Magic Chess
Haley Amason, Joshua Burbridge, Brittany Nottingham, Thong Tran
School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

Abstract—Various solutions for intelligent chess engines and speech-recognition abound, but they are not often run in unison. This project explores the possibility of combining these elements to create a unique style of gameplay. Magic Chess is a voice-controlled physical chess board that moves chess pieces automatically and supports both one- and two-person play. It utilizes the open-source engines TSCP and PocketSphinx as well as a custom-designed motor control program. The board houses a small computer, a microprocessor, two custom-designed PCBs, and two stepper motors that control a metallic X-Y stage. This paper outlines the design process associated with the project.

Index Terms—Artificial Intelligence, Electromagnets, Motors, Speech Processing

I. INTRODUCTION

The inspiration for this project came from the fictional chess board Wizards’ Chess featured multiple times within the Harry Potter fantasy book series. In Wizards’ Chess, the game is played by vocalizing commands to the board (e.g. “Knight to E5!”), and the pieces move magically to their destination. It was clear early on that a chess engine would be a necessary component of the project, as well as a module for speech recognition. It was also desired that the board support one- and two-player modes and include varying difficulty levels, as an actual chess program would. Progressively, the rest of the design fell into place after branching out from there. The pieces would be moved by an electromagnet underneath the board, which would be operated by an X-Y stage not unlike those found in machine shops and vending machines. A small computer would be necessary to run the speech recognition component and the chess engine. Additionally, it would be necessary to have a microcontroller and custom PCB to control the stepper motors responsible for the motion of the board. Thus, a high-level design for Magic Chess was created.

II. OVERVIEW

The largest physical component on the board is the X-Y stage. In order to move the pieces across the board, the group decided to use an electromagnet. The X-Y stage is what moves the magnet. The magnet is mounted onto the x-axis, on which it can move left and right. The x-axis is then mounted on 2 y-axes, on which it can move forward and backward. The result is that the magnet can now move anywhere in the X-Y plane, power up to move a piece across the board, then power down to prevent attracting other pieces.

Inside the casing for the board, the remaining hardware is mounted. This includes the motherboard and solid state drive used to run the chess engine and the speech recognition module. In reality, this small computer has enough power to be capable of far more, but the group felt it wise to leave room for upgrades in case the board required more intensive software.

Also mounted within the case is the Stellaris microprocessor, which is responsible for sending the control signal to the PCBs to move the stepper motors as well as providing a user interface via the OLED and push buttons.

Next, the 2 custom-designed PCBs are contained inside the case. Each PCB is assigned to a stepper motor and is responsible for relaying control signals issued by the Stellaris as well as providing the power for the stepper motors to move.

Finally, the power supply is contained within the case. It was designed in order to provide the correct amount of power to each hardware component within the board. The only requirement for operating the board is that it be plugged in to any standard 15A wall socket.
III. SPECIFICATIONS

- Board will be 24” x 24” x 14” or less
- Board material must allow for significant enough flow of EM waves for pieces to move
- LED Screen depicting current and previous move
- Micro ATX form factor motherboard for computer to run AI component of project
- Secondary custom PCB including microcontroller and stepper motor control hardware
- Case for the final product will be made out of either aluminum or plexi-glass to preserve a professional and put-together final product.
- Stepper motors will move fast enough to be able to complete piece movement in 3 seconds or less.
- Chess Engine must have reasonable response time; most moves by the AI should take less than 5 seconds to compute.
- Speech recognition module must be able to interpret commands quickly. This should take 3 seconds or less to avoid confusion and/or repeated commands.
  - The software must have efficient algorithm design. It should run in at least polynomial time, with search algorithms running in \(O(n \log n)\) or better.
- The AI unit will have 3 difficulty levels: Easy, Medium, and Hard, with additional look-ahead capabilities with increasing difficulty level.
- At least two modes of play including:
  - Player VS. Player
  - Player VS. AI
- Voice Recognition for at least two different human voices
- Ability to do route management for the pieces as each piece moves differently
- Ability to remember routes for 6 different pieces and identify incorrect moves
- Use of electromagnets and permanent magnets to get pieces to move without human interaction – pieces may be as much as 0.25 pounds.
- Spaces must be large enough for pieces to move among each other without colliding
- Final project must be sturdy for possible repeated use by children.

