Omega Group

Dr. Dickerson

Junior Design

12/3/2020

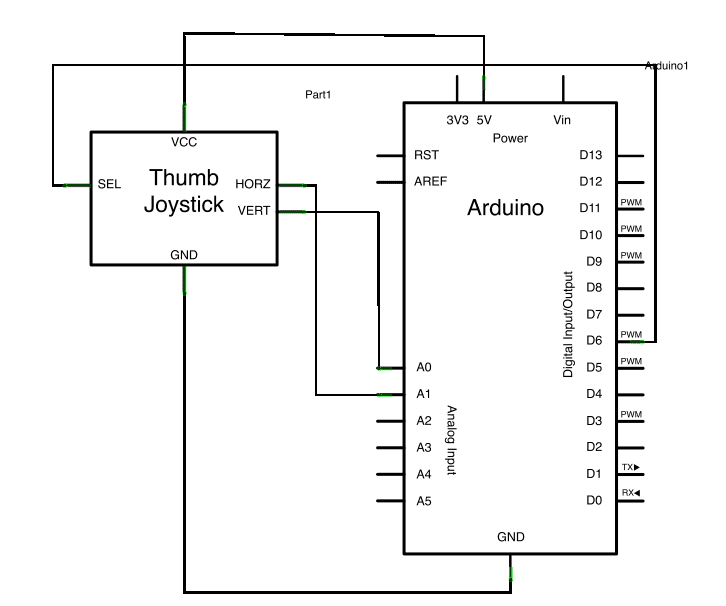
**Final Report**

**Design Overview**

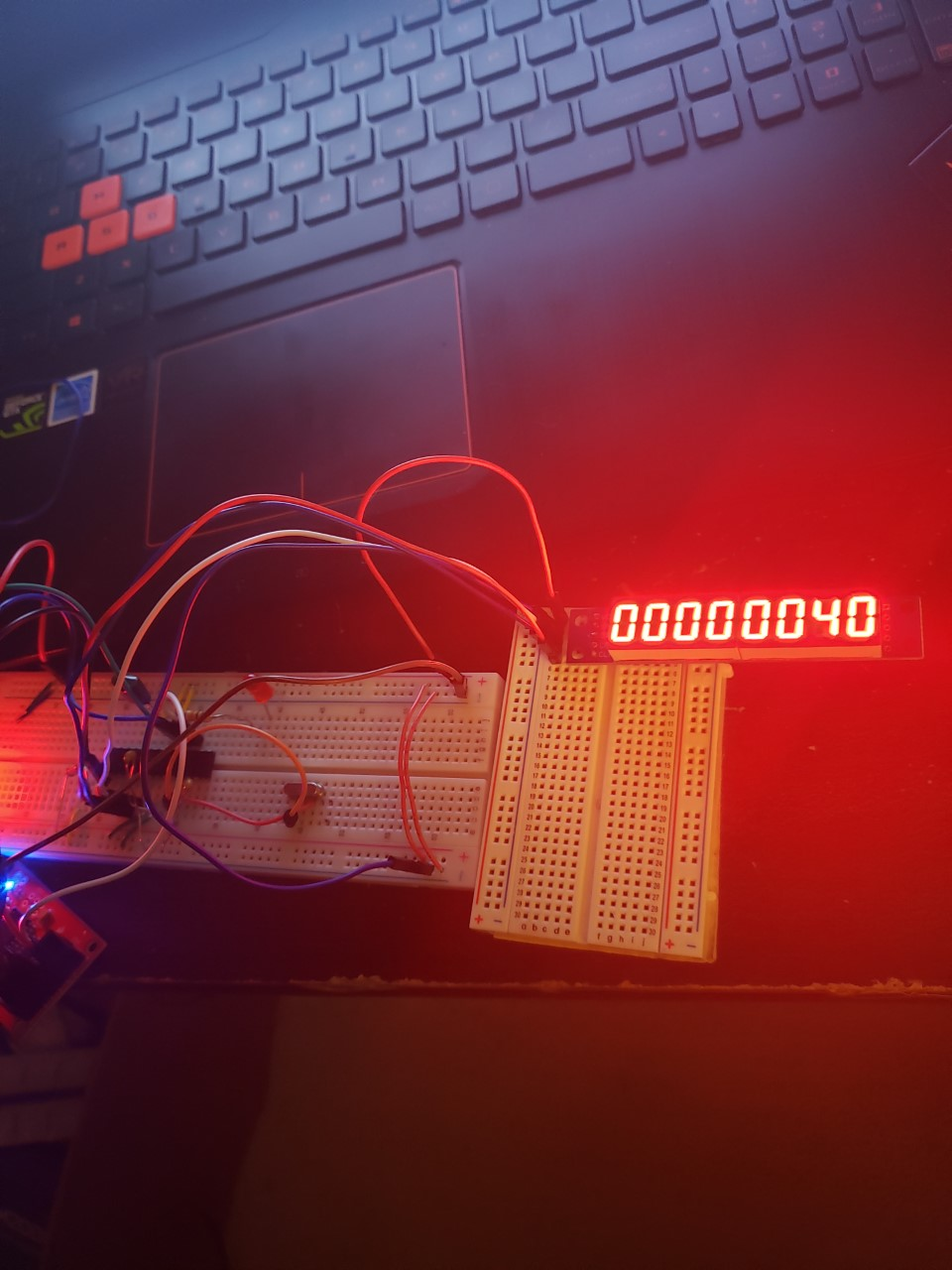
For our final design for the Bop-It project, we used a button, joystick, and a tilt sensor for our three inputs. The button and the tilt sensor are simply pressed and tilted whenever they are selected by the system. However, when the joystick is selected, an LED will light up and the user must push the joystick in the direction of that LED. The LEDs and the sequence of tasks to complete are all pseudo randomly generated. The score and time left in the round are displayed on 8 seven segment displays. A speaker is used to notify the user of which input is required to beat the round. If the user selects the wrong input or the time runs out, they lose, and the final score is displayed. Every couple of rounds the time given to provide the correct input reduces.

