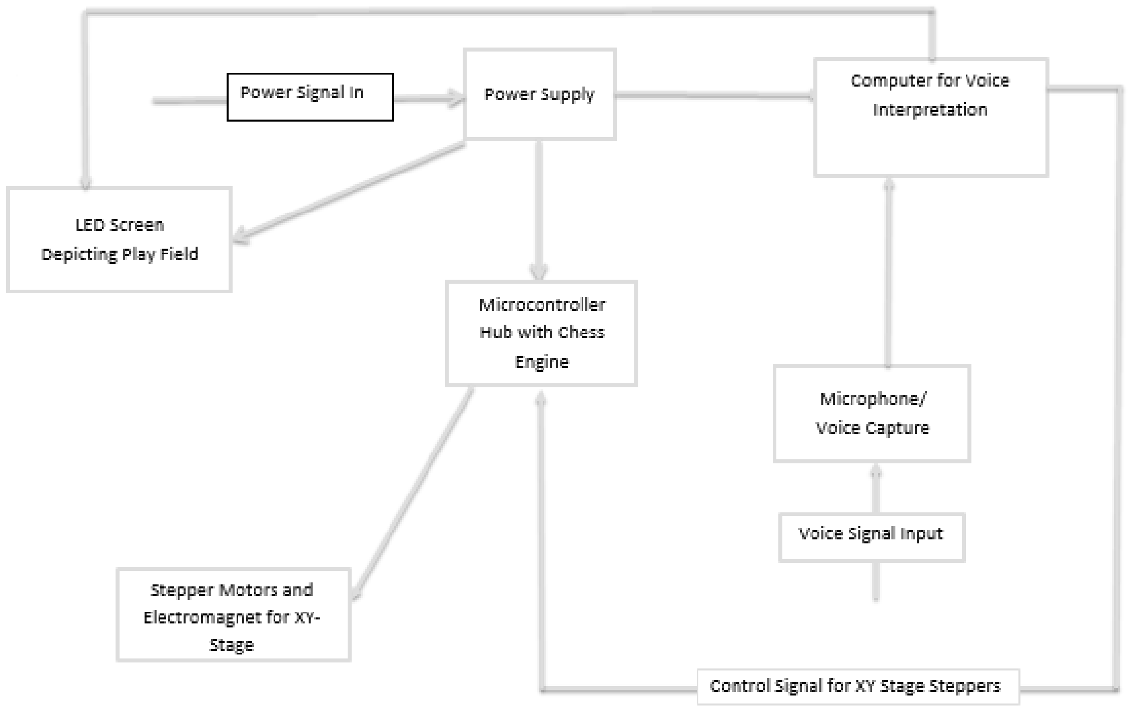

Power Signal In
Power Supply
Computer for Voice Interpretation
LED Screen Depicting Play Field
Microcontroller Hub with Chess Engine
Microphone/ Voice Capture
Voice Signal Input
Stepper Motors and Electromagnet for XY-Stage
Control Signal for XY Stage Steppers

Figure 5-3 Revised Hardware Block Diagram

Section 5.1.3.2: Software Block Diagram

After some research, it became clear that the problem of writing a chess program is extremely complex. Writing the chess engine alone could easily have been expanded and completed over the course of a few years due to the sheer number of things one must consider when writing one. Clearly, two programmers working for only a few months could not create an engine with all the characteristics necessary for a project such as this. Instead of writing a custom chess engine, it was decided that the project will use an open source chess engine which combines the functions of the chess engine controller and the artificial intelligence unit. The updated software diagram is shown below in Figure 5-4. When it is the computer’s turn, the game flow changes slightly. This is reflected in Figure 5-5.

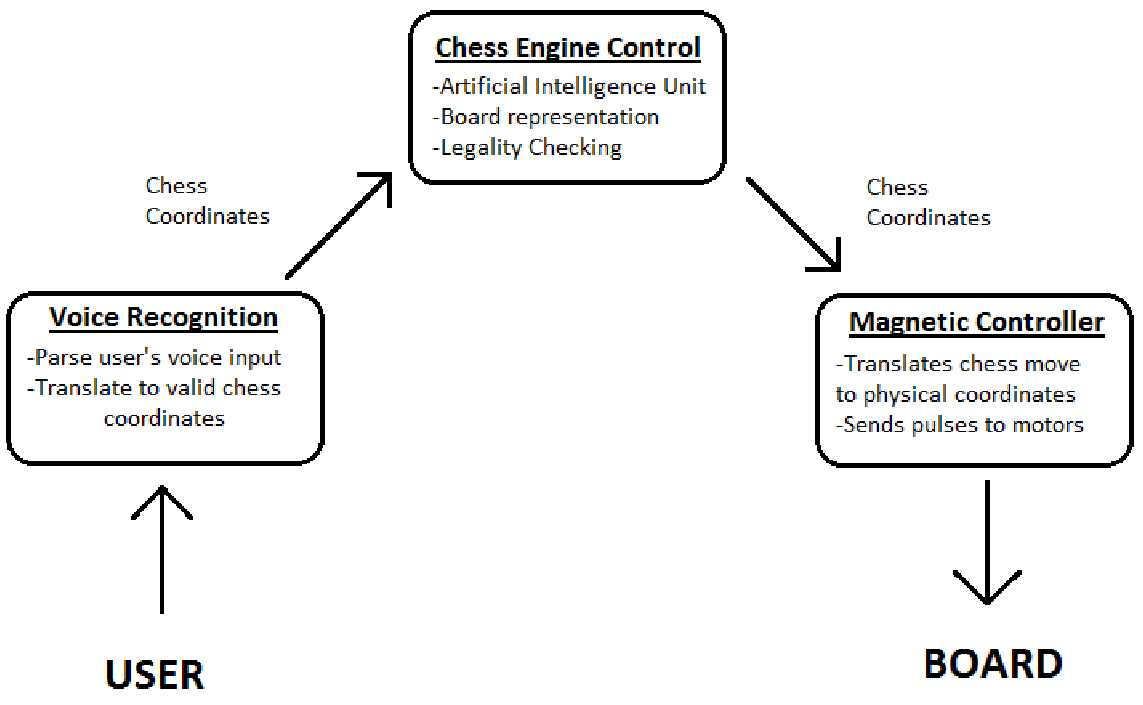


Figure 5-4 Revised Software Block Diagram for Player Move

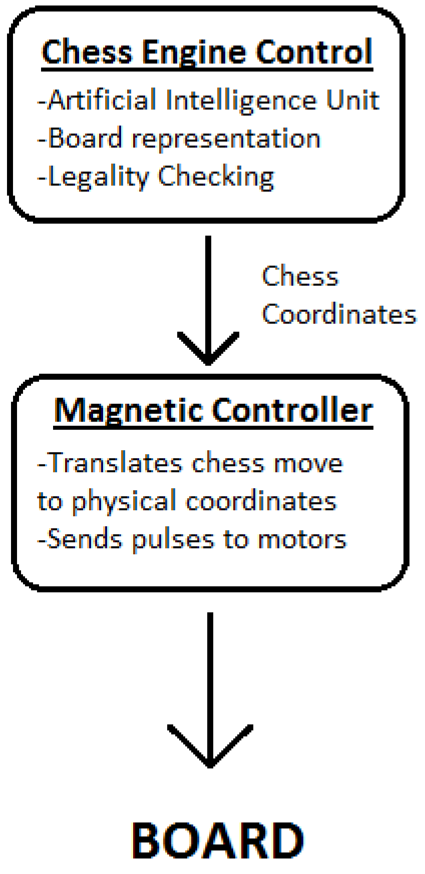


Figure 5-5 Revised Software Block Diagram for Computer Move

Section 5.2: Hardware Designs and Details

This section contains all the design details based on the final block diagram for the hardware portion of the project. The sections are broken down by block in the block diagram and some are broken down further to examine parts of each block.

Section 5.2.1: Power System Details and Related Diagram

The power system was researched with two scenarios in mind. One was that the power supply chosen for the motherboard would have enough SATA pin connectors and wattage to power not only the motherboard but also the microcontroller and motor controller. The second possibility considered was if the power supply chosen would not be able to power anything other than the motherboard. In the second case an AC-to-DC converter circuit(s) would need to be designed and constructed or an AC adapter with multiple DC voltage outputs would need to be purchased.

Research on the design of an AC-to-DC circuit brought up many concerns, one of which was the fact that the simulation software available to help with the design of the circuit would not be able to account for the environment inside the chess board as well as having commercially available components to test with. Once the circuit could be properly simulated and the parts could be shopped for, what was commercially available would most likely differ from the components used in the design. Also, more problems could arise while testing the circuit with the microcontroller and motor controller that could cause damage to the power supply circuit and to the micro- and motor controller. This risk would delay the project’s completion and could drain the project budget.

It was apparent to the group that all efforts be focused on finding an AC wall adapter to use to power the electrical components if a suitable power supply for the motherboard could not be found. Also the hardware of the project needs to be constructed and tested before the software can be installed. In order to keep the project on track and to reduce any risk of damage to other hardware in the chess board the focus of the power supply would be finding an adequate motherboard power supply to provide power to all electrical hungry components in the chess board.

Section 5.2.3 reveals that a suitable power supply was found. The rest of this section will detail on connecting the motherboard power supply to the PCB.

Section 5.2.1.1 Motherboard Power Supply

The motherboard power supply is manufactured by FSP Technology Incorporated also known as FSP Group. The model number of the power supply is FSP220-60LE(80). Below is a table showing the important specifications of the power supply. For the complete list of specifications please refer to the specs sheet attached in the appendices. The table is followed by a schematic of the power supply reprinted pending permission by FSP Group.

FSP220-60LE(80) Power Supply Specifications

Spec	Description
Type	Mini ITX / Flex ATX
Max Power	220 W
Main Connector	20(+4) Pin
+12 V Rails	2
SATA Power Connector	2
Input Voltage	115 / 230 V
Output Voltage	+3.3V @ 14A, +5V @ 16A, +12V1 @ 16A, -12V @ 0.8A, +5VSB @ 2.5 A

Table 5-1 FSP220-60LE(80) Power Supply Specifications

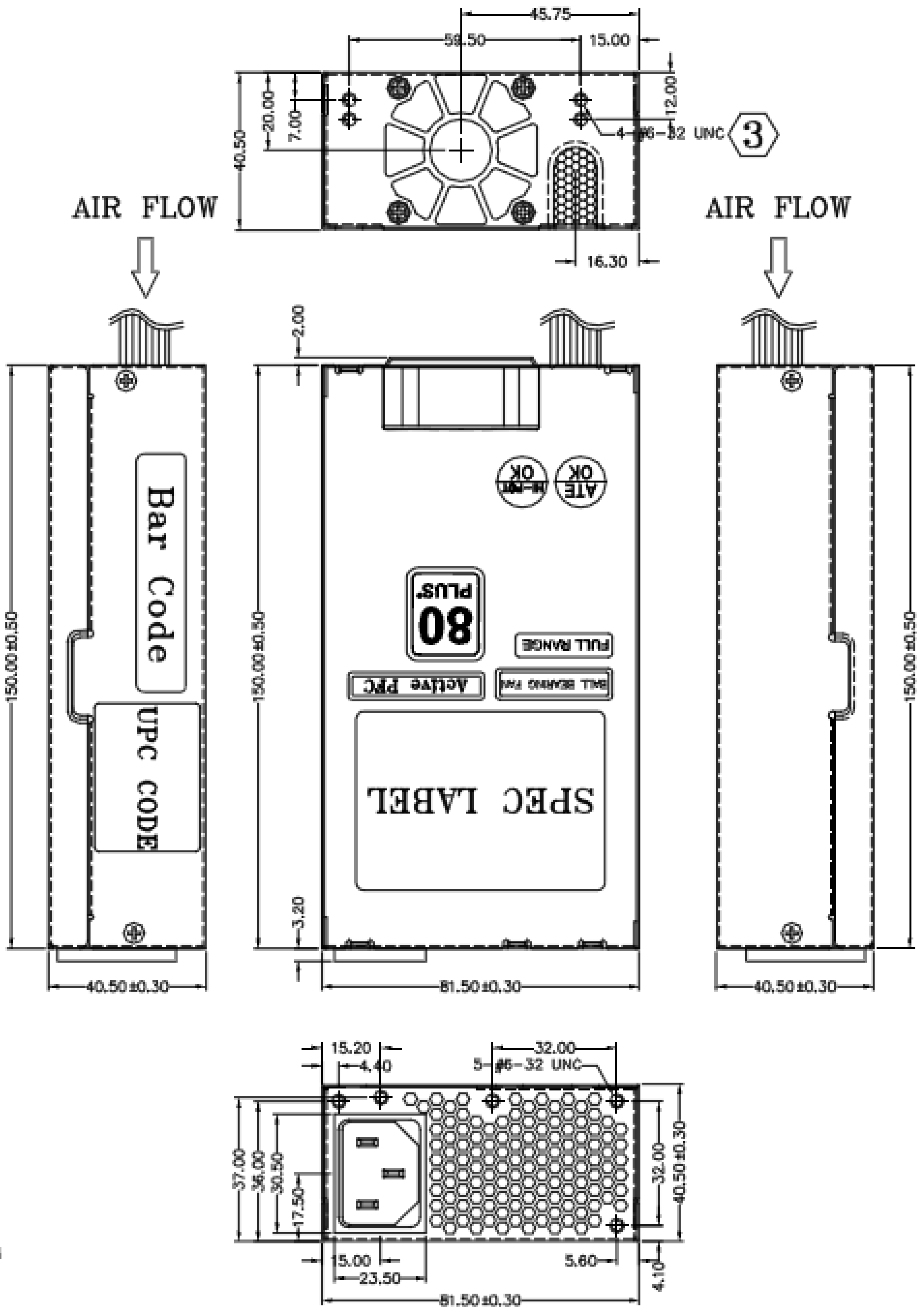


Figure 5-6 FSP60LE(80) Power Supply Layout, permission for use granted by FSP Group. See appendix for email records.

The FSP220-60LE(80) power supply was chosen because it featured a 1-watt standby mode which is complaint with the Erp/EuP standard. Also the power supply came with 2 SATA connectors to use to power any addition electrical components discovered to be necessary during testing. The Molex connectors (PC and PD) would be used to power the PCB if it were discovered that the PCB would not receive enough amperage from the SATA power connectors to function. The 20 (+4) pin main connectors are important to power the mother board. The tables below lists all the pin connections and their features. The main connector pins have a length of about 200 millimeters and the two Serial ATA 15 pin connectors have a length of about 310 and 150 millimeters. The disk driver connectors vary in size; PA (see figure 5) is about 250 millimeters, PB is about 310 millimeters, PC is about 150 millimeters, PD is about 150 millimeters and PE is about 310 millimeters. For the full list of specifications the diagram see the appendix.

Main Connector 20(+4) Pin P1 / 24

Pin No.	Signal [V]	**Pin No.**	Signal[V]
1	+3.3	13	+3.3/+3.3VS
2	+3.3	14	-12
3	COM	15	COM
4	+5	16	PS – ON
5	COM	17	COM
6	+5	18	COM
7	COM	19	COM
8	PW – OK	20	-
9	+5Vsb	21	+5
10	+12V2	22	+5
11	+12V2	23	+5
12	+3.3V	24	COM

Table 5-2 Main Connector Pin Specifications

Serial ATA 15 Pin PF AND PG

PF		**PG**	
Pin No.	Signal [V]	**Pin No.**	Signal[V]
1	+12V2	1	+12V2
2	COM	2	COM
3	+5	3	+5
4	COM	4	COM
5	+3.3	5	+3.3

Table 5-3 SATA Pin Specifications

Disk Drivers

PA		**PB**		**PC**		**PD**		**PE**	
Pin No.	Signal [V]	**Pin No.**	Signal [V]	**Pin No.**	Signal [V]	**Pin No.**	Signal [V]	**Pin No.**	Signal [V]
1	+12V2	**1**	+12V2	**1**	+12V2	**1**	+12V2	**1**	COM
2	COM	**2**	COM	**2**	COM	**2**	COM	**2**	COM
3	COM	**3**	COM	**3**	COM	**3**	COM	**3**	+12V1
4	+5	**4**	+5	**4**	+5	**4**	+5	**4**	+12V1

Table 5-4 Disk Drivers Pin Specifications

The SATA connectors will be used to power the motor controller and microprocessor. The microprocessor (see section 5.2.4) will require 3.3 V to operate; therefore Pin numbers 3 and 5 of the serial ATA pin connector will be used. The motor controller will require 12 V to properly control and move the stepper motors. Therefore, pin numbers 1 and 2 will be used to power the motor controller. The figure below shows the breakdown of the pin connectors from the power supply and how they are organized.

