ECE 445

Senior Design Laboratory

Design Document

AI Chess Robot with Computer Vision

Team 33

Zack Alonzo (zalonzo2)

Jose Flores (joseaf3)

Joshua Hur (jhur22)

TA: Zicheng Ma (zicheng5)

February 7th, 2024

Contents

Introduction

1.1 Problem

Our project's goal is to address the need for a tangible and interactive chess-playing device, enabling users to play in the physical world against a chess AI rather than relying on digital platforms. Designed for both beginners and advanced players, the chess-playing robot would provide an engaging alternative to mobile apps, allowing for skill development and strategic thinking in a hands-on manner.

1.2 Solution

We plan to develop an autonomous chess-playing robot that eliminates the need for a human opponent by incorporating a chess algorithm with varying difficulty levels. Using a system involving a magnet and motors beneath the board, the computer opponent's chess pieces will move autonomously while the human player will simply pick up and place their pieces. Then, our robot will analyze the current board position by capturing an image through a camera and will identify all the pieces on the board by identifying each piece's color, associating it with the corresponding chess piece. With this updated board, we will now be able to determine the optimal move based on the chosen difficulty level and current board position. When identified, our code will output the necessary information to the system with the magnet and the motors underneath the board to move its intended piece and wait for the subsequent human player's move (additionally, a button press will "submit" the player's move).

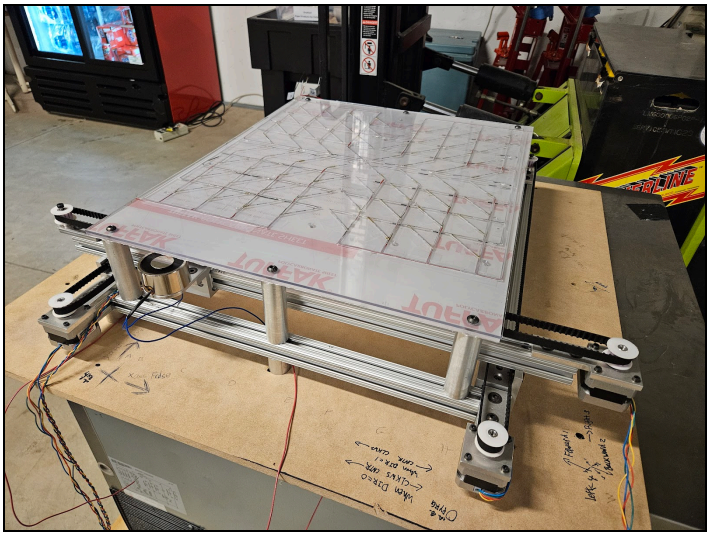

Figure 1. Chess board's rail & magnet system

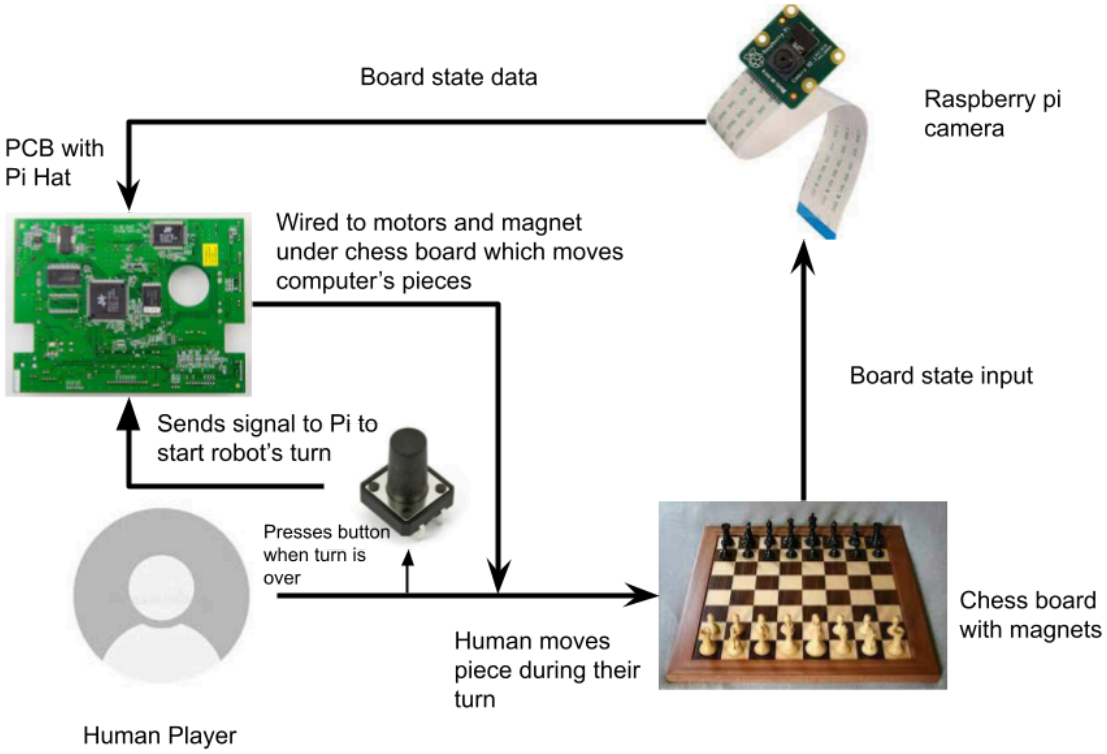

Figure 2. High-level project overview

1.3 High-level Requirements List

- Computer vision algorithm correctly identifies chess piece positions and their identity on the board with 95% ± 5% accuracy.
- Chess AI is implemented in a way that is able to identify when the human player has cheated with 95% ± 5% accuracy.
- Rail and magnet system grabs the intended chess piece to the intended location on the chess board with 95% ± 5% accuracy.

