Design Report
05/11/16

Team 08
Alexis Bonnema, Nicholas Bramer, Paul Brouwer,
Derek De Young, Curtis Kortman

Team Advisor
Professor Mark Michmerhuizen

Engineering 340 – Spring 2016
Executive Summary

Chess is an ancient game that has not seen significant change for many years. While there are some chess variations dating back to the 13th century, the version played today was largely determined near the end of the 15th century (Calvo, 1998). This project shows how modern technology can meld with a historic game to enhance the interaction and experience with the game. The group designed and built a prototype of an automated chessboard.

The design of the chessboard could have taken on many modern influences like bright displays, lots of graphics, or motion and high-speed action. However, the group decided to respect some of the humbler origins of the game, and created an interface that adds vast capabilities to the game, without altering the feel or environment associated with chess. The selected design included several goals. The first goal was to create a physical chess game where one player can play a computer. Additionally, the board would provide capability for a remote chess player with a cell phone app to play the person sitting with the board. The second main goal was that the chess pieces move without a visible movement mechanism to preserve the traditional look and feel of chess. To achieve a traditional character, the chessboard would include a minimalistic interface to start the computerized games and time the turns. Furthermore, the chessboard would be made of wood. Finally, to improve accessibility, a stretch goal of the project was to add voice control to the chess table.

The team working on the chessboard consisted of five Senior Electrical and Computer Engineering majors from Calvin College. The team members are Alexis Bonnema, Nick Bramer, Paul Brouwer, Derek De Young, and Curtis Kortman. Calvin College professor Mark Michmerhuizen served as the group’s mentor, giving advice and monitoring the group’s progress. Gentex employee Eric Walstra also generously offered his services as an industrial consultant giving insight into project management.

The group designed and built the prototype by May 7, 2016. The group had striven to maintain a budget set forth by the engineering department of $500, but went over budget. Treasurer Nick Bramer judiciously approved purchases to verify features could be sufficiently implemented, while minimizing fiscal waste. To reduce strain on the budget, the team contacted the sales manager, Jim Kortman, at Donald Engineering Inc. who obtained many of the parts for the team at discount.

In addition to working on design and construction, the project included presentations, documentation, reports, and the maintenance of a website. The team has tracked project progress since its inception in September of 2015. Dictated in this report is the detailed explanation of decisions and progress from start to finish. The team successfully achieved their goals in creating an automated chess board on time for senior design night. The team demonstrated different play types: voice control, playing via app, playing over the board, and playing against a computer. All systems worked without error.
Table of Contents

1 INTRODUCTION ............................................................................................................................... 1
  1.1 PROJECT BACKGROUND ............................................................................................................... 1
  1.2 TEAM DESCRIPTION .................................................................................................................... 1
    1.2.1 Alexis Bonnema .................................................................................................................. 1
    1.2.2 Nicholas Bramer ................................................................................................................. 2
    1.2.3 Paul Brouwer ....................................................................................................................... 2
    1.2.4 Derek De Young ................................................................................................................ 2
    1.2.5 Curtis Kortman ................................................................................................................... 2
  1.3 INSPIRATION ............................................................................................................................... 3

2 PROJECT MANAGEMENT ................................................................................................................ 4
  2.1 TEAM ORGANIZATION ............................................................................................................... 4
  2.2 SOFTWARE TEAM ..................................................................................................................... 4
  2.3 BOARD AND PIECES TEAM ...................................................................................................... 4
  2.4 XY-TABLE TEAM .................................................................................................................... 4
  2.5 SCHEDULE AND WORK HOURS ........................................................................................... 4
  2.6 BUDGET ..................................................................................................................................... 5
  2.7 METHOD OF APPROACH ......................................................................................................... 5

3 RESEARCH ....................................................................................................................................... 7
  3.1 MARKET RESEARCH ................................................................................................................ 7
  3.2 RELATED TECHNOLOGIES ....................................................................................................... 7
    3.2.1 Reed Switches .................................................................................................................... 7
    3.2.2 Chess Engines ................................................................................................................... 7
    3.2.3 Voice Control .................................................................................................................... 8

4 SYSTEM DESIGN ........................................................................................................................... 9
  4.1 SYSTEM REQUIREMENTS .......................................................................................................... 9
    4.1.1 Software Requirements ...................................................................................................... 9
    4.1.2 Hardware Requirements .................................................................................................. 9
    4.1.3 Performance Requirements ............................................................................................. 9
  4.2 SYSTEM ARCHITECTURE ......................................................................................................... 9
  4.3 ELECTRONIC ARCHITECTURE ............................................................................................... 11
    4.3.1 Raspberry Pi 2.0 ................................................................................................................. 11
    4.3.2 Motor System ..................................................................................................................... 12
    4.3.3 Electromagnet .................................................................................................................... 12
    4.3.4 Reed Switch Array ............................................................................................................ 12
    4.3.5 Microphones ..................................................................................................................... 12
    4.3.6 Power Supplies .................................................................................................................. 12
    4.3.7 Capacitive Filter ............................................................................................................... 12
    4.3.8 Power Filter ...................................................................................................................... 12
    4.3.9 LED Matrix ....................................................................................................................... 13
    4.3.10 Buttons ............................................................................................................................ 13
  4.4 THE CHESS TABLE .................................................................................................................. 13
    4.4.1 Frame Design ...................................................................................................................... 13
    4.4.2 Table Design ..................................................................................................................... 14
    4.4.3 Playing Surface .................................................................................................................. 15
    4.4.4 Button Panel ...................................................................................................................... 15
4.4.5  LED Matrix ........................................................................................................... 15
4.5  SOFTWARE ARCHITECTURE ....................................................................................... 15
  4.5.1  Chess Game ........................................................................................................... 16
  4.5.2  Chess Engine ......................................................................................................... 16
  4.5.3  Hardware Interfacing Classes .................................................................................. 17
  4.5.4  Chess Table ........................................................................................................... 17
  4.5.5  Chess Board ......................................................................................................... 17
  4.5.6  iOS Application/Web Server .................................................................................... 17
4.6  DESIGN ALTERNATIVES AND SELECTION ................................................................. 18
  4.6.1  Chess-piece Movement ......................................................................................... 18
  4.6.2  Microcontroller ...................................................................................................... 18
  4.6.3  Board ...................................................................................................................... 20
  4.6.4  Piece Tracking ....................................................................................................... 20
  4.6.5  Table ....................................................................................................................... 21
4.7  PRE-INTEGRATION TESTING ...................................................................................... 23
  4.7.1  Surface .................................................................................................................. 23
  4.7.2  Reed Switch Sensor Board ...................................................................................... 26
  4.7.3  Voice Control ......................................................................................................... 29
  4.7.4  Software ................................................................................................................ 30
  4.7.5  LED Matrix ............................................................................................................ 30
4.8  INTEGRATION AND TESTING .................................................................................... 32
  4.8.1  Reed Switches ........................................................................................................ 32
  4.8.2  Electromagnet ........................................................................................................ 32
  4.8.3  XY-Table Motors ................................................................................................... 32
  4.8.4  Software ................................................................................................................ 34
  4.8.5  Wire Management ................................................................................................. 34
  4.8.6  LED Matrix ............................................................................................................ 37
  4.8.7  Voice Control ......................................................................................................... 41
5  BUSINESS PLAN ........................................................................................................... 42
  5.1  MISSION FOR THE COMPANY .................................................................................. 42
  5.2  POTENTIAL CUSTOMERS & TARGET MARKET ...................................................... 42
  5.3  COMPETITIVE STRATEGY ....................................................................................... 42
  5.4  MARKET SIZES & TRENDS ..................................................................................... 42
  5.5  BUSINESS STRATEGY & DISTRIBUTION ............................................................... 42
  5.6  SWOT ANALYSIS ...................................................................................................... 42
    5.6.1  Strengths ............................................................................................................... 42
    5.6.2  Weaknesses .......................................................................................................... 42
    5.6.3  External Opportunities ......................................................................................... 43
    5.6.4  External Threats .................................................................................................... 43
  5.7  UNIQUENESS & EXISTING/POTENTIAL COMPETITORS ........................................ 43
6  CONCLUSION ............................................................................................................... 44
7  ACKNOWLEDGEMENTS .................................................................................................. 45
  7.1  PEOPLE ..................................................................................................................... 45
    7.1.1  Jim Kortman - Donald Engineering ..................................................................... 45
    7.1.2  Phil Jasperse .......................................................................................................... 45
    7.1.3  Professor Yoon Kim .............................................................................................. 45
    7.1.4  Professor Mark Michmerhuizen .......................................................................... 45
Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Senior Design Team 08</td>
<td>1</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Remaining Balance after each Purchase</td>
<td>5</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Reed Switch Sensitivity Lobes</td>
<td>7</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Full System Overview</td>
<td>10</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Electronics Hardware Block Diagram</td>
<td>11</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Power Filter Schematic</td>
<td>13</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Steel Frame, Support Structure, and XY-Table</td>
<td>14</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Table Design</td>
<td>15</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Software Hierarchy</td>
<td>16</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Proof of Concept iOS App</td>
<td>17</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Support for first test surface</td>
<td>23</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Bottom Side of First Test Surface</td>
<td>23</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Second test surface made of half inch Polycarbonate</td>
<td>24</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Taped Up Wooden Chess Surface</td>
<td>25</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Wooden Chess Surface (Without Engravings)</td>
<td>25</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Wooden Chess Surface (With Engravings)</td>
<td>26</td>
</tr>
<tr>
<td>Figure 17</td>
<td>CNC Making Engravings in Plastic</td>
<td>26</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Completed Reed Board Trenches</td>
<td>27</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Reed Switch Array Test Circuit Diagram</td>
<td>28</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Completed Reed Board</td>
<td>29</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Close Up of Reed Switch and Diode</td>
<td>29</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Loop-based step control</td>
<td>33</td>
</tr>
<tr>
<td>Figure 23</td>
<td>PWM with Interrupts Code Architecture</td>
<td>33</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Unorganized Prototype Board</td>
<td>35</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Wiring Schematic</td>
<td>36</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Final Wiring in Table</td>
<td>36</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Noise that the LED matrix is seeing</td>
<td>37</td>
</tr>
<tr>
<td>Figure 28</td>
<td>Analog Filter</td>
<td>38</td>
</tr>
<tr>
<td>Figure 29</td>
<td>Ribbon Cable and Connector</td>
<td>38</td>
</tr>
<tr>
<td>Figure 30</td>
<td>Bottom Side of Ribbon Connector</td>
<td>39</td>
</tr>
<tr>
<td>Figure 31</td>
<td>Menu Options for LED Matrix</td>
<td>39</td>
</tr>
<tr>
<td>Figure 32</td>
<td>Error Handling for LED Matrix</td>
<td>40</td>
</tr>
<tr>
<td>Figure 33</td>
<td>Movement Display for LED Matrix</td>
<td>40</td>
</tr>
<tr>
<td>Figure 34</td>
<td>Capture Display for LED Matrix</td>
<td>41</td>
</tr>
<tr>
<td>Figure 35</td>
<td>Time Analysis</td>
<td>48</td>
</tr>
</tbody>
</table>
Table of Tables

Table 1 - Listing of Donated Items and Values ........................................................................................................4
Table 2 - Decision Matrix for Project Management Software ..................................................................................5
Table 3 - LED Matrix Serial Commands ..................................................................................................................13
Table 4 - Decision Matrix for the Selection of Piece Movement Mechanism ............................................................18
Table 5 - Decision Matrix for the Selection of the Microcontroller .........................................................................19
Table 6 - Decision Matrix for Piece Tracking ........................................................................................................20
Table 7 - Decision Matrix for Choosing a Chess Table Design ................................................................................22
Table 8 - Parallelized Motor Tests ..........................................................................................................................34
Table 9 - Itemized List of Purchases .........................................................................................................................49
1 INTRODUCTION

1.1 Project Background

The Senior Design Project is part of Engineering Courses 339 and 340: the capstone courses in the Engineering program. For the design project, a team of three to five members work together to develop a solution to a design problem. Students choose their own design project. Team 08 chose to design and prototype an automatic chessboard with artificial intelligence (AI) capabilities.

