4th and 5th Grade STEAM Educational Trivia Game

ECE 3011 Junior Design Project

Trivia Trek Game

Sec A_T01

Prof. Brothers

Rachel Ha CmpE rachelha54@gatech.edu

Michael Uveges EE muveges3@gatech.edu

Shayan Noorani EE shayan@gatech.edu

Wilson Williams CmpE wwilliams79@gatech.edu

Submitted

2024 March 31

Table of Contents

Executive Summary

Learning happens best when you're having fun. To catalyze the learning process in children, we've designed a trivia board game in which teachers can ask questions specifically for their students to learn and review. Regarding the design of Trivia Trek, the game board vaguely resembles a "Chutes & Ladders" style pathway with tiles of various colors. In addition, we will feature a device that displays the questions and answer options, contains push buttons for the various user interactions, and controls a motor that determines how many spaces the player moves. In short, our device serves to eliminate the hassle and chaos often presented during board games played amongst young children. However, even with the complexity that comes with centralizing all the components of a board game, the total cost will be under $100.

A prototype of this game has been developed. An initial software framework has been completed to form a complex software project that encompasses the interactions between all the electrical components, the game, and the players. A PCB houses all the electrical connections between the I/O devices and MicroBit. A box safely houses and protects the electrical components from the targeted young audience. Finally, a game board completes the idea of the game and gives the players a way to visualize the state of the game and facilitate their learning.

Nomenclature

STEAM: Science, Technology, Engineering, Art, Mathematics

4th and 5th Grade STEAM Educational Trivia Game

1. Introduction

To make learning more enjoyable, our team has designed a trivia board game. This board game will serve as a teacher's aide as they can customize the content of the questions the students will be asked. Thus, Team One is requesting a MicroBit processor and $100 in funding to develop Trivia Trek, a trivia board game.

1.1 Motivation

The design problem is to strategize and build a STEAM educational kit for K-5th grade students. The product should be aesthetically pleasing to the user and help them learn about math, science, and fine arts topics.

1.2. Objective

The intended use of Trivia Trek is to create a competitive game between elementary aged students in which they will compete by answering questions relating to specific and relevant Georgia Milestone standards. Pictured in Figure 1 is the preliminary design of the product. The game can be played by one or up to four students. The players will set the electronic box in the designated space on the board. Players will be given questions relating to different topics, which are determined by which section of the game board the player is in. The players will answer the questions and, if answered correctly, will move along the board. Once a player reaches the end, the game will end.

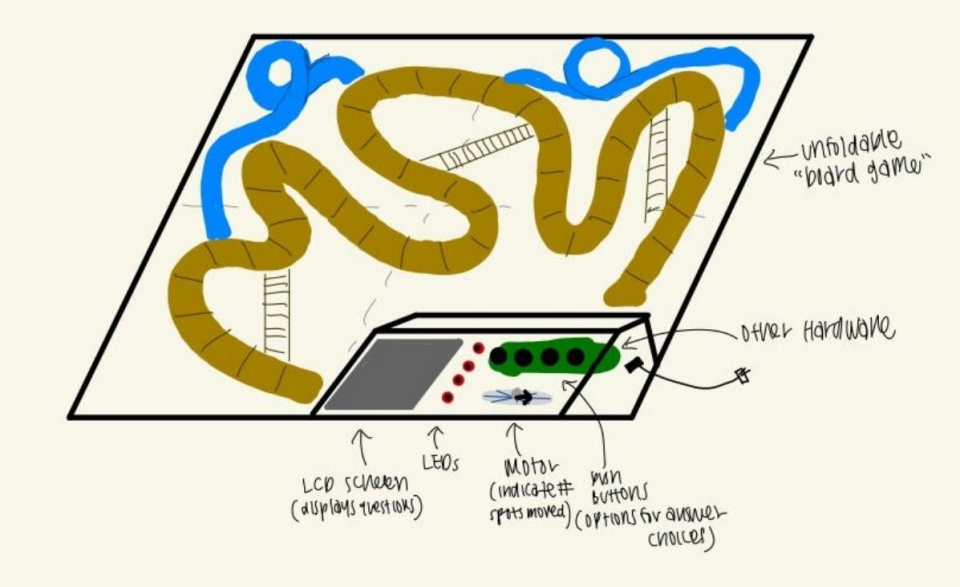

Figure 1. Preliminary sketch of board game and electronic box.

There will be several points of interaction between the system and the users. One is the game board in which players will move physical pieces across the board. A player will select the difficulty of the game with a proximity sensor. The system will also take input from the players via pushbuttons. Information, including the questions and game status, will be shared by the system to the players with an LCD display and LEDs. A motor will also spin a spinner to inform the players how many positions to move on the board.

The stakeholders of this system are 4th and 5th grade students, their teachers, and their parents. The students will benefit by learning about relevant standards in a fun, competitive environment that a teacher or parent may struggle to provide. Teachers and parents can communicate relevant knowledge to younger students with this product in a way that encourages students to learn. To accomplish this, the system must encourage a competitive environment while effectively teaching and testing students.

1.3 Background

Teachers and parents are constantly searching for new and engaging ways to teach young students certain topics. Students learn in various ways, such as lectures, reading, visuals, and games. This project teaches students through a game. Creating a competitive environment while maintaining an effective learning environment is difficult, but it is something this product seeks out to do. This document describes the project's design, requirements, goals, and much more.

2. Project Description, Customer Requirements, and Goals

When presented with the task of creating an educational toolkit for children, our team greatly considered what would be optimal for teachers, students, and parents. Minimizing the work for the players, our centralized game system asks specific, curated questions, contains buttons for answer options, accounts for question difficulty, keeps track of the time allotted per question, and determines how many spaces players will move. This design not only eliminates the risk of losing game pieces, not knowing what the next step is, and arguing between the students, but also allows for teachers and parents to ensure that they are being asked questions supplementary to their class or home content. Pertaining to game specifics, our team has detailed specific engineering requirements. Firstly, using a proximity sensor, players will be able to choose between Easy (<10 inches) and Hard (>10 inches) to determine the difficulty of the game. The questions are predetermined by the overseer and require 3 topics and a minimum of 3 questions per topic. However, more questions per category are encouraged to reduce the likeliness of repeat questions. Once players are prompted the question, there will be four push buttons that act as control, scroll, and select, to gather the player's response. Additionally, our device will recognize the number of players and maintain their locations throughout the board. This also allows the device to tell the players how many spaces they should move and who the winner is. Additionally, a constraint that may be presented during the playing of the game is running out of questions for a certain category if that category is landed on consecutively. However, to ensure that each question is unique to the previous, we have instilled the requirement of having at least three

questions per category. But again, more questions would reduce the likelihood of repeats amongst the players.

