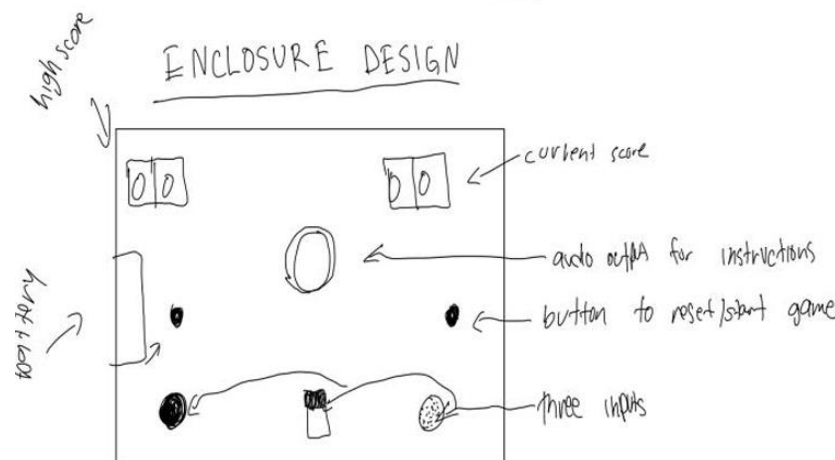


## Design Overview:

Our groups design consisted of five inputs, three push buttons, two switches, and a potentiometer as well as two outputs, an LCD screen and speaker all connected to an Arduino 328 mega chip to simulate a game that resembled the popular children's toy, Bop-It.

Our game had three user inputs for selecting the correct answer based off our LCD screen and the speaker the design also had a way to start the game, restart the game, and turn the device on and off. The design was created using these inputs and outputs combined with an Arduino 328 mega to program all our inputs and outputs. We then created a 3-D printed box to place or finished design in.

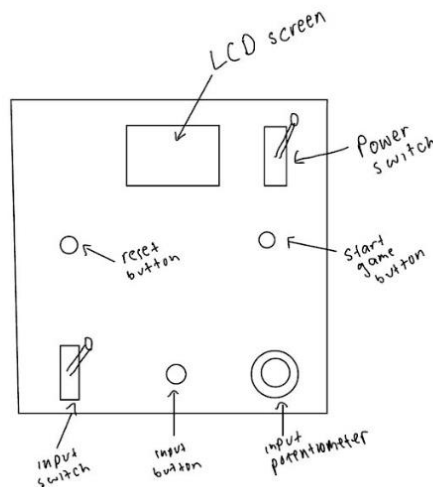
Our original idea was quite like our final implementation with some minor tweaks in the hardware. Below is our original design schematic we created before we began the project:



Initial Design Layout Schematic

From this design to our final implementation, we ended up switching the multiple LED displays to a single programmable LCD display due to it being less components and simpler to implement. Our current LCD displays the current score, the high score, and a timer for the user to see how much time they have left to enter their answer. We also

switched the microphone on the bottom right of the schematic to a potentiometer because we thought it would be more user friendly as well as easier to implement. We also added a switch to turn the game on and off and also a push button to restart the game that is not seen on our initial design. Below is a final schematic of our finished design for reference:

