Chess Robot

ME 450 Section 2 Team 5 - Winter 2021

University of Michigan

Sam Goldman, Andrew Kwolek, Kenji Otani, Ian Ross, Jack Zender

Executive Summary

Chess is perhaps the most standardized board game on the market consisting of only 2 players, an 8x8 square board, and 32 total pieces. There is no hidden information, and the game is reduced to pure skill and strategy with minimal luck factoring in. Although standardized, Chess is extremely complex. With over 10^{120} potential games of chess, nearly each game is novel. Chess is still thriving and is more popular than ever, despite being over 1,500 years old. Perhaps the most beneficial aspect of Chess is its application in child development. Chess is a game that develops children's (and adults') problem-solving skills, concentration, decision making, creativity, and memory. A plethora of studies have been conducted to prove these benefits. In recent years, technology and mobile applications have transitioned the game of chess from a board to our phones and computers. This increase of accessibility has further popularized the game. The virtual chess applications allow kids to play the computer (or even other humans) when they don't have a physical opponent around them to play. This shift to online chess has become even more relevant amidst the COVID pandemic as playing over-the-board (in person chess) has turned impossible. This shift, although benefiting children in certain ways, has increased screen times and decreased the relative benefits of over-the-board chess. This project aims to connect the benefits of online chess with a reduction of screen time to help with chess education.

The Chess Robot is designed to be an autonomous robotic arm that is able to compete in a chess match against a human. The robot moves their own pieces and captures the opponent's pieces in efforts to win a standard game of chess. Robotic arms that play chess have been created before, but are either industrial grade and expensive or very cheap/homemade and slow. This project aimed to create a functional chess robot that maximizes speed at a relatively low cost. It was also designed with potential for mass manufacturing in mind. With additional design and development, the skill of the robot should be easily changed, since the software is easily customized; thus, as the player improves, so will the robot. There will also be an opportunity to play chess against other humans through two intermediary robots or for one player to play while making use of an online chess platform. This feature, if fully developed, will enable chess instructors to play with children and not be limited by geographical proximity, further expanding the reach of chess education.

With a clear project objective and developed list of requirements and engineering specifications determined, a series of solution concepts were ideated, developed, and evaluated such that an initial prototype design began to emerge. First, the project was broken into key mechanical and software subsystems, and functional decompositions were developed. With these in mind, several potential subsystems were brainstormed with an emphasis on design heuristics and consideration of our functional requirements. These ideas were initially generated independently of their function in the larger project, but then developed based on the feasibility of integrating all the subsystems. This development resulted in a few emergent design categories for each

subsystem, with a focus on the mechanical movement of the robot between board squares, the gripping of different chess pieces, securing the robot to a given surface or table, and the best software and controller system to use. These more thought out solutions were then evaluated against one another using pugh matrices and final subsystem design decisions were reached.

Following a complete and thorough concept generation process, a parallel development of a mechanical systems model in CAD and a software base for our robot was initiated. The CAD model was developed in SolidWorks and went through multiple design iterations with the help of staff and peer feedback, as well as rapid physical prototyping. With the development of a physical prototype, design problems related to the system kinematics that could not be seen in SolidWorks were identified. In conjunction with this process, a software base was developed to actuate the robot, as well as manage the status of an ongoing chess match. To achieve this, software from a similar project that integrated a chess engine with a robotic system was adopted to our mechanical design. With a finalized CAD design, the appropriate actuators and controllers were determined via engineering analysis. A bill of materials and list of parts for procurement that would fit within our budget was also compiled. The parts were acquired, and an initial prototype was built.

Once the mechanical prototype was fully assembled, all mechatronic components were wired together and programming was completed such that testing could commence. Initially, all subsystems were tested independently to ensure each of the components was functioning as desired. There were a few critical errors encountered during this testing phase, which required extensive revisions and retesting. Most notably, one of the system actuators became permanently unresponsive and required the removal of a controllable ‘wrist’ joint from the system. After redesigning the mechanical system to accommodate this, testing continued on the independent subsystems. The robotic arm was able to lift pieces and move them to directed squares, and the slider mechanism was able to move back and forth between columns of the chess board. Overall, additional testing and redesign is necessary to achieve the full, cohesive functionality prescribed in the requirements and specifications. A plan for this development is outlined in the report.

Table of Contents

Problem Definition

Problem Description and Background

The game of chess as most people recognize it today began being played around the world in the early 6th century. Chess consists of an 8x8 grid of checkered squares and 32 pieces of six types and requires exactly two players. The game's premise is entirely strategy based, as all rules and conditions are available to both players and no information is hidden. [1] In addition to an ability to think several moves ahead of their opponent, the best chess players must also be able to recognize a variety of complex board positions, as there are over 10^{120} legal and distinct configurations of pieces. In recent years, chess has gained significant popularity on digital platforms, with companies like 'chess.com' and 'chess24' allowing players to connect and compete with each other across the world.

While the development of chess into an online market has been beneficial for long-distance connection, game accessibility, and chess engine software integration, it has also created drawbacks in the areas of physical health and chess education. The extended screen time associated with virtual chess can cause eye-strain, fatigue, sleep problems [2], and be detrimental to brain development in young people [3]. Furthermore, there are concerns related to ethics and cheating during remote play, as virtual players could rely on a chess engine or other illegal aid without their opponent's knowledge. The purpose of this project is to address the challenges presented with primarily virtual chess while still encouraging the benefits of chess education; such as strategic thinking, problem solving skills, creativity, and memory.

To achieve this goal our team developed a robot that can play autonomous games of chess with the aid of open-source chess engine software. Our robot addresses the concerns related to increased screen time, while also encouraging the cognitive development related to chess education at a young age. The product, with some additional development, will be usable by players of a variety of skill levels and age groups, including young children first learning to play chess.

Problem Statement

After analyzing the effects of both online chess and over-the-board chess, a clear problem has been revealed. Our finalized design objective which guided the project development was to "*Continue developing Chess education while minimizing online-chess's added screen time and keeping its long-distance multiplayer benefits.*"

Stakeholder Engagement and Outreach

To best address our project goals, we identified a few key stakeholders in chess education products or services and sought their feedback to best inform our work. First, we conducted a

research interview with Evan Rabin, a chess National Master and the CEO of Premier Chess [4], a chess education company that partners with individuals, schools, and corporate organizations to teach chess and the valuable skills that come from chess play. Evan shared some of his concerns related to remote chess and gave a great deal of insight to our initial design conversations. His input was critical in the decision to integrate a chess clock as a 'must-have' for our project, as well as the importance of a fast moving mechanism to ensure our device was able to compete effectively in timed games. Furthermore, Evan cited the importance of standard chess notation in the educational process, even stating that some students will not play their first move until they have a sufficient knowledge of proper notation and terminology. Evan explained that players are required to write their moves in standard notation in competitive games, all online platforms use chess notation to record games, and that chess notation is universal. We determined that incorporating chess notation into our robot was a necessity. He also explained that there are various time controls on games (1, 3, 5, 15, 30, 60, 120+ minutes). He noted that the robot must make a move in around 5 seconds to ensure that the game can be completed in the restricted time control. He further mentioned that it would be very beneficial, but not strictly necessary, for the robot to be able to play out old games to assist in studying positions, learning openings, and improving as a chess player. We took this into consideration when defining our requirements and determined that this was an aspirational function since Evan noted it wasn not critical to creating a meaningful product. Evan finally explained that each player is a different skill and if they were playing the robot, not a multiplayer game, the robot should be at their skill level. We agreed and determined that varying skill levels is a necessary function of our product. Evan enjoyed our conversation and encouraged us to continue our conversation with him throughout the design process. We valued his opinions and incorporated them into our problem definition.

We also hoped to speak with parents of young children who play chess, such that they could provide us with insight about safety and usability concerns. We spoke with our own parents who provided insight into buying a chess robot and what features would need to be in place in order to feel comfortable leaving their child alone with the machine. One parent noted that while their child loves chess, they strongly dislike how much time they spend playing on their phone. They also mentioned that they need safety mechanisms, although not being sure exactly how to implement them, that restrict the robots range of motion to protect their child. [5] This comment further proved why safety restrictions were a must-have requirement. The parent continued in saying that they would want the robot to be portable since they would not want the robot in their kitchen or living room forever; upon asking where they would like to store the robot permanently, the parent indicated that a shelf or closet would be suitable. We took this feedback into consideration when defining our requirements and engineering specifications.

Another stakeholder was The Latin School of Chicago high school Chess Team, a team that Sam Goldman founded six years ago. After recently winning their state tournament, the team reached out to Sam for advice on continuing their chess education in college. They are likewise

extremely interested in assisting in developing the Chess Robot and providing insight into its potential uses on their team for practice, casual play, and competition. Many of the current members are aspiring engineers as well, so their input into the solution development process had additional value from a technical perspective.

We also conducted some early patent research that has given us some insight into potential subsystem designs, including a design for robotic arm [6] and cartesian slide system with magnetic piece inserts [7]. Neither of these contained any schematics but served as useful ideation aids in our brainstorming phase.

Finally, we conducted some research of existing designs that would help achieve some of our project goals. These included other chess-playing robots [8] as well as a computer-vision algorithm developed for chess boards by a student at Stanford University [9].

Requirements and Specifications

Our project requirements and specifications were focused on creating a robotic chess opponent that is safe for users of all ages and compatible with a variety of skill levels. Due to the open ended nature of this project, we decided to split our requirements into three tiers: must haves, nice to haves, and long term and aspirational. These tiers provided us with stretch goals for functions of the robot and gave structure to our design priorities as defined by research and stakeholder input. Each requirement has an associated specification that is tangible, measurable, and unambiguously defined. A justification for these quantifiable specifications is also included, with external resources referenced accordingly. The subsections below will showcase the different tiers.

Must-Haves

This tier contains all of the most highly prioritized requirements which we have identified as necessary functions or characteristics of our product. In order for our project and design to minimally solve the presented problem, the must-have requirements needed to be met. The requirements, engineering specifications and corresponding justifications, as well as embedded links are presented in Table 1 below.

Table 1: “Must Have” Requirements, Specifications, and Justifications.

Requirement	Specification	Justification
Portable	■ Carriable by an average child across a room (<= 4.5 kg) ■ Foldable for storage (<= 0.03 m^3 volume folded)	■ Based on doctor’s recommendations for max backpack weight [10] ■ Should rest comfortably on a shelf with depth of 0.3 m [11]
Durable	■ Must not fall over due to own inertia while the arm is moving ■ Functional runtime of ~2000 hours before recalibration	■ Mechanism should be stable and not come loose from any motion ■ Runtime of 2000 hrs accounts for 40 hrs a week for a year
Function Using Standard Size Chess Board and Pieces	■ Standard USCF tournament set has 5.7 cm squares and a king's height of 9.5 cm	■ Robot plays on standard tournament [12] set to simulate a setting the player would realistically play in
Autonomously Lift and Move Pieces In the Correct Orientation and to the Correct Location	■ Lift pieces with 0.635 cm clearance over all other pieces based on two kings clearing each other ■ Execute response moves within 5 seconds ■ Place each piece upright with its center 5 mm or less from the end move square’s center ■ Must be able to lift pieces of 52g with a 1.75x safety factor (91g) .	Clearance and the deviation of pieces from the square’s center based on research for other PID controlled robotic arms [13] 5 second move time recommended by stakeholder Piece weight range determined by double-weighted standard tournament pieces [14] with a 1.75x safety factor [15]
Receive Manual Input in Standard Algebraic Chess Notation From Human Player	■ Standard algebraic chess notation ■ Ex. f4 e5 = White pawn to square f4, black pawn to square e5	■ Robot should understand and use chess’ universal notation [16]
Play Timed Chess Games	■ Uses one chess clock as ‘trigger’ for robot to make a move ■ Standard games are in increments of 30s,1min,5min,10min,15min,30min (per player)	■ Timer interaction [17] is fundamental to competitive chess and the device should simulate that environment
Play Game of Chess According to all Standard Rules	■ 7th Edition of the US Chess Federation’s Official Rules of Chess [18]	■ Important for device to simulate official tournament chess rules
Autonomously Make its Own Decisions When Playing a Move	■ Plays one legal move per turn with max 10 second delay from player’s previous move.	■ Crux of the problem; core component of overall functionality ■ 3 seconds to “think”, 7 to move
Have a Mechanical Structure that Allows for Safe and Restricted Movement	■ Zero exposed wires and sharp edges ■ OSHA spec for robotic arm movement [19]	■ Needs to be safe for children to use when playing chess ■ Cited as stakeholder priority
Vary Levels of Difficulty Prior to Game	■ Software depth 1 (ELO increment ~66) [20]	■ The robot should be accommodating to players of different skill levels

Nice to Haves

This tier contains requirements that we find to be technically realistic, but may not be able to be achieved in the timespan of one semester. We have determined that these requirements would be beneficial to implement, but are not detrimental if neglected. Our project will still be a success with simply the must have requirements, but we strived to satisfy these requirements as well and believed that they were achievable. The nice-to-have requirements, engineering specifications and corresponding justifications, and embedded links are presented in Table 2 below.

Table 2: "Nice to Have" Requirements, Specifications, and Justifications.

Requirements	Specification	Justification
Limit Cost of Functional Prototype to $400	■ Make all efforts to keep price below $400 ■ Reach out to potential sponsors (Chess.com, other professors, high school chess team) and apply for grants	■ Given budget is $400 so we must attempt to stay without these limits ■ If we see that we are likely going over budget, then we will attempt to gain funding from other sources ■ Although this is nice-to-have, it is a must have that we either limit cost to 400 or find gain additional funding
Can Adjust Skill Level Mid-Game to Meet Real-Time Player Performance	■ Dynamically determine ELO 200-2800 and increment 100 ■ Can run engine up to depth 25	■ Creates a more dynamic game and educational experience consistent with that of playing a real human that can make adjustments on the fly
Can Play Chess Variant Games	■ Chess 960, bughouse, begin with missing pieces	■ Adds breadth to the robot's functionality, noted as stakeholder interest ■ Chess Variants [21]

Long Term and Aspirational

This tier contains all of the desired requirements that are potentially out of scope of our semester timeline. These requirements would be necessary if we continue developing the Chess Robot after the semester concludes but were not necessary to show its basic functionality and indicate successful project completion. The presented long-term requirements would be very difficult to satisfy but would significantly improve functionality and use. The long-term and aspirational requirements, engineering specifications and corresponding justifications, and embedded links are presented in Table 3 on the following page.

Table 3: Long-Term and Aspirational Requirements, Specifications, and Justification.

Requirements	Specifications	Justification
Can Play a Complete, Fully Autonomous Game	■ Can identify start of turn, location of all pieces, and make a move with 0 manual intervention	■ Simulates a fully autonomous player, making it more user-accessible
Can Play Games With Any Size of Board and Pieces	■ Ability to identify board grid autonomously and break into 64 squares ■ Can autonomously translate moves into distances on the board and place pieces with 100% accuracy	■ Allows for wider functionality and use for any type of chess set ■ Not necessary to implement for desired functionality given constraints
Can Store Games Digitally and Upload For Analysis	■ Can store ‘pgn’ files or ‘txt’ files for 100 games at a time	■ Adds a unique feature for player improvement

Concept Exploration

Subsystems Approach to Concept Generation

Developing the robot's mechanical and software elements required a complex breakdown of many subsystems. There were functions related to the controls of the robot, from user input to software output, and a few key mechanical subsystems that are essential in executing board moves. To play a complete and autonomous game of chess, several different components must work in functional and physical synchronization. As a result, a wide variety of potential design solutions needed to be explored and evaluated. Concept generation and development for each major subsystem was therefore carried out independently of each other, preventing any ideation from being limited by systems integration concerns. Concept evaluation and selection therefore relied in some capacity on each subsystem's ease of manufacturing and assembly into the larger robot. With this in mind, two functional decompositions were developed to aid in the breakdown of the robot into its most critical parts. Shown below in Figure 1 is the functional decomposition of mechanical subsystems in the robotic chess arm. It includes the Brain/User Interface, House Software/API, Position Controller, and Gripper along with each core function associated with them. This functional decomposition also demonstrates how each function communicates with other functions from the same or different subsystem. Essentially, a roadmap of functions is constructed to help piece together all of the different parts of the overall system.

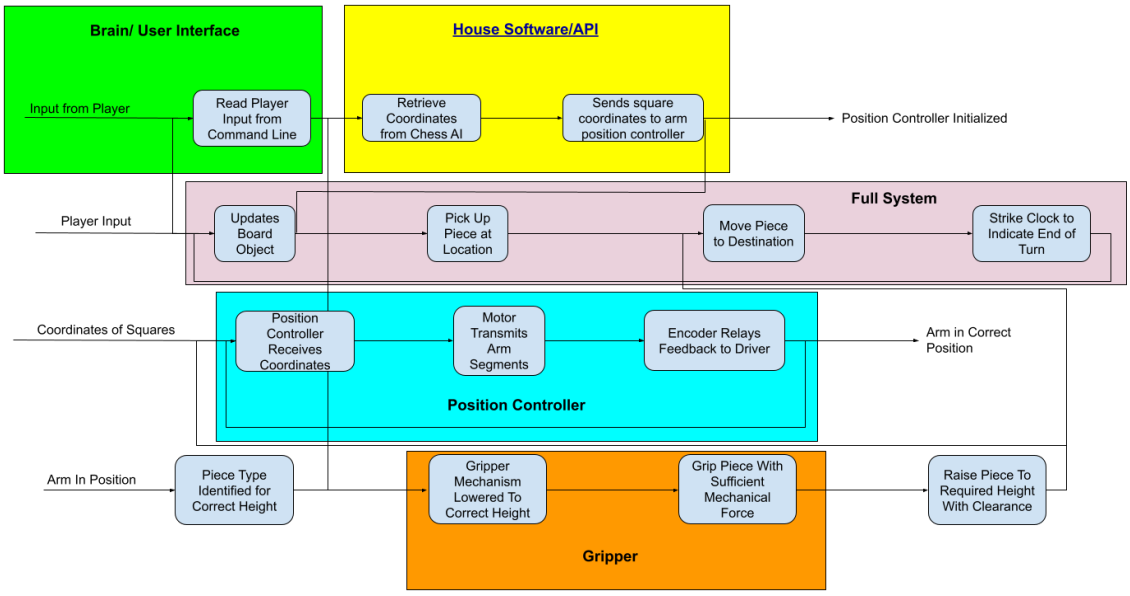


Fig. 1. A functional decomposition of the robot's mechanical subsystems by function

A second functional decomposition was also developed to focus more on the software subsystems' functionality. The software's functionality isn't as dependent on the design decisions as some of the other subsystems so a more detailed functional decomposition was developed and

is shown below in Figure 2. The house software handles player interaction, communication with chess engines and API's, and passes instructions to the controller subsystem which in turn relays instructions to the mechanism to physically move the robot. The main inputs to the house software (i.e. the piece of software we will be writing ourselves to drive the project), are a player's move at a point in time, and the desired difficulty setting prior to the game. When a player plays and inputs a move, the house software checks the validity of a given move, then requests a move from a chess engine. It then takes that move and converts it into a set of coordinates or instructions that the controller and mechanism must carry out. This functional decomposition outlines the basic functionality of the software systems. In reality, developing the code base for this project was an intensive endeavour.

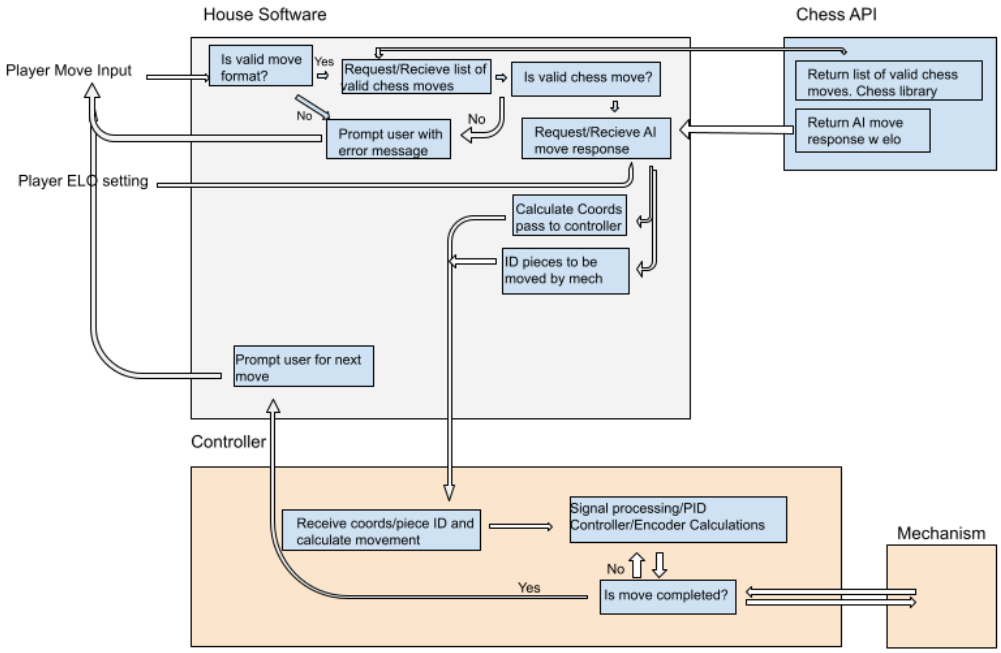


Fig. 2. A functional decomposition of the robot's house software and integration with Chess API as well as user input

With a clear breakdown of the robot's required functions into subsystem categories, different concepts for potential solutions were then considered.

Concept Generation

After adequately dividing our robot into subsystems, we generated potential solution concepts through initial brainstorming. Individually, we developed as many ideas as possible and did not limit our ideation to feasible solutions. Although we all had some preliminary ideas based on our prior understanding of robotic arms and existing chess robots, we challenged ourselves to think out of the box in the brainstorming stage. An example of an individual brainstorming list

separated by subsystem is shown below in table 4, and related images and sketches can be found in appendix B.

Table 4: List of brainstormed ideas organized by robot function.

Mechanical Arm	Piece Grabbing

1. Cartesian plane sliding mechanism
2. Polar arm mechanism
3. Linkage arm
4. Materials - durability
  a. Aluminum
  b. Plastic
  c. PVC
  d. Steel
5. Types of motors - durability
6. Horizontal Movement plane Arm
7. Arm on Rail
8. Pneumatics
9. Wheels on Table that move a drop down gripper
10. Magnets under table
11. Squares actually move and slide with pieces on them
	1. Gripper
	  a. What kind of claw? 2,3,4 prong etc.
	  b. How do we execute a claw mechanism?
	2. Magnetic
	  a. Custom pieces? 3D printed
	  b. Embed magnets in bought pieces
	  c. Solenoid mechanism
	3. Scooper
	  a. Scoop the pieces up from underneath
	  b. Scoop from under head of piece
	4. Expandable rod that goes into the piece and expands and then retracts to release it
	5. Vacuum sealed to pick up pieces
	6. Small pushing mechanism to push every piece
	7. Each piece has a wheel under it to move itself, without need for a gripper

Attachment to Table	**Piece Location**
1. Weights on the mechanism 2. Suction cups 3. Velcro 4. Vice grips and clamps 5. Glue 6. Bolt into table	1. Computer vision 2. Rfid on pieces 3. Manual notation based 4. Pressure plates 5. Capacitor sensors 6. QR sensors on each piece

After individually brainstorming ideas based on subsystems, we attempted to generate even more ideas through the use of design heuristics. We took our individual list and expanded on it by considering each existing idea and going through the list of design heuristics to edit/add to them. For example when considering idea 1b above (polar arm mechanism) and thinking about design heuristic 65, telescoping, and 32, expand and collapse, we added the following ideas:

a. Arm that contracts each segment when it is not moving to save space
b. Arm segments that when turned completely vertically can collapse into the base

c. Arm segments that can be separated and clicked together easily for easy storage but also easy assembly

Another example of utilizing design heuristics occurred when considering a vertical gripper and design heuristic 19, change flexibility. We added the following ideas to our list:

a. Gripper that matches shape of each individual piece for custom grip
b. Adjusted shape of gripper surface to match base of pieces
c. Adjusted flexibility of claw-like gripper end pieces to add friction and better grasp piece's surface

After cycling through the design heuristics, our lists grew significantly. We proceeded to meet as a team and compile our ideas to determine the overlaps and talk through our ideas in efforts of combining the ideas and building even more. Through this discussion we generated even more ideas for all subsystems. After compiling and generating ideas as a team, we overlaid our subsystems and ideas on our functional decomposition to identify gaps and red flags. Since the functional decomposition is meant to identify all functions that our robot must satisfy as well as how those functions relate to each other each, it is critical that our design ideas satisfy all functions and integrations appropriately. To do this, we took each square in the functional decomposition and used them to further categorize our subsystems and ideas. Through this process we identified that we had not developed sufficient ideas for the controlling aspect of our robot. We then generated more controller and integration ideas and ensured that those ideas satisfied the remainder of the squares on the functional decomposition.

