Autonomous Robotic Chess Board

Alec Adair, Kenneth Bonar, Kara Douville, Austin Payne, *Department of Electrical and Computer Engineering, College of Engineering, University of Utah*

***Abstract*—Chess can either be played either on a physical board or on a computer. However, few attempts have been made at melding the traditional feel of physical chess and the software functionality offered by virtual chess. Our senior project provided users and their opponents with the flexibility to pick their own environment. This was accomplished by building a robotic chess board where pieces were moved using an electromagnet and magnetic chess pieces. Users were then able to interact with the chess robot by moving their own pieces and compete against an artificial intelligence, where the opposing pieces moved autonomously.**

***Index Terms*—chess, microcontroller, electromagnet, stepper motor, hall effect, robotics, WiFi.**

I. Introduction

A. Background

Chess is a two player strategy game that dates back to the 6th century with origins in India and Spain [1]. Centuries later, Claude Shannon wrote the first important paper concerning methods for programming chess-play in 1949 [2]. Other great thinkers, such as Alan Turing, have also published papers on chess playing algorithms. He implemented one of the first chess-playing algorithms, dubbed Turochamp, to demonstrate the capabilities of contemporary computers [3].

Since 1949 chess algorithms have continued to evolve, even to the point of beating chess Grandmasters [4]. One such chess engine is Stockfish, an open source project that is consistently ranked as "the strongest open source chess engine in the world" [4], [5]. With the ubiquity of chess engines and the Internet, people can now play chess on a range of devices and compete against opponents from all over the world.

For our senior project we built an Internet-connected, autonomous robotic chess board that allowed an individual to play chess either virtually or physically against another player located anywhere in the world. Additionally, we used the Stockfish chess engine to allow players to hone their skills against an artificial intelligence (AI).

Being fans of fantasy, inspiration for the functionality of the board was taken from the "Harry Potter" series. On our board, the opponenet's pieces moved autonomously, similar to "Wizard's Chess" from the popular novel Harry Potter and the Sorcerer's Stone [6]. Our board maintained the tactile feel of chess, while adding an element of surprise and smooth automation.

After deciding on our project at the beginning of the year, we subsequently researched other solutions and products. Currently, another board is being created and crowd sourced, called Square Off [7]. While Square Off is similar to our implementation, it lacks certain elements that we felt were important, such as a generic interface for connecting a variety of chess clients together. We were able to implement all of these features and set our chess board apart from those currently in production.

B. Motivation

The motivation for our team to do a game-based project was twofold: to showcase our skills in both computer and electrical systems and to provide an entertaining, rewarding, and polished product for demonstration. The final chess robot incorporated several electrical systems focused on moving and coordinating pieces. The software skills that we have developed played an integral role in the implementation of the various interfaces that were developed for game play. Several other examples of chess machines have been built, but few incorporated the software sophistication and cross-platform development that we achieved. In the end, the board enabled users to play chess against other players or an artificial player, yet maintained a traditional kinetic experience.

C. Design Approach

At a high level, we built 1) an HTTP chess server with artificial intelligence capabilities and 2) an Internet connected chess robot that could connect to the chess server. The chess server was designed to be device agnostic so that a variety of clients could connect and play, including our chess robot. It utilized an open source chess engine called Stockfish to provide artificial intelligence that players could compete against. The chess server was also able to manage multiple games and enforce all the rules of chess so that resource constrained clients, such as our chess robot, could play even with limited computing power.

The chess robot was constructed using a stepper motor driven, two axis machine underneath a chess board. The pieces of the chess board were embedded with magnets so that the machine could move the pieces using an electromagnet. In addition, the chess board contained a magnetics sensor in the center of each square; this allowed a user to physically move their pieces and have the board determine which move was made. The chess robot also connected to a standard WiFi network so that the board could connect to the chess server and start new chess games, relay player moves, and receive opponent moves and update the positions of the chess pieces.

We used an Agile project development approach to design the chess robot. We had an initial design and a relatively good idea of how to achieve it, but throughout the process we iteratively tested hardware and software modules. We constructed and verified the hardware components of the robot first, then subsequently added and validated the various software modules. Finally, we implemented the chess server and completed end-to-end integration testing by playing games on the chess board. Our approach also incorporated risk assessment and management, tasks with the highest difficulty were given the highest priority (see Section V-B). We also implemented integration testing at every phase of the project to ensure that each component worked together.