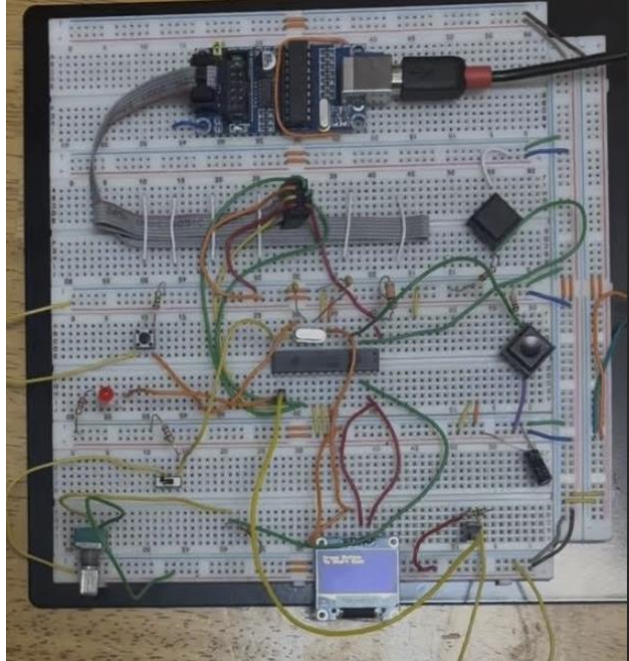


Final Design Layout Schematic

## Design Verification:

To test our design, we created a bread board prototype that included all of our inputs and outputs we intended to implement. Once we had finished designing our prototype we tested it thoroughly. Our main way of testing the hardware was to write small Arduino scripts for each one of our inputs and outputs before we implemented it into our final prototype/Arduino script. This way of testing helped us debug components easily because we did not have to examine as many components as we could have if we were to put them all together at once. Once a piece was tested, we would add it to our main prototype and repeat on the next component that had not yet been tested. Once that was completed, we put them all together into our final prototype and tested that as well.

For the software aspect, because we knew that all the hardware components were already working from our previous tests, it was much easier to debug. A lot of our software debugging involved us finding an error in how the components were working versus how we expected them to and then going to that specific area of code and talking out the logic with one another until we found the issue. This proved a successful method, and we were able to fully implement our working prototype on the breadboard. Below is a photo of the final design of our initial prototype we created and tested:

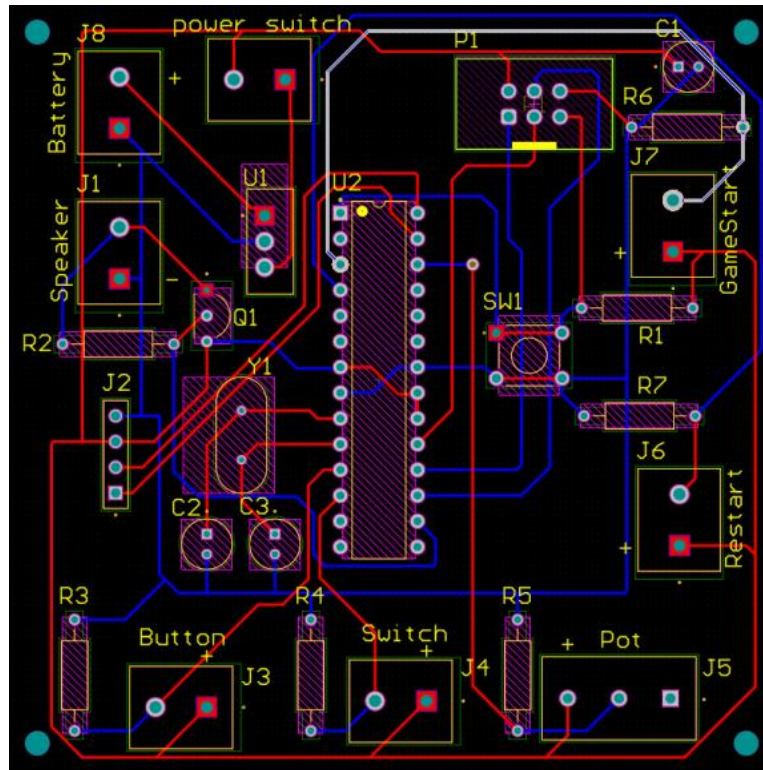


Initial Prototype

### **Electronic Design Implementation:**

The components that we used on our PCB are block connectors, 4 pin headers, 6 pin headers, a voltage regulator, a MOSFET, a crystal oscillator with capacitors, various resistors, and lastly the ATmega chip. Each of these components played a vital role in the functionality of our design and were chosen to accomplish a specific task.