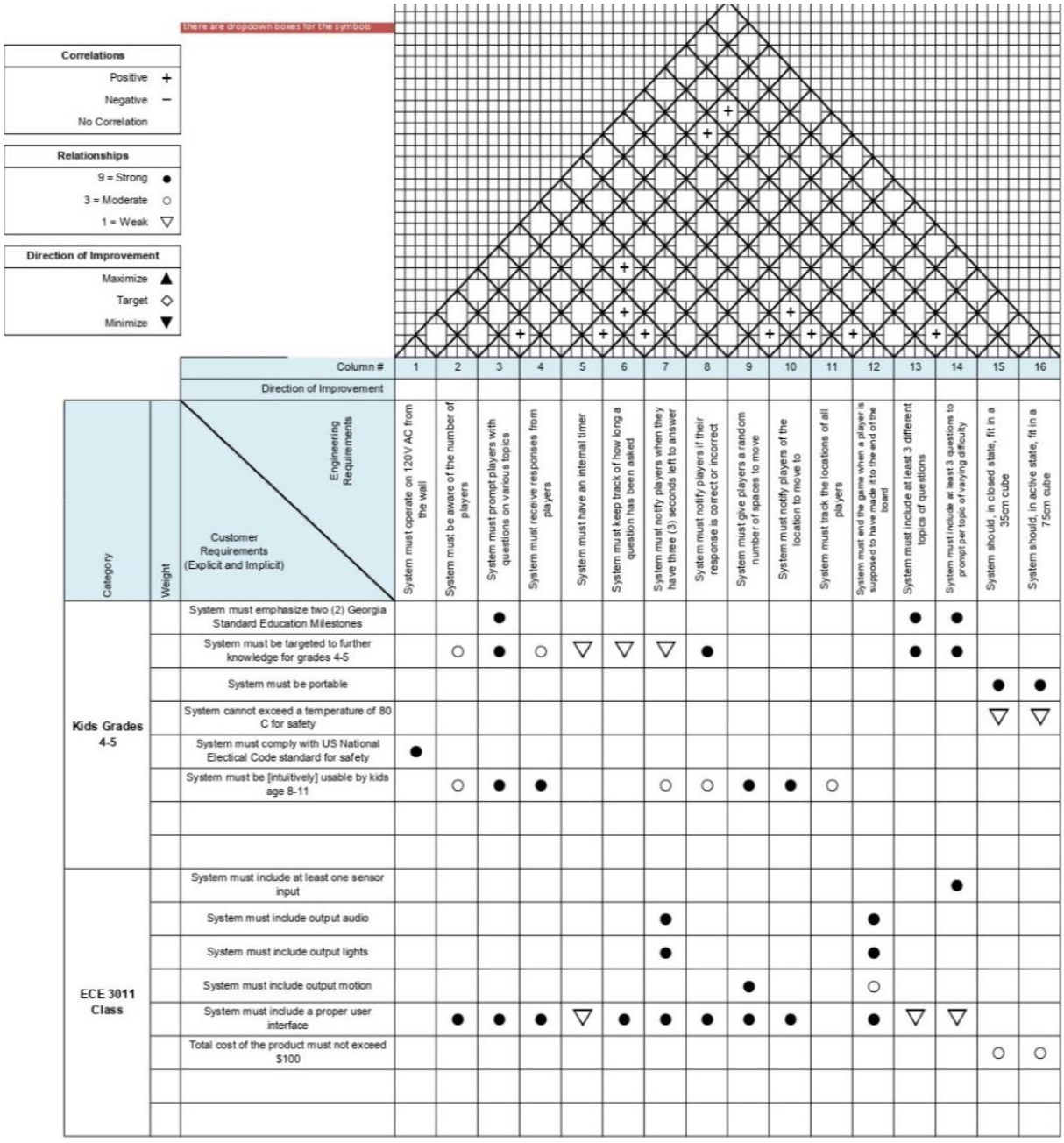

Figure 2. House of Quality (QFD) chart translating customer needs and technical requirements.

3. Technical Specifications & Verification

Description	Updated Value	Measured Value
Weight	< 10 kg	4.0 kg
Power-down size	< 35cm x 35cm x 35cm	24cm x 24cm x 14cm
PCB Size	< 15cm x 15cm	15cm x 15cm
Current from 120 VAC outlet	< 5A	1.2A expected
Maximum power consumption	< 60W	2.5 W
Number of question categories	>= 2 categories	3 categories
Utilizes LEDs	Yes	Yes
Output motion	Yes	Yes
Utilizes motion sensor	Yes	Yes
Maximum case temperature	< 80 degrees C	< 80 degrees C

4. Design Approach and Details

The STEAM trivia game system is broken down into four sub-systems: *Power Distribution*, *Graphic Design*, *User Interface*, and *Game Master*. *Power Distribution* will receive a 120V AC power supply and convert it into 5V and 3.3V DC power sources for the MicroBit and other components. *Graphic Design* is responsible for the physical game board, prompting questions to the user, and rotating a spinner. The *User Interface* sub-system receives user input with I/O devices and will have visual indications of game status. *Game Master* contains a bank of trivia like questions to ask the user and

maintains the status of the game and player positions. These sub-systems and their relations with each other are depicted in Figure 3 below.

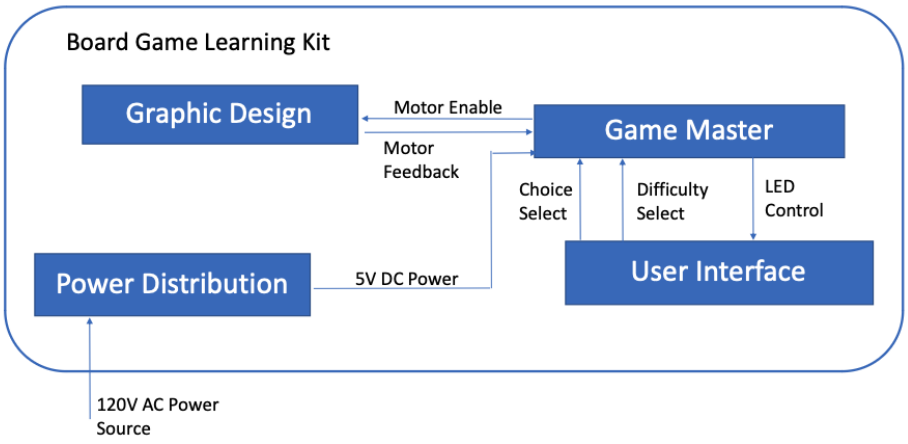

Figure 3. System block diagram that notes all sub-systems and interfaces between sub-systems.

As seen in Figure 3 above, there are seven interfaces between the different sub-systems. The interface labeled *120V AC Power Source* will be a 120V AC power supply coming from a standard U.S. wall outlet. *5V DC Power* is outputted from the Power Distribution sub-system and will supply the MicroBit with necessary power. *Choice Select* is an interface that transmits data from user input when players answer the questions in the game. *Difficulty Select* is an interface that transmits the difficulty of the game, which is selected by the user. The *LED Control* interface from *Game Master* to *User Interface* will carry data to be shown visually on the game board through the LEDs. The interface *Motor Enable* will control the speed and direction of a motor to move a spinner. The Motor Feedback interface will communicate the motor's position with respect to the board.

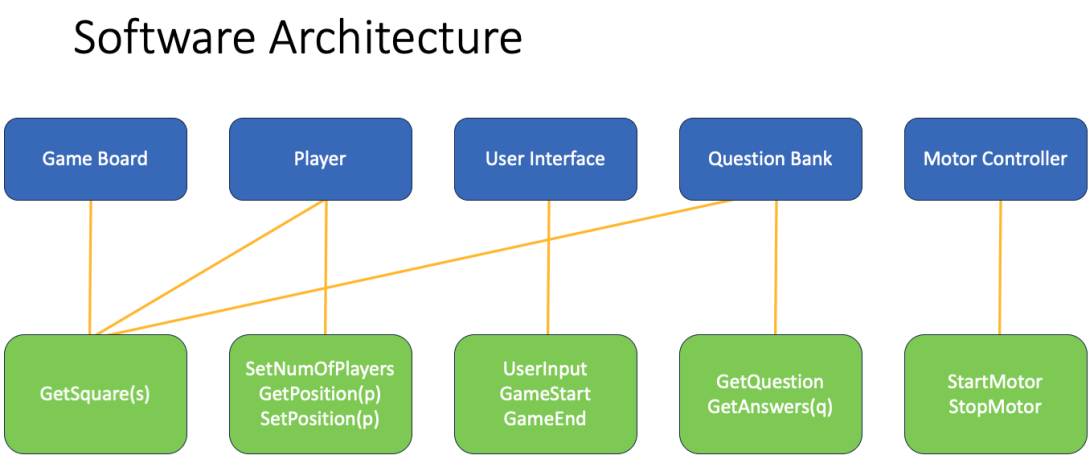

Figure 4. High level software architecture. Blue boxes represent larger portions of the software code that are based off the sub-systems shown in the system block diagram. Green boxes include the main functions that each blue box has access to.