Concept Development

Once we were comfortable with the diversity and quantity of generated ideas, we quickly filtered out the non-feasible solutions. For every subsystem, we selected the feasible solutions that would satisfy our functional requirements. For the main arm structure, there are three potential solutions and are shown in Figures 3, 4, and 5, below:

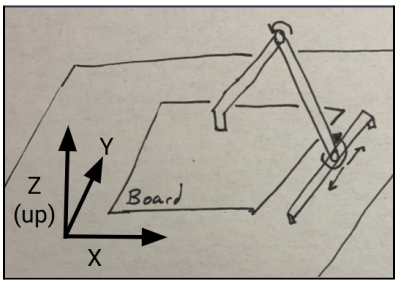
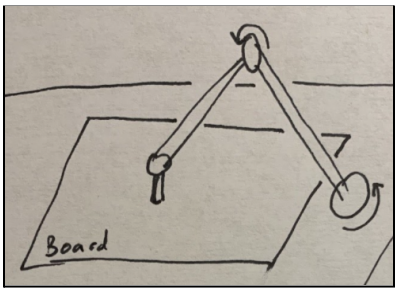


Fig. 3. 1D Track with Rotating Arm



Fix. 4. Polar Base and Arm

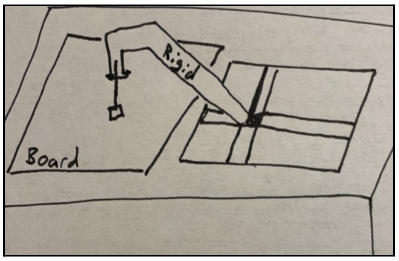


Fig. 5. 2D Track with Rigid Arm

The 1D track with rotating arm (1D Track for short) has a base that moves along a track in the Y direction. The arm will extend by having rotating joints at the base and at the top of the arm, allowing the arm to reach out in X to access each individual square. This design simplifies

control as we only need to define the location of 8 squares in X and divide the board into 8 distinct files in Y. For example the only difference between square a8 and c8 is the distance the base travels on the track. The Polar Base and Arm (Polar for short) is the most similar to an industrial robotic arm. While the arm remains the same as the 1D Track, the base is fixed in X,Y but can rotate to ensure that each square is accessible. The Polar design is the hardest to control as each square must be defined independently. The Polar arm would also be the fastest but hardest to manufacture and assemble. The final feasible design is the 2D Track with Rigid Arm. The 2D Track consists of a base that is slightly larger than the size of the chess board and a rigid arm. The rigid arm requires that there be a gripper that can move vertically to grab pieces and lift them over the others. This design proves to be the simplest to control as each position on the board directly corresponds to the position of the base on the track.

For the gripper subsystem, we limited our concepts to three primary designs, as shown below in Figures 6, 7 and 8.

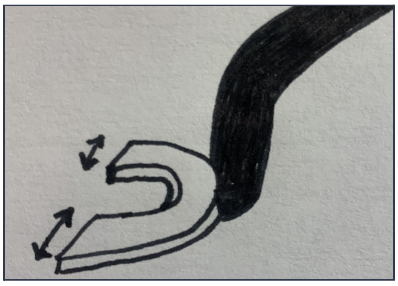


Fig. 6. Diametric Gripper

Fig. 7. Top-Down Gripper

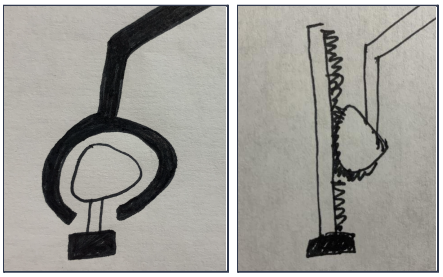


Fig. 8. Magnetic grippers: Ball and socket (left), rack and pinion (right)

The Diametric Gripper grabs each piece from the side by expanding and contracting its opening depending on the piece size. This gripper can be used easily in conjunction with the 1D Track and the Polar Arm and can be used with the 2D track if a lowering mechanism is employed. This solution would be difficult to manufacture in-house and there were limited off the shelf options. Next, the top down gripper can be lowered to the piece and grab it from above. Like the diametric gripper, it can be used with the 1D track and the polar arm with no modification, but a lowering mechanism must be employed to use with the 2D track. While the top-down gripper would have been very difficult to manufacture in house, there were numerous off-the-shelf options available. There is also potential to add additional foam inserts to better shape and grip each respective piece. Finally, the magnetic grippers designs utilize an electromagnet that can be engaged to lift pieces and disengaged to release them. This design requires that the pieces be modified such that there is a magnet on top of each piece. The ball and socket option allows for easier placement of the pieces as there is more room for error in placement. The rack and pinion version allows for integration with all three potential arm structures. For both versions, there was potential to manufacture in-house but also have off-the-shelf options.

The remainder of the subsystems were the safety restrictions, integration with chess clock, and securement to the table. For these subsystems, the design decisions were made in the solution development phase. For example, to secure the robot to the table, we generated many ideas including the use of suction cups, C-Clamps, velcro, weights, and magnets. After further development of our overall design, we will have more information about selecting the appropriate method. Similarly for safety restrictions we generated ideas including hard stops, laser cages, torque sensors, ultrasonic sensors, time sensor, velocity limits, physical padding, and a panic button. For integration with the chess clock, we can either utilize a real clock and engage using the gripper, or we can use a virtual clock through the players API and not physically engage with it. Implementation of these ideas will be made in later stages of the design process.

Concept Evaluation and Selection

After the concepts were developed, a metric was required to determine which of the different concepts, for each category, would be the most suited for pursuit as a fully realized mechanical design. For some of the concept categories, no major decisions needed to be made because they are not mutually exclusive, such as the safety aspect, where multiple different safety features can be implemented without impeding the functionality of other ones. We decided to use a customized Pugh Charts to help us evaluate and ultimately decide on specific designs. The following sections contain information regarding the evaluation and selection of specific designs generated during our concept exploration phase.

Mechanical Movement Mechanism

The mechanical design of the arm was rated on categories based on the requirements and specifications. The weight of them were arbitrarily determined based on the amount that the criteria affects the original specifications. Additionally, we made sure to prioritize safety at all costs, so safety was maxed to 5 to ensure that safety was highly considered. In general, safety of a mechanism was determined by analysing the degrees of freedom, the predictability of movement, and volume of space that would be dangerous to the user. The criteria that were deemed nice to have, but not absolutely necessary to reach the original specifications, such as controllability and appearance, were given a lower score because they are mostly inconveniences towards our work rather than the final product. The rest of the criteria were given three point weights for the pugh matrix. One idea deemed to be the most neutral idea was decided to be the baseline arm design, with all other ideas being compared against it over all of the criterias. Table 5 shows the pugh matrix for the consolidated design idea for the overall arm design shown in Figures 3, 4, and 5.

As shown in Table 5 on the following page, the arm design that was selected by this process was the 1D Cartesian Motion w/ Rotating Arm design. This is the overall arm design that was developed in the solution development phase of this project.

Table 5: Pugh Matrix for the overall arm design.

		Overall Arm Design		
Criteria	Weight	Polar Base and Arm	2D Cartesian Motion w/ Drop Down Gripper	1D Cartesian Motion w/ Rotating Arm
Weight	3	+	-	0
Speed	3	+	-	0
Size	3	+	-	0
Safety	5	-	+	0
Manufacturability	3	-	0	0
Controllability	1	-	+	0
Cost	3	-	0	0
Appearance	1	+	-	0
	+	10	6	0
	0	0	6	22
	-	12	10	0
	Total	-2	-4	**0**

Piece Gripping Mechanism

Once the ideal mechanical subsystem for reaching specific board squares was determined, we then had to consider how best to move all the different types of pieces between squares. The pieces needed to be lifted vertically to avoid knocking into each other and had to be lifted high enough such that two kings could clear each other with 0.635 cm of clearance. After developing our brainstormed ideas into three major categories (diametric gripper, top-down gripper, and magnetic collection), we also researched other possibilities and discovered a universal gripper which uses granular solid and a flexible balloon to grab items. Having these four major categories we developed a pugh matrix to select our best piece gripping mechanism. Criteria were considered for this matrix based on stakeholder input and design criteria as researched in our problem definition phase. Weights were assigned to these criteria on a 1-5 scale, with 5 being the most important and 1 being the least. Each criteria for each design option was then assigned a +, -, or 0 score based on how that design would theoretically perform compared to a 'baseline' design. To calculate the scores each weight was multiplied by a +, -, or 0 accordingly and the total positive and negative points combined then compared. The baseline design, in this case the diametric gripper, was assigned 0 scores throughout and the remaining designs were then assessed. The final scores for each gripper design option, as well as their ratings in each criterion can be found on the following page in Table 6.

As shown in table 6 on the following page, the gripper design that was selected by this process was the top-down gripper, a design for which many off-the-shelf options are available within our price range and sold with an accompanying actuator.

Table 6: Pugh Matrix for the gripping mechanism

		Gripping Mechanism			
Criteria	Weight	Diametric Gripper	Top-Down Gripper	Magnetic	Universal Gripper
Weight	3	0	+	+	-
Speed	1	0	0	+	+
Size	3	0	+	+	-
Safety	5	0	0	+	+
Controllability	1	0	0	+	+
Piece Modification	5	0	0	-	0
Cost	3	0	0	+	-
Motors	3	0	+	+	-
Manufacturability	3	0	+	0	-
Satisfies Nice to Have	3	0	0	-	+
	+	0	12	19	10
	0	30	18	3	5
	-	0	0	8	15
	Total	0	**12**	11	-5

Programming and Brain Decisions

The first design decision for the programming and controller brain subsystems was to determine the coding environment for the house software. Python was quickly determined to be the optimum programming language for this project. Python offers a very useful library called python-chess, which offers an easy to use framework for coding in the “language” of chess and easy API and engine interaction [22].

The next design decision was which controller brain to use for the project. A pugh chart was used to choose between using an Arduino or a Raspberry Pi. Three criteria were determined: Linkability to Python, Reference Material available, and project group Familiarity. The criteria were weighted on a five point scale with weights of five, three, and one respectively. Linkability to Python received a five because python is the environment chosen for the project and the less languages being implemented simultaneously the better. Reference Material received a three because having source material makes building a code base easier. Familiarity received a one because the group has a lot of programming experience. Raspberry Pi’s run on python, which was seen as a large advantage given our house software is being written in python. Using an Arduino board that runs C code would be more difficult to link to the other parts of our code base. The next advantage for the Raspberry Pi was a wealth of source code material for building chess robots using a Raspberry Pi board. The project RaspberryTurk has an open source git repository that will be very helpful for building our own code base [23]. The project team was less familiar with using Raspberry Pis, but this was identified as a less important heuristic for

determining which board to use. From the pugh chart below in Table 7, it was determined that the best board to use was the Raspberry Pi.

Table 7: Pugh Matrix for the controller brain

		Controller Brain	
Criteria	Weight	Arduino	Raspberry Pi
Linkable to Python	5	0	+
Reference Material	3	0	+
Familiarity	1	0	-
	+	0	8
	0	9	0
	-	0	1
	Total	0	**7**

Solution Development

Overall Solution Development Approach

Through concept exploration, we determined the structure of the arm, slider mechanism, and overall control systems. After the high-level design selection was complete, we moved into solution development to create a detailed final design of the Chess Robot. As a team, we delegated each subsystem to individuals and pairs of teammates. We believed that while the arm and the slider needed to be integrated, the detailed design of each should not affect the function and design of the other subsystem, excluding the connecting interface. Thus, we completed parallel development in designing each subsystem before creating the combined design.

We began subsystem detail design by creating an initial virtual prototype. To ensure consistency and version control, each subsystem was modeled using the latest Solidworks version and a grabcad folder was created. We first created a model of a standard chess board with standard size pieces so each subsystem could be sized appropriately. Initial Arm and Slider subsystems were then developed and assembled together, as shown in Figures 9, 10, and 11 below.

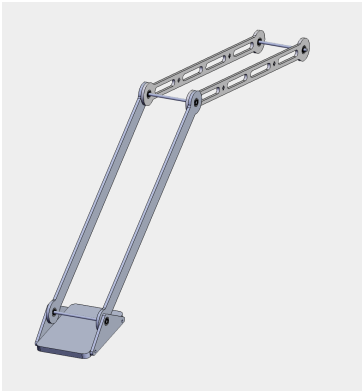


Fig. 9. Basic arm structure

Fig. 10. Slider Subsystem

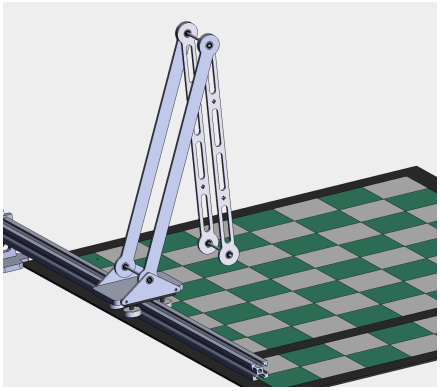


Fig. 11. Initial complete assembly

After creating the initial assembly in Solidworks, we were able to all visualize the same robot. Previously, we each had our own distinct thoughts on what the Chess Robot should look like, so creating a very basic assembly provided a foundation for our detailed design. The initial design was also useful in ensuring no major mistakes regarding range of motion, interference, tipping, and durability were made before the detailed design. Following the initial virtual design, we met as a team to create a physical mockup to further analyze the basic design and prepare to move into detailed design. Our physical mockup is shown in Figures 12 and 13 below.

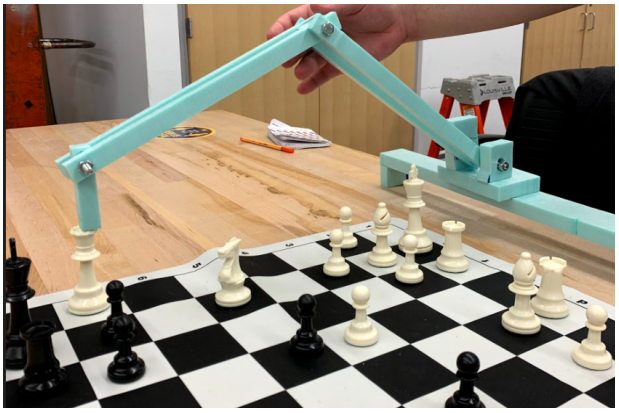


Fig. 12. Side view of to scale physical mockup

Fig. 13. Front view of to scale physical mockup

The physical mockup enabled us to analyze the structure of our design, joint design, and identify areas of potential failure. Even more, we were able to meet in person with a basic physical prototype to further discuss transmission and actuation and final design decisions. In particular, the physical mockup presented potential problems from driving links. As we rotated the links, it was evident that the complementing link would lag behind and had potential to fail and present control errors. We sought to drive both links in tandem to solve this problem. Our solution to this problem is explored in depth on Page 25. Similarly, we identified struggles with joint design as small radial motions would compound and present significant control accuracy problems at the end of the linkage. We aimed to simplify joint design to minimize these effects. At this meeting, we were also provided with a similar slider system from a previous ME 450 project. We then analyzed the provided slider and motor and determined that it could successfully be adapted for our purposes. Full design and engineering analysis of the slider is explored on Page 33.

After creating the physical mockup, we continued developing the virtual mockups while completing significant engineering analysis to justify design decisions. We continued our subsystem design approach and came together throughout to ensure integration would be successful. We developed a simplistic solution that maximizes reliability and controllability while working to meet all engineering specifications. The upcoming sections dive into detailed design and analysis of each subsystem and the software development process.

Mechanical Arm

The mechanical arm subsystem of our robot was designed to perform most of the kinematic functions related to picking up, moving, and replacing pieces on the board. Having decided on a motorized arm and top-down gripper which would operate in a plane perpendicular to the linear base slide, the next decision was around arm degrees of freedom and number of actuators to be used. We ultimately decided that having three degrees of freedom and three actuators would

maximize the controllability while still ensuring a simplistic and compact design. A side view of our final arm highlighting the degrees of freedom is shown in Figure 14 below.

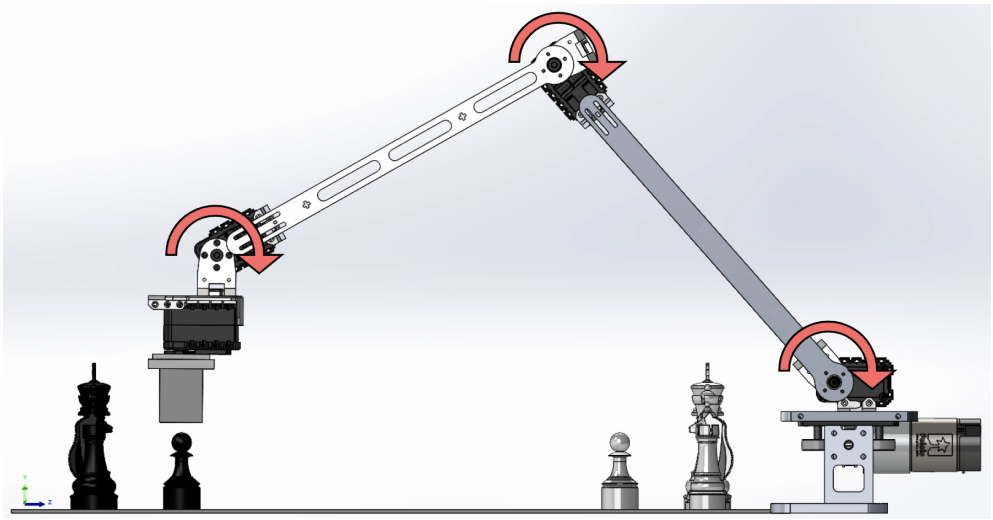


Fig. 14. Side view of the Chess Robot highlighting notable degrees of freedom in the arm. Where the right two actuators control overall arm positioning while the left actuator controls the angle of the gripper.

Having three degrees of freedom ensured that each piece in each square could be reached. Further kinematic analysis is explored on Page 26. After determining the overall structure, motion, and degrees of freedom for the chess robot, we attempted to select actuators. After researching actuators that are optimal for our control purposes (easy to program, precice, work well in tandem with other actuators, and are compact), we determined that Dynamixel actuators would provide the ideal control for the Chess Robot purpose. They are cheap, compact, have small resolution, integrate with other Dynamixel actuators well, and have a variety of motor mounts. Although we determined the most ideal brand of actuators, we had to determine which model would best suit our purpose. Ultimately we decided to use the AX-12A actuator which allows for direct control and no transmission; the speed and torque analysis that led to this decision is shown on Pages 27-29. Next, from our physical mockup, we realized that there may be control and lag issues if we only drove a single side of the robot link (one of the two links). Thus, we aimed to mount the actuators to directly drive both links at all three driving locations. Thus, we used multiple Dynamixel axles to achieve this functionality. The mounting positions are shown in Figures 15, 16, and 17 below.

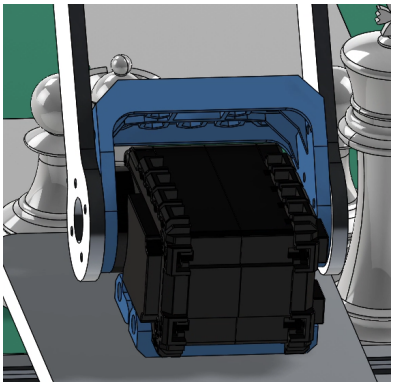


Fig. 15. Base actuator driving both primary links in unison

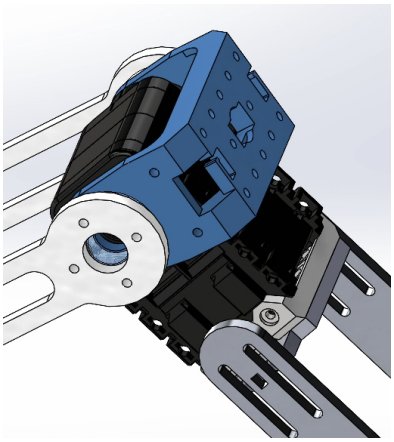


Fig. 16. Reach joint actuator driving both reach links in unison

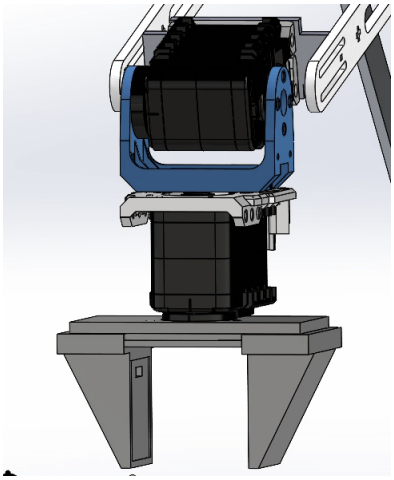


Fig. 17. Actuator driving gripper rotation uniformly

The AX-12A provides convenient mounting capabilities and assisted us in ensuring reliable actuation and control. The mounts will be fastened to the respective links and bases using provided screws. Also shown in Figures 16 and 17 is a slot for adjustable positioning and a locating feature for our predetermined optimal positioning. While kinematic analysis has been completed at the given rotational positioning, there is flexibility to make length adjustments as needed, which we felt was an important design feature. This mounting platform used to connect each link and stabilize and control the position of the actuators in Figures 16 and 17 and shown in Figure 18 below.

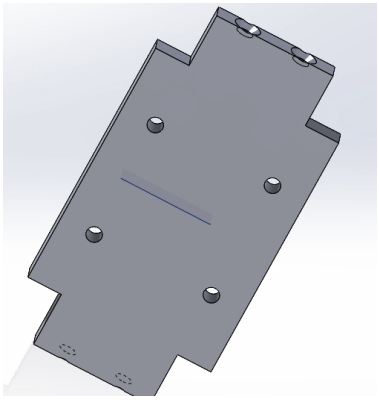


Fig. 18. Aluminum actuator mount with threaded holes on face and on side

Next, the base links and reach links are shown below in Figures 19 and 20 below.

Fig. 19. Aluminum base links

Fig. 20. ABS or Acrylic reach links

The base links were made of aluminum to adequately lift and support the actuators, reach links, and gripper. The reach links, though, were made from acrylic as they have a longer moment arm to the base, are not lifting as much mass, and with the goal of minimizing weight and tipping potential. Material selection analysis and tipping analysis have been completed and are shown on Pages 29 and 30.

The entire assembly in an isometric view is shown in Figure 21 below.

Figure 21: Final CAD assembly shown in a front-facing and rear-facing isometric views.

Engineering Analysis | Kinematic Analysis

To verify the mechanical arm's effectiveness at fulfilling our requirements and specifications, it was first put through a virtual kinematic test to verify that all necessary arm positions could be reached. This included testing each of the six types of pieces at each of the eight board depths, or a total of 48 different locations. The arm was able to reach all of these locations without inflecting at any joints, binding, or interfering with itself. Images detailing the arm's position in the most extreme cases of motion can be seen on the following page in Figure 22.

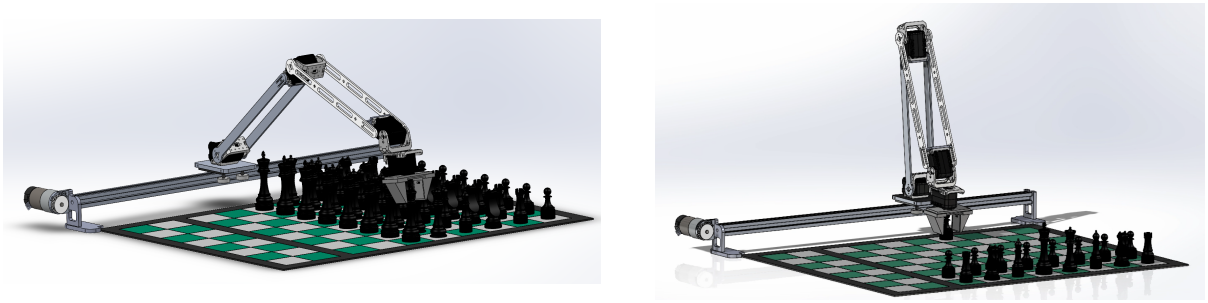


Fig. 22. The mechanical arm is positioned in the most extreme cases of piece location and joint articulation, shown for the tallest piece at the farthest rank (left) and the shortest piece at the closest rank (right).