D. Demonstration Expectations

At the final project demonstration, the user was be able to play either single player or multi-player chess, in four different game modes:

- No player: A.I. on board vs A.I. on board
- One player: user at board vs A.I.
- One player: user on web vs A.I.
- Two players: user on web vs user on web

We chose to demonstrate user at board vs AI and AI vs AI, as these two games modes were most conducive to demonstrating the chess robot.

II. Project Tasks

A. Hall Sensor Array

Initially, we planned on using an array of reed switches to keep track of chess piece movement on the board, with one reed switch position at the center of each board square. However, we ultimately chose to use an array of hall sensors. Hall sensors are able to detect DC magnetic fields and output an analog voltage in proportional to the field applied to the sensor and provided better biasing and magnetic field measurement.

Each index in the array correlated to a square on the chess board. Because chess is deterministic, it was sufficient to know the initial state of the game and track piece movement throughout the duration of the game. Each chess piece was embedded with a magnet and the hall sensors raised or lowered the output voltage based on the proximity of a piece, thereby signaling when a piece had moved.

This part of the project was completed by Alec, who designed the entire analog system as well as the 64 to 1 multiplexer needed to take readings from the hall sensors. This task was comprised of subtasks:

- Implement firmware polling of the array
- Determine optimal magnet strength
- Create 3x3 proof of polling concept (6 General Purpose Input/Output (GPIO) pins)
- Design analog system
- Design and construct a 64 to 1 multiplexer system
- Establish hall sensor threshold
- Test hall sensors and multiplexer design
- Determine piece movement based on board states

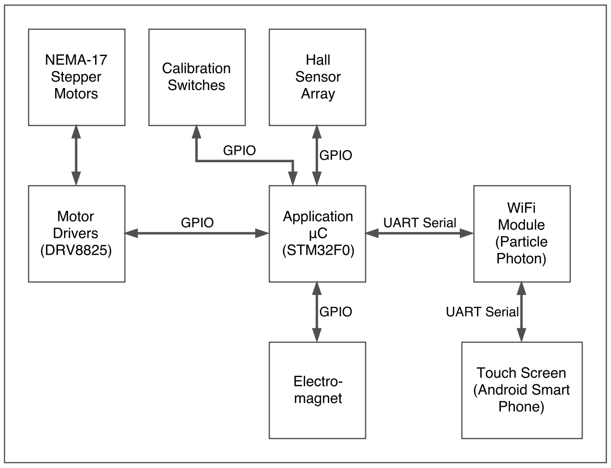

Fig. 1. High Level Block Diagram

B. Mechanics

Computer numerical control (CNC) is the automation of machine tools by means of computers executing preprogrammed sequences [8]. The chessboard housed a two-axis CNC machine that moved an electromagnet beneath the chess board. Each chess piece contained a small neodymium magnet such that the force between the piece and the electromagnet was enough to move the chess piece, but small enough so as not to alter the other pieces' positions. The electromagnet interfaced with a central microcontroller via GPIO. The CNC machine was driven by stepper motors that interfaced with DRV8825 motor drivers, as seen in Fig. 1, and controlled the movement of the electromagnet. Printing all of the parts took approximately 30 hours and was completed by Ken and Kara. The assembly of the CNC took 10 hours and was completed by the group. Austin implemented the software driver to control the electromagnetic switch and stepper motors. This task was comprised of the following subtasks:

- Print 3D-printed parts
- Order steel tubing and electronics
- Assemble CNC, seen in Fig. 2, from V1 Engineering [9]
- Cut acrylic board
- Design electromagnetic holder
- Implement electromagnetic switch
- Print chess pieces
- Wire motors and electromagnet

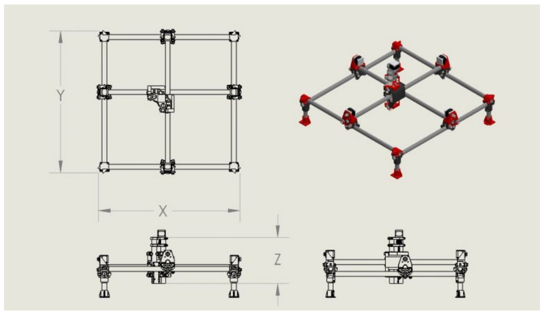

Fig. 2. Diagram of CNC machine used to move chess pieces [10]

