Fredrick Laudati, Leslie McDonough, Jill Zitcovich

Design Overview

For our second design project, our team was tasked with drawing inspiration from the classic childhood game *Bop-It* and putting our own unique twist on the concept. After brainstorming and discussing each of our original proposals, we decided on the concept proposed by Jack: a Bop-It game reimagined as a day in the life of an ECE student.

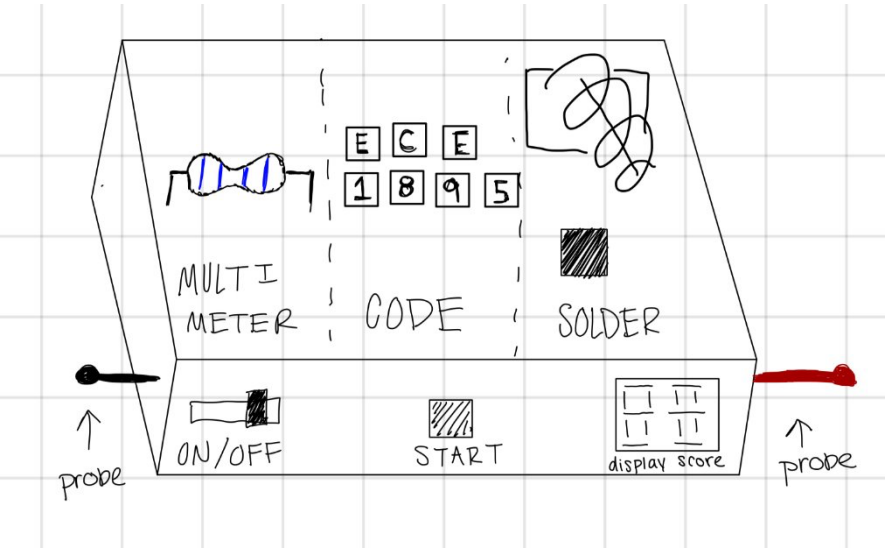

Figure 1: Preliminary Sketch

The final design integrates tasks that simulate common activities and techniques we use daily in our coursework and labs on the twelfth floor. The project combines both creative problem-solving with technical challenges to provide a fun yet educational experience.

Original Design Concepts

In the early stages of development, our group considered several different approaches in reimagining Bop-It. We considered how we would want each task to be completed. For example, for the code-it command, we originally considered requiring the user to input key presses in a particular order. After asking potential target users (other students) about what they thought, the feedback we got told us that this may be too complicated and take away from the simplicity and speed that characterizes the original bop it. We also originally considered having multiple different resistors of different values but decided for this first iteration of the device, we would not pursue that avenue. Challenges that prevented this from being implemented are detailed later in the report.

Final Design

The final design of our ECE Bop-It incorporates three primary inputs, each representing a unique task from and ECE student's daily routine:

1. **Multimeter-It!**
   User is prompted via a command from the speaker to measure the resistance across a resistor using two multimeter probes. One multimeter probe is connected to VCC, while the other is connected to an analog input pin of the Arduino. The task is deemed successful when the correct analog value is measured and detected by the Arduino, simulating the frequency use of multimeters in circuit analysis.
2. **Code-It!**
   To mimic the numerous coding assignments we encounter, the user is prompted to press a set of keycaps mounted at the center of the game board. The keys are connected in parallel, with each key connecting a digital input pin of the Arduino to VCC. Pressing any key sends a high signal to the Arduino, marking the task as completed.
3. **Solder-It!**
   The final task replicated the precision and skill involved in soldering. The user is prompted to use a soldering iron to touch a designated metal pad on the board, completing a connection. The soldering iron is connected to VCC and the metal pad on the board is connected to a digital input pin on the Arduino. The Arduino detects the that this task is completed when the digital input pin goes high. Successfully completing this task simulates the critical job of soldering in assembling and repairing PCBs, an important skill used when assembling this project.

Design Verification

The initial prototype of our design underwent thorough testing to verify both the hardware and software functionality. To ensure the final design would meet the desired specifications, we employed a step-by-step approach, starting with physical prototyping on a breadboard and then refining the associated software.

Hardware verification was done through a breadboard prototype. This is ideal for testing as it is flexible and decreases and chances in error when designing final PCB for manufacturing. Connectivity tests were performed using a multimeter to verify proper wiring and ensure that all components were securely connected.

The circuit was built step by step. First the Arduino circuit was built ensuring that the inputs could be detected in code and verifying which pins on the Arduino would be used as input and output. Then the LCD screen was added to the circuit, ensuring that both the code to print the score to the LCD screen was correct and the circuit connecting the LCD display to the Arduino worked. Finally, the DFPlayer mini mp3 player was added to the breadboard and tested to ensure that the circuit and code was in place to play audio files.