Engineering Analysis | Required Speed and Torque Requirements

Next the required rotational speed of the arm needed to be calculated such that a torque analysis could be conducted with reasonable parameters. Considering our specification that the robot needed to make its moves in 7 seconds, and the slider mechanism would require roughly 4.5 seconds to make it's movements, the mechanical arm needed to complete its range of motion in 2.5 seconds. To understand the maximum angular range the arm would have to operate in, the law of cosines was used to calculate a range at each joint. The equation for the law of cosines is

$$c^2 = a^2 + b^2 - 2abcos\gamma \qquad (1)\ [24]$$

where a,b, and c are side lengths in inches, and γis the angle between two sides in degrees, assuming notation outlined in the figure associated with the reference material. Considering this equation, c values were calculated for the most extreme board positions to understand the full angular rotational needs of the system. The maximum c value was determined to be ~19 inches and the minimum value was determined to be ~3 inches, as verified using the Solidworks model. Given these values, angle measurements for gamma and beta were determined using the law of cosines, resulting in the values listed below in Table 8.

Table 8. Joint distances for the arm's base joint and gripper joint and the corresponding joint angles for the base and reach joints, in degrees.

Distance from Gripper Joint to Base Joint (c)	Resulting Reach Joint Angle (γ)	Resulting Base Joint Angle (β)
3 inches	15.50°	62.96°
19 inches	129.52°	23.95°

Given these parameters and knowing that the maximum articulation in a single, non-capturing move would be associated with moving from one end of the board to another and back again, the maximum joint articulations were determined. They are 228.04° at the reach joint and 78.02° at the reach and base joints, respectively. When considered with the 2.5 second time limit for articulation, this equates to a rotational speed requirement of 91.22°/s at the reach joint and 31.21°/s at the base joint. This is equivalent to 15.2 RPM at the reach joint and 5.2 RPM at the base joint.

With an understanding of the rotational speed requirements of the arm, and having verified the kinematic movement of the system, a motion simulation analysis could then be reasonably conducted in Solidworks. This analysis assumed standard Earth gravity in the constant negative y direction and a constant rotational velocity of 15 rpm, a quantity verified based on our speed requirements. Furthermore, the simulation was conducted for the two extreme conditions of the arm's actuation, namely lifting a king piece on the farthest rank and lifting a pawn piece on the closest rank. The torque experienced by the motor was determined by rotating the system about the joint's center and rotating at a constant speed to avoid any unwanted additional system torques. This analysis yielded torque curves are shown in Figure 23 on the following page, showing torque versus time over a nominally-chosen one second span that includes key moments of inflection in the system.

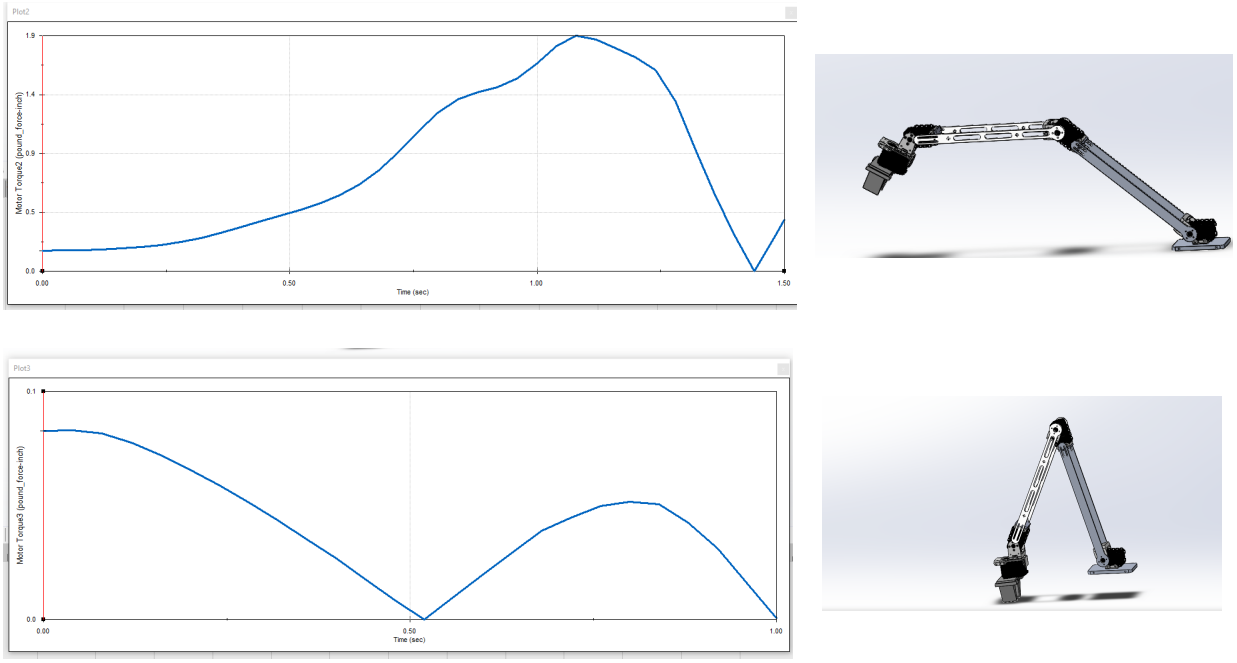


Fig. 23. The model of the mechanical arm was tested under constant rotational speed with standard gravity conditions to determine the torque experienced by the base actuator.

The test was conducted for the greatest reaching (left) and contracting (right) positions. The greatest torque experienced in either case was roughly 9 lb-in, or roughly 1.02 Nm.

Engineering Analysis | Rotational Speed and Torque Verification

With knowledge that the maximum torque experienced by the base joint will be roughly 1.02 Nm, and considering that the Dynamixel AX-12A actuator has a rated stall torque of 1.5 Nm [25], it was reasonable to state that the AX-12A would be a sufficient actuator for our needs at the base, reach, and gripper rotation joints. This assumption was able to be made because the greatest system loads for the mechanical arm will be experienced at the base joint, where both pairs of links and three actuators are applying torque on the system. At the reach joint and gripper rotational joints, significantly less mass is applying torque, so it is reasonable to assume that this actuator will be sufficient.

In addition to knowing that the AX12-A would provide sufficient torque to the system at these key joints, the speed requirements also needed to be considered. To evaluate if the actuator would permit the required rotational speed at this torque, a torque-speed curve for the actuator was graphed using the provided actuator specifications [25]. This plot can be found below in Figure 24.

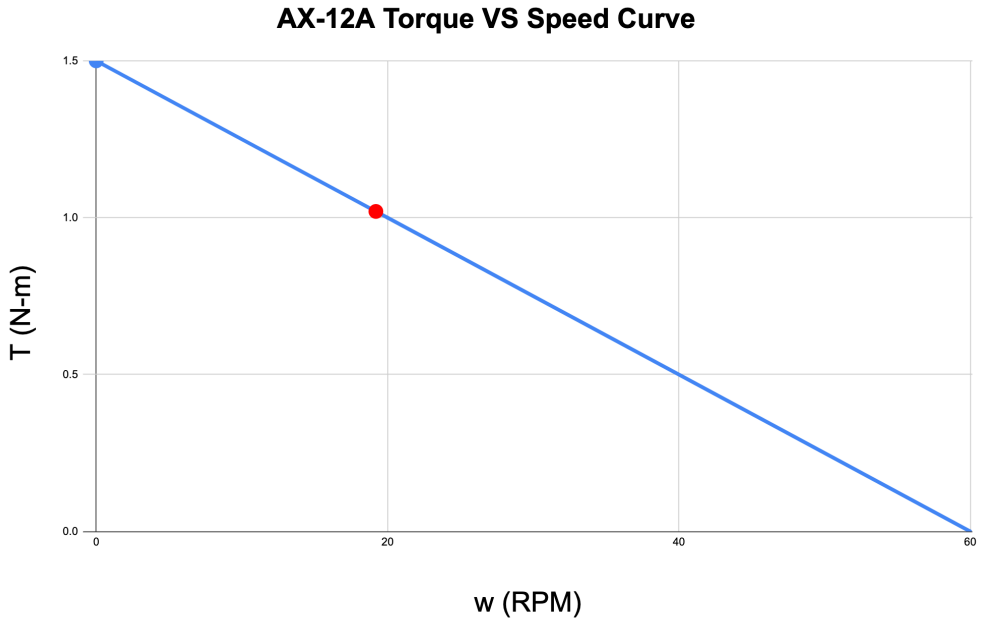


Fig. 24. The torque versus speed curve for the Dynamixel AX-12A actuator, shown in blue, with the operating point of our system at 1.02 Nm, shown in red. At this point, the motor can operate at a maximum speed of 19 RPM.

Given that the 19 RPM limit is significantly greater than the 5.2 RPM base joint speed requirement calculated previously, it was reasonable to state that the AX-12A would provide both the required torque and rotational speed necessary to fulfill the movement time requirements set forth in our problem definition phase.

Engineering Analysis | Center of Mass Location and Tipping Conditions

To ensure that the entire robot system would not tip during regular use, a center of mass analysis was conducted using the Solidworks model of the system, the results of which can be seen below in Figure 25.

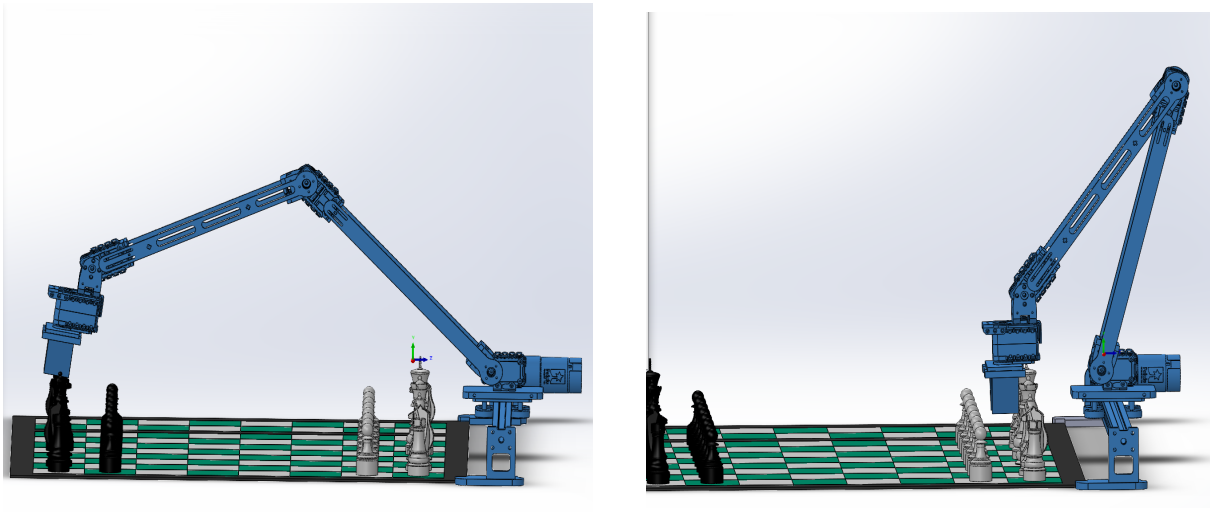


Fig. 25. The center of mass for the entire Chess Robot in both its extended and retracted positions with accompanying center of mass location. In the retracted position (right), the center of mass is located directly above the slider mechanism, and in the extended position (left), the center of mass is located roughly 2 inches from the slider base.

There was notable concern that in the extended position, the robot might have tipped over due to the system center of mass being extended beyond the robot's feet. To resolve this we extended the slider base feet to prevent the system from tipping over during normal operation.

Engineering Analysis | Reach Link Material Validation

The reach links of the mechanical arm were designed to be significantly lighter than the base links, both to limit the total torque required to actuate the system at the base and reach joints, and to prevent the center of mass from being too far from the slider base. The two primary candidates for reach link material were ABS plastic and acrylic. These have been chosen because they could be easily manufactured using 3D printing or laser cutting, were fairly cheap to acquire, and are significantly lighter than aluminum. To verify that both of these materials are valid candidates, a torque analysis under loaded conditions was performed on the reach link geometry. This analysis assumed that the link was parallel to the ground, such that the line of action from the joint to the application of force was greatest, maximizing torque. The link was assumed to be a fixed cantilever, with the maximum bending moment applied just beyond the joint itself. The system was also assumed to be accelerating from 0 to 15.2 RPM in 1 second. To convert this to translational acceleration, we consider that 15.2 RPM is roughly equal to 1.6 rad/s. We then considered the center of the 10 inch link is located 5 inches or 12.7 cm from the joint. This means that the tangential acceleration applied to the reach link is given by the Equation 2 on the following page

$$a_t = r * \alpha \quad (2)[26]$$

where alpha is the angular acceleration in rad/s^2, a_t is the tangential acceleration in m/s^2, and r is the radius, in meters. Given this equation, the tangential acceleration applied to the link at its center of mass is roughly 20.32 m/s^2 at the center of mass and roughly 40.64 m/s^2 at the location of the gripper. Given this value, and using mass and area moment of inertia values evaluated in Solidworks, the following diagram shown in Figure 26 can be constructed .

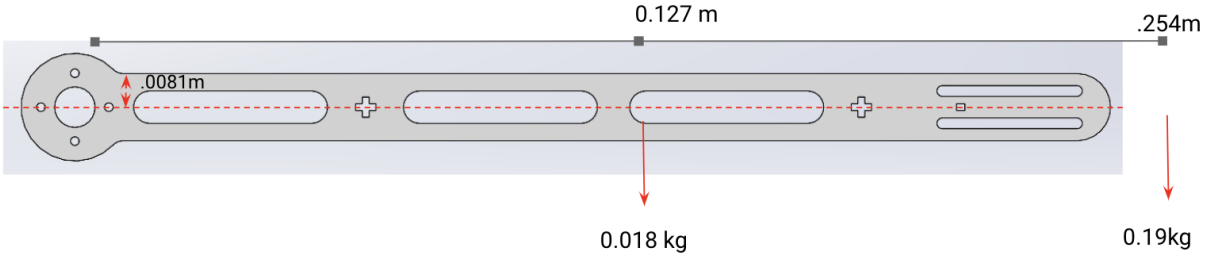


Fig. 26. The reach link from a side view, with a diagram of the applied forces and the distance from the reach joint to the point of application.

Given this diagram, we can calculate the total torque applied to the reach link at the location of maximum bending. This maximum moment is calculated using the following expression.

$$M = [0.127m * (0.018kg) * (9.8m/s^2 + 20.32m/s^2)] + [0.254m * (0.19kg) * (9.8m/s^2 + 40.64$$

Solving this equation for M gives a total moment just beyond the joint of 2.50 Nm. To find the total stress applied by this moment, we can use the equation for stress applied to a beam by a bending moment, shown in Equation 3:

$$\sigma = My/I \quad (3)[27]$$

where σ is the applied stress, M is the applied moment, y is the distance from the centroid to the point of application in m, and I is the area moment of inertia in m^4. Given this and considering a y value of 8.1mm as diagrammed, and an I value of 2193.54 mm^4 as calculated in Solidworks, we calculate the stress to be 3.057 MPa, or roughly 9.25 MPa. With this knowledge we compared the predicted applied stress to the yield strength of both materials to verify they are both valid choices for our reach links. This validation can be visualized below in Table 9.

Table 9. The applied bending stress of our reach link, and the yield strengths of ABS and Acrylic.

Reach Link Maximum Applied Stress	ABS Plastic Yield Strength	Acrylic Yield Strength
9.25 MPa	18.5 - 51 MPa [28]	64.8 - 83.4 MPa [29]

The information shown in this table clearly demonstrates that under maximum stress conditions, both the ABS plastic and the acrylic would not yield. This made them both viable candidates as materials for our reach links, but we selected acrylic because of its readily available stock material, and extremely simplistic manufacturing through laser cutting.

Base and Slider Mechanism

For the linear motion of the arm to navigate between squares, our team decided to mount the arm to cart on a slider track. There were many methods of linear movement, but we decided to go with this one due to its simplicity, and high speed relative to other options, such as a power screw. This high speed is important to meet our move timing for the requirement "Autonomously Make its Own Decisions When Playing a Move".

After the design of the arm, we realized that our budget would be tight for the base, so we sought out recycled parts. A previous ME 450 team constructed an inverted pendulum machine, but the project was unfinished and the parts were laying around the shop. With the permission of our advisor, we were able to recycle the parts used in their project for ours. This saved money on prototyping, and with additional validation served to supplement functional parts.

The structure of the base was composed from two legs made of aluminum plate, and an Actobotic Xrail which was supported by these two legs. A cart ran along the Actobotic Xrail, and was attached to a timing belt, which was driven by a motor on one of the two legs. The mechanical arm is mounted onto the cart. Figure 27 below shows the CAD model of the base.

Fig 27. The slider mechanism for the Chess Robot. The timing belt is not included in this figure.

Engineering Analysis | Slider Speed

To meet the move timing portion of the "Autonomously Make its Own Decisions When Playing a Move", we ensured that the robot was able to make a move within 7 seconds. It needed to move back and forth across the board a maximum of 3 times per move to ensure that it can handle the most complex action possible, which is capturing a piece with a pawn, and promoting it to a queen. This required the removal of 2 pieces from the board, and the placement of a piece from the sidelines onto the far side of the board. The board is calculated to require a maximum travel of 21.25 inches when accounting for the size of the slider and the extra columns of squares on the side that will be used for piece storage and placement of the chess clock. It would have to do this action three times, for each individual portion, so it would only have 2.33 seconds for that action. Therefore, the speed that the arm can move on the slider needed to meet or exceed 9.107 inches per second.

The motor that is being used is a BILDA 5201 Series with an attached encoder and a 26:1 gearbox. This model is discontinued from production, but came with the spare parts we acquired. The spare parts also came with a set of timing belt pulleys that have a radius of 0.5 inches from the center of the pulley to the center of the timing belt. The motor has a no load speed of 210 RPM, so assuming that the wheels on the slider cart are frictionless, we can estimate that the top speed of the arm when attached to this motor will be 10.99 inches per second, which was faster than the required speed to meet our requirements.

However, speed was not the only important factor to consider. The motor also needed to accelerate to that speed quickly, otherwise it would have been effectively much slower than the estimations suggest. To ensure that this doesn't cause any problems, we also analyzed the torque of the motor to ensure that it could reach top speed in a short period of time. Equation 4 below was created using a combination of Newton's second law, the motor torque-speed curve, and geometric formulas to model the acceleration of the mechanical arm on the slider system.

$$r * mx'' = t_{stall} - k\,(60\tfrac{sec}{min})\,\tfrac{x'}{2\pi r} \tag{4}$$

This differential equation was solved in Matlab to show the velocity over a period of time to see how fast the mechanism reaches top speed. Figure 28 below shows velocity of the mechanism in respect to time.

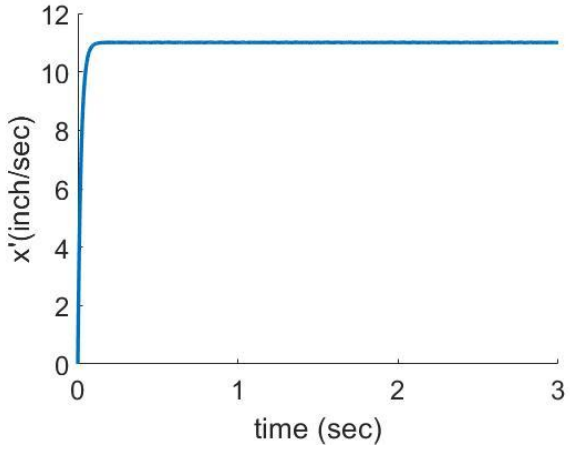


Fig. 28. Time versus velocity of the mechanism. As clearly shown, the mechanism reaches maximum speed within 0.15 seconds, and will not cause any significant reduction in the performance of the slider.

Design Changes For Manufacturing

Prior to manufacturing, the legs of the base require the holes to be relocated to match the holes on the parts provided from the recycled parts. Additionally, the base must be extended towards the player to prevent the robot from tipping during normal operation of the robot. Once these two modifications are made and analysis is redone with all of the changes taken into account, then drawings can be drafted of the parts. These mechanical drawings will be used for fabrication of a functioning prototype, which will be used to verify the robot's capability to fulfil the functional requirements and specifications.

Design Context Factor Analysis

An initial design context factor analysis was conducted for Design Review 3. An Engineering Inclusivity assessment was made in order to evaluate the Incorporation of Inclusive Design of our Stakeholder Interaction and Problem Definition and Design Decisions. An Environmental Context assessment was also conducted to evaluate two driving questions: *Does the system make significant progress towards an unmet and important environmental or social challenge?* and *Is there potential for the system to lead to undesirable consequences in its lifecycle, overshadowing benefits?*

Engineering Inclusivity

The Incorporation of Inclusive Design was the main evaluation point for assessing the Engineering Inclusivity of the Chess Robot project process. For Stakeholder Interaction, we assessed that due to the inherent student-driven nature of the Chess Robot project, stakeholders and their respective impact on the project would be limited. In order to combat this, we see the

potential to make an effort to bring in more stakeholders and increase the inclusivity of the design project. An example of this effort would be the recent addition of the Latin School of Chicago's chess team as a stakeholder. We hope that this addition increases the inclusivity of our design by allowing voices to be heard from the target audience of the Chess Robot product. We also identified that inclusivity could be increased by adding more mentor-like stakeholders and parents of differently abled children (specifically to gain more insight on how we could bring robot aided chess to everyone). During the Problem Definition and Design Decisions phases of the project, we made an effort to focus on the ideas of financial, usage, and geographic inclusivity. We intend to build a chess robot for as little money as possible such that anyone of any economic background could afford a robotic chess companion. Portability was an important part of these processes focusing on producing a lightweight and compact robot to promote inclusivity of children and differently abled persons. The robot is also marketed to all chess players in all locations and communities with the goal of maximizing long distance chess competition through a robotic physical medium.

Environmental Context Assessment

The Environmental Context Assessment of the Chess Robot project focussed on the evaluation of the questions: *Does the system make significant progress towards an unmet and important environmental or social challenge?* and *Is there potential for the system to lead to undesirable consequences in its lifecycle, overshadowing benefits?* We believe that the system makes significant progress towards the unmet and important challenge of excessive screen time being harmful for children. Having a physical chess robot to interact with gives children an alternative to playing chess on a screen. We also believe that building a system compliant with its requirements would result in a product that provides a physical chess companion over its lifecycle. There does not seem to be undesirable consequences at this point in time. In order to further evaluate the environmental impact of the system, an ADAMS power consumption analysis will be conducted and the results used to conduct a lifecycle eco-audit. In response to feedback from the Design Review 3 presentation, we have determined that the end-of-life procedure for the system will be to recycle or reuse plastic and metal components and electronic components will be taken to an electronic waste disposal facility for proper recycling.

Additionally, we conducted an Eco Audit to more accurately estimate the environmental impact and costs associated with producing the Chess Robot. The results of the eco audit are shown in Appendix D. Notably, the robot has a CO2 footprint of approximately 131 lb/year and total environmental energy burden of 7.27×10^{5}kcal/year with the majority caused from the "use" stage. The material, manufacture, and transportation impact is fairly standard.

Design Review and Verification

Risk Assessment

A detailed risk assessment was carried out by identifying potential risks and describing them, then assessing how each risk could be influenced by designers, how likely each risk was to occur, and how severe the impact of each risk would be. A general mitigation strategy was also proposed for each of the listed risks. This assessment was considered for both design-end and user-end concerns, including issues of safety, functionality, and affordability. Overall, most of the identified risks had a high-degree of designer control, but also a significant potential impact. With more time and resources, it is likely that most if not all of these risks could be significantly mitigated or eliminated. The complete risk assessment can be found in table form in appendix G.