IV. HARDWARE

A. The X-Y Stage

The stage is an incredibly important component to the project because it is responsible for moving the electromagnet around the entire play area. Two X-Y stages were considered for the project. One stage could be bought pre-built but would cost a significant amount of money to purchase. The other stage was much more affordable but would require team assembly. The assembly required stage was chosen to keep the projects budget at a minimum.

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

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

Figure 2: X-Y Stage

B. The MicroATX Computer

A motherboard was required to run the AI component of the project. The project needed a motherboard that mainly was affordable and small enough in size to fit comfortably in the chess board, and light as to not add any unnecessary weight to the chess board. Also the project would require enough RAM for the CPU to run the program quickly so there would be no additional delays. Luckily, the group was donated an ASROCK AD525PV3 motherboard and enough RAM that met the demands of the AI and the
specifications of the overall design. A 64 GB SSD was purchased for the motherboard. The features of the chosen motherboard are shown in the table below.

### Table 1: Properties of the ASRock AD525PV3

<table>
<thead>
<tr>
<th>Part</th>
<th>Brand and Model Number</th>
<th>Clock Speed</th>
<th>Mem</th>
<th>Vt</th>
<th>Wt</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB</td>
<td>ASRock AD525PV3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.5</td>
<td>$0.00</td>
</tr>
<tr>
<td>CPU</td>
<td>Intel D525 (Dual-Core Atom)</td>
<td>1.8 GHz</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>RAM</td>
<td>Patriot G2 PQG316 G1600E LQK</td>
<td>-</td>
<td>8GB</td>
<td>-</td>
<td>-</td>
<td>$0.00</td>
</tr>
<tr>
<td>SSD</td>
<td>ADATA ASP600 S3-64GM-C</td>
<td>6 Gb/s</td>
<td>64GB</td>
<td>-</td>
<td>-</td>
<td>$64.99</td>
</tr>
<tr>
<td>Power Supply</td>
<td>FSP GROUP FSP220-60LE(80 )</td>
<td>-</td>
<td>-</td>
<td>230V</td>
<td>3 lbs</td>
<td>$45.99</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4.5 lbs</td>
<td>148.99</td>
<td></td>
</tr>
</tbody>
</table>

D. The PCBs

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

The motor controller IC is the Texas Instruments DRV8818. This control IC is capable of supplying 2.5 A per winding and only requires a single pulse to a STEP input to move the motor forward one step. It, like the Allegro Microsystems stepper drivers (compared in chapter 5), also has available full, half, fourth, and eighth stepping modes available by combination of two logic inputs. The TI stepper motor controller is capable of supplying the proper voltages and currents to the stepper motors the group has chosen. The table below summarizes the important aspects of the motor control chip.

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

The “Magic” in the chess piece movement is due to the interaction of the electromagnet attached to the XY stage and the permanent magnets placed at the bottom of each chess piece. In order to have the strongest attraction possible between the chess pieces and the electromagnet, only round earth magnets were considered in the design and therefore only round electromagnets were considered. DC electromagnets were chosen for their simplified use with the microcontroller and because they are more readily available than AC and consequently cheaper.