**Design Verification**

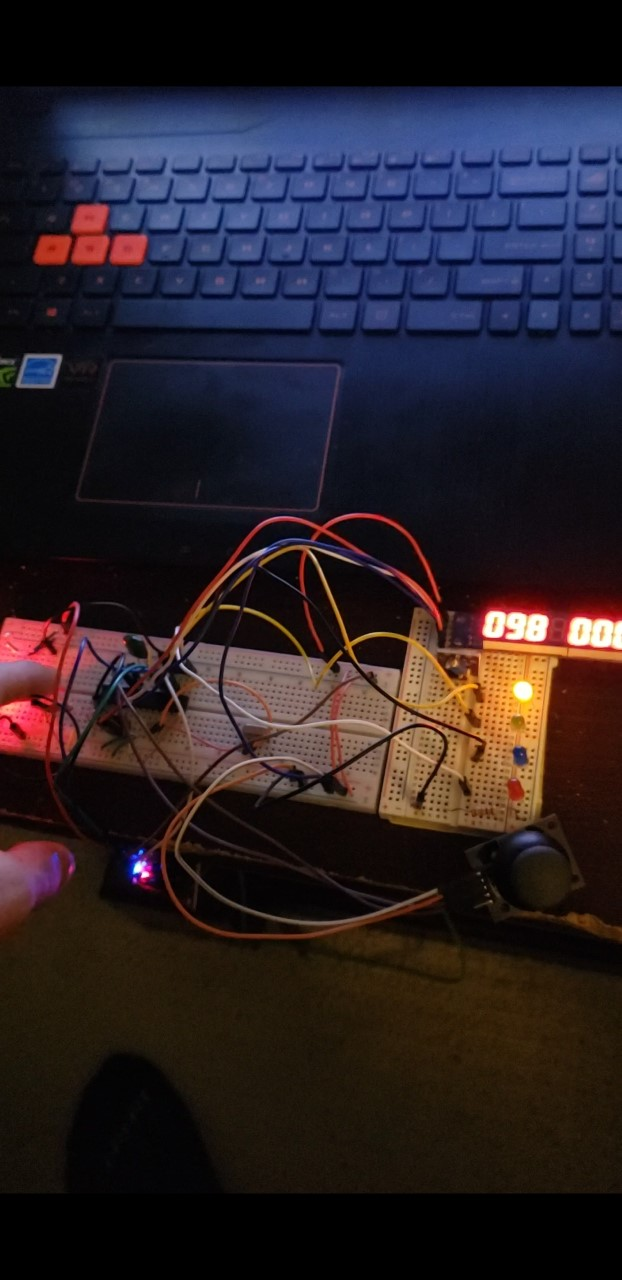
To verify our design, we started by implementing one component at a time. First, we started with the joystick, since we deemed that this would be the most difficult. We set the joystick up according to the example at [https://www.arduino.cc/en/Tutorial/BuiltInExamples/JoystickMouseControl](about:blank). However, instead of an Arduino, we used a ATmega328 microcontroller. Next, we verified the x and y axis inputs using the serial monitor on the Arduino software.