Detailed Design Solution

The final design solution consisted of several key components that each played a key role in the overall functionality of the system. On the mechanical side, the key components consisted of the slider and traveller along with their corresponding motor, the arm links and associated actuators, and the gripper mechanism. The hardware components consisted of a Raspberry Pi Model 3, Arbotix-M Robocontroller, a bread board, and a limit switch. For testing purposes, we used an Arduino Leonardo to prototype both mechanical systems simultaneously. Lastly, the software consisted primarily of python scripts repurposed from the Raspberry Turk project by Joey Meyer, with some C code written for the slider mechanics. Figure 29 below shows the full assembly in its entirety.

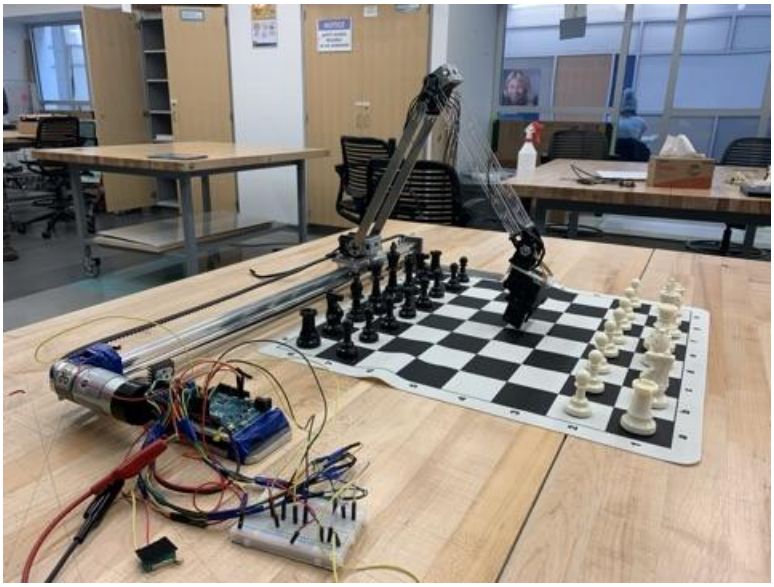


Fig. 29. Full chess robot assembly with all components attached.

Mechanical System

The following section will present the final mechanical system including the base and slider, the mechanical arm, and the gripper. The goal of the base and slider was to accurately and efficiently translate between files while balancing the weight of the mechanical arm given any of its possible configurations. The mechanical arm was tasked with operating within the full length of the chess board and having easy access to any rank on the board, while being able to provide sufficient torque at all positions to return to its original state. Finally, the gripper was responsible for picking up any piece with sufficient force such that it could move the piece to any square on the board.

As discussed earlier in the report, we wanted to use the slider to simplify the control of the mechanical arm while supplying the system with smooth translation in the x direction. The slider was manufactured in house and constructed to operate along an Actobotics X-Rail and driven by a rubber belt drive. The belt drive was powered by a servo motor which was programmed in C and operated by an Arduino controller. A simple limit switch was also contained in the circuit to recalibrate the slider after each successful move. The final slider design is seen below in Figure 30 and the base can be seen in Figure 31.

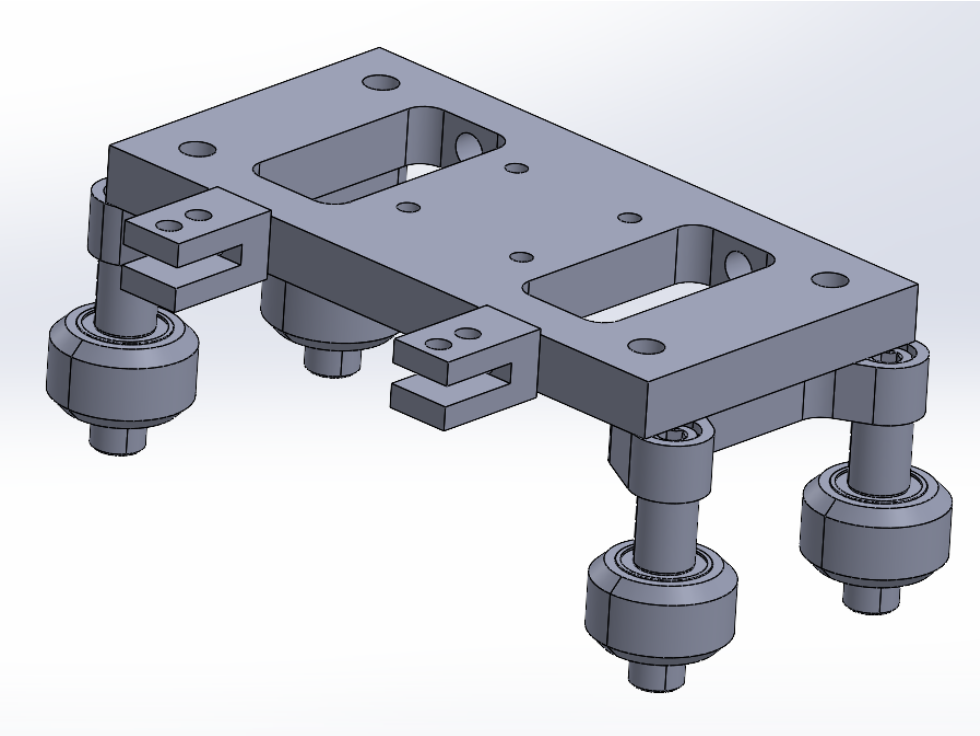


Fig. 30: Slider Cart subassembly CAD

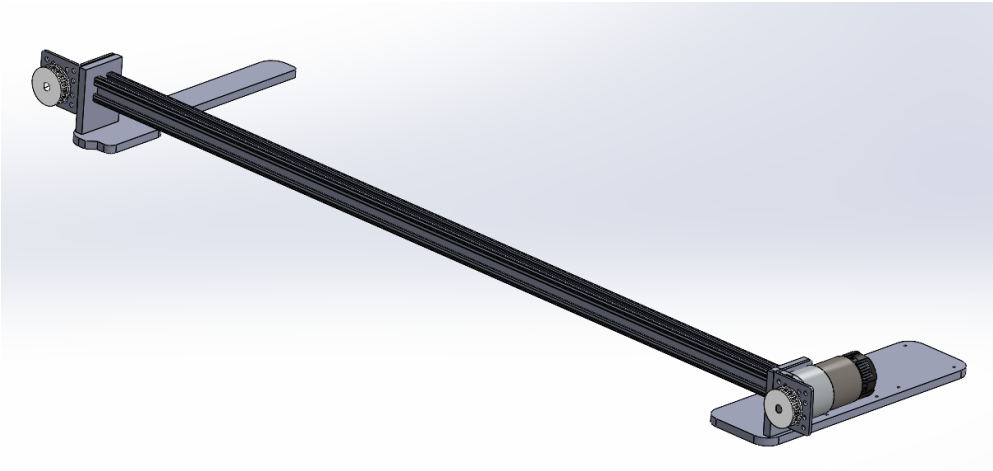


Fig. 31: Slider Base subassembly CAD

The mechanical arm design was constructed using two links of different material. The link closer to the base was made of Aluminum because it needed to bear a larger load and thus needed a strong material. The link farther from the base was made using Acrylic such that the motors would not have to provide as much torque to lift larger loads at the end of the arm. The shoulder, elbow, and wrist joints were all fastened such that the motors were mounted directly to them and could drive the links without any transmission. This strategy allowed us to avoid extra mechanical design and make a more concise overall system. The overall arm design was difficult to maintain and control, but it was very important to us that the robot resembled an arm to offer a bit more a personable experience. This design decision was factored into our pugh charts along with other reasons of why the arm and slider design was selected which can be referenced earlier in the report.

The actuators used for the arm were Dynamixel AX-12A Robot actuators which provided a fixed range from 0° to 300° and 0 to 1023 encoder counts. This resulted in a precision of 0.29°/count which allowed for the fine degree of control we were looking for. In order to utilize this precision, we needed to generate a conversion system which allowed us to convert from the encoder counts of the shoulder and elbow motors to the position in xz coordinates such that we could accurately pick up the chess pieces. The analysis on how that coordinate system was derived can be found in Appendix I and the final design of the mechanical arm is shown below in Figure 32.

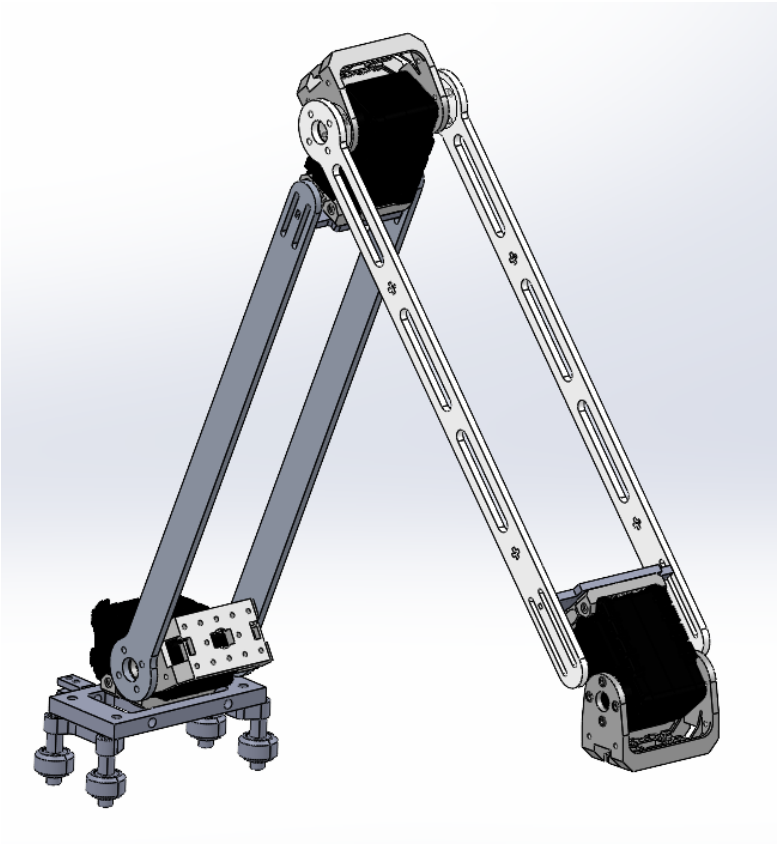


Fig. 32: Arm subassembly construction CAD

Finally, the gripper mechanism was attached to the end of the mechanical arm. It used the same actuator as the arm joints and was assembled in house using a kit supplied by the manufacturer. The gripper operated between 512 counts and 900 counts which gave it a smaller range of motion than the arm motors. Rather than being concerned with the degrees of rotation of the motor, we converted the counts to millimeters in order to accurately measure the amount the gripper closed per count. Given that the gripper when fully open was 32.91 mm with a range of 388 counts, we estimated that the precision of the gripper was approximately 0.239 mm/count. With this information, we were able to control how far the gripper closed based on which piece it was tasked with picking up. This process was touched upon previously in the software development section. Adhesive pads were also supplied as a part of the gripper and they were fitted to the inside of the plastic edges to allow for a bit more compliance as well as provide additional friction to ensure the pieces would not be released prematurely.

The gripper was initially meant to be positioned using a third Dynamixel motor which would act as the wrist joint. However, after running into technical issues, we decided to make a device that would always center the gripper over the square no matter the arm's orientation. We did this using a bolt and nut, which allowed the gripper to hang freely, and a rubber band to provide some damping as well as offset the gripper's center of mass since it was not symmetrical. This design

alternative ended up being very effective and worked as intended. The final gripper design is shown below in Figure 33.

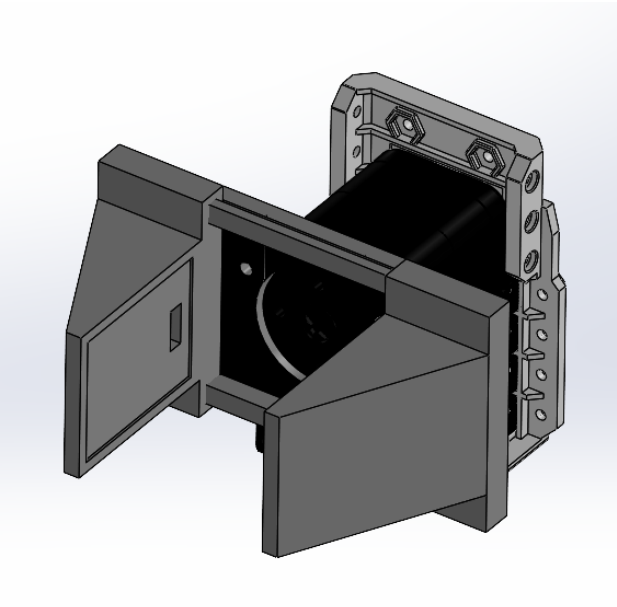


Fig. 33: PhantomX Parallel AX-12 Gripper CAD

Hardware Components

The three primary components of the circuitry are the Raspberry Pi 3, the ArbotiX-M robocontroller, and the H-Bridge circuit board. The Raspberry Pi 3 is powered with a 12 volt barrel plug, and sends the command information through a serial bus to the ArbotiX-M microcontroller. The Raspberry Pi is programmed using python. The ArbotiX-M is coded in C, and is the primary control component in the hardware. It takes in the serial output of the Raspberry Pi as commands and encoder information from the slider, and sends the signals required to run the Dynamixel servos and the slider motor. The ArbotiX-M runs a combination of code to convert the python input into commands for the Dynamixel servos, and a PID controller for the slider motor. The ArbotiX-M sends PWM information to the H-Bridge circuit, which both sends a PWM signal to the slider motor at 12 V, and also controls the direction of the motor for precise control. Figure 34 below shows the overall circuit arrangement.

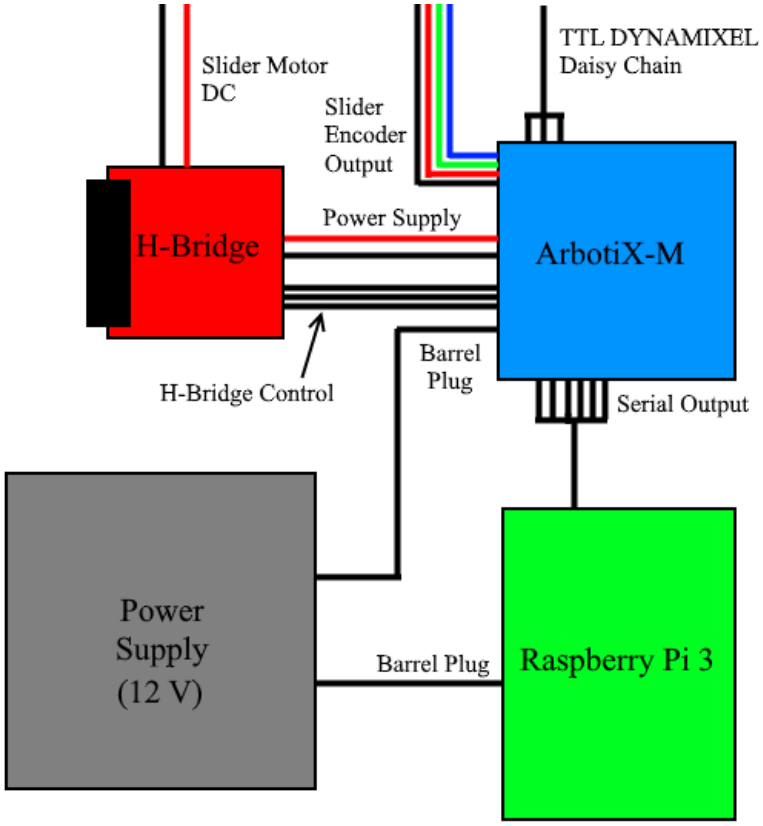


Fig. 34: Idealized hardware diagram design to control all components.

Software Components

The Software Development process culminated in a robust code base that accomplished all features as determined by the requirements and specifications of the project except for variable difficulty ratings. The general development process began with forking Joey Mayer's Raspberryturk source code repository. His code supported event driven automation with computer vision using a background daemon process (essentially a program that runs on its own with no user input). Due to lack of time and funds for the semester, computer vision was ruled out for our project. The first step in the development process was converting the Raspberryturk source code to support manual input for human moves instead of computer vision. This also meant that we wouldn't need the event driven daemon process and thus that was removed as

well. A basic user interface was developed and is shown below in Figure 35.

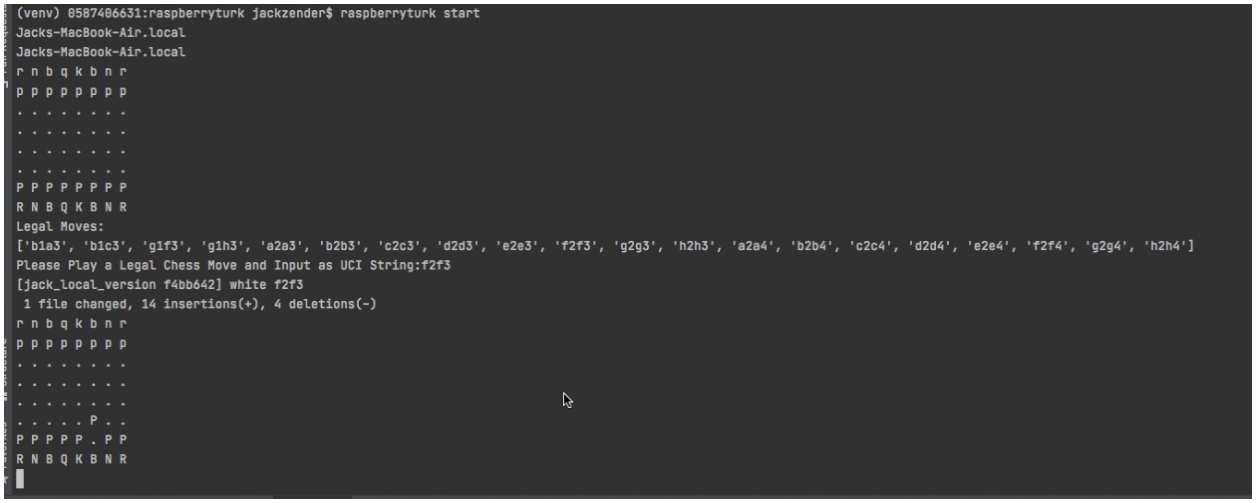


Fig. 35. User interface for manual input of human moves. Human inputs a move and the stockfish AI plays a move in return and the process repeats.

After development of the user interface, attention was immediately turned to the movement engine portion of the source code. The raspberryturk solved an inverse kinematics problem in the xy plane to map points on the board to encoder counts for it's two servos. Our implementation has an arm moving in the xz plane so this kinematics problem was converted to our coordinate system and the source code was updated accordingly. We then added support for the gripper which has a similar driver as the arm servos. Slider support was added using raspberry pi GPIO output to the arduino running C code sourced from ME350 that then takes instructions from the GPIO and converts it to positioning the slider to the right column on the board. After rigorous unit testing of the arm and slider code drivers. We focused our attention to updating the movement engine. The movement engine converts chess moves outputted by stockfish into usable instructions for the arm, gripper, and slider drivers. This is done by converting a move to a set of single movements (to square, from square, type of piece). Then converting the squares to a point for the arm to move to and a column for the slider to move to. The gripper then grabs the piece and the process is repeated to move to the right square to drop the piece off.

A flowchart for the software process is shown below in Figure 36.

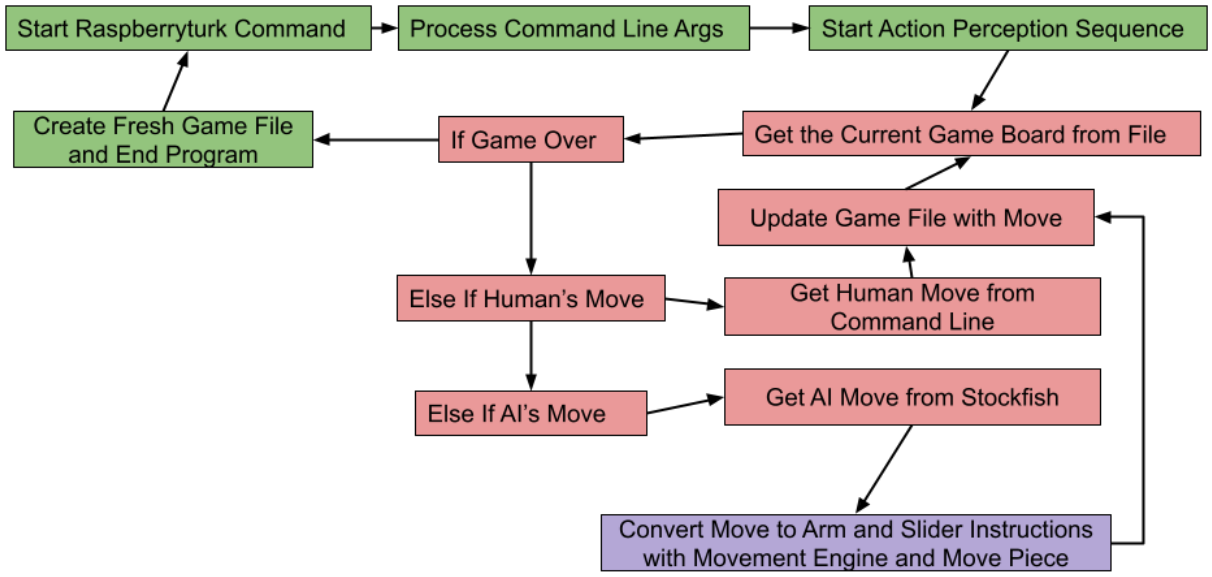


Fig. 36. Flowchart for the software component.

The directory tree for the project source code is shown below in Figure 37.

```
.
├── __init__.py
├── __main__.py
├── core
│   ├── __init__.py
│   └── game
│       ├── __init__.py
│       ├── human_player.py
│       ├── player.py
│       └── stockfish_player.py
└── embedded
    ├── __init__.py
    ├── agent.py
    ├── game
    │   └── __init__.py
    ├── motion
    │   ├── __init__.py
    │   ├── arm.py
    │   ├── arm_movement_engine.kdtree
    │   ├── arm_movement_engine.py
    │   ├── arm_movement_engine_pts.npy
    │   ├── coordinator.py
    │   ├── gripper.py
    │   ├── pypose
    │   │   ├── __init__.py
    │   │   ├── ax12.py
    │   │   └── driver.py
    │   └── slider.py
    └── raspberryturkd.py
```

Fig. 37. Directory tree for project source code.

A file by file breakdown of code functions is provided below in Table 10.

Table 10: Purpose of each file in the directory tree.

Filename	**Function and Purpose**
__init__.py	Set up file paths for use by the program
__main__.py	Command line processing and starts program
core/__init__.py	Placeholder for module compiling
core/game/__init__.py	Placeholder for module compiling
core/game/human_player.py	Support for human controlling of Robot's moves
core/game/player.py	Defines player class, human and stockfish inherit this class and defines its functions
core/game/stockfish_player.py	Support for ai controlling of Robot's moves
embedded/__init__.py	Placeholder for module compiling
embedded/agent.py	Executes perception-action sequence
embedded/raspberryturkd.py	Has support for event-driven daemon process but creates Agent object that then calls the perception-action sequence for our purposes
embedded/game/__init__.py	Contains helper functions for manipulating the board file and game (get_board, get_game, start_new_game, apply_move etc.)
embedded/motion/__init__.py	Placeholder for module compiling
embedded/motion/arm.py	Uses the arm movement engine to manipulate servos (move_to_point, set_speed, return_to_rest, etc.)
embedded/motion/gripper.py	Support for picking up and dropping pieces (grab_piece, release_piece, etc.)
embedded/motion/slider.py	Support for telling slider C code on arduino where to move (move_to_column, etc.)
embedded/motion/arm_movement_engine.py	Uses the .kdtree and .npy files to solve the inverse kinematics problem and converts points to servo instructions
embedded/motion/coordinater.py	Coordinates arm, gripper, and slider objects to execute a single movement (move_piece, execute_move, etc.)
embedded/motion/pypose	This folder contains the serial protocol used to communicate with the servos

The source code github links for this project and the unit testing environment is provided in Appendix H.

