Dots and Boxes: Recreation of a Classic Game

A Technical Report submitted to the Department of Electrical and Computer Engineering

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Jacob Ryan Taylor

Spring, 2023

Technical Project Team Members

Joseph Lee

Boheng Mu

James Tsai

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Harry Powell, Department of Electrical and Computer Engineering

Statement of work:

Joseph Lee

My primary role for the project was processing output from the game code to draw the corresponding actions on the light emitting diode (LED) matrix. Initially, the microcontroller would communicate via serial peripheral interface (SPI) to a driver chip that would interface with the LED matrix. I created C code that would create the necessary SPI signals to interact with LEDs on the matrix, but when testing with the driver and matrix, there were problems that we were unable to debug due to lack of documentation. We decided to communicate directly to the matrix without the driver chip. I worked with Boheng to develop this new method of writing directly to the matrix which involved optimizing the code to run faster to get a result with little aliasing and flickering on the matrix.

My secondary role was designing and printing the 3D printed enclosure that would house all the components of the project. Additionally, I worked with Jacob to assemble and test the printed circuit board (PCB).

Boheng Mu

My primary role in the project was designing and implementing game logic. Some of the tasks include processing the input from James, designing the player movement mechanics, and the output routine. I also coordinated the integration of the embedded code between teammates, managed the collaborative GitHub code base, delegated pins mapping needed on the MSP, and designed the interrupt routine that will run the entire game. For the game logic, I recreated the game in C code from scratch, designing the overarching architecture of the code. At first, I designed the game for terminal output and keyboard due to the lack of an LED board and input components. Then I transformed the game to fit with the actual components that we have and integrated it with the code that other teammates have written.

Along with the game design, I was also tasked to finish the LED matrix implementation, with the help of Joseph. Which involved finding a way to drive the LED board directly through its 16-pin interface. I wrote the routine that would refresh the board with the desired image of a 32 x32 matrix. Additionally, I implemented anti-ghosting methods and, with the help of Joseph, optimized the code to prevent flickering.

Jacob Taylor

My primary role for the project was the design and assembly of the hardware. The hardware includes the power supply, input devices, scoreboard, and LED matrix. My job was to design an overall schematic with different hierarchical blocks to represent the subsystems on KiCad, design the footprints for parts that did not have an ECAD model, design of the PCB, and assembly of the PCB. I was also responsible for the research of parts that were needed along with helping get those ordered. I fully soldered the PCB was used various test points and header pins to make our hardware testable before application.

My secondary role was to help design the 3D printed enclosure that the hardware would sit in. I also helped with the construction and assembly of said enclosure.

James Tsai

My primary role for the project was processing the user input devices: the joystick, rotary encoder, and push button. I created the initial input schematics using KiCad, and I wrote the interrupt routines for the inputs and implemented a queue for the inputs to be read by the game logic correctly using embedded C programming. I worked with Boheng to create initial testing for the input devices by verifying expected output in the Code Composer Studio (CCS) terminal, as well as verifying functionality with the LED matrix using a breadboard.

My secondary role was to program the LED matrix game messages, which include the scoreboard, error messages, and game winner messages. Based on the current game state and processed inputs, the messages were displayed at the bottom of the LED matrix in various colors.

Table of Contents

Contents

Statement of work: 2
Table of Contents 5
Table of Figures 6
Abstract 8
Background 8
Physical Constraints 10
Design Constraints 10
Cost Constraints 11
Tools Employed 11
Societal Impact Constraints 12
Environmental Impact 12
Sustainability 12
Health and Safety 12
Ethical, Social, and Economic Concerns 12
External Considerations 13
External Standards 13
Intellectual Property Issues 13
Detailed Technical Description of Project 15
Hardware Technical Description 15
Software Technical Description 23
Project Time Line 27
Test Plan 30
Final Results 32
Costs 33
Future Work 33
References 34
Appendix 39

Table of Figures

(This should list the page of each figure used in your document, including the full caption.) Word has tools to help you do this very easily)

Figure 1. Dots and Boxes Movement 6

Figure 2. Power Supply Schematic 14

Figure 3. Joystick Schematic 15

Figure 4. Rotary Encoder Schematic 15

Figure 5. Pushbutton Schematic 16

Figure 6. Single Scoreboard Schematic 17

Figure 7. LED Matrix Connection 18

Figure 8. MSP432 Schematic 19

Figure 9. Overall Schematic 20

Figure 10. Board Layout 20

Figure 11. Fully Populated PCB 21

Figure 12. Control Algorithm Flow Chart 23

Figure 13. Proposed Gantt 26

Figure 14. Finalized Gantt Chart 27

Figure 15. Input Test Plan 28

Figure 16. Output Test Plan 29

Abstract

This project will recreate the classic game dots and boxes using electric components. The system will contain a light emitting diode (LED) array for display, a joystick and rotary encoder for players' input, a push buttons to confirm players' moves, two 7-segment displays as the scoreboard, a microcontroller to control the electrical components, and a software program to determine the logic of the game. The microcontroller will deterministically receive player input from the joystick, rotary encoder, and button, then pass it to the software program where the game state will be updated and reflected on the LED array and the scoreboard. This project transforms a game that is normally played with paper and pen into a more efficient and autonomous platform, providing a better experience.

Background

Introduction to Dots and Boxes

Dots and Boxes is a turn-based, zero-sum game between two players. It was first theorized in 1981 by a French mathematician [1] and is now enlisted as a chess game for the Computer Olympiad. Despite its simple rules, the game offers a multitude of outcomes and strategies and is now the playground for many artificial intelligence and machine learning researchers. The rules of the game are as follows:

(1) Players take turns drawing lines between dots on a *n* by *m* grid

(2) If a player completes a box, they claim the square and may draw another line until they cannot complete a box (Figure 1)

(3) Player with the most claimed boxes wins

(4) Some adaptation of the game sets a timer for both players, when one timer has reached its end, the remaining blocks will be awarded to the player with time remaining.

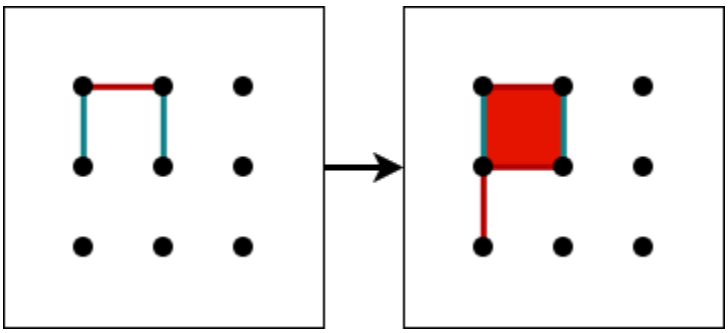

Figure 1. Dots and boxes movement

Motivation

This game is far from trivial and stimulates the brain by computationally challenging the players to obtain the greatest number of boxes. There has been a widespread usage of arcade entertainment systems, but unfortunately, there is no noteworthy adaptation of this simple, yet intriguing, game. Additionally, setting up the game on pen and paper requires much overhead, and throughout the game, players must keep track of the ownership of the square. Such limitations lead to paper and pen games usually smaller in size and unable to demonstrate the true marvel of the game, where its complexity increases exponentially with size.[2] For this reason, we want to recreate this classic game with electronic components where players can focus on the game play with an intuitive interface.

Previous Work

There are numerous previous works on the deconstruction of the game [1, 2, 3], and it is more focused on the gameplay and how to win the game with strategies. Numerous adaptations of the game have been found on the market including John Sands' board game adaptation and many web versions of the game. [4] Currently, there's no widely available physical and reusable board for dots and boxes, some remaining copies of the John Sands' version can be found on secondhand enthusiasts' market. Our version of the dots and boxes will offer players an easy interface to select moves and alleviate the burden of constructing the board or the need to keep track and count the squares. Additionally, in contrast to the complete software version, our adaptation invites players to play face-to-face in person, with a physical interface. This is much needed today where our lives are increasingly virtual and isolated.

Course Background

In order to complete this project, we will draw on many experiences and knowledge from previous courses that we have taken. We will use our knowledge of basic circuit design, printed circuit board (PCB) design, circuit testing, and power delivery from the Fundamentals of Electrical Engineering series (ECE 2630, ECE 2660, ECE 3750). We will use our knowledge of embedded computing, deterministic input/output, microcontrollers, and motor control, from the Embedded Computing and Robotics series (CS/ECE 3501, CS/ECE 3502). Furthermore, robotic logic design and state representation knowledge will be used from the Embedded Computing and Robotics series and Digital Logic Design (ECE 2330). Reconstruction of the game in software will draw on many computer science courses taken including Software Development Methods (CS 2110) and Program and Data Representation (CS 2150) for data structures and coding basics. We will be drawing from Advanced Software Development Methods (CS 3240) for software development skills. With the synthesis of the aforementioned courses, we will construct a working replica of the dots and boxes game.

Physical Constraints

Design Constraints

There were many design constraints set by the Capstone course. Since the design team contained both electrical engineering and computer engineering majors, there was a requirement that the device had to combine hardware and an embedded system to satisfy all requirements for the major design experience put forth by the University of Virginia. Each design team was also limited to two PCB send outs out of three possible dates, October 17, October 31, and November 15. This would imply that the design of the PCB had to be completed by November 15.

CPU Limitations

The microcontroller used in the project is the MSP432P401R [35]. This microcontroller was used since it contained a sufficient number of General Purpose Input/Output (GPIO) pins, as well as a 14-bit resolution analog to digital converter (ADC14). The CPU is limited by its ADC14 having a maximum reference voltage of 3.3V without using external voltage sources. This constraint led us to add a 3.3V linear regulator to supply the voltage input for the joystick input device.

Software Availability

All software used was readily available for free download online. KiCad [8] was used to create schematics for the PCB design. CCS [5] was used to write the embedded C code for the user inputs, LED matrix, and game logic. GitHub[7] was used for software management and version control online.

Manufacturing limitations

The PCB was manufactured by Advanced Circuits [23] so manufacturing limitations were to be followed according to the rules required to receive a discounted price for 2-layer boards. The most relevant limitations that needed to be followed were tolerances for minimum silkscreen line width, minimum 1 oz. copper trace width, and annular ring size for vias and component holes. The following shows the tolerances for each respectively.

- 5 mil minimum line width
- 3 mil minimum trace width
- 5 mil annular ring minimum for vias
- 7 mil annular ring minimum for component through holes

The enclosure was 3D printed using a Creality Ender-3 Pro [37] which has a maximum build volume of 220mm length, 220mm width, and 250mm height. This volume constraint would limit the size of prints that could be achieved in a single gcode file sent to the printer.

Part Availability

All parts for hardware except the MSP432P401R were obtained through DigiKey Electronics [11]. Several basic components such as the inductor for the power supply proved hard to find suitable replacements for, and an LED matrix that interfaced with the

microcontroller was difficult to find as well. All chosen parts were chosen based on price, datasheet specifications, and usability.

Cost Constraints

Fairly Unique Name 4 was given a budget of $500 to purchase the necessary components for the project. For our project, it was estimated that close to $100 of the budget would be spent on the LED matrix as it was the biggest and most complex component of the project. Due to the group already having access to a microcontroller for the project, much of the budget remained open to use and as such, duplicates of components such as the input devices were purchased for synchronous development.

Tools Employed

Hardware

The main tool for the design of hardware was KiCad. KiCad is a design suite that allows the editing of schematics, symbols, PCB, and footprints, all of which were used for this project. KiCad also allows for the use of hierarchical blocks and global labels to help better organize schematics, as well as a symbol editor and library to edit or create new symbols for components that are imported into the project file. The same applies with the PCB editor and footprints for components. The most important skill that was needed to effectively use KiCad was the global and local libraries that were required for parts that were imported in or created using the symbol and footprint editor.

