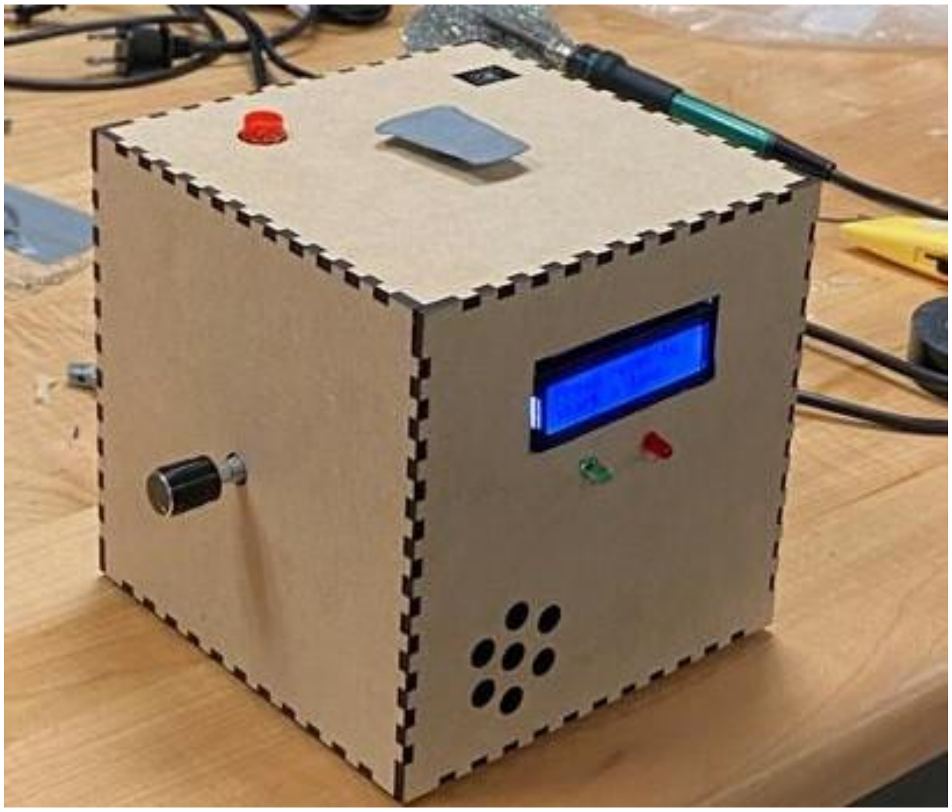
Dispose It

University of Pittsburgh

Spring 2022

Junior Design Fundamentals - ECE 1895

Dr. Samuel Dickerson



Rayan Hassan
Destiny Elings
Ashley Ajuz

<u>Design Overview</u>

Description and Purpose of Design:

Our proposal for Pitt Bop It is "Dispose It". It will be a Bop It that incorporates several waste disposal methods and will be used as a way to teach children/users in what ways certain types of waste should be disposed of. The three actions on Dispose It will be Trash It, Recycle It, and Compost it. Our proposal will have "Trash It" be a push button that has a trash can as the button's design. "Recycle It" will involve a twisting mechanism that has the design of a recycling symbol. "Compost It" will be a pulling motion with a leaf as its design.

Original Design Concepts:

Our group had originally wanted to design "Dispose It" as a 3D cube with three different sensors attached to its sides to be used for the directed actions.

- We envision that the front face of the cube would contain the speaker for audio cues and a screen that will let the user know what their current score is. We were undecided on whether or not to use a LCD screen or a 7-segment display, but we were leaning more towards using a 7-segment display so that is what we used in the first sketch of our enclosure.

- For the sensors, we had a concrete idea of what we wanted the first two sensors to be (a button for the Trash-It action and a rotary encoder for the Recycle-It) however we were stumped on what to use as the last sensor. After discussing some ideas, our team decided that we wanted to try and use a pulley that the user could pull out to complete the Compost-It action.