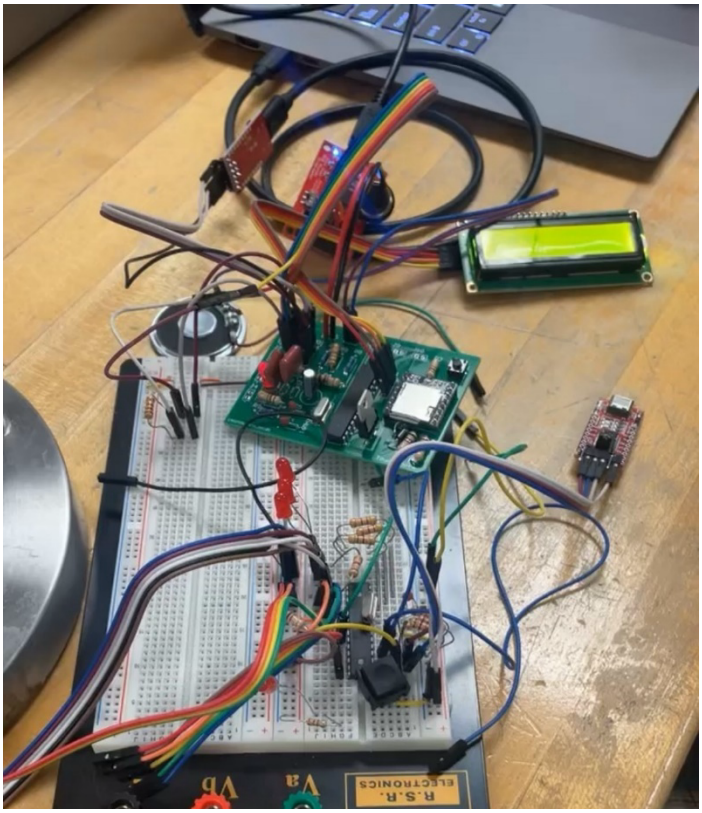

Figure 2: Breadboard Prototype, shows working components during testing

During our verification of the DFPlayer circuit design, we discovered that both the circuit and the code was incorrect. We researched multiple different circuits online for the DFPlayer and tested each, along with their corresponding code, until we found a pair of code and hardware design that worked properly.

After confirming the functionality of each individual component, the full game software was written and tested on the bread board to ensure that the code had no bugs before it was flashed to the PCB. One difficulty discovered was that the value read on the analog pin was often too low and would fluctuate. This issue was accentuated when users were rushing to get the input in time because they often would not make good contact with the multimeter probes. The software solution to this problem is detailed in the software implementation section.

Once the PCB was manufactured, the code was written to the Atmega and, again, the system was tested in parts. First, it was confirmed that the startup button worked in starting the program by turning on the LCD and displaying a score of 0 as well as playing the startup message. When this was achieved, we had verified that the power circuit, lcd circuit, and DFPlayer circuit were correct. Before putting the board in the enclosure, wires were used to short the solder and code inputs and a resistor to connect the multimeter input to effectively ensure that each input functioned.

Electronic Design Implementation

After verifying the functionality of our hardware and software prototypes, we transitioned to designing and manufacturing the PCB for our Bop-It game. This phase required careful planning to ensure that all components would work together seamlessly and that the final PCB design would replicate the same success as our breadboard prototype.

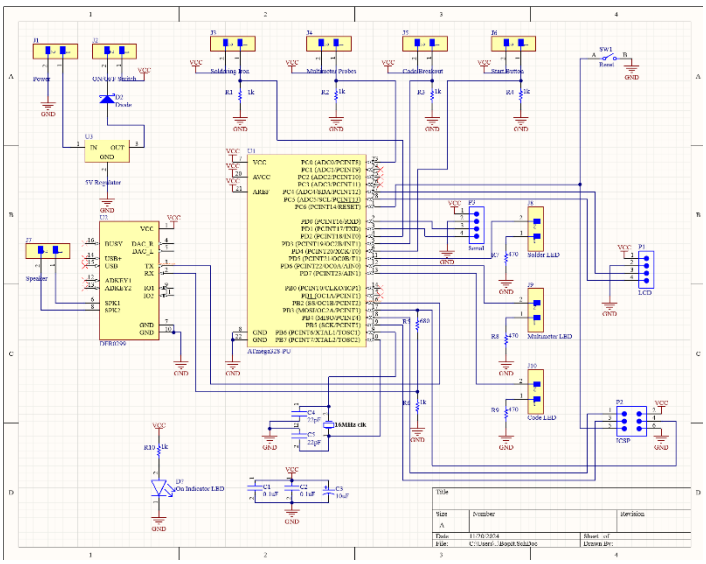

Figure 3 3: Main PCB Schematic

Our main PCB schematic was designed to incorporate all the necessary components while also adhering to the constraints of cost, size, and functionality. Key components include:

Capacitors: Used to filter the power supply and stabilize voltage, ensuring consistent operation of the circuit.

LEDs: One LED is for debugging, indicating that the circuit is properly powered. Others signal to the user that they properly carried out the task.

Reset Button: Allows for easy debugging and resetting of the system when needed.

Serial Header: Facilitates debugging by allowing communication between the PCB and external devices.

ICSP Header: Allows the board to be reprogrammed while in the PCB so the timing constraints for the game can be determined through experimentation

Pin Headers: Enabled easy soldering of wires to the PCB for the inputs and outputs.