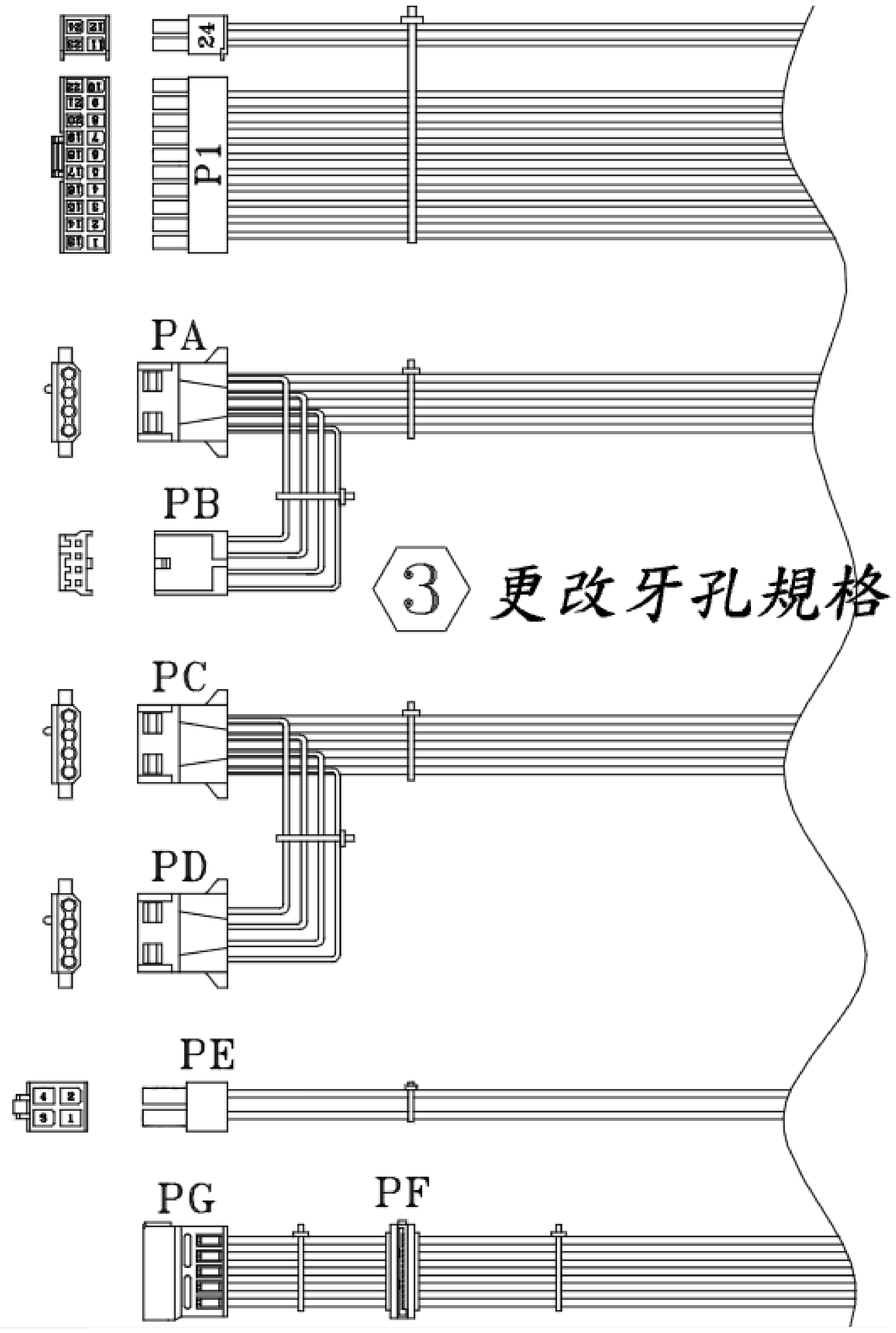


Figure 5-7 FSP220-60LE(80) Pin Configuration, permission for use granted by FSP Group [26]

Section 5.2.2: Electromagnet and Permanent Magnet Design and Diagrams

This section details the research related to the electromagnet and permanent magnet design as determined by the board and piece material choices made in section 5.2.8.

Section 5.2.2.1: Research Related to Magnet Design

The design of the magnets of the board (both permanent and electromagnetic) will depend not only on the material of the board, but the material of the pieces and their weights. Two materials for the pieces and board were tested and the results are listed in tables in section 5.2.8.

Two materials for the magnets were also tested: a set of ferrite and a set of neodymium. The neodymium ones were chosen because of the immediate discovery that they were not only stronger, but smaller in size than the ferrite magnets. Hard specifications were only recorded for the neodymium magnets because of their immediately apparent advantage.

After choosing the material of the board, magnets, and pieces, two different configurations were examined: an electromagnet interacting with steel washers inside the pieces or an electromagnet interacting with permanent magnets embedded inside the pieces. After basic experimentation, it was determined that an electromagnet interacting with a permanent magnet was a better choice for two reasons:

1. The axis of the electromagnet and the permanent magnet would line up in the center of the pieces instead of being on the perimeter in the pieces that were fitted with washers.
2. If the pieces were to come too close to one another, the pieces with the washers would stick together and form a chess-piece train as the washers would stick out from the base because of their size. The permanent magnets that were tested for the base of the pieces maxed out at 0.5" diameter, so they were able to be contained within the pieces.

Several characteristics of each magnet tested were evaluated and experimented with to determine the proper configuration of magnets to use for the project. The table of characteristics that were evaluated is below.

The dimensions of the magnets were measured with an analog micrometer that is accurate to a ten-thousandth of an inch. Measurements were rounded for ease in comparing them to one another.

Magnet	Dimension (in)	Surface Gauss	Pull Capacity (lb)
Medium Disk	0.375 dia x .0625 d	1895	7.372
Small Disk	0.25 dia x 0.0625 d	2163	3.566
Square	1 l x 1 w x 0.25 d	3182	78.8
Short Cylinder	0.375 dia x 0.25 d	4127	8.80
Tall Cylinder	0.375 dia x 0.35	4218	8.078

Table 5-5 Magnet Characteristics [10]

The magnets listed above were chosen for their variety in size and pull capacity.

Before buying an electromagnet, the group tested the interaction between two permanent magnets to more understand the requirements needed for the electromagnet. An overall observation was recorded for each pairing, then a consensus was reached after reviewing each observation and comparing the benefits to each set tested. The following table documents which magnets were paired and the observation of each pairing. The trial electromagnet is the magnet that was put underneath the board to simulate the eventual electromagnet and its properties.

Trial Electromagnet	Piece Permanent Magnet	Observation
Medium Disk	Tall Cylinder	Magnet attraction between piece and trial electromagnet was good. There was no slipping even when the trial magnet was moved quickly. Problems arose when more than one piece was in any given area. Multiple pieces in one area led to attraction of pieces together and the formation of a chess-piece train.
Medium Disk	Short Cylinder	Medium disk and short cylinder observations were almost exactly the same as the tall cylinder observations.
Medium Disk	Medium Disk	Attraction between the medium disks through the glass of the board was good, but movement of the trial electromagnet could not be abrupt due to tipping and slipping of the pieces.
Small Disk	Tall Cylinder	Attraction between the small disk and the tall cylinder was good, but not as good as the small disk and short cylinder. Pieces slipped quite often and could not be moved very quickly. If several pieces were in the area, then they would become attracted to each other and disrupt the proper movement of the pieces.
Small Disk	Short Cylinder	The small disk and short cylinder had good attraction, but slipped and could not be moved quickly across the board. In addition, the piece density in any given area had to be low to avoid the “chess-piece” train.
Medium Disk	Small Disk	The small disk and medium disk had good attraction and could be moved at a respectable speed. The pieces could be within 0.0625” of each other and not interact.
Small Disk	Small Disk	The small disk and small disk interaction was almost negligible. These pieces would have to be moved very slowly and would probably not meet the specifications of moving pieces within 3 seconds or less.

Table 5-6 Trial Magnet and Piece Magnet Observations

After observing the difference between the different configurations of pieces and trial magnets, it was decided that this project would use the small disk and medium disk configuration. This meant that whatever electromagnet we choose should have similar characteristics as the medium disk magnet.

Section 5.2.2.2: Electromagnet Design

As previously discussed in chapter 3, there are multiple types of electromagnets available on the market. DC or AC electromagnets are available in a round or square form. The strength on the magnet increases with the voltage. To increase the attraction between the chess pieces and the electromagnet only round permanent magnets were considered in the design and therefore only round electromagnets will be considered in the section below. Also, the chess pieces should have smooth movement across the board (as if they were being moved by an invisible hand). AC electromagnets could cause the chess pieces to move with a slight unevenness. To keep the movement of the chess pieces as smooth as possible, only DC electromagnets were considered. Also, DC electromagnets are more readily available than AC and consequently cheaper.

To better understand the interaction between the permanent magnets and the electromagnet the group tested two permanent magnets on the game board, one acting as the electromagnet. This test helped narrow down the selection of electromagnets to consider. The section below describes the seven types of round DC electromagnets considered for the project based on the results discuss in section 5.2.2.1.

The first electromagnet considers (Type A) is manufactured by APW Company and has item number EM07-12-222. This electromagnet is a 1 watt round electromagnets that requires 12 volts to have a pull of 4.5 pounds. This electromagnet weighs .96 ounces and is .75 inches in diameter and is .62 inches in height. The housing is finished in Zinc or electroless nickel plating. The magnet comes with 18 inch long leads. The cost of this electromagnet is $34.09. The above study in section 5.2.2.1 calls for a pull of about 7 pounds to move each chess piece individually. 4.5 pounds will most likely not be enough to move the chess piece at all or with a steady motion.

The second electromagnet (Type B) is manufactured by Bunting Magnetics Company and has item number BDE-0812-12. This electromagnet is a 1.5 watt round electromagnet that requires 12 volts to pull 10 pounds. This electromagnet weighs 1.7 ounces and is .75 inches in diameter and 1.25 inches in height. The cost of this electromagnet is $55 dollars. The 10 pound pull would meet the 7 pound pull from the permanent magnet study but this electromagnet is second most heavy out of all the electromagnet considered and costs the most.

The third electromagnet (Type C) is manufactured by Bunting Magnetics and has item number BDE-1007-12. This electromagnet is a 1 watt round electromagnet that requires 12 volts to pull 10 pounds. This electromagnet weighs 1.9 ounces is 1 inch in diameter and .719 inches in height. The cost of this electromagnet is $53 dollars. This electromagnet is slightly smaller than the second one and weighs .2 ounces more. This electromagnet is the second most expensive out of the selection being considered but will meet the pull requirement set by the permanent magnet study.

The fourth electromagnet (Type D) is manufactured by APW Company and has item number EM075-6-122. This electromagnet requires 1 watt of power or 6 volts to pull 4.5 pounds and weighs .96 ounces. The cost of this electromagnet is $29.24. This electromagnet requires fewer volts than the first electromagnet and has the same pull as the first electromagnet. The magnet's housing is finished in Zinc or electroless nickel plating. The magnet comes with 18 inch long leads. This electromagnet weighs the same as the first and is the same size as the first electromagnet (.75 inches in diameter by .62 inches in height) but is cheaper. However, the 4.5 pound pull will most likely not be enough to pull the permanents magnets on the chess pieces.

The fifth DC electromagnet (Type E) is also manufactured by APW Company and has item number EM0755-12-212. This electromagnet requires 1 watt of power and can pull 5.5 pounds with 12 volts. The electromagnet dimension is .75 inches in diameter and .375 inches in height. The housing is finished in Zinc or electroless nickel plating and the magnet comes with 18 inch long leads. The electromagnet weighs .8 ounces and costs $35.36 dollars. This electromagnet is the lightest out of the selection being considered and the smallest. Its weight and size make it an ideal choice to use but the 5.5 pound pull will most likely not be quite enough to move the chess pieces at a smooth and steady rate.

The second to last electromagnet (Type F) is manufactured by APW Company and has item number EM075-6-222. This electromagnet equires 1 watt of power and can pull 6 pounds with 6 volts. The dimensions of this electromagnet are .75 inches in diameter and .62 inches in height. The housing is finished in Zinc or electroless nickel plating. The magnet comes with 18 inch long leads. The electromagnet weighs .96 ounces and costs $29.24 dollars. This electromagnet requires one of the smallest voltages and weighs less than electromagnet "B". It's also one of the least expensive. The 6 pound pull may meet the permanent magnet requirement of 7 pounds.

The last electromagnet (Type G) considered is also manufactured by APW Company and has item number EM075-12-222. This electromagnet requires 1 watt of power and can pull 6 pounds with 12 volts. The dimensions of this electromagnet are .75 inches in diameter and .62 inches in height. The housing is finished in Zinc or electroless nickel plating. The magnet comes with 18 inch long leads. The electromagnet weighs .96 ounces and costs $29.24. The only noticeable difference between this electromagnet and electromagnet “F” is that this electromagnet requires more volts to attract 6 pounds of the permanent magnet. Either way this electromagnet will require more current than the other.

The table below describes a variety of round DC electromagnets that are being considered to be used in the project along with their properties and cost. The project only requires one electromagnet to interact with the small permanent magnets on the chess pieces. The information was gathered by looking at a variety websites that sold small electromagnets. [10], [11].

Type	Size (diameter x height) [inchs]	Volts	Pull [lbs.]	Wt [oz]	Average cost	Overall Advantage
A	.75 x .62	12	4.5	.96	$34.09	
B	.75 x 1.250	12	10	1.7	$55	
C	1 x .719	12	10	1.9	$53	
D	.75 x .62	6	4.5	.96	$29.24	
E	.75 x .375	12	5.5	.8	$35.36	
F	.75 x .62	6	6	.96	$29.24	✓✓
G	.75 x .62	12	6	.96	$29.24	✓

Table 5-7 Round DC Electromagnets

After considering the differences between all electromagnets, the last electromagnet will be purchased for the project. Magnets B and C did meet and exceed the 7 pound requirement set by the permanent magnet study, but 10 pounds might too much of a pull and could create a drag in the movement of the chess pieces across the board. Magnets F and G both came with a pull that would be enough to attract and pull the chess pieces but required 6 and 12 volts of input to do so, respectively. It was decided that magnet G would be the best to use as the power supply supplied by the motherboard (FSP220-60LE) would give 12 volts off of its SATA power connection and it would be easier to connect magnet G to the SATA power without having to lower the voltage for magnet F to use.

Section 5.2.3: XY-Stage with Stepper Motors and Related Diagrams

The XY Stage portion of this section can be broken down into two separate research areas: team assembled stage or a pre-assembled stage. After the initial research on the project was done, it was decided to use stepper motors instead of DC motors or any other type because steppers do not require a closed-loop control system and are very precise and easy to control without having complicated control circuitry. The biggest issue with stepper motors is making sure there is not a backlash of current coming from them into any other circuit and causing those circuits to burn up. The block diagram of the XY-Stage and stepper motor control is located in the following figure.

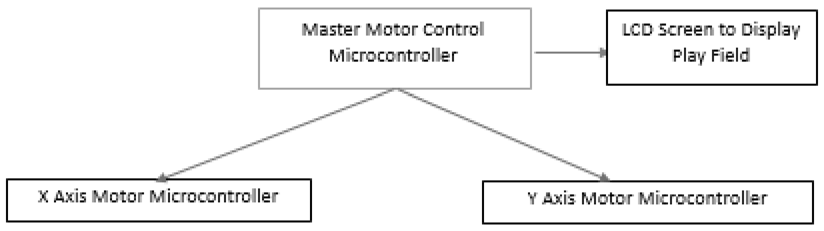

Master Motor Control Microcontroller
LCD Screen to Display Play Field
X Axis Motor Microcontroller
Y Axis Motor Microcontroller

Figure 5-8 XY-Stage Microcontroller and Stepper Block Diagram

Section 5.2.3.1: XY-Stage Mechanical System

A how-to tutorial to build the mechanical portion of the XY-Stage was found on instructibles.com. The stage on this how-to was built in order to allow visitors to a website to draw a design in sand remotely. Acknowledgements go to Mr. Carl S. for designing the low-cost XY-table and sharing the instructions on how to build it with the public.

The whole stage was constructed out of t-slot extruded aluminum in 1”x1”, 1”x2” and 1”x3” dimensions. Sliding glass door acrylic rollers were used to stabilize the stage. Gears were attached to the output shafts of each stepper motor and combined with belts to move the x and y portions of the stage independently.

A brief description of the XY-stage construction will be covered here, but as this is a mechanical design, and not an original design, an appendix with the full instruction set will be included at the end of this document. A photo courtesy of the original designer is provided in the figure below.

