Tartan's Gambit

Design Review

Authors: Lillie Widmayer, Luis Ortega, Juan Cortes: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—A system capable of creating a fully remote chess playing experience. Users are able to interact with either the custom board system, or the web app, and have game status updates be received by a remote opponent. The board setup automatically moves pieces to update the game status. Games can be played from the start, or from custom scenarios for practice. Provides a more covid-friendly and customizable chess playing experience than previous solutions.**

***Index Terms*—3D Printing, Arduino, Chess, CNC, Computer Vision, Django, RaspberryPi, Remote Interaction**

I. Introduction

Life during a pandemic can quickly become boring, and although online renditions of popular board games, and even novel versions of such games, have become ubiquitous and instantly accessible, online game play just isn't the same as playing in real life. The Tartan's Gambit is a physical and digital chess application intended to provide the user with an authentic physical chess playing experience without having to be in physical proximity to their opponent. Additionally, the Tartan's Gambit can be utilized by professional chess players to practice particular scenarios. The Tartan's Gambit is designed to help quell the quarantine boredom by immersing the user in an authentic physical chess playing experience without having to leave their home.

To ensure an enjoyable user experience, the Tartan's Gambit must have a response time of 10 seconds or less between a player making a move on the virtual board and that move being reflected on the physical board and vice versa. The gantry system must not interfere with surrounding chess pieces when it is moving a piece and must be able to pick up and place any piece on any tile of the board.

II. Design Requirements

The application area of the Tartan's Gambit is a game of chess. Although competitive games of chess are timed, individual moves are often meticulously thought out by each player meaning that there is rarely a rapid succession of moves in a short time period. Therefore, a maximum reaction time of 10 seconds for our system is appropriate. When a game of chess is played, pieces are not knocked over unless a player is surrendering his/her king, so naturally another requirement of the gantry system is that it does not knock over any pieces when making a move. The grabber on the gantry system must also not drop the piece it is moving part way through the move. Because the gantry system must receive and execute a move within 10 seconds, the grabber must be able to firmly hold a piece for 10 seconds. The range of the grabber must cover the entire board and grab any piece. The components of the system should also be neatly organized and overall aesthetically pleasing to ensure positive user interaction.