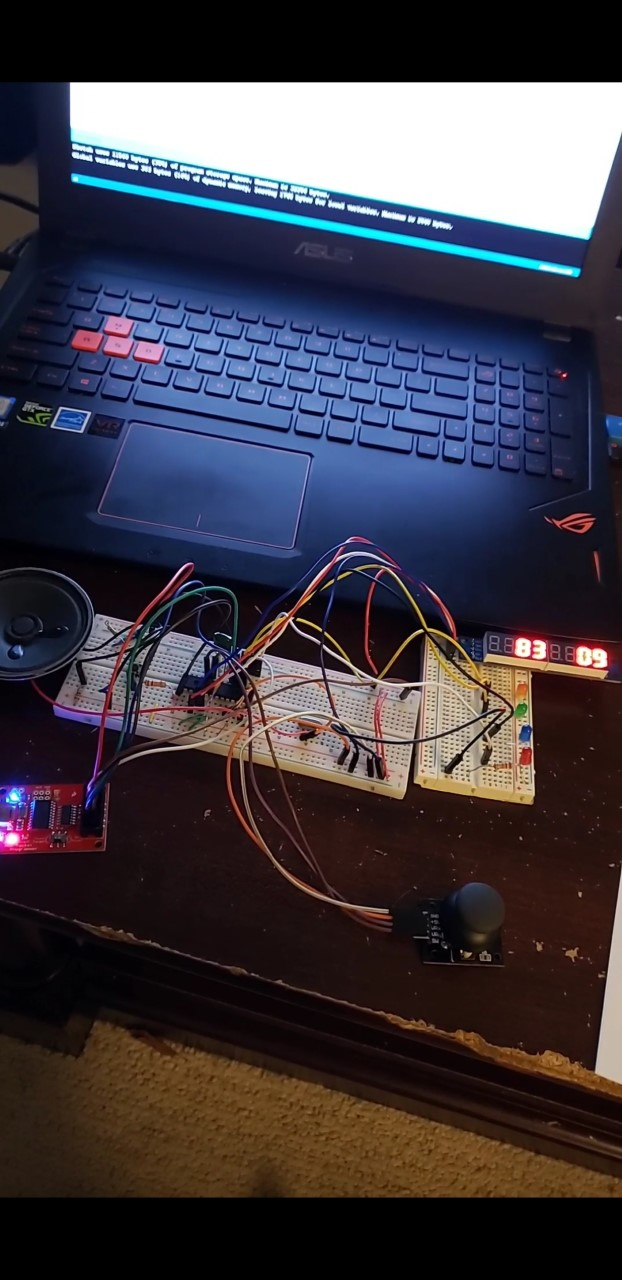
The next step was to add the screen. We connected the screen to the joystick circuit using the example at [https://www.instructables.com/Arduino-and-MAX7219-8-Digit-7-Segment-BCD-Counter/](about:blank). We read in the joystick’s x and y axis values and display them on the screen:



Our next goal was to work in the LED’s for the joystick, timer, score, and button for our breadboard verification. We added the LED’s and added code to randomly assign one to be lit up. We section the joystick’s x and y axis and gave each LED its own quadrant. The breadboard model now had two of our inputs working as a standalone game:



Our last step in the breadboard verification was to acquire a tilt sensor, speaker, add them to the circuit, and create functions for both. Once we incorporated everything, we tested the functionality by testing every possible combination of requested input vs what the user selected. For example, we would push the joystick up whenever the tilt input was required to make sure the game recognized that as a loss. Below is the final breadboarded version:



**Electronic Design Implementation**

**Software Implementation**

To write the code for the microcontroller, we used the Arduino software and the built-in libraries for our components. In the setup section of our code, we initialize our inputs and outputs, they are:

* Output Pins:

4 LEDs

Speaker

Screen

* Input Pins:

Joystick

Button

Tilt Sensor

Also, in the setup section, we define the variables needed, such as the user’s score and time per round. We also incorporated a start up tune that the speaker plays and defined objects corresponding to the joystick and display.

