Design Project 2: Bop-it! Project Final Report

Design Overview

The Stock Market Bop-It game is an interactive simulation of stock trading where players use physical actions to make decisions. The game involves three main actions: hitting the "Buy" button to purchase a stock, yelling into the sound detector to sell a stock, and shaking the device to hold onto a stock. The Bop-It design is modeled after a cash register, adding to its realism.

Originally, we had thought to use a microphone for the sound input. However, upon further review and research, we determined that a microphone was unnecessary as it would require an amplifier and recording capabilities, which were not needed for detecting sound. Instead, we replaced the microphone with a sound detector, simplifying the design.

Initially, the design for the Bop-It game included an RFID card swipe to start the game. During testing, it was discovered that the RFID scanner required a 3.3V input, while the rest of the circuit used a 5V input. We evaluated alternative RFID scanners, but after reviewing the (few good) available libraries, we decided to replace the RFID card swipe with a button, as it was not critical to the game's functionality and only an added feature.

Design Verification

The design verification for the Bop-It first included tests of our sensors. These tests revolved around gaining a better understanding of how they worked and how to best interface them with the Arduino. This first set of testing lead us to our first change in our device which was removing the RFID card swipe and replacing it with a button for the start. We made this decision after extensive testing on the RFID card swipe and realized that it was going to be much less reliable and cause more problems than it was worth, leading to replacing it with a simple button.

The next step in our design verification was to interface all the sensors with the Arduino. This process was crucial for further progress as it set up the layout for the PCB and allowed us to finalize the software for a full test of our initial prototype. This step involved us using a breadboard and the datasheets for the sensors to determine where we must interface the sensors.

The last thing we did during design verification was to determine the best way to make the enclosure. At first, we started with the idea of gutting an existing toy cash register and putting our Bop-It inside. This plan got changed as we realized the many problems that come up when using a pre-existing enclosure including hole sizing. Our plan after that included changing our enclosure design to one that we would laser cut holes to better fit all our sensors and our general design.

Electronic Design Implementation

To implement the PCB design, we began by designing a schematic based on testing of each component used in the circuit. To ensure error-free implementation, the circuit was first fully tested on a breadboard before the PCB schematic was created. To organize the multiple pin outputs on the circuit, we created an excel file for each pin output on the ATMEGA328P IC.

For a stable power supply, we added a voltage regulator to the circuit to provide a constant and controlled voltage to the rest of the circuit. Additionally, a capacitor was included from power to ground and a crystal with two capacitors on opposite sides of the crystal to set up the ATMEGA chip and ensure its proper functioning. To connect components like the gyroscope, sound sensor, LCD screen, start and

buy buttons, and speaker, we used jumper components, since these components are not attached to the board but rather to the housing design.

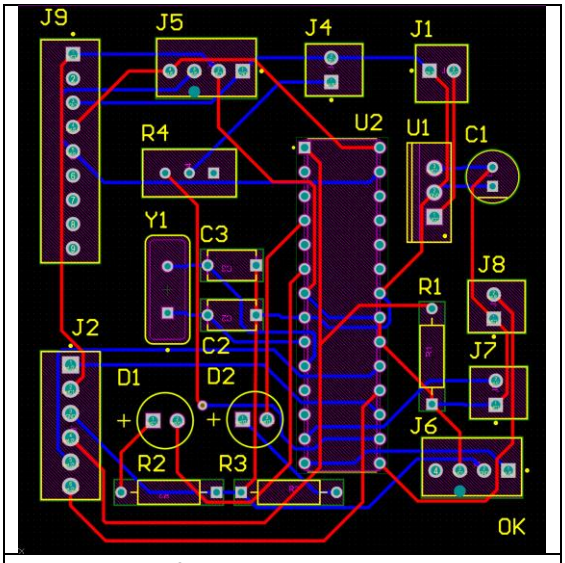
A potentiometer was added in series with the speaker to control the volume level before it was put into the housing. Two LED lights and resistors were also added to the design to signify if the action was completed within the time frame for our game. We included a jumper to reprogram the ATMEGA in case we found a flaw in the code and needed to easily change it.

To lay out the PCB, we placed the jumpers for the gyroscope, sound sensor, LCD screen, buy and start buttons, speaker, pins for reprogramming, and power supply around the perimeter of the board for an organized way of connecting the board to the components. The ATMEGA328P IC was placed in the middle of the board since most components were connected to it, making routing more efficient. The potentiometer was also placed in the middle for easier adjustment of resistance without interfering with wires coming off the board.