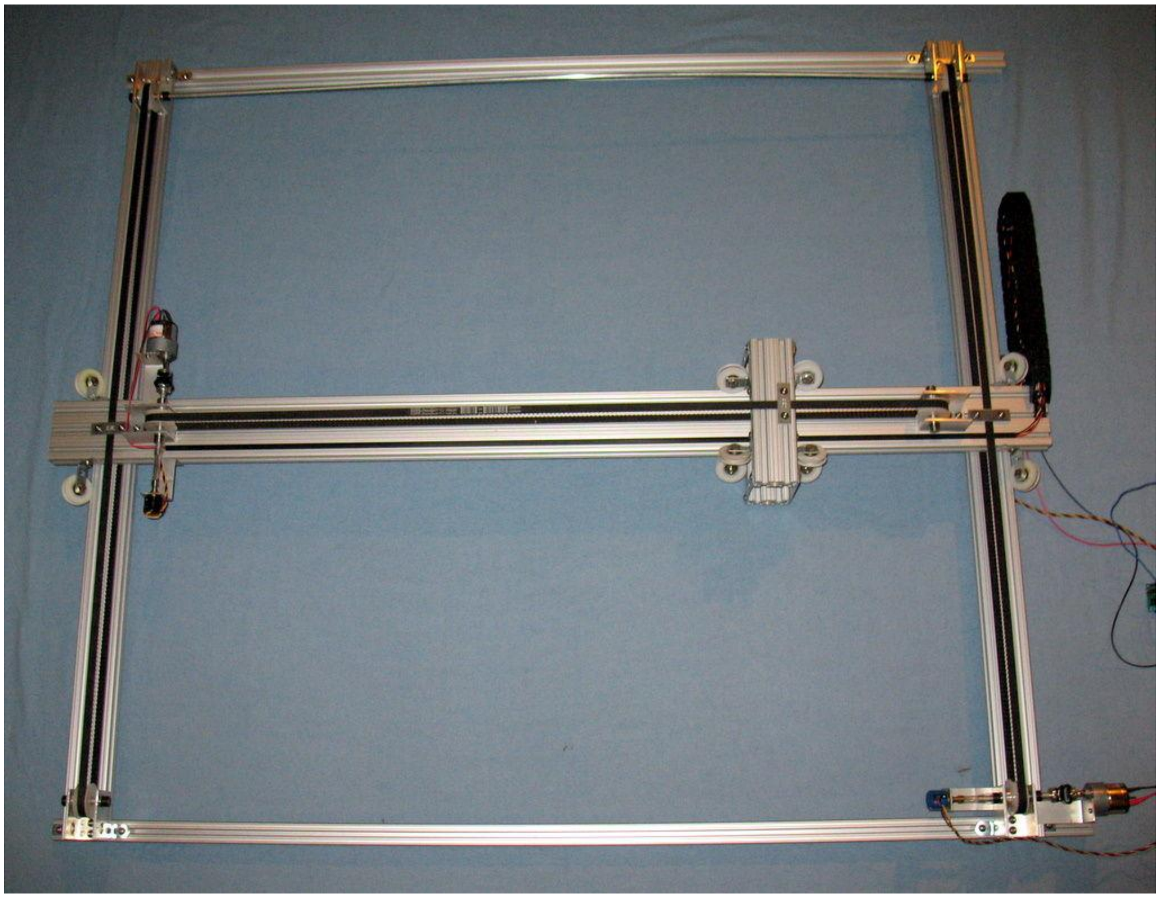


Figure 5-9 Photo of Finished XY-Stage Provided by Carl S. on instructibles.com Permission Granted by Creative Commons License Details in App. A [11]

Also, in the how-to, servo motors were used. This project will instead be using stepper motors for their ease of use and relative cheap price. The next section will discuss the various stepper motors that were considered for this project.

Section 5.2.3.2: Stepper Motors for XY-Stage

Four stepper motors were evaluated based on their cost and the fact that they covered a range of weights, sizes and torques as well as other characteristics. The stepper that was chosen was valued because of its ability to move the mass of the stage as required by the specifications detailing the time spent in moving a piece in Chapter 2. All characteristics will be compared in a table later in this section after a discussion about each motor has been made.

Section 5.2.3.2.1: Stepper Motor 1 ROB-10847

The first stepper motor that was evaluated was a fairly heavy-weight stepper from SparkFun Electronics with an internal product number of ROB-10847. This particular motor is manufactured by Wantai Motor with a product number of 57BTGH420 [12]. This motor is a uni-polar hybrid stepper that is four-phase. It includes a 1/4" shaft that would make connecting this motor to the gear and belt system. This motor is also the most powerful in terms of torque of the four motors that were evaluated. Using this motor would also drive up the current supplied by the power supply to the system. This motor was chosen because of its accuracy of 1.8 degrees per step, and the relative price of the motor.

Section 5.2.3.2.2: Stepper Motor 2 ROB-10848

The second stepper motor that was chosen for evaluation was also from SparkFun Electronics with internal product number of ROB-10848. It is also manufactured by Wantai Motor with internal product number of 39BYGL215A [13]. It is a two phase hybrid motor that has a linear step distance (instead of degrees per step) of 3.9×10^{-4}inches. This motor, since it is very accurate, would minimize errors from overstepping the motor or slightly miscalculating the distance of linear travel at the microcontroller level. It has a threaded shaft, so some thread locker would have to be used to keep the gear in one spot during operation. Care would have to be taken to make sure the belt doesn't shift off the gear due to slipping back and forth on the stepper motor shaft. This motor is $29.95 and in stock at SparkFun.

Section 5.2.3.2.3: Stepper Motor 3 ROB-09238

The third motor that was evaluated was from SparkFun Electronics with an internal product number of ROB-09238. It was manufactured by Mercury Motor with an internal product number of 42BYG011 [14]. The step angle, voltage, and holding torque were similar to motor 2, but the price of the motor is less. It has a 5mm drive shaft, which would require parts measured in the metric system (or extremely close imperial system) measurements. This motor is $14.95 and in stock at SparkFun.

Section 5.2.3.2.4: Stepper Motor 4 ROB-10846

The final stepper motor that was evaluated was also from SparkFun electronics with an internal product number of ROB-10846. The motor was made by Wantai Motor with an internal product number of 42BYGHM809 [15]. The motor is a hybrid, two-phase motor with an advantage in accuracy over stepper motors 1-3 as its step angle is only 0.9 degrees per step. This may cause concern depending on the operating frequency of the microcontroller that is sending pulses to the stepper, though, as it takes more steps to complete a revolution, and therefore, more steps to go the same linear distance as the other motors being considered. This motor is the second cheapest motor in stock at SparkFun, coming in at $16.95, but is not currently available.

Motor Number	Step Angle (deg/step)	Steps/ Rev	Length (in)	Rated Votlage (V)	Rated Current (A)	Holding Torque (lb-ft)	Weight (lb)
Wantai 57BTGH 420	1.8	200	2.2	3.3	2.0	0.651	1.54
Wantai 39BYGL 215A	Not Incl.	200	1.3	12	0.4	0.159	0.40
Mercury 42BYG0 11	1.8	Not Incl.	1.3	12	.33	0.169	0.44
Wantai 42BYGH M809	0.9	400	1.90	2.8	1.7	0.304	0.75

Table 5-8 Stepper Motor Characteristic Comparison [14] [13] [12] [15]

Based on these characteristics, and the advantages and disadvantages noted in the previous sections, the group has narrowed down the choice of stepper motors between stepper 1 and stepper 3. The group feels that a large amount of holding torque is not necessary given the lightness of the pieces and the probability of having to use micro-stepping techniques for accuracy is low.

With the weight of the x-axis crossbar being around 2 pounds including the second stepper motor, the electromagnet, plus the piece of aluminum and the stage, the group has decided to go with stepper motor 1 for its holding torque rating. For the x-direction movement of the stage, the group has decided to go with stepper 3 since the weight being moved will not be as great as the motor for the x-axis crossbar.

Section 5.2.4: Motor Control Board with Microcontroller

The motor control board will consist of a microcontroller, power regulation parts, a motor controller IC, and all interfacing hardware. This section will go through all part selection and requirements of each component and detail the advantages and disadvantages of each part.

Section 5.2.4.1: Microcontroller

This section will outline the choices for the microcontroller that will be interfacing with the motor controller IC to drive the stepper motors of the XY-stage. Three different microcontrollers were considered: the Stellaris M3 LM3S8962, MSP430F5507 and the Atmega 328.

Section 5.2.4.1.1: Stellaris M3 LM3S8962

The Stellaris M3 line of processors designed and manufactured by TI are an ARM based processor line. It has 32-bit RISC performance and 50MHz operation with 33 general purpose IO pins with maximum 5V tolerances. The LM3S8962 has 128kB flash memory with 32kB SRAM and pre-programmed ROM with the Stellaris boot loader and peripheral library. The Stellaris M3 is also capable of I^2C and USB standard communication. The LM3S8962 also has an on-chip low-dropout voltage regulator and a 3.3V brown-out detection circuitry.

The high-level block diagram for the Stellaris M3 LM2S8962 is located in the figure below.

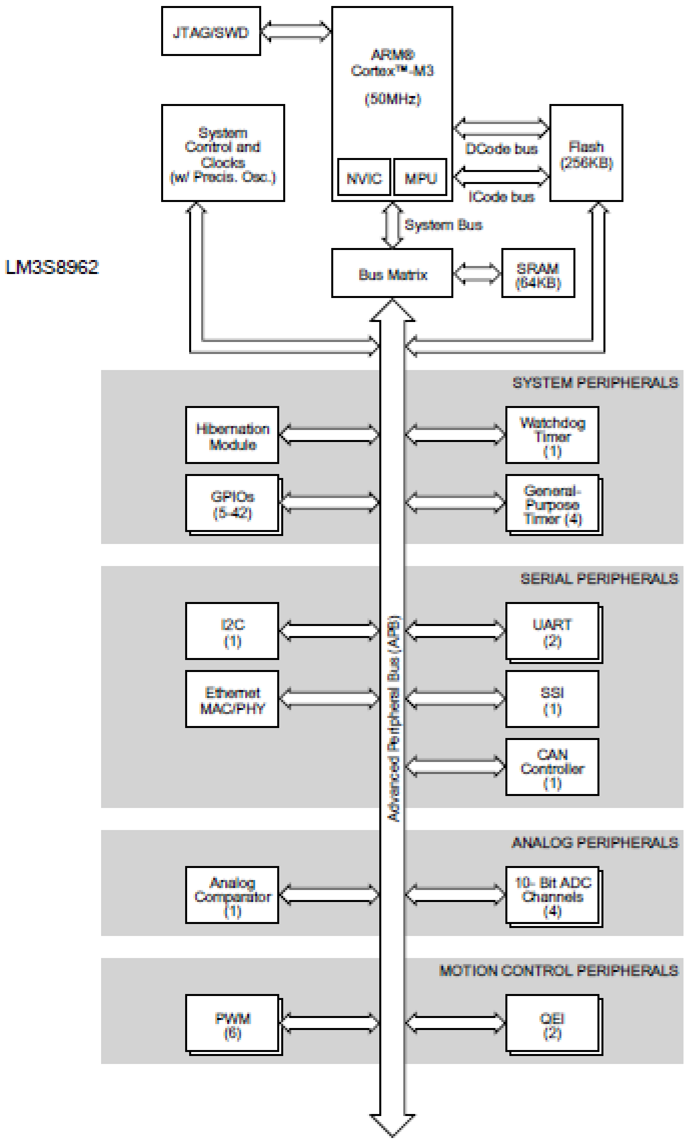


Figure 5-10 Stellaris M3 LM3S8962 High Level Block Diagram - Permission from TI for Use [16]

Section 5.2.4.1.2: MSP430G2553

The Texas Instruments MSP430 line of microcontrollers has a reputation for being low power consumption devices. The MSP430 line has the ability to go into hibernation mode, thereby cutting power consumption. It also has an extremely fast wakeup time when a command is received. The MSP430 is also an affordable microcontroller that supports USB, USCI and I^{2}C communications as well as has the ability to interface with a capacitive touch pad. It is capable of doing 32-bit operations even though it is only a 16 bit microcontroller.

The particular MSP430 microcontroller the group chose to evaluate was a MSP430G2553. This particular microcontroller (unlike the Stellaris) comes in a DIP package, which is easy to solder. This particular model also has Touch-Sense enabled IO Pins to use with a capacitive touch wheel. It has 24 General Purpose IO Pins, watchdog and brown out reset circuit protection, and is readily available at DigiKey [17]. The pin-out diagram and functional block diagram of the MSP430G2553 is located in the two figures below.

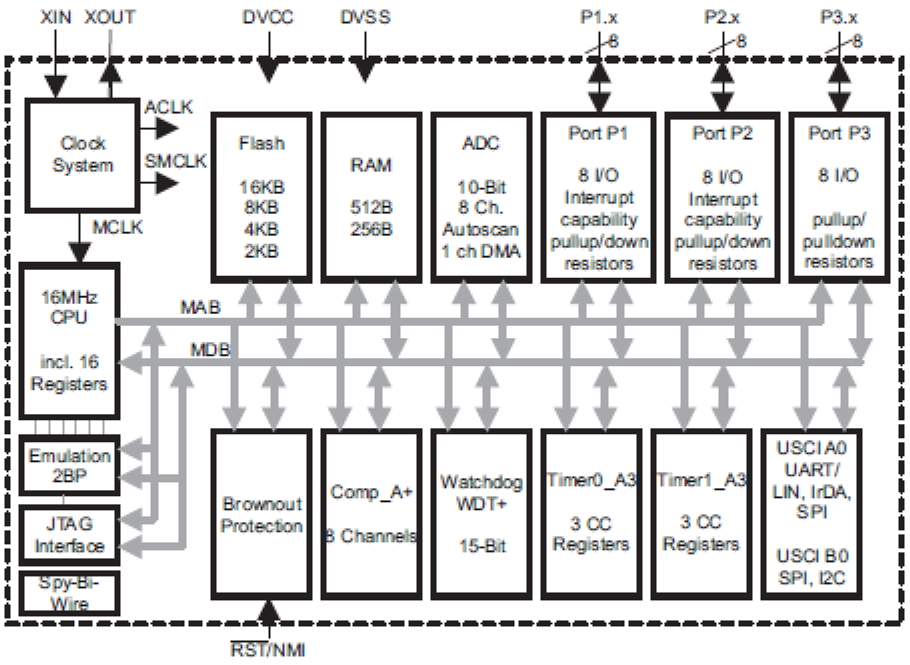
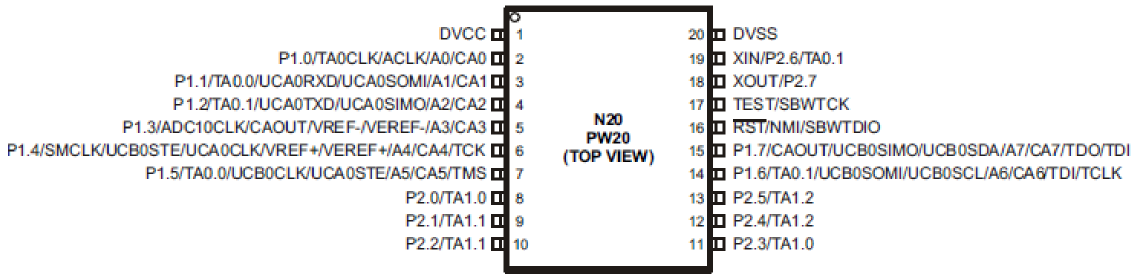


Figure 5-11 Functional Block Diagram for MSP430 - Permission from TI for Use [17]



DVCC 1
P1.0/TA0CLK/ACLK/A0/CA0 2
P1.1/TA0.0/UCA0RXD/UCA0SOMI/A1/CA1 3
P1.2/TA0.1/UCA0TXD/UCA0SIMO/A2/CA2 4
P1.3/ADC10CLK/CAOUT/VREF-/VEREF-/A3/CA3 5
P1.4/SMCLK/UCB0STE/UCA0CLK/VREF+/VEREF+/A4/CA4/TCK 6
P1.5/TA0.0/UCB0CLK/UCA0STE/A5/CA5/TMS 7
P2.0/TA1.0 8
P2.1/TA1.1 9
P2.2/TA1.1 10
N20 PW20 (TOP VIEW)
20 DVSS
19 XIN/P2.6/TA0.1
18 XOUT/P2.7
17 TEST/SBWTCK
16 RST/NMI/SBWTDIO
15 P1.7/CAOUT/UCB0SIMO/UCB0SDA/A7/CA7/TDO/TDI
14 P1.6/TA0.1/UCB0SOMI/UCB0SCL/A6/CA6/TDI/TCLK
13 P2.5/TA1.2
12 P2.4/TA1.2
11 P2.3/TA1.0

Figure 5-12 Pin-out of MSP430G2553 - Permission from TI for Use [17]

Section 5.2.4.1.3: Atmega 328

The Atmega 328 microcontroller is a microcontroller commonly used in conjunction with an Arduino board for easy development. It is an 8-bit microcontroller with up to 23 general purpose IO lines, 1kb EEPROM, 2kb SRAM and five power saving modes. It has a maximum operating frequency of 20 MHz, and 16 touch channels. The Atmega 328 also comes in a 28-pin dip package, which allows for easier assembly when the time comes.

The pin-out diagram for the Atmega 328 is located below.