Diode: Protects the circuit form potential damage caused by reversed polarity if the power supply is incorrectly connected.

Clock Crystal: Placed near the ATmega microcontroller to minimize both noise and improve timing accuracy.

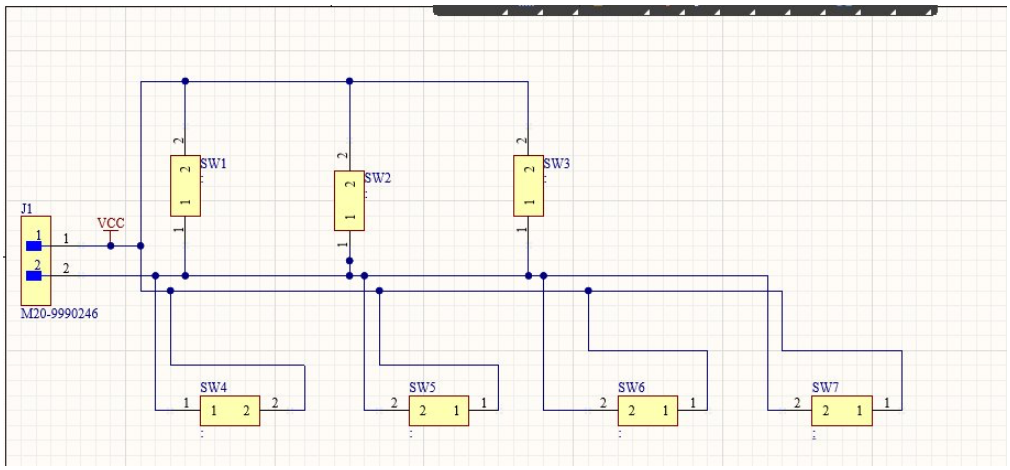

Figure 4 4: Breakout Board Schematic

The breakout board is simple, just involving keycaps placed in parallel so that pressing any keycap completes the circuit. There is a breakout board so that the sensitive components can be put on the main board and protected in the enclosure while the keycaps can sit at the top of the design where the user could access them.

PCB Layout

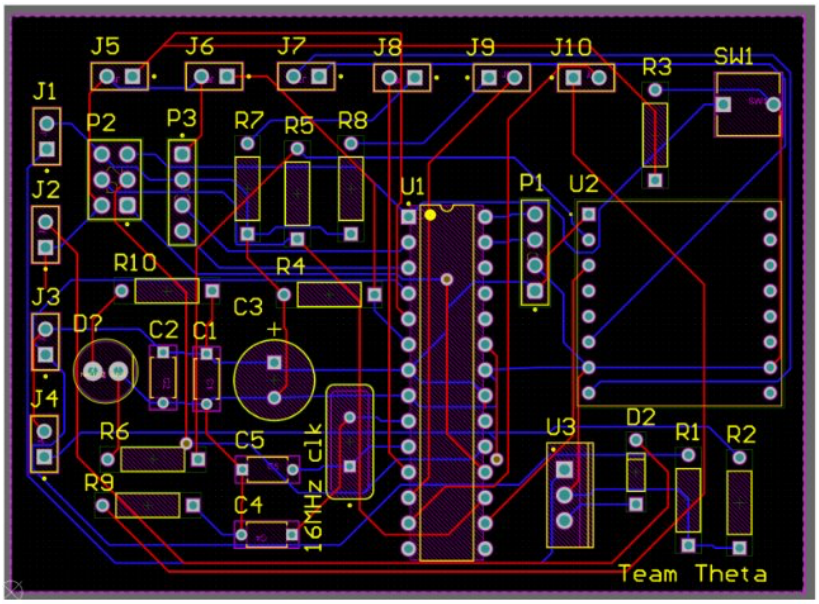

Figure 5 5: Main Board PCB Design

The layout of our PCB was designed with functionality, efficiency and compactness in mind. To optimize the arrangement:

- **Key Caps on Separate Board:** The keys for the *Code-It* input were placed on a breakout board positioned on top of the enclosure. This separated them from the main PCB, reducing the size and improving accessibility.
- **Two Port Connectors:** Placed around the edges of the board to help with wire management and keeping the inside of the enclosure neat.
- **Routing and Vias:** Careful attention was paid to routing paths to avoid crossing connections. Vias were employed to simplify routing and connect traces on different layers of the board.
- **Compact Design:** The overall board was kept as compact as possible to reduce manufacturing costs.
- **Reset Button Placement:** Positioned at the edge of the board for easy access during debugging and testing

We adhered to PCB best practices, such as placing the clock crystal close to the microcontroller to minimize noise and grouping the related components, so they logically streamlined connectivity. The design process was lengthy, as we wanted to be careful in paying attention to detail, and there were times where last-minute changes were made to certain components, so the board would have to be re-routed upon this change.