Verification

To verify that our project was successful, we were able to perform a demonstration of all of the individual components of the functionality of the robot. This included slider motion, arm motion, piece gripping, and chess engine implementation. This was done in person within the X50 lab on campus. From this, we were able to show that all of the individual components were functioning, and that remaining work that needed to be done to reach full verification was the integration of the three primary systems to work as one.

The chess robot currently does not meet most of the specifications chosen at the start of the project. A verification compliance matrix was used to determine whether or not we were able to meet our specifications. A verification compliance matrix was made as a tool for design verification and was used to make sure that all technical and stakeholder requirements are met. The verification compliance matrix is also a useful tool for the project team to stay up to date on the compliance of each requirement as well as the action needed to work towards compliance on a specific requirement. Another important aspect of the verification compliance matrix is that it allows stakeholders to see that their requirements are met with justification at a quick glance. Table 11 below shows the results of the verification compliance matrix, which can be found in full the Appendix D.

Table 11: Final Results of compliance matrix

Category	Count
Total # Requirements	**10**
Total # Compliant	**3**
Total # Intend to Comply	**4**
Total # Partial Compliant	**2**
Total # Non-Compliant	**1**

Each requirement reported as compliant as successfully tested based on metrics determined based on the original requirements and specifications. The portability requirement (R1) was met after finding the final mass of the robot to be 1.275 kg, and the overall depth footprint being 0.23 m. These two metrics ensure that a child could carry the chess robot and that it would fit on a shelf of standard depth. The “Function using standard chess board and pieces” requirement (R3)

was met by testing the gripper to ensure that it could grab every kind of piece (Pawn, Knight, Bishop, Rook, Queen, King) and be able to reach all ranks from 1 to 8. Finally, the chess robot met the “Receive Manual Input in Standard Algebraic Chess Notation From Human Player” requirement (R5) by being able to play a complete game of chess using standard algebraic notation within the Raspberry Pi testing environment.

Discussion and Recommendations

Creating the Chess Robot as our Mechanical Engineering Capstone Project was a well-rounded culmination of the University of Michigan Mechanical Engineering undergraduate education. We aimed to solve a real world problem by creating a tangible solution. Through this process we utilized an array of our developed skills and experiences to help guide our decision making. Specifically, we harnessed concepts and experiences from many courses including static mechanics, dynamics and vibrations, controls, and prior design and manufacturing courses. For many team members, creating the Chess Robot proved to even better represent our time at Michigan by combining mechanical engineering with computer science. We chose a challenging systems integration problem since the majority of us have either the computer science major or minor and we wanted to find a successful way to involve it in our capstone project. Additionally, this capstone project was an open ended project in which we were able to combine our personal passions (of Chess) with our educational background. Considering all of these factors, we were excited to pursue this project and worked hard to successfully complete it.

While we set out to make a fully functional Chess Robot that could compete in an entire standard game against a human, we ultimately came up a little short. At the end of the semester we had created a successful mechanical system and controls for many movements for both the arm and the slider, but failed to combine everything into a fully functional prototype. Specifically, our robot can make individual moves in each respective file (or column) on the board but currently is not set to move across columns to make a non-vertical move. The primary reason for this is because we were unable to properly integrate the slider mechanism with the arm. So while we can move the slider between individual files and can move the arm across individual ranks (rows), we cannot accomplish them simultaneously. This primary problem stemmed from a few different factors including working with a burnt-out robocontroller, difficulty controlling the individual subsystems, a broken actuator, and communicating across languages and hardware. We did not anticipate as many uncontrollable issues occurring and our schedule was set back significantly. While we planned out our schedule aggressively throughout the semester, these problems caused significant setbacks specifically at the end of the semester. One primary lesson we learned and would have redone is working to finish creating the mechanical system as early as possible to leave buffer time for the inevitable mechatronics problems. The mechanical design and manufacturing was effective and according to schedule, so in hindsight, we would reallocate the time to leave more room for control. Another significant unanticipated problem was the overworking of our base actuator. The actuator that rotates the entire arm would often overheat

and shut itself down. During our solution development and engineering analysis phase, we confirmed that the actuator had enough torque to lift the arm at its fully extended state in the required time, but we neglected to consider that it would be constantly used. Upon redesign, we would add a larger safety factor for max torque and would likely use the Dynamixel MX-64 actuator for the base actuator. The Dynamixel MX-64 would be a successful alternative as the controls and integration would remain similar and it is suited to operate at our required max torque for longer periods of time (specs show that it can operate successfully at ⅕ * stall torque, or 1.46 Nm which is greater than our max torque requirement). As each of these problems arose, we were extremely agile in debugging, coming up with creative solutions, and were determined to make it work. We continually challenged our engineering thinking, analysis, and creativity.

We reflected on this progress and despite not meeting many of the requirements that we had defined, we determined that our project was a success nonetheless. At its core, the Chess Robot is a compact electromechanical system that, with the proper controls, can play a game of chess. At its current state, it can do many individual tasks and with a few more weeks, we are confident that the Chess Robot would be nearly fully functional. Notably, the gripper can grip each piece, the arm can move to the correct position with increasing accuracy, and the slider controls are improving as well. As a team we believe we developed a robust mechanical design, highlighted by firm joints, minimal pinch-points, and plenty of room for adjustability. Upon redesign we would have used a more powerful actuator, created a 3D-printer cover for appearance and cable management, and worked to create a less-wobbly slider-roller.

Taking the feedback from peers, instructors, stakeholders, and ourselves, we aim to continue integrating our subsystems and aim to finish developing the Chess Robot over the summer. Throughout this semester, we have gained skills in project management, problem definition, creative solution generation, engineering analysis, manufacturing, and control. Even more, we learned to manage our aggressive goals by way of tiered requirements, organized and efficient team meetings, and dedication from each team member. We managed to create a semi-functional Chess Robot with primarily virtual communication and limited lab hours, while managing to stay safe and healthy. The Chess Robot proved to be a successful capstone project that we are thrilled to finalize in the months to come.

Engineering Standards

Our team considered a few standards during our concept generation process. We wanted to make sure that our robot would follow the OSHA guidelines for robotics safety to the best of our ability, while also keeping in mind the power of the robot. Many of the standards are designed around large industrial robots, but some of the guidelines like proximity sensors, hard stops, emergency stop buttons, and emergency braking. For the full listing of the engineering standard that we looked at, refer to Appendix K.

Conclusion

Our team has completed the Problem Definition and Concept Exploration components of the ME 450 Chess Education Robot project. As part of the Problem Definition component, we have defined our Problem Statement to be "*Continue developing Chess education while minimizing online-chess's added screen time and keeping its long-distance multiplayer benefits".* We have conducted stakeholder engagement and outreach. We interviewed Premier Chess CEO, Evan Rabin, receiving critical feedback that helped us prioritize our requirements and specifications and intend to conduct interviews with parents to gain more information about pricing requirements. We've developed a comprehensive list of Requirements and Specifications using a three tiered approach. The "Must-Have" requirements are a set of requirements fundamental to the functionality of the Chess Education Robot. The "Nice-to-Have" requirements are upgrades on some of the "Must-Have" requirements that could be completed in the semester if time permits. The "Long Term and Aspirational" requirements are requirements that bring advanced functionality to the Chess Education Robot but are unrealistic to achieve in the scope of one semester. As for the Concept Exploration component, we decided to take a subsystem approach to concept generation by developing functional decompositions for the mechanical and software subsystems. We then conducted brainstorming sessions and techniques to postulate solution ideas for each subsystem. These ideas were iterated on and sketched out various design ideas in the concept development phase. Lastly, in the concept and evaluation phase, we used pugh charts to determine final design decisions for the various subsystems. The next step for the project is the Solution Development phase, where we begin developing physical solutions for the project. The Overall Solution Development Approach was to develop the various subsystems in parallel and to produce initial virtual prototypes in Solidworks. A physical mockup was generated to help all team members visualize the same system. Then, each subsystem was iterated on and engineering analysis was conducted in order to drive a final system solution. After arriving at a final solution, we manufactured and assembled the system before moving on to controls and testing. Ultimately we produced a robust electromechanical system and created efficient controls for many individual motions. Due to numerous unanticipated problems with sourced items and in control, we were unable to fully integrate all of the desired functionality. Over the upcoming months, our team plans to continue the design process and finish created a fully autonomous chess robot.

Authors

Sam Goldman

Sam Goldman is a senior Mechanical Engineering student with a minor in Computer Science. He grew up in Chicago, Illinois where he was drawn to math, science, and chess. Sam founded his high school chess team and competed in the Illinois State Chess tournament. He aims to combine his passions for engineering and chess in this student initiated project. Sam has previously engaged in biomechanical research with the goal of determining if elderly subjects were at risk of falling. Most recently, Sam worked as a hardware engineering intern at GE Healthcare where he designed an alignment fixture stand for CT assembly. Sam served as the president of his fraternity and is involved in the Jewish Heritage Program on campus. Outside of engineering and his campus involvement, he enjoys competing in sports, trying new food, and most of all, playing chess.

Andrew Kwolek

Andrew Kwolek is a senior dual degree student in Mechanical Engineering and Computer Science. He grew up in Northbrook, Illinois where he gained an affinity for math and science at a young age. Upon graduation, Andrew plans to use his experience to pursue a career in autonomous robotics. Andrew has worked in both independent research, where he designed a subsystem for a fully robotic prosthetic ankle, and on a design team, where he was a member of the University of Michigan Solar Car Team. Most recently, Andrew worked as a robotics intern at Sarcos Robotics in Salt Lake City, UT, where he aided in controls development of a full-body exoskeleton. Outside of his professional life, Andrew volunteered as a COE peer mentor, where he guided incoming freshmen through their first semester of undergraduate life. He also enjoys watching sports, and making music in his free time.

Kenji Otani

Kenji Otani is a senior in mechanical engineering with a minor in computer science. He grew up in the metro Detroit area. Kenji has been fascinated by robotics since he was a child, and wants to pursue a career in mechanical design, mechatronics, and robotics. He has tailored a career path that he hopes will help him design robots for in the future. Kenji has worked on both the SAE Baja and SPARK electric racing design teams located in the Wilson Center. Additionally, he has worked with the student chapter of ASME over the duration of his time at the University of Michigan, climbing to the External Vice President of the chapter. Outside of engineering, Kenji is also the vice president of a student band that plays music inspired from modern japanese culture, and is also a participant with the Kendo Club at the University of Michigan, competing against students of other clubs and universities across the midwest.

Ian Ross

Ian Ross is a senior studying mechanical engineering with a minor in business. He grew up on Long Island, New York. Ian has experience with robotics dating back to middle school and has played chess for most of his life. He is working towards a future that allows him to employ his engineering and business studies to solve big-picture problems and better the communities he is a part of. Additionally, he is a member of Fraternity and Sorority Life at Michigan, having served as the president of his chapter of Beta Theta Pi and of Michigan's Interfraternity Council. Ian also volunteers with the Michigan Men program, a branch of SAPAC focused on peer-facilitated dialogues about masculine identity, healthy relationships, values-driven communities, and personal wellness.

Jack Zender

Jack Zender is a senior studying mechanical engineering with a minor in computer science. He was born in Chicago and moved to Ann Arbor in 2010 where he attended high school and then the University of Michigan. He is very interested in mechanical theory, controls systems, and using computer science as a way to enhance mechanical engineering solutions. Jack was able to put both his mechanical engineering background and computer science skills to use last summer while interning for Traxen. a small startup based out of Plymouth, MI. He worked closely with the controls team developing software for the intelligent cruise control product IQ-Cruise for semi-trucks. Jack's largest achievement during his internship was developing a python GUI script that built off of existing code to provide an easy to use tool for converting raw CAN trace data into CSV format for further analysis. After graduating, Jack hopes to build a career in a space where he can use all of his skills to come up with unique solutions to whatever problem he's given.

Works Cited

Information Sources

[1] Gendler, Alex. “A history of chess.” *TED: Ideas Worth Spreading*, Sep. 2019, https://www.ted.com/talks/alex_gendler_a_brief_history_of_chess?language=en

[2] Sutter, John D. “Trouble Sleeping? Maybe It's Your IPad.” CNN, Cable News Network, 13 May 2010, www.cnn.com/2010/TECH/05/13/sleep.gadgets.ipad/index.html.

[3] 11 Min Read Children's Health. “What Does Too Much Screen Time Do to Kids' Brains?” *NewYork-Presbyterian*, 5 Oct. 2020, healthmatters.nyp.org/what-does-too-much-screen-time-do-to-childrens-brains/.

[4] Evan Rabin. Personal Interview. 01 February 2021.

[5] Parents, Anonymous. Personal Interview. 06 February 2021.

[6][18] 朱其罡 . *Robot for Playing Chess*. 28 May 2014.

[7] 江兴方 , et al. *Chinese Chess Robot Device Based on Real Chessboard Man-Machine Chess Playing*. 19 Oct. 2016.

[8] Quayle, Chris. *Chess Robot Full Video*, Chris Quayle, 23 July 2017, www.youtube.com/watch?v=svEf53gvRgc.

[9] Ding, Jialin. *ChessVision: Chess Board and Piece Recognition*. Stanford, web.stanford.edu/class/cs231a/prev_projects_2016/CS_231A_Final_Report.pdf.

[10] Barone, Cara. “How Heavy Is Too Heavy for a Child's Backpack?” *Sutter Health*, Sutter Health, www.sutterhealth.org/health/childrens-health/how-heavy-is-too-heavy-for-a-childs-backpack.

[11]”Shelf Design Guidelines” *Brezlin*, www.brezlin.com/design/shelvingguidelines.html.

[12] Team, Chess.com. “Chess Board Dimensions: Basics and Guidelines.” *Chess.com,* Chess.com, 5 June 2019, www.chess.com/article/view/chess-board-dimensions.

[13] Chaiyapol Kulpate , Raman Paranjape & Mehran Mehrandezh (2008) Precise 3D Positioning of a Robotic Arm Using a Single Camera and a Flat Mirror, International Journal of Optomechatronics, 2:3, 205-232

[14] “Chess Pieces and Their Weights.” *Chess Pieces: Triple Weighted vs Unweighted | Wholesale Chess*, 2021, www.wholesalechess.com/pieces-and-weights.html.

[15] Khan, Zak. “FAQ: How to Choose a Safety Factor so a Motor Design Lasts?” *Motion Control Tips*, www.motioncontroltips.com/faq-choose-safety-factor-motor-design-lasts/#:~:text=Most%20documentation%20and%20motor%20selection,to%20output%20enough%20energy%20for.

[16] “Algebraic Notation (Chess).” Wikipedia, Wikimedia Foundation, 26 Jan. 2021, en.wikipedia.org/wiki/Algebraic_notation_(chess).

[17] “Time Controls in Chess - Chess Terms.” Chess.com, www.chess.com/terms/chess-time-controls.

[18] Mikolyzk, Thomas. “US Chess Rulebook: The Official Rules of Chess, 7th Edition, Tim Just, Chief Editor.” *US Chess Federation*, United States Chess Federation, 19 July 2019, www.uschess.org/index.php/Official-Rules/US-Chess-Rulebook-The-Official-Rules-of-Chess-7th-Edition-Tim-Just-Chief-Editor.html.

[19] “Department of Labor Logo UNITED STATES DEPARTMENT OF LABOR.” *Guidelines For Robotics Safety | Occupational Safety and Health Administration*, 21 Sept. 1987, www.osha.gov/enforcement/directives/std-01-12-002.

[20] Ferreira, Diogo R. “The Impact of the Search Depth on Chess Playing Strength.” *ICGA Journal*, vol. 36, no. 2, 2013, pp. 67–80., doi:10.3233/icg-2013-36202.

[21] Chess.com https://www.chess.com/article/view/chess-variants

[22] Niklasf. “Niklasf/Python-Chess.” *GitHub*, 19 Feb. 2021, github.com/niklasf/python-chess.

[23] Meyer, Joey. “Joeymeyer/Raspberryturk.” *GitHub*, github.com/joeymeyer/raspberryturk.

[24] “About - Stockfish.” *Stockfish*, stockfishchess.org/about/.

[25] “Dynamixel AX-12A Robot Actuator.” *From Robotis*, www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx.

[26]OpenStax. “Angular Acceleration.” *Lumen*, courses.lumenlearning.com/physics/chapter/10-1-angular-acceleration/.

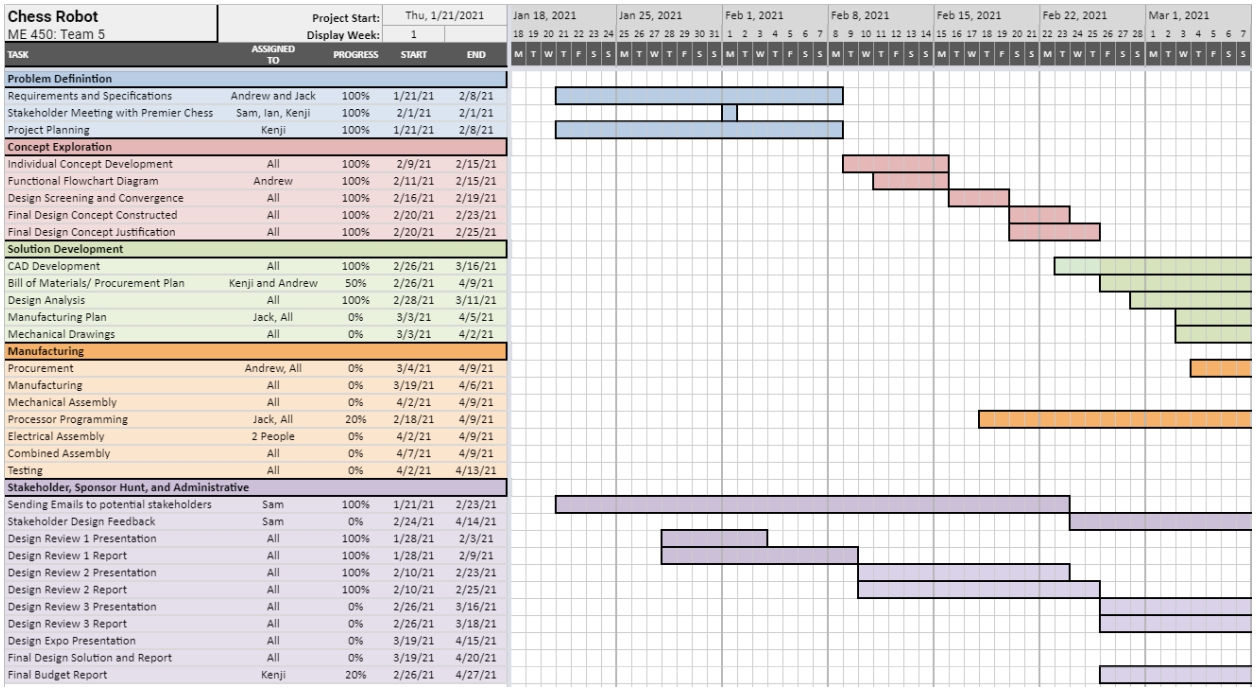
[27]“Calculate Bending Stress of a Beam Section: SkyCiv Cloud Structural Analysis Software.” *SkyCiv Cloud Structural Analysis Software | Cloud Structural Analysis Software and Calculators*, 18 Feb. 2021, skyciv.com/docs/tutorials/stress-tutorials/calculate-bending-stress-of-a-beam-section/.

[28] Dielectric Manufacturing. “Material Properties of ABS - Acrylonitrile-Butadlene-Styrene.” *Dielectric Manufacturing*, Dielectric Manufacturing, 24 Mar. 2020, dielectricmfg.com/knowledge-base/abs/.

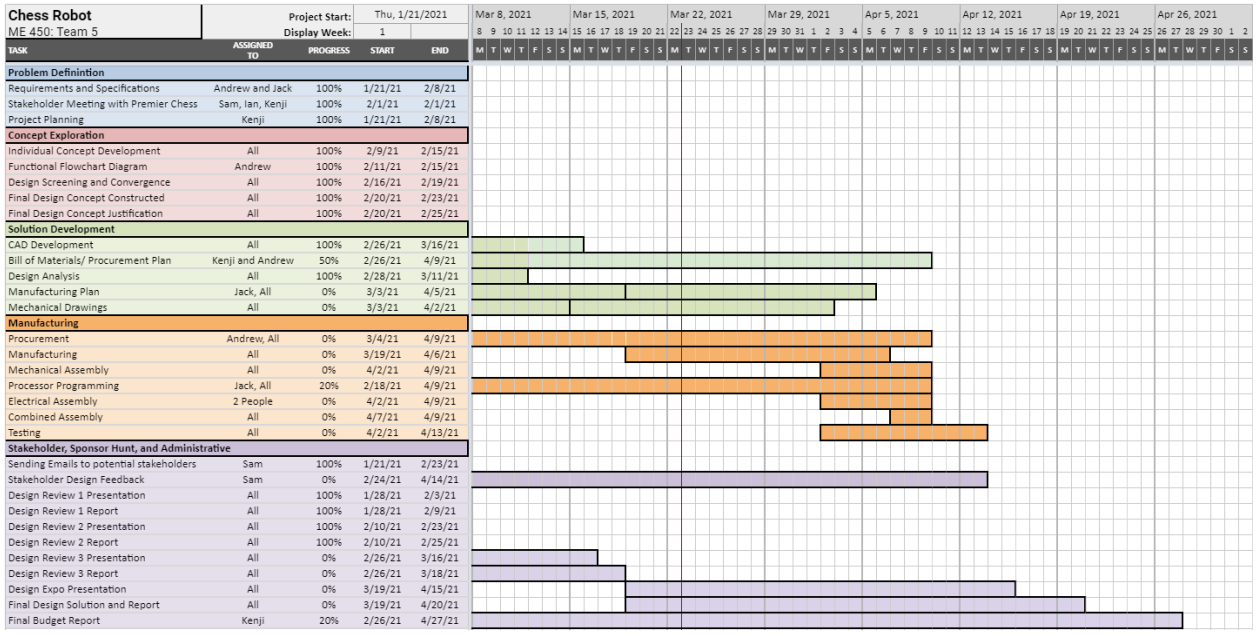
[29] “Overview of Materials for Acrylic, Cast.” *MatWeb*, www.matweb.com/search/datasheet.aspx?bassnum=O1303&ckck=1.