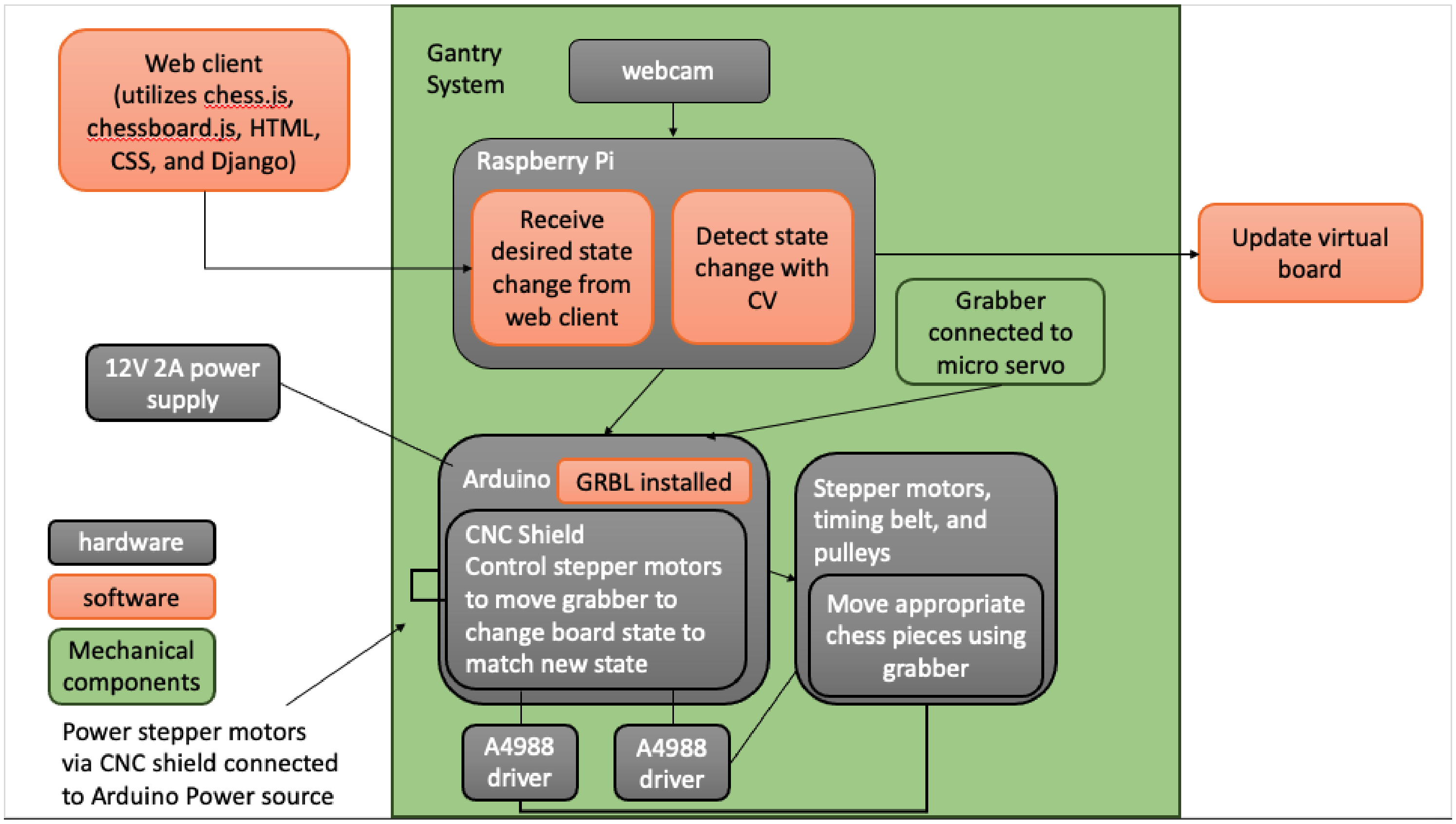

Web client (utilizes chess.js, chessboard.js, HTML, CSS, and Django)
Gantry System
webcam
Raspberry Pi
Receive desired state change from web client
Detect state change with CV
Update virtual board
Grabber connected to micro servo
12V 2A power supply
Arduino
GRBL installed
CNC Shield Control stepper motors to move grabber to change board state to match new state
Stepper motors, timing belt, and pulleys
Move appropriate chess pieces using grabber
hardware
software
Mechanical components
Power stepper motors via CNC shield connected to Arduino Power source
A4988 driver
A4988 driver

Fig. 1. Block diagram of the overall system

III. Architecture and/or Principle of Operation

The Tartan's Gambit system can be separated into 2 primary components. The first is the physical gantry system that will be interacting with the chess board and pieces. The second is the web client interface that will allow a remote user to control the gantry and play a remote game of chess with the opponent who has the physical board.

The gantry system is where all hardware and mechanical components come into play. The x and y motion of the grabber is controlled by an Arduino fitted with a CNC shield and A4988 drivers. The Raspberry Pi will receive a packet of information from the web client and will send this data to the Arduino in g-code format. The Arduino will have GRBL installed which is a firmware that will allow the Arduino to receive g-code commands. We are confident that the movement of the grabber will be very accurate since the movement mechanism is the same as a 3D printer which can print objects in great detail.The mechanical design, without the electrical components or timing belt pictured is shown in Figure 2.

The web interface will be hosted using EC2 via Amazon Web Services. The web application features an intuitive user interface with meaningful visuals and feedback messages to facilitate seamless gameplay. The player on the web application side of the game will establish a wireless connection to the Raspberry Pi on the gantry via Django Channels, and after this connection is established, he/she will be able to begin a game with the opponent on the physical board side of the system. In order to ensure a connection is properly established before gameplay begins, the web client player will always make the first move.