The software architecture of this system contains five main components: game board, player, user interface, question bank, and motor controller. The physical game board will have squares with paths in between them as short cuts and obstacles. Each tile will need a property that informs if the player shall remain at the same spot, advance, or move backward. Each player will be represented by a Player class. Each player object will have a current position, and the software can retrieve and update this position. The question bank will be a list of questions associated with a topic and difficulty level. Each question will also be mapped to a list of answers with one of them being correct. User input will be a key component of this system so data via sensors and pushbuttons will be collected and sending data to users visually through LEDs. Finally, there is a motor that will move a spinner on the board, so the software needs access to control the motor to spin it to a certain number on the wheel.

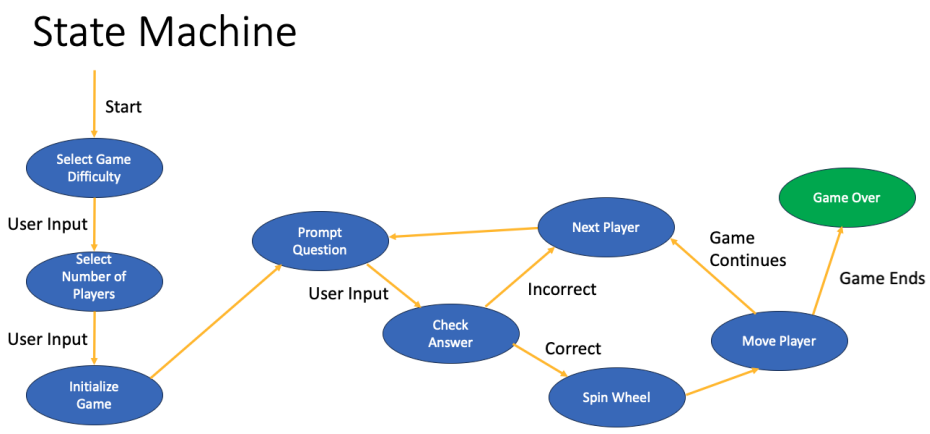

Figure 5. State machine describing the flow of the game. Each bubble is a stage in the software that will perform certain functions to operate the game. Descriptions along the arrows are actions that need to be taken to move from one state to the state the arrow is pointing to.

The game flow, as shown in Figure 5, begins with the users selecting certain parameters to start the game. This data is needed to let the system know which question sets to pull from and how many players to rotate between. After the game has started, questions will be prompted until the game ends. Once a question appears, user input is needed to change state. This user input is compared with the expected input and determines how to proceed. If the inputs match, the wheel is spun, and the player shall move along the board according to the wheel. This process continues, keeping track of player status, until a player has reached the end of the board.

4.1 Design Concept Ideation, Constraints, Alternatives, and Tradeoffs

Graphic Design Subsystem

The Graphic Design Subsystem's main function was to design and bring to life a visually appealing, functional, and easy-to-navigate gameboard. This role required the testing of different design softwares such as Canva, Adobe Draw, and Procreate, and deciding which would be best suited for the project.

While the original mockups were created using Canva (Figure 6), the final design was created using Procreate (Figure 7).

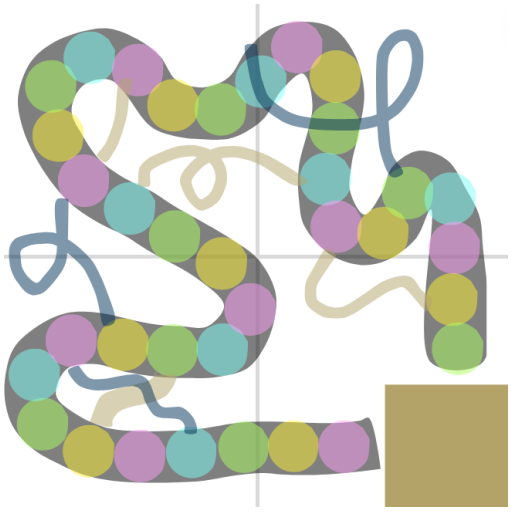

Figure 6. The final draft of the game board design created in Canva.

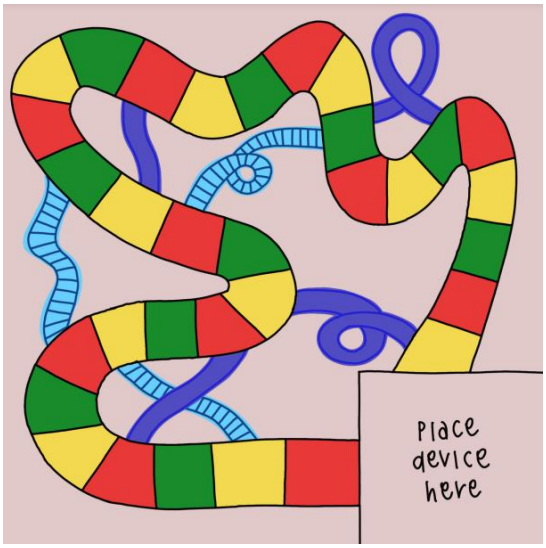

Figure 7. Graphic displaying the layout of the physical game board.

When designing the board, children's attention span, visual appeal, and route intricacy were considered. In doing so, the decision for 3 categories, 3 ladders, and 4 chutes was determined with the intention that the game would not last too long, eliminating the loss of children's attention, while also ensuring that there

was ample “chaos” pertaining to the chutes and ladders requiring moving back and forth between certain spaces.

For the creation of the physical game board, there were two main considerations: First, determining what material it would be made of and second, determining which adhesive and sealants to use. Originally, we had planned to utilize wood as the base of the board. However, after creating a mockup of the board using cardboard (Figure 8), cardboard seemed like a much better material to work for the actual product as it was easier to handle, took much less time, and most importantly, it mimicked an actual game board. While fabrication of wood would have been impressive, it loses that authentic, “children’s game-like touch” that most appeals to younger children. Additionally, for adhering the design to the board, Mod Podge was used. On the first rendition, wood glue, the only available adhesive at the time, made the corners of the board curl in and resulted in a bumpy surface that was unpleasing to the eye and touch. On the second rendition, a well-known adhesive and sealant, Mod Podge, was used which created a better appealing board.

Figure 8. First physical mockup of game board on cardboard.

Game Master Subsystem

The main function of the Game Master subsystem that a user will see is the display of trivia questions and answers. The Game Master subsystem is software heavy, with roles pertaining to the game state, keeping

track of player status on the game board, retrieving trivia questions and answers, and generating random spaces for players to move on the board. Along with these roles that are handled by the MicroBit, the subsystem will display trivia questions and answer choices to a user via an LCD display.

A different concept for conveying questions and answers to a user could be through a speaker. This speaker could read the question aloud to the user and follow it up with answer choices. By using the LCD display instead of the speaker, additional hardware and pins on the MicroBit had to be used since the design is already utilizing the speaker in another subsystem. Regarding usability, the LCD display made more sense in the design. If the user wants to cycle through answer choices, it will take more time with the speaker rather than reading off a display. The design is targeted to allow users to control the flow of the game rather than the system's components.

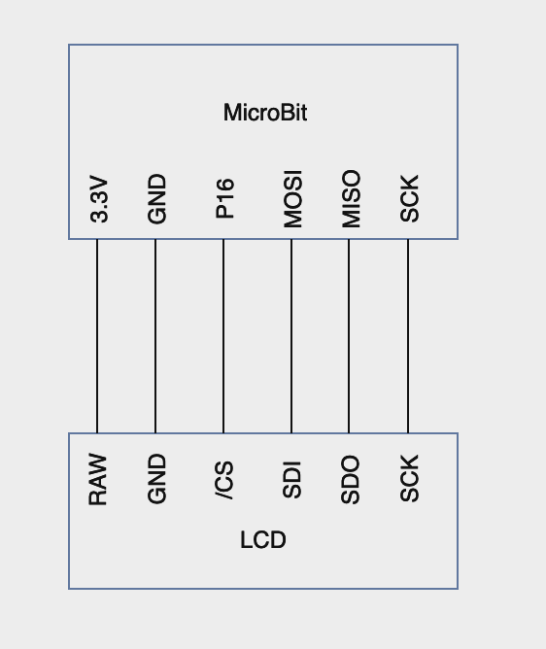

Figure 9. Schematic showing the electrical connections between the MicroBit and LCD display.