1.2 Team Description

The name chosen for this team is The Knight. The Knight team consists of five senior engineering students of the electrical & computer concentration. The team based the idea of an automated chessboard on the combination of their desires to work with a physical system and with a system that was internet connected. The members of the team are pictured below in Figure 1.

![Figure 1 - Senior Design Team 08](image)

From Left to Right: Derek De Young, Alexis Bonnema, Curtis Kortman, Paul Brouwer, Nick Bramer

1.2.1 Alexis Bonnema

Alexis Bonnema is a senior studying Electrical/ Computer Engineering at Calvin College and pursuing a minor in Mathematics. She grew up in South Holland, Illinois, a south suburb of Chicago. This past summer, Alexis worked at Calvin College in a research position designing a current- and voltage-
adjustable power supply for an array of high-power LEDs. This included opportunities for working with printed circuit board design and assembly as well as microcontroller software.

Alexis is currently on the leadership team for the college's chess club, and also enjoys ballroom dancing, writing, crocheting, and playing board and video games with friends.

1.2.2 Nicholas Bramer

Nick Bramer is a senior at Calvin College studying Electrical/Computer Engineering as well as pursuing a Computer Science Minor. During the summer of 2014, he worked at Gentex on the production line in the bending area, which led into a summer internship in 2015. During the internship, he had the opportunity to program ABB robots, write VB and jscript code, work with Citect and Iconics graphics, wire panels, and do ladder logic programming with PLCs. Nick's future plans are to work at Gentex as a technician in the glass coating department.

In his spare time, Nick spends a great deal of his time playing soccer, as well as furthering his knowledge of software languages, and different operating systems.

1.2.3 Paul Brouwer

Paul Brouwer is a senior Electrical & Computer Engineering major at Calvin College, and is also pursuing minors in Mathematics and Computer Science. During the summer of 2014, Paul worked at Steelcase, Inc's Innovation Management Office, researching how to apply the Internet of Things to a manufacturing setting. In the summer of 2015, he researched with Dr. Larry Pileggi at Carnegie Mellon University, developing a non-volatile D-latch using magnetic tunnel junctions. After graduating, Paul plans to attend Carnegie Mellon University to pursue a Ph.D. in Computer Engineering.

Outside of the classroom, Paul has taken piano lessons for 16 years and has continued to take them every semester at Calvin. He has been involved with the Calvin Gospel Choir and Oratorio Society, and his hobbies also include playing tennis, working on minor coding projects, and building jigsaw puzzles.

1.2.4 Derek De Young

Derek De Young is a senior at Calvin College studying Electrical/Computer Engineering. During the summer of 2014 and 2015, he had the opportunity to intern at Federal-Mogul Powertrain. There he had the opportunity to work on and optimize multiple databases dealing with the lifecycle of their products, as well as integrating their design software into an existing database. He also is passionate about web and mobile app development. After graduating, Derek will be working at Epic Systems in Verona WI as a Software Developer.

In his spare time, he enjoys playing intramural sports such as basketball and volleyball, as well as researching new and upcoming technologies and learning more about web/app development.

1.2.5 Curtis Kortman

Curtis Kortman is a senior at Calvin College studying Electrical/Computer Engineering. During the past summers, he had the opportunity to work at Johnson Controls (now Visteon), and perform research at Calvin College regarding building energy efficiency and the Internet of Things. He has worked on several projects including designing an audio circuit for automotive instrument clusters, designing an automated test rig for testing automotive batteries, and implementing a sensor network at Calvin College to monitor energy usage.
During his spare time, Curtis enjoys being outside and active while fishing, hunting, biking, or hiking. He also enjoys working with his hands on different projects around the house, as well as transportation like mopeds and bikes. Following graduation, he will be working at Gentex in Zeeland as a computer vision development technician.

1.3 Inspiration

The idea for The Knight came from a game called Wizard's Chess in *Harry Potter and the Sorcerer's Stone*, in which Harry and his friends move chess pieces by speaking their moves. Being able to also move the pieces by hand or by using a mobile app were added to flesh out the project. Later, the team stumbled upon the stories of physically disabled people who play chess and other mental games over the internet, but are unable to play the same games over the table without assistance. The World Chess Federation (FIDE) appointed an official "FIDE Advisor Chess for Disabled" in 2011, and has held two World Chess Championships for Disabled, which include sections for physically disabled, deaf, and blind or visually impaired (braille chess) players [1] [2]. Although the smooth, even surface of The Knight makes it unsuitable for braille chess, it could be useful for physically disabled players. FIDE specifies that for events, "If a competitor cannot press his own clock or move his own pieces, an assistant shall be available unless the opponent is willing to do so" [3]. The Knight could potentially allow these players to play over-the-board chess without a human assistant.

The Knight was formerly known under the name Prophylaxis, a chess term meaning "to thwart an opponent's plan before it even begins." The name was changed because it was being confused with preventative medical and dental care. The new name, The Knight, refers both to the most distinctive chess piece on the board and to the mascot of Calvin College.
2 PROJECT MANAGEMENT

2.1 Team Organization
With a team of five members, it was imperative that the team communicated early who was going to be working on what part of the project. To accomplish this the team divided into three main sections.

2.2 Software Team
The software team was composed of Derek De Young and Nicholas Bramer. It was their duty to get the upper level code working. This includes the tasks of interfacing with the chess engine and sending commands to the lower level classes to get the current state of the board, move the motors, and turn on the magnet. Along with the physical board’s code, the software team has also worked on an iOS app and a web server so players can play against the others from anywhere in the world on a physical chess board.

2.3 Board and Pieces Team
Paul Brouwer and Alexis Bonnema worked on interfacing with the physical board and magnets. They worked on testing reed switches, magnets, and electromagnets, and researched how to correctly wire and use an array of reed switches to determine the location of pieces on the board. In addition, the team determined the necessary parameters for building a chessboard and pieces with embedded magnetics.

2.4 XY-Table Team
Curtis Kortman took charge of designing and building the XY-Table. It was his duty to get the code working that interfaces the motors to the upper level code written by the software team. He then has to develop the 3D model of the XY-Table so he can machine and assemble the different components to make the physical finished table.

2.5 Schedule and Work Hours
Figure 35 in Appendix 1 shows the hourly work breakdown for the team as well as the total hours worked on different aspects of the project.

2.6 Budget
The team’s assigned budget is $500. In a desire to be good stewards of the resources available, the team desires to adhere to this amount. To achieve this goal, the team made contact with industrial fluid power and motion company Donald Engineering. The company provided them with stepper motors, drivers, linear rail systems, and motor mounts free of charge. The donated items have amounted to a significant amount of money, delineated in Table 1. Figure 2 shows the remaining balance of Team 08’s budget. See Table 9 in Appendix 2 for an itemized list of where money was spent.

<table>
<thead>
<tr>
<th>Items</th>
<th>Qty</th>
<th>Price/Piece</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nema 23 Closed Loop Stepper Motor</td>
<td>1</td>
<td>$257.00</td>
<td>$257.00</td>
</tr>
<tr>
<td>Nema 23 Stepper Motor</td>
<td>1</td>
<td>$42.00</td>
<td>$42.00</td>
</tr>
<tr>
<td>MBC12101 Stepper Motor Drivers</td>
<td>2</td>
<td>$93.00</td>
<td>$186.00</td>
</tr>
<tr>
<td>Linear Rails</td>
<td>3</td>
<td>$65.00</td>
<td>$195.00</td>
</tr>
<tr>
<td>Carriages</td>
<td>4</td>
<td>$42.00</td>
<td>$168.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td><strong>$848.00</strong></td>
</tr>
</tbody>
</table>
2.7 Method of Approach

Before work began, the team chose a design methodology. The team evaluated many different project management software packages, and decided on using Asana as seen in Table 2. Asana’s philosophy is that one person is accountable for each task. In this way, no task is lacking clear ownership, and so its completion is not jeopardized by a lack of understanding of who is in charge. Team 08’s design methodology consisted of dividing the work into different smaller projects between all of the members of Team 08. Each person took charge of one aspect of the entire project, and within the project, one person took charge of each sub task. Section 2.1, Team Organization, details the division of roles.

Each week the members of Team 08 meet to discuss progress of each of the sections of the project and progress of the project as a whole. Communication outside of meetings occurs through Asana. The team also utilizes Asana to create a schedule with due dates for small tasks and entire projects. Asana can give great reminders to team members to make sure projects stay on track. For communicating documents, OneDrive stores all documents so that team members can work on the reports and other files simultaneously. In addition to these programs, GitHub provides version history and merge.
capabilities to store all of the team’s code in an effective manner. This program ensures the safety of the team’s code, and allows the team to return to previous versions in the case anything gets broken.

Research is another big aspect of the project. Much of the research consisted of simply testing parts; however, for the more expensive parts, such as the microcontroller, a great deal of online research occurred before buying the parts. As said before, the research method used for the cheaper parts, such as the electromagnet and reed switches, consisted of buying a variety of parts and testing them under different conditions, in order to see which part best-suited Team 08's design needs.

Throughout all of Team 08’s, design work, Christian perspective and design norms held great importance in deciding methods of approach and implementation. As seen in the Christian Perspective section in this report, Colossians 3:17 gives the commandment to do everything in the name of Jesus Christ, "giving thanks to God the Father through him." Team 08 took this calling seriously throughout the design of this project and made sure to incorporate this calling into every decision and methodology. This calling had a great impact on the way that Team 08's members communicated with each other, as communication was done in a respectful and loving manner.