Texas Instruments' WeBench was used to help design the power supply for the project [25]. Since a TI brand switching and linear regulator was used for the 5-volt and 3.3-volt supply rails, WeBench offered recommended component selections given a set of parameters such as input and output voltages and currents. The National Instruments Virtual Bench [39] was used to test outputs of the MSP432 as well as verify functions of the PCB when assembled.

Software

All the embedded software was written in C using Code Compose Studio [5]. The MSP432 Driver Library [18] provided many basic functions that were used. GitHub [7] was used for version control in a central online storage. The main software skills that were improved involved finding various methods to speed up code, and adhering to the Barr Coding Standard [16]. Game design was a new skill that was acquired; integrating inputs, game logic, and outputs were key to creating a cohesive game.

Mechanical

Autodesk Fusion 360 [25] was used to design the 3D printed enclosure and Ultimaker Cura [40] was used to slice the model to 3D print the necessary components. The Creality Ender-3 Pro was used to print the components used in the housing for the final enclosure. Due to the limited build volume of the Ender-3, the CAD model had to be segmented and printed in smaller pieces and assembled when all the pieces were printed. Joseph was familiar with these programs and the Ender-3 but had to improve his knowledge on the Ender-3 to get higher quality prints. Additional skills were developed in designing for 3D printing components efficiently while

minimizing printing errors. A rear handed circular saw was used to cut the wood base of the enclosure.

Societal Impact Constraints

Environmental Impact

The environmental impact implications of the dots and boxes game come from the waste generated from 3D printing the physical device housing, and the disposal of the entire device once it is no longer functional or used. Depending on the type of 3D printer used, the excess materials used to 3D print could be disposed of in a landfill or a body of water. This leads to pollution of terrestrial, aquatic, and atmospheric systems [12]. The other major waste component is generated from recycling the PCB and microcontroller when disposing of the entire device. Most of those parts are burned to recover the valuable metals used in production, and the residues are disposed of in a landfill [13]. Proper recycling procedures must be followed to ensure that disposal results in minimal waste. Possible methods include desoldering electrical components from the PCB before disposal and using biodegradable 3D printing filament.

Sustainability

The lifespan of this device depends heavily on the durability of the 3D printed enclosure. Due to the size of the 3D printer employed, the enclosure had to be printed and assembled in multiple pieces, which inevitably makes the overall structure weaker. Another source of wear would come from the prolonged use of the input devices. An example of this would be the lifetime of the rotary encoder being rated for only 30,000 cycles. Since the rotary encoder is used twice per move on average, this could lead to a premature destruction of the device if the encoder is used more than it must be. Since all the components in the device are either 3D printed or electrical, it would be difficult to replace or repair these parts without having the proper resources or altering electrical connections. Some design changes could be made if this device were to go into mass manufacture that could help limit these problems. A cheaper, more durable design for the enclosure could be implemented to help increase drop protection, and different input devices that have a longer lifetime could be used to limit the frequency of repair or replacement. A manufacturer could also change the design of the circuitry to make individual components less permanent so a user would not have to desolder in order to replace a worn component.

Health and Safety

All components of the device will be protected from abrasion, and all openings to electrical components shall be effectively closed to minimize the risk of electric shock and electrostatic discharge [14]. The device will minimize sharp corners in the physical housing, to prevent cuts on skin. Since the physical device housing is 3D printed, it acts as an insulator, which prevents current flow to the user. It is possible that small components unintentionally detach from the device and can pose a choking hazard for young children.

Ethical, Social, and Economic Concerns

The applications of this device and the cost of its components make it unsuitable for consumers of lower socioeconomic status, due to high retail price and its main intended usage as

an entertainment device reducing the usability of such consumers. This device is most suitable for classroom and amusement venue environments, which can promote gaming and be seen as inappropriate for those who are against digital gaming. Since this device would be seen as too expensive for consumers of lower socioeconomic status, the application of this device in the classroom may only be available to more privileged schools. Lower privileged schools often lack funding, so they would not have the extra resources needed to supply a classroom with this device. Also, many lower privileged schools are attended by minorities and children of color, so this device could also cause some major discrepancies in the quality of education that these students receive [22].

The main components with the highest cost are the PCB, LED array, user input components, wall transformer, and microcontroller. The total cost of those components is estimated to be under $300, and considering subcomponents, the total cost is projected to be well under $500. If the device were to go into production, the major economic constraint would be the supply of LED arrays. There is limited availability of LED arrays that have the programmable input type that works with our microcontroller. Other components can be produced in mass quantities with minimal issues.

External Considerations

External Standards

The PCB design and manufacturing process will follow the standards set by IPC-2221 for general board design [15]. The component mounting and interconnection structures will follow the same standards.

Programming the device using embedded C/C++ will follow the Embedded C Coding Standard set by the Barr Group to minimize bugs in firmware and improve the maintainability and portability of embedded software [16].

If this device were to be used in an amusement venue, it would have to fall under the United States Department of Labor Occupational Safety and Health Administration (OSHA) Industry Group 7999 standards for Amusement and Recreation Services [17]. Future iterations of this design could lead to its classification of a device used in an establishment engaged in providing amusement or entertainment services.

Components that are housed in the 3D printed frame will be protected from abrasion, and any openings in the frame will be effectively closed, according to OSHA standard 1910.305(b)(1)(i) [14].

Intellectual Property Issues

[19] references a patent for a “dot game device”, similar in game rules to our Dots and Boxes implementation. What the patent claims is new is "a dot game device comprising a gameboard having a generally planar upper surface and including a matrix of horizontally and vertically oriented dots including a central dot and a plurality of concentric rows of spaced dots

with the central dot and the first two concentric rows of dots forming a first game and the next two concentric rows of dots forming a second and third game, a plurality of equal length straight line segments for connecting adjacent dots, a plurality of equal length non-straight line segments for connecting adjacent dots and game pieces in the form of generally circular tokens positionable in enclosed areas formed by straight line segments interconnecting adjacent dots to indicate an area captured by a player when the area is completely enclosed by straight line segments." Our implementation of the Dots and Boxes game is still patentable since it utilizes electronic components to process game logic digitally, as opposed to the patent's implementation requiring users to physically place tokens and line segments. The other claims of the patent pertain to the structure of the physical game device, which differs from our capstone design in the structure of the game pieces.

[20] references a patent for "game of skill and strategy wherein at least two players alternate turns and place at least one game token in at least one open border slot surrounding one or more playing zones". What the patent claims is new is "a method of playing a game between a plurality of players using a game board, said game board including a playing area divided into a plurality of zones by a plurality of border slots, each of said slots adapted to receive at least one game token therein, and said zones sized to receive at least one game token therein, said method comprising the steps of: in repeating and alternating turns of each player, each player placing at least one game token in said at least one of said border slots, said at least one game token converting an open border slot into a closed border slot; claiming one or more zones by placing at least one game token within one or more of said zones when one of said players converts the last open border slot to a closed border slot completely surrounding said one or more of said zones; and determining a winner of said game by evaluating multiple winning criteria based any combination of a collection of groups of slots, a collection of groups of zones, or a number of zones claimed by each player. Our implementation of the Dots and Boxes game is still patentable since it is an original and unobvious improvement to this patent. Our design includes a physical device that uses a digital interpretation of game pieces rather than physical tokens to implement the game strategy detailed in the patent. The other claims of the patent pertain to additional rules of the game and methods to play the game, which are also encompassed in our design.

[21] references a patent for "an apparatus which may be used to demonstrate the effect of adding and subtracting colors and which may also be used as a tick-tack-toe-like game". The patent has the independent claim for "A three-dimensional tick-tack-toe-like game apparatus comprising: a square frame member having four sides and an open top portion; located within said frame member; a first track means connected to said frame member wherein said first track means is parallel to one pair of said sides and wherein said first track means forms a first preselected number of columns; a second track means connected to said frame member wherein said second track means is vertically spaced from said first track means and wherein said second track means forms a second preselected number of columns; a first preselected number of light transmitting elements having preselected colors slidably mounted within each of said columns of said first track means wherein said elements form a first layer having said first preselected number of columns and a first preselected number of rows, said first preselected number of rows being determined by the number of elements in one of said columns whereby said first layer

comprises a preselected number of squares having a preselected color pattern; a second preselected number of light transmitting elements having preselected colors slidably mounted within each of said columns of said second track means wherein said elements form a second layer having said second number of preselected columns and a second preselected number of rows, said second number of rows being determined by the number of elements in one of said columns whereby said second layer comprises a preselected number of squares having a preselected color pattern; a light source located within said frame member and adjacent to said second layer of elements which illuminates said first layer of elements through said second layer of elements; and additional light transmitting elements having preselected colors wherein said additional elements are adapted to slidably replace any one of the outer elements in any row or column of said first layer and said second layer thereby changing the color of one of said squares when viewed through said open top portion." This patent could be comparable to the implementation of the LED matrix in our game representation. However, our design is patentable since it presents an original usage of light emitting elements for a game, as well as original input methods for the game. The patent assumes users play the game physically by placing elements on the game board, but our implementation of the game involves users using the input devices (joystick, rotary encoder, push button) to control player movement. The other claims in the patent pertain to how a tick-tack-toe game is implemented with the light emitting elements, which is unrelated to our capstone design.

Detailed Technical Description of Project

The system design can be divided into the main categories:

1. Hardware
    a. Components
    b. Schematics
    c. PCB Design
2. Software
    a. Game Algorithm
    b. User Input Processing
    c. Output Processing

Hardware Technical Description

The hardware for this device includes a PCB that contains components and connectors to each individual subsystem for the overall system. The subsystems include the input controls, scoreboard, microcontroller, power supply, and LED matrix. The design of these subsystems is outlined and detailed in the sections that follow.

Power Supply

The power supply input is a 12V 60W wall transformer [26] that connects to the PCB via a barrel jack connector [27]. The 12V power is then stepped down to 5V and 3.3V using a 5V switching regulator [28] and a 3.3V linear regulator [29]. The 5V switching regulator utilizes two input capacitors at 47uF and 1uF. It is important to have low ripple voltage presented at the input of the regulator so the large values of capacitance account for this. The design choice of two

capacitors in parallel deals with the minimum capacitance needed since ceramic capacitors tend to have high voltage coefficients. The regulator also makes use of an output capacitor at 270uF, an inductor at 3.3uH, a boot-strap capacitor at 0.01uF, and a Schottky diode rated for 150V at 10A. These parts act as the most vital external components of the switching regulator and allow the regulator to experience low losses. The 3.3V linear regulator makes use of an input capacitor at 2.2uF, an output capacitor at 1uF, and a noise reducing capacitor at 0.01uF. All design choices for the power supply were determined using the desired output voltages and current, datasheet specifications, and TI WeBench recommendations.