As seen in the figure above with the use of the SDI, SDO, /CS, and SCK pins, the LCD display uses the SPI communication protocol. When referring to computer aspects of the project, communication between

the display and the MicroBit controller is extremely fast, using less processing power and time than other protocols.

Power Distribution Subsystem

One of the main concerns of the Power Distribution system was would just powering the micro bit enough power to power all components at the same time such as the neopixel LED's as well as things like the servos and LCD's as when breadboarding, each component worked well off just power from the microbit when testing each component individually. However, to make sure that if that was not the case, the design of the PCB and connections were made so that with the use of jumpers one can switch if the power is being provided by a digital out of the microbit or from the power inputs of the board itself.

The initial plan was to use a 12V barrel jack and buck it down to ensure that our current draw was much lower to allow us to utilize smaller wire gauges without much thought. In addition, we were also going to use a PCB mounted buck converter. However, due to cost as well as availability of products in the ECE shop, we decided it was better to change our design to use a 5V barrel DC power supply and then use a buck that is detached from the PCB to give us a 3V3 rail and a 5V rail.

One of the harder peripherals to implement was the servo motor, as that and the neo pixel were the only ones on the board that ran off 5 volts. However, servo motor would also require the microbit to also output 5V for control signals and feedback signals would also be at 5V logic, thus, to protect the microbit voltage dividers were utilized to protect the input pins into the microbit and an N-Channel MOSFET driving circuit was used to amplify the PWM output of the microbit to be at a 5V logic level.

Fusing and reverse polarity protection was thought of at first when creating the PCB but was later forgotten about when the PCB schematic was being made. To continue to address these concerns, an inline

fuse was thought to be sufficient for fusing as well as a label for which type of barrel jack to use to know which polarity the barrel jack should be to decrease the possibility of reverse polarity to the entire electrical system. The problem with using this solution is that it may not be as clean as it could have been if all of that was on the PCB, however, if there were to be a mass production version of this product it would be a quick addition to the PCB without much consequence.

User Interface Subsystem

Regarding the user interface of the game, we needed to design a system that was intuitive for fourth and fifth grade students to play our game. To meet the customer's requirements, we needed to implement at least one sensor, lights, motion, and audio; this was all considered for the idea of the game interface. Our original thoughts were to use a time-of-flight sensor to allow the players to select the difficulty of the game, positioning their hand in the difficulty range they wish to play. Buttons would allow the players to send a response back to the Game Master. LED lights would be implemented as a timer, flashing when the player has 3 seconds left to answer.

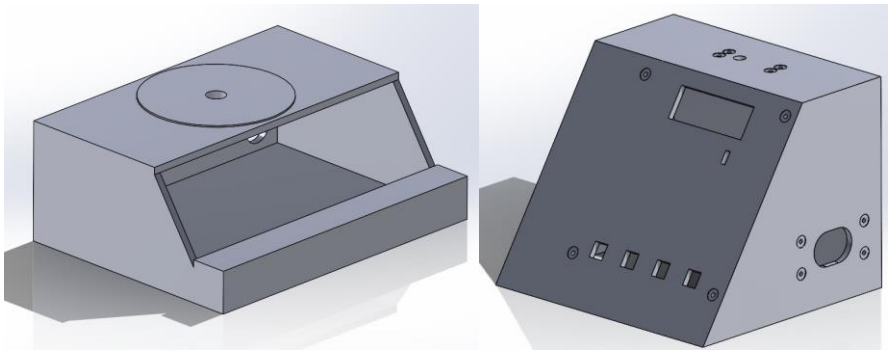

Figure 10. Original idea mockup (left) and the updated rendition (right).

Very few adjustments have been made in terms of the ideas and functionality of the user interface, but the design has been changed. The design of the box was originally going to be rather freeform, but by deciding to fix the screen and buttons to the PCB the design of the box needed to change to account for this. Originally, the plan was to have three different levels of the front face: a flat one for the buttons, an angled one for the screen, and another flat top for the spinner. With the new knowledge, the new solution will be a

two-faced design, one on which the PCB will be mounted with access to the buttons and screen, and another for the spinner to sit. The LED lights, speaker, and power adapter would all have their own cutouts for access. In the final iteration, we realized this design, with a few additions as time progressed. Engravings were added to the box in two places: the first is on the front to add indication for the player selection, the second to accurately create the spinner face on the top face of the box. The box was enlarged to make room for the PCB and servo motor together in the box, as well as adding some additional space to one side of the box to accommodate the microbit. Finally, one of the panels was designed to be easily removed for viewing and access purposes.

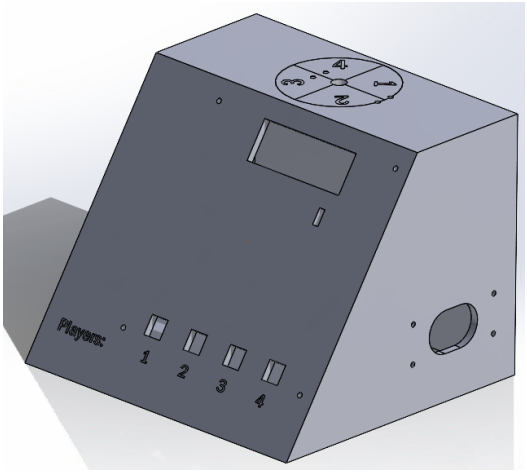

Figure 11. Final revision of the box design.

One constraint of this design is the differences in the heights of the parts. The microbit, LCD screen, buttons, speaker connector, are all at different heights above or below the PCB. In mounting the PCB to the front face of the box, there needs to be clearance enough for all components to fit while leaving access to the buttons, time-of-flight sensor, and screen. The screen is the highest above the PCB and defined as minimum standoff height. To provide access to the buttons, the decision was made to 3D print new button caps that would protrude from the box despite the increased standoff height. For the time-of-flight sensor, a hole was drilled in the front face of the box such that the laser could extend out. Another consideration was how to fasten other parts that weren't a part of the user interface. Many of the components previously considered were mounted to the PCB, and so mounting the PCB quickly became a primary concern. However, components such as the buck converter and power adapter still

needed to be involved in the planning of the box; this was first realized when considering the clearance between the servo motor and the PCB. Most of these other components have their own mounting methods, except for the power adapter. A mounting case was 3D printed to hold it in place.

4.2 Engineering Analyses and Experiment

Graphic Design Subsystem

The Graphic Design subsystem focuses on designing and developing the physical gameboard for Trivia Trek and ensuring the position spinner via Feedback Servo works properly. While the gameboard design process doesn't quite require breadboarding, it was crucial that we solidify how it would look and work, as its completion is essentially a precursor for the tracking portion within the software framework. Through various rounds of revisions, we secured the design of the gameboard and have created our final gameboard. We are confident that when elementary students play with this gameboard, they will be well-engaged, the path is fair, and the start-to-finish time will not be too long or short.

Pertaining to the spinner movement portion within this subsystem, we have ensured that its role is successful by utilizing a Continuous Feedback Servo grounded in series with a capacitor. Firstly, we implemented a short portion of code that would have the MicroBit send various PWM pulses to the Servo. Although that process took longer than expected, we were soon able to get the motor running. An obstacle we had to overcome was that because the signal and voltage coming in was too large, it had to be adjusted by using a capacitor to essentially balance the values out. In doing so, we now receive feedback with values ranging from 0 to 1023, insinuating 1024 unique Servo placements which will then inform players how many spaces to move ahead and who the winner will be.

Game Master Subsystem

The two main functions of the Game Master subsystem are to display trivia questions and answers to the user and maintain game state in software. To prototype proper display of questions and answers, breadboarding was used. A timer was kept in the MicroBit and displayed properly on the LCD. This confirmed that information can be properly communicated and displayed between the MicroBit and the LCD using the SPI communication protocol, which can be seen in the figure below. Software framework for the game has been created to show a high-level view of how the game will function. Many classes for the different hardware and software components provide the blueprint for the properties and functionalities of the components. Appendix C shows the C++ header files for each class used in the software framework. This information, along with the completed class blueprints, detail the software functionality that makes up a large portion of the device.