In addition to this calling, the design norms of integrity, trust, and caring played a great role in the methodology of this project. The caring design norm pushed Team 08 to make decisions based on the customer's needs and wants out of the chess board, while the trust design norm pushed Team 08 to create a quality chess board that would work 100% of the time, or at least very close. The integrity design norm pushed Team 08's members to be honest with each other, while also being honest with the customer about the progress of the project and problems that occurred throughout the project. The integrity design norm lined up with Team 08's calling seen in Colossians 3:17, and had the same effect on the methodology of the project as well.
3 RESEARCH

3.1 Market Research

Automated chess boards are not a new idea. Several hobbyists have worked on “smart chessboard” projects that sense the position of pieces on a chess board. [4] Also, for a time, a product from Excalibur Electronics was available, which consisted of a chess board that could play against a player by moving the pieces. The latest development in technologically advanced chess boards comes from Chess Automated, a board that provides much the same functionality as the Excalibur did, with the addition of being able to connect to the internet and play against online opponents. Chess Automated is not yet being commercially produced. Both of these automated chess boards, as far as we can tell, use pressure sensors to detect pieces and magnets to move them. More about these existing designs and the differences between them and The Knight can be found under Section 5 - Business Plan.

3.2 Related Technologies

3.2.1 Reed Switches

One of the common ways to build a sensitive chess board involves reed switches. [4] [5] Reed switches are small cylindrical electromechanical components which can sense magnetic fields. They do this using two contacts which bend to touch each other in the presence of a magnetic field. The position at which a magnet first causes a reed switch to turn on is called the reed switch’s pull in point. [6] If the reed switch contacts are already touching, however, the magnet must move farther away before the reed switch will turn off, resulting in a slightly more distant drop out point. [6] The area defined by all pull in and drop out points is called the reed switch’s sensitivity lobe, the area for which a reed switch is sensitive to magnetic fields. A reed switch has different sensitivity lobes depending on the orientation of the magnetic field. [7] The shape of these lobes for sensitivity to a vertical magnet can be seen in Figure 3a, while the lobes for a horizontal magnet are shown in Figure 3b. [7] Most notably, a reed switch will not sense a vertical magnet centered over it. This information affected design choices the team made.

![Figure 3 - Reed Switch Sensitivity Lobes](image)

3.2.2 Chess Engines

In general, a chess engine is a computer program that is given a state of a chess game and analyzes chess moves to decide on the best possible move for a given turn. In our project having a strong chess engine
was essential if serious chess players were to take our project seriously. Chess engines are considered stronger the more looks ahead that they can look. For example, if a chess engine can only look 3 to 5 moves ahead, it is considered a beginner level chess engine, but if the engine can reach depth of 25 or greater moves ahead it is considered at a Grand Master level.

Doing research, we found that major chess engines communicate on a known protocol, UCI. UCI is an acronym for Universal Chess Interface [8]. UCI would allow our chess table to communicate with the best chess engines in the world using a known protocol. The team chose to use Stockfish, which is considered one of the strongest open source chess engines in the world that communicates using UCI.

3.2.3 Voice Control

Much of the research on voice control centered on finding a method for speech recognition that did not require an internet connection, yet was reliable and adaptable to the specific task of controlling a chessboard. Research showed that Pocketsphinx, developed by Carnegie Mellon University, was one of the most widely used and adaptable offline speech recognition libraries available, with support for custom dictionaries, language models, and grammars – where grammars are ways to specify that users can only speak words in certain orders [9].

Pocketsphinx also was found to be somewhat adaptable to interfacing with Python code, as a small number of GitHub projects were discovered that wrapped some of the C code of Pocketsphinx into Python-callable functions, preserving the core functionality of Pocketsphinx.

As Raspberry Pis do not come with any built in recording device such as a microphone, research was done to find microphones that could work with a Raspberry Pi and still be of high enough quality for voice control to be relatively reliable. As USB microphones have sound cards built into them, they offer a fairly low-cost method for speech recording that can be easily added onto a Raspberry Pi [10].
4 SYSTEM DESIGN

4.1 System Requirements

The system that the team develops should not be unnecessarily complex, and contain no vestigial pieces. In the end, the team strives for the simplest implementation of the following system requirements of both software and hardware.

4.1.1 Software Requirements

The team chose to follow the following four table objectives:

- The table shall determine the current location of the chess game pieces.
- The table should determine the player’s desired move via voice
- The table shall allow chess to be played against artificial intelligence
- There should be a multiplayer web server and mobile application for multiplayer chess

4.1.2 Hardware Requirements

The team chose to follow the following three hardware objectives:

- The hardware shall determine if a piece is on top of each square of the game board.
- The hardware shall move chess pieces around the chessboard
- The hardware should convert speech to electrical signals

4.1.3 Performance Requirements

The board should be able to move pieces at a reasonable speed, where completing a move takes no longer than 2.5 seconds for a non-capturing move and no longer than 6 seconds for a move where a piece is captured. The speed will also be slow enough to ensure reliable performance.

When moving pieces, the board should not make noise above the level of hushed conversation. In this way, noise will not interfere with the players’ enjoyment or focus on the game. Power consumption will be limited to a reasonable amount and the mechanics of the board will function reliably.

Voice recognition will be powerful enough to allow players to speak at a rate nearly as fast as standard speech while still providing accurate translation. The app will have a delay under 2 seconds when sending a desired move to the board.

4.2 System Architecture

The architecture to achieve the aforementioned requirements is detailed in a system overview below. The architecture is divided into two areas: software and hardware. Even though the software and hardware are intertwined, they can be thought of separately. The system architecture also includes a multiplayer web server which will be used for a multiplayer game through an iOS application. Although the web server and phone application may be a part of the system architecture, they are not required in order to play chess. Figure 4 shows a block diagram of the entire system.
Figure 4 - Full System Overview

**Chess Table**

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Main interface for the user</th>
</tr>
</thead>
</table>
| Actions | - Move physical pieces using XY-Table and Sense where pieces were moved  
|         | - Take in voice commands and converts to a move  
|         | - Allows for multiple game modes |

**Hardware Components**

- Raspberry Pi 2  
- Arduino Nano (x4)  
- XY-Table  
- LED Matrix  
- Buttons and Microphones

**Software Components**

- Multitude of Python, C++, and Arduino Sketches

**Bluetooth LE Module**

Phone to Board communication: Sends moves and error messages to and from the board.

**iOS Application**

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Allows for a multiplayer game between two iOS devices</th>
</tr>
</thead>
</table>
| Actions | - Play a practice game  
|         | - Play a multiplayer game with the server  
|         | - Connect to the chess table with Bluetooth |

**Software Components**

- iOS Application written in Swift

**Socket.IO Server**

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Allows for a multiplayer game between two iOS devices</th>
</tr>
</thead>
</table>
| Actions | - Acts as the communication  
|         | - Use web sockets to allow live communication |

**Software Components**

- Node.js File
4.3 Electronic Architecture

The components that constitute the electrical architecture include microprocessors, stepper motors, a piece-sensing array, a button panel, two microphones, power supplies, Bluetooth module, and an electromagnet. Each of these is detailed below, noting how they interact within the overall architecture. A graphical summary is portrayed below in Figure 5, refer to Figure 5 to understand how the components listed below are connected.

4.3.1 Raspberry Pi 2.0

The Raspberry Pi microcontroller connects the otherwise disparate components of the design. The Raspberry Pi controls the motor drivers, electromagnet, and the LED lights. By controlling the motor drivers and electromagnet, the Raspberry Pi controls the movement of the chess pieces. The microprocessor also reads in the Reed Switch Array and inputs from the user such as voice and button presses to start a new game and select the game mode.
4.3.2 Motor System

The motor system can move an electromagnet underneath the playing surface in order to move pieces around the board. In order to accomplish this, motor one moves the electromagnet in the x direction and motor two in the y direction. The motor driver are controlled by individual Arduino Nanos control and power the two motors. The Arduinos communicate with the Raspberry Pi with a handshake method. They are sent a command and send back "done" when they are finished.

4.3.3 Electromagnet

The electromagnet is used to drag the pieces around the board. The electromagnet’s power is turned on and off by the Raspberry Pi’s GPIO connected to an NPN Darlington 1A transistor. 24V powers the electromagnet. The team has ensured that the magnet does not retain any residual magnetic field that will interfere with pieces or reed switches when not in use.

4.3.4 Reed Switch Array

To interpret a human move, the chess engine tracks the pieces on the board. It records whether or not a piece is currently on top of a square at the beginning and end of each turn. A piece’s presence on top of a square is detected by the reed switch array. A high signal represents the presence of a piece. A low signal represents the absence of a piece. The reed switch array outputs its data to the Raspberry Pi.

4.3.5 Microphones

Two were required in order to implement voice-activated piece movement from both sides of the board. These only record human speech for three seconds as directed after a button press, and connect directly to the Raspberry Pi via USB.

4.3.6 Power Supplies

Several different supply voltage levels were required in order to power the Raspberry Pi and Arduino (5V), motors and electromagnet (24V), and LEDs and a fan (12V). The power supplies are plugged into a power strip connected to a switch on the bottom of the chess board. The cable from the switch runs out one of the chess table legs.

4.3.7 Capacitive Filter

This filter was added to the power rails of the LED Matrix. This filter included two 10uF tantalum capacitors and one 3300uF electrolytic capacitor. This power supply operated at about 5.2V but without the filters would spike up to ~6.0V for a few microseconds.

4.3.8 Power Filter

This filter was added between the 24V power supplies and the stepper motor controllers. When the stepper motor controllers were disconnected the noise would completely disappear. Under the direction of Professor Yoon Kim, the team added a filter as shown in Figure 6.
4.3.9 LED Matrix

A 32 by 16 LED matrix was chosen for the user interface design. The LED matrix is controlled by an Arduino Nano because of its good timing control. The Arduino Nano is controlled by the raspberry pi with serial commands. There is no handshake as the display will broadcast the last image direction sent to it until it is sent another image or a clear command. The LED matrix is used to display:

<table>
<thead>
<tr>
<th>IMAGE</th>
<th>SERIAL COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Knight Logo</td>
<td>&quot;logo&quot;</td>
</tr>
<tr>
<td>Menu Options</td>
<td>Plain text – Ex. &quot;Pick Side&quot; &quot;New Game&quot;</td>
</tr>
<tr>
<td>Current Turn</td>
<td>Plain text – Ex. “White Move”</td>
</tr>
<tr>
<td>Microphone Prompts</td>
<td>Plain text – Ex. &quot;Rec Voice&quot;</td>
</tr>
<tr>
<td>Chess Pieces</td>
<td>&quot;Pawn&quot; &quot;Knight&quot; &quot;Queen&quot; &quot;King&quot; &quot;Rook&quot; &quot;Bishop&quot;</td>
</tr>
<tr>
<td>Move</td>
<td>“move” followed by the string representation of the piece moving and the position to move to – Ex. “movePe4”</td>
</tr>
<tr>
<td>Capture Sign</td>
<td>&quot;capture&quot; followed by the string representation of the piece getting captured – Ex. “CaptureQ”</td>
</tr>
<tr>
<td>Board Errors</td>
<td>Plain text depending on Error – Ex. “Fix Board” “Move Error”</td>
</tr>
<tr>
<td>Piece Placement</td>
<td>&quot;rb&quot; followed by “r”, “b”, “g”, or “x” based on if a piece was in that position or not – Ex. “Rbxxxxbbgbb….bbxxxxrxxxx…”</td>
</tr>
<tr>
<td>Winning and Losing Prompt</td>
<td>Plain text – Ex. &quot;You Lose” &quot;You Win&quot;</td>
</tr>
</tbody>
</table>

4.3.10 Buttons

Capacitive touch buttons were chosen for the user interface design, to allow for the buttons to be placed under the wooden table top for a more aesthetically pleasing design. The design uses four buttons. One button is a selection button while another button is for scrolling. These two buttons together can be used to go through the menu options, while the select button is also be used to end a player’s turn. The other two buttons are both used to prompt the voice control. After a button press the microphone records for three seconds and then the recording is converted into a chess move.