The two capacitors were placed to the left of the ATMEGA328P IC, as that was the most efficient route without interfering with the rest of the circuit. The 5V regulator was placed next to the capacitor across the power and ground terminals, which is next to the power jumper. The remaining diodes and resistors were placed below the crystal to fill the board. We used an autoroute program to correctly route the circuit and had no errors in routing but faced some trouble in routing the circuit to the buy button input due to its unique design. To solve this problem, we soldered two jumper wires from the button leads of the cash button PCB through the speaker holes and to the main PCB.

During testing, we found that the buy button would only work as an input if it was turned on, and that whenever the main power supply was turned off, the cash button would make a sound even when it was not pressed. We also discovered that we initially used too high of a resistance for the green and red LEDs, resulting in them being too dim, so we replaced the 10k resistor with a 1k resistor to achieve the desired brightness level.

One error we made was not including a power switch on the board, but we easily rectified this by connecting the positive lead from the battery wire to a switch, which had a jumper wire to the positive end of the power supply jumper, while the negative end of the battery connected to the ground terminal jumper.



PCB Layout of circuit.

Software Implementation

The software developed for this project is based on Arduino. The main functionalities of the software are to read data from sensors, display prompts on the LCD screen, play sound, and handle user input. The sensors used in the project are a gyroscope, sound detector, and buttons. The LCD screen is used to display instructions to the user. The code is structured in a way that a game loop is run continuously until the user fails to complete a task in time or reaches a score of 99. During each iteration of the game loop, the state of each sensor is read and instructions are displayed on the LCD screen. The subroutines developed and used in the final software implementation are as follows:

1. setup(): This subroutine is used to initialize the variables and sensors used in the project.
2. loop(): This subroutine is the main game loop, which runs continuously until the user fails to complete an action in time or reaches a score of 99. The loop first reads the state of the sensors and decides which random action to prompt the user to perform. This decision is fed to the actionDetector() sub routine where it displays the unique instruction on the LCD screen, makes a distinct noise and waits for user input. If the user performs the action within the given time, the score is incremented, and the loop continues. If the user fails to perform the action within the given time, the loop is terminated, and the final score is displayed on the LCD screen.
3. actionDetector(int action, int actionPin, String actionText, bool digital): This subroutine is called when the user is prompted to perform one of the three actions. It plays the melody associated with the action, displays the required action to the LCD screen and lights up the green LED upon completion, or the red LED upon failure. It can handle both digital and analog sensors.
4. playMelody(int melody[], int length): This subroutine is used to play the melody associated with each action. It takes an array of notes and their duration as input and plays them using the tone() function.

The code is structured in a modular way, with each subroutine responsible for a specific task. This makes the code easy to understand, modify, and maintain. The use of descriptive function names also helps in understanding the purpose of each subroutine. The code follows good software engineering practices, such as modularization, documentation, and readability. The code is also well commented, making it easy to understand for other developers.

The design process involved breaking down the problem into smaller tasks and implementing them one by one. The first step was to initialize the sensors and test their functionality. Then we designed the game loop and implemented the code to read sensor data and display instructions on the LCD screen. We then added sound and LED feedback to the game. The main challenge in the design process was to handle user input within a limited time frame. We overcame this challenge by using the millis() function to measure the time elapsed and terminate the loop if the user fails to perform the action within the given time. We also had to optimize the code to make it run efficiently within the limited resources of the Arduino board.

Enclosure Design

At first our enclosure design was going to be of a toy cash register with 3D printed parts to make the hole fit our needs. This idea soon changed as we worked through the process of disassembly and eventually changed to using a laser cutter to create our enclosure. We decided on this design method due the simplicity of our box allowing for quick and easy laser cutting.

The structure of our enclosure is meant to mimic the look and size of a cash register to fit the theme of our Bop-It. There are a couple of cutouts on our Bop-It to allow interactions with our inputs. There was a total of 6 cutouts all done with the laser cutter. The cutouts hold the LED screen, LED lights, speaker, sound detector, buttons, and a power switch. Our design also features a piece of a toy cash

register that holds both the speaker and sound detector while adding an aesthetic microphone to make the feel and visual appeal of our Bop-It more like a cash register.