Design

2.1 Block Diagram

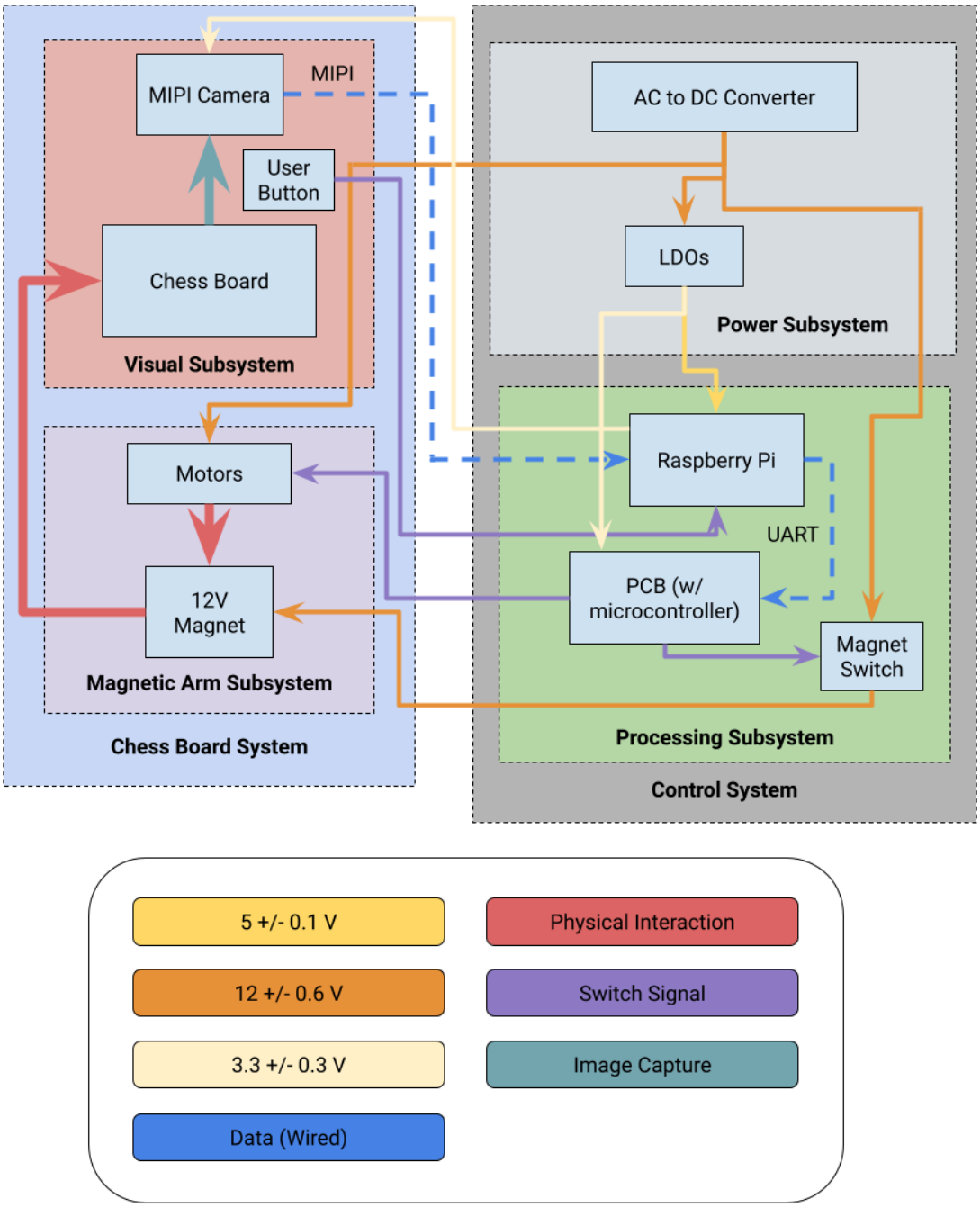

Figure 3. Block diagram of the chess-playing robot

2.2 Subsystem Overview

The design is broken down into four subsystems, which are implemented in hardware and software.

2.2.1 Power Subsystem

The Power System is responsible for powering all electrical and mechanical pieces associated with our project, such as the motors, magnet, camera, Raspberry PI, and microcontroller. It is comprised of:

- (1) AC/DC Converter Output: 12V ± 0.6V, 5A
- (1) 12V to 5.1V Buck Converter
- (1) 12V to 3.3V Buck Converter

The main source of power will come from a wall outlet where the AC to DC converter will output the 12 volts we need to power our project. The 12V output will be connected straight to the 3 Stepper motors and to the MOSFET that leads to the Electromagnet. From the output of our 12V to 5.1V Buck Converter, we will connect the Raspberry PI 4 as a type USB-C connection. One final Buck Converter will be used to step down 12V to 3.3V where it will be used as input to our microcontroller.