4.4 The Chess Table

4.4.1 Frame Design

The base of the table is made out of steel to accomplish a sense of permanence and ensure that the frame remains rigid and square. The steel was readily available at school, and some of the team members
wanted to learn to weld. The hollow square stock allowed us to run the cables up into the enclosed portion of table. The table surface was set at a height of 36 inches to resemble a counter top. A bar stool would fit well underneath it.

The steel frame supports a webbing that holds up our chess board surface. The webbing is also made of steel as seen in Figure 7. The steel also gives a smooth surface for the XY table to be mounted on.

![Figure 7 - Steel Frame, Support Structure, and XY-Table](image)

4.4.2 Table Design

The sides of the table are made with stacked plywood for the aesthetics; we had to make the side panels out of thin pieces of wood to allow all of our electronics to fit inside, yet also make them strong enough to support the edges of the table if someone leaned on it. The stacked plywood worked well for this purpose. These are bolted to the steel frame so they are removable for troubleshooting and repairs, yet held firmly. The surface of the board is glued onto the frame. It is made of \( \frac{1}{4} \)" plywood. A complete Table Design can be seen in Figure 8.
4.4.3 Playing Surface

In order to play the game of chess, a 64 square (8x8) board is needed, as well as sixteen squares (2x8) on each side to hold captured pieces. The squares are sized near regulation size, which is 2 ¼", which is enough room for the motor system and the electromagnet to move pieces past each other. The playing surface encapsulates all of the electronics and hides it.

4.4.4 Button Panel

Due to the variety of playing options the board is capable of, as well as chess-related options such as pawn promotions, buttons were needed for users to select game modes and decide what piece to promote a pawn to that has crossed to the other side of the board. These buttons were engraved into the surface to make them fit with the design and keep them unobtrusive to the playing surface.

4.4.5 LED Matrix

The LED matrix is mounted into the top surface under white printer paper and a clear plastic cover to dim the lighting for a more aesthetically pleasing look.

4.5 Software Architecture

This section discusses the architecture and structure of the software. The team used a combination of open source software and custom written software. The open source software includes the chess engine and other frameworks and libraries that are acknowledged in the Acknowledgments section. The code that was written by the team includes modifications to the chess engine, classes for interfacing with hardware, the XY-Table class and the chess table class that encapsulates all of the code for the physical chess table. The overall hierarchy of the software objects can be seen in Figure 9.
4.5.1 Chess Game

This class is the main interface between the physical board, the chess game, and the chess engine. The chess game object is the main object that gets created in the main file. The object is a highly modified object from “command-line-chess”, an open source lightweight chess game. This object is what handles the flow of the game and communication between all other objects inside of it.

4.5.2 Chess Engine

Initially the team planned to use a lightweight, open source, python chess engine, but after some testing and more research on chess engines, the team decided to go with an open source chess engine named Stockfish. Stockfish uses a standard chess protocol named UCI that makes it easy to interact with. The interaction between our python code and the chess engine was handled with a python library named
Python Chess. The combination of Stockfish and the Python Chess library provides the chess table with the ability of a grand master level of play.

4.5.3 Hardware Interfacing Classes

Multiple classes were written to interface between the hardware and the chess game. These classes are imperative to the success of the project because they connect the hardware to the software that is running the chessboard. These classes include the communication to the motors, the electromagnet, the matrix of reed switches and all of the user input and output, which includes buttons, microphones, and an LED matrix. In addition, a Bluetooth wrapper for sending and receiving moves to the iOS Application was implemented.

4.5.4 Chess Table

The chess table object encapsulates a XY-Table object that was written to handle all of the motor movements and magnet commands. The object is able to take move commands generated from the chess engine to control the board as well as respond by reading the current state of the board and returning a move to send back into the chess game.

4.5.5 Chess Board

The chess board object is where the state of the chess game is kept. It also handles whether or not a move is legal and general piece tracking. This was also highly modified from “command-line-chess”. We needed to modify what this object to allow for piece tracking. Giving each piece its own identification we were able to tell what piece was captured and where to put the captured pieces in the captured bins.

4.5.6 iOS Application/Web Server

One of the coolest features of the chess table is its ability to connect to an iOS application over Bluetooth and play against an opponent across the world. To accomplish this, first a proof of concept iOS application was written using Apple’s new language, Swift. The application allows users to play a practice game against themselves, as well as play online on a multiplayer server. Currently, the game server can only handle one game at a time and will kick off other players that try to join. In addition, when playing in a multiplayer game, the user has the option to connect their app to the chess table via Bluetooth. This allows the user to play against their online opponent on a physical chess table. A screenshot of the iOS app can be seen in Figure 10.

Figure 10 - Proof of Concept iOS App
4.6 Design Alternatives and Selection

4.6.1 Chess-piece Movement

4.6.1.1 General Requirements

Requirements for chess piece movement are detailed in Section 4.1.2, and are summarized as follows: The chess pieces shall have the capability of automatic movement. The pieces shall move without a visible movement mechanism. The movement should be smooth and quiet.

4.6.1.2 Alternatives

The team discussed the use of different movement mechanisms. The proposed mechanisms included linear actuators, stepper motors with ball screws, stepper motors with timing belts, purely magnetic movement, and a claw-like device to pick and place the pieces.

4.6.1.3 Selection

Stepper motors with a timing belt fit the needs of the project the best. The decision matrix (Table 4) shows some of the different qualities that went into making the decision. Ultimately, the stepper motors with ball screws proved to be a suboptimal solution because of cost and the fear of oscillation with too long of a screw shaft. Using linear actuations did not instill confidence with accuracy without expensive components, and it did not offer a compact solution. Purely magnetic movement also looked expensive, and required an overly complicated implementation for what the team was trying to accomplish.

<table>
<thead>
<tr>
<th>Option</th>
<th>Compact</th>
<th>Feasibility</th>
<th>Cost</th>
<th>Sound</th>
<th>Precision</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Actuators</td>
<td>40</td>
<td>90</td>
<td>65</td>
<td>50</td>
<td>45</td>
<td>62</td>
</tr>
<tr>
<td>Ball Screws</td>
<td>80</td>
<td>65</td>
<td>60</td>
<td>90</td>
<td>100</td>
<td>77</td>
</tr>
<tr>
<td>Timing Belts</td>
<td>80</td>
<td>90</td>
<td>75</td>
<td>90</td>
<td>90</td>
<td>85</td>
</tr>
<tr>
<td>Purely Magnetic</td>
<td>100</td>
<td>30</td>
<td>50</td>
<td>100</td>
<td>100</td>
<td>70</td>
</tr>
</tbody>
</table>

4.6.2 Microcontroller

4.6.2.1 General Requirements

The microcontroller must be powerful enough to handle at least 96 reed switches (8x8 board and two areas for captured pieces) and python software including the chess engine, piece tracking, and other functions needed in the software. The microcontroller must be user friendly and will allow the python code to be easily added into the microcontroller. The microcontroller also must have enough pins to handle all of the reed switches, motors, buttons, and anything else such as LEDs that could be added in the future.

4.6.2.2 Alternatives

Arduino Mega, Raspberry Pi 2.0 and other Raspberry Pi models.

4.6.2.3 Selection

A decision matrix used in the decision of the microcontroller may be seen in Table 5 below. However, once the decision was made to use a Raspberry Pi 2.0 some challenges arose.
Table 5 - Decision Matrix for the Selection of the Microcontroller

<table>
<thead>
<tr>
<th>Option</th>
<th># of Pins</th>
<th>Clock Speed</th>
<th>Display</th>
<th>User-Friendly</th>
<th>Pre-Knowledge</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi A/B</td>
<td>60</td>
<td>75</td>
<td>100</td>
<td>90</td>
<td>95</td>
<td>81</td>
</tr>
<tr>
<td>Raspberry Pi A+/B+</td>
<td>80</td>
<td>85</td>
<td>100</td>
<td>92</td>
<td>95</td>
<td>89</td>
</tr>
<tr>
<td>Raspberry Pi 2.0</td>
<td>90</td>
<td>95</td>
<td>100</td>
<td>94</td>
<td>95</td>
<td>94</td>
</tr>
<tr>
<td>Arduino Mega</td>
<td>95</td>
<td>85</td>
<td>100</td>
<td>85</td>
<td>50</td>
<td>85</td>
</tr>
</tbody>
</table>

Raspberry Pi 2.0 outperformed the alternatives. Listed below are some of the Raspberry Pi 2.0 specifications.

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB LPDDR2 SDRAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core

One of the main reasons the Raspberry Pi 2.0 was selected was due to its 40 GPIO pins. More pins are ideal in the choice of the microcontroller due to the great amount of components that will need to be attached to the Raspberry Pi 2.0. The Raspberry Pi A and B have 17 pins, and the Raspberry Pi A+ and B+ both have 26 pins. These numbers of pins may not have been enough to handle all of the reed switches and then the possibility of buttons and LEDs. The Raspberry Pi 2.0 also had better specifications all around than the other Raspberry Pi models. At the time the microcontroller decision was made, some requirements were uncertain, including how much RAM and how many USB ports would be needed, so it was safer to choose the microcontroller with the better specifications.

The Arduino boards were also considered and the Arduino Mega (ATmega2560), at 54 I/O pins, did have enough; however, the other specifications were not as good as the Raspberry Pi 2.0. For instance, the clock speed of the Arduino Mega is 16 MHz compared to 900 MHz (quad-core) with the Raspberry Pi 2.0. Other microcontrollers were also looked into; however, none matched the performance of the Raspberry Pi 2.0 while also having enough pins to support all of the reed switches and LEDs that would be needed.

In addition to the performance of the microcontroller, the usability and user-friendliness of the microcontroller was also of great importance in the selection of the microcontroller. The Raspberry Pi had been used by some of the team members of Team 08 beforehand and has proven to be very user-friendly. Also, the remaining members of Team 08 were interested in learning how to use a Raspberry Pi microcontroller as it is a common microcontroller for small project use.
4.6.3 Board

4.6.3.1 General Requirements

The board shall be thin enough for the reed switches to work efficiently and for the electromagnet to work efficiently as well. The board shall hold the pieces and the pieces shall be able to slide across the board easily.

4.6.3.2 Alternatives

The board may be made of wood, plastic, metal, or glass. There are various choices for the thickness of the board.

Various square sizes are also possible and square size is typically chosen based on the size of the pieces.

4.6.3.3 Selection

A wooden surface was chosen for its simple elegance and rigidity at widths around ¼” to 3/8”. Wooden pieces are also easier to embed magnets within than metal or glass pieces, and for aesthetic purposes the board and pieces were desired to be made of the same material. Wood is also considered a more elegant and classic look for a chessboard than plastic, which was the only other material relatively easy to embed magnets within.