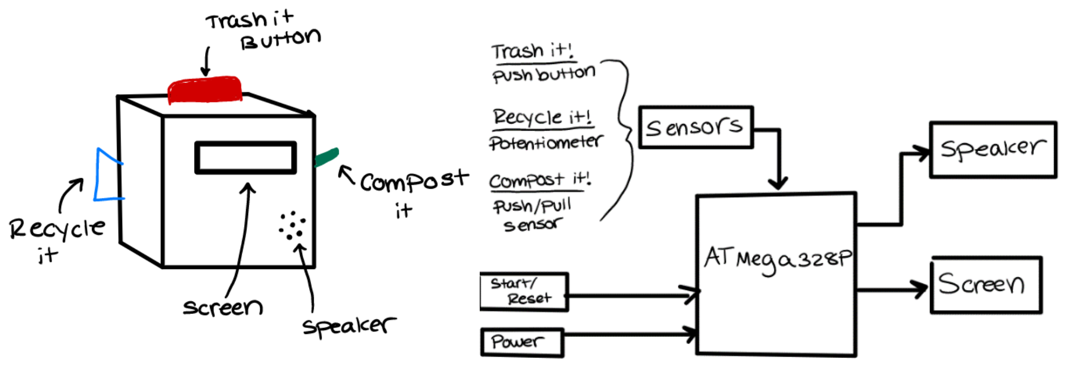


Figure 1
Sketch of design and Block diagram

<u>Example of how the original design would operate:</u>

Using the speaker on the front face of the enclosure, the toy will prompt users by saying a piece of waste, and the user selects the correct disposal method. For example, when the device says paper, the user would twist Recycle It. When the device says plastic bag, the user would push Trash It. When the device says banana peel, the user would pull Compost It.

Final Design:
While we did decide to stick with the 3D cube design from our original proposal, there were a few changes we made throughout the duration of the project. The major changes made are as follows:

1. **Utilizing a LCD screen instead of a 7-segment HEX display**
   The main reason we had wanted to use a HEX display before was because we were all more familiar with how it functioned in comparison to the LCD screen. We had also assumed that because we would only need to display the user's numerical score and no additional text, that it would be simpler to incorporate in our design and would not take up many pins on the ATMega. This did not end up being the case as we later learned that the HEX display would require a decoder that would end up using more digital pins than we had expected. We also realized that it would be beneficial for the user to not only know their score, but to also see the given waste object written out for them to read rather than solely relying on the audio to tell them the current object they need to dispose of. These two things are what led us to using a LCD screen in replacement.
2. **Changing sensors used for Compost-It action**
   After a tireless search on multiple websites and vendors, we were struggling to find a sensor that would act as the pulley we imagined. The closest thing we found was a flex sensor that when bended a certain amount would trigger a change in the circuit. We kept that sensor as an option, but we were still hoping to find a way to get the sensor we wanted so turned to the TA's for some advice. However, they too were unsure of a sensor that would accomplish what we wanted so they instead suggested that we constructed one ourselves out of the sensors that were available. We did consider this option, but due to concerns about time constraints we ultimately decided to scrap this idea and use a new sensor (the force-sensitive resistor) for the Compost-It action. Now if the user needed to compost something, they would just press down on the force sensitive resistor pad on the side of the enclosure.
3. **Redesigning audio cues**
   Originally we wanted to save audio segments of the items the user would have to dispose of. So for example, if the next item that needed to be disposed of was a pizza box, the audio would say "pizza box" and the user would pick the correct disposal method. However, we ran into issues storing and using the equipment needed to store audio bytes (we will discuss this further in the Software Implementation of this report) so we changed the audio cues to be simple sound effects.

Citations and References:

Boxes.py, https://www.festi.info/boxes.py/.

Fusion 360 Tutorial for Absolute Beginners— Part 1 - Youtube.
https://www.youtube.com/watch?v=A5bc9c3S12g.

"Interface I2C 16x2 LCD with Arduino Uno (Just 4 Wires)." *Arduino Project Hub*, https://create.arduino.cc/projecthub/akshayjoseph666/interface-i2c-16x2-lcd-with-arduino-uno-just-4-wires-273b24.

"What Can We Help with?" *GitHub Support*, https://support.github.com/.