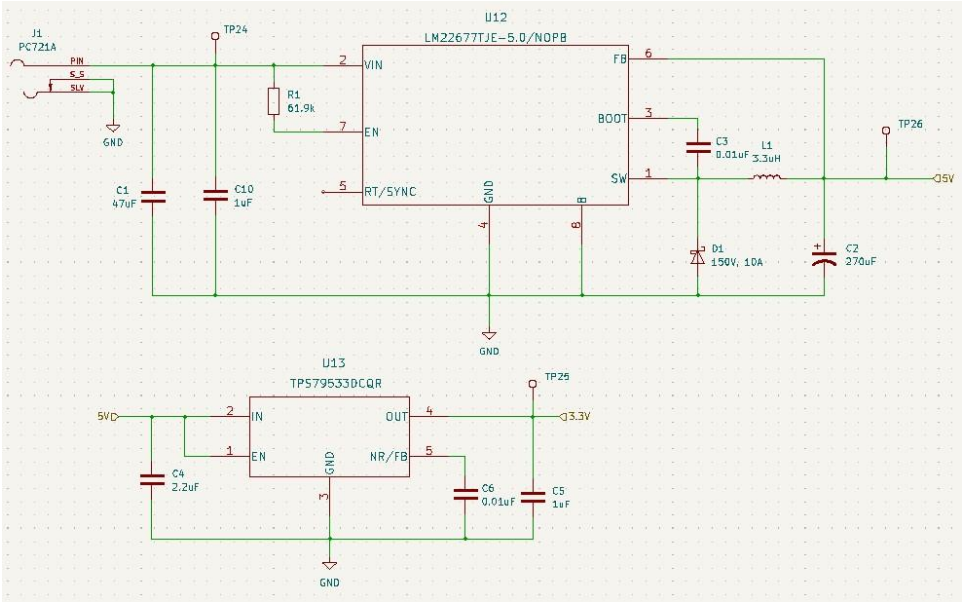

Figure 2. Power Supply Schematic

Inputs

In order to make the software development for the game as easy as possible, three inputs were chosen to be used to allow for an intuitive feel for playing the game. A two-axis joystick [30] was selected to be used as a cursor that would hover over a dot in the grid. A rotary encoder [31] was selected to allow the user to rotate around the dot that was being hovered over to help select an edge. A pushbutton [32] was selected to allow the user to lock in their choice of line.

The joystick is an analog output device that consists of two potentiometers, one for each axis. As the user moves the joystick in certain directions, the changing output voltages would be fed to the MSP432 ADC to be processed as moves.

The pushbutton is simply a button that was designed to pull down the voltage being presented to the microcontroller when pressed. This presents an inherent danger to the microcontroller as voltage spikes and over currents have potential to be delivered to it. Input protections such as transient voltage suppression (TVS) diodes [38], current limiting resistors, and a capacitor that also protects against over voltage and smooths change in value that is presented to the microcontroller. Since the rotary encoder presents the same kind of input into the microcontroller, the same input protection was used for it as well.

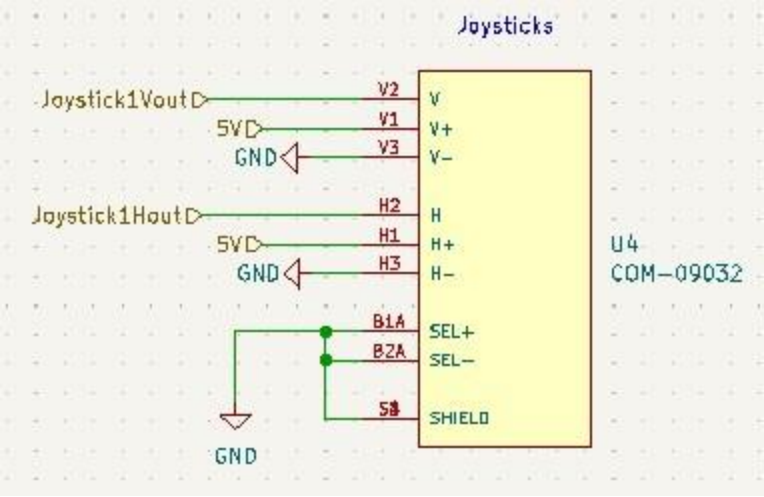

Figure 3. Joystick Schematic

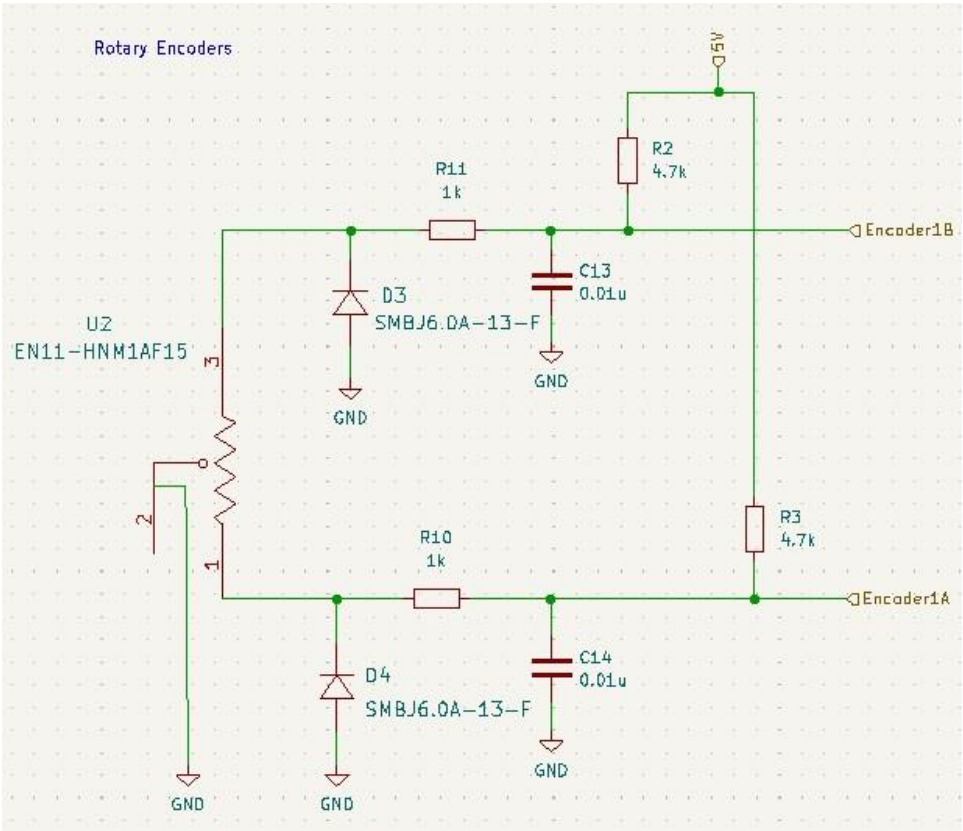

Figure 4. Rotary Encoder Schematic

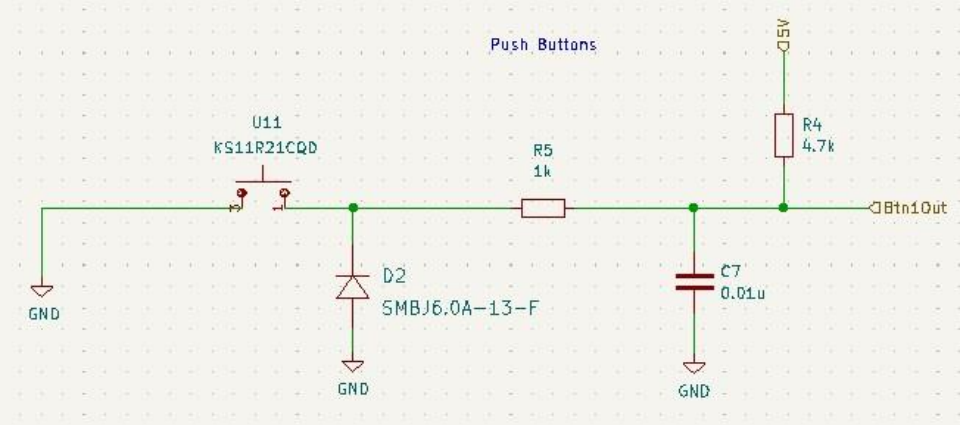

Figure 5. Pushbutton Schematic

Scoreboard

The scoreboard consists of two, 2-digit 7 segment displays [33] and accompanying driver chips [34]. The 7 segment displays were a common cathode configuration. The driver chips used I^2C to communicate to the microcontroller and display the needed score on their respective 7 segment displays. Bypass capacitors at 0.1uF were used for each as well as a 22pF capacitor to set the internal oscillator in the chip to allow for blinking if necessary. A 56kΩ resistor was used to set the maximum current that was to be delivered to each segment on the 7 segment displays as well as a recommended pullup resistor on the data line to the MSP432. It is important to note that the software for these chips did not work for the end product, but it is still necessary to mention them in the overall schematic.

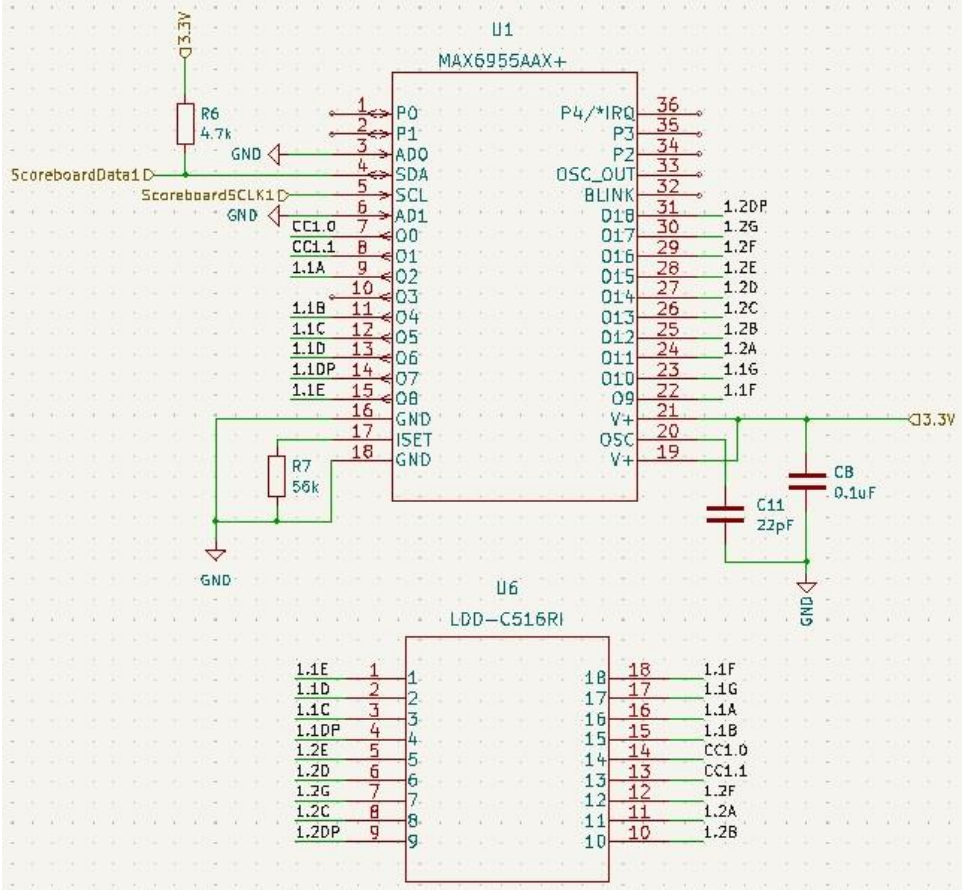

Figure 6. Single Scoreboard Schematic

LED Matrix

Function	MSP Port Pin	Function	MSP Port Pin
R1	3.2	G1	6.1
B1	3.3	GND	GND
R2	4.1	G2	4.0
B2	4.3	GND	GND
A	1.5	B	4.2
C	4.7	D	4.4
ClK	4.4	LAT	4.5
OE	5.5	GND	GND
Joystick X axis	5.0	Joystick Y axis	5.1
Encoder A	3.5	Encoder B	3.6
Push Button	5.2		

Table 1. LED matrix port mappings

The interface to the LED matrix consists of 16 pins shown in Table 1, three of which are ground pins. The 32x32 LED matrix board is driven by 4-bit multiplexing, where two lines, 16 rows apart, will be written at a time. The select bits for the rows are indicated by A, B, C, and D; A being the least significant bit and D being the most significant bit in row selection top to

bottom. R1, G1, and B1 are the color select bits for the pixels of the first[top] row, respectively, R2, G2, and B2 are the color-select bits second[bottom] row. These bits dictate if the red, green, or blue LEDs will be turned on or not. LAT indicates the end of a row being written, CLK is the clock bit, and OE is the output enabled. OE allows different displays to be daisy-chained, which is not needed for this project and always set to zero. All data pins were connected to the MSP, shown in table 1, where they were set to GPIO output pins.