The square sizes for the board were chosen to be 2 ¼”, in order to accommodate 1” diameter chess pieces. This would provide the pieces room to slide past each other, as in the case where a knight on the first move wants to move past the row of pawns. 2 ¼” is also a standard size for chess board squares.

4.6.4 Piece Tracking

4.6.4.1 General Requirements

Chess pieces shall be easily tracked and their positions shall be stored in a database for use throughout the game by the software.

4.6.4.2 Alternatives

Choices include reed switches, Hall Effect sensors, RFID chips, and pressure sensors, as well as an above-the-board camera to monitor the board position.

4.6.4.3 Selection

Reed switches were chosen to implement piece tracking because of their relatively low cost and for being easier to build into the board design than the alternatives, as shown in the decision matrix below in Table 6.

<table>
<thead>
<tr>
<th>Option</th>
<th>Cost</th>
<th>Layout</th>
<th>Size</th>
<th>Precision</th>
<th>Pre-Knowledge</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reed Switches</td>
<td>90</td>
<td>80</td>
<td>95</td>
<td>90</td>
<td>70</td>
<td>87</td>
</tr>
<tr>
<td>Hall Effect Sensors</td>
<td>85</td>
<td>70</td>
<td>75</td>
<td>95</td>
<td>70</td>
<td>80</td>
</tr>
<tr>
<td>RFID chips</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>95</td>
<td>70</td>
<td>82</td>
</tr>
<tr>
<td>Pressure Sensors</td>
<td>90</td>
<td>80</td>
<td>95</td>
<td>75</td>
<td>70</td>
<td>84</td>
</tr>
<tr>
<td>Camera</td>
<td>50</td>
<td>70</td>
<td>85</td>
<td>100</td>
<td>70</td>
<td>75</td>
</tr>
</tbody>
</table>
The main reason for not choosing RFID-based piece tracking was because of the relatively high cost of RFID tags, as one tag per piece would be required, and a method of sensing that could cover every square would have to be determined, which would likely have involved using an RFID reader below every square—a very expensive sensing method.

Pressure sensors would also be expensive, as each square would need a pressure sensor to tell if a piece was on it. In addition, the pressure sensor would have to be built into the board surface and would likely increase the width of the board beyond the practical range of the electromagnet to reliably attract and move pieces.

Utilizing an above-board camera would have an advantage in that the different types of pieces could also be distinguished, rather than simply telling if a square is occupied or not. The disadvantages of the camera are the need for complex image processing and for an addition to the board for mounting the camera above the playing surface. This would erode the aesthetic quality of the board, as without the camera the chess pieces themselves would be the tallest items in the design and the board would look and feel more like a standard chessboard.

Hall Effect sensors were promising in that they could work with the already-existing magnets embedded in the chess pieces, but they would also be more expensive than reed switches or pressure sensors. Reading an array of 96 Hall Effect sensors (covering both the 8x8 board and the two areas designated for captured pieces) would be difficult due to their analog measurements, compared to the digital readings (open or closed) from reed switches.

The reed switches, then, were the best method that was not overly expensive or detracted from the functional and aesthetical qualities of the board. Since the reed switches are small, shaped similar to resistors, and relatively inexpensive, an array of reed switches with one per square can be implemented. In order for a large array of reed switches to align with the squares of the chess board without the use of excessive connection points, it is common for reed switches to be arranged in rows and columns. In this configuration, each reed switch must be connected to a diode in order to avoid false positives. Although these diodes add to the cost of the reed switch array option, this option still has a reasonable cost while maintaining a thin surface width of the board for the electromagnet to work effectively.

Reed switches also have the advantage that the pieces already need to have magnets embedded within them so that they can be moved by the electromagnet. The reed switches, then, use the already-existing magnets in the pieces for the purpose of sensing piece locations: the magnetic fields from the magnets in the pieces cause reed switches directly below the pieces to close so that current can flow through. When no piece is present above the square, then, the reed switch remains open and no current can flow through it. This allows for testing by checking for open or closed circuits as they result from the state of the reed switches.

4.6.5 Table

4.6.5.1 General Requirements

The frame determines the overall shape of the product. It shall encapsulate all of the electronics and support the product as a whole. Additionally, it should be aesthetically pleasing, without incurring additional motor noise.
4.6.5.2 Alternatives

The choices include a chess table, or a table top chess board. Additionally, we could choose to make the product out of wood, metal, or a combination of the two.

4.6.5.3 Selection

Making a chess table, rather than a tabletop chess board, with a frame of steel was decided on for the encapsulation of the electronics. The main influence of the decision was the ability to keep the steel frame relatively square for the XY-Table to move on.

We decided to make an entire table. We liked the sense of permanence that it gave. A table top setup would look out of place on many tables. We also doubted our ability to make the prototype thin enough to look appropriate as a table top fixture. Therefore, we thought it would look more well integrated, and less obviously an electronic chess board, as an entire piece of furniture. To keep the table simple and elegant, as well as easier to build, we decided to build it with a square or rectangular shape. We also decided to use metal because of the accessibility of the materials, and the extra rigidity and to avoid the effects of wood warping. This is summarized in Table 7: our decision matrix below.

<table>
<thead>
<tr>
<th>Table 7 - Decision Matrix for Choosing a Chess Table Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>Table Top - Wood</td>
</tr>
<tr>
<td>Table - Wood</td>
</tr>
<tr>
<td>Table - Steel</td>
</tr>
<tr>
<td>Square Table</td>
</tr>
<tr>
<td>Round Table</td>
</tr>
</tbody>
</table>
4.7 Pre-Integration Testing

4.7.1 Surface

Initial testing was done to determine the design and materials to be used for the main board surface. Two ideas were thought of and tested. The first idea was to have wood on top of plastic with L-bar supports, reed switches, diodes, and wires put in trenches in the wood or plastic. The prototype, seen in Figure 11, was much smaller than the final design of the table would be, being the size of a 4 x 3 square chess board. The wiring including the reed switches and diodes were glued into engravings on the bottom of the wood, as seen in Figure 12.

![Figure 11 – Support for first test surface](image1)

![Figure 12 – Bottom Side of First Test Surface](image2)
Software tests were created that moved from space to space on a 4 x 3 board that was made to test the reed switches and diodes. The test was successful for 12 of 12 spaces after much testing. All of the reed switches appeared to work and the range of the reed switches appeared to be good. Additional tests were created to test piece movement past other pieces. It was expected that since the magnets in the pieces were facing the same direction they would resist each other and cause each other to move when the pieces moved too close together. Through testing it was seen that the pieces could actually touch each other and still did not push each other away. More tests were created to deal with pieces lagging behind the electromagnet when they were moved. Different options were tested such as moving the piece slightly past its destination and then back again, and it appeared that the majority of the ideas would work as good solutions.

The tests that were implemented to test this design were completely successful and validated this design as a good design. The only issue with the design was the aluminum L-bars, which did cause some interference with the reed switch array and shorted them out; however, tape was put over the 2 L-bars, which made a T-bar, in order to fix this. It was decided that a different insulation material would be used in the final design, or that the aluminum rails would actually be used as wires in the final design.

The other idea, seen in Figure 13, was a simple plastic layer with the reed switches, diodes, and wires put in trenches cut into the plastic.

![Figure 13 – Second test surface made of half inch Polycarbonate](image)

The Board was too thick and sticky for the pieces to be able to move across it; however, doing the engravings in the plastic worked well. This test was important because it showed the feasibility of doing the engravings in the plastic, which would be better for the first design idea. The implementation of this design would be much neater as the trenches that the wires would flow through would be machined nicely into the plastic. A layered design, with the circuit in the lower plastic layer, and a removable surface layer above it, could possibly allow people to see the circuit design of the reed switches with their diodes.
Team 08 came to the decision to use the first design idea, but to engrave into the plastic, put the circuit in the plastic layer, and use the aluminum T-bars as wires.

The wooden surface was created by taping off squares that would be stained with a dark wood stain as seen in Figure 14. The entire board was then covered with a clear wood varnish. The tape was then removed and the entire board was covered with a dark wood stain, with the clear wood varnish stopping the dark wood stain from absorbing into the wood in those sections. Figure 15 shows the wooden chess surface without the final engravings we put into the surface.

Figure 14 - Taped Up Wooden Chess Surface

Figure 15 – Wooden Chess Surface (Without Engravings)
After the surface was complete, the CNC machine was used to engrave piece icons into the capture bins, and the engravings were then filled with wood filler mixed with a dark wood stain. Figure 16, shows the complete wooden chess surface with engravings and wooden chess pieces on top.

![Figure 16 - Wooden Chess Surface (With Engravings)](image)

4.7.2 Reed Switch Sensor Board

The small 4 x 3 reed switch board was tested in the surface tests; however, as the designs progressed, the final 8 x 12 array was created. The CNC was used to put grooves in the plastic for the reed switches and diodes, as seen in Figure 17.

![Figure 17 – CNC Making Engravings in Plastic](image)
Figure 18 shows the completed trenches in the plastic.

![Completed Reed Board Trenches](image)

After the trenches were complete, the reed switches with their partnering diodes were tested individually.

### 4.7.2.1 Single Reed Switch Testing

Initial testing of the reed switches was accomplished by connecting the reed switch to be tested to an ohmmeter to be able to tell when the reed switch was open or closed. All reed switches acquired were classified as "nominally open," so they only close and allow current to pass when in the presence of a strong enough magnetic field.

The first test was to figure out the maximum range of the reed switches by placing each reed switch as if it was embedded below the board and testing how far away the magnet could be placed in order for the reed switch to close. Three different magnets and five different reed switches were tested.

The second test was to determine the range of sensitivity for a reed switch at a set vertical distance away from a magnet (a distance of 1/4"), as if the reed switch was embedded horizontally below the board and the magnet was on top of the board moving horizontally. The range of sensitivity for the reed switches to close was determined to be roughly circular, with two circles of sensitivity each offset from different ends of the reed switch. All combinations of the three magnets and five reed switches were tested.

The results of these tests showed that reed switches that could close given smaller magnetic fields were optimal, and that large magnets were better than small magnets. Both of these result in the largest possible sensitivity radius for the reed switch, which was about 1 inch.

Tests were additionally performed with and without a 3/16" wooden board between the magnet and reed switches, and these tests demonstrated that the wooden board had negligible effect on the magnetic field and sensitivity range of the reed switches. The most promising magnet and reed switch combination resulted in approximately a one-inch sensing radius at each end of the reed switch when sensing through a 3/16" board. This used the largest magnet (1/2” diameter) with the most sensitive reed switches.
4.7.2.2 Reed Switch Array Testing

In order to effectively analyze an array of reed switches with a minimal number of I/O pins, the testing strategy developed was to test one row of the board at a time, giving it a HIGH input and rest LOW. Then, the columns of the array can be read, and columns that are HIGH contain a piece in that row and columns that are LOW do not. For each reed switch, a diode must be placed in series in order to not allow current to flow in the wrong direction during the testing process. This schematic can be seen in Figure 19.