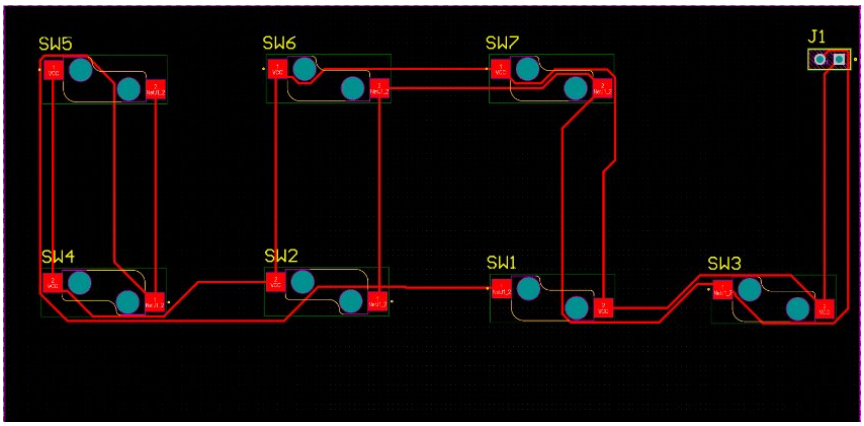

Figure 6 6: Breakout Board PCB Design

A sperate breakout board was created to house the keys of the Code-It input. This breakout board sits on top of the enclosure, allowing the keys to be easily accessible to the user while protecting the remaining electronics within the enclosure. The key switches were spaced to fit properly on the breakout board and ensure smooth operation during gameplay.

Software Implementation:

Our software, written in C, was designed to seamlessly interface with all hardware modules, enabling the functionality of our game. The program manages user interactions, task commands and feedback via visual and audio indicators.

The program begins by initializing the serial monitor to print messages that aid with debugging. It also initializes the DFPlayer by sending a startup message to the player and waiting for a message back. This ensures that the DFPlayer is properly reading the SD card before the program starts. The startup process also involves reading one of the unused analog pins to seed the random number generator. Since this analog pin is not connected, the Arduino will just read noise and the value read will be random, providing a random seed for the random number generator. This ensures that the pattern of inputs the device requests varies between each run of the game. Finally, the lcd screen is started up and all the global variables are reset.

The program then enters a run loop that waits for the start button to be pressed. Once the start button is pressed, another loop is entered and the game stays in this loop until the gameOver variable is set or the score reaches 99. At the beginning of each run of this “playing” loop, the lcd is updated to display the new score and a random command is generated. There is a delay of 1

second after the command audio starts playing to ensure that there is no glitch in playing the audio. This delay also prevents the previous user input from being read as the current input if they don't stop doing the action immediately. At the end of each playing loop, the time to enter the command is set to be 98% of the previous value, with a minimum time of 1 second. This value was chosen to make the game become exponentially harder but not reach the saturated minimum time to quickly. The equation of the time that the user has to give each input is shown in the below graph.

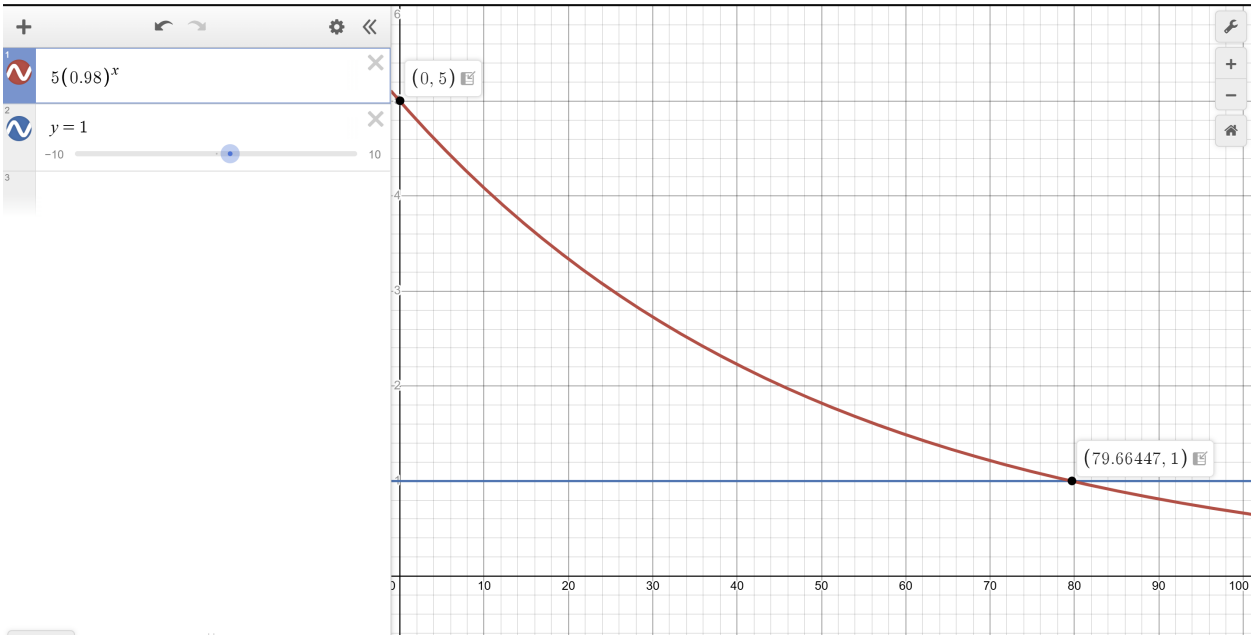

Figure 7: Time given to user to complete command

After a command is issued, the code enters another while loop where it waits for an input to be made or the time for this command to run out. When the user provides an input, the code checks that it is the correct input. If it is correct, this inner while loop is exited and the "playing" loop starts over, issuing a new command. If it is incorrect, the gameOver variable is set and the code breaks out of this inner loop. Because gameOver is set, the "playing" loop will not run again.