Moving on to the crucial components of our design, the first and most notable component is the ATmega chip. This lies directly in the middle of our PCB and acts as the brain for our entire circuit. It reads in the inputs from the connector and sends them to the 4-pin connector so it can be displayed on the screen. Directly connected to the bottom left of the ATmega chip is the crystal oscillator and capacitors, these components produce the clock cycle that the ATmega chip runs on. The next important component that is probably overlooked in most cases is the resistors connected to the input blocks. These are crucial components in our design because they are functioning as pull-down resistors and make it so the ATmega does not read a false high and prematurely end someone's game. Lastly, there are two components that play a similar role to each other, and those two components are the voltage regulator, and MOSFET connected to the speaker. The voltage regulator, labeled U1 in the schematic, takes in a 9v supply and steps it down to 5 volts so it can be used by the ATmega. Similarly, we were not certain if the speaker would require more voltage than what the ATmega could supply. So, we have the MOSFET act as a switch that supplies 5 volts to the speaker when triggered by ATmega.


After talking about the components that we used in our design, I would like to talk about the actual layout of PCB, and some of the best practices that we tried to follow. When it comes to the physical layout of our board, we tried to keep all the connectors near

the outside edge of the board. This was done so the connectors would be easily accessible, and so the wires could be inserted into the connectors without interfering with other components. Another thing to note is that we tried to keep all of the game inputs at the bottom of the board, since all of the input were going to be placed at the bottom of the top panel of the enclosure. This was done so we would not have to have the wires stretching from one end of our enclosure to the other. Although this PCB was significantly more complex compared to the PCBs we have designed in the past, we still tried to follow the best practices to the best of our ability. The first practice that we followed was to keep components that we knew were going to be connected close to each other. This was done to reduce the length of the traces being ran. The other practice that we tried to follow was to reduce the capacitive nature of PCBs by keeping the power and ground traces sperate from each other and reducing the number of times they overlap. The last practice that I tried flowing that wasn't mentioned in this class, but was mentioned in ECE 1890, was to reduce the amount of traces that meet at right angles. This last practice has very little effect on this PCB since it is so low power, but is good practice if I ever design higher power boards.

Messages						
Class	Document	Source	Message	Time	Date	No.
[Warning]	Sheet1.SchDoc	Compiler	Net VCC has no driving source (Pin J8-2, Pin U1-1)	2:20:00 PM	3/29/2025	1
[Info]	Project2_Bopit.PrjPcb	Compiler	Compile successful, no errors found.	2:20:00 PM	3/29/2025	2

Details	
Net VCC has no driving source (Pin J8-2, Pin U1-1)	
Wire NetJ8_2	
Pin J8-2	
Pin U1-1	



Design Rule Verification Report

Date:3/29/2025

Time:2:21:59 PM

Elapsed Time:00:00:00

Filename:C:\Users\Public\Documents\Altium\Project2\_Bopit\PCB1.PcbDoc

Warnings:0

Rule Violations:0

Summary

Warnings	Count
Total	0
Rule Violations	Count
Clearance Constraint (Gap=8mil) (OnLayer:Top Layer) (All)	0
Clearance Constraint (Gap=7.874mil) (All) (All)	0
Clearance Constraint (Gap=8mil) (OnLayer:Bottom Layer) (All)	0
Short-Circuit Constraint (Allowed=No) (All) (All)	0
Un-Routed Net Constraint (All)	0
Modified Polygon (Allow modified: No) (Allow shelved: No)	0
Width Constraint (Min=3.5mil) (Max=100mil) (Preferred=15mil) (All)	0
Routing Topology Rule/Topology=Shortest (All)	0

Overall, there were only some small challenges that we face when designing our PCB and schematic. These challenges were mainly just getting the correct footprints for all of the components that we used. Without these correct footprints it would have been a lot more challenging to successfully put together the PCB. The two images above show our successful schematic and design verification.

## Software Implementation:

Our code was created using the Arduino IDE using C++. We also included a library specifically for our LCD display called adafruit. Our software design consisted of a declaration of all our input and output pins as well as other variables we needed to keep track of things such as the timer and highscore. The initializations were then followed up with multiple sub routines to start the game, keep track of the inputs, and determine the user interactions with the game.