The loop section of our code is where the game repeats. Here we seed our random function by using the input to an analog pin that is not being used. We then use the random function to get a number between 1-3 to select which of the three inputs are to be used. Before the appropriate function call is made, a frequency is played to the user to let them know which input is the correct one. Every ten rounds the user passes, the time per round is reduced by one second. This leaves them with one second in the last ten rounds of the game.

There are 8 other functions used in our code:

* HexToBCD

This function converts a decimal to binary-coded-decimal. This is used to output the right numbers to the display.

* getJoystickAndScore

This function is called whenever the joystick input is randomly selected. Inside, another random number between 1-4 is generated to select a LED to light up. This LED corresponds to a quadrant of the joystick. If the user pushes the stick towards this LED, the score is increment by one and the next level is started. If they select the wrong input, or the time runs out, they lose and the game ends.

* getButtonAndScore

Like the joystick code, this function is called whenever the button input is selected. If the user pushes the button in time, the score is increment by one and the next level is started. If they select the wrong input, or the time runs out, they lose and the game ends.

* getTiltAndScore

This is the function for when the tilt sensor is randomly selected. The user must tilt the device within the allowed time to pass the level. If they do not, they fail.

* updateScore

This function updates the score to the display. It is called throughout the three input functions.

* endGame

endgame is called whenever the player fails. The function plays a defeated melody, and the final score is displayed on the screen.

* winner

This function is called if the user’s score reaches 99. The function plays a winning melody, and the final score is displayed.

* easterEggTone

If the user is on round 9, and they hold the button in while pressing the joystick left, the main theme from the super mario games will play from the speaker. We reworked the code at <https://create.arduino.cc/projecthub/jrance/super-mario-theme-song-w-piezo-buzzer-and-arduino-1cc2e4> to achieve this.

**Enclosure Design**

The encapsulation creation process proved to be more difficult than anticipated. Given Omega Group’s inexperience with 3D modeling and printing software, the modeling process utilized Fusion 360 per the recommendation of one of the posted Canvas tutorials. Said tutorial demonstrated extremely basic operations using Fusion 360 to create a semi-complex conduit structure, which taught how to offset layers of an object, create a sketch off of a plane of a pre-existing sketch, and creating holes for screws and I/O devices. With the conduit tutorial completed, the encapsulation was drafted to use similar skills used in the tutorial.

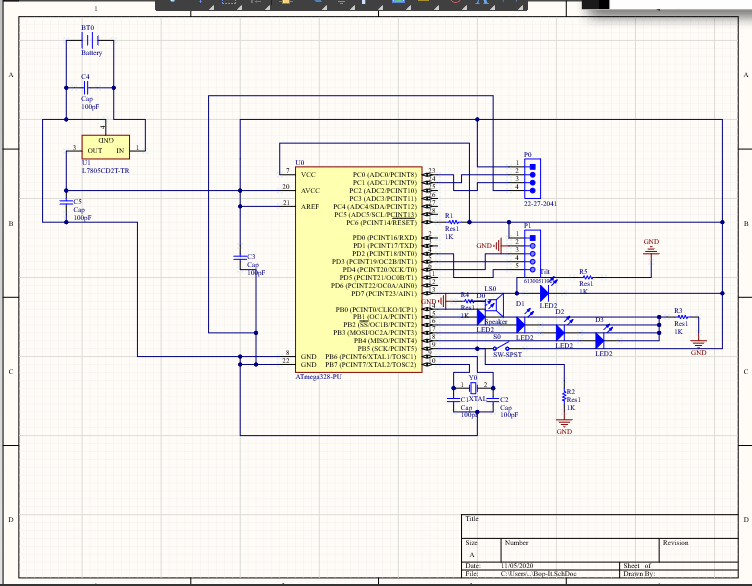
The intention for the encapsulation was to hold an ergonomic form, but after realizing that a higher level of mastery over Fusion 360 was required to do so, a simple box form was selected. Printing the encapsulation was outsourced to a peer and friend within the department, who offered to print the encapsulation so long as printing filament was provided. After sending the files drafted in Fusion 360 to said peer, he recommended some changes that would reduce print time without sacrificing the integrity of the structure. Notably, the thickness of the base and walls were reduced by over half of their original size.