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Size (diameter x height) [inches]</th>
<th>Volts</th>
<th>Pull [lbs.]</th>
<th>Wt [oz]</th>
<th>Average cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>.75 x .62</td>
<td>12</td>
<td>4.5</td>
<td>.96</td>
<td>$34.09</td>
</tr>
<tr>
<td>B</td>
<td>.75 x 1.250</td>
<td>12</td>
<td>10</td>
<td>1.7</td>
<td>$55</td>
</tr>
<tr>
<td>C</td>
<td>1 x .719</td>
<td>12</td>
<td>10</td>
<td>1.9</td>
<td>$53</td>
</tr>
<tr>
<td>D</td>
<td>.75 x .62</td>
<td>6</td>
<td>4.5</td>
<td>.96</td>
<td>$29.24</td>
</tr>
<tr>
<td>E</td>
<td>.75 x .375</td>
<td>12</td>
<td>5.5</td>
<td>.8</td>
<td>$35.36</td>
</tr>
<tr>
<td>F</td>
<td>.75 x .62</td>
<td>6</td>
<td>6</td>
<td>.96</td>
<td>$29.24</td>
</tr>
<tr>
<td>G</td>
<td>.75 x .62</td>
<td>12</td>
<td>6</td>
<td>.96</td>
<td>$29.24</td>
</tr>
</tbody>
</table>

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

This electromagnet requires 1 watt of power and can pull 6 pounds. The housing is finished in Zinc or electroless nickel plating. The magnet comes with 18 inch long leads.

The chosen permanent magnets for the project were the ones used for the electromagnet/permanent magnet test. The permanent magnets are neodymium or rare earth magnets. They were chosen because of the immediate discovery that they were not only stronger, but smaller in size than the ferrite magnets. The properties of the permanent magnets are as follows; they are .25 inches in...
diameter and 0.0625 inches in depth. They have a surface gauss of 2163 and a pull capacity of 3.566 pounds.

**F. The Stepper Motors**

The stepper motors are used to move the X and Y axes of the stage. The two stepper motors are identical and were purchased from SparkFun Electronics with an internal product number of ROB-09238 and are manufactured by Mercury Motor with an internal product number of 42BYG011 [1]. Each has a 5mm drive shaft, which would require parts measured in the metric system (or extremely close imperial system) measurements. The motor turns 1.8 degrees each step and comes with a rated voltage of 12 volts and rated current of .33 amps. The motor’s holding torque is .169 lb-ft which is enough to move both the axes. It is also relatively light at .44 pounds.

**G. Power**

The project needed a power supply large enough to power not only the motherboard but the PCBs as well. AC/DC converter circuits were researched but found to be extremely dangerous to build in the fact that if built incorrectly they could severely damage the electrical components. Much trial and error would need to be done to design a suitable power supply. Later, wall converters were considered because they are already built and tested by the manufacturer. Ideally, a motherboard power supply would be found that could power both the motherboard and PCBs.

The motherboard power supply chosen is manufactured by FSP Technology Incorporated also known as FSP Group. Below is a table showing the important specifications of the power supply. The motherboard came with one requirement that its power supply be ErP/EuP rated. The group wanted a power supply that came with plenty of extra connectors to power the PCBs. The FSP220-60LE(80) power supply was chosen for the project’s power supply because it featured a 1-watt standby mode which is compliant with the ErP/EuP standard. Also, the supply came with enough Molex connectors PC and PD to power the PCBs. The 20 (+4) pin main connectors are important to power the motherboard. The summary of the specifications of the power supply are shown in the next table.

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

---

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

---

**V. SOFTWARE**

Many problems that needed to be handled by the chess board were complex. In particular, creating a program intelligent enough to defeat the average human at a game of chess is a task usually tackled by a team of programmers over the course of a few years. Additionally, creating an agile and accurate speech recognition program was a task that could have comprised an entire design project in and of itself. Therefore, it was necessary to use open-source software for both the chess engine and speech recognition module. These were customized via the provided APIs to meet the requirements of the project.

The development environment for the software was primarily Code Composer 5.3 for anything related to the Stellaris and Visual Studio 2012 otherwise.

The diagram below outlines the structure of a single turn in Magic Chess as well as the general role each component plays.

---

**Figure 5: High-Level Software Block Diagram**
A. Pocketsphinx

Speech recognition works by recording the voice audio and then breaking it into utterances which are separated by pauses. Then the speech recognition library would compare the utterances with each possible combinations of words using a dictionary and language model. However, speech recognition in and of itself is not completely accurate. Inherently it is inaccurate.