For our manufacturing process we used laser cutting. Because of the size of our Bop-It, laser cutting wasn't as simple as we had thought. One of our pieces was too big to be cut out in one piece. The solution to this problem was to modify the cutout to make it 2 sperate pieces. One unplanned feature we got from laser cutting that we didn't really plan was for a clear top and bottom, however, this was a happy accident as these pieces allow you to see into the enclosure and look at the circuit without opening the box.

Assembly and System Integration

For the PCB assembly, pinheads were used at each of the jumpers and soldered accordingly, with the exception for the gyroscope which had pinheads soldered to it. These pinheads fit perfectly into the PCB and then soldered. Once everything was soldered female to female, and female to male wires to connect each circuit component to the PCB and give each component the length needed to fit within the design.

Our design had a couple steps in assembly. The first was to get the pieces without cutouts together to form a base for the rest of the enclosure to be built on. This proved to be more challenging than we thought as the pieces were cut to fit very tightly together leading to problems assembling and lead to some pieces cracking or breaking during assembly. The next step in this process was to attach the inputs and outputs to their intended cutouts with hot glue. This also had some problems due to tight cutout and we had to use a Dremel to remove the extra material. Another problem encountered was that one of our cutouts was oriented the wrong way and lead to the top not fitting on as intended.

We tested our design thoroughly at each step of the way first starting with the PCB implementations. Then as we attached more parts to the enclosure, we tested it incrementally before fully enclosing the design. One modification we made to our design was an added set of pins to our board to allow easy access to the Arduino for code updates and changes if we ever needed it.

Design Testing

Electronic Design Testing

There wasn't much debugging to do throughout the whole process other than our big green button in the center of the product. Since this was already made into a separate product, and there was power used to make a "Cha-Ching" noise, when we tried to implement it into our design it didn't work as an input to the Arduino at first. We hypothesized that there was probably a different part of the device it was grounding to and not giving the Arduino a high signal, it was looking for. It took a few times of resoldering it, but after trying a few permutations about the button leads, we landed on a set that worked both as an input and to produce the sound we were looking for.

One small thing that had to be debugged was the LEDs. At first, they did not light up but upon further testing it turned out they were put in the wrong way. On the silkscreen there is a "+" sign on the left side of the LED. This turned out to be the negative end of the LED. We have learned now to check to see if the part design in Altium is correct before hooking up a circuit. Since the voltage here was so low it did not damage anything but if it were a higher voltage circuit it could have caused problems and broken components. Also, the LEDs were much dimmer than our group initially intended. To fix this problem our group decided to lower the resistance of the resistors in series with the LEDs from 10k to 5k.

Software Testing

To ensure the software components of our design were functioning correctly, we began by unit testing each individual component. Through this process, we were pleased to discover that our Arduino, programmer, sound detector, buttons, LEDs, gyroscope, LCD screen, and speaker were all functioning flawlessly. The only roadblock we encountered was with our RFID scanner, which required a 3.3V voltage rather than the 5V voltage we were using for the rest of the circuit. While we could have employed a voltage regulator to address this issue, we made the decision to keep the design simple and not introduce multiple voltages. For a comprehensive look at our unit tests and code, you can find them readily available on our GitHub page along with the attached folder.