The first print job of the encapsulation failed given due to unknown reasons, but it is suspected that the error had to do with the printer settings from a previous project. The second attempt at printing was successful, which produced the final product in seven hours. The box holds inside of it the PCB, speaker, and battery, while the I/O devices are mounted to the exterior. On the lid of the box, holes and spaces were included to allow wires from the PCB to connect to the components. This way, the messy wiring would be hidden from view, the user would only need to see the components of the device that they needed to interact with. The encapsulation is separated into a box component as well as a lid component, which are secured together with the use of screws. The components lying on the lid of the box are secured with adhesives such as tape, given its ease to both administer and take off.

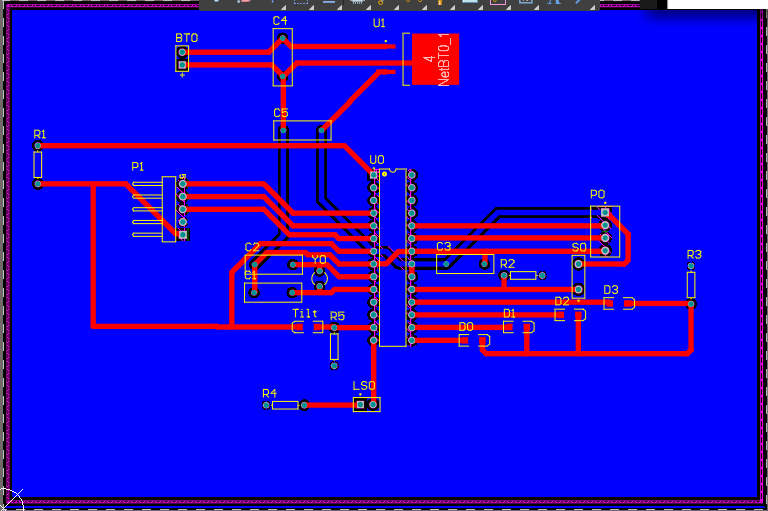
**Assembly and System Integration**

After our breadboard prototyping and before assembly, we created a schematic and a PCB layout of the design that we intended to implement. The integration of all hardware components is a critical point in any engineering project.

Schematic:

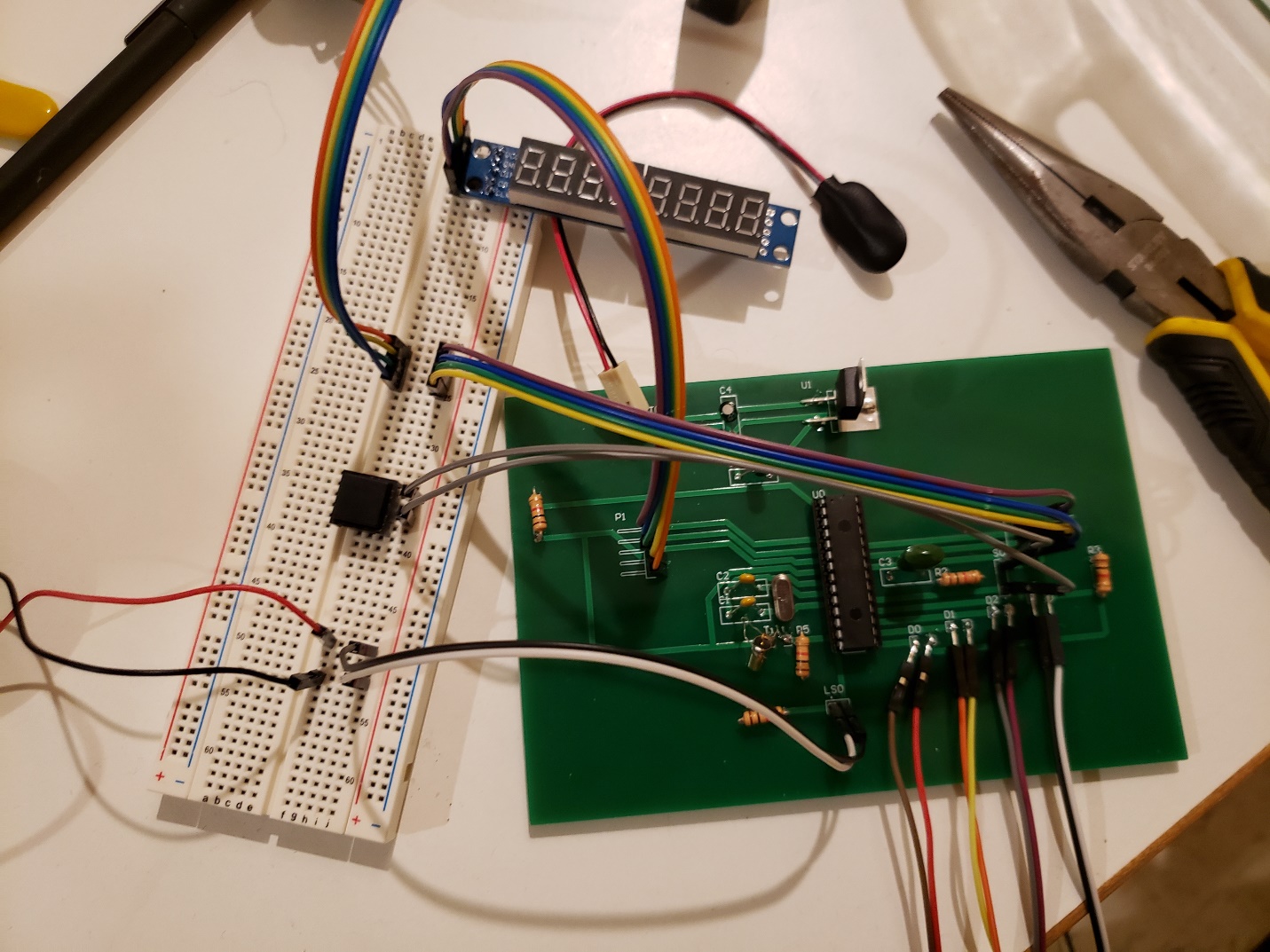


PCB Layout:



After our team analyzed the schematic and PCB layout in depth, we agreed upon submitting the above design to the manufacturer. Once we got a hold of the physical PCB, we immediately began soldering and debugging, as shown below.

Soldered PCB and Testing/Debugging:

Our initial testing took place without the voltage regulator circuit, we used a 5V battery to power the board. We did this so that we could isolate any potential problems. Many of the main components were handles off the PCB on a breadboard and we used a basic SPDT switch in place of our BIG RED BOP-IT BUTTON. All components were verified to be working so we moved to integrating this into our 3D printed enclosure and continue testing.

**Design Testing**

**Need to do.**

Road Trip, Let’s Play!



**Budget and Cost Analysis**

Below is a compiled list of the parts ordered and parts that we already had on hand. Parts on hand were estimated to cost ~$25 for the sake of our budget analysis.

|  |
| --- |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| -----------------------------------------------Parts Ordered---------------------------------------------------- |
|  |

|  |
| --- |
| SPARK FUN # |
| # Quantity Seller # Manufacturer # Description PPU Total | |
|  |
|  |

|  |
| --- |
| 1. 2 COM-09151 COM-09151 SPEAKER 8OHM 500MW 2.25 4.50 |
|  |

|  |
| --- |
| 2. 1 COM-09179 COM-09179 Pushbutton 33mm 1.95 1.95 |
|  |

|  |
| --- |
| 3. 1 COM-09181 COM-09181 Big Pushbutton 11.95 11.95 |
|  |

|  |
| --- |
| 4. 1 COM-16454 COM-16454 Piezo Siren 6.95 6.95 |
|  |

|  |
| --- |
| 5. 1 PRT-0009 PRT-0009 9V Snap Connector 1.25 1.25 |
|  |

|  |
| --- |
| 6. 1 PRT-09518 PRT-09518 9V Barrel Jack 2.95 2.95 |
|  |
|  |

|  |
| --- |
| 7. 1 PRT-10811 PRT-10811 DC Barrel Jack 0.95 0.95 |
|  |

|  |
| --- |
| 8. 4 PRT-10218 PRT-10218 9V Battery 1.95 7.80 |
|  |

|  |
| --- |
| 9. 1 SEN-10289 SEN-10289 Tilt Sensor 1.95 1.95 |
|  |

|  |
| --- |
| AMAZON # |
|  |

|  |
| --- |
| 10. 1 LS0058 LS0058 Joysticks 9.99 9.99 |
|  |

|  |
| --- |
| 11. 1 28036 28036 Quad Tilt Sensor 9.99 9.99 |
|  |

|  |
| --- |
| -----------------------------------------------Parts On Hand--------------------------------------------------- |
|  |