Appendices

Appendix A: Gantt Chart Enlarged

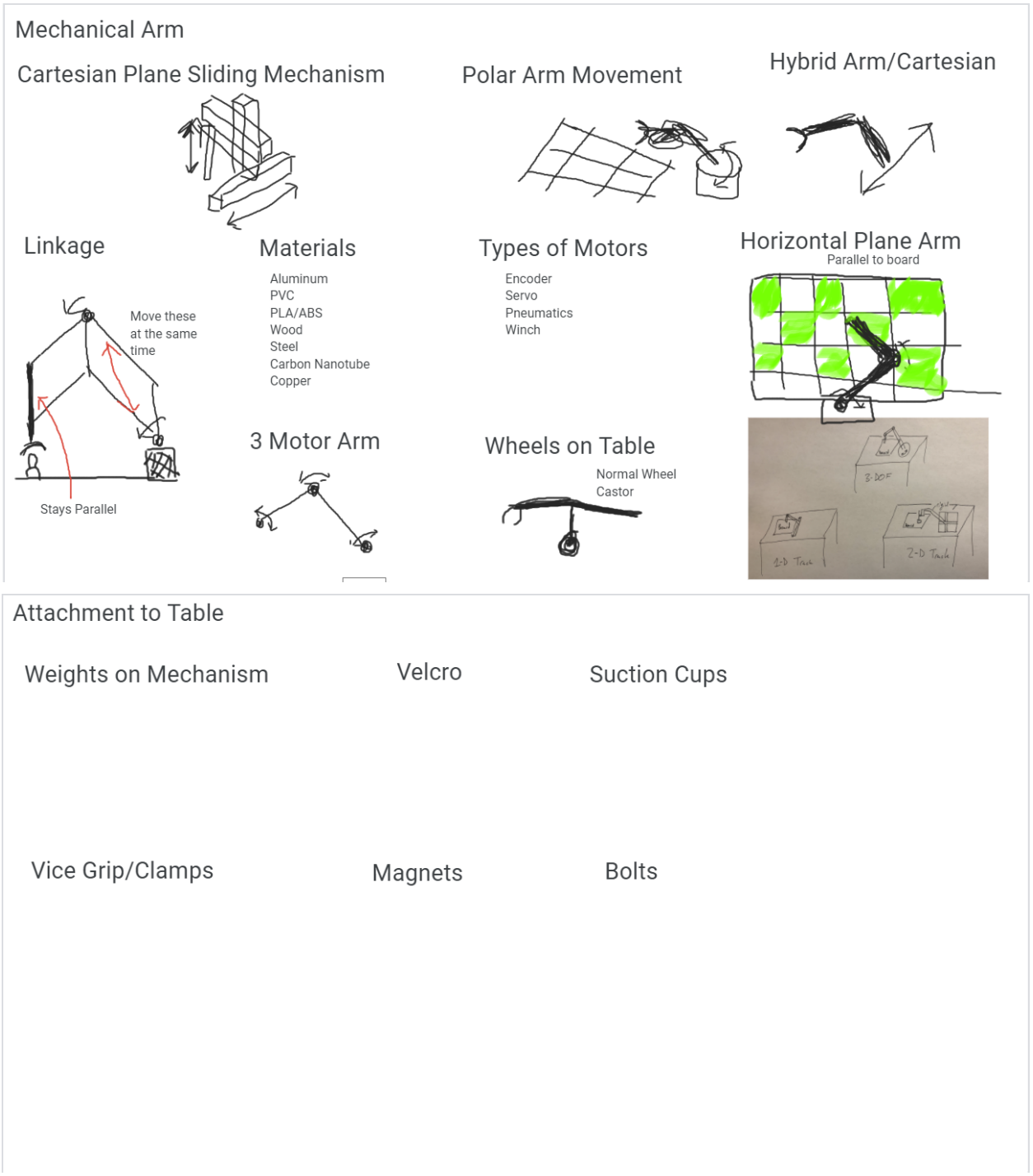


Chess Robot ME 450: Team 5		Project Start:	Thu, 1/21/2021	
		Display Week:	1	
TASK	ASSIGNED TO	PROGRESS	START	END
Problem Definintion				
Requirements and Specifications	Andrew and Jack	100%	1/21/21	2/8/21
Stakeholder Meeting with Premier Chess	Sam, Ian, Kenji	100%	2/1/21	2/1/21
Project Planning	Kenji	100%	1/21/21	2/8/21
Concept Exploration				
Individual Concept Development	All	100%	2/9/21	2/15/21
Functional Flowchart Diagram	Andrew	100%	2/11/21	2/15/21
Design Screening and Convergence	All	100%	2/16/21	2/19/21
Final Design Concept Constructed	All	100%	2/20/21	2/23/21
Final Design Concept Justification	All	100%	2/20/21	2/25/21
Solution Development				
CAD Development	All	100%	2/26/21	3/16/21
Bill of Materials/ Procurement Plan	Kenji and Andrew	50%	2/26/21	4/9/21
Design Analysis	All	100%	2/28/21	3/11/21
Manufacturing Plan	Jack, All	0%	3/3/21	4/5/21
Mechanical Drawings	All	0%	3/3/21	4/2/21
Manufacturing				
Procurement	Andrew, All	0%	3/4/21	4/9/21
Manufacturing	All	0%	3/19/21	4/6/21
Mechanical Assembly	All	0%	4/2/21	4/9/21
Processor Programming	Jack, All	20%	2/18/21	4/9/21
Electrical Assembly	2 People	0%	4/2/21	4/9/21
Combined Assembly	All	0%	4/7/21	4/9/21
Testing	All	0%	4/2/21	4/13/21
Stakeholder, Sponsor Hunt, and Administrative				
Sending Emails to potential stakeholders	Sam	100%	1/21/21	2/23/21
Stakeholder Design Feedback	Sam	0%	2/24/21	4/14/21
Design Review 1 Presentation	All	100%	1/28/21	2/3/21
Design Review 1 Report	All	100%	1/28/21	2/9/21
Design Review 2 Presentation	All	100%	2/10/21	2/23/21
Design Review 2 Report	All	100%	2/10/21	2/25/21
Design Review 3 Presentation	All	0%	2/26/21	3/16/21
Design Review 3 Report	All	0%	2/26/21	3/18/21
Design Expo Presentation	All	0%	3/19/21	4/15/21
Final Design Solution and Report	All	0%	3/19/21	4/20/21
Final Budget Report	Kenji	20%	2/26/21	4/27/21



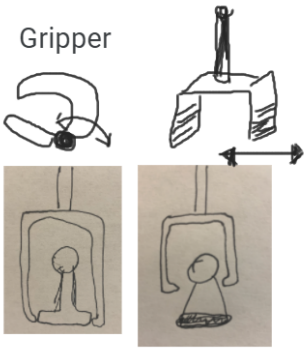
Chess Robot ME 450: Team 5		Project Start:	Thu, 1/21/2021	
		Display Week:	1	
TASK	ASSIGNED TO	PROGRESS	START	END
Problem Definintion				
Requirements and Specifications	Andrew and Jack	100%	1/21/21	2/8/21
Stakeholder Meeting with Premier Chess	Sam, Ian, Kenji	100%	2/1/21	2/1/21
Project Planning	Kenji	100%	1/21/21	2/8/21
Concept Exploration				
Individual Concept Development	All	100%	2/9/21	2/15/21
Functional Flowchart Diagram	Andrew	100%	2/11/21	2/15/21
Design Screening and Convergence	All	100%	2/16/21	2/19/21
Final Design Concept Constructed	All	100%	2/20/21	2/23/21
Final Design Concept Justification	All	100%	2/20/21	2/25/21
Solution Development				
CAD Development	All	100%	2/26/21	3/16/21
Bill of Materials/ Procurement Plan	Kenji and Andrew	50%	2/26/21	4/9/21
Design Analysis	All	100%	2/28/21	3/11/21
Manufacturing Plan	Jack, All	0%	3/3/21	4/5/21
Mechanical Drawings	All	0%	3/3/21	4/2/21
Manufacturing				
Procurement	Andrew, All	0%	3/4/21	4/9/21
Manufacturing	All	0%	3/19/21	4/6/21
Mechanical Assembly	All	0%	4/2/21	4/9/21
Processor Programming	Jack, All	20%	2/18/21	4/9/21
Electrical Assembly	2 People	0%	4/2/21	4/9/21
Combined Assembly	All	0%	4/7/21	4/9/21
Testing	All	0%	4/2/21	4/13/21
Stakeholder, Sponsor Hunt, and Administrative				
Sending Emails to potential stakeholders	Sam	100%	1/21/21	2/23/21
Stakeholder Design Feedback	Sam	0%	2/24/21	4/14/21
Design Review 1 Presentation	All	100%	1/28/21	2/3/21
Design Review 1 Report	All	100%	1/28/21	2/9/21
Design Review 2 Presentation	All	100%	2/10/21	2/23/21
Design Review 2 Report	All	100%	2/10/21	2/25/21
Design Review 3 Presentation	All	0%	2/26/21	3/16/21
Design Review 3 Report	All	0%	2/26/21	3/18/21
Design Expo Presentation	All	0%	3/19/21	4/15/21
Final Design Solution and Report	All	0%	3/19/21	4/20/21
Final Budget Report	Kenji	20%	2/26/21	4/27/21

Appendix B: Concept Generation Jamboard



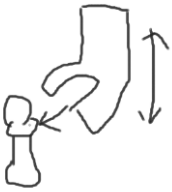
Piece Grabbing

Gripper



Magnetic

Scooper



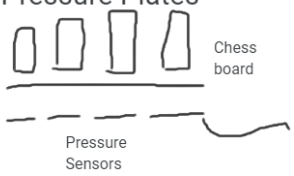
Expandable Rod

Universal Gripper

Locate Pieces

Computer Vision

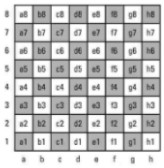
Pressure Plates



RFID on Pieces

Capacitor Sensors

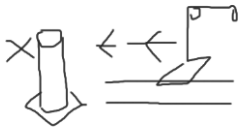
Notation Based Input



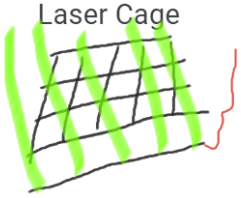
QR Code Stickers

Safety Features

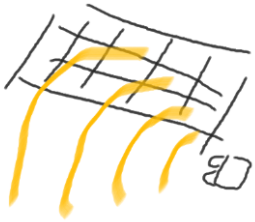
Hard Stops for ROM



Laser Cage

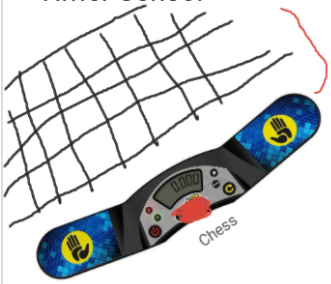


Ultrasonic Sensors



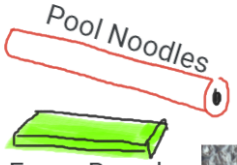
Torque Sensor

Timer Sensor



Limit Velocity

Padded Mechanism



Foam Board

Bubble Wrap

Panic Button

Appendix C: Preliminary Software For Chess Engine / API

Filename : chess_play_vs_stockfish.py

```
import chess
import chess.engine


def is_SAN():
    # TODO Takes: SAN string move Returns: if valid SAN move
    # takes: move string, Returns: if the move is a valid SAN move
    return True


def is_legal_move(move, board):
    # Takes, UCI move, board object Returns: if UCI move is legal
    return move in board.legal_moves


def get_stockfish_move(board):
    engine =
chess.engine.SimpleEngine.popen_uci("/usr/local/bin/stockfish")
    move = engine.play(board, chess.engine.Limit(time=0.1))
    engine.quit()
    return move


def get_move(player, board):
    # asks player for move in SAN, converts to UCI and checks if its legal
and returns UCI move
    while True:
        move = input("What is your move " + player + "? \n")
        m = board.parse_san(move)
        if is_legal_move(m, board) and is_SAN():
            return m
        print("Error: wrong move format or illegal move \n")


def main():

    board = chess.Board()
    while True:
        first_move = get_move("Player 1", board)
        board.push(first_move)
        if board.is_checkmate():
            print("Player 1, your move was", first_move)
```

```
            print("\nPlayer 1 wins")
            print(board)
            return 0
        second_move = get_stockfish_move(board)
        board.push(second_move.move)
        if board.is_checkmate():
            print("Player 1, your move was", first_move)
            print("Stockfish, your move was", second_move.move)
            print("\nStockfish wins")
            print(board)
            return 0
        print("\nPlayer 1, your move was", first_move)
        print("Stockfish, your move was", second_move.move)
        print(board)
        print("\n")


if __name__ == "__main__":
    main()
```

Appendix D: Eco Audit

Appendix E: Compliance Verification Matrix

Ref. #	Requirement	Specification	Compliance Status	Risk Level	Justification	Action
R1	Portable	Based on doctor's recommendations for max backpack weight [10] Should rest comfortable on a shelf with depth of 0.3 m [11]	Compliant	Medium	Currently compliant with weight and size but design isn't complete	Continue development and take steps to stay within portable guidelines
R2	Durable	Mechanism should be stable and not come loose from any motion Runtime of 2000 hrs accounts for 40 hrs a week for a year	Intend to Comply	Medium	Some durability analysis completed such as tipping point and yield strength but more needed	Determine initial analysis' to conduct to verify durability requirement (may require research)
R3	Function Using Standard Size Chess Board and Pieces	Robot plays on standard tournament [12] set to simulate a setting the player would realistically play in	Compliant	Medium	CAD Model Supports Standard Size Chess Board	Continue development and ensure functionality with standard board
R4	Autonomously Lift and Move Pieces In the Correct Orientation and to the Correct Location	Clearance and the deviation of pieces from the square's center based on research for other PID controlled robotic arms [13] 5 second move time recommended by stakeholder Piece weight range determined by double-weighted standard tournament pieces [14] with a 1.75x safety factor [15]	Intend to Comply	Medium	CAD Model and initial analysis show that it is possible to move any piece to any location but physical verification needed	Continue physical model development and software development to ensure proper piece moving
R5	Receive Manual Input in Standard Algebraic Chess Notation From Human Player	Robot should understand and use chess' universal notation [16]	Compliant	Medium	Basic software developed to take and parse manual inputs	Incorporate with raspberryturk source code
R6	Play Timed Chess Games	Timer interaction [17] is fundamental to competitive chess and the device should simulate that environment	Partial Compliant	High	Similar to Lift and Move pieces, this process is theoretically possible but no physical verification as of now	The largest hurdle will be software handling of timer interaction. Must research and spend time on piece of code handling this
R7	Play Game of Chess According to all Standard Rules	Important for device to simulate official tournament chess rules	Intend to Comply	Medium	As long as Lift and Move and Timed Chess Games requirements are compliant this will bank on software competency	Make sure software plays the game according to Chess Rules

R8	Autonomously Make its Own Decisions When Playing a Move	Crux of the problem; core component of overall functionality 3 seconds to “think”, 7 to move	Intend to Comply	Medium	Basic software developed to receive moves from stockfish	Incorporate with raspberryturk source code
R9	Have a Mechanical Structure that Allows for Safe and Restricted Movement	Needs to be safe for children to use when playing chess Cited as stakeholder priority	Partial Compliant	Medium	Safety measures incorporated in the CAD but no physical verification or safety testing as of now	Get physical system and software working then run safety test suite
R10	Vary Levels of Difficulty Prior to Game	The robot should be accommodating to players of different skill levels	Non-Compliant	High	Proving difficult to implement varied skill because of limitations in chess engine communication	Further Research into chess engine interaction and engine theory

Appendix F: Bill of Materials

#	Part Name	Qty.	Material	Source	Acquired	Est. cost
1	Left Foot	1	Aluminum	Manufactured	Shop Stock	$9.29
2	Right Foot	1	Aluminum	Manufactured	Shop Stock	$15.49
3	Base Leg	2	Aluminum	Manufactured	Shop Stock	$4.37
4	Slider Rail	1	Aluminum	Servocity + Manufactured	Purchased	$10.99
5	Slider Base	1	Aluminum	Manufactured	Shop Stock	$3.41
6	Base Link	2	Aluminum	Manufactured	Shop Stock	$7.29
7	Reach Link	2	Acrylic Plastic	Manufactured	Shop Stock	$0.50
8	Base Link Connector	1	Aluminum	Manufactured	Shop Stock	$0.69
9	Reach Link Connector	1	Aluminum	Manufactured	Shop Stock	$0.83
P1	Raspberry Pi 3 Model B	1	Circuit Board	Microcenter	Purchased	$29.99
P2	RobotiX-M Robocontroller	1	Circuit Board	Trossen Robotics	Purchased	$39.95
P3	Dynamixel AX-12A Robot Actuator x 3	3	Complex	Trossen Robotics	Purchased	$134.70
P4	FTDI Cable 5V	1	Cable	Trossen Robotics	Purchased	$17.95
P5	PhantomX Parallel AX-12 Gripper w/ servo	1	Plastic	Trossen Robotics	Purchased	$64.95
P6	Bioloid Frame F8	1	Plastic	Trossen Robotics	Purchased	$1.75
P7	Bioloid Frame F3 (Metal)	3	Aluminum	Trossen Robotics	Purchased	$14.85
P8	Bioloid Frame F2 (Metal)	1	Aluminum	Trossen Robotics	Purchased	$4.95
P9	V-Wheel Kit A	1	Steel + Plastic + Aluminum	Servocity	Recycled Project	$34.99
P10	X-Rail Roller Bracket (2 Pack)	1	Aluminum	Servocity	Recycled Project	$9.99
P11	90° Single Angle Pattern Bracket	2	Aluminum	Servocity	Recycled Project	$3.78
P12	0.250" 15 Tooth Pinion Pulley	2	Aluminum	Servocity	Recycled Project	$17.98

P13	1206 Series Pattern Adaptor (16-1)	2	Aluminum	Servocity	Recycled Project	$9.98
P14	63" x 3/8 in Timing Belt	1	Neoprene + Fiberglass	Servocity	Recycled Project	$11.85
P15	5202 Series Yellow Jacket Planetary Gear Motor (26.9:1)	1	Complex	Servocity	Recycled Project	$39.99
P15	XL Belt Mount B	2	Aluminum	Servocity	Recycled Project	$7.98
P16	Stainless Steel Round Shafting (1/4" Diameter x 2")	1	Stainless Steel	Servocity	Recycled Project	$1.09
P17	Screws	34	Steel	Hardware Store	Recycled Project	$3.00
					Total Cost:	**$502.58**

Appendix G: Risk Assessment

Risk	Risk Description	Level of Designer Control	Risk Likelihood	Risk Impact	Risk Mitigation Actions
User Safety	As a result of electircal components and moving parts, there are some safety concerns related to electric shock and impact injury	High	Possible	Major	Design and implement emergency stop button for arm and slider mechanisms, and develop insulated housing for all mechatronic components
Physical Stability	As a result of moving parts and varying center of gravity, there is some concern that the robot may tip over in normal operation	High	Unlikely	Moderate	Have designed and implemented extended base 'feet' for the slider assembly to prevent tip in the direction towards the chess board
Affordability	Due to early prototyping and a lack of cost estimates, there is concern about the financial costs of the robot being too high for the average consumer	Moderate	Likely	Moderate	Propose redesign of product that uses less expensive materials, or a mass-production model that significantly reduces the per unit manufacturing cost
Robot Play Speed	With consideration of 'edge-case' moves like castling, pawn promotion, or complex piece captures, there is some concern that the robot will take too long to move	High	Possible	Major	Redesign the control system to make more time-efficient moves and supplement more powerful actuators that can apply higher torques and move more quickly
Robot Play Difficulty	There is concern that the robot is currently not capable of playing at variable difficulties, thus not fulfilling a must-have requirement	High	Likely	Major	Through continued design a variable difficulty setting could be implemented by adding some pregame user input software and designing a physical, screenless interface for users to set play difficulty
Thermal Management	There is some concern that the control circuit and mechatronic components may overheat during extended use	High	Possible	Major	Begin by formally testing the thermal management of the existing system, then develop a ventilation and/or heat sink system to best cool the control boards
Movement Accuracy	There is some concern that the robot will have difficulty making precise movements and be able to accurately and repeatably execute directed moves	High	Possible	Major	To ensure accurate movement, tune the slider PID controller through extensive testing and develope a more rigid interface between the slider and arm mechanisms

Appendix H: Github links for Source Code

Project Source Code: https://github.com/jackzend/raspberryturk/tree/raspi_local_version

Unit Testing Environment: https://github.com/jackzend/arm_testing

Appendix I: Arm Coordinate System Analysis

The main function of this analysis is to convert the encoder counts of the shoulder motor s_1 and the encoder counts of the elbow motor s_2, to coordinates in the yz-plane.

$$s_1,\ s_2 = g(z,\ y)$$

This problem becomes a reverse kinematics problem, and since we know the lengths of the links, we can determine the coordinates in terms of the angles of rotation.

$$f_z(\theta_1, \theta_2) = z_{offset} + cos(\theta_1) \cdot l_1 + cos(\theta_1 + \theta_2) \cdot l_2$$
$$f_y(\theta_1, \theta_2) = y_{offset} + sin(\theta_1) \cdot l_1 + sin(\theta_1 + \theta_2) \cdot l_2$$

The length of the link connected to the shoulder motor is represented as l_1 and the length of the link connected to the elbow motor is l_2. The next step is to find the rotational offset of the motors given the position of the links. In other words, we must figure out how far the motor is rotated from its zero position given its mounted orientation. We did this by finding the distance between our axis and the motor's default axis. We found that the rotational offsets were as follows:

$$\theta_1^{'} = -\ 111.14^{\circ}, \theta_2^{'} = 237.35^{\circ}$$

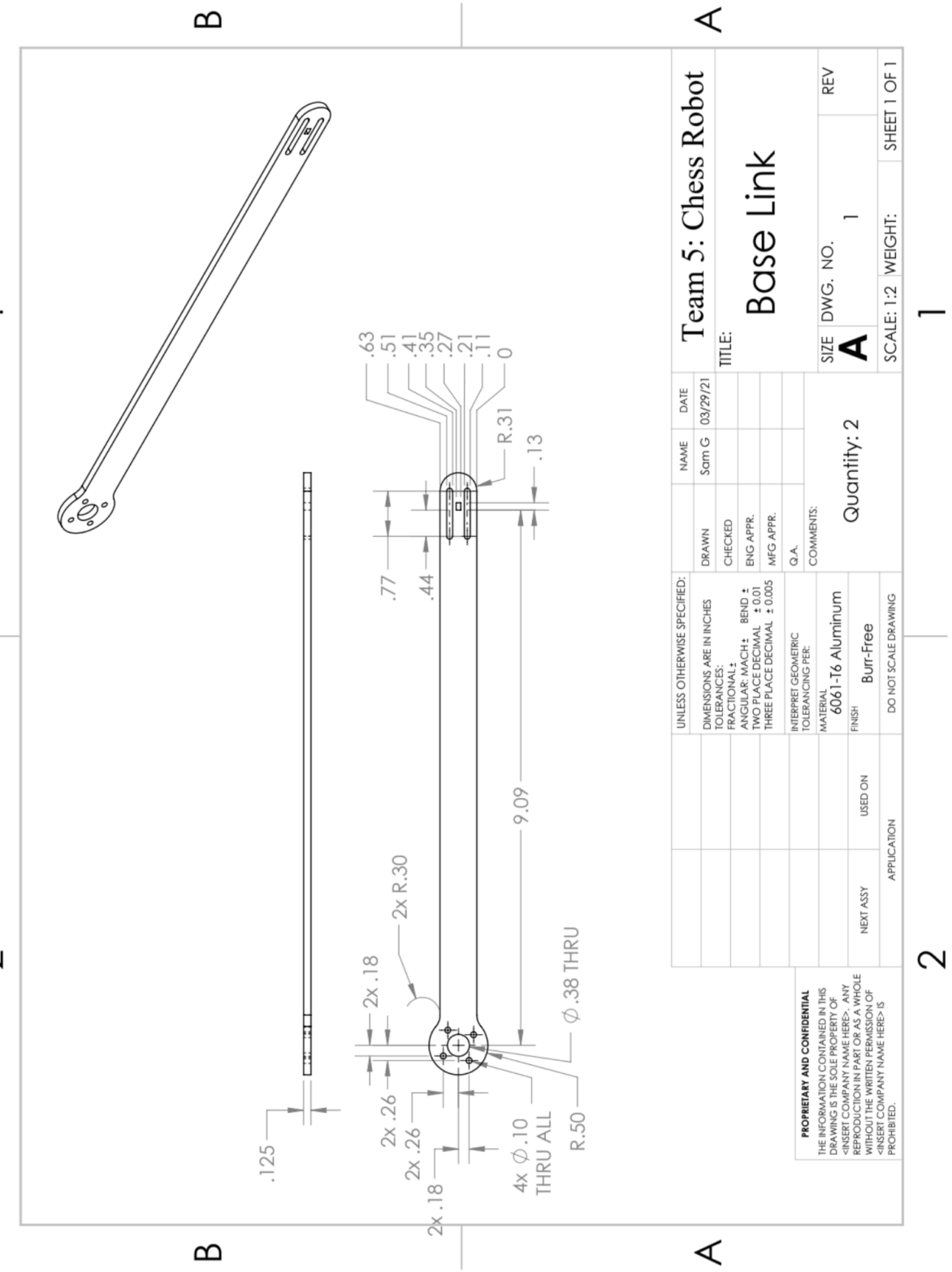
With the rotational offsets we can find the conversion for each servo from encoder counts to degrees of rotation.