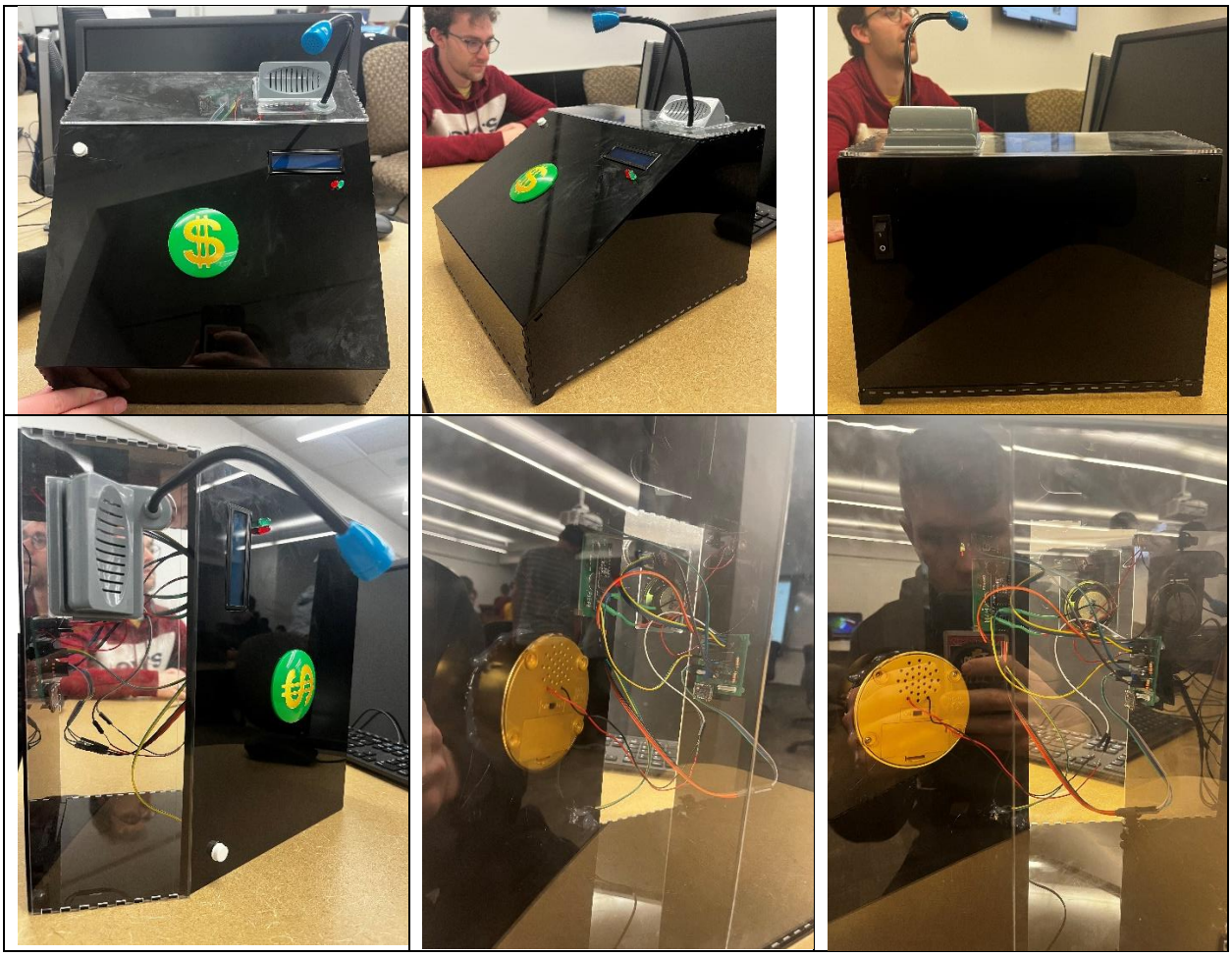
Once we had confirmed that each individual component was in working order, we then began the process of integrating these individual tests into a cohesive game loop. We began by adding some initial text to the LCD screen and verifying that we could initialize it into both the setup() and loop() routines. We then moved on to integrating the start button and ensuring that an input to the digital pin would correctly prompt different text. With these initial tests in place, we created an action handler that would generate a random action to be displayed on the LCD screen and then wait for user feedback. Initially, we used three buttons to represent the various inputs and to establish functioning digital pins. As we progressed, we slowly swapped out two of these buttons for their respective game input counterparts, testing each new addition as we went along to identify and address any potential issues. Fortunately, each new addition worked exactly as we had intended.

Once we had verified that all inputs were functioning correctly on the breadboard, we then began to add sound to the game. To achieve this, we developed a subroutine to handle generating appropriate noises for each action. After establishing the subroutine, we then integrated it with the action handler and thoroughly tested the game loop. The final step was to add LEDs to indicate when players had made correct or incorrect actions, giving us a fully functional game loop with integrated inputs and outputs. With these tests complete, we were confident in our design and ready to begin the process of soldering the components to the board and assembling it inside the enclosure.

Enclosure Testing

The testing of the enclosure started with our first model for our enclosure and lead to failure after we realized that it would be difficult to get the premade enclosure to fit our needs and it would be simpler to create a laser cut enclosure. After this first design, we moved to a laser cut model of the enclosure which had its own set of problems during testing. The first problems arose as we were assembling it, it included parts cracking or breaking and cutouts being slightly off in size. The last problem we found with the box is the cutout for the top was oriented the wrong way leading to us having to improvise the attachment of the top piece.