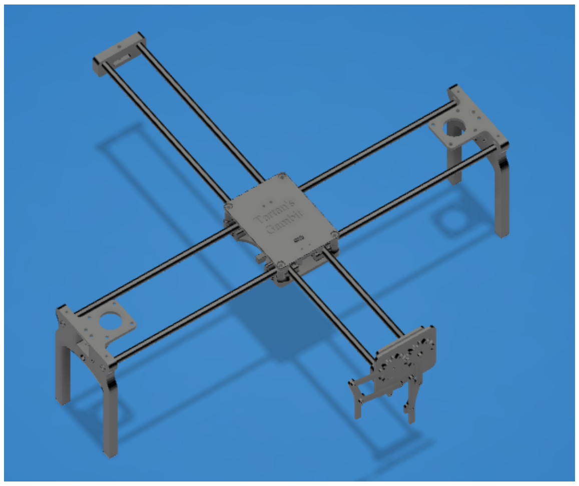


Figure 2. Gantry System

IV. Design Trade Studies

Piece Detection

RFID

There were many approaches that we considered for detection. The first design used RFID tags on each piece with readers on each square. This option was costly because we needed RFID tags for sixty-four pieces. The RFID detection could work with thin boards or if the board were modified, however we wanted to avoid modifying the board since one of our requirements is accessibility. We also considered this a risky approach because of the field overlap due to the proximity of squares and pieces.

Contact Plates

A simpler alternative that we considered was contact plates on each square. This technology makes the detection aspect straightforward as piece location becomes binary. However, the downsides to this approach is needing to modify the board and pieces. For contact plates to plates would be required on the bottom of each piece and on top of each square. Similarly, to RFID we wanted the ability to use any board. We also considered aesthetics within our design because of the user experience. We want the board and pieces to be familiar to the user for the best experience.

Computer Vision

We decided to use computer vision to avoid modifying boards and pieces. Computer vision is more complex than the contact plate approach but because of existing platforms such as OpenCV and BoofCV we do not need to implement algorithms. We knew that we could use edge detection and Hough polar algorithms to track lines and circles, squares, and pieces respectively so we would not need object detection. This is due to the user inputting the initial state of the game so we can track piece location based on what square becomes vacant and occupied.

For our computer vision library, we chose between OpenCV and BoofCV. Both libraries are widely used and have different approaches for algorithms. BoofCV's implementation of Hough Polar algorithms is faster than OpenCV's, however OpenCV has faster implementations of Gaussian blur and edge detection [boofcv.org]. This led us to use OpenCV as our main computer vision library.

Piece Movement

We primarily considered two options for our piece movement, a robotic arm, and a gantry system.

Robotic arm

We considered using a three-degree of freedom robotic arm because of range of motion and flexibility. We went away from this approach because of the complexities fine control of a robotic arm entails.

Gantry system

We decided to use a 3-axis gantry system to move our pieces. We chose a gantry because it lends itself to be controlled easier than the robotic arm. In addition, we wanted to base our movement on systems that use CNC controls to move gantries for fine control. We chose to use an X-Y gantry system with stepper motors to move around the board and a gripper for our Z movement of picking up and placing down pieces.

We chose between below-board and above-board gantries. The below-board gantry hides the piece movement which improves aesthetics, but it does change how the pieces would move. To use a below-board gantry we would need to move the pieces through the board. Pieces would have to be moved around other pieces rather than lifting them up as well. We thought that this made the system unnecessarily difficult and chose to go with an over-board gantry instead. The over-head gantry allows us to meet our requirement of minimal interference with other pieces when moving a piece. We also thought that being able to see the gantry pick up, displace, and drop the pieces added to the user experience.

Gripper Design

The gripper design started off as a servo claw attached to a rack and pinion controlled by a continuous servo. This approach did achieve our goals for picking up pieces and moving them, but it added an extra degree of difficulty because of the extra motor. The claw was also a tight fight when grabbing pieces because of the small clearance between pieces.