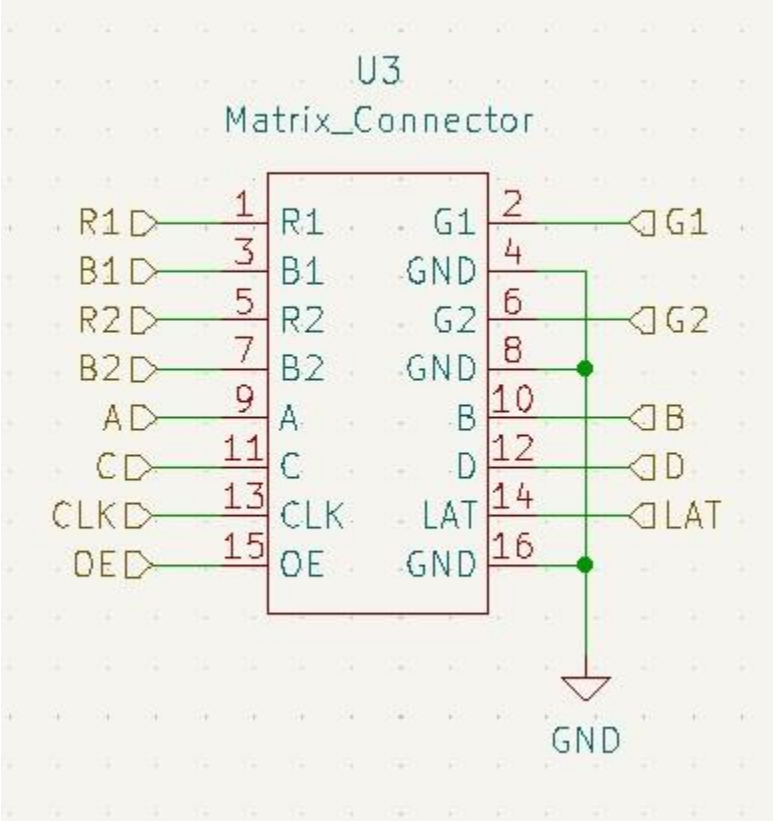

Figure 7. LED Matrix Connection

Microcontroller

An MSP432P401R was chosen for this application because it was something the design team already had. Figure 8 simply shows how the pins for the MSP432 were connected. Test points were placed on the unused pins of the MSP432 to allow for readings of outputs that would be difficult to reach otherwise. The MSP432 was powered using a 5V USB-A connector [36].

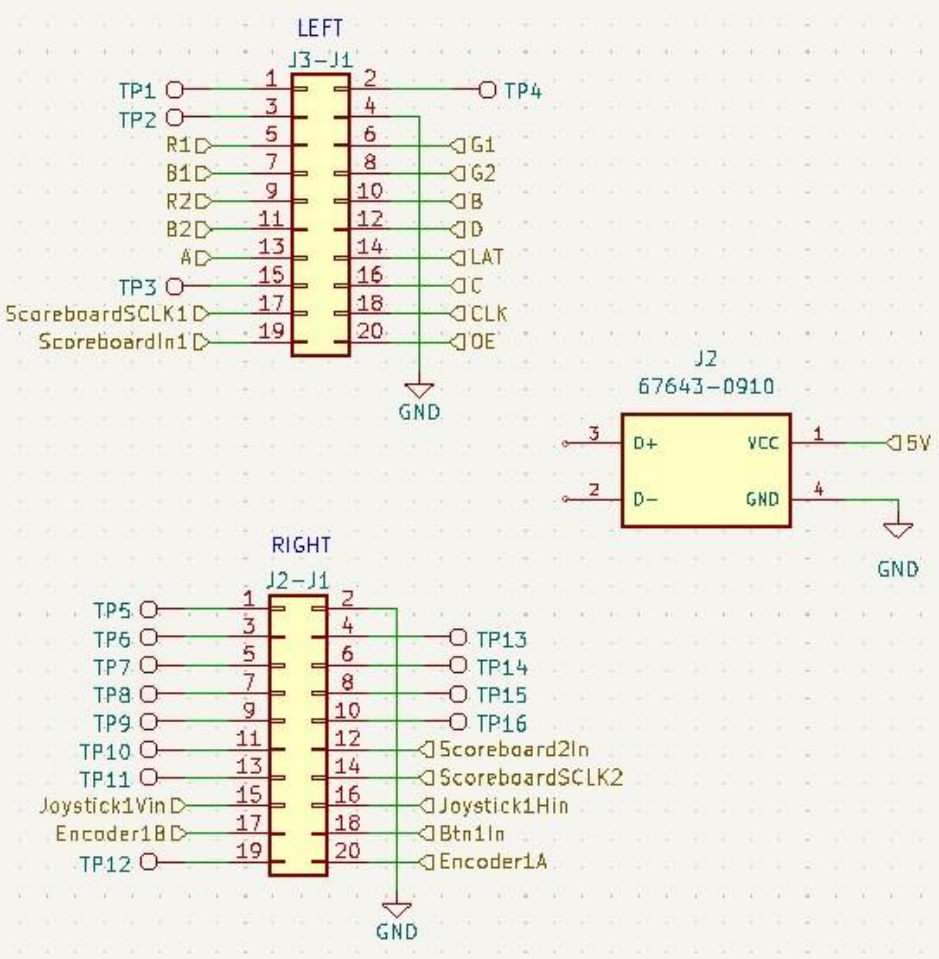

Figure 8. MSP432 Schematic

Overall System

Figure 9 shows how the overall system's subsystems interface with one another in a top-level diagram.

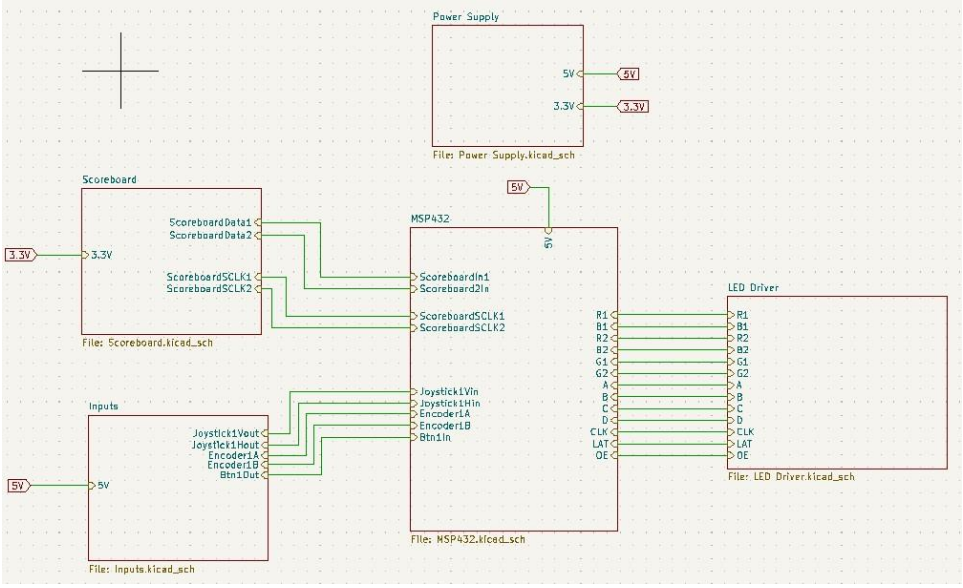

Figure 9. Overall Schematic

PCB Design

The PCB was designed using the constraints that are outlined in the physical constraints section according to Advanced Circuits. With that being said, since it was vital that the user had access to the inputs and scoreboard, the placement of those parts would need to be placed in a sensible way. The rest of the components were placed in a way where they would not interact with the major components and 3D enclosure. Figure 10 shows the final board layout.

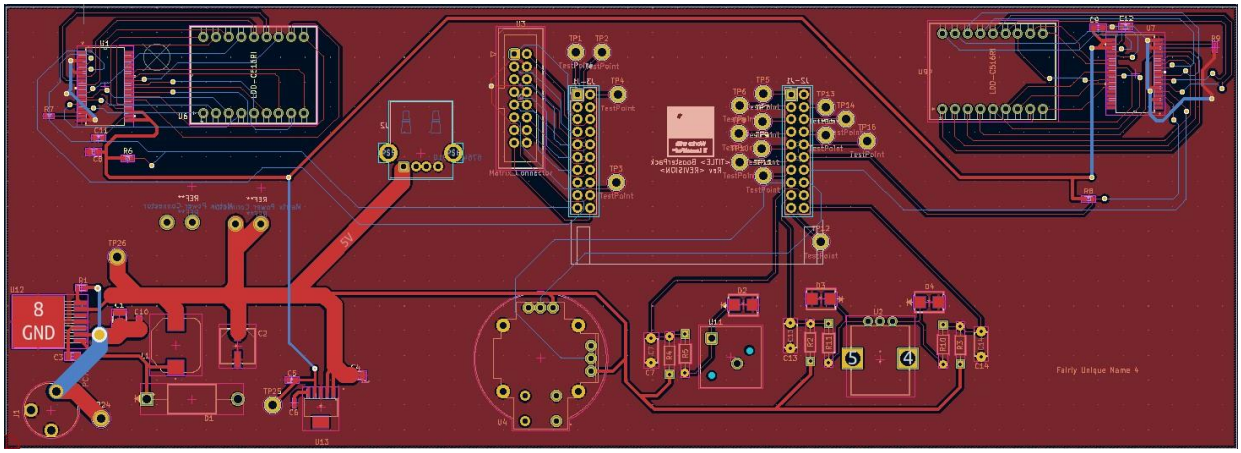

Figure 10. Board Layout

Due to the use of many surface mount components and the ease of manufacturing, a ground plane was included on the top layer which is the red zone shown in Figure 10. Once the major components were place in a sensible way, the rest of the components were placed in a way that made routing traces and placing test points easier. A trace width calculator was used to ensure that the large amounts of current needing to be delivered to certain parts of the board could be efficiently delivered. The bypass capacitors that were used for each chip were placed as

close to the input terminals as possible. Vias were used to help limit the amount of long, winding traces that would have surrounded the perimeter of the PCB had they not been used otherwise. This PCB also had to be small enough to fit inside of a 3D printed enclosure. Spade connectors that were not shown on the schematic were added so the LED matrix could be powered from the PCB instead of another wall outlet. Figure 11 shows the fully populated board.