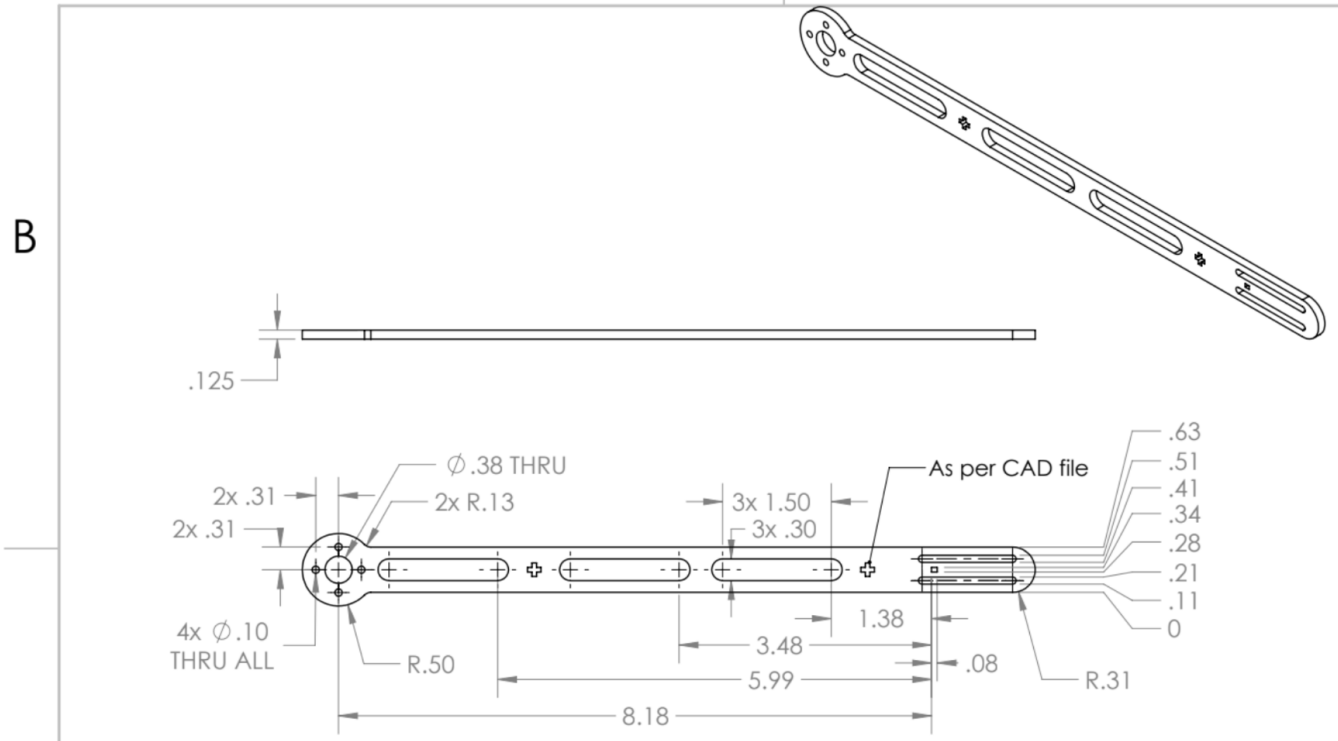
$$\theta_1(s_1) = \frac{1023 - s_1}{1023} \cdot 300^{\circ} + \theta_1^{'}$$
$$\theta_2(s_2) = \theta_2^{'} - \frac{1023 - s_2}{1023} \cdot 300^{\circ}$$

We can now find the final equations which convert the encoder counts of each motor to coordinates.

$$f_z(s_1, s_2) = z_{offset} + cos(\frac{1023 - s_1}{1023} \cdot 300^{\circ} + \theta_1^{'}) \cdot l_1 + cos((\frac{1023 - s_1}{1023} \cdot 300^{\circ} + \theta_1^{'}) + (\theta_2^{'} - \frac{1023 - s_2}{1023} \cdot 300^{\circ})) \cdot l_2$$
$$f_y(s_1, s_2) = y_{offset} + sin(\frac{1023 - s_1}{1023} \cdot 300^{\circ} + \theta_1^{'}) \cdot l_1 + sin((\frac{1023 - s_1}{1023} \cdot 300^{\circ} + \theta_1^{'}) + (\theta_2^{'} - \frac{1023 - s_2}{1023} \cdot 300^{\circ})) \cdot l_2$$

Appendix J: Mechanical Drawings and Manufacturing Plans





		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± 0.01 THREE PLACE DECIMAL ± 0.005	DRAWN	Sam G	03/29/21
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL Acrylic	COMMENTS:		
NEXT ASSY	USED ON	FINISH	Quantity: 2		
APPLICATION		DO NOT SCALE DRAWING			

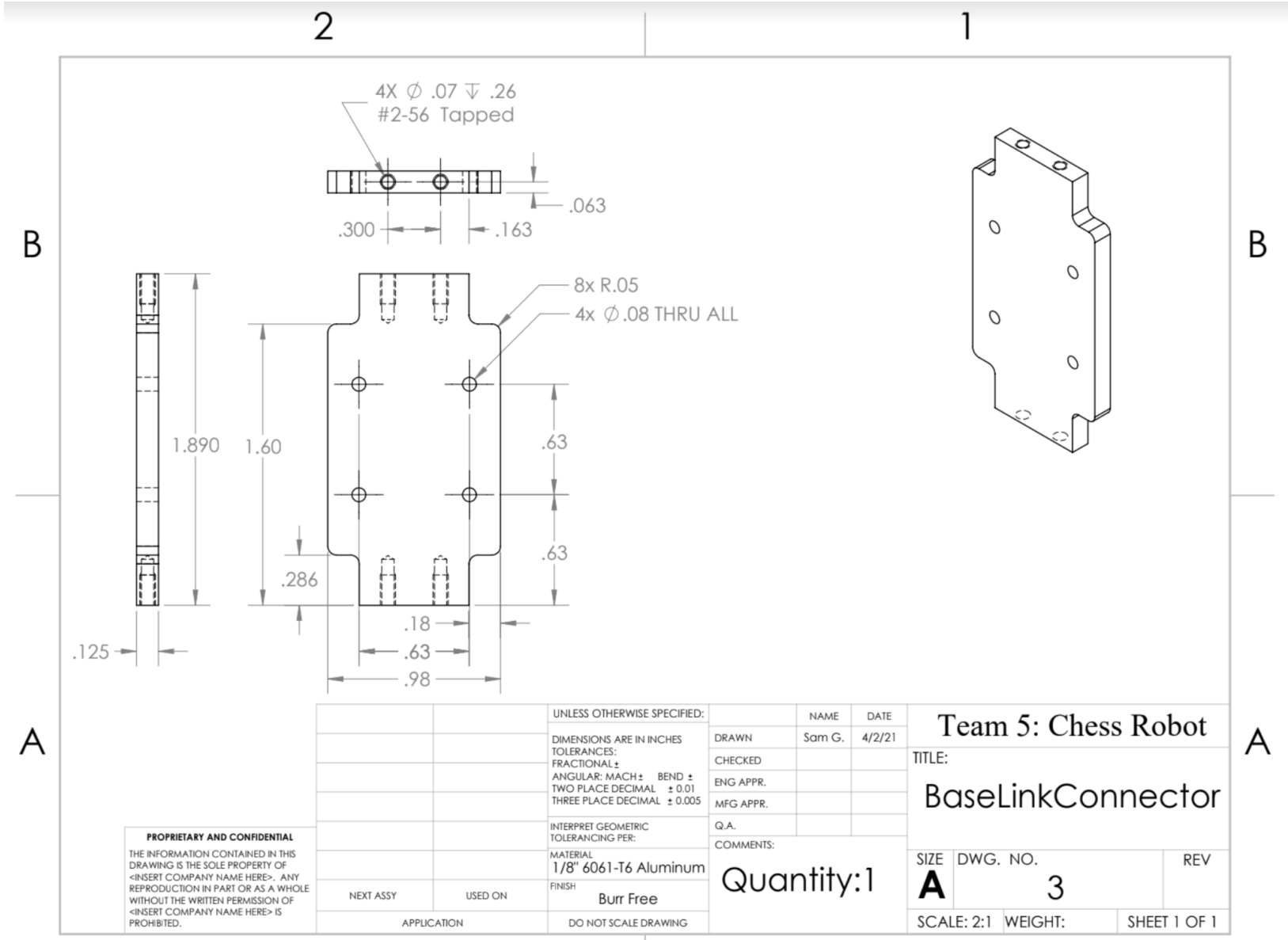
Team 5: Chess Robot

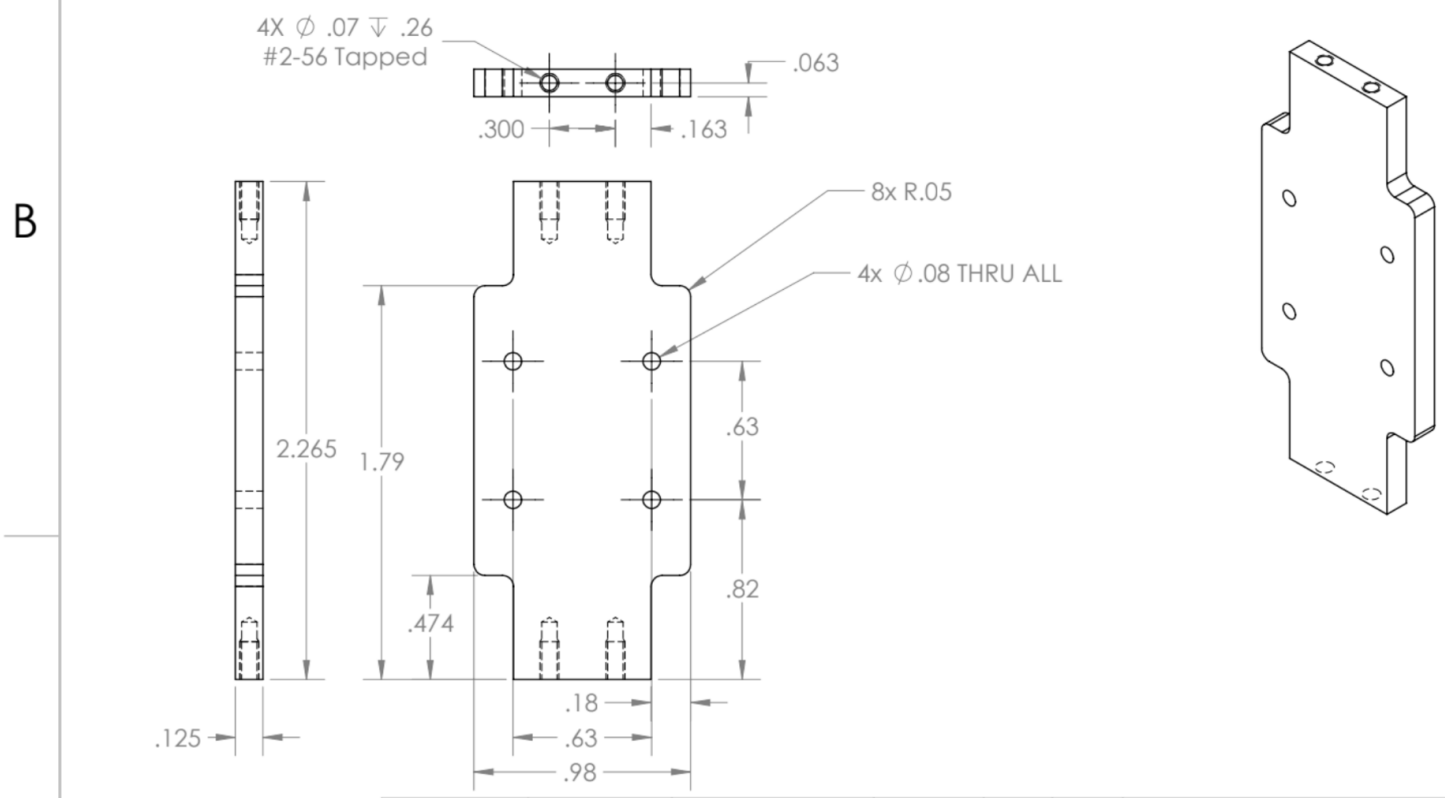
TITLE:

ReachLink

SIZE	DWG. NO.	REV
A	2	

SCALE: 1:2 WEIGHT: SHEET 1 OF 1


4X ∅ .07 ↧ .26
#2-56 Tapped
.063
.300
.163
8x R.05
4x ∅ .08 THRU ALL
1.890
1.60
.63
.63
.286
.18
.125
.63
.98
2
1
B
A
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ± 0.01
THREE PLACE DECIMAL ± 0.005
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
1/8" 6061-T6 Aluminum
FINISH
Burr Free
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
NAME
DATE
DRAWN
Sam G.
4/2/21
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
Quantity:1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
Team 5: Chess Robot
TITLE:
BaseLinkConnector
SIZE
A
DWG. NO.
3
REV
SCALE: 2:1
WEIGHT:
SHEET 1 OF 1



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± 0.01 THREE PLACE DECIMAL ± 0.005	DRAWN	Sam G.	4/2/21
			CHECKED		
			ENG APPR.		
			MFG APPR.		
			Q.A.		
		INTERPRET GEOMETRIC TOLERANCING PER:	COMMENTS:		
		MATERIAL 1/8" 6061-T6 Aluminum	Quantity: 1		
NEXT ASSY	USED ON	FINISH Burr Free			
APPLICATION		DO NOT SCALE DRAWING			

Team 5: Chess Robot

TITLE: ReachLinkConnector

SIZE	DWG. NO.	REV
A	4	

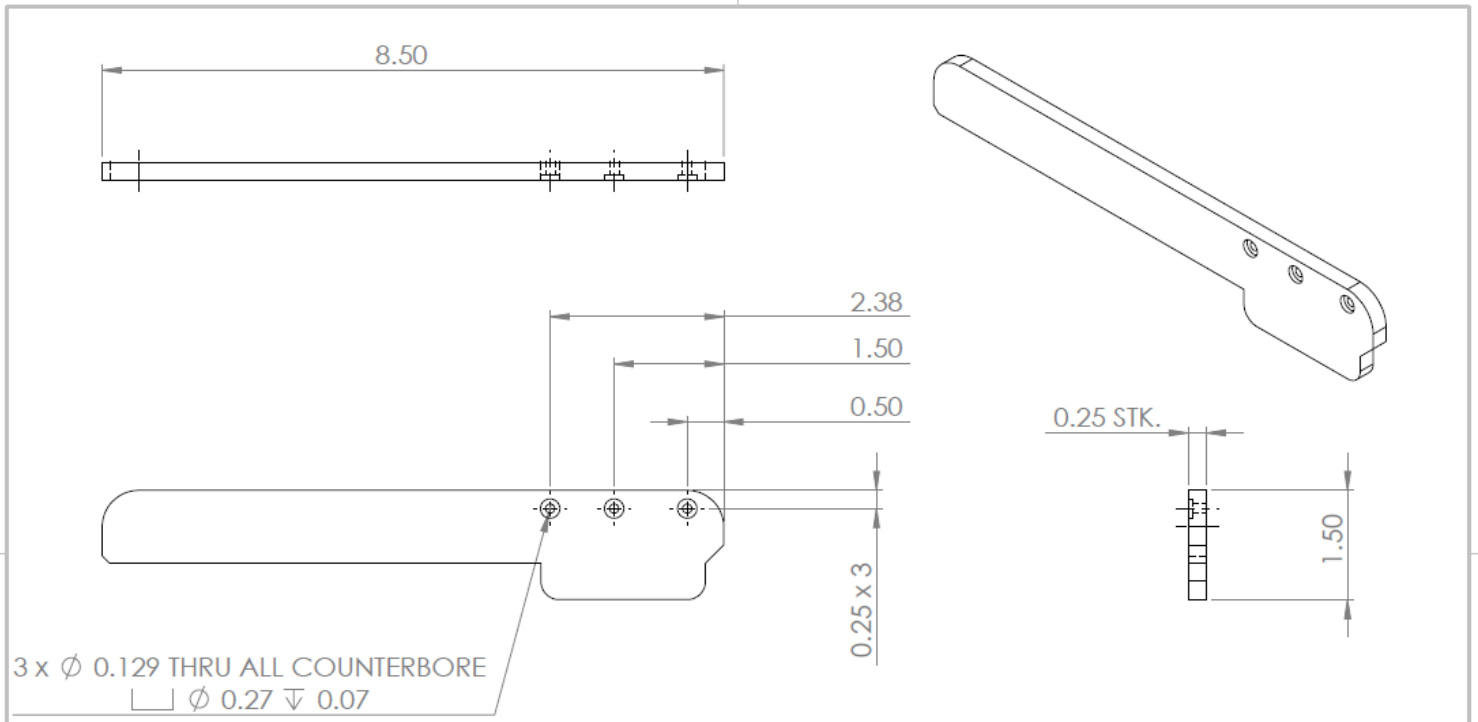
SCALE: 1:1 | WEIGHT: | SHEET 1 OF 1

Manufacturing Plan

Part Number: 3 and 4
Part Title: LinkConnector
Team Name: 5, Chess Robot

Raw Material Stock: 1/8" 6061-T6 Aluminum

Step #	*Process Description*	*Machine*	*Fixture(s)*	*Tool(s)*	*Speed (RPM)*
1	Waterjet to dimension	Waterjet	-	-	-
2	Debur all edges	-	-	File	-
3	Hold part in vice on top of parallels	Mill	Vise	Parallels, edge stop	-
4	Find edge of shoulder as shown on drawing	Mill	Vise	Edge finder	1200
5	Bring 1 side to length by removing 0.010" until at specified dimension	Mill	Vise	End mill	500
6	Zero out on edge that you just brought length	Mill	Vise	-	-
7	Bring opposite side to length as specified in drawing	Mill	Vise	End mill	500
8	Turn part so short side is vertical	Mill	Vise	-	-
9	Locate edges and zero out	Mill	Vise	Edge Finder	1200
10	Go to hole location, centerdrill, and drill	Mill	Vise	Centerdrill, 50 Drill bit	1000
11	Repeat step 10 for secod hole	Mill	Vise		
12	Repeat steps 8-11 for opposite side	Mill	Vise		



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN	KO	4/4
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL: 0.25" Aluminum Plate	COMMENTS: Waterjet, Then Drill Holes		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

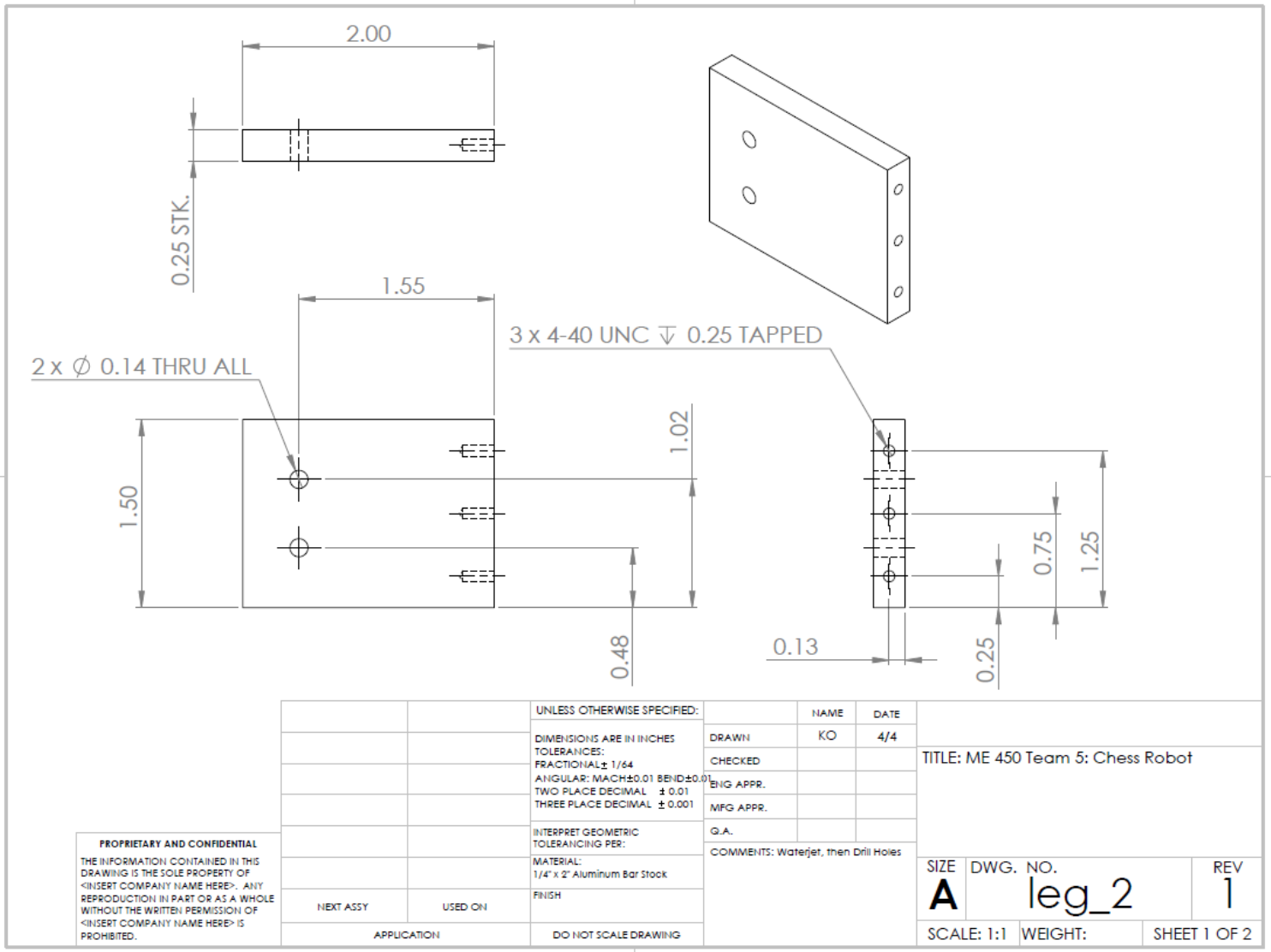
TITLE: ME 450 Team 5: Chess Robot

SIZE	DWG. NO.	REV
A	left_foot	0
SCALE: 1:2	WEIGHT:	SHEET 1 OF 2

MANUFACTURING PLAN
RAW MATERIAL STOCK: Aluminum Plate 1/4" x 12" x 18"

STEP	PROCESS DESCRIPTION	MACHINE	FIXTURE	TOOL(S)	SPEED (RPM)
1	Waterjet Profile	Waterjet			
2	Deburr Post Waterjet			Deburring Tools	
3	Hold part in vise on top of parallels with > 0.125" material sticking out	Mill	Vise	1.375 Parallels	
4	Insert Edgefinder in Mill	Mill	Vise	Drill Chuck, Edgefinder, Drawbar	
5	Use edgefinder to locate Y datum at along vise jaws.	Mill	Vise	Drill Chuck, Edgefinder	1000
6	Use edgefinder to locate X datum at right edge of part.	Mill	Vise	Drill Chuck, Edgefinder	1000
7	Remove edgefinder and insert centerdrill to drill chuck.	Mill	Vise	Drill Chuck, Center Drill 1, Draw Bar	
8	Centerdrill all holes	Mill	Vise	Drill Chuck, Center Drill 1	1200
9	Swap Centerdrill with Drill bit.	Mill	Vise	Drill Chuck, #30 Drill Bit, Draw Bar	
10	Drill all holes thru.	Mill	Vise	Drill Chuck, #30 Drill Bit	1500
11	Swap Drill bit for counterbore drill bit.	Mill	Vise	Drill Chuck, #I Counterbore, Drawbar	
12	Countbore Holes to depth	Mill	Vise	Drill Chuck, #I Counterbore	1200
13	Remove and Deburr			Deburring Tools	