For the project, the chess engine works by using the chess coordinate system. This would require the speech recognition library to be able to understand, interpret, and input the commands into the chess engine. Since the number of commands is limited to only a few letters and numbers (e.g. “A5B6”) this should improve the accuracy of the speech recognition library.

CMU Sphinx is used as the voice recognition library for the project. CMU Sphinx comes in two major versions, Sphinx 4 and Pocketsphinx. Pocketsphinx was the version chosen for this project due to its purpose of being made for smaller portable and embedded systems.

Pocketsphinx uses both a dictionary and a language model for its speech recognition. The English acoustic model that was packaged with Pocketsphinx contains over 18,000 vocabulary terms. This was unnecessary as it not only contained more vocabulary than needed but it also caused large inaccuracies with the voice commands as Pocketsphinx would search each of the word combinations that matches the utterances from the user’s voice. The solution was to simply create an acoustic model along with a dictionary and language model specifically for this project. The custom English language model was created only using letters A through H and numbers 1 through 8 as these would only be the commands required to play the game using chess coordinate system. This narrow vocabulary vastly improves accuracy of the speech recognition as Pocketsphinx no longer needs to search through the entire English library and try to match the correct words to the voice command. This also allows another side benefit where the response time of Pocketsphinx is vastly improved.

B. The Chess Engine

Once Pocketsphinx has finished parsing the user’s voice command, the coordinates are sent to the chess engine, which updates the representation of the board within the computer. However, it is first necessary to explain how the chess engine was chosen. After establishing that the chess engine should be open-source, the next step was to figure out exactly which chess engine would be most suitable. In order to achieve this, 9 different engines with varying capabilities (GNU Chess, Tom Kerrigan’s Simple Chess Program, Fruit, Gerbil, Crafty, Stockfish, Glaurung, Micromax, and Faile) were tested against a strict set of standards. These standards included robustness, adaptability, availability of documentation and readability of code, and use of processing resources. Among open-source chess engines, there are two primary interface formats: WinBoard/XBoard and UCI. Because the set of chess engines under test fell into both categories, it was necessary to use multiple open-source GUIs to test them. These GUIs are built to allow any chess engine configured to use the standard interface to be plugged in and tested. This allows developers to worry only about the core chess rules and not have to create a new GUI for every new engine. Fortunately, these GUIs have benchmarking capabilities, which means it was easy to test things such total memory being used, and percentage of CPU used by the chess process. These results corresponded to the benchmarks given by the system itself, so the performance of the engines could be monitored to an acceptable degree of accuracy.

Ultimately Tom Kerrigan’s Simple Chess Program (TSCP) was chosen as the best candidate for the project. This was due in part to its very low memory usage, even at a high difficulty level (~2MB). It implemented all the necessary design features such as single and multiplayer support, varying difficulty levels, and even the ability to choose which color to play. The source code was also very simple compared to the other engines, with the total length being about 25% of the average length. TSCP was used with Tom Kerrigan’s explicit permission.

When TSCP is configured for 2 players, it receives a stream of commands from Pocketsphinx and alternates between the two to perform each player’s move on the board. When TSCP is configured for one player, it receives a command from Pocketsphinx, updates this move on the board, make its own move, then proceeds to wait for another command. However, the move must still be implemented physically on the board. Therefore, after every turn, TSCP sends the coordinates of the move to the Motor Controller, which takes care of the rest.

C. The Motor Controller

The Motor Controller receives coordinates from TSCP and uses them to calculate how many pulses to send to the stepper motors in order to move the magnet into position. This program runs on the Stellaris board, which is where the square wave for the motors is generated. Because stepper motors are highly accurate, it is easy to calculate the number of steps required to move the magnet from point A to point B. In general, the number of microsteps to move between 2 squares on the chess board is saved as a constant by which the difference between the columns and rows of point A and point B is multiplied. Once the
number of microsteps has been calculated, the program utilizes the pulse width modulation capabilities of the Stellaris to send an exact number of cycles to each stepper motor.