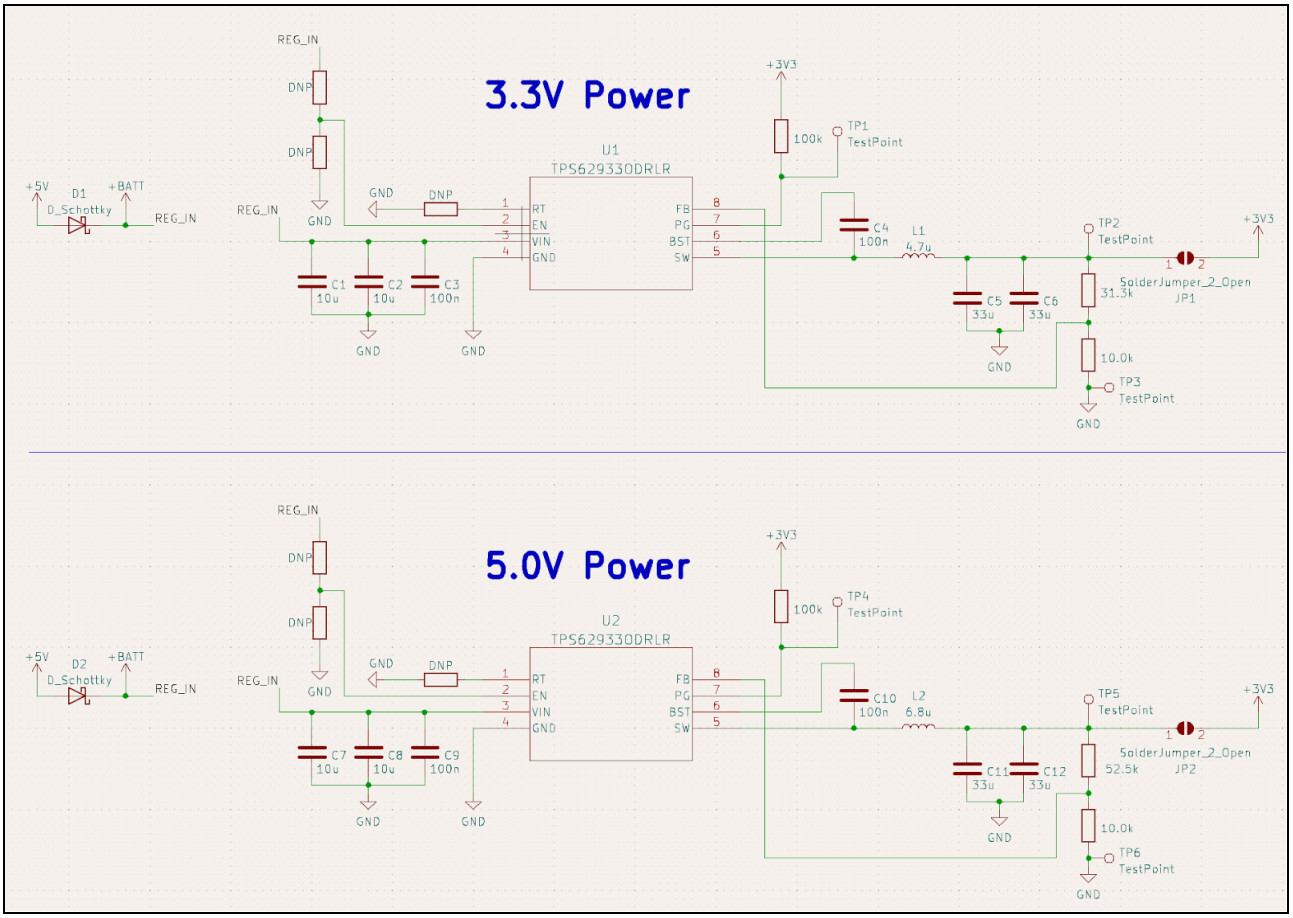

Figure 4. PCB schematic design for our Buck converters (3.3V & 5.0V)

Requirements	Verification
• Maintain a voltage of 12 ± 0.6V from the AC/DC output	• Use a voltage measuring device such as an oscilloscope or a multimeter and place the positive side terminal on the output of the AC/DC converter • Next, place the negative side on the ground terminal of the pcb. • Read the device display and verify that the output is within the expected ranges
• Output 5 ± 0.1 V and 3.3 ± 0.1 V from our Buck Converters	• Use a voltage measuring device such as an oscilloscope or a multimeter and place the positive side terminal on the Buck converter's output • Next, place the negative side on the ground terminal of the pcb. • Read the device display and verify that the output is within the expected ranges. 5 ± 0.1 V for one Buck Converter and 3.3 ± 0.1 V for the other.
• Magnet switch is able to turn the magnet off and on by cutting and reconnecting the current to the magnet	• First, use a voltage measuring device such as an oscilloscope or a multimeter and place the positive side terminal on the magnet switch input and the negative side terminal on pcb ground. Read the display and ensure that the input is what the magnet requires. • Second, place the positive terminal of the device on the magnet switch output while keeping the negative terminal on ground. If the switch is off the display should read as 0V. If the switch is on, the display should read the same as the input. • Lastly, flip the switch and ensure the output is the other switch state. From on to off and from off to on. Repeat second step

Table 1: Power Subsystem – Requirements & Verification

2.2.2 Processing Subsystem

The Processing System is where all of the image processing, data analysis, path planning, and execution takes place. It is comprised of:

- (1) Raspberry PI 4 Model B
- (1) ESP32 S3 Microcontroller
- (1) N-Channel MOSFET (Magnet Switch)

The system is split up between the 2 components, one is the Raspberry Pi, which will handle the computer vision code we write as well as the python library for the chess AI we use and the microcontroller will be handling the planning and execution of moves once a move has been made by the chess AI. The only communication between these two devices will be telling the microcontroller the best move or if the human player has cheated. Since the Raspberry Pi will handle the computer vision, it needs to be connected to the camera to receive the images, and the microcontroller needs to be connected to the motors and magnet in order for it to execute the moves. Whenever a piece needs to be moved, the magnet will turn on by sending current to it straight from the power input. Whenever we need to turn the magnet off, a switch will cut off the current to the magnet which is controlled by the microcontroller.

Requirements	Verification
● Computer vision algorithm correctly identifies chess piece positions and their identity on the board with 95 ± 5% accuracy	● Randomly generate a chess board state using an online tool and place the pieces as shown. ● Once set up, press the button to signal to the algorithm to capture a screenshot of the board and begin processing. ● Check the internal representation of the chessboard and ensure that it is correct by Forsyth-Edwards Notation (FEN) standards. ● Repeat 100 times and ensure that the accuracy fall within expected performance
● Microcontroller plans a path to move the necessary pieces and executes it with 95 ± 5% success rate	● Play chess games against the robot and keep track of its successes and failures. ● Continue to play until the robot reaches 100 moves performed and count up all the robot's successes. Ensure that its accuracy falls within expected performance.