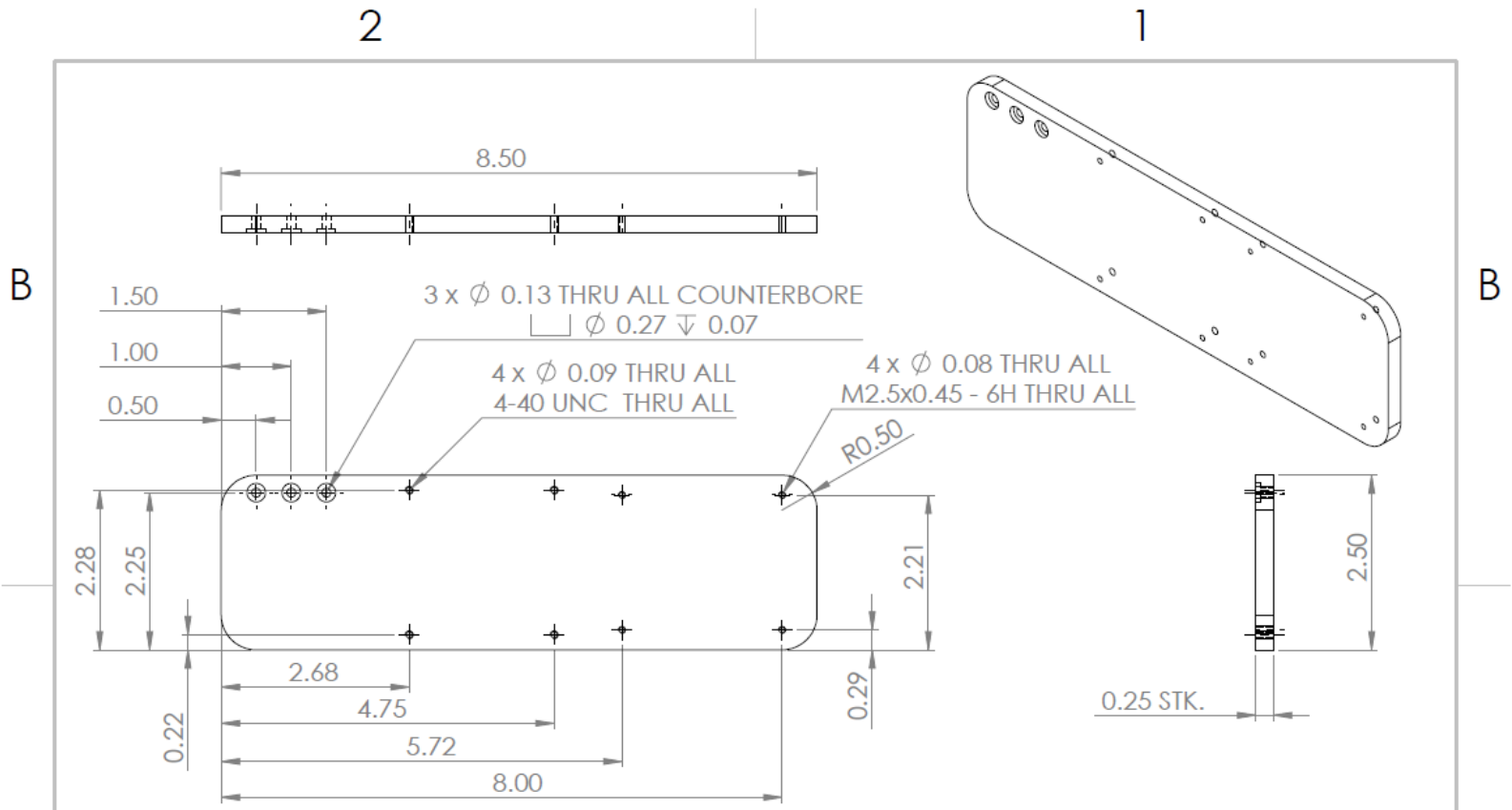
SHEET 2 OF 2



MANUFACTURING PLAN
RAW MATERIAL STOCK: Aluminum Bar 1/4" x 2"

STEP	PROCESS DESCRIPTION	MACHINE	FIXTURE	TOOL(S)	SPEED (RPM)
1	Cut Stock on Bandsaw	Bandsaw	Clamp	Bandsaw	
2	Deburr Post Bandsaw			Deburring Tools	
3	Hold part in vise on top of parallels with > 0.125" material sticking out	Mill	Vise	1.375 Parallels	
4	Insert Edgefinder in Mill	Mill	Vise	Drill Chuck, Edgefinder, Drawbar	
5	Use edgefinder to locate Y datum at along vise jaws.	Mill	Vise	Drill Chuck, Edgefinder	1000
6	Use edgefinder to locate X datum at left edge of part.	Mill	Vise	Drill Chuck, Edgefinder	1000
7	Remove edgefinder and insert centerdrill to drill chuck.	Mill	Vise	Drill Chuck, Center Drill 1, Draw Bar	
8	Centerdrill 2 face holes	Mill	Vise	Drill Chuck, Center Drill 1	1200
9	Swap Centerdrill with Drill bit.	Mill	Vise	Drill Chuck, #27 Drill Bit, Draw Bar	
10	Drill all holes thru.	Mill	Vise	Drill Chuck, #27 Drill Bit	1500
11	Remove part, and rotate so side is facing up. Repeat Steps 4-7	Mill	Vise		
12	Centerdrill 3 side holes	Mill	Vise	Drill Chuck, Center Drill 1	1200
13	Remove centerdrill and insert drill bit to drill chuck.	Mill	Vise	Drill Chuck, #43 Drill Bit	
14	Drill 3 Holes to depth	Mill	Vise	Drill Chuck, #43 Drill Bit	1500
15	Tap Holes	Mill	Vise	4-40 Tap Drill, Tap Handle	
16	Remove and Deburr			Deburring Tools	

SHEET 2 OF 2



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ±0.05 THREE PLACE DECIMAL ±0.005	DRAWN	KO	4/4
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL: 0.25" Aluminum Plate	COMMENTS: Waterjet, Then Drill Holes		
		FINISH			
NEXT ASSY	USED ON				
APPLICATION		DO NOT SCALE DRAWING			

TITLE: ME 450 Team 5: Chess Robot

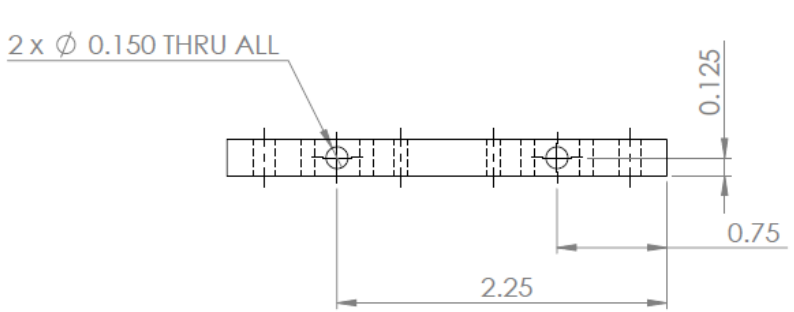
SIZE	DWG. NO.	REV
A	right_foot	1

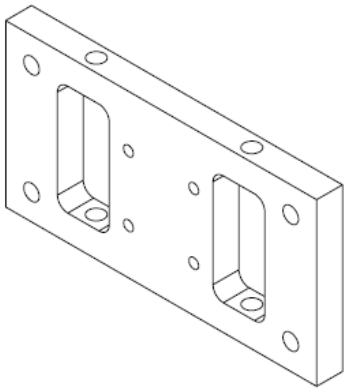
SCALE: 1:2 | WEIGHT: | SHEET 1 OF 2

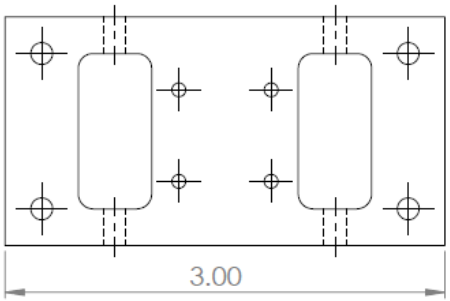
MANUFACTURING PLAN
RAW MATERIAL STOCK: Aluminum Plate 1/4" x 12" x 18"

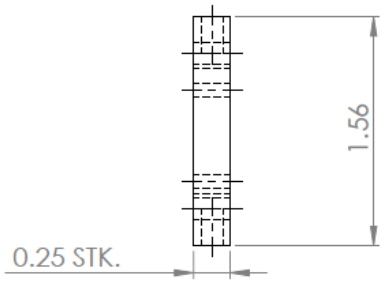
STEP	PROCESS DESCRIPTION	MACHINE	FIXTURE	TOOL(S)	SPEED (RPM)
1	Waterjet Profile	Waterjet			
2	Deburr Post Waterjet			Deburring Tools	
3	Hold part in vise on top of parallels with > 0.125" material sticking out	Mill	Vise	1.5" Parallels	
4	Insert Edgefinder in Mill	Mill	Vise	Drill Chuck, Edgefinder, Drawbar	
5	Use edgefinder to locate Y datum at along vise jaws.	Mill	Vise	Drill Chuck, Edgefinder, Worktable Stop	1000
6	Use edgefinder to locate X datum at right edge of part.	Mill	Vise	Drill Chuck, Edgefinder	1000
7	Remove edgefinder and insert centerdrill to drill chuck.	Mill	Vise	Drill Chuck, Center Drill 1	
8	Centerdrill all holes	Mill	Vise	Drill Chuck, Center Drill 1	1200
9	Swap Centerdrill with Drill bit.	Mill	Vise	Drill Chuck, #30 Drill Bit, Draw Bar	
10	Drill 3 holes on right thru.	Mill	Vise	Drill Chuck, #30 Drill Bit	1500
11	Swap Drill bits.	Mill	Vise	Drill Chuck, #46 Drill Bit	
12	Drill 4 holes on left Thru	Mill	Vise	Drill Chuck, #46 Drill Bit	1500
13	Swap Drill bits.	Mill	Vise	Drill Chuck, #43 Drill Bit	
14	Drill 4 holes on left Thru	Mill	Vise	Drill Chuck, #43 Drill Bit	1500
15	Tap Holes with Tab Drill		Vise	M2.5 Tap, 4-40 Tap, Tap Drill Handle	
16	Flip Part over in vise, then repeat steps 4-6, using the left side instead of the right	Mill	Vise	Drill Chuck, Edgefinder	1000
17	Swap Drill bit for counterbore drill bit.	Mill	Vise	Drill Chuck, 1/4" Endmill	
18	Counterbore Holes to depth	Mill	Vise	Drill Chuck, 1/4" Endmill	1200
19	Remove and Deburr			Deburring Tools	

SHEET 2 OF 2









		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN	KO	4/5
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS: Waterjet, then Drill Holes		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE: ME 450 Team 5: Chess Robot

SIZE	DWG. NO.	REV
A	slider_plate	
SCALE: 1:1	WEIGHT:	SHEET 1 OF 2

MANUFACTURING PLAN
RAW MATERIAL STOCK: Aluminum Plate 1/4" x 12" x 18"

STEP	PROCESS DESCRIPTION	MACHINE	FIXTURE	TOOL(S)	SPEED (RPM)
1	Waterjet Profile	Waterjet			
2	Deburr Post Waterjet			Deburring Tools	
3	Place in Vise with Top sticking out	Mill	Vise		
4	Insert Edgefinder in Mill	Mill	Vise	Drill Chuck, Edgefinder, Drawbar	
5	Use edgefinder to locate Y datum at along vise jaws.	Mill	Vise	Drill Chuck, Edgefinder	1000
6	Use edgefinder to locate X datum at left edge of part.	Mill	Vise	Drill Chuck, Edgefinder	1000
7	Remove edgefinder and insert centerdrill to drill chuck.	Mill	Vise	Drill Chuck, Center Drill 1, Draw Bar	
8	Centerdrill 2 side holes	Mill	Vise	Drill Chuck, Center Drill 1	1200
9	Swap Centerdrill with Drill bit.	Mill	Vise	Drill Chuck, #25 Drill Bit, Draw Bar	
10	Drill all holes thru.	Mill	Vise	Drill Chuck, #25 Drill Bit	1500
11	Remove part, and rotate so other side is facing up. Repeat Steps 4-10	Mill	Vise		
16	Remove and Deburr			Deburring Tools	

SHEET 2 OF 2

Appendix K: Safety Standards

OSHA instruction STD 01-12-002: Guidelines For Robotics Safety

Guidelines for Robotics Safety

U.S. Department of Labor

Occupational Safety and Health Administration

Washington, D.C.

A - 1

OSHA Instruction PUB 8-1.3 SEP 21,1987 Office of Science and Technology Assessment

FOREWORD

The purpose of this instruction is to inform OSHA compliance officers and employers and employees about safety concerns that have arisen with the growing use of robotics systems in manufacturing. Industrial robots can be used to perform hazardous tasks but in doing so they can create new hazards. With the burgeoning use of robots in industry, it is feared that without adequate guarding and personnel training, injury rates for employees working with robots may increase.

Current guidelines for robot safety include the American National Standards Institute (ANSI) ANSI-RIA R15.06-1986, "American National Standard for Industrial Robots and Robot Systems - Safety Requirements," and the National Institute for Occupational Safety and Health (NIOSH) December, 1984 Alert "Request for Assistance in Preventing the Injury of Workers by Robots." Copies of the ANSI Standard are available from the American National Standards Institute, Inc., 1430 Broadway, New York, NY 10018. The NIOSH Alert was prepared by its Division of Safety Research, 944 Chestnut Ridge Road, Morgantown, WV 26505.

This instruction provides general introductory material describing the features of robots and robotics systems which present unusual hazards and will describe some of the more common safety systems employed to alleviate these hazards. The ANSI Standard defines consensus provisions for the construction, reconstruction, modification, installation, safeguarding, care, testing, and start-up of robots and robotics systems as well as training for robot and robotics systems operations and maintenance personnel. The NIOSH Alert contains safety recommendations that are based on its field evaluation of the first identified robot-related fatality in the United States.

INTRODUCTION

Robots are reprogrammable, multifunctional, mechanical manipulators that typically employ one or more means of power: electromechanical, hydraulic, or pneumatic. Industrial robots have been used chiefly for spray painting, spot-welding, and transfer and assembly tasks. A robot performs its tasks in a physical area known as the robot operating work envelope. This work envelope is the volume swept by all possible programmable robot movements. This includes the area where work is performed by robot tooling.

A robot can have one or more arms which are interconnected sets of links and powered joints. Arms are comprised of manipulators which support or move wrists and end-effectors. An end-effector is an accessory tool specifically designed for attachment to a robot wrist to enable the robot to perform its intended task. Examples of end-effectors include grippers, spot-weld guns, and spray paint guns. The ANSI R15.O6-1986 Standard defines an industrial robot system as that which includes industrial robots, end-effectors, and any equipment, devices and sensors required for the entire robot system to perform its tasks.

A-2

OSHA Instruction PUB 8-1.3 SEP 21, 1987 Office of Science and Technology Assessment

Most robots are set up for an operation by the teach-and-repeat technique. In this technique, a trained operator (programmer) typically uses a portable control device (commonly referred to as a teach pendant) to manually key a robot and its tasks. Program steps are of the up-down, left-right, in-out, and clockwise-counterclockwise variety. Robot speeds during these programming sessions are required to be slow. The ANSI Standard currently recommends that this slow speed should not exceed 10 in/sec (250 mm/sec).

The very nature of robotics systems operations has introduced a new type of employee into the industrial workplace, the corrective maintenance worker. This individual is normally present during all operations of a robotics system and is responsible for assuring continuing operation - adjusting speeds, correcting grips, and freeing jam-ups. The corrective maintenance worker may also be the trained programmer who guides a robot through the teach-and-repeat technique. It is necessary for this individual to be near the robot from time to time, which raises concerns about his or her safety and the safety of other workers who may also be exposed.

Recent studies in Sweden and Japan indicate that many robot accidents do not occur under normal operating conditions but rather during programming, adjustment, testing, cleaning, inspection, and repair periods. During many of these operations, the operator, programmer or corrective maintenance worker may temporarily be within the robot work envelope while power is available to moveable elements of the robot system.

This guideline describes some of the elements of good safety practices and techniques used in the section and installation of robots and robot safety systems, control devices, robot programming and employee training. A comprehensive list of safety requirements is provided in the ANSI R15.06-1986 Standard.

TYPICAL ACCIDENTS

The following are documented accidents involving robots that occurred recently in Japan, Sweden, and the United States:

- A worker attempted to remove an imperfectly formed piece from a conveyor with both hands while the operation limit switch of a material feed and removal robot remained in its active position. The worker's back was forced against the robot.

- After adjusting a metal shaving machine, an operator was caught between the machine and a just-extended arm of a material feed and removal robot.

- A welding robot went functionally awry and its arm flung a worker against another machine.

- A worker removed the cover of an operating assembly robot to retrieve a fallen part and caught his hand in the robot's drive train.

- A worker attempted to retrieve a part needed in an ongoing assembly without shutting off an assembly robot's power supply. His hand was caught between the robot's arm and the unit being assembled.

- A robot's arm functioned erratically during a programming sequence and struck the operator.

- A fellow employee accidentally tripped the power switch while a maintenance worker was servicing an assembly robot. The robot's arm struck the maintenance worker's hand.

- An operator performing troubleshooting on a metal plater robot maneuvered the robot's arm into a stopped position. This triggered the robot's emergency stop mode which delayed venting of a pneumatic air storage device. When the return mode was activated, the robot's arm moved suddenly and jammed the operator's thumb against a structural member.

- An automatic welder robot operator made a manual adjustment without stopping the robot. He was hit in the head by one of the robot's moving parts when the next batch of weldments arrived.

- A materials handling robot operator entered a robot's work envelope during operations and was pinned between the back end of the robot and a safety pole.

SAFETY SYSTEMS

The proper selection of an effective robotics safety system must be based on hazard analysis of the operation involving a particular robot. Among the factors to be considered in such an analysis are the task a robot is programmed to perform, the start-up and the programming procedures, environmental conditions and location of the robot, requirements for corrective tasks to sustain normal operations, human errors, and possible robot malfunctions. Sources of robot hazards include:

1. Human errors;

2. Control errors;

3. Unauthorized access;

4. Mechanical hazards;

5. Environmental hazards; and

6. Electric, hydraulic, and pneumatic power sources.

OSHA Instruction PUB 8-1.3 SEP 21, 1987 Office of Science and Technology Assessment

An effective safety system protects operators, engineers, programmers, maintenance personnel, and others who could be exposed to hazards associated with a robot's operation. A combination of methods may be used to develop an effective safety system. Redundancy and backup systems are recommended, particularly if a robot can create serious hazardous conditions.

Guarding Methods:

1. Interlocked Barrier Guard

This is a physical barrier around a robot work envelope incorporating gates equipped with interlocks. These interlocks are designed so that all automatic operations of the robot and associated machinery will stop when any gate is opened. Restarting the operation requires closing the gate and reactivating a control switch located outside of the barrier. A typical practical barrier is an interlocked fence designed so that access through, over, under, or around the fence is not possible when the gate is closed.

2. Fixed Barrier Guard

A fixed barrier guard is a fence that requires tools for removal. Like the interlocked barrier guard, it prevents access through, over, under, or around the fence. It provides sufficient clearance for a worker between the guard and any robot reach, including parts held by an end-effector, to perform a specific task under controlled conditions.

3. Awareness Barrier Device

This is a device such as a low railing or suspended chain that defines a safety perimeter and is intended to prevent inadvertent entry into the work envelope but can be climbed over, crawled under, or stepped around. Such a device is acceptable only in situations where a hazard analysis indicates that the hazard is minimal and inter locked or fixed barrier guards are not feasible. Interlocked or fixed barrier guards provide a positive protection needed to prevent worker exposure to robotic systems hazards.

4. Presence Sensing Devices

The presence detectors that are most commonly used in robotics safety are pressure mats and light curtains. Floor mats (pressure sensitive mats) and light curtains (similar to arrays of

photocells) can be used to detect a person stepping into a hazardous area near a robot. Proximity detectors operating on electrical capacitance, ultrasonics, radio frequency, laser, and

television principles are currently undergoing reliability testing in research laboratories because of recognized limitations in their capability of detecting the presence of personnel. Although some of these devices are already available in the safety equipment marketplace, care must be used in their selection to insure adequate safety and reliability. At this time, such proximity detectors are not recommended for such use unless a specific analysis confirms their acceptability for the intended use.

Effective presence sensing devices stop all motion of the robot if any part of a worker's body enters the protected zone. Also, they are designed to be fail-safe so that the occurrence of a failure within the device will leave it unaffected or convert it into a mode in which its failed state would not result in an accident. In some cases this means deactivation of the robot. Factors which are considered in the selection of such devices include spatial limitations of the field, environmental conditions affecting the reliability of the field, and sensing field interference due to robot operation.

5. Emergency Robot Braking

Dangerous robot movement is arrested by dynamic braking systems rather than simple power cut-off. Such brakes will counteract the effects of robot arm inertia. Cutting off all power could create hazards such as a sudden dropping of a robot's arm or flinging of a workpiece.

6. Audible and Visible Warning Systems

Audible and visible warning systems are not acceptable safeguarding methods but may be used to enhance the effectiveness of positive safeguards. The purposes of audible and visible signals need to be easily recognizable.

CONTROL DEVICES

The following characteristics are essential for control devices:

1. The main control panel is located outside the robot system work envelope in sight of the robot.

2. Readily accessible emergency stops (palm buttons, pull cords, etc.) are located in all zones where needed. These are clearly situated in easily located positions and the position identifications are a prominent part of personnel training. Emergency stops override all other controls.

3. The portable programming control device contains an emergency stop.

4. Automatic stop capabilities are provided for abnormal robot component speeds and robot traverses beyond the operating envelope.

5. All control devices are clearly marked and labeled as to device purpose. Actuating controls are designed to indicate the robot's operating status.

6. Controls that initiate power or motion are constructed and guarded against accidental operation.

7. Each robot is equipped with a separate circuit breaker that can be locked only in the "off" position.

8. User-prompt displays are used to minimize human errors.

9. The control system for a robot with lengthy start-up time is designed to allow for the isolation of power to components having mechanical motion from the power required to energize the complete robot system.

10. Control systems are selected and designed so that they prevent a robot from automatically restarting upon restoration of power after electrical power failure. The systems also prevent hazardous conditions in case of hydraulic, pneumatic or vacuum loss or change.

11. A robot system is designed so that it could be moved manually on any of its axes without using the system drive power.

12. All control systems meet OSHA 29 CFR 1910 Subpart S standards for electrical grounding, wiring, hazardous locations, and related requirements.

INSTALLATION, MAINTENANCE AND PROGRAMMING

Good installation, maintenance, and programming practices include the following:

1. The robot is installed in accordance with the manufacturer's guidelines and applicable codes. Robots are compatible with environmental conditions.

2. Power to the robot conforms to the manufacturer's specifications.

3. The robot is secured to prevent vibration movement and tip over.

4. Installation is such that no additional hazards are created such as pinch points with fixed objects and robot components or energized conductor contact with robot components.

5. Signs and markings indicating the zones of movement of the robot are displayed prominently on the robot itself and, if possible, on floors and walls.

6. Stops are placed on the robot system's axes to limit its motions under rated load and maximum speed conditions.

7. A lock-out procedure is established and enforced for preventive maintenance or repair operations.

8. The robot manufacturer's preventive maintenance schedule is followed rigorously.

9. A periodic check of all safety-critical equipment and connections is established.

10. Stored energy devices, such as springs and accumulators, are neutralized before robot servicing.

11. Only programmers have access to the work envelope and full control of the robot when it is in the teach mode.

12. All robot motion initiated from a teach pendant used by a programmer located within the robot work envelope is subject to the current ANSI slow speed recommendation of 10 in/sec (250 mm/sec).

TRAINING

Effective accident prevention programs include training. Some points to be considered in training programs include:

1. Managers and supervisors in facilities that use robots are trained in the working aspects of robots so that they can set and enforce a robotics safety policy from an informed viewpoint.

2. The employer insures that his or her company has a written robotics safety policy that has been explained to all personnel who will be working with robots. This safety policy states by name which personnel are authorized to work with robots.

3. Robot programming and maintenance operations are prohibited for persons other than those who have received adequate training in hazard recognition and the control of robots.

4. Robot operators receive adequate training in hazard recognition and the control of robots and in the proper operating procedure of the robot and associated equipment.

5. Training is commensurate with a trainee's needs and includes the safeguarding method(s) and the required safe work practices necessary for safe performance of the trainee's assigned job.

6. If it is necessary for an authorized person to be within the work envelope while a robot is energized, for example during a programming sequence, training is provided in the use of slow robot operation speeds and hazardous location avoidance until the work is completed. Such training also includes a review of emergency stops, and a familiarization with the robot system's potentially hazardous energy sources.

REFERENCES

- National Institute for Occupational Safety and Health (NIOSH) Alert "Request for Assistance in preventing the Injury of Workers by Robots." National Institute for Occupational Safety and Health, Division of Safety Research, 944 Chestnut Ridge Road, Morgantown, West Virginia 26505.

- American National Standards Institute (ANSI) American National Safety Standard ANSI-RIA R15.06-1986, "Industrial Robots and Industrial Robot Systems - Safety Requirements." American National Standards Institute, Inc., 1430 Broadway, New York, New York 10018.

- Robotic Industries Association, 900 Victors Way, P.O. Box 3724, Ann Arbor, Michigan 48106.

- Occupational Safety and Health Administration publication 3067, Concepts and Techniques of Machine Safeguarding, U.S. Department of Labor, 1980 (reprinted 1983). Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20210.