In addition to controlling the motors, this portion of the software also sends the controls to enable and disable the motors, set the direction, and turn the electromagnet on and off. In a general turn, the motor controller will move the magnet into place, activate it, move it to the destination (pulling the piece along with it), and then deactivate the magnet. This is how the moves are “animated” on the surface.

D. Arch Linux

Since the project is using a microATX motherboard, an operating system would be required in order to have the system running both the chess engine and the voice recognition library. Windows 7 and Windows Vista were considered, along with different flavors of Linux, namely Arch, Debian, Ubuntu, and Fedora. Initially, Debian was chosen as the operating system. This is because it runs on an old Linux kernel, which means it is normally quite stable. However, Debian eventually caused problems because of its requirement to have an internet connection during installation. To configure this connection, Debian had to first be installed. But to install Debian, the network card had to be configured. Because of this catch-22 situation, we decided to use Arch Linux. Arch Linux is a GNU/Linux distribution that has a philosophy of Keep It Simple Stupid. This is achieved by allowing the user to set up the operating system in any way the user wishes. For this project Arch Linux was installed with only the Linux kernel plus the minimal amount of software required to run the operating system. In addition, the voice recognition library and chess engine were also installed as required. This allows the system to remain lightweight and efficient. In addition, this also minimizes the possibility of the system crashing as there are fewer software components running.

VI. INTEGRATION AND TESTING

Integration was a challenging aspect of this project. The design includes various software and hardware components, many of which were not designed to work with each other. In some cases, common protocols were used such as the StellarusWare library written for the GPIOs and UART for communication. In other cases, namely Pocketsphinx and TSCP, integration was more difficult.

In the research phase of the project, software was tested on personal computers to ensure proper functionality and to aid in understanding its operation for development purposes. Final testing of the TSCP engine continued on a personal computer. This was done simply to measure the amount of RAM used and measure the average response time given similar hardware specifications. The voice recognition software was also initially tested on a personal computer for proof of concept of the smaller dictionary. The amount of testing that can be done for the magnetic controller on a personal computer is limited, as the physical performance of the motors’ functions needed to be measured.

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

In the final testing phase, the board was fully built, and a user is now able to play a full game as described in the previous sections. Because it has been verified that users can play full games on the board, both against the computer and against each other, it is assumed that the system is operating completely functionally. Hopefully no further troubleshooting will be necessary.

VII. CONCLUSION

Magic Chess combines various technological components to create a unique and exciting gameplay experience. It is our hope that others enjoy playing it as much as we do.

ACKNOWLEDGEMENTS

We would like to acknowledge SoarTech for their generous sponsorship of this project. We would also like to acknowledge Dr Samuel Richie and Josh Haley for their guidance and Tom Kerrigan, Carl S., and Garrett Chapman for their invaluable contributions.
BIOGRAPHY

**Haley Jane Amason** is a senior at the University of Central Florida and will be graduating with a Bachelor’s of Science in Electrical Engineering in May of 2013. She currently works at Universal Studios Orlando on the Reliability Centered Maintenance team within Technical Services as their Engineering Coordinator. Haley has been working with RCM for almost two years and looks forward to continuing a career in Reliability/Quality Engineering.

**Joshua Burbridge** is a senior in Computer Engineering at the University of Central Florida, with minors in Mathematics and Secure Computing and Networking. He will be graduating with a B.S. in May 2013. His immediate future plans are to begin the Computer Engineering MS program at UCF. His research interests include Algorithm Design, Bioinformatics, and Computational Complexity.

**Brittany Nottingham** is graduating in May 2013 with a Bachelors of Science in Electrical Engineering. Her interest in electrical engineering began in high school and she has chosen to specialize in power systems and hopes to obtain a job in that field. Brittany expects to stay in Orlando upon graduation and plans to work full-time at ABB while completing a graduate program part-time.

**Thong Tran** is currently a senior at University of Central Florida and will be graduating with a Bachelor of Science in Computer Engineering. He hopes to pursue a career in game development, and obtain a Masters in Computer Engineering in the future.

REFERENCES