Our first subroutine consisted of a set up function used to initialize all the pins connected on the board as well as a set up for the LCD display we used. Our next routine was a function named loop that used multiple loops throughout to keep track of almost all our functionality while the game is running. This function is responsible for checking the win condition of the game (when a player reaches the score of 99), checking if the game has been started by the user, outputting a random instruction for the user as well as the correct tone from the speaker, setting the high score, reading the user input, keeping track of the timer, and speeding up the game as the users score got higher. Our final routine was called check input and was responsible for receiving the users input from the previous instruction. This function was used to take in the previous input from the user and react to that, whether it was correct or incorrect. If it was correct the score would increment and the game would continue, if the input was incorrect or the user ran out of time then the game would end and output the failure message and tone while also restarting the game.

The design process we took while creating this code was simple. We essentially started with a very simple design with barely any code and from there started to slowly implement all the functionality we wanted to include within our design, debugging as we went, until we had fully implemented the software. This proved successful to us and we were able to methodically implement everything we wanted to with little friction other than a few bugs here and there.

A lot of our challenges coding this design originated from our inexperience with the IDE. We had a lot of issues when it came to implementing the LCD display because of this. We did not know the correct pins or that we needed a separate library to implement it so there was a slight learning curve when it came to implementing that component in the software. We also ran into an issue with the font where we could only adjust the font in whole numbers, so it was either too small or much too large. We spent a lot of time trying to figure out a solution but, in the end, we just had to use the smaller font. We also had a few bugs in our code such as the game not starting/restarting, the timer being too fast, and some delay between the speaker and the text direction. These were all easy to resolve and just originated with us overlooking something simple.

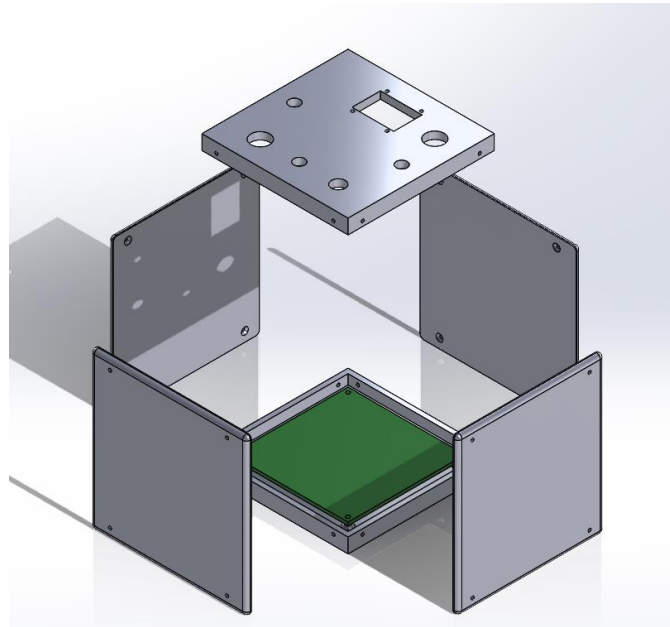
The link to view our full code for this project will be provided at the end of the report. The link will be for our GitHub we used while making the design. The code is located in the Bopit\_project folder within the repository and the file is named Bopit\_project.ino.

### **Enclosure Design:**

For our enclosure design, we wanted to go with a small, compact portable game that could easily fit all of our components into it. The reason for this was because we were



originally going to give the bop it to Nick's younger siblings so they could play with it. We also wanted to keep the cost of the enclosure down, and didn't know if we would have to remake it so we decided to go with 3d printing. Since we were going to use 3d printing we wanted to make the enclosure as 3d printer friendly as possible this means reducing the amount of hanging edges and reducing the amount of thin vertical structures. To reduce the vertical structures we split up the model into several different part which were the front and back walls, left and right walls, top face, and bottom face. The reason for this constraint was to give our enclosure the best chance of being 3D printed on its first attempt.