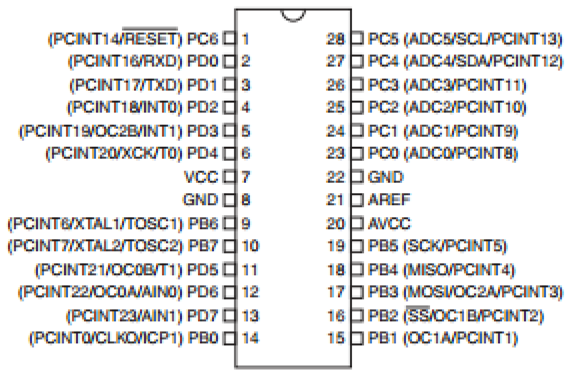


Figure 5-13 Atmega328 Pin-Out Diagram - Permission from Atmel for Use [18]

The important aspects of each microcontroller are compared in the table below.

Value	Stellaris M3	MSP430	Atmega 328
Supply Voltage	3.3 V	3.3V	5 V
Bits	32-Bit	16-Bit	8 Bits
Language Supported	C, Assembly	C, Assembly	C, Assembly
Architecture	RISC	RISC	RISC
Communications	USB, I^2C	USB, USCI, I^2C	
Output Voltage	Min 2.4V	3V	5V
Package	64-Pin LQFP	20 Pin DIP	28PDIP
Cost	$11.25	$2.79	$2.88
Advantage		✓	

Table 5-9 Microcontroller Comparison

For reasons of price, ease of use and ability to immediately acquire the part, the group has chosen to use the MSP430 series microcontroller to relay the stepper motor command to the stepper motor control IC. However, also for ease of use, and having already acquired the part, the group will use a Stellaris M3 evaluation board to provide the microcontroller hub and LCD screen. The evaluation board is further discussed in section 5.2.6.

Section 5.2.4.2: Motor Control IC

This section of the documentation will compare three separate motor control ICs for the control of the stepper motors. This IC will receive its commands from the microcontroller and move the stepper motor accordingly. The three ICs that were compared were the Allegro Microsystems A3967, the Allegro Microsystems A4983, and the TI DRV8818.

Section 5.2.4.2.1: Allegro Microsystems A3967

The first choice for the motor controller IC is the Allegro Microsystems A3967. The A3967 chip has two full h-bridge circuits capable of driving a bi-polar stepper motor. Since the A3967 has a built in translator, it only needs a pulse on the step input to take a full step. Depending on the logic on two logic inputs, the A3967 is also capable of half-steps, quarter-steps and eighth-steps. The A3967 will accept logic inputs from 3-5V, so it would be suitable for use with several different microcontrollers. It also has a drive capability of 30V and 750mA [19]. This motor controller would be suitable for use with stepper motors 2 and 3 only.

Section 5.2.4.2.2: Allegro Microsystems A4983

The second choice for the motor controller IC is the Allegro Microsystems A4983. It is similar to the A3967, but has a drive capacity of 35V and 2A. It also auto-selects the current decay mode, which makes the motor operate quietly and with reduced power dissipation. The A4983 will accept logic inputs from 3-5V, so it would be suitable for use with several different microcontrollers. The A4983 only comes in a surface mount package, so it would have to be soldered to the PCB by a professional mounting company, which adds to cost [20].

Section 5.2.4.2.3: TI DRV8818

The third, and final, choice for the stepper motor controller IC is the Texas Instruments DRV 8818. This control IC is capable of supplying 2.5 A per winding and only requires a single pulse to a STEP input to move the motor forward one step. It, like the Allegro Microsystems stepper drivers, also has available full, half, fourth, and eighth stepping modes available by combination of two logic inputs. The TI stepper motor controller also comes in a package that is able to be soldered by hand instead of machine and is capable of supplying the proper voltages and currents to any of the four stepper motors the group has chosen.

The TI also has the advantage of helping the group qualify for the TI design competition and having numerous design examples available on their website for use [21].

Value	Allegro A3967	Allegro A4982	TI DRV8818
One Pulse Step	✓	✓	✓
Full, Half, Fourth and Eighth Modes	✓	✓	✓
Max Voltage Supply	30V	35V	35V
Max Current Supply	750mA	2A	1.9A
Package Type	24-Pin SOIC	20 PDIP	28-Pin HTSSOP
Step Frequency	500kHz	500kHz	500kHz
Documentation	✓	✓	✓✓✓
Cost	$3.51	$3.46	$6.88
Advantage			✓

Table 5-10 Motor Controller IC Comparison [19] [20] [21]

The advantage for the motor control chips goes to the Texas Instruments DVR8818 because it would help the group qualify for the Texas Instruments Design Competition and because of the numerous design examples provided by Texas Instruments. The design examples make an easier job of implementing the chip in the circuit and also provide a reference to make designing the motor control board with a microcontroller much simpler.

Section 5.2.4.3: Motor Controller PCB

This section will display the schematic and planned PCB for the two motor controllers and the one or possibly two microcontrollers necessary for their operation. The figure below displays the Schematic layout of the board.

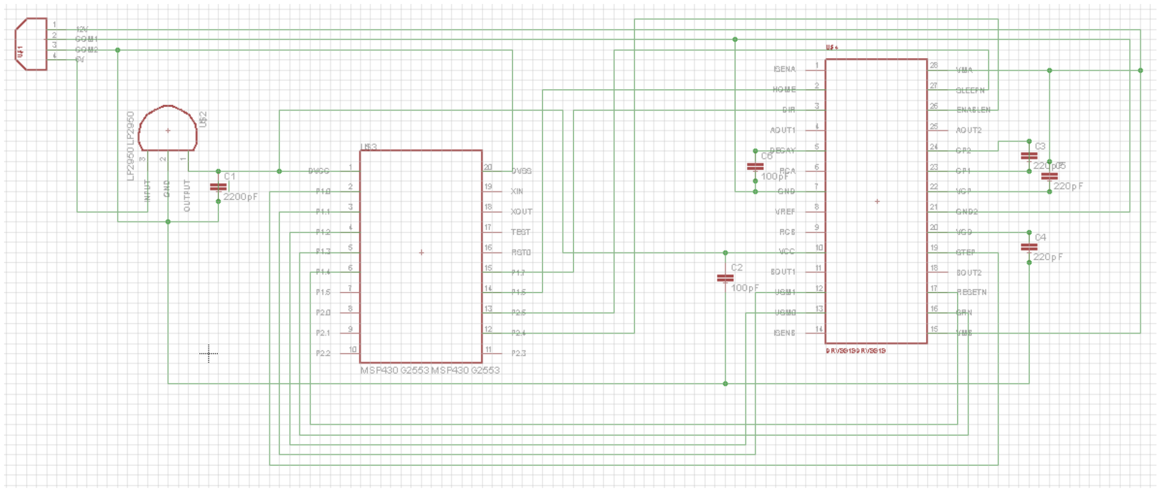


Figure 5-14 Schematic Diagram of Motor Control PCB

Section 5.2.5: Voice Capture Hardware

The hardware component which would running the voice capture will be a simple Logitech microphone. The microphone would be connected to the computer directly through USB. The microphone would be responsible for simply capturing the voice commands and transmitting the command into the computer for the voice library, CMU Sphinx to use and interpret. The microphone must be able to receive sound at the frequency of 60 to 7000 Hz at the minimum, since this is about the widest range of sound produce by the human voice. Fortunately, just about every microphone out on the market is more than capable of fulfilling this requirement. As a result there should be no issues involving the microphone not being able to correctly record the voice commands sent by the player. Another issue is that the microphone should have an off/on button or a button to turn the microphone on when one of the player wishes to issue a command for the voice library to interpret.

Section 5.2.6: LCD Screen and Related Diagrams

For the LCD screen, a team member already owned a Stellaris evaluation kit from Texas Instruments. The evaluation kit comes with a LCD screen built in, and has numerous communication methods that would make communication not only with the voice recognition computer, but with the MSP430s that are comprise the motor control network as well. Some of the features of the Stellaris M3 evaluation kit include:

- 10/100 Ethernet Controller
- OLED graphics display 128x96 pixel resolution

- Navigation buttons
- Speaker
- MicroSD card slot
- USB interface for power supply

These and other features of the Stellaris EKS-LM3S8962 can be found in the user manual [16].

The block diagram of the Stellaris EKS is in the figure below:

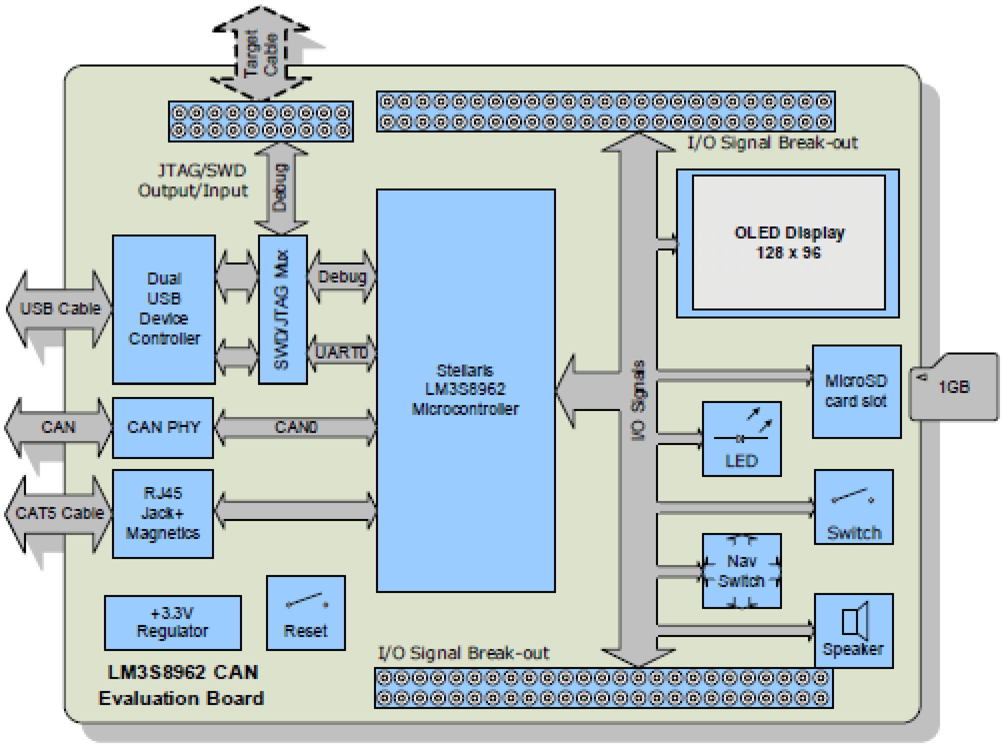


Figure 5-15 Stellaris EKS Block Diagram - Permission from TI for Use [16]

Section 5.2.7: AI/ Voice Recognition Computer and Related Diagrams

In order to run the AI component of the project a motherboard was required. The project needed a motherboard that was affordable, didn’t require a lot of power, small in size to fit comfortably in the chess board, and light weight. Also the project would require enough RAM the software and a CPU that could run the program quickly so there would be no additional delays. In order to save room inside the chess board the group decided to look specifically at Mini ITX boards.

Three combinations of motherboard, CPU, RAM and power supply were put together to compare on the size, speed, power consumption and cost.

The first motherboard combination (Combination A) considered, consisted of an ASRock AD525PV3 motherboard that came with an Intel Dual-Core Atom Processor D525 (1.8GHz operating frequency) and two DDR3 DIMM RAM that could store up to 4 GB of memory. The motherboard was 6.7 inches x 6.7 inches in size (as all mini-itx form factor motherboards) and weighed 3 pounds. The motherboard required an ErP/EuP ready power supply. The power supply chosen for the motherboard was a FSP220-60LE(80) that had an input voltage of 230 V and an output of +3.3V at 14A, +5V at16A, +12V1 at 16A, +12V2 at 10A, -12V at 0.8A, +5VSB at 2.5A. The power supply was 5.91 inches x 3.21 inches x 1.6 inches in size and weighed about 3.0 pounds. The power supply also came with two SATA power connectors which would be useful to possibly power the microcontroller, electromagnet and the stepper motors controlling the XY stage. The possibility of this will be discussed further in section 3.2.6. The total cost for the motherboard/CPU combo, RAM and power supply is about $148.99 plus shipping. However, the motherboard/CPU combo and RAM were donated to the group. Bringing the actual cost down of combination A to just the power supply cost at $45.99 plus shipping.

The second motherboard combination (Combination B) consisted of an ASRock motherboard model number A75M-ITX. The motherboard comes with an FM1 socket type and therefore can accept A4/A6/A8/E2 APU CPU types. The CPU chosen for the motherboard was an AMD brand A-series APU model number AD3400OJHXBOX with an operating frequency of 2.7GHz. The motherboard accepts DDR3 2400+(OC) / 1866(OC) / 1600(OC) / 1333 / 1066 / 800 standard memory. Therefore, the chosen RAM for this combinations was the Kingston HyperX Blu Red Series 8GB DDR3 1600 model number KHX16C9B1RK2/8. The RAM product comes with 2, 4GB RAMS equally the 8GB mass storage amount. The power supply was the same as for combination A (a FSP220-60LE(80) that had an input voltage of 230 V and an output of +3.3V at 14A, +5V at16A, +12V1 at 16A, +12V2 at 10A, -12V at 0.8A, +5VSB at 2.5A. Also comes with 2 SATA power connectors and weighs 3.0 pounds) because of it meeting the required ErP/EuP power requirement set by the manufacturer of the motherboard. The total cost for the motherboard, CPU, RAM and power supply is about $224.96 plus shipping.

The third motherboard combinations (Combination C) consisted of a Motherboard/CPU combo. The motherboard is manufactured by ASRock model number is E350M1. The CPU that comes with the combo is an AMD E-350 APU Dual-Core with a 1.6GHz operating frequency. The motherboard accepts DDR3

1066 types of RAM. The RAM chosen for combination C was manufactured by Crucial model number CT2KIT25664BA1067. The RAM is 2, 2GB sticks of RAM therefore comes with 4GB capacity. The power supply was the same as for combination A (a FSP220-60LE(80) that had an input voltage of 230 V and an output of +3.3V at 14A, +5V at16A, +12V1 at 16A, +12V2 at 10A, -12V at 0.8A, +5VSB at 2.5A. Also comes with 2 SATA power connectors and weighs 3.0 pounds) because of it meeting the required ErP/EuP power requirement set by the manufacturer of the motherboard. The total cost for the motherboard/CPU combo, RAM and power supply is $180.97 plus shipping.

The tables below summarize the description of each motherboard / CPU / RAM / Power Supply combinations.

Combination A

Part	Brand and Model Number	Operating frequency	Memory	Input Voltage	Weight	Cost
MB	ASRock AD525PV3	-	-	-	1.5 lbs	~$90.00
CPU	Intel D525 (Dual-Core Atom)	1.8 GHz	-	-		
RAM	Patriot G2 PQG316G1600ELQK	-	8GB	-	-	$13.00
Power Supply	FSP GROUP FSP220-60LE(80)	-	-	230 V	3 lbs	$45.99
Total*	-	-	-	-	4.5 lbs	$148.99

Table 5-11 Motherboard Combination A

*Total cost includes estimated cost of the motherboard/CPU combo and RAM which as mentioned earlier were donated to the group. Total cost excludes shipping.

Combination B

Part	Brand and Model Number	Operating frequency	Memory	Input Voltage	Weight	Cost
MB	ASRock A75M-ITX	-	-	-	1.3 lbs	$89.99
CPU	AMD AD3400OJHXBOX	2.7GHz	-	-	-	$48.99
RAM	Kingston KHX16C9B1RK2/8	-	8GB	1.65V	.8 lbs	$39.99
Power Supply	FSP GROUP FSP220-60LE(80)	-	-	230 V	3 lbs	$45.99
Total*	-	-	-	-	5.1 lbs	$224.96

Table 5-12 Motherboard Combination B

*Total Cost excludes shipping.