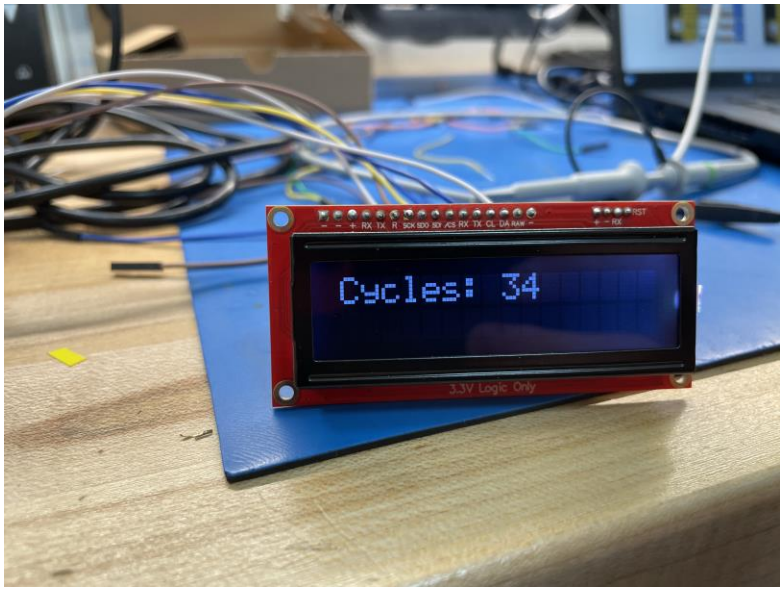

Figure 12. Confirmation that the MicroBit can successfully communicate with the display. MicroBit is maintaining a cycle count and displaying the count to the LCD using the SPI communication protocol.

The mentioned risk reduction techniques have been completed to verify the design up to the current point in the design process. As software components were developed, incremental testing confirmed that the code functions in its intended way. Full game simulation testing was conducted to verify that a game can

be completed from start to finish and meet the design requirements. This was done by multiple people playing the game, getting answer choices right and wrong, and ultimately, someone winnnig the game.

Power Distribution Subsystem

The main function of the power distribution subsystem is to ensure that all the peripherals and the microcontroller can have a safe and steady supply voltage that would be needed for safe utilization of each individual peripheral. For risk reduction the buck was tested out with a range of input voltages as well as its output tested to see how much the voltage output would vary with load current. With various amounts of load even above 1.5A the voltage drop in the buck was almost negligible, which validated that our buck would be sufficient to use in our design. We were also able to test and every peripheral that we were using was able to run off the microbit power other than the nonpixel LED's this means that the current draw is minimal for each device as the max current output of a microbit is 90mA.

With the testing of our buck, as well as its use with the barrel jack, we know that our power distribution system should be able to work with our design. In addition, when the PCB arrives, we can confirm that the power distribution is sufficient if it can keep all devices running during a simulated game at the same time without any effect in performance.

User Interface Subsystem

The main function of the user interface is to provide a seamless connection between the players and the software functionality of the game. Also, it needs to be intuitive such that anyone can understand and perform the intended functions. The components were tested to ensure their functionality by operating tests on them; these risk reduction techniques proved that the components functioned. Once the component functions were confirmed and the PCB design was complete, design and prototyping could begin.

All the components for the subsystem will be housed in a box, which will be used by the players to play the game. The first tests will be fitting tests, ensuring that the components can be accessed interfaced with form outside the box while remaining mounted inside. This is more difficult than it seems, since we're mounting multiple UI components to the PCB, which is to be located inside the box, and they have different heights associated. The final design must leave all components accessible while keeping the electronics safe; the product is intended for 4th-5th graders.

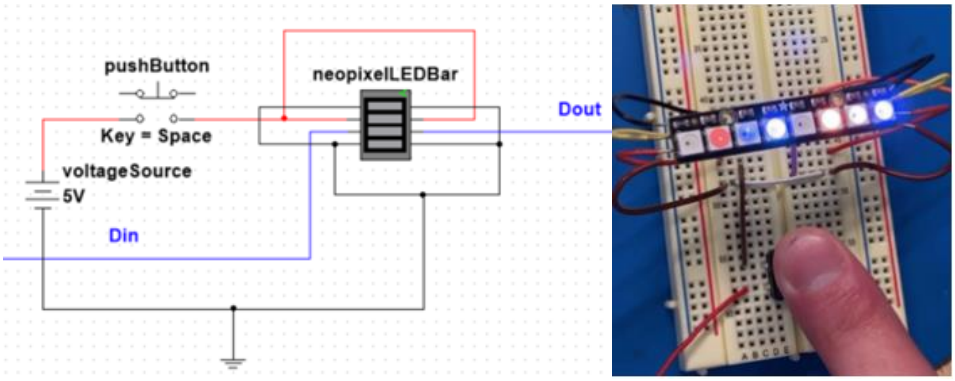

Figure 13. Schematic (left) and demonstration (right) of the neo pixel LED strip function.

The final design will be tested in two ways. First, power was applied to the user-interface components to confirm their function. For the push buttons, we will read the voltage in the depressed and released positions to be sure of the desired function. For the other components, a test script will be run through the microbit that would perform a distinct function on each component. Second, the box was presented to third-party users and their unbiased/untrained use of the product was observed. Ideally, with the instructions and materials provided, these users will be able to play our game safely and understand the uses for each component.

Full System

Now that sub-system testing had verified their respective designs, full system level testing needed to be completed after integration. To do so, variations in games were tested to ensure games could be completed

under different circumstances. To begin, different difficulties were selected along with different number of players. This tested the system's ability to handle games being played where different style questions were needed and numerous players' statuses maintained. While testing this, different user interactions to questions needed to be tested. After a question is prompted, the user can do one of four things: select the current answer, view previous answer, view next answer, or reread the question. During testing, it was verified that users can cycle through answers in either direction or reread the question at any time. The system's response to both correct and incorrect answers was confirmed as the LEDs and speaker signified with the correct color and sounds and the motor only spun the wheel when a correct answer was inputted. The MicroBit was also ensured to maintain correct state of player positions since correct updates after correct answers were made, and when a player encountered a slide or ladder on the game board, the MicroBit successfully handled these edge cases. Finally, when a player had officially won the game, the correct notification via the speaker, neo pixels, and LCD display was made to the players. After this notification, players could successfully reset the game and play again under different conditions without powering off the device.

4.3 Codes and Standards

Standards that we would need to follow are the ASTM F963-23 "Standard Consumer Safety Specification for Toy Safety, covering requirements and test methods for toys intended for children under 14 years of age" [2]. Parts of this standard that matter the most are ones that mention what forces our enclosure should handle. In addition, the product also follows the standards that ensure no hazardous protrusions. Trivia Trek also contains many questions addressing the Georgia 5.NR.4: Read, write, and compare decimal numbers to the thousandths place, and round and perform operations with decimal numbers to the hundredths place to solve relevant, mathematical problems [3]. Trivia Trek also contains questions in the ESGM4.CR.2.b. Compose rhythmic patterns in simple meter [4]. Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I2C) are two standardized communication protocols used in the system to exchange data.

4.4 Broader Impacts & Considerations

Considerations of issues related to user health, mainly the product's texture and weight, were considered in the project design. Users must not be scraped by the product and be able to easily transport it. The wheel that acts as a dice must not spin too fast as to cause harm to individuals when in operation. This influenced the design of the box to have smooth edges and corners. The weight wasn't a huge factor in the design since the parts are all lightweight and small enough for almost anyone to transport. The decision-making process considered some cultural factors. Since the product is intended for young children to use in school and at home, the design must look friendly and appeal to children, teachers, and parents from all different backgrounds.