C. Robotic Stepper Motor Interface

Having a precise stepper motor driver and interface was key to making sure the chess pieces moved to accurately and didn't interfere with other pieces in transit. In order to simplify path planning and avoid collisions, the base of the chess pieces were designed to be half the width of a chess square. When a piece was moved, it was offset onto the lines dividing squares and moved taxi-cab style to its destination. The motor driver interfaced with the microcontroller as seen in Fig. 1. The stepper motor firmware drivers were able to move with millimeter accuracy, accumulating less than 1% error after 50 linear movements. In addition, the stepper firmware compensated for minor errors in the chess board construction, such as off center hall sensors and unequal square width. Austin designed and implemented the stepper motor firmware drivers as well as wired the stepper motor drivers with help from Alec and Ken. This projects subtasks included:

- Write robust and accurate motor drivers
  - Implement Pulse Width Modulation (PWM) control
  - Account for error in millimeter to step conversion
  - Calibration routine to initialize state of CNC
- Wire DRV8825 to steppers and microcontroller

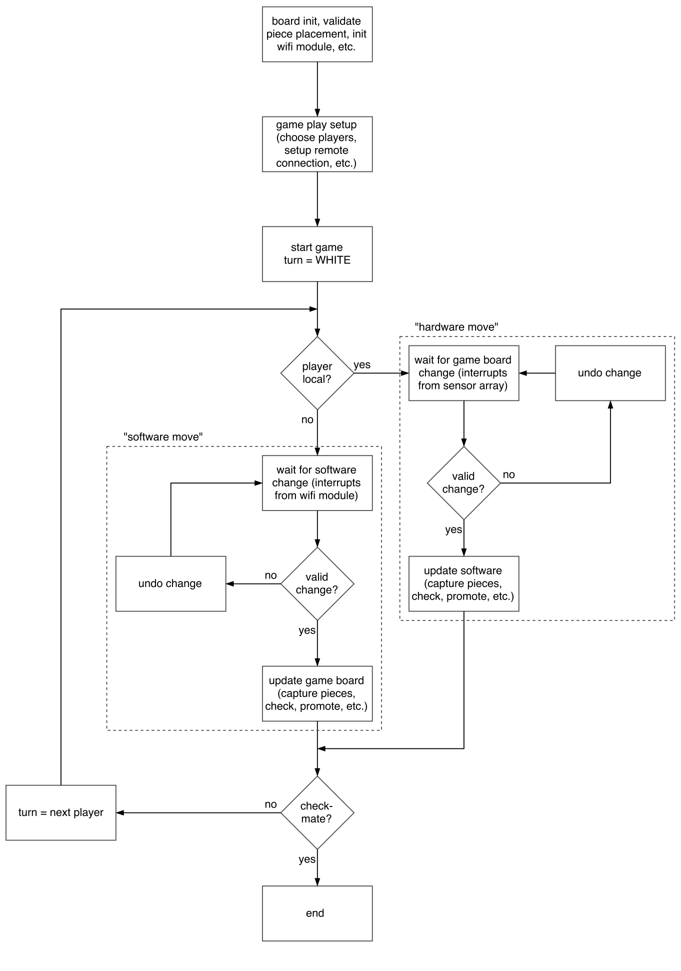
board init, validate piece placement, init wifi module, etc.
game play setup (choose players, setup remote connection, etc.)
start game turn = WHITE
player local?
yes
no
"hardware move"
wait for game board change (interrupts from sensor array)
undo change
valid change?
no
yes
update software (capture pieces, check, promote, etc.)
"software move"
wait for software change (interrupts from wifi module)
undo change
no
valid change?
yes
update game board (capture pieces, check, promote, etc.)
turn = next player
no
check-mate?
yes
end

Fig. 3. High Level Algorithmic State Machine (ASM)

D. Interface Integration

In an effort to avoid one mass integration, integrated the different interfaces incrementally. Our project relied on the successful integration between each major task. The interface between different elements can be seen in Fig. 1. This task consists of several subtasks:

- Design WiFi module to application controller UART interface
- Design motor control to MCU strategy

E. Network and Server Architecture

Our network architecture consisted of a simple server-client architecture built on top of the HTTP protocol. Using HTTP facilitated a device agnostic architecture and permitted the rapid development of both the embedded firmware client and a web client. Game state was maintained in the server using an open source JavaScript library called chess.js. In addition, we used a strong, open source chess engine called Stockfish to add artificial intelligence to the chess server. The chess game logic was then exposed as a Representational State Transfer (REST) API using the standard HTTP verbs GET and POST. GET was used to fetch games, player moves, and ask the server to compute the best move using Stockfish. POST was used create new games and move pieces on the board.

Austin designed and implemented the chess server as well as the HTTP client on the chess robots WiFi module. Kara implemented the web client for virtual chess games. The networking portion of the project included the subtasks listed below.

- Design the network protocol
- Implement HTTP server and client logic on WiFi Module
- Design WiFi to application controller UART driver
- Implement HTTP web application

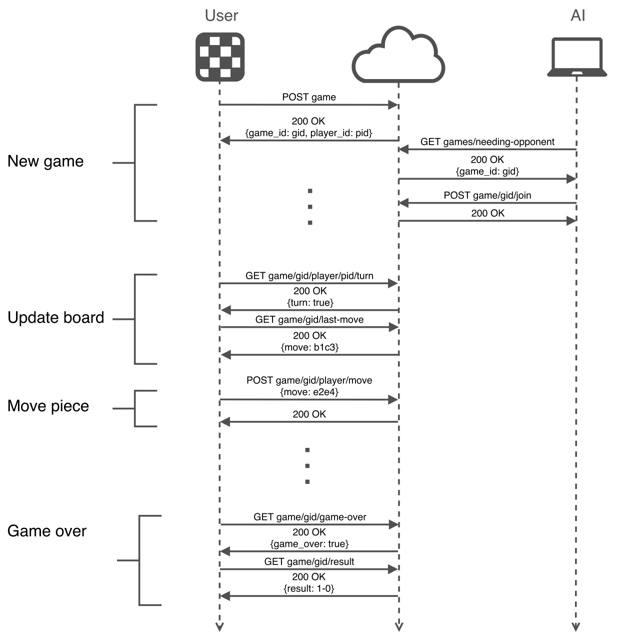

Fig. 4. Example Network Interactions

F. Web Application

The web application is our graphic user interface for the server. This web application allowed a user to create a new game, join a pending game, play against the AI, or observe a game in progress. The game being played was abstracted from the board so a user could play against a user on the board or against another player on teh web applciation.

The server that kept track of the game doubled as our web server - it simultaneously served HTML while updating current chess games. This allowed us certain development efficiencies, as we didn't have to communicate from game-server to web-server. The web-page shows valid moves to the user as well as a countdown clock. When the clock expires, the client communicates the loss to the server, and displays the winner. We used a thorough open-source chess project to render the board and pieces.

Finally, the application was successfully deployed via Heroku, a cloud application platform. The application is registred under the domain name "www.theturk.us," named after the fake chess-playing machine constructed in the late 18th century.

G. Stretch Goals

At the beginning of the summer, we had several stretch goals we had hoped to accomplish. By demonstration day, we were able to implement several of the additional features. Instead of an small LCD screen with interactive buttons, we wanted to have a capacitive touch screen interface. We accomplished this by building an Android application that interacted with the board via serial communication with the WiFi module.

III. Specific Task Interface

A. Hardware-Software Interface

The application controller acted as a central controller of the robotic chess board. It acted on specific hardware interrupts, made decisions, and controlled external hardware. The following interfaces handled hardware communications:

- WiFi: UART
- Sensor Array: 16 General Purpose Input/Output (GPIO) Array
- Motor Driver: 4 GPIO per axis
- Electromagnet: Power MOSFET and 1 GPIO

Both the Android user interface and the WiFi module communicated with via UART. We designed an ASCII based serial protocol consisting of a command number following by comma separated parameters in order to trigger certain actions on the disparate controllers. For instance, when a user started a new game on the Android phone, this triggered a serial command directed to the WiFi module that created a new game on the chess server and started the chess robot calibration sequence.