Since only one row is tested at a time, a decoder can be used to shrink the number of GPIO pins needed on the Raspberry Pi. A 12-by-8 array is needed for the final design, due to the necessity of having a 2-by-8 bank of squares on each side of the chessboard for captured pieces. Therefore, a 4-to-16 decoder could be used to control the 12 rows of the board, and the total number of GPIO pins needed would only be 12 (4 for the rows, 8 for the columns).

![Figure 19 - Reed Switch Array Test Circuit Diagram](image)

A small 3-by-2 array of reed switches was created for testing purposes, and a 2-to-4 decoder was used to control the 3 rows. This circuit diagram is shown above. The array was successfully tested through running Python code on the Raspberry Pi, and code was able to locate the closed reed switches and create a matrix of the state of the board, filled with 0s where there is no piece and 1s where pieces reside.

After the reed switches were tested they were added to the final reed board surface. Figure 20, shows the completed reed array board with a blow up of one reed switch in Figure 21.
4.7.3 Voice Control

Initial testing for the feasibility of voice control was accomplished by setting up and running Pocketsphinx open source voice recognition software on Windows. Pocketsphinx allows for deciphering raw data and WAV files into text, all of which is done without needing to connect to the Internet, as using the Google speech recognition API would have required. This made Pocketsphinx the ideal candidate for voice recognition natively on the Raspberry Pi.

A custom dictionary was created that included only the words needed for operating the chess table. By having a small dictionary, translation was much more efficient since fewer words needed to be considered
to find the best match. A custom grammar was also created; this allowed Pocketsphinx to only recognize words in a certain order, rather than any combination of acceptable words.

Since users were to be required to input locations of pieces based on the coordinates of the chessboard square – for example, “A3 to B4” – issues were found with reliably recognizing the difference between similar sounding letters, namely B, C, D, E, and, to a lesser extent, G. In order to make voice control reliable, then, military codes for letters were used instead: Alpha, Bravo, Charlie, Delta, Echo, Foxtrot, Golf, and Hotel for A through H. After extensive testing, Pocketsphinx was able to reliably recognize commands correctly on Windows, while running continuously.

Testing Pocketsphinx on the Raspberry Pi was also successful when running the C code version continuously. A Python wrapping of Pocketsphinx was found on GitHub, and this was made to run correctly when decoding single raw or WAV files. In order to get the Python version to run continuously, however, it was discovered that the recordings from the microphones would have to be continuously resampled into 16 kHz from 24 kHz, which was the microphone default. After several attempts to perform the resampling, it was decided that it would be best to implement voice control by having the user press a button to signal when to record, rather than record continuously.

4.7.4 Software

4.7.4.1 Get Move from Reed Board Testing

One of the most important parts of the project was the ability to decipher a user’s move from the reed switch matrix underneath the surface of the chess board. To test this functionality, a test file was written that simulated an entire game of play, including multiple types of moves, captures, and specialty moves such as castling. This file let us test converting the ones and zeroes that we received from the reed switch matrix into a chess move object that we could pass to our board object and the chess engine.

4.7.4.2 Chess Engine Testing

The testing of the chess engine was done by playing numerous games at different levels to ensure that the communication between Stockfish, the chess engine, and the physical board was sound. The library Python Chess made communication between Stockfish and our code extremely easy, and since Python Chess is a mature library we hardly had any problems interfacing with it.

4.7.4.3 App/Server Development Testing

The iOS App was created with the functionality to play a practice game, or play against oneself. This allowed the team to diligently test all of the chess logic to ensure that the logic was correct.

Once the chess logic was complete we needed to test the transfer of moves over the server. The server was tested by trial and error. The server is actually quite simple, and requires only 138 lines of code.

Like the server, the Bluetooth communication was tested by trial and error. The Bluetooth module that we chose had libraries online that made sending and receiving messages quite simple. We also found a wrapper for the Bluetooth module for Swift, the language of our iOS application. These libraries are referenced in our acknowledgments section.

4.7.5 LED Matrix

The LED Matrix was connected to the Raspberry Pi, and example code provided with the matrix was modified to create a simple class to allow for printing two words to the matrix—one for the top half and one for the bottom half. This was accomplished after initial wiring difficulties due to an incorrect wiring
diagram in the LED matrix tutorial. The LED matrix code was in C++, so the text printing class managed to interface with the C++ code by using Python to call sub processes to run the C++ code.

Software was also written in Python to allow the LED Matrix to display images by setting different pixels on the 32 x 16 matrix to specific colors. This software all worked until the LED matrix was tested for use during gameplay and the Raspberry Pi could not handle sending signals to the LED matrix and running the chess engine. It was then decided to use another Arduino Nano to control the LED matrix, similar to how the motor control worked. Software was written to send serial messages from the Raspberry Pi to the Arduino Nano and vice versa. These messages trigger functions on the Nano to display text or images on the LED matrix. This system was then tested during the play of a game and using the Nano for the LED matrix proved to solve the problem.
4.8 Integration and Testing

4.8.1 Reed Switches
The full 8 x 12 reed switch array was created after grooving plastic with the CNC machine, cutting aluminum rails for support, and gluing the plastic to the aluminum. Because the reed switch array requires long wires in both the horizontal and vertical directions, and the wires cannot touch each other, a soldering iron was used to burn off insulation from the horizontal wires in the locations where they would have to connect to the reed switches.

The aluminum rails of the reed switch array abutted the vertical wires of the array, making the aluminum rails conductors in the array. In order to electrically isolate the aluminum rails, they were screwed into wooden supports and the wooden supports were screwed into the main table frame. Finally, the wires from the reed switch array were connected to the Raspberry Pi GPIO pins and the decoder for the inputs.

4.8.2 Electromagnet
For initial testing of the electromagnet, two 3/16" boards were placed together to create a 3/8" board, and magnets were placed on top of the board and the electromagnet was moved below the board. The 3/8" separation was expected to be close to the width of the actual board when accounting for the space needed to embed the reed switches below the board.

Tests were performed by adding weight above the magnets to simulate the weight of chess pieces and to approximate the actual amount of friction for wooden pieces with felt. The electromagnet was found to attract the biggest magnet from 1/2" and move it reliably with the additional weight. The smaller magnets that would be embedded in the pieces, however, were not reliable for moving the additional weight.

When having multiple pieces near each other with embedded magnets, the pieces were found to repel each other slightly. This actually was helpful to keep the electromagnet from grabbing more than one piece at a time and allowing the pieces to be able to be moved close by each other.

Two more electromagnets were acquired that were more powerful than the original electromagnet. Testing revealed that both worked well with the 1/2" magnet, and the difference in attracting power was small but noticeable. After determining that the weight of the electromagnet was not an issue for the XY-Table, the various electromagnets were tested on a smooth, polished surface using test pieces with felt on the bottom. The largest electromagnet was found to move the pieces the most reliably, so it was chosen for the final design. This electromagnet needed to be run with 24 volts, so it was wired to be turned on and off through a GPIO pin on the Raspberry Pi.

4.8.3 XY-Table Motors
The stepper motors turn in 1/2000 rotation increments. These increments are called steps. Steps are controlled by sending a pulse to the “clock+” input of our stepper motor drivers. Sending a precise number of steps ensures the accuracy of the stepper motors’ location. The timing of the steps is also critical. The speed of the steps must ramp up as the motor starts, and slow down as the motor stops. This ensures that the motors do not lose their position due to inertia.

Team 08 encountered a problem with the motors early on, as the motors generated a lot of heat. The team understood this is from the fact that the motors always run at full current, even when off, but the temperature was unsafe. The motors reached temperatures too hot to touch. Using a potentiometer, the team adjusted the current through the motors from 4 A down to 1.5 A. The motors still functioned well,
and were safe to touch. The torque will have to be monitored in the final implementation to maintain an optimum balance of heat generation and sufficient power.

4.8.3.1 Sequential Motor Movement

4.8.3.1.1 For Loops

Team 08 initially decided to use Python loops to send the steps to the motors. Figure 22 shows an example of the architecture implemented first (a rough sketch – it would not actually work).

```
for (steps = 0; steps < stepsToTurn; steps++):
    stepPin = HIGH;
    wait(1);
    stepPin = LOW;
    wait(1)
```

Figure 22 - Loop-based step control

This solution failed because it did not have good timing performance. The time between each pulse was variable. The team hypothesized background processes running in Linux consumed processor time, and caused variability in the timing. Gaps in timing could be heard by loud chunking sounds and vibrations in the motors.

4.8.3.1.2 PWM with Interrupts

Since the timing was not consistent with for loops and wait statements, the team decided to use the software PWM function provided with the GPIO library of the Raspberry Pi. Calling the PWM for the same amount of time did not allow for precise numbers of rotation, so the team counted the number of steps issued from the PWM with an interrupt. This architecture, shown in Figure 23, proved the best results. The rotation never had errors larger than one degree, and it enabled smooth motion.

```
steps = 0
detectInterrupt(stepPin)    # set an interrupt on the step pin
PWM(stepPin)               # turn on the PWM signal

handleInterrupt:           # handle interrupts on the step pin
    steps++                 # turn off the pwm after desired number of steps
if (steps == stepsToTurn) stopPWM(stepPin)
```

Figure 23 - PWM with Interrupts Code Architecture

4.8.3.1.3 Parallelized Motor Movement

A big part of the motor control is being able to move both the X motor and the Y motor at the same time to allow for diagonal movements for pieces such as the queen and bishops. Both multiprocessing and multithreading were researched and tested in order to see which would be a better fit for implementing the parallelization of the motors. Through various tests, it was found that neither multithreading nor
multiprocessing on the Raspberry Pi would work well because when the processes would run at the same
time the motors would experience a great deal of chunking and the motors would run at different speeds,
even with the PWM with interrupts architecture. This again appeared to be because there were
background processes running at the same time. Several tests confirmed that one motor consistently
moved slower than the other did. The speed of the motors was also variable from test to test. Finally, the
motors would not move one revolution accurately. Table 8 shows the poor results of these tests.

<table>
<thead>
<tr>
<th>Test #</th>
<th># of Revolutions</th>
<th>Motor A Time [s]</th>
<th>Motor B Time [s]</th>
<th>A° off</th>
<th>B° off</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.908</td>
<td>1.064</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.805</td>
<td>1.105</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1.753</td>
<td>1.815</td>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1.846</td>
<td>1.935</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2.589</td>
<td>2.734</td>
<td>15</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>2.489</td>
<td>2.745</td>
<td>56</td>
<td>48</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3.265</td>
<td>3.648</td>
<td>18</td>
<td>65</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>3.483</td>
<td>3.726</td>
<td>77</td>
<td>56</td>
</tr>
</tbody>
</table>

Since parallelization was a failure, Team 08 decided to order Arduinos for motor controllers.
The Arduinos run sequentially with one core (with no background tasks!). Since we want a parallel
execution, the Raspberry Pi will call programs on each of the Arduinos simultaneously creating a
parallelized process. Each Arduino will control one motor; one Arduino will control the x motor and the
other will control the y motor. The Raspberry Pi and the Arduino can communicate via a USB-Serial
connection.