Table 2: Processing Subsystem – Requirements & Verification

2.2.3 Visual Subsystem

The Visual System will serve as the part of the project that the human interacts with and the part where the AI receives its visual input to the algorithm via the camera. The System is comprised of:

- (1) Chess Board created by machine shop
- (1) Arducam for Raspberry PI
- (32) Colored chess pieces
- (1) User controlled button

The chess board will be provided by the machine shop, and it already includes the Magnetic Arm System underneath the board. Additionally, there will be the camera that will be hung above the board looking down that will be used as input to the Raspberry Pi.What the camera sees is what our image processing code will work on and send to the chess AI. There will also be a button that the user will press at the end of every turn to signal that it is the end of their turn and the robot will begin analyzing the board. This will loop until there is either a winner or a stalemate

Requirements	Verification
● Raspberry Pi can recognize the camera connection	● Plug the two device into each other ● Check the Raspberry Pi's connection recognition ● If it does not recognize the camera, troubleshoot with a Raspberry Pi/MIPI camera manual
● Camera only sees the board to avoid distractions	● Mount the camera to the centering apparatus ● Output what the camera sees onto a separate window or jpg/png ● Adjust the height of the apparatus accordingly until the outer edges of the camera sees the outer border of the chess board
● Pictures of the board are able to be sent to the Raspberry Pi	● Power the Raspberry Pi and connect the camera to the Pi ● Once powered, check the read value from the Raspberry Pi

Table 3: Visual Subsystem – Requirements & Verification

2.2.4 Magnetic Arm Subsystem

The magnetic arm system will receive data from the processing system telling the Magnetic Arm system how to move its motors and when to turn on the magnet in order to grab and move the AI's pieces effectively. It is comprised of:

- (3) Mercury Motor SM-42BYG011-25 2 Phase 1.8° 32/20
- (1) KK P-50/27 50 kg Electromagnet

There will be two motors parallel to each other and dedicated to operating in the same 3-dimensional plane. They will be responsible for moving the third motor which will rest perpendicularly on each of the other two motors. This will allow the rail system to move in 4-directions. Resting on the third motor's axis will be the electromagnet that will be fed voltage to turn on and off the magnetism. The magnet will pick up and drop off pieces through the metal pieces that are secured underneath each chess piece. To allow special movements from the knight chess pieces, the magnet will drag pieces along the lines of the chess board to maneuver around them. As a result, the chess board tiles will be around 1½ times larger than the diameter of the magnet.

Requirements	Verification
● Rail system can move to specific chess board positions with 95 ± 5% accuracy	● First verify that it can move to specified chess board positions with the processing subsystem's software which will be similar to computer numerical controller (CNC) ● Then, generalize the chess positions with a random number generator ● Run the program for 100 trials
● Rail system can move from one chess tile to another while holding onto a chess piece without bumping into other pieces with 95 ± 5% accuracy	● Use the program from the magnetic arm subsystem's first verification process, but power the magnet when it travels from one tile to another ● Have a chess piece over the magnet to be dragged around, and place pieces around the board to test for bumps ● Run it for 100 trials
● Magnet will grab and hold on to desired chess piece with 95 ± 5% success rate	● Use the program from the magnetic arm subsystem's first verification process, but power the magnet when it travels from one tile to another ● Have a chess piece over the magnet to be dragged around ● Run it for 100 trials

Table 4: Magnetic Arm Subsystem – Requirements & Verification

2.3 Physical/Hardware Design

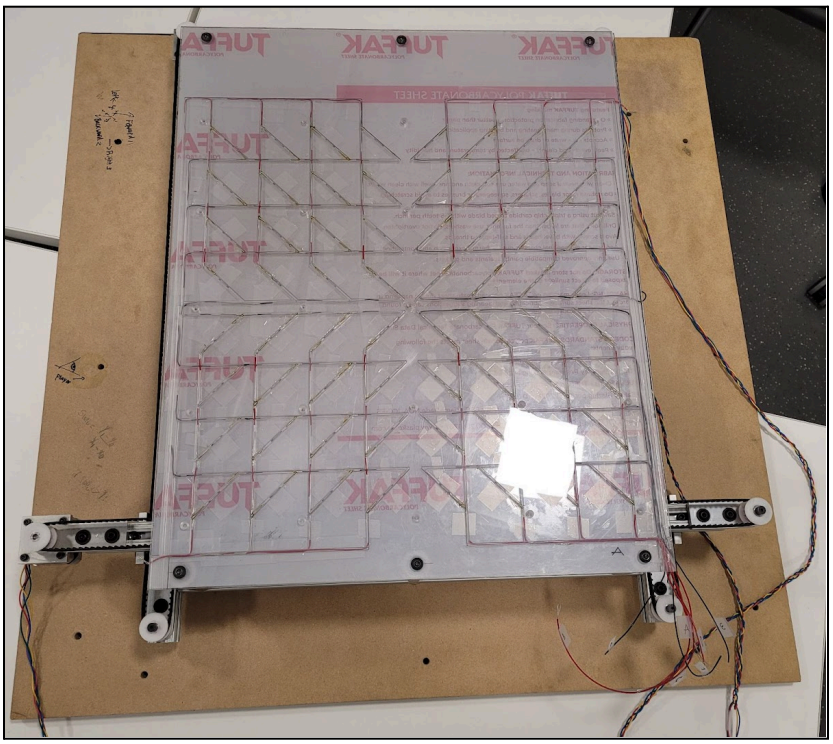

Figure 5. Birdseye view of the magnetic arm subsystem

This is a bird's eye view of the chess board we are using for the project. The board itself, which is two ¼ inch layers of plexiglass, measures 22.5 x 18 inches. The actual range of operation for the magnetic arm system underneath the board is approximately 19 x 18 inches. This is all information that was given to us by the workers at the machine shop.

2.4 Software Design

The process will begin with the camera sending its RGB image to the Raspberry Pi for imaging processing. Once the Raspberry Pi finishes its interpretation of the chess board's game state

and returns a chess move to execute, it will be sent to the microcontroller. The microcontroller will then handle the movement and powering of the rails and magnet respectively.