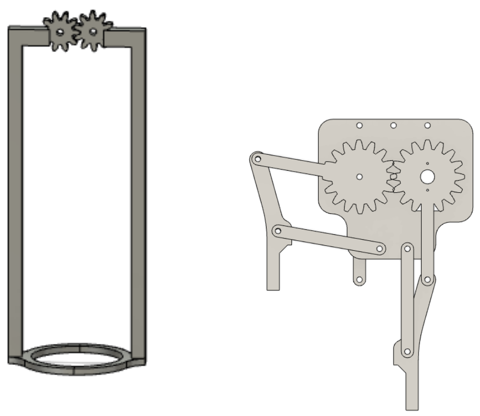


Figure 3. (Left)The first gripper design that was to be attached to a rack and pinion. (Right) The current four-bar linkage gripper showing extension.

This led us to our current gripper mechanism. We are using a four-bar linkage claw that is controlled by a servo motor. The claw is designed such that it extends as the claw closes, allowing us to maneuver within the limited space. The complexity of this design comes from getting the four-bar linkage dimensions right rather than having to control two motors at the same time as with the previous design. Additionally, each leg of the gripper will be fitted with a piece of sturdy foam to allow the gripper to conform to whatever chess piece it is grabbing and effectively lift it.

Web Client

For our web client we are using HTML, CSS, and Django as they are what we have worked with in other courses. Our web server will be hosted using Amazon Web Services' EC2 in order to allow the Raspberry Pi to establish a connection and to facilitate remote testing. Through the use of Django, we will be able to integrate the chessboard.js open source library to render the interactive chess board in the web client and chess.js to validate moves. The web client will also provide the user with meaningful feedback during gameplay such as if a move is invalid, indication that a connection with the board is stable, and when a move on the physical board is being processed. We will be using Django Channels to facilitate communication between the Raspberry Pi and the web client.

Hardware

We chose an Arduino UNO to take advantage of CNC shield controls for our gantry.
We also needed a main hardware element that could interface with the Arduino and had enough computation for the OpenCV and web server hosting.

We considered using the Jetson Nano because of its superior GPU, however our computer vision does not require us to analyze many frames continuously, so we would not be using it to full capabilities. We decided instead to use a Raspberry Pi with a Linux subsystem. The raspberry pi also has camera modules that connect through MIPI serial interfacing that we could take advantage of. We found the Raspberry Pi to sufficiently fit our needs for the system.