4.8.4 Software

In order to test the software effectively, a test file was created. The test file tests a get move method,
which is used to obtain a move from a matrix of 1’s and 0’s and an updated matrix of 1’s and 0’s. The
updated matrix is obtained from the reed switches that are connected to the GPIO ports, and the matrix for
comparison is obtained from the chess engine’s board. The chess engine’s board is converted into a
matrix of 1’s and 0’s for comparison while the reed switch data is also made into a matrix of 1’s and 0’s.

The test file was found to be very effective in testing the get move method. Various board layouts were
created for testing and different situations were tested.

4.8.5 Wire Management

A wire diagram was created using draw.io in order to make it easier to wire the circuit and to notice
changes that could happen. As seen in Figure 24, the original wiring of the circuit was unorganized and
difficult to understand for all but the person who wired it in the first place.
The wiring was created in draw.io, as seen in Figure 25, and organized and cleaned up a great deal of the circuit. Some of the components of the unorganized circuit such as combinations of NOT and AND gates were taken out and replaced with different components, which cleaned the circuit up greatly. Two perforated boards were ordered to be used for wire management and were screwed to a piece of wood along with the motor controllers, terminal strips for organization, and the power supply. The terminal strips were used in order to make parts of the circuit removable, such as the limit switches and LED matrix. The completed circuit, seen in Figure 26, gave our circuit much more organization and allowed an understanding of the circuit to be much more feasible.
Figure 25 – Wiring Schematic

Figure 26 – Final Wiring in Table
4.8.6 LED Matrix

After the LED matrix was fully functional by itself, using the Arduino Nano, the LED display was connected to the rest of the circuit as seen in Figure 26. The LED matrix flickered multiple pixels along with the images that it was producing, due to noise from either the power supply or the motor drivers. An oscilloscope was connected to the system to see the noise, which is shown in the upper left corner of Figure 27, and it could clearly be seen that the noise would disappear once the power supply was disconnected. First, an AC filter was used to try to solve this noise issue. The ac filter was used to filter out the ac power going into the power supply. This did not appear to solve the noise issue so capacitors were added to the LED display in an attempt to filter the noise out before the power reached the LED matrix. This appeared to have some mild effects on the noise and capacitors were then soldered directly to the LED matrix board to have more of an effect on clearing out the noise. This was effective; however, this solution did not fully rid the display of the noise. Through further testing it was deciphered that one of the motor drivers was causing the noise and so a filter needed to be added between the power supply and that motor driver. A filter was then created by the team and may be seen in Figure 28.

![Figure 27 – Noise that the LED matrix is seeing](image-url)
This appeared to get rid of the majority of the noise; however, it did not get rid of all of it. After further testing the team decided to use ribbon cables with each of the signals running every other wire through the ribbon cable with ground wires running in between all the signals. The ribbon cable, seen in Figure 29, with one of the soldered connectors seen in Figure 30, appeared to get rid of the rest of the noise, leaving the LED Matrix looking clean and pleasing to the eye.
After the LED matrix noise issue was resolved, continued work was done on the display. The matrix was used for user interface and showed users different menu options to choose from, as seen in Figure 31.

The matrix was also used for error handling. When an incorrect move was made or the board was set up incorrectly the LED matrix would display a pixelated image of the board with red dots where all the pieces were that are out of place and green dots where the pieces should be, as seen in Figure 32.
As seen in Figure 33, the display also displayed the pieces that were moving when they were moved by the chess engine or by voice control and the position they were moving to. Also, if there was a captured piece the piece would be displayed next to a skull signifying that it was captured, as seen in Figure 34.
4.8.7 Voice Control

In order to integrate voice control into the table, two microphones were acquired to be placed one on each side of the board so that two human players could play against each other with both using voice control.

As detailed previously, the voice control class created in software required users to speak the coordinates of the source square and the coordinates of the destination square, such as “alpha three to bravo four,” in order to make a normal move. To improve user experience however, support for commands such as “knight to echo three” was added: the voice control class would look to see if there was a knight able to move to the square; if more than one knight was found that could move there, or none at all, it would send back an informative error message and prompt the user to speak again.

The voice control class also required the additional functionality of testing if a move was legal so that users could only move their own pieces and only capture their opponent’s pieces. Additionally, the voice control class was made to test whether the destination square is reachable by the piece being moved—this disallows pawns to move backwards and disallows rooks, bishops, and queens to move through pieces to get to their destinations.

The final integration of voice control with the main software was accomplished by allowing users on their turn to either push a button and then give a voice command or make a move by hand and then press a different button to have the board check the move.
5 BUSINESS PLAN

5.1 Mission for the Company
The company mission is to create the most immersive chess experience available with a physical chessboard.

5.2 Potential Customers & Target Market
The Knight is targeting novice and expert chess players alike. With this system, novice chess players would be able to increase their skills by playing against various levels of the chess engine. For expert chess players, it can be hard to find a competitor who is at the same talent level as they are. The Knight system would let the expert player play against an excellent chess engine or an online player while still using a physical chessboard. The table could also be used for the physically handicapped giving them the ability to speak to the board to dictate moves without the need to physically move the chess pieces.

5.3 Competitive Strategy
The company seeks to compete based on differentiation. As there is no large presence in the market for automated chessboards, competition based on cost is not a consideration. From the few boards that do have built in chess engines, the team seeks to differentiate their product by physically moving the chess pieces, as well as having the ability to dictate moves and play via mobile application.

5.4 Market Sizes & Trends
The market of smart board games is very small compared to other gaming markets. The Knight hopes to bring more attention to the market by bridging the gap between virtual game board games on mobile apps and the physical board games. The company plans to release an iOS app that is a fully featured, multiplayer chess game. The app will be able to connect with the chessboard, which may bridge the gap between virtual and physical gaming.

5.5 Business Strategy & Distribution
The product will be distributed in two ways. First, the board will be available via internet sellers such as Amazon, and shipped to customers using their services. Second, the board will be shown at chess tournaments and tech shows; this will show off the board to niche markets to gain interest. The combination of these two distribution channels should provide enough sales capacity.

5.6 SWOT Analysis

5.6.1 Strengths
The Knight automated chessboard will have the capability of automatically moving chess pieces via voice control, via an app using a web server, with artificial intelligence, and through manual movements. In addition, the board will be aesthetically pleasing to the user, and will allow the user to choose which features of The Knight the user wants to use, in order to better suit the user's wants and needs.

5.6.2 Weaknesses
The chess table will be heavy, require access to a power outlet, and cost a lot compared to conventional chess sets. These limitations shift its appeal away from individual players and toward groups or organizations with greater resources and motivations.
5.6.3 External Opportunities

Chess is a historically significant game, and its popularity is relatively stable. The Knight team, therefore, does not need to worry as much about passing trends among their user base. Also, because it is based on chess, the product potentially appeals to a diverse audience around the world. Other automated chessboards may have some of the same features as The Knight; however, none have been found that have all of the features that The Knight has. As The Knight is more of a combination of existing possibilities than a new paradigm, and because of the cost of a patent, The Knight team will not look to patent the chessboard.

5.6.4 External Threats

The Knight chess table will include features that other automated chessboards have already adopted. Due to this, some of the features of the automated chess table will have less of a “wow” factor, and may not operate as well as other single-feature automated chessboards for a given feature.

5.7 Uniqueness & Existing/Potential Competitors

A small number of products featuring an automated chessboard have been on the market in the past or are currently on the market, yet research has revealed no product as fully functional as The Knight.

One notable board is the Mirage, from Excalibur Electronics. The board was first produced in 1997, but is no longer on the market. The Mirage included the capability of playing versus an artificial intelligence, and used an electromagnet to move the pieces. Each square contained a pressure sensor so that the artificial intelligence would know the most recent move of the human player. One of the major weaknesses of the board, however, was the unreliability of some the moves, as other pieces would occasionally be knocked out of position. The Mirage also was not capable of voice control, nor did it feature app support.

Another similar project, but one that is currently in development, is Chess Automated. Chess Automated also allows for playing against an artificial intelligence on a physical board, and also features an app that allows players to play via the internet from anywhere in the world. The Chess Automated team has currently finished four prototypes, and boasts a sleek design. Similar to the Mirage, Chess Automated uses pressure sensors for piece tracking but does not have the capability of using voice control (Chess Automated, n.d.).

The uniqueness of The Knight, then, is based on seamless integration of a wider variety of features than any other design, past or present.
6 CONCLUSION
The Knight offers a wide range of features to allow for different ways for users to play on a physical chessboard. By allowing users to play against an artificial intelligence, play by hand, play by voice, and play via an app, The Knight contains more features than any physical chess board on the market, while still maintaining the look and feel of a classic game of chess. The many features of The Knight make the table especially appealing to players who desire to play chess on a physical board, but have a harder time finding opponents, and players who have limited mobility, and would benefit from being able to play a game with voice commands.
7 ACKNOWLEDGEMENTS

7.1 People

7.1.1 Jim Kortman - Donald Engineering
Jim Kortman aided the group by donating the linear rails and carriages. He also gave the group tips on how to keep the bearings from getting jammed on the rails to keep the motion smooth.

7.1.2 Phil Jasperse
Phil Jasperse was essential in the fabrication of the chess table. His oversight and guidance was warmly welcomed while welding the table, milling aluminum parts, CNC machining, and general wood working.

7.1.3 Professor Yoon Kim
Professor Yoon Kim’s knowledge of analog systems and electrical noise was a large reason that we were able to figure out the issues with electrical noise on our LED display.

7.1.4 Professor Mark Michmerhuizen
As the advisor for Team 08, Professor Mark Michmerhuizen helped guide the design process. He consistently gave helpful feedback on documentation and helped with maintaining professionalism.

7.1.5 Eric Walstra
Eric Walstra, from Gentex Corporation, was Team 08’s industrial advisor and on multiple occasions offered helpful feedback to the team. He also helped with planning to make sure the project would be finished in time.

7.1.6 Rich Goldy
Rich Goldy Donated the stepper motors and motor drivers that were used for the XY-Table.

7.1.7 Calvin College Engineering Department
Special thanks goes to the Engineering Department at Calvin College for making this project possible through providing funding, giving the education necessary to develop a successful design, and continually teaching from a faith-based perspective. Thanks especially to Michelle Krul for her organizational work and Bob DeKraker for procuring the materials needed for the project.

7.2 Code Repositories and Libraries

7.2.1 Python-Chess
Python-chess is an open source Python library that we used as a wrapper for communicating with a UCI chess engine. This allowed us to send the state of our chess board to the chess engine and in return get the best move to be made. Python-Chess was vital to the success of this project.