2.4.1: Raspberry Pi - Image Processing

The Raspberry Pi will utilize OpenCV's color manipulation and image processing functions to filter the camera's raw image into interpretable data. The raw image will be sent through OpenCV's *cvtColor(img, cv2.COLOR_BGR2HSV)* function to transform it to the HSV color space. HSV color space allows for easier manipulation and specifications for desired colors, in terms of hue, saturation, and value. In this example *img* is the raw image that will be sent as input through the Raspberry Pi, and *cv2.COLOR_BGR2HSV* is the color space conversion code to convert from RGB to HSV color space.

```
img = cv2.imread('image')
hsv_img = cv2.cvt_COLOR(img, cv2.COLOR_BGR2HSV)
```

Figure 6. Example of OpenCV's cvt_COLOR() function

Once converted to the HSV color space, we can define color ranges for each of our 6 unique colors and their lighter and darker alterations. With the masks for each of the 6 colors' lighter and darker spectrums, we will be able to utilize openCV's *inrange(hsv_img, red_lower, red_upper)* function where in this example *hsv_img* is the input array of pixels that will be checked, *red_lower* is the lower bound, and *red_upper* is the upper bound to check for. Anything outside of that range will be omitted, but it will keep the same size of the input.

```
red_lower = (0, 50, 50)
red_upper = (10, 255, 255)
red_mask = cv2.inRange(hsv_img, red_lower, red_upper)
```

Figure 7. Example of OpenCV's inRange() function

Once we have a range for the specific color, we will apply *only_red_img = cv2.bitwise_and(img, mask=red_mask)*. *Img* can be either RGB or HSV color space while the *mask=* parameter is checking which pixels are within tolerance and if they are not, they are zeroed.

```
only_red_img = cv2.bitwise_and(img, mask=red_mask)
```

Figure 8. Example of OpenCV's bitwise_and() function

We will repeat this process 11 times because we need to identify and isolate 12 different colors. When all the chess pieces on the board are identified, their positions will be calculated based on pixel coordinate math. It will convert these positions into FEN string notation to send in as input for the python-chess library. Sent through the library, it will utilize algorithms and functions of python-chess and calculate the optimal move for its turn. It will then output that instruction into the microcontroller.

2.4.2: ESP32 S3 - Rail and Magnet Instructions

The ESP32 S3 microcontroller will utilize Arduino programming languages, such as C/C++ and Python, to interpret the Raspberry Pi's chess move output to orchestrate the rail movements and magnet switching on and off. Having received the chess move input from the Raspberry Pi, the ESP32 S3 microcontroller will interpret the input and transform it into an output that can instruct the motors and magnet. In order to instruct the correct devices of what to do, it will send the proper signals, such as 3.3V, to the appropriate GPIOs connected. Because the magnet requires 12V to operate and the microcontroller can only output 3.3V, we will utilize a MOSFET to use the microcontroller as an input to control the 12V going into the magnet, allowing us to turn the magnet on and off.

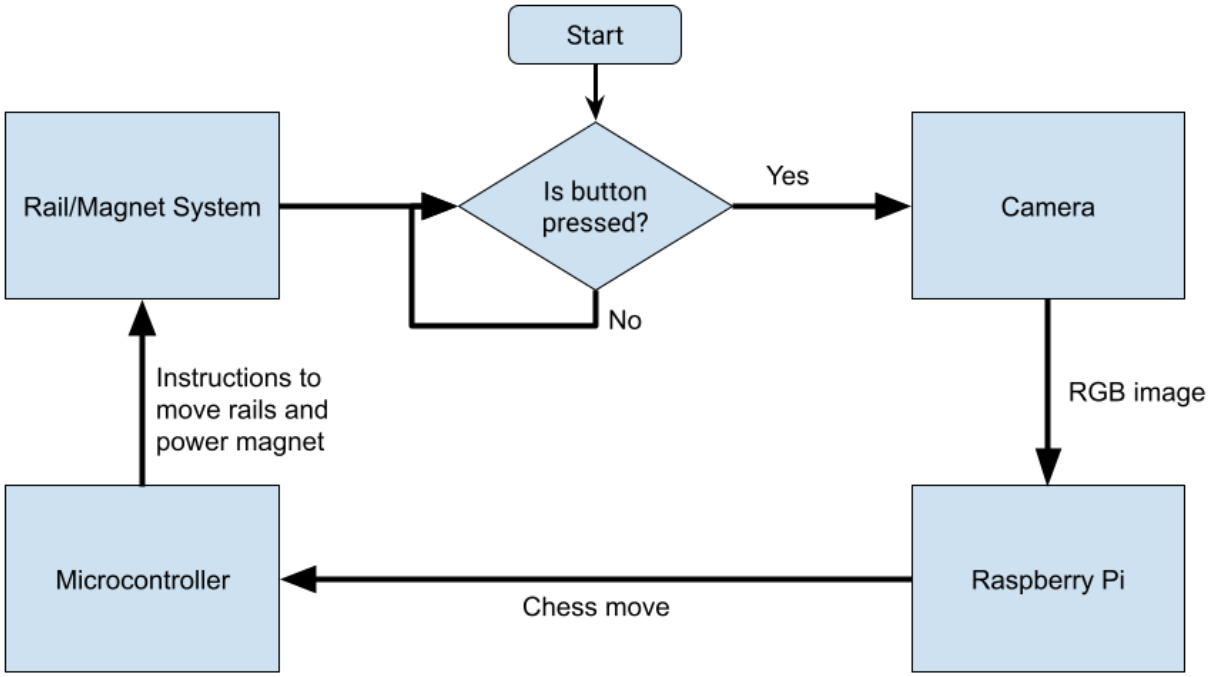

Figure 9. Flow chart of process execution

2.5 Tolerance Analysis

First, we will talk about our project's power needs. In order to calculate our tolerances, we need to decide on how we are going to power our chess-playing robot. Since our Project Proposal, we have received TA input, and we have decided to use a 12V 5A AC to DC adapter rather than the 5A LDO that we had previously discussed. The new AC to DC adapter can be found here, and the data sheet describes a tolerance of 5% (0.6V).