Once the code exits the "playing" loop, the code checks if the user reached 99. If they did, it updates the LCD (because the LCD will be at 98 since it is only updated at the beginning of the loop), and the win audio is played. Otherwise, the game over audio is played. Either way, the device then waits for the code to be restarted.

The main challenge faced while writing the code was that the multimeter action was not being read properly by the Atmega sometimes. Using the serial monitor, it was determined that this issue was due to the Atmega reading too low of a value and thinking that the user did not touch the probes across the proper resistor. This issue often happened if the user did not make solid contact with each leg of the resistor or the contact was too brief. Since the game is made for

speed, though, this was a bug that we wanted to fix. Therefore, the code provided below loops until a non-zero value is read. In addition, it reads values for half a second and takes the highest value read to ensure that if the user touched the multimeter probe, the correct analog input is read. This does not slow down the program because this type of polling only occurs if the multimeter it command is entered (the analog pin reading went high enough that there must have been some action by the user).

```
//If the anologue pin is not close to 0 (being pulled down), the multimeter action is being complete
if(analogRead(MULTIMETER_PIN) > 10){
  //If this is the correct action, check to make sure the correct resistor is being probed (the analogue input is in the right range)
  if(currentTask == MULTIMETER_IT){
    Serial.println("Correct Action");
    //Poll the analog pin for 500ms (and until the resistorVal read is not 0) and take the highest reading to make sure we are getting an accurate reading
    int resistorVal = analogRead(MULTIMETER_PIN);
    Serial.println(resistorVal);
    int start_time = millis();
    while(millis() - start_time < 500 || resistorVal == 0){
      int tempVal = analogRead(MULTIMETER_PIN);
      if(tempVal > resistorVal){
        resistorVal = tempVal;
      }
    }
    Serial.println(resistorVal);
    //If the correct resistor is being probed, increment the score and light up the correct choice light for this action
    // //If the reading is 0, something is wrong so keep reading the analog input until we get a valid input
    // while(resistorVal == 0){
    //   resistorVal = analogRead(MULTIMETER_PIN);
    //   Serial.println(resistorVal);
    // }
    if(resistorVal > 100 && resistorVal < 800){
      correctChoice = true;
      digitalWrite(MULTIMETER_LIGHT, HIGH);
    //Otherwise, set game over to true
    }else{
      gameOver = true;
    }
```

Figure 8: Example of function for Multi-meter It!

Enclosure Design

For the design of our enclosure, we used makercase.com to download a simple 12” x 5” x 1.5” box with fingers so that we could glue together the acrylic pieces. We strove to make the design as small, light, and portable as possible so these dimensions represent the minimum size we could use while still comfortably fitting all the inputs. To make it more specific to our design, the original box was edited in AutoCad to properly fit the components on the lid, buttons on the front, and have holes for the wires in the back. A caliper was used to measure our components so that the holes for the wires, LEDs, screws, and speaker were the proper size. One important consideration we made was ensuring that there was a way for the battery to be replaced by a user even when the box was fully glued together. For this, we had screw holes for the keycap PCB to be screwed to the top of the box so it could be removed whenever the battery needed to be replaced. The enclosure design is depicted below.