The sensor array acted as input to the controller to indicate if each chess square is occupied. To avoid using large amounts of GPIO pins, we used a sensor array with complex hardware multiplexing. This effectively reduced the number of GPIO pins from 64 to 1.

B. User Interface

A screen-based interface, using an Android phone, was incorporated so that the user could directly interact with the game. The touchscreen was used to start the game, end a players turn after a piece was moved, and perform more complex actions such as capturing opponent pieces and castling. This touchscreen utilized our serial communication protocol to send and receive serial commands via the USB port on an Android phone. A basic USB-TTL adapter was used, as well as a USB serial library from the Android API. This allowed the user to press buttons on the screen that triggered serial output to the WiFi module.

IV. Testing and Integration Strategy

Our testing for each section involved validation and testing of each major step as it was completed. We first validated the hardware, mainly the driver and motor function as most of our risk lay in this task. Next, we tested the sensor array and movement hardware. The integration of the different communication protocols provided a significant challenge. Final validation and testing of the game consisted of playing through the different game scenarios and debugging each one.

V. Group Management

A. Team Roles

Our team made an effort to share all responsibilities. Each team member was in charge of owning a main component of the project and being the primary motivator for ensuring their component was completed on time.

- Alec - Hally array, magnetics, and analog hardware
- Austin - Embedded firmware, networking, and software architecture
- Ken - Axis hardware and Android user interface
- Kara - Web client

B. Communication Plan

Our team established a set of weekly meetings to ensure the proper amount of time was allocated to the project. Every Tuesday and Thursday afternoon was reserved for working on the chess board. Our meetings were structured loosely as a scrum, an Agile framework for completing complex projects [11]. During these meetings, each team member described what they accomplished over the past week and then detailed what they hoped to execute in the upcoming week.

Outside of our weekly meetings, we used Basecamp, a web-based project management tool, to facilitate virtual group communication. We created distinct message boards that allowed us to share specific information with those who needed it, as well as archive information needed at a later date.

VI. Risk Assessment

A. Motor Control

- Risk: high
- Possible issues: trouble with calibrating motor position, mathematics to determine motor position relative to board position, troubleshooting motor drivers

- Mitigation: Use open-source stepper motor driver from RepRap [12]
- Target Date: Verify that motor drivers are functioning with precision and calibrated by May 31st

B. Reed Switch Array

- Risk: medium
- Possible issues: can't get array working, tolerances is too tight
- Mitigation: use RFID tags or Hall sensors
- Target Date: Verify that all readouts are accurate on 8x8 scale by May 31st

C. Parts Sourcing

- Risk: medium
- Possible issues: 3D printing parts could fail, time to reprint, parts are not high enough quality to support CNC frame, motors, and chess board
- Mitigation: purchase parts from a professional company that prints custom high quality 3D parts
- Target Date: Completed

D. Application Software

- Risk: medium
- Possible issues: chess algorithm is difficult to implement and application can't be built in time
- Mitigation: implement Dream Chess, an open-source chess game [13]
- Target Date: Verify that is is possible to run our own chess application off of the microcontroller by August 31st

VII. Bill of Materials

A. BOM

- Particle Photon
- ARM-based Cortex-M7
- Reed Sensors
- Acrylic Board
- CNC 3D Printed Parts [9]
- RAMPS 1.4 Shield
- DRV8825 Drivers and Heat Sinks
- 12v 5A Power Supply
- GT2 Belt (4M = 24 x 24)
- GT2 16T Pulley
- 608 Bearings (2-RS, Z, ZZ)
- 3/4 Steel Tubing (23.5mm OD)
- NEMA-17 Stepper Motors
- Resistors - 100 ohm
- Wiring Harness
- Zip Ties
- PLA Filament
- Spindle: Dewalt 660 (600 W)
- 4' x 4' Plywood Sheet

B. Other Resources and Mentors

- Jon Davies
- Lassonde Center

C. Vendor List

- Amazon
- Ebay
- Home Depot
- Speedy Metals
- totalElement
- Particle

VIII. Conclusion

A. Final Status

The final project presented on demo day was very much the product we envisioned. We essentially implemented all of the features that we had originally planned, as well as our stretch goal of interfacing an LCD screen for user input. We were able to achieve our project's end goal of having a user sit at the robotic chess board and play against an AI. The project was ultimately demonstrated by guests interacting with the machine as well as watching AI vs AI. Our definition for success was a user playing an entire game through one of the game modes without having to correct the board (such as picking up a piece that was accidentally knocked over). We also desired an aesthetically pleasing final project. We feel that we achieved all of our base goals and delivered a functioning robust autonomous chess robot prototype.