Other points of information to consider:

- Raspberry Pi 4 has a recommended input voltage of 5.1V with a range of -0.5 to 6V

	Voltage (V)	Current (A)	Min Power (W)	Max Power (W)
Stepper Motors	12	0.33	3.96	3.96
Electromagnet	12	0.83	9.96	9.96
Raspberry Pi 4	5 +/- 0.1	3.0	14.7	15.3
ESP32 S3	3.3 +/- 0.3	0.5	1.5	1.8
Arducam IMX219	3.0 +/- 0.3	0.3	0.81	0.99

Table 5. Power Draw Calculation Table

As shown in Table 1, our current draw will be 4.96A. The new 12V 5A AC to DC adapter has an output wattage range of 57.0 (11.4*5) to 63.0 (12.6*5). Considering the tolerance analysis, the AC to DC adapter meets the voltage, current, and power requirements for all components.

Next, we will talk about the stepper motors. According to the datasheet for the Mercury Motor SM-42BYG011-25 2 Phase 1.8° 32/20, its time constant τ is inductance (H)/resistance(Ω).

Inductance: 46+/-9.2 mH (36.8 to 55.2 mH)

Resistance: 34+/-3.4 Ω (30.6 to 37.4 Ω)

τ (electrical time constant) = inductance / resistance

τ_lower = 0.0368 / 37.4 = 0.984 ms

τ_upper = 0.0552 / 30.6 = 1.804 ms

This means it will charge up the coil to 63% of its rated value in 0.984 ms to 1.804 ms.

For comparisons, we viewed Sanyo Denki SS2422-5041 to observe the difference of capabilities.

Inductance: 2.9 mH.

Resistance: 5.4 Ω

τ (electrical time constant) = inductance / resistance

τ = 0.0029 / 5.4 = 0.537 ms

Percent diff upper = (abs(0.537/1.8044) - 1) * 100 = 70.23%

Percent diff lower = (abs(0.537/0.984) - 1) * 100 = 45.42%

Comparing the time constant of our stepper versus the compared stepper, we see that the compared stepper has a time constant that is 57.83+/-12.41% faster at charging its coil than ours. This difference does not matter for lower speed torques, as time constant does not matter as much for these values (“Stepper Motor Basics”). However, for higher speed torques, this does matter. Unfortunately, these are the stepper motors provided to us by the machine shop, and we do not have the budget for new ones, so our chess robot may move chess pieces slower than what is ideal.

Finally, we will talk about an analysis of our linear regulator. We want to make sure that our linear regulator, which is powering our stepper motors, is not overheating. We will be following

the specifications provided in the linear regulator's data sheet ("LM3940"). Here is a schematic of the linear regulator:

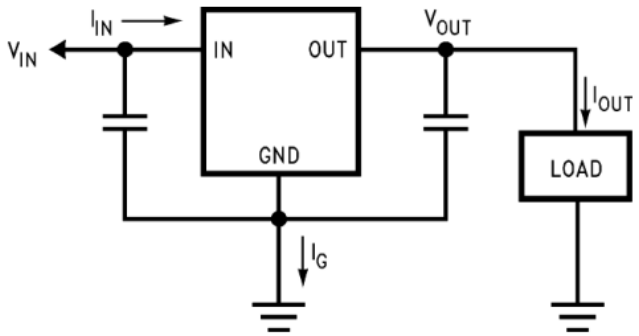

Figure 10. Linear regulator schematic

As preliminary knowledge, we chose the 3A variant of the buck converter because the Raspberry Pi can draw a maximum of 3A. Since the 5V from the buck converter that goes to the Pi also goes to the linear regulator, our Iin = 3A. Now, let's calculate the power dissipated by the regulator (Pd):

Iout = 1A, Iin = 3A, Vin = 5+/-0.5V (4.5 to 5.5V), Vout = 3.3+/-0.099V (3.201 to 3.399V)

Lower Pd:

Iin = Iout + Ig

Ig = Iin - Iout = 3 - 1 = 2

Pd = (4.5 - 3.399)*1+4.5*2 = 10.101

Upper Pd:

Iin = Iout + Ig

Ig = Iin - Iout = 3 - 1 = 2

Pd = (5.5 - 3.201)*1+4.5*2 = 11.299

Next, we will calculate TR(max), which is the maximum allowable temperature rise. We will do this using the formula TR(max) = TJ(max) − TA(max), where TJ(max) is max ambient temperature (which we will assume to be 20°C in ECEB), and TA(max) is max allowable junction temperature, which for commercial grade parts is 125°C. Plugging into the formula, we get TR(max) = 125 - 20 = 105°C. Lastly, we will use this formula to calculate the max allowable value for the junction-to-ambient thermal resistance:

$$R_{\theta(JA)} = T_R\ (max)/P_D$$

For lower Pd: 105/10.101 = 10.400°C/W

For upper Pd: 105/11.299 = 9.293°C/W

Both of these values are lower than 23.3°C/W, which is the max allowable value for the junction-to-ambient thermal resistance for the TO-220 package of the LDO, which is what we used. This means we will need a heatsink. However, the datasheet recommends we use either a standard heat sink or a copper plane on our PCB. Since we already have this, we should be dissipating the extra heat that the linear regulator is not able to dissipate, meaning our stepper motor drivers will be powered with 3.3V by a device that will not be overheating.

2.6 Cost Analysis

The cost analysis for the chess robot involves a breakdown of the parts and labor expenses. The parts required that affect our budget, as indicated in Table 6, have a total cost of $120.27, which is within our budget of $150. Adding the cost of parts that will not affect our budget from Table 7, the total cost of all parts is $285.86. As for the labor expenses, the estimate we received from the machine shop was 60 hours at $50/hour, totalling $3000. For the labor costs

of our group, the average starting salary for a computer engineer at UIUC is $109,176. At 40 hours a week, this comes to $52.49/hour. We estimate that the project will take 90 hours per group member so the total cost of our labor is 3 x 52.49 x 2.5 x 90 = $35,430.75. With the machine shop labor and parts, the total overall is $38,716.55.