Although our design was able to work in the end there were some problems, we would have addressed it if we had more time to cut out a second enclosure. Another part of the testing was making sure our enclosure would hold together as it was played as it had to be shaken for one of the commands. We tested this and the other inputs on the box though a series of test run to ensure the enclosure could handle being played. The only flaw that was found during this testing was that if the buy button is pressed down too hard, it sometimes would shift and not fit into the cutout as well. This didn’t lead to any problems with the inputs, just an aesthetic problem. Below you will find some photos of the fully assembled product.



Budget and Cost Analysis

PCB Price:

Quantiy/Price Matrix

Raw Material	TG Value	Build Time	Price/pcs	PCB Cost
Kingboard KB-6160	TG130	5 days	0.25/pc	$2543

Item	Quantity	Base Price	Total Cost of Component
Gyroscope	10000	12.95	129500
Buy Button	10000	20.95	209500
Microphone sensor	10000	13.99	139900
speaker	10000	2.3	23000
9 pinhead connectors	30000	0.24	7200
Wire connectors (120 pack)	2500	6.98	17450

PCB	1	2543	2543
ATMEGA328P IC (10 pack)	1000	50	50000
Potentiometer	10000	0.6	6000
PVC	30000	35	1050000
Crystal	10000	0.18	1800
10uf Cap	10000	0.25	2500
LED	20000	0.18	3600
Resistor	30000	0.45	13500
		Total cost of Design	$1,656,493

Team

Lucas worked on the software aspects of the design. He headed writing the code and uploading it to the Arduino, as well as early stage breadboarding and testing. Then, as we met regularly as a group, we all helped during debugging of the circuit and ultimately creating what would be the final design for the PCB schematic. He also handled creating our initial workflow and GitHub page.

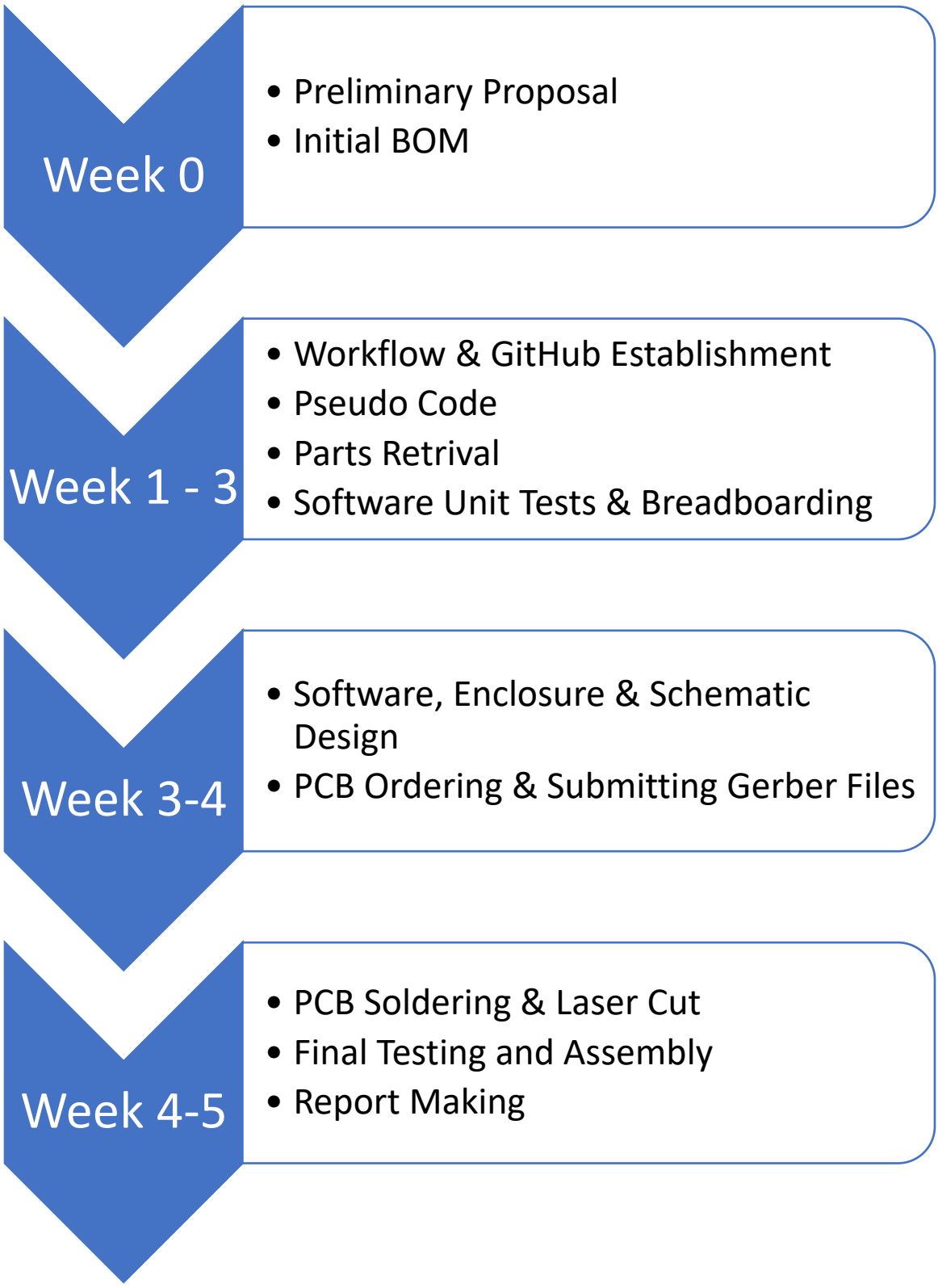
Andrew worked on the enclosure design and helped on the electrical side including back checking the PCB schematic, testing sensors, and organization of inputs. He also handled the enclosure final cutout and assembly as well as assisting in the hookup of the PCB and sensors to the enclosure frame.