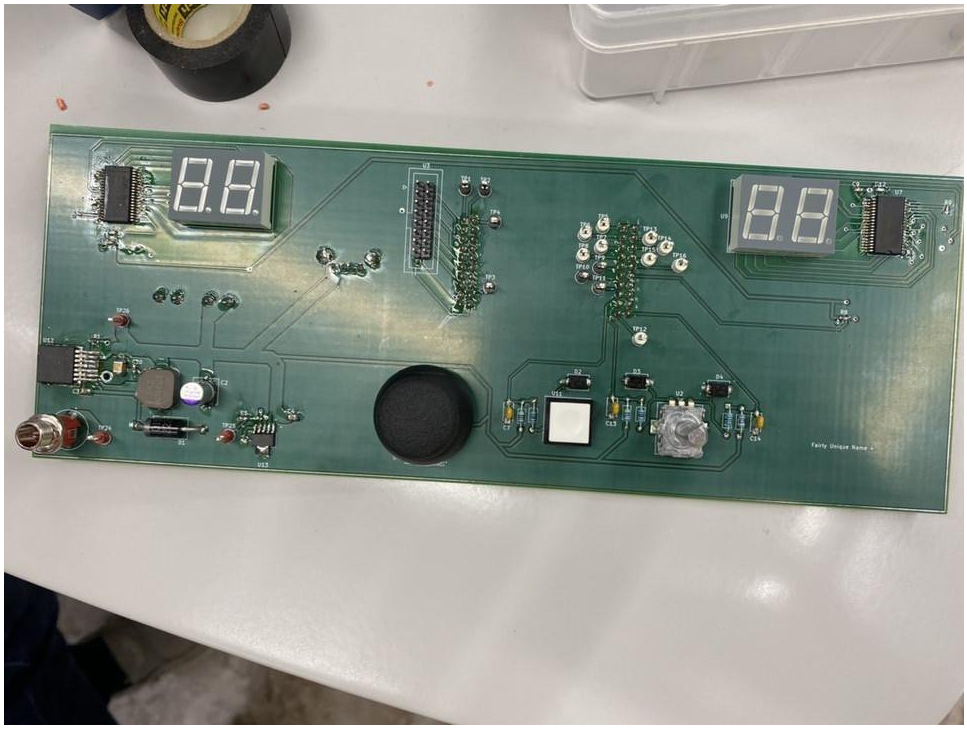

Figure 11. Fully Populated PCB

Software Technical Description

User Input Processing

User input processing involves obtaining data from the user input devices (joystick, rotary encoder, push button) and converting that data into various states that are put in a queue. The queue is processed by the game algorithm, which interprets the moves and sends instructions to the output devices (LED matrix, scoreboard).

The COM-9032 joystick is used to select a dot in the game. It consists of two 10k Ohm potentiometers that vary a voltage input between 0 and 5 volts. The neutral position of the potentiometers is interpreted as 2.5 V. To process the voltage inputs from the joystick, the ADC14 functionality of the MSP432P401R is used. The ADC14 is configured for SHM pulse-mode, SMCLK clock source, sequence-of-channels mode, 32 clock cycles sample and hold time, and multiple sample conversion continue enabled. The conversion storage registers MEM0 and MEM1 are mapped to pins 5.1 and 5.0 respectively, with MEM1 set to end of sequence. Joystick values are sampled every 1 ms, and the converted values between 0 and 16383 are stored in their MEM registers. Since the final design powers the joysticks with the 5V supply instead of the

3.3V supply, unintentionally, the ranges for the conversion values were modified as follows: neutral is MEM0 >11000 && MEM1>11000, up is MEM0 < 5000, down is MEM0 > 14000, left is MEM1 < 5000, and right is MEM1 > 13000. Based on these expressions, the state of the joystick is stored in a global variable and processed by the input queue.

The EN11 rotary encoder input device is used to determine which line around a dot the player wanted to draw. It consists of two outputs, A and B, which generate square wave pulses 90 degrees out of phase upon rotating and making contact with the common pin (which is ground), with A preceding B for counterclockwise rotations and B preceding A for clockwise rotations. The pull up resistors are set in the PCB design, so the common pin is set to ground accordingly. To process the encoder A and B outputs, they are set to pins 3.5 and 3.6 respectively. Upon a pin 3.5 (A) interrupt, the interrupt handler checks if 3.6 (B) is high. If it is, the state of the encoder was set to clockwise, otherwise, the state is set to counterclockwise in a global variable.

The KS single pole key switch is used as a push button to lock in the player's line selection. It is configured on pin 5.2 with a pull up resistor set in the PCB. The button is polled every 5 ms, which is its standard bounce time, and if pin 5.2 is low when polled, the state is updated to be pressed in a global variable, and if the pin is high when polled, the state is set to not pressed.

The input queue was created to solve the issue of moves being processed by game logic every 100 ms. If there was no limit in processing the moves, the joystick and encoder inputs would be processed too quickly, resulting in excessive moves. Without the queue, the final move at the end of the 100 ms would be processed, which can result in unintended moves; for instance, if the joystick moved right and back to neutral within 100 ms, the move right is not processed. The input queue maps each input to an integer number and stores each move done within 100 ms in a queue, so the game logic can process only the first move.

Game Algorithm

A high-level control algorithm for game logic is shown in Figure 12 and can be broken down into four parts: input validation, game logic computation, output update, and score collection.

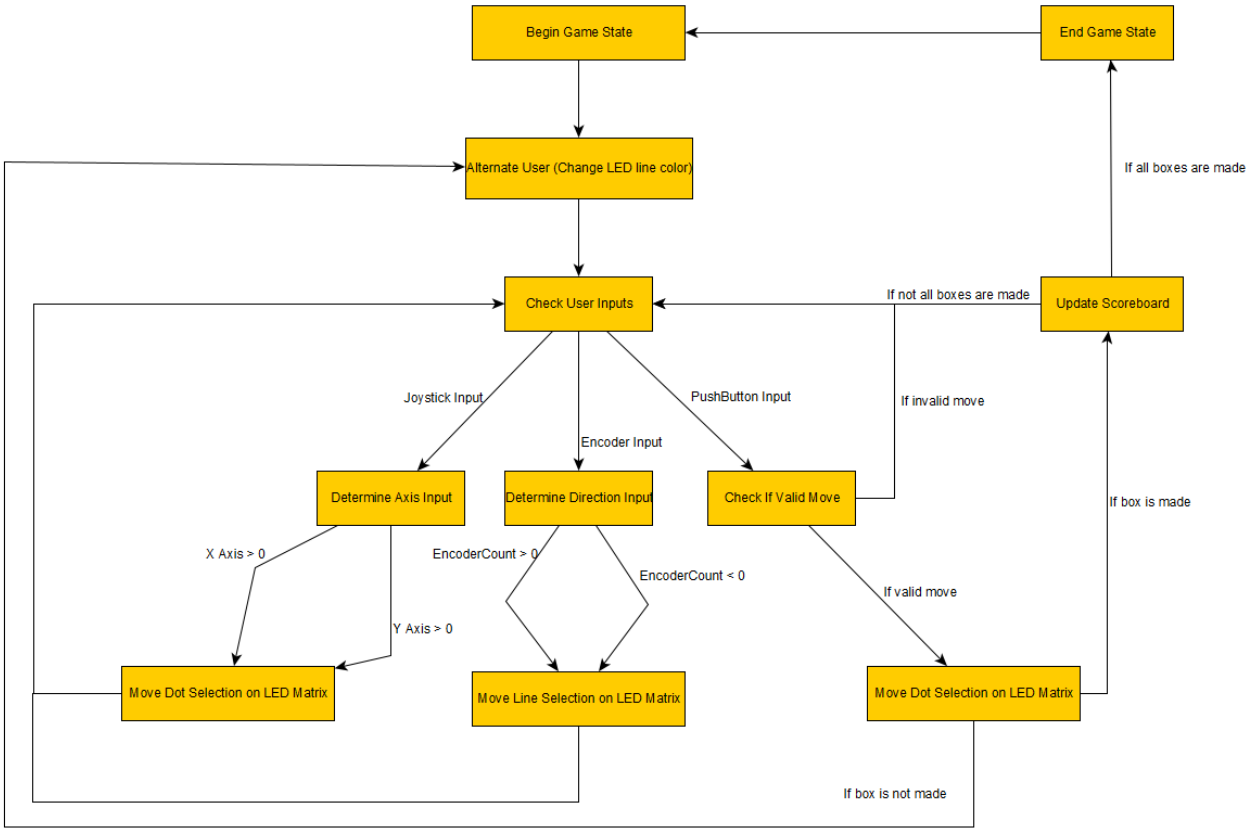

Figure 12. Control Algorithm Flow Chart

For every 100ms, the LED board the game logic processing function was called in the while loop of the main function. At the beginning of this function, a snapshot of the first input in the input queue was taken. Then this input was validated, this filters illegal moves which includes, selecting an edge that was already selected, selecting an edge when no edge is highlighted, and attempting to move out of bounds. If any illegal move was detected, then the input queue was cleared and the game waits for the next cycle.

Once a legal move is processed, they are categorized into three categories: dot movement, edge movement, and edge confirmation. Note that each category corresponds to a different input mechanism. Dot selection simply takes in the joystick values, given as up, down, left, and right, unselect the current dot, and select the dot in the desired direction. The selected dot is green and dots that are not selected are blue. The past selected the dot, and any selected edge was reverted to its original color. Selecting a new dot is done by incrementing or decrementing the selected dot coordinates by 2.

Edge movement takes in the encoder state, which is either clockwise or counterclockwise, which dictates the direction that the selected edge will rotate around the selected dot. At the beginning of the game or when the hovered dot moves, there will be no

hovered edge around the dot. In this case, the right edge will hover on a clockwise turn and left edge will hover on a counterclockwise turn. If there is already a hovered edge, then the old edge will simply be un-hovered and the next immediate edge in the given direction will be hovered. The code is written to account for dots on the corner or the edge of the board where there may be only 2 or 3 edges around the dot. An already selected edge will be teal, the current selection of edge will be red if it's a valid move and blue if it's an invalid move. An invalid move is hovering over an already selected edge that is in play.

Edge confirmation is the act of selecting the hovered edge as the player's action, this is signaled by the press of the button. This move will be invalid if there are no hovered edge, or the hovered edge are already played. In the case of a valid confirmation, then the edge will be marked as played ,turning from red to purple, and adjacent boxes will be checked for score. All four edges of adjacent boxes will be checked, and if they are all selected, then the player gets to claim that box and increment their score. The player also gets to go again. Claimed boxes will display white for player 1 and yellow for player 2. Note that the code accounts for edges that only have one adjacent box.

Finally, if an edge is confirmed as selected then a cleanup sequence will begin. First, the game logic will see if the game has ended, this is done by simply adding the score of both players and comparing it with the total number of boxes. If the game has ended, the no more moves will be processed, the winner will be displayed, and the player can press the button to reset the game. If the game has not ended, then the player will change, and the score will be updated. The input queue will be cleared.

Output Processing

Each row of the LED Matrix consists of 32 LED pixels. The end of each row data is indicated by the toggle of LAT. After which the RGB data of the 32 pixels were written, separated by a clock toggle. This process was repeated 16 times to write all 32 rows of the LED board, note that two rows were written each time. During initial testing of the LED board, there was a ghosting issue, where the image of the previous image seems to persist for a short amount of time. To combat this issue, after each refresh of the board, a routine to clear the board to complete black is called. This successfully resolved the issue and decreased the brightness of the LED board, which came as a bonus.

The LED board is divided into two sections, the bottom 5 rows arere reserved for displaying the current player, the score, and any error messages. The top 26 rows were used to display the dots and boxes, with one row reserved as a buffer between the bottom and the top section. The top section of the board is updated whenever a valid move is detected since that will cause a change in the representation of the dots and boxes. The bottom section of the board is updated whenever the button is detected as pushed since that will cause a change in the player score and the current player. Both sections reside on a large 32x32 matrix where the value in the matrix maps to a different color, shown in Table 2, but the methods that update the different sections are different. Note that these are updates to the RGB data of the pixels, the board is constantly refreshed.