Combination C

Part	Brand and Model Number	Operating frequency	Memory	Input Voltage	Weight	Cost
MB	ASRock E350M1	-	-	-	1.7lbs	$109.99
CPU	AMD E-350 (APU, Dual-Core)	1.6GHz	-	-		
RAM	Crucial CT2KIT25664BA1067	-	4GB	1.5V	0.14lbs	$24.99
Power Supply	FSP GROUP FSP220-60LE(80)	-	-	230 V	3 lbs	$45.99
Total*	-	-	-	-	4.84 lbs	$180.97

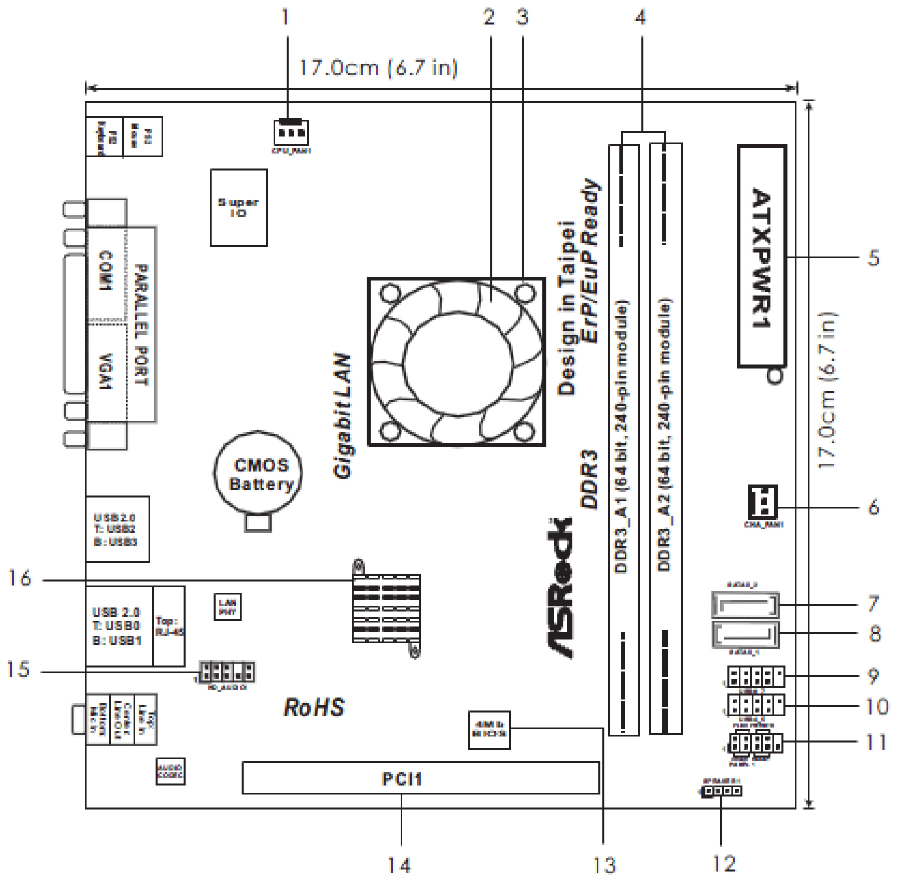
Table 5-13 Motherboard Combination C

*Total cost excludes shipping.

Comparing all three combinations against each other; combination B would be the most desirable for its high clock rate that would execute the AI program the fastest. However the 8GB of RAM is costly and at most half of that memory will be used during the game. Combination B is also the most expensive out of the three combinations and weighs the most.

Combination C is the second most expensive and has the slowest clock rate out of the three combinations. A slow clock rate is extremely undesirable to the project because when the player is playing against the AI we want the response time to be as fast as it can be along with allotting for the amount of time it will take to move the pieces across the board. As stated in the project specifications, the chess engine should respond in under or equal to 5 seconds and the pieces need to be at their proper locations in under or equal to 3 seconds. Therefore the CPU needs to take as little as possible time to respond to have the time limits met.

Combination A has a good clock rate of 1.8 GHz and enough memory storage of 4 GB. Also the Motherboard, CPU and RAM were all donated to t the group leaving only the power supply needed to be purchased. Combination A meets the project specifications and costs the least at about $45.99 (plus shipping) for the power supply. Therefore the project will most likely use the parts listed out in combination A. Below is the motherboard layout of the AD525PV3. Permission was given by ASRock to reprint the diagram of the motherboard from their user manual attached in the appendix.



1	CPU Fan Connector (CPU_FAN1)	9	USB 2.0 Header (USB6_7, Blue)
2	CPU Fan	10	USB 2.0 Header (USB4_5, Blue)
3	CPU Heatsink	11	System Panel Header (PANEL1, White)
4	2 x 240-pin DDR3 DIMM Slots (DDR3_A1, DDR3_A2; Blue)	12	Chassis Speaker Header (SPEAKER 1, White)
		13	BIOS SPI Chip
5	ATX Power Connector (ATXPWR1)	14	PCI Slot (PCI1)
6	Chassis Fan Connector (CHA_FAN1)	15	Front Panel Audio Header (HD_AUDIO1, White)
7	Secondary SATAII Connector (SATAII_2; Blue)		
8	Primary SATAII Connector (SATAII_1; Blue)	16	South Bridge Controller

Figure 5-16 ASRock AD525PV3 Motherboard Layout. Permission given to reprint diagram from the AD525 / AD425PV3 User Manual

Section 5.2.8: Board Construction

The board construction will largely influence the decisions made on which type of magnets are to be used. First, the board was chosen and is made of glass. After the board was chosen, two sets of pieces, one plastic and one glass were measured to determine diameter and weighed to determine weight. Finally, two sets of magnets, one set of neodymium and one set of ferrite magnets were tested to determine the best type of permanent magnets to use as well as the strength required of the electromagnet to interact with the permanent magnets on the bottom of the pieces. This experimentation process is documented below and includes tables of piece diameter, board thickness, magnet strength and dimensions and space dimensions.

Section 5.2.8.1: Play Area Dimensions and Material

Since a team member already had a chess board ready to use for the project, that met the requirements in Chapter 2, the team decided to base all further research on that chosen board. The chosen board and its dimensions and material are located in the table below.

Aspect	Value
Material	Glass
Chess Play Area Dimensions	11.5” x 11.5”
Chess Board with Bezel	13.69” x 13.69”
Chess Board Depth	.1890”
Chess Board Square Size	1.375” X 1.375”

Table 5-14 : Board Dimensions and Material

Section 5.2.8.2: Piece Dimensions and Materials

After the board was chosen, the issue of several pieces reacting to the magnetic field of the electromagnet at the same time was discussed. Two materials that are known insulators were chosen to be tested to minimize the chance of this occurring. The two materials that were chosen were plastic and glass as they are both readily available and inexpensive to buy and test. The two characteristics of each set of pieces to be tested include both diameter and weight. The following two sections discuss the necessity of testing these two characteristics and including them in the decision making process.

Section 5.2.8.2.1: Diameter of the Piece Bases

The issue of the diameter of the pieces were discussed when the problem of moving the knight, since it can jump over other pieces, was discovered. As the pieces were going to be moved in a plane, simply picking up the piece and moving it was not an option. Two alternatives were discussed: having an algorithm that would move pieces out of the way so the knight could get through, or having the spaces large enough and the pieces small enough that the knight could fit in between the other pieces when moving. To have pieces small enough to fit in between one another, the pieces had to be, at the largest, half the size of the square size. Doing the math, the diameter of the pieces had to be, at most, 0.6875” to fit between one another while moving. All measurements are in inches, and were taken using an analog micrometer accurate to a ten-thousandth of an inch and are in table 5-2.

	Measurements and Material	
Piece	**Glass**	**Plastic**
Pawn 1	0.9306	0.6837
Pawn 2	0.9424	0.6818
Pawn 3	0.9341	0.6800
Pawn 4	0.9448	0.6846
Pawn 5	0.9254	0.6830
Average Pawn	0.9355	0.6826
Queen	0.9323	0.7355
King	1.0150	0.7340
Rook	0.9310	0.6745
Bishop	0.9339	0.6767
Knight	0.9270	0.7577
Average Piece Size	**0.9417**	**0.6991**

Table 5-15 Piece Material and Diameter

The plastic pieces clearly had the advantage as they have a significantly smaller diameter than the glass pieces. The next characteristic to be tested would determine if there was an overall better choice, or if the team would have to compromise and come up with a solution to deal with the problems that arise when making compromises.

Section 5.2.8.2.2: Weight of Pieces

Another deciding factor of which set of pieces to use was the weight of the pieces themselves. Heavier pieces require stronger magnets, therefore driving up the cost of the project. The weights of the pieces are in the table below and are measured in ounces.

Piece	Measurements and Material	
	Glass (oz.)	**Plastic (oz.)**
Pawn 1	0.51	0.06
Pawn 2	0.51	0.06
Pawn 3	0.50	0.06
Pawn 4	0.51	0.06
Pawn 5	0.48	0.06
Average Pawn	0.50	0.06
Queen	0.93	0.13
King	1.51	0.13
Rook	0.62	0.08
Bishop	0.71	0.08
Knight	0.83	0.13
Average Piece Weight	**0.69**	**0.08**

Table 5-16 Piece Material and Weight

Given the size and weight of the pieces chosen for evaluation, the clear choice is the plastic pieces. Not only are they eight times lighters than the glass pieces, but they are also much smaller in diameter – thereby eliminating the necessity of an algorithm to move pieces out of the way for a knight piece to move its intended path.

Section 5.2.8.3: Board Supports and Frame

For the board supports and frame, the team agreed that the material composition of the frame should be wood or plastic to insulate from electromagnetic waves, and also be easy to cut and attach together to form a cohesive, well-built frame. The two materials that were chosen were wood and plastic; the plastic frame being made from acrylic sheets being glued together. Another consideration of the frame for the board was whether or not it would be conducive to showing the internal mechanisms of the board itself. The group thought this last aspect important when showing the final project to the panel of professors.

The frame dimensions and layout is diagramed in the figure below for clarity.

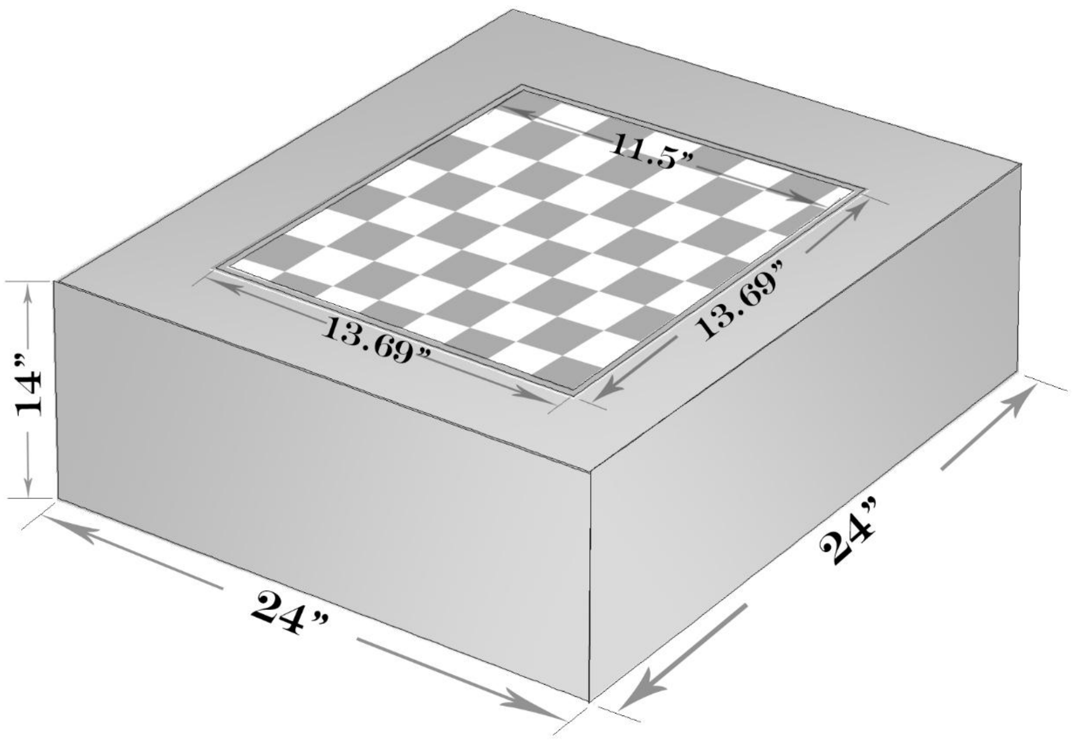


Figure 5-17 Chessboard Dimensions

Section 5.3: Software Implementation Details

Section 5.3.1: Chess Engine Control Module

The Chess Engine Control module will use an open-source program called TSCP, or Tom’s Simple Chess Program. Permission from Mr Kerrigan is currently pending, but he has stated that he approves all uses of TSCP as long as they credit him and do not attempt to profit off of it. Most of TSCP will be left as-is. The only primary additional feature that will be added is the varying difficulty level. This will be implemented by limiting the depth of the search tree to different degrees based on the user’s wishes. It is infeasible to deepen the search tree because this would violate all of the qualities that make TSCP a good engine – in particular, its small size. Instead, the current depth at which it operates (approximately 6 ply) will be used as the “hard” setting. For a “medium” setting, the search will be forced to be 4 ply, and for the “easy” setting, 2 ply.

TSCP will need to be compiled in Debian Linux. It supports the XBoard protocol, which is native to Linux. This should not be a difficult task, which is why, in the Final Coding Plan section, the deadline for having this done is rather early.

After the difficulty levels are complete and the program has been configured for Debian, the final implementation detail of concern is the interface. TSCP takes algebraic notation as input and produces the same as output. TSCP also outputs the results of each level of searching as well as an ASCII representation of the board, but this will be of little use. It will be necessary to create an interface class between the chess engine and both the voice recognition and magnetic controller modules to translate their respective inputs and outputs. This way, the translation can happen outside the primary software units, which increases modularity, a desirable trait in large systems.

Section 5.3.2: Voice Recognition Software

CMU Sphinx is the speech recognition software chosen to run on the computer system alongside with the chess engine for this project. To be more specific, the version of CMU Sphinx being used will be version 4 and it will either be the main version of Sphinx or PocketSphinx depending on which of the variant is more compatible and less error prone in terms of compiling on the Debian operating system. Both variants functions are basically the same. The difference is for which system they were designed for. The main version of Sphinx is written in Java and is designed for general purpose machines such as desktop computers. The other version PocketSphinx is written in C and is designed for embedded systems. As such it can also run computer system desktops. Both versions require the source code to be downloaded and compiled on the operating system. However, since C is lighter on resources than Java, it is most likely that the PocketSphinx version will be used.

Sphinx itself will be responsible for interpreting voice commands from both players and translating that command into a command that is understood by the chess engine. In order to do that, Sphinx will need to be able to understand and recognize short verbal phrases issue by the player. As such it will be required to understand algebraic expression for chess, which is a standard used to record chess matches. In addition, it will also be trained to understand and recognize informal commands used by players such as “Pawn to D4”. The language model that comes with sphinx should be more than enough to interpret and translate English vocal commands given by players. However, if it’s not enough, Sphinx is capable of being trained. But that should not be required as the basic English language acoustic model should be more than enough to interpret and translate all the possible commands by both players.

The voice recognition portion will not simply be able to tell the difference between either of the players. It will accept commands and send them to the chess engine. The engine will know whose turn it is, and therefore assume that it is that player that issued the command. Therefore the Sphinx will not need to recognize which voice command is from which player. It will accept all voice commands and translate them into a command for the chess engine.

Section 5.3.3: Magnetic Controller Module

The function of the magnetic controller is to dictate how the electromagnet will move the pieces throughout the game. The requirements for this module are that it must receive physical coordinates that describe the move that is about to occur, and translate these into a series of pulses to the stepper motors to control the electromagnet to make the desired move. Because separate motors control the magnet's position on the x-axis and the y-axis, they can operate essentially simultaneously, meaning the motion of the magnet can be diagonal when possible, as opposed to being on a fixed grid. This will slightly increase the responsiveness of the system.

The program that controls the stepper motors will be coded in C for simplicity and to be lightweight. One issue that will have to be dealt with is the fact that in some cases, usually opengame, the chess engine will finish deciding what move to make and send it to the magnetic controller before the controller has physically finished performing the first move. Therefore, the magnetic controller must be constantly listening for input, and must have some sort of buffer or queue to handle multiple requests in order. It was decided that the maximum number of moves it would need to hold in a request queue would be about 5. This carries the assumption that the users will operate the system as intended and not try to flood it with multiple requests. Due to the somewhat slow nature of Chess, input overflow will not be a problem with the other modules, as users typically need to ponder their moves for at least a few seconds, the maximum amount of time in which the software modules are expected to complete their functions.

Section 5.3.4: Operating System Design Summary

The operating system that will most likely run on the computer component of the project will be Debian Linux. The installation of the Debian Linux operating system will be a baseline install. It will only be loaded with baseline programs and drivers needed to operate all the hardware components of the project. Then the selected chess engine will be installed and compiled on the operating system along with any required dependencies. Afterward, CMU Sphinx will be compiled and installed onto the operating system. In addition, the language model will also be loaded onto the operating system to save time on training the speech recognition library. As such Debian will be responsible for running both the chess engine and voice recognition software. In addition, it will also be sending commands from the chess engine to the microcontroller to run the electromagnetic component which will handle moving the chess pieces.

Additionally, Debian will also be adjusted to remove any unnecessary programs that are often preloaded with the installation of the operating system. However, dependencies will be kept as it is possible that some of the programs required to run the operating system utilize those dependencies.

Chapter 6 : Project Prototype Construction and Coding

Section 6.1: Parts Acquisition and BOM

This section will be broken up into sections by blocks of the block diagram. Each block of the block diagram will have a section.