7.2.2 Command-Line-Chess
Command-Line-Chess is a lightweight command line chess game written in python. This program was simple enough to edit and rewrite portions to fit the application that we needed for our chess table.

https://github.com/marcusbuffett/command-line-chess
7.2.3 Socket.IO

Socket.IO enables real-time bidirectional event-based communication. We used this for our webserver and our iOS application. Since Socket.IO is open source, it was free for us to use, as well as the documentation to use it was amazing.

http://socket.io/

7.2.4 NRFManager for Swift

NRF Manager is a Bluetooth wrapper for the Bluetooth module that was purchased from Adafruit. This wrapper made it incredibly easy to communicate to and from the Arduino in the iOS application.

https://github.com/MichMich/nRF8001-Swift

7.2.5 Pocketsphinx

Pocketsphinx is an open source library for speech recognition that was developed by Carnegie Mellon University. Pocketsphinx allows for offline speech recognition, unlike most speech recognition software; this made it the ideal candidate for voice recognition for The Knight, allowing for users to control the board with their voice without having to connect it to the Internet. Because Pocketsphinx implements offline speech recognition, however, it is less accurate and slightly slower than most online recognition implementations.

http://cmusphinx.sourceforge.net/

7.2.6 Stockfish

Stockfish is one of the strongest chess engines in the world. Unlike most chess engines, Stockfish is open source so it is free for us to view, modify and contribute to the code base. We used Stockfish as our chess engine because it uses UCI (Universal Chess Interface) which allows us to communicate to it using our Python-Chess wrapper.

http://stockfishchess.org/

7.2.7 THE-KNIGHT (Our Repository)

This is where the team kept all of the code and handled version control.

https://github.com/DerekDY/THE-KNIGHT
REFERENCES


[5] https://sites.google.com/site/bergersprojects/reedcb/matrix


[8] https://ucichessengine.wordpress.com/


9 APPENDIX

9.1 Appendix 1 – Time

Figure 35 - Time Analysis
## 9.2 Appendix 2 – Purchases

Table 9 - Itemized List of Purchases

<table>
<thead>
<tr>
<th>Date</th>
<th>Team member</th>
<th>Description</th>
<th>Debit</th>
<th>Credit</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>9/27/15</td>
<td>All</td>
<td>Beginning Balance</td>
<td>0.00</td>
<td>500.00</td>
<td>$500.00</td>
</tr>
<tr>
<td>9/30/15</td>
<td>Paul</td>
<td>Sensor Reed Switch SPST</td>
<td>6.93</td>
<td>0.00</td>
<td>$493.07</td>
</tr>
<tr>
<td>9/30/15</td>
<td>Paul</td>
<td>Reed Switch SPST</td>
<td>8.77</td>
<td>0.00</td>
<td>$484.30</td>
</tr>
<tr>
<td>9/30/15</td>
<td>Paul</td>
<td>Diode Gen Purr 110V 200 mA DO35</td>
<td>1.84</td>
<td>0.00</td>
<td>$482.46</td>
</tr>
<tr>
<td>9/30/15</td>
<td>Curtis</td>
<td>5.6lbs DC 12V Holding Electromagnet Lift Solenoid</td>
<td>5.51</td>
<td>0.00</td>
<td>$476.95</td>
</tr>
<tr>
<td>9/30/15</td>
<td>Paul</td>
<td>Reed Switch SPST 7-30AT</td>
<td>0.62</td>
<td>0.00</td>
<td>$476.33</td>
</tr>
<tr>
<td>9/30/15</td>
<td>Paul</td>
<td>Reed Switch SPST 10-50AT</td>
<td>0.50</td>
<td>0.00</td>
<td>$475.83</td>
</tr>
<tr>
<td>9/30/15</td>
<td>Paul</td>
<td>Reed Switch 200V 10W 20-30AT</td>
<td>0.74</td>
<td>0.00</td>
<td>$475.09</td>
</tr>
<tr>
<td>9/30/15</td>
<td>Paul</td>
<td>Reed Switch Modified 200V 25-40AT</td>
<td>0.89</td>
<td>0.00</td>
<td>$474.20</td>
</tr>
<tr>
<td>9/30/15</td>
<td>Paul</td>
<td>Reed Switch 200V 10W 10-15AT</td>
<td>0.60</td>
<td>0.00</td>
<td>$473.60</td>
</tr>
<tr>
<td>9/30/15</td>
<td>Paul</td>
<td>Magnet 1/4&quot; DIA X 1/16&quot; Thick</td>
<td>0.21</td>
<td>0.00</td>
<td>$473.39</td>
</tr>
<tr>
<td>9/30/15</td>
<td>Paul</td>
<td>Magnet 1/4&quot; DIA X 1/8&quot; Thick</td>
<td>0.38</td>
<td>0.00</td>
<td>$473.01</td>
</tr>
<tr>
<td>9/30/15</td>
<td>Paul</td>
<td>Magnet 1/2&quot; DIA X 1/8&quot; Thick</td>
<td>1.28</td>
<td>0.00</td>
<td>$471.73</td>
</tr>
<tr>
<td>10/30/15</td>
<td>Paul</td>
<td>DC 24V Holding Electromagnet Lift Solenoid</td>
<td>9.98</td>
<td>0.00</td>
<td>$461.75</td>
</tr>
<tr>
<td>10/30/15</td>
<td>Paul</td>
<td>DC Switching Converter Power Supply</td>
<td>15.58</td>
<td>0.00</td>
<td>$446.17</td>
</tr>
<tr>
<td>11/7/15</td>
<td>Curtis</td>
<td>Raspberry Pi 2.0</td>
<td>38.72</td>
<td>0.00</td>
<td>$400.62</td>
</tr>
<tr>
<td>11/7/15</td>
<td>Curtis</td>
<td>EasyAcc Micro USB Charger</td>
<td>8.99</td>
<td>0.00</td>
<td>$391.63</td>
</tr>
<tr>
<td>11/7/15</td>
<td>Curtis</td>
<td>Samsung 16GB Micro SDHC</td>
<td>7.64</td>
<td>0.00</td>
<td>$383.99</td>
</tr>
<tr>
<td>11/7/15</td>
<td>Curtis</td>
<td>Adafruit: Bluetooth 4.0 USB Module (v2.1 Back-Compatible)</td>
<td>11.95</td>
<td>0.00</td>
<td>$372.04</td>
</tr>
<tr>
<td>11/7/15</td>
<td>Curtis</td>
<td>Adafruit Bluetooth 4.0 USB Module</td>
<td>11.95</td>
<td>0.00</td>
<td>$360.09</td>
</tr>
<tr>
<td>11/27/15</td>
<td>Curtis</td>
<td>Open Ended 6mm Width GT2 Belt</td>
<td>1.80</td>
<td>0.00</td>
<td>$358.29</td>
</tr>
<tr>
<td>11/27/15</td>
<td>Curtis</td>
<td>2GT 20 Tooth 6.35mm Bore Pulley</td>
<td>1.85</td>
<td>0.00</td>
<td>$356.44</td>
</tr>
<tr>
<td>11/27/15</td>
<td>Curtis</td>
<td>UM GT2 Pulley 20 Tooth 5mm or 6mm Bore</td>
<td>0.40</td>
<td>0.00</td>
<td>$355.04</td>
</tr>
<tr>
<td>11/27/15</td>
<td>Curtis</td>
<td>Idler Pulley Post 6mm</td>
<td>0.20</td>
<td>0.00</td>
<td>$354.84</td>
</tr>
<tr>
<td>11/27/15</td>
<td>Curtis</td>
<td>Timing Belt Tensioner Spring</td>
<td>1.80</td>
<td>0.00</td>
<td>$353.04</td>
</tr>
<tr>
<td>11/27/15</td>
<td>Curtis</td>
<td>Toothed clamp for open ended belt</td>
<td>0.90</td>
<td>0.00</td>
<td>$353.14</td>
</tr>
<tr>
<td>11/27/15</td>
<td>Curtis</td>
<td>KR13 Needle Roller Bearing</td>
<td>0.65</td>
<td>0.00</td>
<td>$352.49</td>
</tr>
<tr>
<td>11/27/15</td>
<td>Curtis</td>
<td>Limit Switch with Roller</td>
<td>0.65</td>
<td>0.00</td>
<td>$351.84</td>
</tr>
<tr>
<td>12/9/15</td>
<td>Curtis</td>
<td>Arduino Nano w/ usb cable * 2</td>
<td>15.98</td>
<td>0.00</td>
<td>$335.86</td>
</tr>
<tr>
<td>12/10/15</td>
<td>Curtis</td>
<td>Bluetooth Low Energy (BLE 4.0) - nRF8001</td>
<td>19.95</td>
<td>0.00</td>
<td>$315.91</td>
</tr>
<tr>
<td>2/3/16</td>
<td>Curtis</td>
<td>1&quot; x 48&quot; Dowel</td>
<td>8.44</td>
<td>0.00</td>
<td>$307.47</td>
</tr>
<tr>
<td>2/5/16</td>
<td>Curtis</td>
<td>4 x 20 FHSCS</td>
<td>14.00</td>
<td>0.00</td>
<td>$293.47</td>
</tr>
<tr>
<td>2/5/16</td>
<td>Curtis</td>
<td>4 x 35 SHCS</td>
<td>6.13</td>
<td>0.00</td>
<td>$287.34</td>
</tr>
<tr>
<td>2/5/16</td>
<td>Alexis</td>
<td>Touch Sensor and LEDs</td>
<td>4.25</td>
<td>0.00</td>
<td>$283.09</td>
</tr>
<tr>
<td>2/10/16</td>
<td>Curtis</td>
<td>6mm x 200mm rod</td>
<td>12.90</td>
<td>0.00</td>
<td>$270.19</td>
</tr>
<tr>
<td>2/23/16</td>
<td>Derek</td>
<td>LED matrix, breakout for pi, connectors, buttons, port expander</td>
<td>76.50</td>
<td>0.00</td>
<td>$193.69</td>
</tr>
<tr>
<td>2/27/16</td>
<td>Curtis</td>
<td>McMasterCarr Aluminum T-Rail</td>
<td>34.98</td>
<td>0.00</td>
<td>$158.71</td>
</tr>
<tr>
<td>3/21/16</td>
<td>Curtis</td>
<td>Amazon Arduino, USB Expander</td>
<td>13.57</td>
<td>0.00</td>
<td>$145.14</td>
</tr>
<tr>
<td>3/30/16</td>
<td>Curtis</td>
<td>Home Depot</td>
<td>60.87</td>
<td>0.00</td>
<td>$84.27</td>
</tr>
<tr>
<td>4/1/16</td>
<td>Curtis</td>
<td>Godwin Plumbing and Hardware - wood glue and dowel</td>
<td>12.90</td>
<td>0.00</td>
<td>$71.37</td>
</tr>
<tr>
<td>4/5/16</td>
<td>Curtis</td>
<td>Amazon - Perfboard w/ power rails x2 (100x160cm)</td>
<td>21.56</td>
<td>0.00</td>
<td>$49.81</td>
</tr>
<tr>
<td>4/6/16</td>
<td>Curtis</td>
<td>McMasterCarr Aluminum T-Rail</td>
<td>30.57</td>
<td>0.00</td>
<td>$19.24</td>
</tr>
<tr>
<td>4/7/16</td>
<td>Curtis</td>
<td>Home Depot - spray paint</td>
<td>13.48</td>
<td>0.00</td>
<td>$5.76</td>
</tr>
<tr>
<td>4/11/16</td>
<td>Curtis</td>
<td>Arduino Nano x2</td>
<td>14.98</td>
<td>0.00</td>
<td>-$9.22</td>
</tr>
</tbody>
</table>