Table 2. Matrix color representation

Matrix Value	Color
0	Black
1	Blue
2	Green
3	Teal
4	Red
5	Purple
6	Yellow
7	White

To drive the top section of the board, the game representation board, given by the game logic, is translated to fit the size of the LED. In the game representation board, an inner box only takes one square of the matrix, but in the LED matrix, multiple pixels could be used to represent the box. The size of the inner box is determined by the maximum size that could fit the desired box arrangement in the given LED matrix. All dots have a dimension of 1x1 pixel, and all edges have a width of 1 pixel and a length of the inner box size. Auxiliary functions were written to aid the process of drawing out the dots, the edges, and the boxes. Each of these functions takes in an x and y coordinate on the game representation board, and a color. The functions then translate the x and y coordinate into an area of LED pixel that represents the game article and writes the desired color on to the LED matrix.

The bottom section of the board is driven differently, where text displays were desired as opposed to individual pixels. Special text and digit matrices, of the size 3x5, were constructed to map the text to their respective pixel representations. A starting point was given, and the 3x5 space to the bottom and right of the starting pixel on the LED matrix would be replaced by the text matrix. The “p” and the “:” to represent which player is playing and to display the score is statically written onto the bottom section, in which the score and the player ID are updated dynamically in the game logic.

The top and bottom sections combine to make a full 32x32 representation of the LED matrix. The refresh routine of the matrix is constantly called in the background to ensure the maximum refresh speed and prevent flickering. The routine is also optimized through the usage of switch statements in row and RBG selection for maximum refresh speed. During the refresh routine of the board, the translated 32x32 matrix is constantly pulled to give the RBG data desired to be displayed.

Note that the final design did not use the 7 segment displays as the scoreboard, as the Inter-Integrated Circuit (I2C) communication protocol was not able to be implemented functionally within the project's time constraints. Instead, the scoreboard and error messages were implemented at the bottom of the LED matrix.

Project Time Line

A free Gantt chart creation tool was used to create a proposed timeline of the project and was updated to reflect our actual progression through the project. Key dates to the project are

shown in blue, namely highlighting the board send out dates, midterm design review, and final demonstration. The larger tasks of PCB design and embedded coding were parallelized and broken up into their relevant subsections. The tasks within the PCB design and embedded coding could be completed in a parallel fashion, with the game algorithm being developed while input and matrix code were developed. PCB design was parallelized but certain sections such as the power supply would be important to finalize first. The CAD design of the enclosure relied on taking dimensions from the PCB after it was assembled to get the real dimensions of the board and matrix required to properly house the component. The final tests of the embedded code relied on ordered parts to arrive, and sometimes relied on testing on the assembled PCB. As such, these tasks relied on the PCB to be completed as soon as possible. It was key to order parts as soon as possible and design the PCB to meet the send out dates as early as possible.

The Gantt chart from the proposal of the project is shown below in Figure 13 with an updated version shown in Figure 14. The largest difference between the timelines is that it took much longer than expected to develop the LED matrix code, and that revisions were necessary to our PCB design. The power supply was revised to take less surface area on the board, and a revised board with changes for the LED matrix and updated power supply was sent out for the final send out. Testing of the PCB was also drawn out longer in reality as various components became available to test with the embedded code as they arrived. The CAD of the enclosure was also moved to the end of the project to ensure that the assembled PCB dimensions could be considered when designing the housing. The embedded code for the LED matrix also took longer than expected as a shift in the design choice from using a driver chip to a direct connection required a fresh start in development. The scoreboard implementation also was put toward the end as the scoreboard drivers chosen were difficult to test without an assembled PCB.

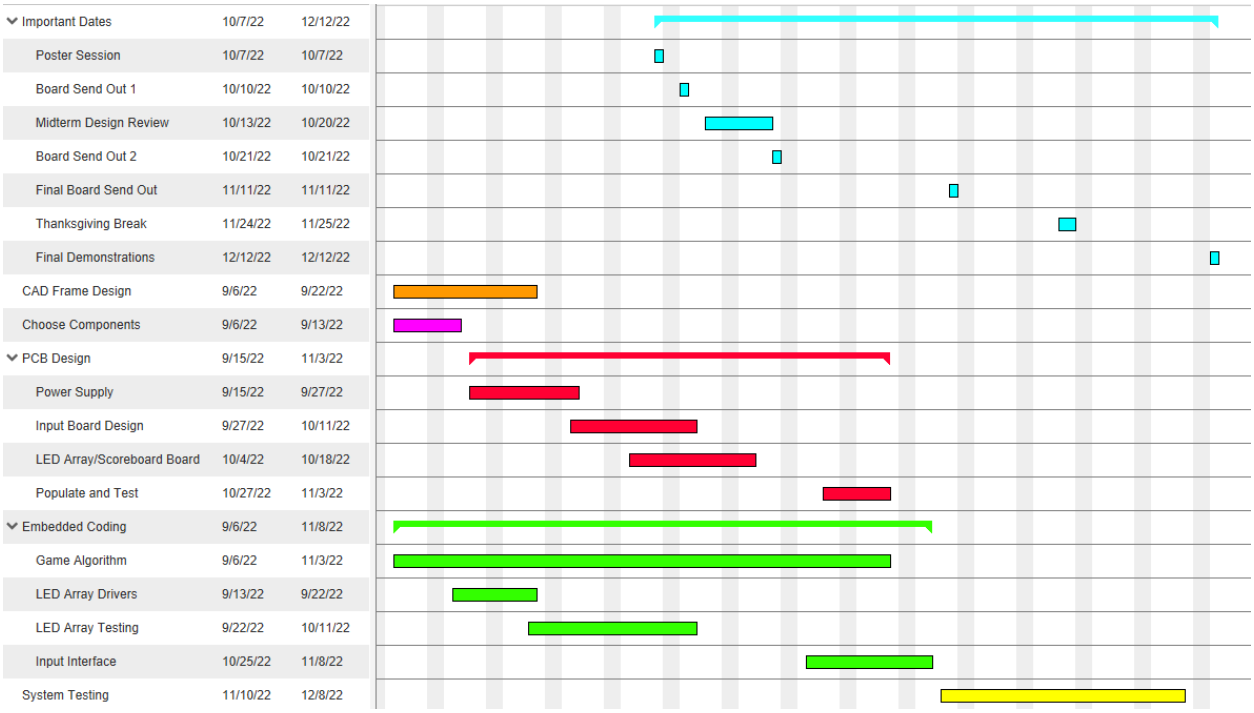

Figure 13. Proposed Gantt

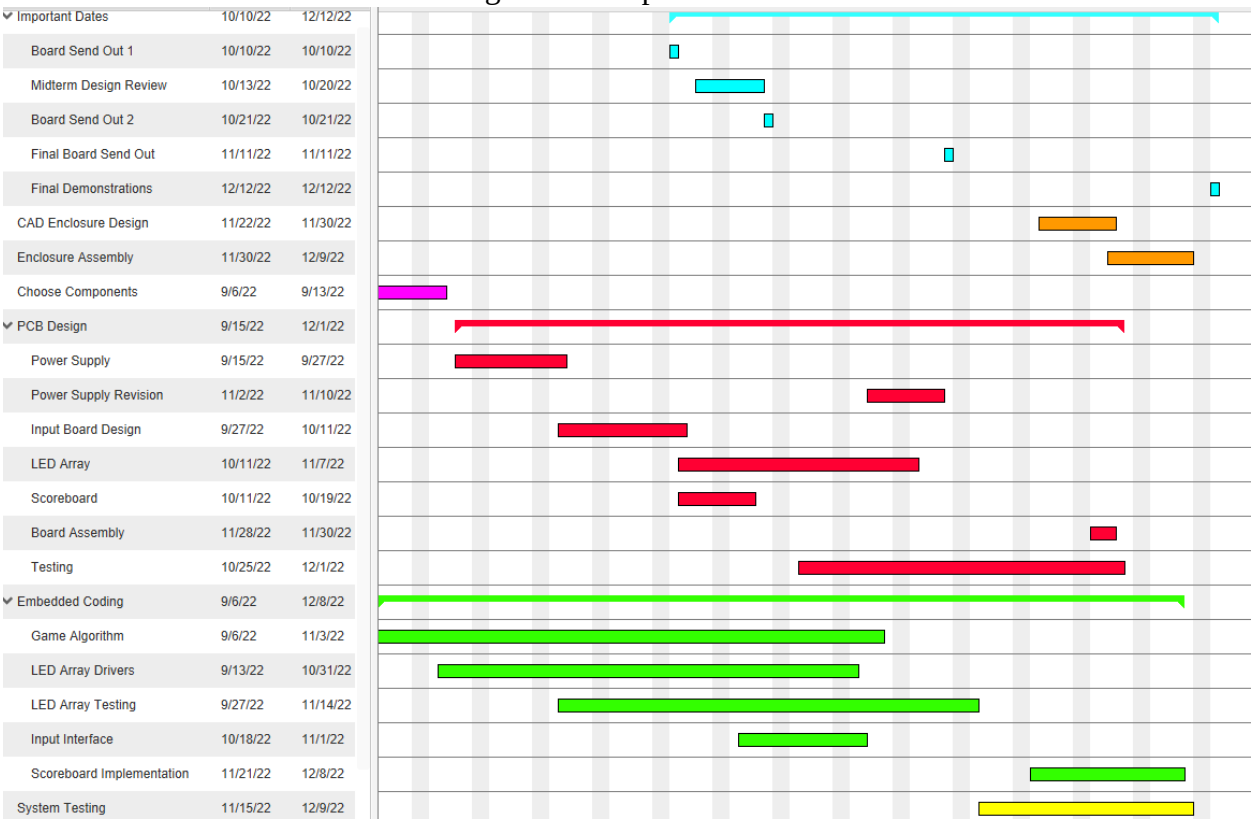

Figure 14. Finalized Gantt Chart

Boheng was the lead member of the group responsible for the dots and boxes game algorithm itself, as well as how that code accepts inputs and sends output to the LED matrix. He acted as a secondary group member for developing input code and writing updated LED matrix code.

Jacob was the primary group member responsible for designing the PCB and the necessary circuits power supply, input protection, and the scoreboard drivers. In doing so, he had to do the necessary research to select components for each subsystem. He assembled and tested the PCB to verify that the circuits on the PCB were working as intended. Beyond this, he helped with the design and assembly of the 3D printed enclosure.

James primarily worked on the user input code, utilizing the joystick, rotary encoder, and push button to change an input image in the game code. He used interrupt routines and implemented a queue for the inputs to be read by the game logic using C programming. James also worked on displaying the score and game messages such as error codes or establishing the winner.

Joseph was the primary group member responsible for the LED matrix drivers. After initial attempts to use a driver chip, the final code for the LED matrix was developed by Boheng with some optimizations being implemented by Joseph. Joseph also took the lead on designing

the 3D printed enclosure and getting the parts printed for the enclosure. Joseph also helped with PCB assembly and hardware testing.