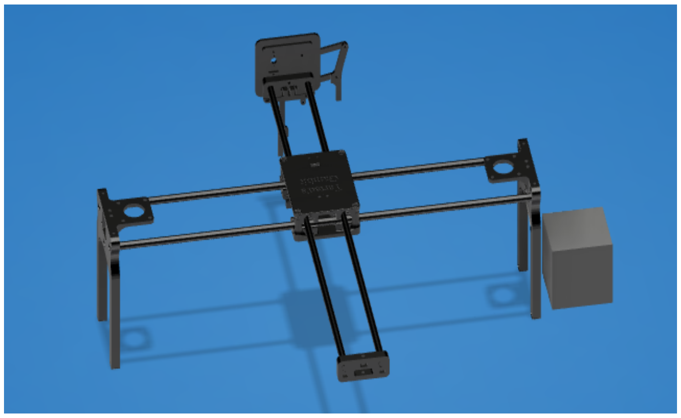


Figure 5. Gantry system with Arduino placement shown

V. System Description

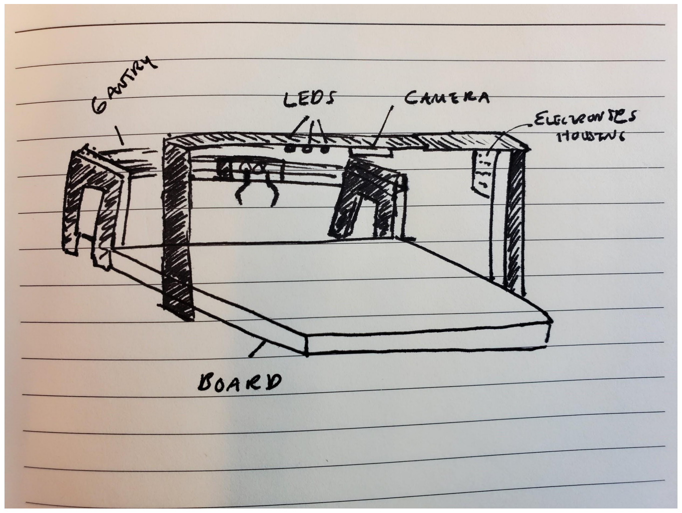


Figure 4. Sketch of camera mount apparatus

A. *Software*

The bulk of our software is broken up into two categories, the OpenCV, and the web client. While other components such as the RaspberryPi and the Arduino will also have important software running on them, these are best described as components of the hardware subsystems.

OpenCV

Our OpenCV algorithm will be running on a RaspberryPi with a RPi Camera Module V2 looking down on the board. This setup will use CannyEdge detection, followed by a HoughLines algorithm to give us the location of the edges. We aim to use this to determine piece location, and square boundaries on the board.

The CV detection will be running on a loop to show the most up to date board status. When a piece is moved by the player, and then remains unmoved for a determined amount of time, it should send a signal to the web server, updating the game state. Additionally, it should be listening for signals from the server, to alert it of a remote move which should be updated. In this case, the algorithm will take the board coordinates of the movement, determine the coordinates for movement, and send this information to the Arduino, where the CNC controller should finish the move. Then, much like when a player moves, the system will send the server a signal to indicate that the move has been completed.

Web Client

When the client receives a signal from the server of a new board state, the displayed board should be updated accordingly. The client will also listen to determine which player's turn it is, and allow players to make moves via the client.

The client will also be running a library called chess.js to perform move validation, and send this signal to the server to confirm the validity of a move. The client should be constantly updating itself to both provide the most up to date information to the server, as well as to ensure smooth movement animations for the user.

The front-end of the client will be made with chessboard.js, a library providing all the tools needed to ensure the game runs smoothly. This includes the ability to set up custom boards using Forsyth-Edwards Notation (FEN) strings. The library also fine tuned control over elements such as the orientation for the players, the animation speed, and the style of the displayed pieces.

B. Hardware

Arduino

The Arduino will be placed in an acrylic housing located beside one of the tall gantry supports as shown in figure 5. The grey box in the right half of the image represents the Arduino. Also contained in this box and connected to the Arduino are the CNC shield and stepper motor drivers. The stepper motors will each be wired to their own stepper motor driver which are connected to the CNC shield. The Raspberry Pi will send data to the Arduino in g-code format. The Arduino will have GRBL downloaded in order to receive the g-code data. This data will then be used to drive the stepper motors to the desired position. The micro servo will be attached to the grabber and wired to the Arduino. Using the state changes recognized through CV, 2 locations will be sent to the Arduino, the piece location and piece destination. When the grabber arrives at the first location, the piece location, the micro servo will be triggered to activate the gripper motion to retrieve the desired piece. Next, the grabber will move to the destination location to place the piece. The data will be received from the Raspberry Pi at a baud rate of 9600. The stepper motors will receive power from the Arduino which will be plugged into a wall outlet using a 12V 2A power adapter.

Raspberry Pi

The Raspberry Pi will be in a housing located on top of the stilt that holds the camera and LED array. It is running the standard Raspbian OS, a Linux distribution made for Raspberry Pi. The Raspberry Pi is powered using a 11.1 V portable battery as the Pi itself draws only 5V at 450 mA. The Pi transfers data to the Arduino through the Arduino's USB Serial connection at a baud rate of 9600. This should be sufficient for us to transmit g-code commands to the Arduino as g-code. The Raspberry PI will be connected to the LEDs through the Pi GPIO pins.

The LEDs indicate which players turn it is, if a move is valid, if a move has been detected, and if an error has occurred. We are showing these states by lighting the LEDs in different configurations. The system camera is a Raspberry Pi Camera Module V2 that is connected directly to the Pi through its ribbon cable. The camera and Pi communicate through a MIPI serial interface protocol. The camera can be directly accessed through Python libraries.