Section 6.1.1: Power System

Item	Part #	Status
Motherboard Power Supply	FSP220-60LE(80)	

Table 6-1 Power System Parts Acquisition and Bill of Materials

Section 6.1.2: Electromagnet and Permanent Magnet

Item	Part #	Status
Small Disk Magnet		Partially Acquired
Round DC electromagnet	EMO75-12-222	

Table 6-2 Magnetics Acquisition and Bill of Materials

Section 6.1.3: XY Stage and Stepper Motors

Item	Part #	Status
1"x1" T-Slot	1010	
1"x3" T-slot	1030	
1"x2" T-slot	1020	
3/4" Angle Bracket	M 1556A24	
1" Angle Bracket	M 1556A41	
1.5" Angle Bracket	M 1556A42	
4" Plate	M 1394A34	
10-32 X 3/8" Stainless screws	M 92949A263	
10-32 x 1/2" Machine Screws	33121	Acquired
Square Nuts 3/8" Width	M 94785A411	
#10 Flat Washers	32481	Acquired
#10 Lock Washers	20211	Acquired
1/4"-20 x3/4" cap screws	27991	Acquired
1/4-20 Jam Nuts	27991	Acquired
1/4" Lock Washers	20551	Acquired
Split Lock Washer 1/4-20	M 92146A029	
Sliding Glass Door Rollers	622001	
Aluminum Bare Angle	6063-T52	
SS Ball Bearing ABEC-5	M 57155K323	
18-8 Stainless Steel Bearing Shim	M 93574A513	
Pulley for XL-Series Timing Belt	M 57105K21	
Timing Belts	M 6484K454	
Black-Oxide Steel Shaft Collar	M 9414T6	
Steel Flat Mending Bracket	M 1394A31	
Stepper Motor #1	ROB-10846	
Stepper Motor #2	ROB-10847	
Stepper Motor #3	ROB-09238	

Table 6-3 XY Stage and Stepper Motors Parts Acquisition and Bill of Materials

Section 6.1.4: Microcontroller, PCB and Motor Controller IC

Item	Part #	Status
PCB Fabrication		
Texas Instruments DVR8818	DVR8818	
Texas Instruments MSP430	MSP430	

Table 6-4 Motor Board PCB Parts Acquisition and Bill of Materials

Section 6.1.5: Voice Interpretation and Interfacing

Item	Part #	Status
Hard Drive for AI Unit	Vertex RS 60GB	
Mini ITX Motherboard and Processor	ASRock AD252PV3	Acquired

Table 6-5 Voice Interpretation Hardware Acquisition and Bill of Materials

Section 6.1.6: LCD Screen and Interfacing

Item	Part #	Status
Texas Instruments Stellaris M3 Development Board	EKS-LM3S8962	Acquired

Table 6-6 LCD Screen Parts Acquisition and Bill of Materials

Section 6.2: Assembly

Section 6.2.1: Power System

The power system consists solely of the power supply for the motherboard. The motherboard will be connected to the FSP220-60LE(80) power supply by the supplies 20(+4) main pin connector (P1; see figure 5). The Serial ATA pin connectors from the motherboard power supply will be connected to the motor controller and microcontroller. The SATA connector delivers +12 volts, which will be used to power the motor controller, and +3.3 volts which will be used to deliver power to the microcontroller. Any possible additional electronic components added to the chess board will be able to be powered by the second 15 pin SATA connector.

Section 6.2.2: Electromagnet and Permanent Magnet

The permanent magnets shall be manually glued to the interior of the felt lining on the bottom of the plastic pieces. Care will be taken to position the magnet at the exact center of the piece to ensure that the piece gets placed in the center of the chess square every time it is moved. The felt will then be re-attached to the bottom of the piece, thereby housing the magnet in the interior of the piece itself.

The electromagnet shall be fixed face-up on the XY-Stage underneath the glass board. The electromagnet will be fixed to the XY-stage by an angle bracket connecting the mounting holes of the electromagnet to the t-slot that is making up the stage itself. The t-slot makes it convenient to align the electromagnet to the exact center of the stage to help aligning the magnet properly to pick up to correct piece and place it correctly.

Section 6.2.3: XY Stage and Stepper Motors

As a full disclosure, this low-cost XY-stage was designed by Mr. Carl S. on instructibles.com. It fit the project requirements perfectly, and also remained in the budget set forth for this section of the project. Thanks to Carl S. for posting this information for others to use!

The basic parts for the mechanical portion of this design are:

- 1”x1”, 1”x2” T-Slot extruded Aluminum
- Miscellaneous hardware for stabilization and assembly
- Pulleys, Timing belts, shaft collars, shafts, and plates for translating the rotational motion of the electric motors to linear motion
- Motors, and their associated hardware.

For this project, the 1”x1” is attached to the 1”x2” T-Slot using angle brackets to form a square frame for the support structure and the x-axis guide. The y-axis support is formed out of more 1”x1” T-Slot and is bridged across the two x-axis T-slot guides. The actual stage and its carriage are formed from the 1”x2” T-slot sandwiching the 1”x1” T-slot that forms the y-axis support. They are stabilized by sliding glass door rollers made of acrylic that make sure there are no bumps or de-railings during movement of the stage.

For the pulley and belt system, pulley brackets are made for the timing belt pulleys and are attached to the t-slot of the x and y axis. Plates are used to tighten the belts to the moving portions of the x and y axis and can be adjusted as necessary depending on length.

For this project, the team will also need to adjust the dimensions of the stage setup in the instructables how-to.

Section 6.2.4: Motor Control PCB and Components

The assembly for the motor control PCB and all its required parts will take place in stages. First, the board will be prototyped on proto-board before being designed in a schematic/PCB board editor. This is to ensure that the design done in the PCB editor (which will most likely be Eagle by Cadsoft) will be a working model.

For the proto-board construction, the parts will be soldered on using lead-based solder and a Hako 396 soldering iron. Parts will be tested to ensure good adherence to proto board pads to make sure that, if there is a problem, it is with the design and not the solder work.

Section 6.2.5: Voice Interpretation and Interfacing

The voice interpretation unit shall be a Linux machine running the voice interpretation software. The motherboard will be connected to the power supply by a 20-pin power plug and will have the RAM inserted into the proper slots. The single hard drive will be connected through to the motherboard through a SATA port and be powered through a SATA power plug. The voice unit shall be interfaced to the chess engine though a USB or regular serial port.

Section 6.2.6: LCD Screen and Interfacing

Since the LCD screen is part of a Stellaris evaluation kit, there is almost no assembly required. The only assembly of the LCD Screen and AI component will come from interfacing it to not only the PC for the voice commands, but also to the MSP430s that will make up the motor control board. This will require up to 5 different output pins coming from the Stellaris itself connected to a UCB port or to a serial port for the PC and connected directly to input lines for the MSP340 that it needs to go to control the motor it needs to control.

Section 6.2.7: Board Frame Assembly

For the board frame assembly, the team will cut the acrylic into the sections required and glue them together with industrial strength glue. The pieces shall be held together while the glue is drying with clamps. The bottom and the sides are going to be assembled first and the PC, Stellaris EKS and motor control PCB will be fitted next to drill mounting holes.

The bottom of the case will be fitted with rubber feet to support the weight of the whole case and make sure that the mounting screws are not making contact with the surface the chessboard is sitting on.

Next, there will be ventilation holes cut and fans mounted to keep the unit cool during operation. There will also be an EMF shield comprised of aluminum foil stretched across the frame to shield the electronics from the magnetic field generated by the electromagnet.

On top of the shield, triangular pieces will be glued in the corners of the frame to provide strength to the case and to provide a place to sit the frame of the XY-stage component. After the XY-Stage component is mounted, all the wires will be routed as to not get tangled when the stage is moved.

Finally, the frame will be topped with the glass chess board the team has chosen to be the play surface.

Section 6.3: Final Coding Plan

The final version of the software block diagram was outlined in an earlier section. Magic Chess will use open source software for the chess engine and voice recognition and custom software for the magnetic controller. Everything will be run on Debian Linux. The plan for completing the software development and integration is detailed below.

Milestone	**Deadline (Week ending)**
Debian Linux configured and installed on board	January 18
Chess Engine(s) fully modified and configured for Linux	February 1
Voice recognition library completed and functioning	February 22
Magnetic controller coded and correctly operating motors	March 15
Software interfaced. Complete full turn as shown in block diagram	March 29

Table 6-3: Final Coding Plan

The pacing of these milestones was chosen to allocate more time to tasks with a greater perceived difficulty. For example, because Linux comes complete and ready to install, it should not be difficult to get the operating system up and running. It should be completed within two weeks after the beginning of the semester. In contrast, customization of the open-source software and the development of original software pose greater challenges, so those milestones were spaced three weeks apart. Interfacing will be worked on throughout the semester, but should definitely be complete by the end of March. Completing everything by the end of March should allow a couple weeks of slack time for continued testing, dealing with unforeseen consequences and technical issues, and preparing for presentation of the year’s work.

Chapter 7 : Project Prototype Testing

For the project prototype testing, the group has decided on a bottom-up approach to make sure each portion is correct before combining them. The testing process for each portion is discussed below in section 7.2.

Section 7.1: Hardware Test Environment

The hardware test environment will consist of an oscilloscope, multi-meter and power supply. Some alligator clips also may be necessary for testing. The hardware testing will mostly take place in the Senior Design lab provided by the College of Engineering and Computer Science at the University of Central Florida.

Section 7.2: Hardware Specific Testing

The hardware specific testing will be done system by system to ensure that each system works properly before combining them to minimize sources of potential problems. The systems will also be combined one-by-one to minimize sources of problems and to work out any bugs before adding more systems.

Section 7.2.1: Power System

For the power system portion that is going to power the motor control PCB, the team is going to test the output of all the pin connections supplied by the power supply. Most importantly the Main 20(+4) pin connector and the PC and PD Molex connectors. The plug will also be tested to fit against the power port on the PCB. All test results will be included here in the revised document. The tables below list out each step the group needs to take to test the power supply for proper function and adequate power supply.

The first test of the power supply will be its 24 pin main connecter. Each pin of the power supply needs to be measured using a multimeter to check that each pin’s signal as specified in the power supply’s user manual (attached in the appendix) is delivering the expected voltage. A small difference in the actual measured voltage to the specified voltage is expected. However, this difference should be extremely negligible.

Main Connector 20(+4) Pin P1 / 24 Testing

Pin No.	Expected Signal [V]	Measured Signal [V]	Pass/Fail	Pin No.	Signal[V]	Measured Signal [V]	Pass/Fail
1	+3.3			13	+3.3/+3.3VS		
2	+3.3			14	-12		
3	COM			15	COM		
4	+5			16	PS – ON		
5	COM			17	COM		
6	+5			18	COM		
7	COM			19	COM		
8	PW – OK			20	-	-	-
9	+5Vsb			21	+5		
10	+12V2			22	+5		
11	+12V2			23	+5		
12	+3.3V			24	COM		

Table 7-1 Main Connector Testing

The disk drivers are important to be tested because the design has designated PD and PC pin connectors to be used to power the PCB. Any possible other additional power requirements discovered when the project is being prototyped and tested will need to use the other pin connectors available. The group needs to ensure that all pin connections supplied by the power supply are working when and if the time comes for them to be used.

Disk Drivers (1 of 2)

PA			PB			PC		
Pin No.	Expected Signal [V]	**Measured Signal [V]**	**Pin No.**	Expected Signal [V]	**Measured Signal [V]**	**Pin No.**	Expected Signal [V]	**Measured Signal [V]**
1	+12V2		**1**	+12V2		**1**	+12V2	
2	COM		**2**	COM		**2**	COM	
3	COM		**3**	COM		**3**	COM	
4	+5		**4**	+5		**4**	+5	

Table 7-2 Disk Drivers Testing (1 of 2)

Disk Drivers (2 of 2)

PD			PE		
Pin No.	Expected Signal [V]	**Measured Signal [V]**	**Pin No.**	Expected Signal [V]	**Measured Signal [V]**
1	+12V2		**1**	+12V2	
2	COM		**2**	COM	
3	COM		**3**	COM	
4	+5		**4**	+5	

Table 7-3 Disk Drivers Testing (2 of 2)

The Serial ATA power connectors also need to be tested for proper functionality because they could also be used if any design modifications are made when prototyping and testing the chess board.

Serial ATA 15 Pin PF AND PG

PF			PG		
Pin No.	Expected Signal [V]	**Measured Signal [V]**	**Pin No.**	Expected Signal[V]	**Measured Signal [V]**
1	+12V2		1	+12V2	
2	COM		2	COM	
3	+5		3	+5	
4	COM		4	COM	
5	+3.3		5	+3.3	

Table 7-4 Sata Connectors Testing

Section 7.2.2: Electromagnet and Permanent Magnets

As the permanent magnets need no special testing, the testing shall only be done on the electromagnet. The electromagnet shall be powered to test the pull against the board and the rest of the pieces. It will also be tested to make sure it needs no extra power except the power provided from the microcontroller that will be controlling the state of the electromagnet. All observations will be included here after testing.

Electromagnet Testing

Step	**Description**	**Observation**
1	Using the power supply available in the senior design, power the DC electromagnet and test for initial proper function. Does the electromagnet require the same voltage called out in its specifications sheet?	
2	Once the electromagnet is connected to the lab's power supply test the magnets attraction to the permanent magnets in the chess pieces	
3	Test the electromagnet (on) with the pieces and the chess board	
4	Connect the electromagnet to the PCB and ensure the magnet does not need any more power than the microcontroller supplies	
5	Test the electromagnet for proper function with the commands sent from the microcontroller	

Table 7-5 Electromagnet Testing

Section 7.2.3: XY-Stage and Stepper Motors

For the testing of the XY-stage, the stage will be constructed and moved around its base to ensure free movement. It will also be tested to fit in the case to make sure the electromagnet is not far from the bottom side of the chess board. The stepper motors will also be tested individually to make sure they can freely turn. All observations will be included here after testing.

XY-Stage and Stepper Motor Testing

Step	**Description**	**Observation**
1	Once the stage is constructed test it for free unobstructed movement	
2	Ensure the XY-stage can fit comfortably inside the chess board casing	
3	Test the stage with the electromagnet to ensure the electromagnet is as close as possible to the underside of the chess board	
4	Test each stepper motor individually for unobstructed movement	

Table 7-6 XY-Stage and Stepper Motor Testing

Section 7.2.4: Motor Control PCB

The motor control board will be prototyped with a proto-board before designing and sending off for the actual PCB. The initial design will be built and tested to make sure it turns the stepper at least one full revolution. The purpose of the PCB testing will not to make sure that the XY-Stage goes to the right location, but that it simply moves the proper direction and to measure the distance it moves per full, half, quarter and eighth step. The test results will be listed here once the document is updated.

Motor Control PCB Testing

Step	**Description**	**Stage Measurements**	**General Observation**
1	Test the proto-board with the stepper motors to ensure it turns the motor through one full revolution and measure the distance the stage moves		
2	Test the proto board for half step movements and measure the distance the stage moves		
3	Test the proto board for quarter step movements and measure the distance the stage moves		
4	Test the proto board for eighth step movements and measure the distance the stage moves		

Table 7-7 Motor Control PCB Testing

Section 7.2.5: Voice Interpretation and AI Unit

The main testing of the voice interpretation and AI unit will mainly be the testing of software and communication between systems. It will most likely be the final piece of the project put into place and tested.

Section 7.2.6: LCD Screen

For the LCD screen testing, the Stellaris EKS will be hooked up to the AI unit (if they end up being separate pieces of hardware) and the display functions will be tested. Once the proper picture shows up on the screen, any further testing will be software related.

Section 7.3: Software Test Environment

In the research phase of the project, software was tested on personal computers to ensure proper functionality and to aid in understanding its operation for development purposes. Testing of the different chess engines has already been covered in great detail in previous sections. Final testing of the TSCP engine continued on a personal computer. This was done simply to measure the amount of RAM used and measure the average response time given similar hardware specifications. The voice recognition software will also be initially tested on a personal computer for proof of concept of the smaller dictionary. The amount of testing that can be done for the magnetic controller on a personal computer is limited, as the physical performance of the software's functions will need to be measured.

After the preliminary testing of each module is done on personal computers, it will be necessary to install the previously specified distribution of Debian Linux onto the hardware that the chess board will use, then run all of the different software components on top of that. Of course, the first test will be to ensure that each component can run properly in Linux, regardless of the accuracy of its operation in regards to our objectives.

In the final testing phase, the board will be fully built, and a user should be able to play a full game as described in the Objectives section. Once it is verified that users can play full games on the board, both against the computer and against each other, it will be assumed that the system is operating completely functionally.