|  |
| --- |
|  |

|  |
| --- |
| 12. LEDS |
|  |

13. Extra Capacitors

14. Extra Resistor

15. Crystal Oscillator

16. Voltage Regulator

17. MAX7219 Hex Display

18. Rocker Switch

19. PTFE for 3D Printing

20. Atmega 328 IC 30.00

**Total Cost: $90.23**

Budget/Cost Analysis Conclusion:

The price of our Bop-It prototype was costly, but cost would significantly decrease after an agreed upon prototype was made. Many parts were not used because some we ordered in bulk and others were for our contingency plan. Discounting the parts not used we get $90.23-$28.94 = $61.29. This price would continue to be driven down based off bulk orders and the exclusion of unused parts.

**Team**

The responsibilities for the Bop-It project were equally delegated according to each teammate’s strengths and interests. As the project can be mainly broken up into three large sections (hardware, software, and enclosure), each teammate was largely responsible for one of those aspects. Gregory took care of many of the hardware assignments, such as creating the PCB schematic and layout, as well as selecting which components were needed for the bill of materials. Brian primarily oriented his attention on the software portions of the project, where he focused on the game mechanics such as counting the score, prompting the timer, and all the other core functions of Bop-It. Ben worked on the enclosure for majority of the project’s time frame, given the group’s unfamiliarity with 3D modeling and printing. He learned how to use Fusion 360, modeled the box for the device in accordance with the design requirements, and printed it once the hardware and software were integrated and finalized.

Team Omega’s tactic to complete the project could best be related to the divide-and-conquer algorithm. Individually, the group members completed their individual assignments with relative ease, without needing to ask other group members for help on how to execute their assignment. In terms of communication however, Team Omega frequently updated each other with everyone’s individual milestones, and asked the other members for their preferences regarding the operability and aesthetics of the design. Given each member was capable of meeting the expectations they set for themselves, the project rolled by smoothly with minimal issues with the design and even less issues (dare I say no issues) with teamwork and coordination.

**Timeline**

The project timeline was relatively linear. The initial stage of the project involved all group members brainstorming to consider which I/O peripherals to use. After general guidelines were set, a prototype was assembled and verified just as quickly as the planning phase lasted. Once Team Omega all signed off on the prototype, tasks were divided as all members started working on their own independent portions of the project: software refinement, PCB modeling, and enclosure modeling. Before the enclosure was completed but after the PCB was printed, the software and the hardware were combined together to test the functionality. Once the combination of the two proved to yield a functional game, everything came together at the end to store it in the enclosure.

**Summary, Conclusions and Future Work**

Team Omega’s project went very smoothly. We started with an idea, moved to breadboard prototyping, then designing a PCB and enclosure, and finally put it all together. We are very satisfied with our results, although we still have many more ideas that we would like to implement! If we continued working on this project in the future, we have a lot of minor fixes that we would make, and some major changes. As far as minor fixes, we have noise at the output speaker, and it is not as loud as we would have liked. This problem could be rectified through some simple capacitive filtering and BJT amplification. Also, we would like to make the hexadecimal display have a decimal point between the ones place and the tenths place; this was a minor software challenge. In addition to these easily rectifiable issues, we would like to get more familiar with 3D printing and make a more ergonomic design. And our last minor fix would be to output .wav files so that we could have a voice commanding the user. We really wanted to make an “angry bop-it” that yells commands at the user, unfortunately we could not get it to work. As far as major changes, we had all sorts of great bop-it ideas that we scrapped. We were interested in a component that allows you to complete a circuit with your finger, one that allows detects if your finger passes through a hole in the enclosure, and many others! This group learned a ton working on this project and worked together extremely efficiently, we are excited to see what other things we can created in the future.