<u>Design Verification</u>

Initial Prototype:

Enclosure:

As discussed in the Design Overview section, the first rendition of the prototype resembled the original sketch as closely as possible. An image of the first modeled prototype is shown below

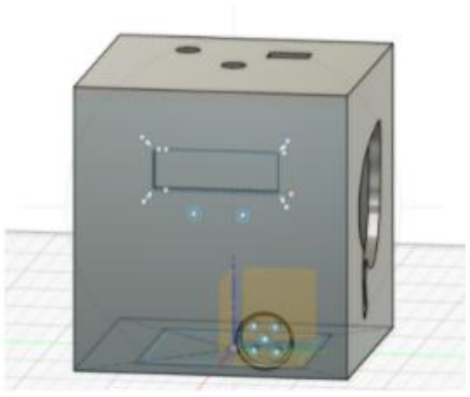


Figure 2

Software and hardware:

We prototyped the hardware and the software together because they are closely associated in microcontrollers. We were lucky enough to acquire the majority of our components from the stock of components we had access to, and from the first order, so we were able to prototype quickly. We tested each component of the design separately to ensure functionality, and to ensure we knew how to interface them with the ATmega328p. To test each component, we referenced existing Arduino projects to see how they used the components and got them to function correctly. The components we had to test in the breadboard prototype were a pushbutton, an LCD screen, a rotary encoder, a force sensitive resistor, and our mini mp3 player module.

A pushbutton was used in our previous assignment using the ATmega328p, so rigorous testing of the pushbutton was unnecessary.

Prior to this project, none of us had worked with programming an LCD so it was important to test this early in the design process. This was the first component we tested, and we referenced the following link, LCD reference. Modifying this code to fit our design, worked successfully. We utilized the LiquidCrystal_I2C library to print to the LCD screen.

Testing the rotary encoder presented many difficulties for us because many reference projects using a rotary encoder looked different from our rotary encoder and had different pins. After referencing many sources, we discovered a useful reference project with a component similar to ours. It detected changes in the phase of the encoder, and we prototyped this in our project by outputting a boolean high value for any change in phase, with no preference for how much the rotary encoder changed direction.

We also had not worked with force sensitive resistors prior to this project. It was our last component to arrive so we were worried about successfully integrating it into our Bop It. We referenced some Arduino projects that used the same FSR as we had ordered so that we could have an idea of how it functions for the PCB design. Once it arrived, we used the same references for ideas on how to test it. We tested that it could detect a squeeze of varying intensity. For our design, we decided that anything above a light squeeze should be detected as action from the user.

We had the most difficulty with the mini mp3 player module. It was our last component to arrive, so we had less time to test it rigorously. We used the sample code from its documentation and the tests were unsuccessful. Due to time constraints, we were not able to discover why the mini mp3 player did not work. To meet the deadline of the project, we decided to play sound through the speaker via outputting frequency values. It also proved a better decision because our design did not incorporate high-quality audio, so the addition of an extra component was not necessary, and its omission reduces the design's cost and complexity.

Testing the hardware and software concurrently was a successful testing scheme because with every successful component test, we had assurance it would be successful when the entire game was assembled.

After testing each component individually, we began to design the gameplay loop that would guide the user through the game. This required a lot of debugging, which will be discussed further in the "Design Testing" section, but we were confident that each component was interfaced correctly, leaving only logic errors in the game design, rather than logic and component-interfacing issues.