Section 7.4: Software Specific Testing

Section 7.4.1: Operating Systems Testing

Each of the five operating systems considered for usage in the project will have to be tested. Until the computer system is put together, it's currently impossible to test each operating system on the computer system the project will be deploying. In addition, as the computer system will be using a solid state disk in place of a hard drive disk, installing and reinstalling multiple operating systems will create unnecessary read/writes on the drive and will cause unneeded wear and tear. As such each operating system will be tested using a virtual machine with settings comparable to that of the computer system that will be deployed according the project's specifications. In addition, to how each operating system performs, careful detail must be taken into consideration on compatibility with not only just the chess engine and voice library but in addition the hardware. In other words, the operating system must be capable of supporting the solid state disk and have the TRIM command available, otherwise it will decrease the performance and lifespan of the solid state disk.

Out of the five operating systems considered and research for running on the computer system, only Windows 7 and Arch Linux do not have a method to try out the operating system without installing it on the hardware. As a result, to save unneeded wear and tear on the solid state disk, those two operating systems will be tested using a virtual machine. However, as Windows 7 is a familiar operating system, it only leaves Arch Linux to be tested on a virtual machine. Debian, Fedora, and Ubuntu all have a live CD, which allows the operating system to be run on the hardware without the need to install it on the system. This will allow each operating system to be tested for compatibility with the hardware. Each operating system will be tested on first responsiveness. The chess engine will first be loaded, installed, or compiled if necessary on to the operating system. Then several games will be play to see how the chess engine runs on the operating system along with the hardware. Next the voice library will be complied and installed on the operating system. Then it will be tested by recording several phrases and sentences players playing the chess game will use. The results will be reviewed to determine how the system quickly responds to voice command and understands each spoken phrase.

The most compatible operating system with the smallest number of errors and adjustments will be used for the running on the computer hardware. If all of the operating systems are equally compatible, then the most familiar and easy to use operating system will be used.

Section 7.4.2: Speech Recognition Testing

As for testing the voice library and speech recognition software, the source code for CMU Sphinx will be downloaded and compiled. The advantage of using CMU Sphinx is that it works in all of the above operating systems mentioned. So testing Sphinx can occur alongside the operating systems. The first phrase is to download both the base code of Sphinx plus the source code for Pocketsphinx. Then the code is to be compiled using GNU GCC compilers. The GNU GCC compilers are universal among all the operating systems. In other words, it works on both Windows and Linux operating systems.

After compiling Sphinx, it will then tested by recording short phrases and commands that a player would give during gameplay. If the system is unable to understand certain phrases, then the system can be trained to understand phrases. In addition, there are two variants of CMU Sphinx. Both the main version and PocketSphinx will be tested as to see which one would provide better optimization of resources. Testing will be required to see which version understands and responds faster to voice commands. Accuracy is not taken into consideration for differentiation between the two versions of Sphinx, as both the main version and PocketSphinx uses the same base code and share the same language models. As a result accuracy will not be thoroughly tested, just simply the response time between each commands will be recorded on compared. However, even if accuracy is not thoroughly tested, the voice library will need to be able to understand and recognize commands given by the player. This speech test will also be done on each of the operating systems considered for the computer component of the project.

Section 7.4.3: Chess Engine Testing

The chess engine will be tested first to ensure it can operate smoothly and within the constraints imposed by the hardware. Because the board will not yet be fully assembled, it will be necessary to attach some visual display (a simple monitor) and play a full game of chess via the command line interface, which is supported by TSCP. Once it is clear that a user can play multiple games in one player and two player modes, with a smooth transition in between each, it will be assumed that the chess engine is functioning as intended.

Section 7.4.4: Magnetic Controller Testing

The final testing for the electromagnetic controller cannot begin until the board is nearly assembled. This is because each movement by the magnet will have to be precisely tracked and recorded. In general, to test the controller and its interface, the interface will be fed a series of chess coordinates (e.g. a2a3) and the output will be measured. The controller unit will need to use the precise coordinates in the X-Y plane of the board to output the series of instructions for the stepper motors. The number of pulses will be checked, as well as the order in which they are applied. Again, this output will be measured against what is expected, and then it will be determined if the software is functioning within its acceptable limits. Because the movements need to be rather precise, there will be little tolerance for errors in this module. This is opposition to the other modules, in which there can be virtually no error tolerance, as one simple mistake could completely derail a game.

Section 7.4.5 Software Integration and Final Testing

After each of the components is tested individually, it will be necessary to integrate them all by defining interfaces so that each component can communicate with the others. The plan to define these interfaces is detailed in earlier sections. First, the chess engine and the voice recognition unit will be tested in unison. The pass criteria for this test is whether or not two people can play an entire game of chess using only voice commands. Again, the tests will utilize the command line interface offered by TSCP. After this, the magnetic controller will be integrated to as great an extent as possible, and these three components will all be tested. If a user can perform many different moves (perhaps 20 or more) with all of the components operating as they should, the software will move on to the final testing phase.

Chapter 8 : Administrative Content

Section 8.1: Milestone Discussion

The milestones for this project included having the research and hardware design complete by the first week of December, having the hardware ordered by the middle of November and prototyped by the beginning of January. PCBs should be ordered by middle-end of January to ensure plenty of time to properly test and troubleshoot any problems. A schedule is provided in the tables below to ensure that the group meets its milestones.

Item	Group Member	Date Due
5 Pages Total Written Each	All	October 27, 2012
10 Pages Total Written Each	All	November 4, 2012
15 Pages Total Written Each	All	November 11, 2012
Parts for XY-Stage Purchased	Brittany Nottingham	November 18, 2012
Parts for Motor Control Board Purchased	Brittany Nottingham	November 18, 2012
20 Pages Total Written Each	All	November 18, 2012
Parts for AI Unit (Computer) Purchased	Brittany Nottingham	November 25, 2013
25 Pages Total Written Each	All	November 25, 2012
Motor Control Board Prototyped	Brittany Nottingham	December 2, 2012
30 Pages Total Written Each	All	December 2, 2012
Parts for Voice Unit Purchased	Brittany Nottingham	December 2, 2012
Senior Design Paper Due	All	December 6, 2012
PCB Preliminary Design for Motor Control Board Complete	Brittany Nottingham/ Haley Amason	December 15, 2012

Table 8-1 Fall 2012 Schedule for Milestone Completion

Item	Group Member	Date Due
Have all hardware (excluding the PCB board) ordered	Brittany Nottingham/ Haley Amason	January 1, 2013
Test the power supply(see section 7.2.1)	Haley Amason	January 7, 2013
Have the XY-stage constructed	Brittany Nottingham/ Haley Amason	January 7, 2013
Test the electromagnet for proper function (see section 7.2.2)	Haley Amason	January 11, 2013
Test the XY-Stage and Stepper Motors (see section 7.2.3)	Brittany Nottingham/ Haley Amason	January 14, 2013
Have the PCB Proto board built	Brittany Nottingham	January 10, 2013
Test the Proto board (see section 7.2.4)	Brittany Nottingham/ Haley Amason	January 14, 2013
Order PCB Board	Brittany Nottingham	January 21, 2013
OS Configued on board	Thong Tran	January 21, 2013
Chess Engine modified	Joshua Burbridge	February 1, 2013
Voice Activation installed	Thong Tran	March 15, 2013
Magnetic Controller Coded	Joshua Burbridge	March 15, 2013
All software interfaced	Thong Tran and Joshua Burbridge	March 29, 2013
Project Complete	All	April 1, 2013

Table 8-2 Spring 2013 Schedule for Milestone Completion

Section 8.2: Budget and Finance Discussion

Our project funding has been graciously donated by Soartech. Soartech builds intelligent systems for government and commercial applications. Their systems emulate human decision making in order to train the user more effectively [22].

With project funding only being $500 instead of the expected $1000 budget, some of the expense of the project will come out of the group member's pockets.

It has also been decided by the group that if the Children's Hospital in Orlando will accept the donation, that the project will be donated to the hospital to hopefully brighten the children's lives.

The budget will be broken down into sections by block diagram with a final tabulation after all the sections have been accounted for.

Section 8.2.1: Power System Cost

Item	Vendor	Part No.	Price	Qty.	Shipping	Total
Motherboard Power Supply	FSP GROUP USA	FSP220-60LE(80)	$45.99	1	$9.29	$55.28

Table 8-3 Power System Budget

Section 8.2.2: Electromagnet and Permanent Magnets Cost

Item	Vendor	Part No.	Price	Qty.	Shipping	Total
Small Disk Magnets	Skycraft		$0.40	32	0.00	$13.63
Round DC Electromagnet	APW Company	EMO75-12-22	$29.24	1	$9.58	$38.32
					Total	$51.62

Table 8-4 Magnetics Budget

Section 8.2.3: XY-Stage and Stepper Motors Cost

Item	Distributer	Part #	Price	Qty	Shipping	Total Price
1"x1" T-Slot	Amazon	1010	$ 12.99	2	$ 28.71	$ 54.69
1"x3" T-slot	Amazon	1030	$ 28.35	1	$ -	$ 28.35
1"x2" T-slot	Amazon	1020	$ 20.67	2	$ -	$ 41.34
3/4" Angle Bracket	McMaster-Carr	M 1556A24	$ 0.37	10	$ -	$ 3.70
1" Angle Bracket		M 1556A41	$ 0.74	8	$ -	$ 5.92
1.5" Angle Bracket	McMaster-Carr	M 1556A42	$ 0.95	4	$ -	$ 3.80
4" Plate	McMaster-Carr	M 1394A34	$ 0.85	4	$ -	$ 3.40
10-32 X 3/8" Stainless screws	McMaster-Carr	M 92949A263	$ 5.10	1	$ -	$ 5.10
10-32 x 1/2" Machine Screws	Home Depot	33121	$ 1.18	1	$ -	$ 1.18
Square Nuts 3/8" Width	McMaster-Carr	M 94785A411	$ 10.00	1	$ -	$ 10.00
#10 Flat Washers	Home Depot	32481	$ 1.18	1	$ -	$ 1.18
#10 Lock Washers	Home Depot	20211	$ 1.18	1	$ -	$ 1.18
1/4"-20 x3/4" cap screws	Home Depot	27991	$ 0.39	3	$ -	$ 1.18
1/4-20 Jam Nuts	Home Depot	27991	$ 0.39	3	$ -	$ 1.18
1/4" Lock Washers	Home Depot	20551	$ 1.18	1	$ -	$ 1.18
Split Lock Washer 1/4-20	McMaster-Carr	M 92146A029	$ 4.03	1	$ -	$ 4.03
Sliding Glass Door Rollers	Home Depot	622001	$ 4.94	12	$ -	$ 59.28
Aluminum Bare Angle	Online Metals	6063-T52	$ 7.41	1	$ -	$ 7.41
SS Ball Bearing ABEC-5	McMaster-Carr	M 57155K323	$ 5.70	12	$ -	$ 68.40
18-8 Stainless	McMaster-	M	$ 9.48	1	$ -	$ 9.48

Steel Bearing Shim	Carr	93574A513				
Pulley for XL-Series Timing Belt	McMaster-Carr	M 57105K21	$ 8.08	6	$ -	$ 48.48
Timing Belts	McMaster-Carr	M 6484K454	$ 14.99	3	$ -	$ 44.97
Black-Oxide Steel Shaft Collar	McMaster-Carr	M 9414T6	$ 0.61	12	$ -	$ 7.32
Steel Flat Mending Bracket	McMaster-Carr	M 1394A31	$ 0.54	2	$ -	$ 1.08
Stepper Motor 1	SparkFun	ROB-10847	$ 23.95	1	$ -	$ 23.95
Stepper Motor 2	SparkFun	ROB-09238	$ 14.95	1	$ -	$ 14.95
					Total Cost	**$ 452.73**

Table 8-5 XY-Stage and Stepper Motor Budget

Section 8.2.4: Microcontroller, Motor Control and PCB Cost

Item	Vendor	Part No.	Price	Qty.	Shipping	Total
PCB Fab			$33.00	1	$0.00	$33.00
Stepper Motor Controller	TI	TI DVR8818	$6.88	2	$3.95	$17.71
Microcontroller for Stepper	TI	TI MSP430 G2553	$2.79	2	$3.95	$9.53
					Total	$98.92

Table 8-6 Motor Control PCB Budget

Section 8.2.5: Voice Interpretation Unit

Item	Vendor	Part No.	Price	Qty.	Shipping	Total
Hard Drive		Vertex RS 60GB	$39.99	1	$5.99	$45.98
Motherboard and Processor	NewEgg	AsRock AD525PV3	$39.99	1	$3.99	$72.98
					Total	$119.96

Table 8-7 Voice Interpretation Unit Budget

Section 8.2.6: LCD Screen and Integration

Item	Vendor	Part No.	Price	Qty.	Shipping	Total
Stellaris M3 EKS	Texas Instruments	EKS-LM3S8962	$89.00	1	$0.00	$89.00

Table 8-8 LCD Screen and AI Unit Budget

Section 8.2.7: Miscellaneous Budget

Item	Vendor	Part No.	Price	Qty.	Shipping	Total
Solder	Amazon	AT-31504	$7.20	1	$ -	$7.20
Test Leads	SparkFun	PRT-11037	$2.95	1	$ -	$2.95
Heat Shrink	SparkFun	PRT-09353	$7.95	1	$ -	$7.95
Hook Up Wire	SparkFun	PRT-11367	$16.95	1	$ -	$16.95
					Total	$35.05

Section 8.2.8: Total Cost

Section	Cost
Power System Component	$ 55.28
Electromagnet and Permanent Magnet Component	$ 51.62
XY-Stage and Stepper Motors Component	$ 452. 73
Stepper Motor Control PCB Component	$ 98.92
LCD Screen and Microcontroller Component	$ 89.00
Voice Control Computer Component	$ 119.96
Total	$902.56

Table 8-9 Overall Budget and Total Project Cost

Chapter 9: Conclusions

Section 9.1: Final Thoughts

Thus far, this project has been both challenging and rewarding. The development team has gained ample experience with technical writing and documentation. It is more apparent now how much work goes into a project of this magnitude, and there are still challenges to overcome. However, we are confident that the design outlined in this document will lay a solid foundation for the prototype and final product while still being versatile enough to allow for changes in response to unforeseen technical issues. We eagerly look forward to building Magic Chess in the months to come.

Section 9.2: Future Directions

Throughout the project definition and research phase, many different sets of features were discussed. Although there was not enough time to implement all of these features – whether core or peripheral – there is certainly room for future growth and development of Magic Chess by other engineers. The possibilities for expansion are discussed in this section.

The first is a difficulty gradient as opposed to fixed difficulty levels. The strength of an opponent in chess is typically measured in units called Elos. An engine with a rating of 2000 or more Elo is considered pretty strong. In Magic Chess, the difficulty levels were created by running the chess engine at three fixed tiers of difficulty or three specific levels of Elo. For the serious chess player, this may not be satisfying enough, as the difficulty of the computer cannot grow in unison with the player's skill level. One additional feature that could be implemented in future iterations of the project is a difficulty gradient. This would be a sliding scale on which the user can graphically select their desired difficulty level. The range of operation may be from 0 Elo (the computer attempts random moves with no strategy) to perhaps 3000 Elo (very hard) and every level in between. This feature was not included in Magic Chess due to its ambition.

Continuing in the vein of instructive chess playing, it may also be helpful to the user to be able to access an "undo" function. This would work best in one player mode. A player can make a move, observe the consequences of the move, and then opt to undo the move and rethink their strategy. This function is obviously not allowed in real games of chess, but could serve as a valuable learning tool for children just learning how to play, which is an important target audience for Magic Chess.

A third function would be to develop some sort of graphical representation of the decision tree in real time. As it stands, the TSCP engine provides data describing the decision-making process it undergoes when analyzing the board. However, the most Magic Chess can do with this is print it out on the LCD screens. For most users of this board, who are inexperienced in chess programming, this data is meaningless. A graphical representation would be an opportunity to give the decision tree more significance, as a user would be able to observe how the computer maps out the possibilities for each move, scores them, and makes its decision based on the data. This would be a far better use of the generated data, provided it has enough appeal and aesthetic. The only drawback would be that this would not look very impressive on the tiny LCD screen, so the board would need to be connected to a separate monitor, sacrificing on portability.

Anyone reading this documentation is invited to investigate these possibilities with greater depth, keeping in mind that Magic Chess is derived from the open-source work of many different people. Any derivative of this work is ultimately a derivative of theirs, so credit must be properly given where it is due.

Works Cited

[1] Excalibur Electronics, "Phantom Force Electronic Chess 740D (Manual)," 2011. [Online]. Available: http://www.chesshouse.com/v/vspfiles/manuals/Phantom%20Force%20Electronic%20Chess%20740D.pdf. [Accessed 23 November 2012].

[2] Schneider Electric, "Integrated Linear Motion Systems," [Online]. Available: http://motion.schneider-electric.com/downloads/whitepapers/linear_motion.pdf. [Accessed 30 October 2012].

[3] "Introduction to Stepper Motors and Drives," Omega Engineering Technical Reference, [Online]. Available: http://www.omega.com/prodinfo/stepper_motors.html. [Accessed 30 October 2012].