Test Plan

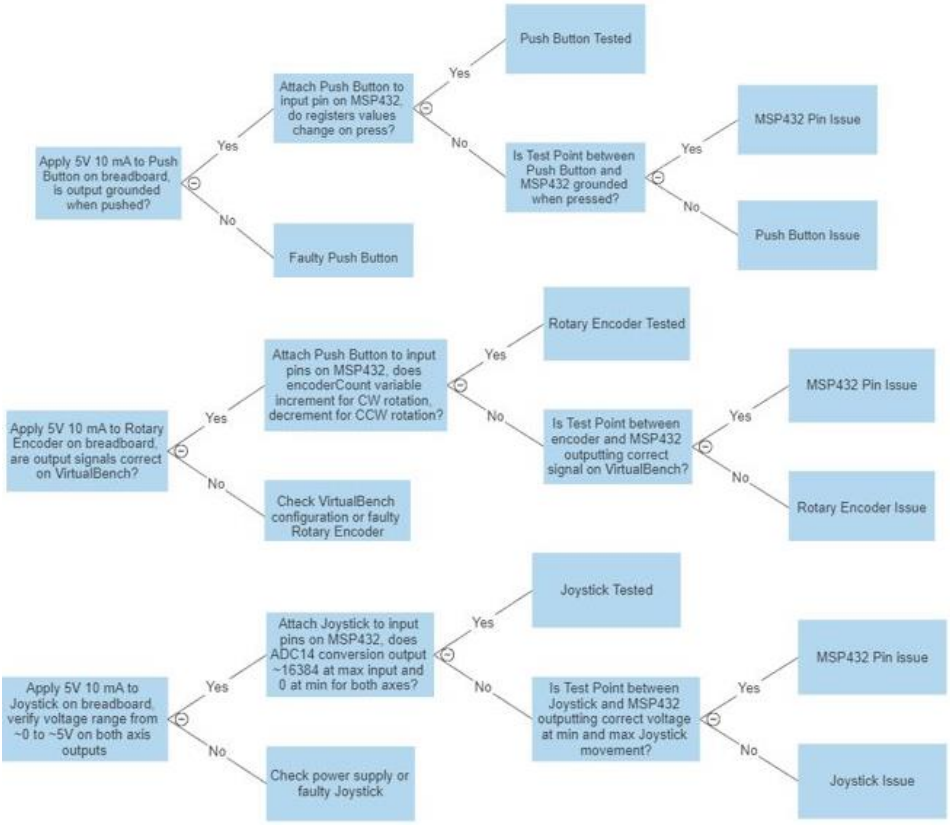

Figure 15. Input Test Plan

The input device testing plan is divided into two main sub modules: initial testing on a breadboard, and testing on the PCB with test pins. The test plan for the input devices was followed, and each was verified initially on a breadboard connected to the MSP432P401R pins. After the components were soldered onto the PCB, the input devices were verified to be working by following the test plan and utilizing the test points to verify proper input signals.

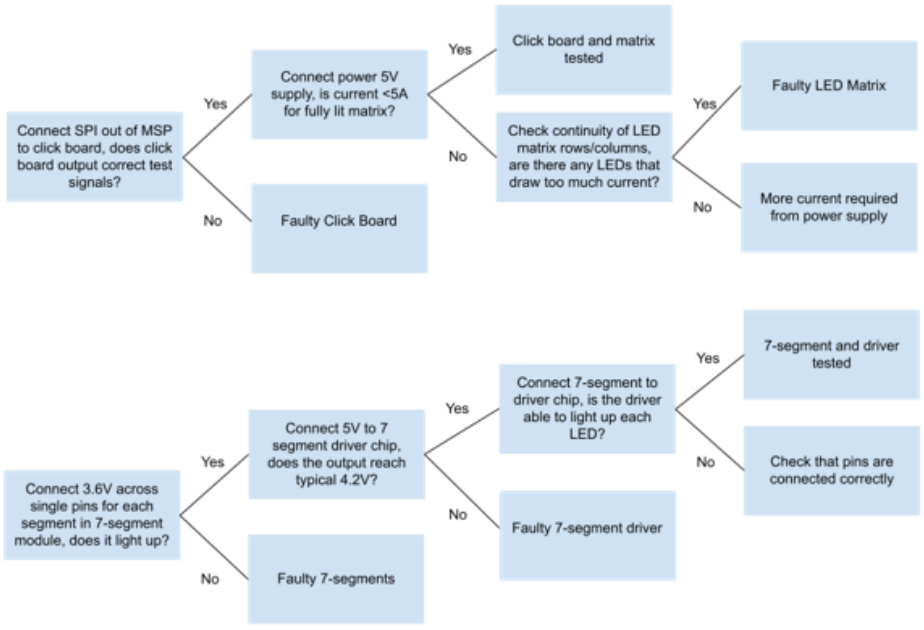

Figure 16. Output Test Plan

The output devices testing plan is also divided into two main sub modules: testing the LED matrix and testing the 7 segment scoreboard. The LED matrix 16 pin connection was tested and verified for a fully lit matrix of varying colors. After the matrix connections were soldered onto the PCB, the LED matrix was tested and verified again for a fully lit matrix of varying colors. The 7-segment display was unable to be tested with the full system due to software setbacks. Rather, the scoreboard feature was tested on the LED matrix.

A version of the game was constructed using the terminal output and keyboard inputs. A game was played on a 4x4 board to test its desired behavior. The output was simulated with a mock write pixel function where all game articles are 1 x 1 pixel. The terminal displays a mock LED board, player score, error codes, and relevant variables. The second part of the test consists of putting the terminal game with actual input components. This is to verify that the input will cause desired output behavior and allows us to isolate problems as they appear. Finally, the terminal output was replaced by the LED matrix to perform a full test.

Once the inputs and outputs were verified individually, the full system was tested by verifying the moves were processed correctly in the CCS registers debugger and on the LED Matrix. The game logic was tested by verifying valid moves and invalid moves. The scoreboard was tested by playing full games and verifying the score is concurrent with what is displayed on the LED matrix. The matrix is verified by cross-referencing the board representation on the CCS expression viewer.

Final Results

The success criteria defined in the initial proposal are defined below.

- Power source powers all components
- Users can accurately input controls
- Scoreboard accurately displays correct score
- Scoreboard accurately displays winner of game
- Invalid moves cannot be made
- Valid moves can be processed
- Input to output response time is shorter than 50 milliseconds
- Game can be restarted without having to be powered down

The device that was achieved adhered to and met these criteria. The power supply could power all components such as the input controls, microcontroller, and LED array. There was no strain on any of the power supply components as the device powered for 3 consecutive hours and no residual heat was being produced by the regulators or components in the circuitry. The game logic was successful in accurately displaying the input controls and processing valid moves made the user. Invalid moves were not processed by the game logic and instead error messages were displayed when an invalid move was attempted. Error 1 was displayed when a user tried to select a line that was already drawn, Error 2 was displayed when a user tried to select a line when a line was not yet displayed, and Error 3 was displayed when the user attempted to move the cursor out of the parameters of the game. The game could be restarted without being powered down if it was completed. According to our final game logic, a game was deemed complete when all boxes in the game had been claimed. All input controls would quit working except for the pushbutton which could be pressed to reset the game to the beginning state.

The scoreboard that was intended at the start of the design did not work as planned. The 7 segment displays that were included in the design with the purpose of displaying the score of each player and then displaying the winner upon completion of the game did not work correctly. However, after learning that it would take too long to debug the software for the 7 segment display drivers, a design change was made to display the score and winner on the LED array. This worked as intended and allowed for more creativity than the 7 segment displays would allow.

The input to output response time is another criterion that partially worked. The LED array was able to refresh its display in under 50 milliseconds as the LED array would have appeared to flicker if it refreshed in over this time. The pushbutton and rotary encoder's response time was also under the 50 milliseconds target. The joystick also met this target but did not appear so to the user. Since the joystick utilizes two potentiometers, one for each axis, in order to make it usable, a dead zone had to be established so the joystick was not overly sensitive. The user had to move the joystick in the desired direction over half the distance to the edge of the joystick's axis for a move to be processed. Once this had been done, the microcontroller was able to process the move in under 50 milliseconds, but since the overall movement of the joystick to the outer regions where a move would be processed took longer than 50 milliseconds, the input

to output response time appeared to take longer than the desired 50 milliseconds. Overall, the device was able to meet all the established criteria as described above.

Costs

The cost of one Dots and Boxes game as designed through this project cost $231.92. A breakdown of the costs associated per 1 unit and estimated cost per 10,000 units is shown in Table 3.

Table 3. Dots and Boxes Costs

	Cost per 1 Unit	Cost per 10,000 Units
MSP432P401R	$24.99	$24.99
32x32 RGB Matrix Panel	$34.95	$34.95
PCB	$33	$10.00
PCB Components	$138.98	$93.20
Total	$231.92	$163.14

The most expensive part of the game is the PCB and PCB components. These costs will be significantly reduced in higher quantities, with the PCB and component costs coming in at $171.98 per 1 unit and $103.20 per 10,000 units, which is a 40% reduction in cost. The unit costs are based on Advanced Circuits' special price for 2-layer boards for the 1-unit production, and their custom quote for the 10,000-unit production. PCB component costs were reached by adding the individual component costs from Digikey for both 1- and 5,000-10,000-unit price listings. A more detailed breakdown of costs can be found in Appendix Table 4. The MSP432P401R price is unfortunately high, and to get 10,000 units may be difficult as suppliers have discontinued stocking it. As such, for a 10,000-unit production, a replacement microcontroller would have to be sourced and made compatible with the PCB design. Similarly, the 32x32 RGB matrix panel is not stocked in high quantity and to get a good deal on many may be difficult. However, since a replacement panel with the exact same input configuration would be difficult to source, it is hard to reduce the cost associated with the LED matrix. Although the 1-unit production was assembled by hand, to produce 10,000 units, use of SMT and pick-and-place machinery would be necessary. For the enclosure, the 3D printing costs of material was $7.82, but using custom molds to create components of the enclosure will be necessary for 10,000 units.

Future Work

This project has multiple ways that it can be improved or expanded upon in the future. The first being an improvement on the durability of the 3D printed enclosure and input parts. As stated in a previous section, the 3D printer that was employed was not big enough to print the entire piece as one. Therefore, a larger printer could be used, or a different material could be used altogether. An improvement could've been made to make the input controls more intuitive to use as many people as possible in our final demonstration mistakenly pushed our rotary encoder and joystick to try and select a move.

This project could be expanded to include an option for an AI to play against instead of it being purely player v. player. This would be a lot more involved, but it would bring a new aspect to our game. It could also be expanded to include more games than just dots and boxes. Instead, the final device could be a general-purpose game console with multiple games loaded onto the device.

A difficulty that was not foreseen that took a considerable amount of time to work around was interfacing with the LED array. It is important to do the proper research on what type of component would best suit your needs before deciding on purchasing it. It is also very important to use devices that have a fair bit of documentation in the form of datasheets and information regarding how to use the component. As much as it is stressed by professors and other faculty, time management and communication are key. There are a lot of problems that could be avoided by a simple conversation and sticking to a strict plan about what is to be done and when is to be done by.

References

[1] G. Liangliang, L. Shuqin, M. Hui, G. Ming, H. Fengli and Y. Luying, "Dots and Boxes Interactive Interface Implementation and Algorithm Improvement," 2019 Chinese Control And Decision Conference (CCDC), 2019, pp. 6270-6275, doi: 10.1109/CCDC.2019.8833006.

[2] Y. Zhuang, S. Li, T. V. Peters and C. Zhang, "Improving Monte-Carlo tree search for dots-and-boxes with a novel board representation and artificial neural networks," 2015 IEEE Conference on Computational Intelligence and Games (CIG), 2015, pp. 314-321, doi: 10.1109/CIG.2015.7317912.

[3] L. Zhang, Y. Zhang, P. Liu and L. Guo, "The research to construct dots and boxes battle platform in computer game," The 27th Chinese Control and Decision Conference (2015 CCDC), 2015, pp. 3749-3753, doi: 10.1109/CCDC.2015.7162578.

[4] "Dots and boxes," *Gametable.org*. [Online]. Available: https://gametable.org/games/dots-and-boxes/. [Accessed: 12-Sep-2022].

[5] "CCSTUDIO," *CCSTUDIO IDE, configuration, compiler or debugger | TI.com*. [Online]. Available: https://www.ti.com/tool/CCSTUDIO. [Accessed: 12-Sep-2022].

[6] Microsoft, "Visual studio code - code editing. redefined," *RSS*, 03-Nov-2021. [Online]. Available: https://code.visualstudio.com/. [Accessed: 12-Sep-2022].

[7] "Where the world builds software," *GitHub*. [Online]. Available: https://github.com/. [Accessed: 12-Sep-2022].