Description	**Manufacturer**	**Quantity**	**Unit Price**	**Link**
Raspberry Pi 4 Model B - 4 GB RAM	Raspberry Pi Ltd	1	$55.00	Link
USB-C to 2 Pin Bare Wire	Maixbomr	1	$8.99	Link
Arducam for Raspberry Pi IMX219 Camera Module with ABS Case, 1080P IMX219 Camera Module	Arducam	1	$17.99	Link
ESP32-S3 Microcontroller	Espressif Systems	1	$1.85	Link
12V 5A Desktop AC Adapter	TT Electronics	1	$16.79	Link
Buck Switching Regulator IC Positive Adjustable 0.8V 1 Output 3A SOT-583	Texas Instruments Incorporated	2	$1.00	Link
~~12 to 3.3/5V Step Down Converter (for prototyping/backup)~~	~~Unbranded~~	~~1~~	~~$3.49~~	~~Link~~
2 oz. 12-Color Acrylic Craft Paint Set	DecoArt, LLC	1	$7.98	Link
Custom 3D Printed Chess Set at Jackson Innovation Studio	N/A	1	~$6.18	N/A
Total Cost			$120.27	

Table 6. Table of Parts Affecting Budget

Note: the way we calculated the cost of the chess pieces was using UltiMaker Cura, where we found that our pawns are 5 grams, knights are 9 grams, bishops and rooks are 8 grams, kings are 7 grams, and queens are 6 grams at 3 cents per gram.

Description	Supplier	Quantity	Unit Price	Our Price	Link
Zinc Flat Washers	Machine Shop	32	$0.06	Free	Link
30kΩ resistor	ECE ES Shop	1	$0.12	Free	Link
1.3kΩ resistor	ECE ES Shop	1	$0.13	Free	Link
51.0kΩ resistor	ECE ES Shop	1	$0.06	Free	Link
1.5kΩ resistor	ECE ES Shop	1	$0.06	Free	Link
10kΩ resistor	ECE ES Shop	2	$0.06	Free	Link
4.7uH Inductor	ECE ES Shop	1	$1.50	Free	Link
6.8uH Inductor	ECE ES Shop	1	$1.80	Free	Link
33uF Capacitor	ECE ES Shop	4	$0.70	Free	Link
N-channel MOSFET	ECE ES Shop	1	$0.80	Free	Link
2.1mm Female Barrel Plug	ECE ES Shop	1	$0.65	Free	Link
LED (for displaying cheat detection)	ECE ES Shop	2	$0.17	Free	Link
Button	ECE ES Shop	1	$2.30	Free	Link
Mercury Motor SM-42BYG011-25 2 Phase 1.8° 32/20	Machine Shop	3	$17.99	Free	Link
KK P-50/27 50 kg Electromagnet	Machine Shop	1	$21.37	Free	Link
1/4 Inch Thick Plexiglass Sheet 18” x 24”	Machine Shop	2	$18.83	Free	Link
Camera Mount	Machine Shop	1	$39.99	Free	Link
Total Cost			$165.59		

Table 7. Table of Parts Not Affecting Budget

Note: nothing in Table 7 will affect our budget and the unit costs are approximations of what the actual cost would be based on the links provided.

2.7 Schedule

Week	Jobs	Person
February 26th - March 3rd	Design first PCB	Everyone
	Fix design document with feedback from design review	Josh, Jose
	Prototype 3D printed chess pieces with different sizes and different sized washers	Zack
	Order all parts needed for project including backup parts	Zack
March 4th - March 10th	Revise PCB if necessary	Everyone
	Begin coding an outline for interfacing our image processing code with Chess AI library	Josh
	Design instructions to send to microcontroller from Raspberry Pi (almost like Gcode for 3D printers)	Zack
	Paint prototyped chess pieces in preparation for testing computer vision once we get our parts/PCB	Zack
	Gather data on how to communicate between microcontroller and motors/magnet switch	Jose
	FIRST PCB ORDER MARCH 5TH	Everyone
March 11th - March 17th (Spring Break)	Continue tasks from previous week if necessary or get a head start on next week's tasks	Everyone
March 18th - March 24th	Finalize physical chess board setup (sheet on top of plexiglass)	Everyone

	Revise PCB design	Everyone
	3D print our finalized design for the 32 chess pieces, assemble with washers, and paint them	Zack
	Begin work on image processing code to identify chess pieces	Josh
	Begin work on chess AI implementation using the python-chess library.	Jose
	SECOND PCB ORDER MARCH 19TH	Everyone
March 25th - April 2nd	Revise PCB design if necessary	Zack
	Begin working on programming microcontroller to decode Raspberry PI instructions to move motors and flip magnet switch	Zack
	Test and finalize the chess AI implementation using the python-chess library.	Jose
	Continue work on and test image processing code to identify chess pieces	Josh
	THIRD PCB ORDER MARCH 26TH	Everyone
April 3rd -April 9th	Revise PCB design if necessary	Jose
	Begin work on the image processing to chess AI pipeline	Jose
	Test and finalize programming microcontroller to decode Raspberry PI instructions to move motors and flip magnet switch	Zack
	Test and finalize work on image processing code to identify chess pieces	Josh
	FOURTH PCB ORDER APRIL 4TH	Everyone
April 10th - April 16th	Test and finalize the image processing to chess AI pipeline	Jose
	Full project testing	Zack
	Revise PCB design if necessary	Josh
	FIFTH AND FINAL PCB ORDER APRIL 11TH	Everyone

April 17th - April 23rd	Fix any minor bugs in code or anywhere else	Everyone
April 24th - April 30th	Final Demo	Everyone

Table 8. Project Schedule

Ethics and Safety

3.1 Overview

As we create the AI chess board, we must worry about the software and hardware components that will make up the project from license coverages to patents and copyrights.