Regarding the schedule, we were relatively on track through the Fall semester, mainly assembling and implementing the necessary modules in our time-frame. While the modules were implemented in a timely manner, they were possibly not tested as thoroughly as they could have been.

B. Summary

Over the past year we have learned several important lessons concerning management and engineering. As far as management, we learned that it is much easier to plan a project than to implement one. We assumed that by the end of the spring and summer we would have had more tasks completed than we did. We also underestimated the amount of time it would take to finish each task. This project has been an excellent lesson in approximating the effort of a task.

In addition, we have found that no matter how well you plan, unexpected problems occur. Engineering requires flexibility and creativity. There is sometimes a difference between the engineering theory that we learn in school and the practical application of the theory. While some problems we have encountered have required a brand new solutions, others have preexisting solutions that we were able to take advantage of, instead of having to "re-invent the wheel".

Our milestones had a number of complex dependencies that were carefully considered before hardware was ordered. We took the approach of deciding what would be necessary for a minimum viable product. Next, we scheduled each task in priority of which would take the largest amount of time. The sensing array and chess board mechanics are co-dependent because they govern the physical movement of the pieces (which is at the core of our project). Additionally, the stepper motor interface and the mechanics are co-dependent because moves must be dictated to the motors themselves. The application software and web-application provide a way for users to remotely use the board. Finally, we had stretch goals that we were able to achieve, but the finished product did not depend on them.

Described in our risk assessment is the level and scheduling that we used to order tasks we thought would pose the largest risks. The motor control had the highest potential to adversely affect our project, and turned low when we verified that the motors were correctly moving. The Switch Array risk was mitigated when we had validated each sensor with input, and attached to the board. Parts Sourcing was completed when we had the parts we need, and had potentially redundant pieces for those prone for failure. The application software risk was mitigated when we exhaustively tested each game play mode.

At a time when everything, from games to human interactions, is moving into a digital format, our board brought back the feeling of being physically involved in playing a game. Our board not only felt like playing on a traditional chess set, but added a set of features that enhanced the user experience, whether played on a digital device, or at a chess board sitting in a chair across from an opponent.

References

[1] H. J. R. Murray, *A History of Chess*. Clarendon Press, 1913.

[2] R. D. Greenblatt, D. E. Eastlake III, and S. D. Crocker, "The Greenblatt Chess Program," in *Fall Joint Computer Conference*. ACM, November 1967, pp. 801–810.

[3] B. J. Copeland, "The Modern History of Computing," in *The Stanford Encyclopedia of Philosophy*, fall 2008 ed., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2008.

[4] (2017, April) CCRL 40/40. [Online]. Available: http://www.computerchess.org.uk/ccrl/4040/

[5] T. Romstad, M. Costalba, and J. Kiiski. (2017) Stockfish. [Online]. Available: https://stockfishchess.org/

[6] J. K. Rowling, *Harry Potter and the Philosopher's Stone*. London: Bloomsbury Publishing, June 1997, vol. 1.

[7] Square Off - Worlds Smartest Chess Board. [Online]. Available: https://www.indiegogo.com/projects/square-off-world-s-smartest-chess-board--2#/

[8] ThomasNet. (2017) More About CNC Machining. [Online]. Available: http://www.thomasnet.com/about/cnc-machining-45330503.html

[9] V. Engineering. (2017) Mostly Printed CNC Machine. [Online]. Available: https://www.vicious1.com/assembly/

[10] ——. (2017) MPCNC Assembly. [Online]. Available: https://www.vicious1.com/assembly/machine-size/

[11] Learn About Scrum. [Online]. Available: https://www.scrumalliance.org/why-scrum

[12] A. Bowyer. (2017) Stepper Motor Driver. [Online]. Available: http://reprap.org/wiki/Stepper_motor_driver

[13] W. van Niftrik. (2017) Dream Chess. [Online]. Available: http://dreamchess.org/