5. Schedule, Tasks, and Milestones

	Week Number												
Task	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
PDR		Wilson	Wilson										
CDR					Rachel	Rachel							
System Design	Michael	Michael	Michael	Michael									
Graphic Design Design				Rachel									
Graphic Design Build - Canva Build					Rachel	Rachel							
Graphic Design Build - Get Canva Build on Cardboard							Rachel	Rachel					
Graphic Design Build - Laser Cut Box Itself									Rachel				
Graphic Design Finalization										Rachel			
Power Distribution Design				Shayan									
Add Power Distribution routing to team PCB					Shayan								
Power Distribution get values for power usage						Shayan							
Power Distribution - Make PCB							Shayan						
Power Distribution - Burn Test and Solder PCB								Shayan					
Power Distribution - finnish wiring									Shayan				
Game Master Design				Wilson									
Game Master Build - Question Bank					Wilson								
Game Master Build - Player Status						Wilson							
Game Master Build - Game board status							Wilson						
Game Master Build - Main Game Flow								Wilson					
Game Master Build - Put questions on LCD and let users scroll through answers to select, finalize game flow									Wilson				
Game Master Test										Wilson			
Hardware examination and interface testing with microbit				Michael									
Layout of buttons, lights, ToF sensor, and LCD on game box					Michael								
CAD model of the box						Michael							
3D print motor mount/spinner							Michael						
3D print game box								Michael					
Mount boards, components, etc to box									Michael				
User Interface Test										Michael			
Integration Phase 1: Power Dist and User Interface										Shayan			
Integration Phase 1: Testing												Shayan	
Integration Phase 2: Game Master and System												Wilson	
Integration Phase 2: Testing												Wilson	
Integration Phase 3: Graphic Design and System												Rachel	
Integration Phase 3: Testing													Rachel
System Testing													Michael
Final Inspection and Demonstration													

Figure 14. Master schedule indicating the phases of each sub-system, tasks to be completed, and integration of each sub-system with the overall system. The leads of each task in the schedule are shown with the respective week the task should be worked on and completed.

Each sub-system is broken down into three sub-tasks to be completed throughout the project. Each sub-system must go through a design, build, and test phase. The respective phases for each sub-system are shown in Figure 14 above. Integration and testing will be essential in combining each sub-system. Within the building phase for each sub-system, specific tasks are specified along with the duration and order of completion. The *User Interface* and *Power Distribution* sub-systems should be the first to be integrated to allow for the software to be integrated into the system. To test the software integration with the remainder of the system, electronic components sourcing the power and making up the user interface will be essential in confirming the quality of the system's software, which is a main component of the *Game Master* sub-system. This integration will most likely be the most challenging aspect of the project and thus, the biggest milestone to overcome.

6. Marketing and Cost Analysis

6.1 Marketing Analysis

There are many games in existence that deal with the concept of answering trivia questions to progress. The closest instance to our game is the Trivia Crack board game by Mattel. This game is a similar concept to that which we've created, where players must answer a question from one of six categories to progress. Our game differs from this in its target audience; the Trivia Crack game was intended for students ages 12 and up, which would extrapolate to 6^{th}-7^{th} grade intellect and above. Our game is meant for elementary school students and allows users to pick a difficulty level suitable for them. Other games meant for elementary school students do not provide such a range of questions, nor are they mechanized. This allowed us to streamline our design and minimize the individual parts required to play the game.

6.2 Cost Analysis (Budget)

Based on the bill of materials (BOM) located in Appendix A, it was determined that the total cost of materials for this product was $88.01. Further, the total labor cost can be found by looking first at the hours spent on the development of each subsystem and then the time spent on integration. Assuming the starting salary of an engineer is $63,000, we can determine that the hourly wage for a starting engineer is about $30.30 per hour. Using this information, the total spent on engineering development is given in Figure 15.

Development	Hours Spent	Money Spent
Game Master Subsystem	15 hours	$11.42
Power Distribution Subsystem	3 hours	$39.38
Graphic Design Subsystem	12 hours	$17.90
User Interface Subsystem	20 hours	$19.31
System Level Integration	15 hours	$0.00
Total	**65 hours**	**$88.01**

Figure 15. Table detailing the hours and money spent for each aspect of the product.

In accordance with the information presented, we've found the total cost of production to be $1969.50, including the time spent on development.

The general recommendation for a new product's price point is to mark up between 30 and 50 percent or the total cost of selling the product. This is distinct from the engineering budget, as it also includes sales expenses such as advertising, business maintenance, and shipping; it also distributes the engineering development cost across all the products sold over a given period. Assuming we would sell 1 million products in 5 years, a breakdown of the suggested selling price considering all these factors is given in Figure 16.

Cost of Materials	$80 million
Fabrication Labor Costs (post design/development)	$2000
Development Cost (distributed over all products)	$5000
Sales Expense Per Unit	$2
Markup	30%
Total Cost Per Unit	$106

Figure 16. Table breaking down the total cost per unit.

7. Project Demonstration

Our project will be demonstrated to other students in ECE junior design. A couple of members on the team will be describing the requirements, design process, building, and testing of the system, sub-systems, and components while different audiences look at and use our game. The following items and actions will be demonstrated.

- Game difficulty being selected via a time-of-flight sensor
- Number of current players being selected via pushbuttons
- Trivia questions of various topics being displayed via an LCD display
- Answer collection from the user via pushbuttons
- A spinning wheel via a servo motor with feedback
- User input verification via neo pixels
- A complete game being played from start to finish with all these components

Prototype testing for the Graphic Design sub-system included servo motor testing. In this testing, it was ensured that the motor can spin for various amounts of time and that precise motor position feedback can be received once the motor stops moving. For the User Interface sub-system, prototype testing encompassed accurate readings from pushbuttons and lighting correct neo pixels. This was done with breadboarding with the MicroBit and utilizing the serial interface of the MicroBit to ensure that correct pushbutton readings can be confirmed. Prototype testing for the Game Master sub-system ensured that communication between the MicroBit and the LCD display is accurate as well as game state can be maintained throughout an entire game. This was tested by playing multiple games with just the MicroBit, LCD, and buttons to ensure the basic purpose of the game can occur. For the Power Distribution sub-system, it was essential that power from a standard U.S. wall outlet can be converted into 5V and 3.3V for the various electrical components. With the use of a barrel jack, buck converter, and transistors, steady 5V and 3.3V sources can be supplied to the various device components. This was confirmed with a standard U.S. wall outlet power supply and multimeter to measure the sub-system's output. The overall system was tested by playing multiple games while the device was powered up. Giving various user inputs in different parts of the game and examining the other I/O devices, different styles of games could be played successfully without the device powering off.

An external demonstration of our project will be conducted soon in the form of a video. Along with this, full source code, PCB design, and component selections will be made available on the team's website.

8. Conclusion & Current Status

At the conclusion of this project, our prototype gives 4th and 5th grade students the ability to compete in a game while learning relevant information relating to their Georgia Milestone standards. The prototype uses various I/O devices to create an easy-to-use game for young students. From our initial design based off the customer requirements, there is very little that varied between that and the final prototype. A couple things that did change that we learned lessons about as the project progressed are the random number generation and servo motor feedback. Random number generation was key in our project as it randomly selected trivia questions and generated spaces to move on the board when a player got a question correct. Every time the MicroBit was powered off and back on, the same sequence of random numbers was generated. This limited variation in different games. To combat this, when players select difficulty with the time-of-flight sensor, the distance, in millimeters, was used as a random seed. This combined with the randomly generated sequence significantly increased variation in different games. Another issue that presented itself was with the servo motor's feedback. The initial design would randomly generate a number to advance a player in the game and this would be conveyed to the users by spinning the wheel, the servo motor, and stopping it on the correct number. However, getting exact feedback from the motor proved to be difficult. To solve this issue, we spun the motor for a random amount of time and then receive its position after it stopped. This position would be used to calculate which area of the wheel the motor stopped on. This changed the approach from randomly generating spaces to move on the board and then telling the motor to the new approach in which a random number was generated to spin the motor and then using its feedback, determine the number of spaces for a player to move. These two changes from the initial to final design made the game flow much better for a group of young students to compete while learning.