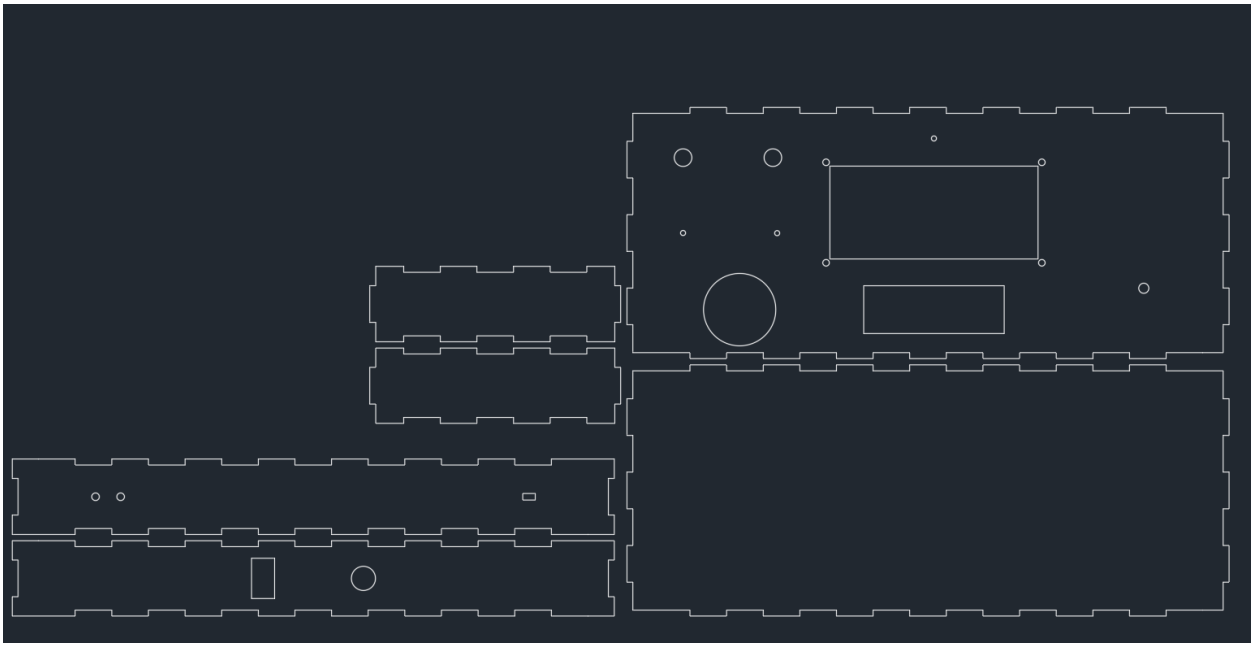

Figure 9: Enclosure Footprint

Additional measures were made to create a polished and fun to use toy. For realism, our enclosure uses an actual soldering iron and soldering iron stand as well as actual multimeter probes. The multimeter probes sit in holes cut into the top of the box for easy use. The hole size was chosen so that the probes do not fall out during transport but can easily and quickly be removed by the user. Another choice we made was to make a larger clay resistor instead of just a regular resistor so that it is easier for the user to see. We put an actual resistor inside the clay so that it still functioned as intended. Using a real soldering iron presented a challenge because the heating element in the soldering iron caused there to not be continuity between the cord and the tip of the soldering iron. Therefore, we took apart the soldering iron and modified it, running some of our own wires, so that it acted as a simple wire.

Fredrick Laudati, Leslie McDonough, Jill Zitcovich

Assembly and System Integration

The assembly and integration of our design required careful attention to detail, from integrating individual components to ensuring the overall functionality of the system. Throughout this process, we encountered and resolved several challenges, which ultimately strengthened our final design.

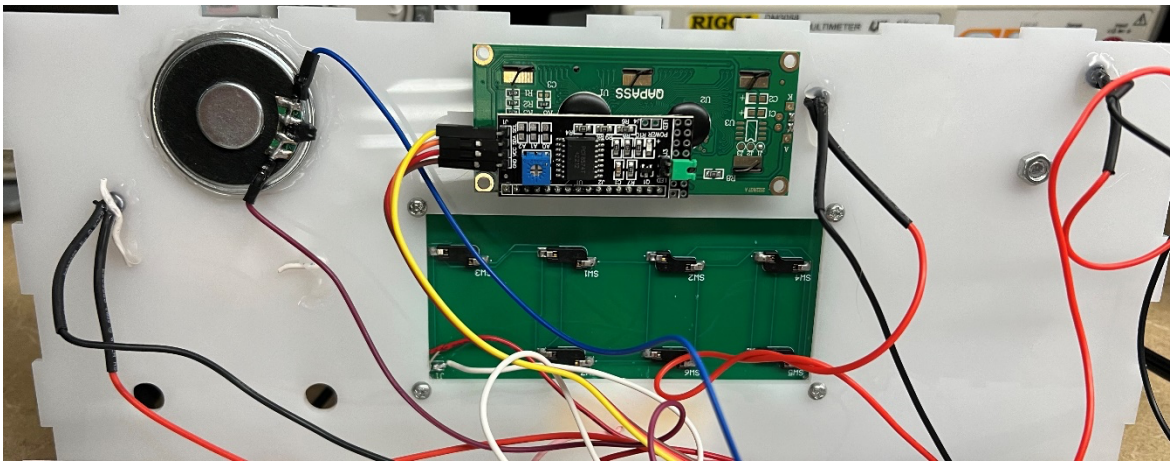

Figure 10: Final Product (top of lid: LCD display, speaker and breakout board shown)