The Pi will be connected to the internet through Wi-Fi. It will access the Web Server through Python HTTP requests. We will be outputting the new game state to the web client through POST requests.

Gantry

Mounted on the gantry will be the GT2 timing belt, 20 teeth 5mm bore timing pulleys, 500mm M8 linear rods, LM8UU linear bearings, micro servo and grabbing mechanism, and Nema 17 stepper motors. The timing belt and pulleys, controlled by the CNC shield on the Arduino, will drive the grabber to the appropriate locations. The linear rods and linear bearings will allow for smooth x-y movement. We chose to implement the grabber using a 4-bar linkage mechanism in an effort to simplify picking pieces up. The 4-bar linkage eliminates the need for an additional motor to drive the grabber up and down to prevent it from knocking over other pieces as it is moved by the gantry.

VI. Project Management

A. Schedule

Figure 6 features our most up to date gantt chart. Our most recent updates include changes to progress statuses, changes to task owners, and changes to task end dates.

The chart in the image is broken into three sections, blue representing planning and research, green representing the design and implementation, and orange representing finalization. The column of colorful cells represents our progress with each task, and as the image shows, we are essentially done with planning and research, and are in the early stages of the design and implementation process.

The design and implementation section is where we aim to begin piecing together subsystems, testing them, and integrating them together. The next several weeks will be primarily focused on this area. We have also included two weeks of slack time, which essentially grant a safety net in the event we run off schedule.

The last two weeks are reserved for the finalization area of our project. Here we plan to work on final reports, flush out the aesthetics of the design, and robustly test our design. This region is where our final project will develop from the early stages to a more well composed, unified piece.

B. Team Member Responsibilities

Our work was split into a few different categories, such as the website, the hardware construction, the software environment, and testing.

Our website setup is to be taken on by Lillie and Juan. Having both taken a web apps development course, they are the best suited for the task.

Hardware construction, such as gantry setup, is planned to be done by all members of the team. We do anticipate that Lillie may take on a little more of this, as she has been in charge of organizing printing, and so is in possession of the fabricated pieces.

The software environment will be mostly managed by Luis and Juan. This includes the OpenCV setup, and the RaspberryPi environment. Luis may take a stronger lead on the RaspberryPi portion, as he has the camera and RaspberryPi module.

Lastly we have our testing portion, this will primarily be taken on by everyone for general testing, or large system testing. Smaller subsystems will be tested by those developing them.

C. Budget

Figure 7 below, shows our current Bill of Materials. Currently we have consumed slightly over ⅓ of our budget, however we anticipate that this will grow, as we send more of our components out for 3D printing. We don't anticipate having to buy many replacement components in the event of component failure, as we did initially purchase extras.

The table includes many components which are not planned to be used in the final design and were purchased as spares or were simply extras that were owned previously and are being used for testing. These include two of the RaspberryPis, and one of the stepper motors. Kits such as the M4 screw kit contain much more than we needed, and the linear bearing kit was a 12-pack, whereas only 8 are planned for in our design.

D. Risk Management

There are a few risks we anticipate with regards to our design. We need to ensure that our gripper can correctly function in order to move the pieces, we have already changed our design to one we feel will give more reliable performance, however we need to be ready in the event that this is not the case. Additionally, our OpenCV detection algorithm and CNC controller need to be thoroughly tested, and constantly revised to ensure that the product can perform reliably throughout an entire game.

Our schedule has risk management built in, with slack time incorporated into our plan. This will allow for any falling behind, or unforeseen errors. Additionally our task break up is rather fluid, in that we are able to change who is responsible for what portions when needed.

Lastly we have our budget. As mentioned above, our spent budget does include a few extra components of those we believe may fail throughout our design process. Our main concern is the cost of fabrication. Our initial plan of 3D printing many components seems to be too costly with campus resources available to us charging high rates per gram of filament. We have done some looking into other methods, including other on campus locations, external providers, and other fabrication methods such as laser cutting.

Overall our risk management has been to account for unforeseen circumstances to arise, and thus avoid being delayed in our process later on. The high cost of fabrication was unfortunately not taken into consideration, and as such we are spending time searching for a solution. However, we have not let this slow down our process, and instead have fabricated components necessary for testing at our current stage.