3.2 Concerns about Chess Algorithm

Creating a chess algorithm from scratch to evaluate countless chess moves and how optimized they are for victory can be challenging and time-consuming; it may require time that is out of scope of a semester's worth of time. As a result, we will be assisted by Python's chess library "python-chess" to compute moves and their varying efficiency. Because we are using a library, there is a need to be aware of the potential licensing conflicts. The library has a GPL v3 license which means that it can be involved in commercial use. In accordance to a GPL v3 license (Free Software Foundation, Inc., 2007), but not limited to:

- Terms and Conditions, Section 4, Paragraph 2:
    - You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.
- Terms and Conditions, Section 7, Group C:

    - Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version.
- Terms and Conditions, Section 8, Paragraph 1:
    - You may not propagate or modify a covered work except as expressly provided under this License.

3.3 Concerns about Hardware Components

We are utilizing a Raspberry Pi to act as a microcontroller in our project's design and with it comes their terms for usages (Raspberry Pi, n.d.). Under Raspberry Pi trademark rules and brand guidelines, they explicitly mention a list of allowances and prohibitions that deal with Raspberry Pi and all they own. If our project ever decides to commercialize, we will need to contact them to obtain a license. We can use the Raspberry Pi's word mark to refer to products or services, or to describe that there is compatibility between products. We cannot use the logo unless it is connected to sale or distribution of genuine products. The Raspberry Pi marks must be less prominent than what it is used for/connected to.

For the MIPI camera, we can use the product for personal use according to the MIPI Alliance's Frequently Asked Questions for the ECE 445 project, but if any desires for commercialization occur, we will stay within their boundaries for intellectual property and more. Lastly, for the CAD models of the chess pieces, the designs we are most likely going to use can be found here and are under a BY-NC-ND 4.0 DEED creative commons license. Since we are not selling this chess robot, will give credit, and will not distribute our modifications, we are following the terms of the license agreement. If we were to sell this commercially, we would end

up hiring someone to design the chess pieces or find a different design online that would allow for commercial use.

3.4 Concerns about Ethical Practices

The Committee on Professional Ethics (COPE) promotes ethical conduct amongst the computing professionals with a code of ethics. In accordance to their code of ethics (ACM Ethics, 2018), some of the guidelines include:

- PRINCIPLES, Principle 1: PUBLIC, Section 1.01:
    - Accept full responsibility of the work
- PRINCIPLES, Principle 1: PUBLIC, Section 1.03:
    - Approve software if it does not diminish the quality of life, privacy, or harm the environment
- PRINCIPLES, Principle 1: PUBLIC, Section 1.06:
    - Be fair and avoid deception in all statements, particularly public ones, concerning software or related documents, methods and tools.
- PRINCIPLES, Principle 3: PRODUCT, Section 3.02:
    - Ensure proper and achievable goals and objectives for any project on which they work or propose.
- PRINCIPLES, Principle 5: MANAGEMENT, Section 5.01:
    - Ensure good management for any project on which they work, including effective procedures for promotion of quality and reduction of risk
- PRINCIPLES, Principle 6: PROFESSION, Section 6.06:
    - Obey all laws governing the project and its components

We will uphold the code of ethics to the best of our ability and not tarnish the University of Illinois Urbana Champaign's reputation.

References

ACM Ethics, "Software Engineering Code," ACM Ethics, Dec. 19, 2018. https://ethics.acm.org/code-of-ethics/software-engineering-code/

Arducam IMX219 Auto Focus Camera Module, drop-in replacement for Raspberry Pi V2 and NVIDIA Jetson Nano Camera. (n.d.). Arducam. https://www.arducam.com/product/arducam-imx219-auto-focus-camera-module-drop-in-replacement-for-raspberry-pi-v2-and-nvidia-jetson-nano-camera/

"DATASHEET Raspberry Pi 4 Model B Release 1," 2019. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf

"Frequently Asked Questions | MIPI," www.mipi.org. https://www.mipi.org/resources/frequently-asked-questions#f

GNU general public license v3.0. Choose a License. (2024, February 14). https://choosealicense.com/licenses/gpl-3.0/

["LM3940 1-A Low-Dropout Regulator for 5-V to 3.3-V Conversion Input Voltage Range: 4.5 V to 5.5 V • Output Voltage Specified over Temperature • Excellent Load Regulation • Specified 1-A Output Current • Requires only One External Component • Built-in Protection against Excess Temperature • Short-Circuit Protected 2 Applications • Laptop and Desktop Computers • Logic Systems 3 Description." Accessed: Apr. 04, 2024. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm3940.pdf?ts=1712148575684&ref_url=https%253A%252F%252Fwww.mouser.it%252F

Raspberry Pi Camera Module 1/4-Inch 8-Megapixel Module Datasheet. (n.d.). Retrieved February 9, 2024, from https://www.arducam.com/downloads/modules/RaspberryPi_camera/RaspberryPi_IMX219_8MP_Camera_Module_DS.pdf

Raspberry pi trademark rules and brand guidelines - raspberry pi. (n.d.). https://www.raspberrypi.com/trademark-rules/

Schematic Checklist - ESP32-S3 - — ESP Hardware Design Guidelines latest documentation. (n.d.). Docs.espressif.com. https://docs.espressif.com/projects/esp-hardware-design-guidelines/en/latest/esp32s3/schematic-checklist.html

“Stepper Motor Basics,” Oriental Motor U.S.A. Corp. https://www.orientalmotor.com/stepper-motors/technology/stepper-motor-basics.html#:~:text=Inductance%20affects%20high%20speed%20torque (accessed Apr. 04, 2024).

“Stepper Motor with Cable - ROB-09238 - SparkFun Electronics,” www.sparkfun.com. https://www.sparkfun.com/products/9238