These constraints eventually led us to the square enclosure of our final design. There were a couple of added benefits that came along with this design, the first was that it gave us easy access to the pcb just by removing a couple of walls. The second benefit is the print time for the enclosure was significantly reduced than if we had just left it as one piece. Fortunately, our measurements were accurate, and we did not have to reprint anything due to it not fitting.

Although, we did not have to do any reprints because of measurements being off there were still some troubles that we had with 3d printing. These troubles had to deal with build plate adhesion, and printing the small holes we need so we could screw our enclosure together. When the printer would try and print these small holes, the small circles it would try and print would peel up from the build plate. To get around this issue I just added a chamfer to the inside of the circle which slightly increases the diameter of the circle where it meet the build plate allowing it to have better adhesion. As mentioned

previously the enclosure was held together with screws, and the inputs were mounted onto the top plate of the enclosure.

Assembly and System Integration:

### **Design Testing:**

After we completed our initial prototype of our final design with the PCB and the 3-D printed enclosure we were very confident that it was a successful design. This was due to the testing mentioned in the Design Verification section as well as the testing after we fully soldered all our components to our PCB board. Our final testing of our design consisted of us just playing the game and seeing how all the components came together.

### **Budget and Cost Analysis:**

The costs associated with our design came from the PCB board that we printed as well as the ATmega chip and our bill of materials we submitted to make our PCB board easier to implement and our inputs better looking. Using the website, PCB Way, we calculated that our board cost about 25 dollars to design and ship. Our bill of materials including our order from DigiKey and Amazon ended up being 41.73 dollars. The ATmega chip we used to design our entire project around is priced at 2.89 dollars on the DigiKey website. To make the enclosure using the 3-D printer the raw material used in printing the enclosure costs around 5 dollars. This was cheaper because we used our own printer we already owned. In total our design cost about 74.62 dollars. To find out how much it would cost to produce our design in bulk and design 10,000 copies our game we used the PCB Way calculator and our bill of materials to calculate the final price to be in the range of 200 – 250 thousand dollars. This was calculated using the discount price for the PCB, the unit price of the mega chip, the LCD, and the per unit of price of each bill of material component we used.

### **Team:**

Due to our team only being a two-person group we were both very hands on with every role in creating this project. Nick LaVine was responsible for most of the software design while Colton Frankenberry was responsible for most of the hardware. For creating the enclosure and PCB, Nick and Colton worked together but Colton did most of the 3-D printing and enclosure design because he has more experience with it than Nick. Overall though we did almost all the project together to help each other out due to lack of members and it also made the project design more fun. The method we used for

communication throughout the project varied between texting, calling, and discord. Through those three means of communication, we were successfully able to organize group meetings and discuss project ideas/issues.

### **Timeline:**

The timeline of our project:

3/18 – Created the GitHub and the workflow for our group project as well as the enclosure design we wanted.

3/20 – Created the bread board prototype for our design as well as the software for the design as well.

3/27 – Created our PCB design and sent it to get it printed.

4/6 – Soldered our components onto the printed PCB and tested its functionality.

4/12 – Finished and printed the enclosure for the design and added our PCB into it.

4/13 – Fully completed our design.

The dates above are not exact but generally accurate for when we finished our significant milestones for this project.

### **Summary, Conclusions, and Future Work:**

This project was the accumulation of a lot of small parts coming together to create our final design. Each part took careful examination and effort to create a successful design implementation, and we were very happy with our outcome of the game. We have both concluded that without working through the design process we would not have been able to successfully complete our working final design. If we had more time to work on this project, we would have made more improvements to both our hardware and software. In our case we would have liked to add more layers to the game. We had the idea in our initial project proposal before we started implementing it that we would add multiple levels to the game such as an easy, medium, and hard mode that the user could select. We would have also liked to implement more input as well as a nicer display. Another thing we wanted to add but did not have enough time is a better speaker that produces a voice instead of a frequency pitch. Finally, the enclosure could have been more creative as well to give our game a special and unique feel to it.

**Links:**

Github - <https://github.com/colton-frank21/JD-Project-2/tree/main>