Owen designed the PCB and helped to test components. He also worked on the electrical aspect of the buy button. Owen helped with the final assembly of the design, ensuring each component went in the correct spot.

Throughout the design process, we would meet weekly in the Fishbowl for two to three hours at a time to finish our design. We used SMS messaging and email to update each other on the design process as well as meeting times. We felt it was best to meet in person to do the physical testing and assembly, but for report generating and presentations making, it was fine to do it remotely. We believe our team worked great together, and that it was our primary reason for being able to complete our project in such a timely manner.

Timeline

Feb 16th – March 26th



Summary, Conclusions and Future Work

In conclusion, the Stock Market Bop-It game was a great way to learn team building skills and practice real on-the-job skills. Through a series of design iterations and testing, our team was able to refine the initial concept and create a fully functioning prototype. The decision to replace the RFID card swipe with a button, switch to a sound detector for sound input, among many other pivots, were crucial design changes that simplified the circuit and improved its reliability. The team's attention to detail in sensor interfacing and enclosure design resulted in a polished final product that met the project's goals.

There are several areas in which the software logic could be improved. For instance, when the ON switch is toggled, the game starts without requiring the start button to be pressed. During initial breadboarding and PCB testing, this didn't pose a problem, but once we assembled everything, the start button was skipped on the first game. After completing or losing the first game, the start button worked correctly, waiting for user input before prompting for actions. We're uncertain why the start button fails during the first round since the code worked correctly initially. Additionally, we overlooked implementing cases where pressing the wrong action would result in losing the game. If we had more time, we would have passed all the sensors to the action handler and added a clause to lose the game if the state of non-prompted actions changed. We could have also attempted to order another RFID scanner running on 5V, but the available libraries were not user-friendly enough for our liking. Lastly, in a later model, we would add a "press to talk" button before speaking into the sound detector to prevent exploiting the loophole of repeatedly pressing the buy button to win the game by triggering the "Cha-Ching" sound.

Reflecting on our Bop-It project, we've identified a few areas in which we could improve the enclosure design. Firstly, we would have opted for a smaller size, as the benefits of a larger size were outweighed by issues with production, assembly, and gameplay. To make the enclosure more visually appealing, we would have liked to add stickers or other decorative elements to showcase the theme and attract more attention to the device. Additionally, we would have improved the box's fit, as the tightness of the enclosure caused some damage during use, and we would have addressed issues with hole sizing to ensure a better fit for components such as the LED screen.

Another significant change we would have made is creating a mount for the battery, as currently, it is only attached to the side of the Bop-It enclosure with hot glue. This would have made replacing the battery difficult, so we would have added a battery mount to the PCB or somewhere in the housing to allow for easy battery replacement if needed. Overall, these improvements would have helped enhance the Bop-It's functionality, ease of use, and aesthetic appeal.