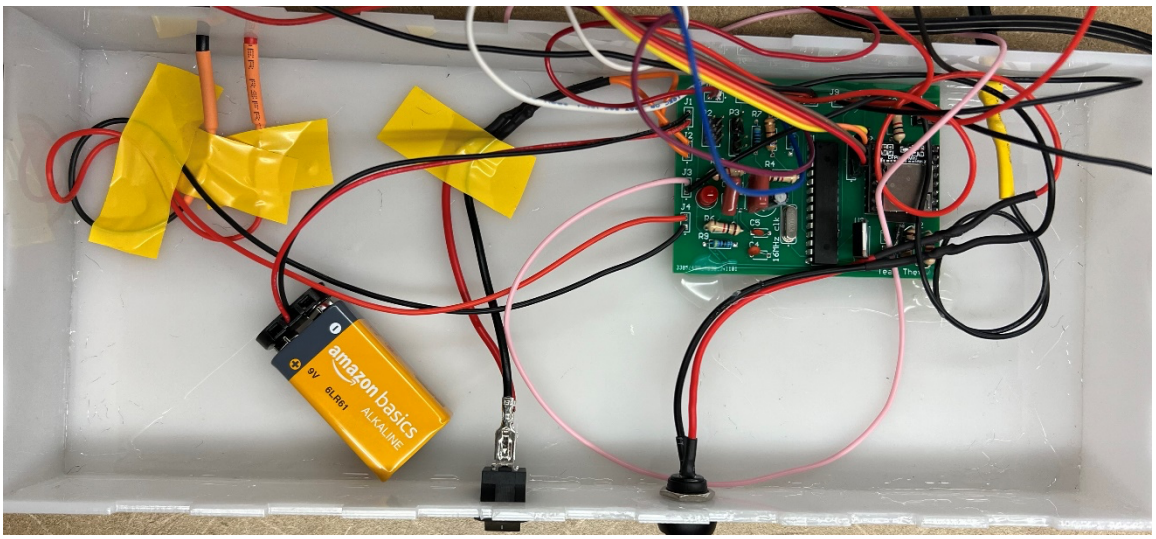

Figure 11: Final Product (bottom: Main PCB Board and power supply shown)

To integrate the soldering iron, we initially considered 3D printing a custom iron to simplify connectivity. However, we determined that routing a wire through the existing soldering iron provided a more efficient and realistic solution. This method allowed us to establish a proper electrical connection between the soldering iron's tip and the rest of the circuit ensuring accurate signal detection and task completion.

When assembling the breakout board for the key switches, we quickly realized the footprint we had downloaded from SnapEDA did not include the posthole for the keycaps we had ordered. Rather than manufacturing a new breakout board, we simply modified the keycaps by removing the post while still enabling a secure fit. This solution saved both time and resources while maintaining the functionality of the board.

Another issue that arose with the capacitor values chosen for the clock system used by the ATmega chip. During initial testing, the board could not be programmed because the Atmega was not receiving the proper clock. The issue was identified by using an oscilloscope and comparing the waveform on the PCB circuit to that of the breadboard circuit. This enabled us to identify that the capacitor values used for the clock on the PCB were incorrect and we were able to correct that. These adjustments restored the board's functionality and highlight the importance of validating design choices and schematic reference.

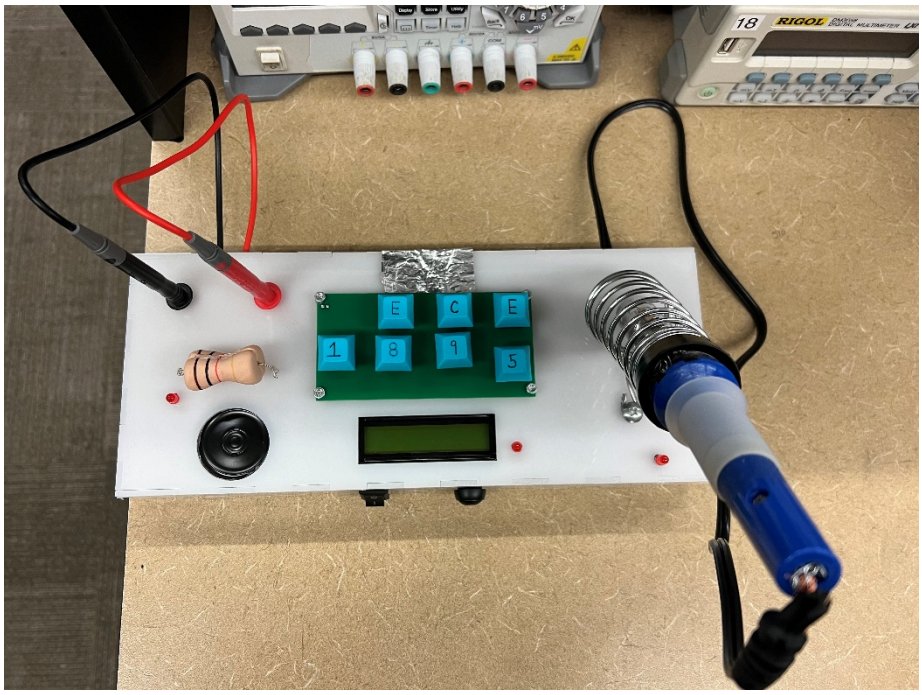

Figure 10: Final Assembled Product

During manufacturing, we burnt out the backlight of the LCD display and had to replace the display to ensure that it was bright enough to read. At first we thought it might be a power issue so we used an external power supply to rule this out as the issue.

To maintain clean and reliable wiring throughout the system, we used heat shrink tubing to insulate the connections and minimize the risk of short circuits. Inside the enclosure, we organized wiring with tape and hot glue to keep it secure and improve visual clarity for

troubleshooting. This same hot glue was used to stabilize the PCB reducing the chance of damage or disconnections when the game is moved or subjected to impact.