The information previously defined in this document, later in the appendices, and listed on the team's GitHub covers the necessary knowledge for an independent team to replicate or extend this work.

9. Leadership Roles & Contributions of Team Members

9.1 Leadership Roles

The leadership roles on the team are *Team Leader*, *Power Distribution Lead*, *Graphic Design Lead*, *Game Master Lead*, and *User Interface Lead*. Wilson is the *Team Leader*. Wilson's role is to ensure that the project remains on schedule with the progress and completion of each sub-system. The *Power Distribution Lead* is Shayan, and the role is to oversee the portion of the system that will supply power. The *Graphic Design Lead* is Rachel, and she is responsible for the design and development of the physical appearance of the game and device. Wilson is the *Game Master Lead,* and his role is to provide a software framework and ensure the codebase is efficient. The *User Interface Lead* is Michael, and he manages the user input and output to provide a clean and easy to use interface for the game's users.

9.2 Contributions of Team Members

As previously mentioned, Rachel, *Graphic Design Lead*, has worked primarily on the physical design of the gameboard as well as the portions of this project pertaining to the success and implementation of the Servo. Additionally, as *Game Master*, Wilson oversees a lot of the software involved in this project. He also maintains the role of Team Leader, ensuring that members contribute to the various assignments. As Power Distribution Lead, Shayan helped finalize the PCB design process and make the detailed BOM. He also provides insight to other members into how their various components may need to be powered and how they can be integrated into our system. As *User Interface Lead*, Michael has primarily worked to ensure success of the various UI components within our project.

10. References

[1] “Georgia Standards of Excellence (GSE),” *www.georgiastandards.org*. https://www.georgiastandards.org/Georgia-Standards/Pages/default.aspx

[2] “Standard Consumer Safety Specification for Toy Safety,” F963, https://www.astm.org/f0963-23.html (accessed Jan. 28, 2024).

[3] “GEORGIA’S K-12 MATHEMATICS STANDARDS,” Curriculum and Instruction, https://www.gadoe.org/Curriculum-Instruction-and-Assessment/Curriculum-and-Instruction/Pages/Mathematics.aspx (accessed Jan. 28, 2024).

[4] K-12- media art GSE 6 15 17 - georgia standards, https://www.georgiastandards.org/Georgia-Standards/Documents/K-12-Media-Arts-Standards.pdf (accessed Jan. 28, 2024).

Appendices

Appendix A - BOM

Item No	Description	Vendor	Vendor Part Number	Link to data sheet	Quantity
1	Team Made PCB 2 layer	JLC PCB	N/A	N/A	5
2	I2C display	Digikey	1738-1411-ND	obot_Gravity_I2C_LCI	1
3	DC barrel Jack to wire Female	Amazon	B00R1XMC6E	N/A	1
4	12-5V buck converter	Digikey	KI-78SR-3.3/1.5-W36	roducts/productdata/88	1
5	pin PCB through hole connec	Digikey	WM10719-ND	olex.com/pdm_docs/ps	1
6	16 PIN PCB mating connector	Digikey	WM13212-ND	olex.com/pdm_docs/ps	1
7	12 LED ring	Digikey	1528-1102-ND	roduct-files/2757/p275	1
8	Microfit 3.0 Crimp	Digikey	WM1837TR-ND	olex.com/pdm_docs/ps	20
9	Servo	Digikey	1738-1245-ND	opasdata/d220001/medi	1
10	N-Channel Mosfet	Digikey	-DMG3414UQ-7CT-N	s.com/assets/Datasheets	1
11	0603 standard passives	Digikey	Digikey	603	12
12	Pushbuttons	Digikey	EG1930-ND	opasdata/d220001/medi	4
13	Ultrasonic sensor	Digikey	1528-2711-ND	e0/opasdata/d220001/m	1
14	22 AWG wires	Digikey	32-24-1-0500-002-1-T	h.us/pdfs/3132-XX-1-0	1
15	5V LDO	Digikey	497-1443-5-ND	3/b0/12/d4/47/88/CD00	1

Unit cost	Total Cost		ECE Stock Y/N	If not in stock, has request to buy parts been sent to Kevin/James Y/N	Obtained Parts Y/N
3	15		N	Y	N
9.88	9.88		Y	N	Y
3.99	3.99		Y	N	Y
4.47	4.47		Y	N	Y
2.98	2.98		Y	N	N
1.54	1.54		Y	N	N
6.76	6.76		Y	N	Y
0.081	1.62		Y	N	N
17.9	17.9		Y	N	Y
0.42	0.42		Y	N	N
0.03	0.36		Y	N	N
2.15	8.6		Y	N	Y
3.95	3.95		Y	N	Y
10	10		Y	N	N
0.54	0.54		Y	N	N
Total Cost	88.01				

Appendix B – Main File

```
// main file
// copy to an arduino .ino file

#include "Bank.h"
#include "Button.h"
#include "Display.h"
#include "Game.h"
#include "LED.h"
#include "Player.h"
#include "Question.h"
#include "ServoMotor.h"
#include "Speaker.h"
```

```
#include <StandardCplusplus.h>
#include <Adafruit_NeoPixel.h>
#include "Adafruit_VL53L0X.h"

// Initialize
Bank bank;
Button redButton(6);
Button yellowButton(5);
Button greenButton(4);
Button blueButton(3);

void setup() {}

void loop() {
  // Initialize things
  Game game;
  Display lcd;
  ServoMotor motor;
  Speaker speaker;
  Adafruit_NeoPixel strip(8, 10, NEO_GRB + NEO_KHZ800);
  Adafruit_VL53L0X lox = Adafruit_VL53L0X();

  LED led(strip); // init LEDs

  strip.begin();          // INITIALIZE NeoPixel strip object (REQUIRED)
  strip.show();           // Turn OFF all pixels ASAP
  strip.setBrightness(10); // Set BRIGHTNESS to about 1/5 (max = 255)

  lox.begin();    // init TOF sensor

  delay(3000);

  led.rainbow();
```

```
lcd.displayStart(); // welcome screen
game.setNumPlayers(redButton, yellowButton, greenButton, blueButton);
delay(1000);
lcd.numPlayersPlaying(game.getNumPlayers());
delay(2000);
led.off();
delay(1000);

int diff;

lcd.diffCountdown();

while(1) {
   VL53L0X_RangingMeasurementData_t measure;
   lox.rangingTest(&measure, false); // pass in 'true' to get debug data printout!
   if (measure.RangeStatus != 4) {  // phase failures have incorrect data
      int dist = measure.RangeMilliMeter;
      if (dist < 175) {
         diff = 0;
         break;
      } else {
         diff = 1;
         break;
      }
      game.setSeed(dist);
   }

   delay(250);
}

game.setGameDifficulty(diff);
int gameDifficulty = game.getGameDifficulty(); // 0 is easy, 1 is hard
lcd.displayDiff(diff);
delay(3000);
```

```
while (1) {
    // get current player's position
    int currPos = game.getCurrPlayerPos();

    // display current player's turn
    lcd.displayCurrPlayer(game.getCurrPlayerNumber());

    // wait until player is ready (blue button pushed)
    led.rainbow();
    while (!blueButton.readButton()) {}
    led.off();
    lcd.displayGettingQ();

    // get topic of current square
    int topic;
    if (currPos % 3 == 1) {
        // math square
        topic = 0;
    } else if (currPos % 3 == 2) {
        // music square
        topic = 2;
    } else {
        // science square
        topic = 1;
    }

    // get question, correct answer, and answers vector
    std::string question;
    std::string correctAns;
    std::vector<std::string> answers;
    if (!gameDifficulty) {
        // get easy question
        Question q = bank.getEasyQuestion(topic, game.getSeed());
```

```
    question = q.getQuestion();
    correctAns = q.getCorrectAns();
    answers = q.getAnswers(game.getSeed());
} else {
    // get hard question
    Question q = bank.getHardQuestion(topic, game.getSeed());
    question = q.getQuestion();
    correctAns = q.getCorrectAns();
    answers = q.getAnswers(game.getSeed());
}

// display full question
lcd.displayFullQuestion(question);

// display answer choices
int currAnswer = 0;

// get user answer
std::string selectedAns = answers[currAnswer];
lcd.displayAnswer(selectedAns);
while (1) {
    if (redButton.readButton()) {
        // current answer is selected
        delay(500);
        break;
    } else if (yellowButton.readButton()) {
        // go to previous answer
        selectedAns = answers[--currAnswer % answers.size()];
        lcd.displayAnswer(selectedAns);
        delay(500);
    } else if (greenButton.readButton()) {
        // go to next answer
        selectedAns = answers[++currAnswer % answers.size()];
        lcd.displayAnswer(selectedAns);
```

```
            delay(500);
        } else if (blueButton.readButton()) {
            // could reread question if we want
            lcd.displayFullQuestion(question);
            lcd.displayAnswer(selectedAns);
            delay(500);
        }
    }

    // check if user answer is correct
    if (strcmp(selectedAns.c_str(), correctAns.c_str()) == 0) {
        // notify player if answer was correct
        led.setColor(strip.Color(0, 255, 0)); // green LEDs
        lcd.correctAnswer();
        speaker.soundCorrect();
        delay(1000);

        // get number of spots to move
        // int diceRoll = game.rollDice(); // get random number 1-6

        // spin motor
        lcd.displayMotor();
        delay(1000);
        int diceRoll = motor.spinMotor((game.getSeed() % 200) + 200);

        // update user position (check if new position is a slide or ladder)
        game.setCurrPlayerPos(game.checkSlide(game.getCurrPlayerPos() + diceRoll));

        delay(1000);

        led.off(); // LEDs off

        // check if user won and break loop if someone won
        if (game.checkWinner()) {
```

```
            break;
        }

        // tell player which spot to move to
        lcd.moveTo(game.getCurrPlayerPos());
        delay(5000);
    } else {
        led.setColor(strip.Color(255, 0, 0)); // red LEDs
        lcd.wrongAnswer();
        speaker.soundWrong();
        delay(1000);
        led.off(); // LEDs off
    }

    // update current player
    game.nextPlayer();
  }

  // put code here for when a player wins
  led.rainbow();
  lcd.gameWinner(game.getCurrPlayerNumber());
  speaker.soundWin();
  delay(10000);

  // wait for blue button then restart game
  lcd.playAgain();
  while (!blueButton.readButton()) {}
  lcd.restarting(); // lcd screen saying restarting game
  led.off();
}
```