[8] "KiCad Eda," *Schematic Capture & PCB Design Software*. [Online]. Available: https://www.kicad.org/. [Accessed: 12-Sep-2022].

[9] "Inventor software," *Autodesk*, 25-Aug-2022. [Online]. Available: https://www.autodesk.com/products/inventor/ [Accessed: 12-Sep-2022].

[10] "From mind to design in minutes," *Tinkercad*. [Online]. Available: https://www.tinkercad.com/. [Accessed: 12-Sep-2022].

[11] "DigiKey Electronics Home," *Digi-Key Electronics*. [Online]. Available: https://www.digikey.com/. [Accessed: 12-Sep-2022].

[12] M. Shuaib, A. Haleem, S. Kumar, and M. Javaid, "Impact of 3D printing on the environment: A literature-based study," *Sustainable Operations and Computers*, 21-Apr-2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666412721000131. [Accessed: 11-Sep-2022].

[13] B. Debnath, P. Roychowdhury, and R. Kundu, "Electronic Components (EC) reuse and recycling – a new approach towards WEEE management," *Procedia Environmental Sciences*, 04-Aug-2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1878029616301499. [Accessed: 11-Sep-2022].

[14] "United States Department of Labor," *1910.305 - Wiring methods, components, and equipment for general use. | Occupational Safety and Health Administration*. [Online]. Available: https://www.osha.gov/laws-regs/regulations/standardnumber/1910/1910.305. [Accessed: 26-Sep-2022].

[15] IPC-2221 Task Group (D-31b) of the Rigid Printed Board Committee (D-30) of IPC, "IPC-2221A," *Generic Standard on Printed Board Design*. [Online]. Available: https://www.ipc.org/TOC/IPC-2221A.pdf. [Accessed: 11-Sep-2022].

[16] Webmaster, "Embedded C coding standard," *Barr Group Software Experts*, 26-May-2016. [Online]. Available: https://barrgroup.com/embedded-systems/books/embedded-c-coding-standard. [Accessed: 11-Sep-2022].

[17] "United States Department of Labor," *Description for 7999: Amusement and Recreation Services, Not Elsewhere Classified | Occupational Safety and Health Administration*. [Online]. Available: https://www.osha.gov/sic-manual/7999. [Accessed: 11-Sep-2022].

[18] Texas Instruments Incorporated, "Texas Instruments MSP432 Driver Library," 2016. [Online]. Available: https://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/MSP432_Driver_Library/3_21_00_05/exports/driverlib/msp432_driverlib_3_21_00_05/doc/MSP432P4xx/html/driverlib_html/index.html. [Accessed: 13-Dec-2022].

[19] J. Strand, 1972.

[20] R. Fanti, 1984.

[21] P. Blanchard, 2012.

[22] L. Darling-Hammond, “Unequal opportunity: Race and education,” *Brookings*, 28-Jul-2016. [Online]. Available: https://www.brookings.edu/articles/unequal-opportunity-race-and-education/. [Accessed: 13-Dec-2022].

[23] “PCB Tolerances | Advanced Circuits.” https://www.4pcb.com/pcb-design-specifications/ (accessed Dec. 13, 2022).

[24] “WEBENCH® Power Designer | Overview | Design Resources | TI.com.” https://www.ti.com/design-resources/design-tools-simulation/webench-power-designer.html?utm_source=google&utm_medium=cpc&utm_campaign=app-null-null-58700006320820903_webenchhtml5-cpc-pp-google-wwe&utm_content=TI_Webench&ds_k=ti+webench&DCM=yes&&utm_source=google&utm_medium=cpc&utm_campaign=&utm_content=&ds_k=ti%20webench&DCM=yes&gclid=Cj0KCQiA4uCcBhDdARIsAH5jyUmwK--HVUOIdlPsUyfLoBDlL5b9A0MJjbN2kbMf3SJ7Xzn3bz5tp7QaAs7sEALw_wcB&gclsrc=aw.ds (accessed Dec. 13, 2022).

[25] “Download Fusion 360 for Free | Free Trial | Autodesk.” https://www.autodesk.com/products/fusion-360/free-trial (accessed Dec. 13, 2022).

[26] “SDI65-12-UDC-P5R CUI Inc. | Power Supplies - External/Internal (Off-Board) | DigiKey.” https://www.digikey.com/en/products/detail/cui-inc/SDI65-12-UDC-P5R/5419181?s=N4IgTCBcDaIIwAYwFoDMA2dBWZA5AIiALoC%2BQA (accessed Dec. 13, 2022).

[27] “PC721A Switchcraft Inc. | Connectors, Interconnects | DigiKey.” https://www.digikey.com/en/products/detail/switchcraft-inc/PC721A/8571718?s=N4IgTCBcDaIMoGEDMBWAjABgLQDkAiIAugL5A (accessed Dec. 13, 2022).

[28] “LM22677TJE-5.0/NOPB Texas Instruments | Integrated Circuits (ICs) | DigiKey.” https://www.digikey.com/en/products/detail/texas-instruments/LM22677TJE-5-0-NOPB/1950776?s=N4IgTCBcDaIDIFkxgGwHY0BUBSBRAtAKwB0ADAPQByA8gAoBCAwpvpQCIgC6AvkA (accessed Dec. 13, 2022).

[29] “TPS79633DCQ Texas Instruments | Integrated Circuits (ICs) | DigiKey.” https://www.digikey.com/en/products/detail/texas-instruments/TPS79633DCQ/1908503 (accessed Dec. 13, 2022).

[30] “COM-09032 SparkFun Electronics | Switches | DigiKey.” https://www.digikey.com/en/products/detail/sparkfun-electronics/COM-09032/6823623 (accessed Dec. 13, 2022).

[31] “EN11-HNM1AF15 TT Electronics/BI | Sensors, Transducers | DigiKey.” https://www.digikey.com/en/products/detail/tt-electronics-bi/EN11-HNM1AF15/2408763 (accessed Dec. 13, 2022).

[32] "KS11R21CQD C&K | Switches | DigiKey." https://www.digikey.com/en/products/detail/c-k/KS11R21CQD/332668 (accessed Dec. 13, 2022).

[33] "LDD-C516RI Lumex Opto/Components Inc. | Optoelectronics | DigiKey." https://www.digikey.com/en/products/detail/lumex-opto-components-inc/LDD-C516RI/252619?s=N4IgTCBcDaIGwHYC0BGALHMSByAREAugL5A (accessed Dec. 13, 2022).

[34] "MAX6955AAX+ Analog Devices Inc./Maxim Integrated | Integrated Circuits (ICs) | DigiKey." https://www.digikey.com/en/products/detail/analog-devices-inc-maxim-integrated/MAX6955AAX/948165?s=N4IgTCBcDaILIEEAaA2AnAVgw5BqAtAHIAiIAugL5A (accessed Dec. 13, 2022).

[35] Texas Instruments, "MSP-EXP432P401R: Out of Box," *MSP-EXP432P401R | Out of Box - MSP-EXP432P401R - Welcome 1.20.00.45 documentation*, 2016. [Online]. Available: https://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/SIMPLELINK_MSP432_SDK/1.20.00.45/exports/docs/simplelink_mcu_sdk/project0/project0/docs/MSP-EXP432P401R.html. [Accessed: 13-Dec-2022].

[36] "0676430910 Molex | Connectors, Interconnects | DigiKey." https://www.digikey.com/en/products/detail/molex/0676430910/917619?s=N4IgTCBcDaIOoFkAsBOFBmAtAOQCIgF0BfIA (accessed Dec. 13, 2022).

[37] "Ender-3 pro 3D printer," *creality*, 2022. [Online]. Available: https://www.creality.com/products/ender-3-pro-3d-printer. [Accessed: 13-Dec-2022].

[38] "SMBJ6.0A-13-F Diodes Incorporated | Circuit Protection | DigiKey." https://www.digikey.com/en/products/detail/diodes-incorporated/SMBJ6-0A-13-F/814625?s=N4IgTCBcDaIMoFkBCApAbAOgAwEEC0AYgCICSAwgCp4ByRIAugL5A (accessed Dec. 13, 2022).

[39] "VirtualBench All-in-One Instrument - NI." https://www.ni.com/en-us/shop/hardware/products/virtualbench-all-in-one-instrument.html (accessed Dec. 13, 2022).

[40] "Ultimaker Cura: Powerful, easy-to-use 3D printing software." https://ultimaker.com/software/ultimaker-cura (accessed Dec. 13, 2022).

Appendix

In this section you should include helpful information that does not fit into the above categories but will be helpful in understanding and assessing your work.

Table 4. Total Costs of Components

Description	Quanity	Price	Price/10,000	Single Total	Per 10k Total
32x32 RGB LED Matrix Panel	1	$34.950	$34.950	$34.950	$34.950
COM-09032 Joystick	1	$4.500	$4.500	$4.500	$4.500
EN11 Rotary Encoder	1	$1.920	$1.102	$1.920	$1.102
KS11 Push Button	1	$2.57	$1.68	$2.570	$1.677
MAX6955 7-Segment Driver	2	$22.43	$17.43	$44.860	$34.860
LDD 7-Segment	2	$3.42	$1.44	$6.840	$2.880
5V 5A Regulator	1	$7.33	$4.36	$7.330	$4.356
3.3V 1A Regulator	1	$4.08	$2.18	$4.080	$2.178
12V 60W Power Supply	1	$31.40	$25.76	$31.400	$25.760
PC721A Barrel Jack	1	$4.66	$2.91	$4.660	$2.913
0603 61.9k Resistor	1	$0.10	$0.00	$0.002	$0.002
UMK107 1uF Capacitor	1	$0.26	$0.04	$0.260	$0.044
SB10150 Schottky Diode	1	$0.52	$0.17	$0.520	$0.173
16SVP 270uF Capacitor	1	$2.17	$0.77	$2.170	$0.770
GRM32 47uF Capacitor	1	$0.58	$0.14	$0.580	$0.139
08055C 10000pF Capacitor	1	$0.10	$0.01	$0.100	$0.012
3.3uH Inductor	1	$2.04	$1.06	$2.040	$1.065
GCJ21 2.2uF Capacitor	1	$0.41	$0.09	$0.410	$0.092
LMK107 1uF Capacitor	1	$0.10	$0.01	$0.100	$0.013
0201ZC 10000pF	1	$0.10	$0.01	$0.009	$0.009
RC0402 56k Resistor	2	$0.10	$0.00	$0.200	$0.003
0603 0.1uF Capacitor	2	$0.11	$0.02	$0.220	$0.032
0603 4.7k Resistor	2	$0.10	$0.00	$0.200	$0.005
0603 22pF Capacitor	2	$0.10	$0.01	$0.200	$0.020
Red Test Points	3	$0.42	$0.17	$1.260	$0.511
Black Test Points	7	$0.42	$0.17	$2.940	$1.193
White Test Points	9	$0.42	$0.17	$3.780	$1.534
SMBJ TVS Diode	3	$0.45	$0.09	$1.350	$0.281
RNMF 4.7k Resistor	3	$0.10	$0.02	$0.300	$0.051
K103 10000pF Capacitor	3	$0.22	$0.04	$0.660	$0.121
RNF181 1k Resistor	3	$0.10	$0.01	$0.300	$0.032
USB Female Connector	1	$1.75	$0.91	$1.750	$0.909
Matrix Connector	1	$8.19	$4.29	$8.190	$4.290
Matrix Spade Power Connector	2	$0.38	$0.16	$0.760	$0.327
MSP432 Connector	2	$1.26	$0.67	$2.520	$1.349
MSP432P401R Launchpad	1	$11.99	$11.99	$11.990	$11.990
				$185.921	$140.142