Although these challenges cause several modifications, it is a part of the design process. This is giving us the opportunity to enhance our design and technical skills.

Design Testing

As mentioned in the other parts of the report, testing was done throughout the design process. We built the circuit incrementally on the breadboard to verify the functionality of each part of the design. We also tested the PCB incrementally to test the functioning of each part one at a time. This incremental approach to testing helped us to isolate potential issues with the system and debug more easily. If we had skipped directly implementing the whole system, we would have had many variables to deal with each time there was a bug and resolving issues would have been a much lengthier process. Because of our incremental approach, when we had issues with the music player, we were able to isolate the problems to the music player circuit and code. In addition, getting the breadboard to be fully functioning before testing the PCB helped with debugging issues on the PCB. When we faced issues programming the PCB, we were able to use the oscilloscope and compare its waveform to what we saw on the breadboard. We could then isolate the differences between the two to quickly identify the issue. Finally, building debugging tools into the PCB and the code, such as serial output and the ability to reprogram the board, ensured that we could quickly identify and resolve software issues as well.

Budget and Cost Analysis

Main PCB:

BOM:

Quantity	Link	Description	Pitt User ID	Unit Price	Extended Price USD
10	https://www.adafruit.com/product/5123	Key Switch	lpm43	$0.95	$8.60
1	https://www.adafruit.com/product/5005	Keycaps	lpm43	$5.95	$5.95

Total Cost for Key Switches: $66,500.00

Total Cost for Key caps: $41,650.00

Price Estimation from Manufacturer:

	Raw Material	TG Value	Build Time	Price/pcs	PCB Cost
✓	Kingboard KB-6160	TG130	7 days	0.26/pc	$2551

An additional $1261.00 for shipping.

Price Estimation for Assembly:

Design Project 2: Bop-It Project Final Report

Fredrick Laudati, Leslie McDonough, Jill Zitcovich

PCBA Price:

Price & Delivery

Lead Time	Unit Price	Quantity	Price
4 days	$0.10/pc	10000pcs	$974.26

Total Cost: $112,636.00

Price Per Board: $11.26

Breakout Board:

BOM:

Quantity	Link	Description	Pitt User ID	Unit Price	Extended Price USD
1	https://www.adafruit.com/product/4958	Switch Sockets	lpm43	$4.95	$4.95

Total cost for switch sockets: $17,325.00

Price Estimation from manufacturer:

PCB Price: Quantity/Price Matrix

Raw Material	TG Value	Build Time	Price/pcs	PCB Cost
Kingboard KB-6160	TG130	7 days	0.05/pc	$454

An additional $144.00 payment for shipping.

Price Estimation for Assembly:

PCBA Price:

Price & Delivery

Lead Time	Unit Price	Quantity	Price
4 days	$0.10/pc	10000pcs	$1014.77

Total Cost: $18,937

Price Per Board: $1.89

To assemble 10,000 copies of our design it would be a total cost of $131,573. Estimating the enclosure to cost to be roughly $10 per acrylic box, it would be an estimated total of $231,573. Meaning it would be roughly $23.16 per Bop-It.

Team:

Throughout the project, each team member had specific responsibilities, but we relied on one another to ensure all tasks were completed effectively. Jack served as the design lead, creating the initial concept and managing the enclosure design. Jill focused on the hardware, developing the breakout board and the original main PCB design. While the PCB design went through several interactions and changes, it became a collaborative effort for the entire team. Leslie concentrated on the software, working closely with Jack to ensure the game interface would work properly for users. We frequently remained in contact with one another, often working on specific components on our own and then coming together in person to integrate or debug them with on another.

Timeline:

Week 1:

Design Proposal, preliminary sketch, pseudo-code, general schematic

Generating list of necessary components and creating BOM

Week 2:

Software debugging

Design of Rough Enclosure

Design of First PCB

Week 3:

Breadboard Prototyping, and creating software

Debugging original prototype

Week 4:

Final PCB Design and verification

Send PCB Design in for manufacturing

Final Order of Parts

Week 5:

Final Code Draft

Final Enclosure Design

Week 6:

Manufacturing and Assembling Enclosure

Final Stages of debugging and assembling Final Product

Week 7:

Final Project Demonstration

Final Project report

Summary, Conclusion and Future Work:

This design project provided valuable experience in teamwork, as we collaborated to achieve a single cohesive product. It enhanced our understanding and ability to utilize GitHub, teaching us how to properly upload and collaborate on code with one another. Additionally, we gained exposure to various CAD design techniques and the process of 3D printing and utilizing different resources offered here on campus.

For future design iterations, we would consider creating the enclosure out of clear acrylic. This would allow for electronics to be shown off to users and the game to be more on theme with our major. To expand the functionality of the game, we would also add more resistors and require them to probe the correct resistor, teaching users the color code of common resistor values. Finally, a volume knob would be a nice feature to add so that the game operates at the proper volume for each environment we are in.

This design project combined both techniques and concepts learned in our ECE courses while also applying a creative mindset. This was incredibly beneficial as we moved forward within our academics and careers as it challenged us to bring forward a successful product.