Appendix C – Header Files

```
#ifndef BANK_H
```

```
#define BANK_H

#include "Question.h"
#include <vector>
#include <cstdlib>

using namespace std;

class Bank {
  private:
    std::vector<Question> easy;
    std::vector<Question> hard;

  public:
    Bank(); // Constructor
    Question getEasyQuestion(int topic, int seed); // gets random easy question with specified topic -
                                                   0 is math, 1 is science, 2 is music
    Question getHardQuestion(int topic, int seed); // gets random hard question with specified topic -
                                                   0 is math, 1 is science, 2 is music
};

#endif BANK_H



#ifndef BUTTON_H
#define BUTTON_H

#include "Arduino.h"

class Button {
  private:
    int pin; // pin on MicroBit

  public:
    Button(int _pin); // constructor
```

```
        int readButton(); // checks if button pushed - returns 1 if pushed, else 0
};

#endif BUTTON_H



#ifndef DISPLAY_H
#define DISPLAY_H

#include <SPI.h>
#include <string>
#include <vector>
#include <sstream>
#include "Arduino.h"

class Display {
   private:
      int csPin;
      void spiSendString(char* data);
      void clearScreen(); // clear the screen

   public:
      Display(); // Constructor
      void displayStart(); // welcome screens
      void diffCountdown(); // countdown from 3
      void displayDiff(int n); // easy or hard mode
      void numPlayersPlaying(int n); // how many players are playing
      void displayCurrPlayer(int n); // "Player n, your turn"
      void displayGettingQ(); // "Picking a question..."
      void displayFullQuestion(std::string _s); // Displays whole question on LCD
      void displayAnswer(std::string ans); // displays an answer choice
      void displayMotor(); // "spinning motor..."
      void correctAnswer(); // notify correct answer
      void wrongAnswer(); // notify wrong answer
```

```
        void moveTo(int n); // "Move to square ##"
        void gameWinner(int p); // player p wins
        void playAgain(); // ask user to push blue button to play again
        void restarting(); // restart game
};

#endif DISPLAY_H



#ifndef GAME_H
#define GAME_H

#include "Player.h"
#include "Button.h"
#include <vector>
#include <map>
#include <cstdlib>

class Game {
   private:
        int numPlayers; // total number of players in game
        int currPlayer; // current player turn (numbered 0 -> n - 1)
        std::vector<Player> players; // vector of n Player objects
        int finalPosition; // last square number + 1
        std::map<int, int> slides; // maps all slides from curr to next position
        int difficulty; // 0 is easy, 1 is hard
        int seed;

        void setPlayers(); // set a vector with specified number of players

   public:
        Game(); // Constructor
        void setNumPlayers(Button _b1, Button _b2, Button _b3, Button _b4); // use buttons to get
                                                                            number of players
```

```
        void setGameDifficulty(int diff); // use distance sensor to get user input difficult (either easy or
                                                                                                    hard)
        int getNumPlayers(); // return number of players

        void nextPlayer(); // updates current player to next player
        int getGameDifficulty(); // get game difficulty (0 is easy, 1 is hard);
        bool checkWinner(); // checks if current player has won the game
        // int rollDice(); // picks random number from 1-6
        int getCurrPlayerPos(); // get current player position
        void setCurrPlayerPos(int pos); // set current player new position
        int checkSlide(int pos); // check if specified position is a slide/ladder
        int getCurrPlayerNumber(); // get current player number
        void setSeed(int n); // set random number seed
        int getSeed(); // get random number seed
};

#endif GAME_H



#ifndef LED_H
#define LED_H

#include <Adafruit_NeoPixel.h>

class LED {
   private:
      Adafruit_NeoPixel& strip;

   public:
      LED(Adafruit_NeoPixel& strip); // Constructor
      void setColor(uint32_t color); // turn LEDs on a specified color
      void rainbow(); // rainbow effect
      void off(); // turn LEDs off
};
```

```
#endif LED_H


#ifndef PLAYER_H
#define PLAYER_H

class Player {
   private:
      int position; // player's square number on the map

   public:
      Player(); // Constructor
      int getPosition(); // get player position
      void setPosition(int newPosition); // set player new position
};

#endif PLAYER_H


#ifndef QUESTION_H
#define QUESTION_H

#include <vector>
#include <string>
#include <cstdlib>

class Question {
   private:
      std::string question;
      std::string correctAnswer;
      std::vector<std::string> answers;
      int topic; // 0 is math, 1 is science, 2 is music

   public:
```

```
        Question(std::string _question, std::string _correctAnswer, std::vector<std::string> _answers, int
_topic); // Constructor
        std::string getQuestion();
        std::vector<std::string> getAnswers(int seed);
        std::string getCorrectAns();
        int getTopic();
};

#endif QUESTION_H



#ifndef SERVOMOTOR_H
#define SERVOMOTOR_H

#include "Arduino.h"

class ServoMotor {
   private:
        int feedbackPin;
        int pin;
        int pulse;
        int err;

   public:
        ServoMotor(); // Constructor
        int spinMotor(int spin); // spin motor and return dice roll 1-4
};

#endif SERVOMOTOR_H



#ifndef SPEAKER_H
#define SPEAKER_H

#include "Arduino.h"
```

```
class Speaker {
   private:
      int pin;
      void playNote(int freq, int dur); // play a note

   public:
      Speaker(); // Constructor
      void soundCorrect(); // correct answer tune
      void soundWrong(); // wrong answer tune
      void soundWin(); // winning tune
};

#endif SPEAKER_H
```