VII. Related Work

The best case of related work we were able to find are the boards made by SquareOff. They offer a smart board, allowing users to play against other players remotely using either the board or an app, or play against AI. Our design should prove to be simpler in that the SquareOff boards use magnets to move the pieces around, which can lead to complicated circuitry and mechanics. Additionally, since the SquareOff boards have all the components embedded within, the game must be set up very precisely, and the cost is relatively high due to requiring the custom set. Our approach, given reasonable production methods, should run at a lower cost, as it can be used simply as an addition to one's current set. Additionally, the inclusion of computer vision in our design allows for the game to be set up in various ways, granting more freedom than the SquareOff board provides.

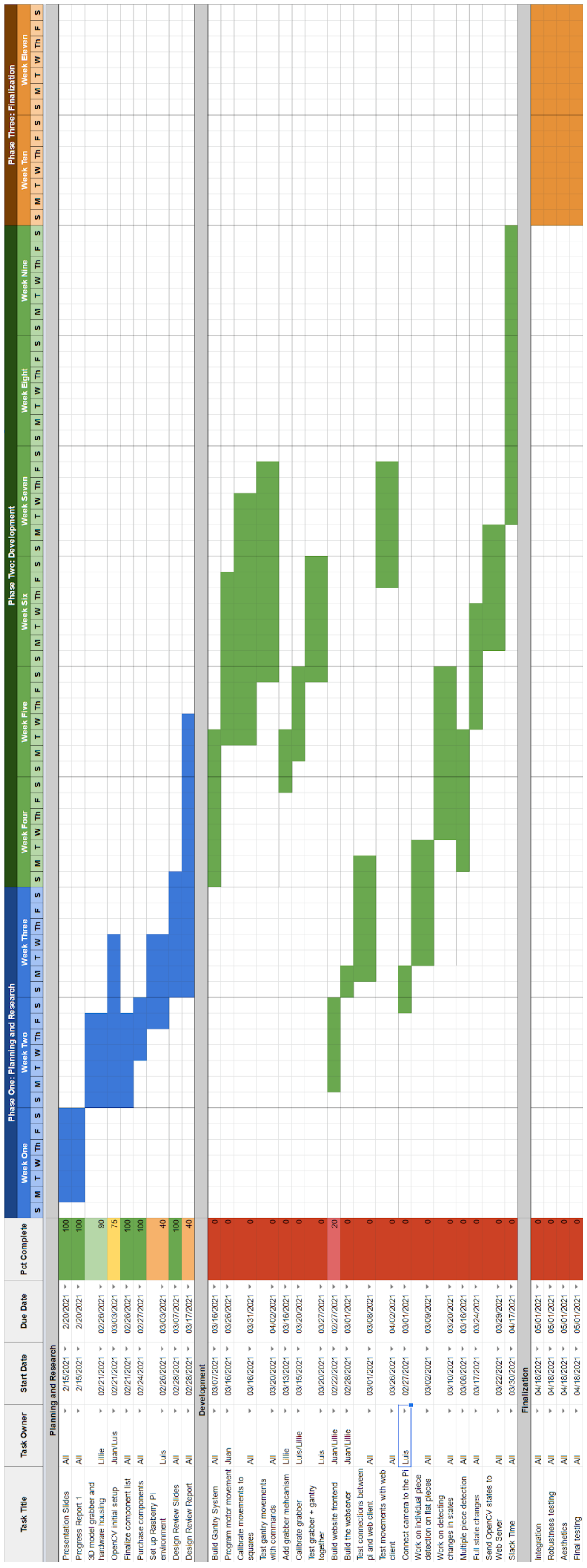
Another similar device would be one which was described in the International Journal of Engineering Research & Technology, Volume 6 Issue 9 from September 2017, titled Design and Implementation of a Wireless Remote Chess Playing Physical Platform. This takes a very similar approach to the SquareOff board, however here we see how the design was implemented. Using magnets on the pieces, and a hall sensor to detect piece location, an under the board gantry system controls an electromagnet in order to move pieces. While this leads to a similarly simple design as ours, it suffers some of the same downfalls as SquareOff, namely that the cost must include a custom chess set. Additionally, hall sensors are very sensitive, and the reliability of this system may not be the best, especially with so many magnets close together on the board. The complexity needed to accurately calculate piece location proved to be an issue for the designers, as Arduinos were too limited in memory, and RaspberryPis were too limited in GPIO pins.