[4] G. Lazaridis, "How DC Motors and Made and How They Work," PCBheaven, [Online]. Available: http://pcbheaven.com/wikipages/How_DC_Motors_Work/. [Accessed 30 October 2012].

[5] STIPAF, "Advantage of D.C. Motors," DC Motors, [Online]. Available: http://www.dcmotors.eu/products/advantages-of-the-motor-dc.html. [Accessed 30 October 2012].

[6] Acroname, "Servo Motors," Acroname, 1 May 2008. [Online]. Available: http://www.acroname.com/robotics/info/articles/servo/servo.html#e1. [Accessed 30 October 2012].

[7] National Instruments, "Servo Motors," [Online]. Available: http://www.ni.com/white-paper/3656/en#toc5. [Accessed 30 October 2012].

[8] D. W. Jones, "Control of Stepping Motors," University of Iowa, 1998. [Online]. Available: http://homepage.cs.uiowa.edu/~jones/step/. [Accessed 30 October 2012].

[9] "Circuits, Code and Construction," [Online]. Available: http://www.tigoe.com/pcomp/code/circuits/motors/controlling-dc-motors/. [Accessed 30 October 2012].

[1 Amazing Magnets, "Amazing Magnets," [Online]. Available:

0] http://www.amazingmagnets.com/default.aspx. [Accessed 25 November 2012].

[1 C. S, "Instructibles.com," 11 March 2012. [Online]. Available:
1] http://www.instructables.com/id/Internet-Arduino-Controlled-T-Slot-XY-Table/. [Accessed 3 December 2012].

[1 Wantai Motor, "42BYGH Specifications," Wantai Motor, [Online]. Available:
2] http://www.wantmotor.com/ProductsView.asp?id=160&pid=75&sid=80. [Accessed 29 November 2012].

[1 Wantai Motor, "39BYGL Specifications," Wantai Motor, [Online]. Available:
3] http://www.wantmotor.com/ProductsView.asp?id=171&pid=75&sid=80. [Accessed 29 November 2012].

[1 Mercury Motor, "ST-42BY Hybrid Step Motor," [Online]. Available:
4] http://www.mercurymotion.com/products/zlbjmd/st-42by.pdf. [Accessed 29 November 2012].

[1 Wantai Motor, "42BYGHM Specifications," Wantai Motor, [Online]. Available:
5] http://www.wantmotor.com/ProductsView.asp?id=157&pid=75&sid=80. [Accessed 29 November 2012].

[1 Texas Instruments, "Stellaris LM3S8962 Evaluation Board," 09 February
6] 2010. [Online]. Available: http://www.ti.com/lit/ug/spmu032b/spmu032b.pdf. [Accessed 30 November 2012].

[1 Texas Instruments, "MSP430G2553," August 2012. [Online]. Available:
7] http://www.ti.com/lit/ds/symlink/msp430g2553.pdf. [Accessed 29 November 2012].

[1 Atmel Corperation, "ATmega 328," [Online]. Available:
8] http://www.atmel.com/Images/doc8271.pdf. [Accessed 29 November 2012].

[1 Allegro Microsystems, "A3967: Microstepping Driver with Translator," [Online].
9] Available: http://www.allegromicro.com/Products/Motor-Driver-And-Interface-ICs/Bipolar-Stepper-Motor-Drivers/A3967.aspx. [Accessed 29 November 2012].

[2 Allegro Microsystems, "A4983 DMOS Microstepping Driver with Translator,"
0] [Online]. Available: http://www.allegromicro.com/Products/Motor-Driver-And-Interface-ICs/Bipolar-Stepper-Motor-Drivers/A4983.aspx. [Accessed 29

November 2012].

[21] Texas Instruments, "DRV8818 Stepper Driver," 2 October 2012. [Online]. Available: http://www.ti.com/lit/ds/symlink/drv8818.pdf. [Accessed 29 November 2012].

[22] "About - History," 2012. [Online]. Available: http://www.soartech.com/about/history/. [Accessed 28 October 2012].

[23] J. Haley, P. Le, B. Reeves and J. Jose, "KnightSweeper 4200," [Online]. Available: http://eecs.ucf.edu/seniordesign/fa2011sp2012/g09/Documents/SD1/GROUP9PAPER_Final.pdf. [Accessed 28 October 2012].

[24] C. S, "Internet Arduino Controlled T-Slot XY Table," Instructables, [Online]. Available: http://www.instructables.com/id/Internet-Arduino-Controlled-T-Slot-XY-Table/#step1. [Accessed 30 October 2012].

[25] Microsoft, "Windows 7 System Requirements," [Online]. Available: http://windows.microsoft.com/en-us/windows7/products/system-requirements. [Accessed 22 November 2012].

[26] Fedora Project, "Objectives," [Online]. Available: http://fedoraproject.org/wiki/Objectives. [Accessed 19 11 2012].

[27] Fedora Project, "Fedora Project - Download Fedora and try it," [Online]. Available: http://fedoraproject.org/en/get-fedora. [Accessed 19 November 2012].

[28] Debian, "Meeting Minimum Hardware Requirements," [Online]. Available: http://www.debian.org/releases/stable/i386/ch03s04.html.en. [Accessed 18 November 2012].

[29] Debian, "Supported Hardware," [Online]. Available: http://www.debian.org/releases/stable/i386/ch02s01.html.en. [Accessed 18 November 2012].

[30] Arch Linux, "Beginer's Guide - ArchWiki," [Online]. Available: https://wiki.archlinux.org/index.php/Beginners_Guide#Step_2:_Boot_Arch_Linux_Installer. [Accessed 23 November 2012].

[3
1] Canonical Ltd., "Get Xubuntu," [Online]. Available: http://xubuntu.org/getxubuntu/. [Accessed 28 November 2012].

[3
2] Canonical Ltd., "Meeting Minimum Hardware Requirements," [Online]. Available: https://help.ubuntu.com/12.10/installation-guide/powerpc/minimum-hardware-reqts.html. [Accessed 17 November 2012].

[3
3] Linux Mint team, "The Linux Mint Blog," [Online]. Available: http://blog.linuxmint.com/?p=2216. [Accessed 30 November 2012].

[3
4] Bunting Magnetics Co, "buymagnets.com," 2012. [Online]. Available: http://buymagnets.com/product/246/Electromagnets-Round/. [Accessed 1 12 2012].

[3
5] Wantai Motor, "39BYGL Specifications," Wantai Motor, [Online]. Available: http://www.wantmotor.com/ProductsView.asp?id=171&pid=75&sid=80. [Accessed 29 November 2012].

[3
6] Wantai Motor, "42BYGHM Specifications," Wantai Motor, [Online]. Available: http://www.wantmotor.com/ProductsView.asp?id=157&pid=75&sid=80. [Accessed 29 November 2012].

[3
7] Atmel Corperation, "ATmega 328," [Online]. Available: http://www.atmel.com/Images/doc8271.pdf. [Accessed 29 November 2012].

[3
8] Allegro Microsystems, "A3967: Microstepping Driver with Translator," [Online]. Available: http://www.allegromicro.com/Products/Motor-Driver-And-Interface-ICs/Bipolar-Stepper-Motor-Drivers/A3967.aspx. [Accessed 29 November 2012].

[3
9] Allegro Microsystems, "A4983 DMOS Microstepping Driver with Translator," [Online]. Available: http://www.allegromicro.com/Products/Motor-Driver-And-Interface-ICs/Bipolar-Stepper-Motor-Drivers/A4983.aspx. [Accessed 29 November 2012].

[4 Texas Instruments, "DRV8818 Stepper Driver," 2 October 2012. [Online]. Available: http://www.ti.com/lit/ds/symlink/drv8818.pdf. [Accessed 29

[40] November 2012].

[41] ASRock Inc, "AD525PV3," November 2010. [Online]. Available: http://www.asrock.com/mb/Intel/AD525PV3/. [Accessed 15 10 2012].

[42] Intel, "Intel Atom processor D525," Intel, [Online]. Available: http://ark.intel.com/products/49490. [Accessed 15 10 2012].

[43] ASRock, "A75M-ITX," ASRock, 2012. [Online]. Available: http://www.newegg.com/Product/Product.aspx?Item=N82E16813157273&Tpk=asrock%20a75m-itx. [Accessed 30 11 2012].

[44] Newegg, "AMD A4-3400 Llano 2.7GHz Socket FM1 65W Dual-Core Desktop APU (CPU + GPU) with DirectX 11 Graphic AMD Radeon HD 6410D AD3400OJHXBOX," AMD, 2012. [Online]. Available: http://www.newegg.com/Product/Product.aspx?Item=N82E16819106014&Tpk=ad3400ojhxbox. [Accessed 30 11 2012].

[45] Newegg, "Kingston HyperX Blu Red Series 8GB (2 x 4GB) 240-Pin DDR3 SDRAM DDR3 1600 Desktop Memory Model KHX16C9B1RK2/8," Kingston Hyper, 2012. [Online]. Available: http://www.newegg.com/Product/Product.aspx?Item=N82E16820104344. [Accessed 30 11 2012].

[46] ASRock, "ASRock E350M1 AMD E-350 APU (1.6GHz, Dual-Core) AMD A50M Hudson M1 Mini ITX Motherboard/CPU Combo," ASRock, 2012. [Online]. Available: http://www.newegg.com/Product/Product.aspx?Item=N82E16813157228. [Accessed 30 11 2012].

[47] Newegg, "rucial 4GB (2 x 2GB) 240-Pin DDR3 SDRAM DDR3 1066 (PC3 8500) Dual Channel Kit Desktop Memory Model CT2KIT25664BA1067," Crucial, 2012. [Online]. Available: http://www.newegg.com/Product/Product.aspx?Item=N82E16820148150. [Accessed 30 11 2012].

[48] "About - History," 2012. [Online]. Available: http://www.soartech.com/about/history/. [Accessed 28 October 2012].

[49] J. Haley, P. Le, B. Reeves and J. Jose, "KnightSweeper 4200," [Online]. Available: http://eecs.ucf.edu/seniordesign/fa2011sp2012/g09/Documents/SD1/GROUP

9PAPER_Final.pdf. [Accessed 28 October 2012].

[50] C. S, "Internet Arduino Controlled T-Slot XY Table," Instructables, [Online]. Available: http://www.instructables.com/id/Internet-Arduino-Controlled-T-Slot-XY-Table/#step1. [Accessed 30 October 2012].

[51] Microsoft, "Windows 7 System Requirements," [Online]. Available: http://windows.microsoft.com/en-us/windows7/products/system-requirements. [Accessed 22 November 2012].

Appendix A: Documented Permission

Email permission from APW company to use their product picture of the EM075-12-222 DC Electromagnet in the paper.

Haley Amason <haleyamason@gmail.com> Wed, Dec 5, 2012 at 10:21 PM

To: info@apwcompany.com

To Whom it May Concern,
I would like to request permission from the APW Company to reprint their EM075-12-222 product picture of the electromagnet found at this web address http://catalog.apwcompany.com/image?&cid=3760&plpver=1001&prodid=1007&itemid=1034. I am using the electromagnet for my Senior Design Project at the University of Central Florida and would like to include the picture in my group's design paper. Please let me know if permission is possible from APW and how I may obtain it. Thank you for your time.

Sincerely,
Haley Amason
UCF Student
Major: EE

Jason Kellenberger <jason@apwcompany.com> Wed, Dec 5, 2012 at 10:30 PM

To: Haley Amason <haleyamason@gmail.com>

Hi Haley,

Thank you for contacting us concerning the use our product image on our website. We are honored you would be willing to include it in your project. As long as you provide credit to APW Company in your paper, you may use the image. I would love to get a copy of your paper when you complete it so we can share it with our employees. The love to see the products they make being used in applications and in projects. Thanks again and I wish you the best of luck with your Senior Design Project.

Best regards,

Jason Kellenberger

President & Owner

Email Permission from FSP Group USA to use their mechanical drawing picture of the FSP220-60LE(80) power supply

RE: Permission to Reuse FSP220-60LE Power Supply Layout

1 message

Ana Brady <ana.brady@fspgroupusa.com> Wed, Dec 5, 2012 at 8:33 PM

Reply-To: ana.brady@fspgroupusa.com
To: haleyamason@gmail.com

Dear Haley,

Thank you for contacting FSP Group USA. If it's the mechanical drawing you are planning on using, you are cleared to do so and there is no need for additional permission.

Let me know if you have any questions.

Sincerely,

Ana Brady
FSP Group USA
Tel. 1.909.606.0960 Ext. 1222
e-mail: ana.brady@fspgroupusa.com

---------- Original Message ----------
From: Haley Amason <haleyamason@gmail.com>
To: info@fspgroupusa.com
Date: December 5, 2012 at 7:20 PM
Subject: Permission to Reuse FSP220-60LE Power Supply Layout

To Whom it May Concern,

I would like to request permission by the FSP Group/ FSP Technology Inc. to reuse their FSP220-60LE diagram of the power supply found in their Mechanical Drawing paper released on November 9th of 2011 on page 2. I am using the FSP220-60LE for my Senior Design Project at the University of Central Florida and would like to include the layout in my group's design paper. Please let me know if permission is possible from FSP Group and how I may obtain it. Thank you for your time.

Sincerely,
Haley Amason
UCF Student
Major: EE

Permission from ASRock Inc. to use their motherboard layout drawing from their AD525PV3 User Manual.

Permission to Reuse AD525PV3 Motherboard Layout
2 messages

Haley Amason <haleyamason@gmail.com> Sun, Dec 2, 2012 at 7:02 PM

To: info@asrock.com.tw

To Whom it May Concern,
I would like to request permission by ASRock Inc. to reuse their AD525PV3 motherboard layout found in their "AD525PV3/AD425PV3" User Manual Version 1.0 published in November of 2010 on page 10. I am using the AD525PV3 for my Senior Design Project at the University of Central Florida and would like to include the layout in my group's design paper. Please let me know if permission is possible from ASRock and how I may obtain it. Thank you for your time.

Sincerely,
Haley Amason
UCF Student
Major: EE

Info <Info@asrock.com.tw> Sun, Dec 2, 2012 at 9:08 PM
To: Haley Amason <haleyamason@gmail.com>

Dear Haley,

Thanks for showing interests in ASRock motherboard. Per your request, please find AD525PV3 layout design in the attachment.

Hope it helps!

Best Regards,

ASRock Inc.

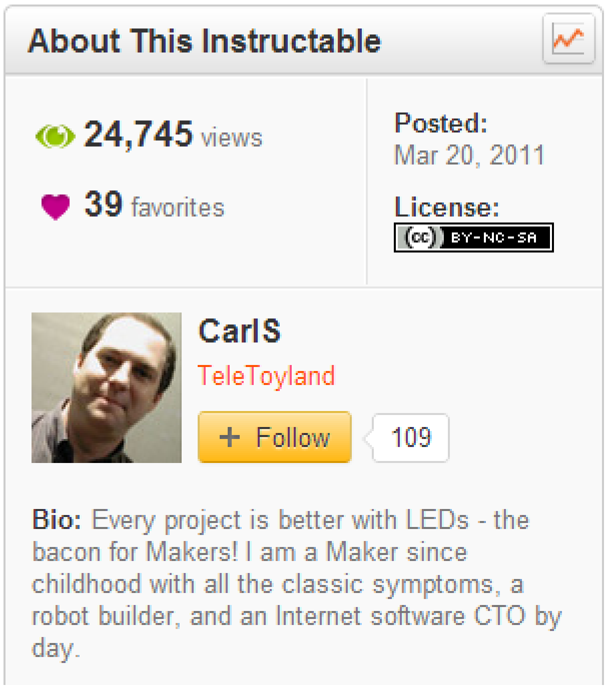
寄件者: Haley Amason [haleyamason@gmail.com]
寄件日期: 2012年12月3日 上午 08:02
收件者: Info
主旨: Permission to Reuse AD525PV3 Motherboard Layout

[Quoted text hidden]

-Texas Instruments Consent

TI grants permission to download, reproduce, display and distribute the Materials posted on this site solely for informational and non-commercial or personal use, provided that you do not modify such Materials and provided further that you retain all copyright and proprietary notices as they appear in such Materials. TI further grants to educational institutions (specifically K-12, universities and community colleges) permission to download, reproduce, display and distribute the Materials posted on this site solely for use in the classroom, provided that such institutions identify TI as the source of the Materials and include the following credit line: "Courtesy of Texas Instruments". Unauthorized use of any of these Materials is expressly prohibited by law, and may result in civil and criminal penalties. This permission terminates if you breach any of these terms and conditions. Upon termination you agree to destroy any Materials downloaded from this site.

Use for XY-Stage components:


About This Instructable

24,745 views

39 favorites

Posted:
Mar 20, 2011

License:

CarlS
TeleToyland

+ Follow 109

Bio: Every project is better with LEDs - the bacon for Makers! I am a Maker since childhood with all the classic symptoms, a robot builder, and an Internet software CTO by day.