Overall while our approach leads to a design less sleek than these two boards, we aim to offer more freedom, at lower cost, and will likely not face issues with storage space.

VIII. Summary

The design and plan of work for our project has undergone many iterations. The components have been planned and changed to best fit our design goals, as well as be feasible to implement. Our plan has also changed as we have become more aware of fabrication costs, we also had to modify our schedule to be more realistic in terms of implementation time.

We seek to continue the project with this level of fluidity in our process, as we find that being able to adapt to our situation will lend itself best to our success.



Task Title	Task Owner	Start Date	Due Date	Pct Complete
Planning and Research				
Presentation Slides	All	2/15/2021	2/20/2021	100
Progress Report 1	All	2/15/2021	2/20/2021	100
3D model grabber and hardware housing	Lillie	02/21/2021	02/26/2021	90
OpenCV initial setup	Juan/Luis	02/21/2021	03/03/2021	75
Finalize component list	All	02/21/2021	02/26/2021	100
Purchase components	All	02/24/2021	02/27/2021	100
Set up Rasberry Pi environment	Luis	02/26/2021	03/03/2021	40
Design Review Slides	All	02/28/2021	03/07/2021	100
Design Review Report	All	02/28/2021	03/17/2021	40
Development				
Build Gantry System	All	03/07/2021	03/16/2021	0
Program motor movement	Juan	03/16/2021	03/26/2021	0
Calibrate movements to squares	All	03/16/2021	03/31/2021	0
Test gantry movements with commands	All	03/20/2021	04/02/2021	0
Add grabber mehcanism	Lillie	03/13/2021	03/16/2021	0
Calibrate grabber	Luis/Lillie	03/15/2021	03/20/2021	0
Test grabber + gantry together	Luis	03/20/2021	03/27/2021	0
Build website frontend	Juan/Lillie	02/22/2021	02/27/2021	20
Build the webserver	Juan/Lillie	02/28/2021	03/01/2021	0
Test connections between pi and web client	All	03/01/2021	03/08/2021	0
Test movements with web client	All	03/26/2021	04/02/2021	0
Connect camera to the Pi	Luis	02/27/2021	03/01/2021	0
Work on individual piece detection on flat pieces	All	03/02/2021	03/09/2021	0
Work on detecting changes in states	All	03/10/2021	03/20/2021	0
Multiple piece detection	All	03/08/2021	03/16/2021	0
Full state changes	All	03/17/2021	03/24/2021	0
Send OpenCV states to Web Server	All	03/22/2021	03/29/2021	0
Slack Time	All	03/30/2021	04/17/2021	0
Finalization				
Integration	All	04/18/2021	05/01/2021	0
Robustness testing	All	04/18/2021	05/01/2021	0
Aesthetics	All	04/18/2021	05/01/2021	0
Final testing	All	04/18/2021	05/01/2021	0

Fig 6. Gantt Chart

Item	Quantity	Price
Raspberry Pi	3	0
Arduino UNO	1	0
CNC shield	1	0
Nema 17 Stepper Motor	3	10.99
Linearing Bearing Rod M8 x 500 mm	2	19.89
624 Bearing	1	12.99
M4 Screw Kit	1	18.99
Timing belt and pulleys	1	15.99
Continuous Servo	1	
M8 Nut Kit	1	14.99
Camera module	1	25.49
LEDs	5	0
LCD Display	1	0
chess board	1	49.99
Im8uu linear bearings	1	10.95
3D printed parts		4.8
12V 2A power adapter	1	17.99

Fig 7. Current Bill of Materials