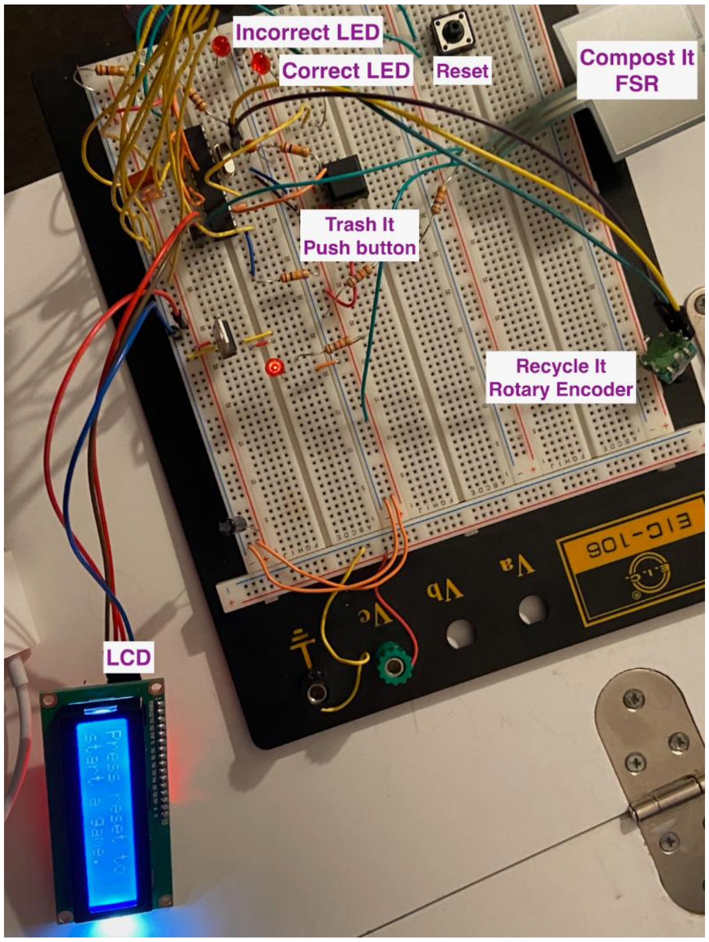


Figure 3

Electronic Design and Implementation

PCB schematic

The PCB schematics changed throughout our progress. The original one is shown below.

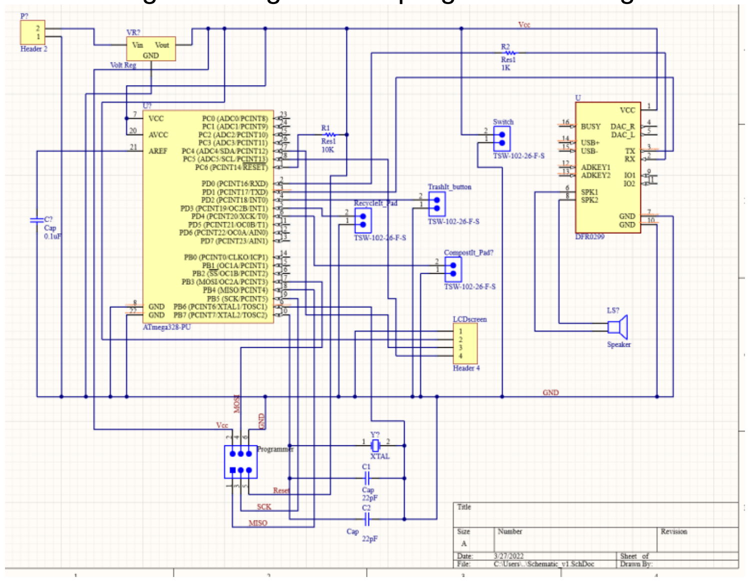


Figure 4
Original PCB schematics

We used a 2 pin header for the battery holder, and a 4 pin header for the LCD connecting it to ground, VCC, SDA and SCL pins from the ATmega238P chip. For the programmer, we just needed a 6 pin header and annotated each one of the pins to make sure they are connected to the right ones (VCC, GND, MOSI, MISO, SCK, Reset). The voltage regulator is also taken from the "components" section, connecting to VCC, ground and the rest of the circuit. For the speaker, we originally intended to use an Arduino MP3 player (DFR0299), which we ended up not using as we explain later. However, we weren't able to find it in the default libraries. Therefore, we imported and downloaded it from SnapEDA. The speaker's terminals were connected to SPK1 and SPK2 of the DFR0299 chip. Pins 7 and 10 connected to ground, and TX and RX respectively go to TXD and RXD pins of the ATmega238P chip. For all the commands (RecycleIt, TrashIt, CompostIt and the ON/OFF switch), we were going for 2 pin headers which we found in the "Manufacturer Part Search".

After running the first Schematics validation, we encountered many errors. The most important ones were the TSW-102-26-F-S related ones (for the switch and the buttons). We got rid of those and replaced them with 2 pin headers from the "components" section, which were able to be found and compiled. Also, after a careful review of the circuit, we noticed several connection errors. The oscilloscope was in parallel with capacitors C1 and C2, which wasn't correct. We fixed that, as well as the AREF pin from the ATmega238P which should have also connected to VCC. Another encountered error was "Contains IO pin and power pin objects" or "output pin objects". For that, we made sure that all connections were possible using the Connection Matrix. Finally, we added generic No ERC to close all the unused pins from the chips. Here is the updated schematics.

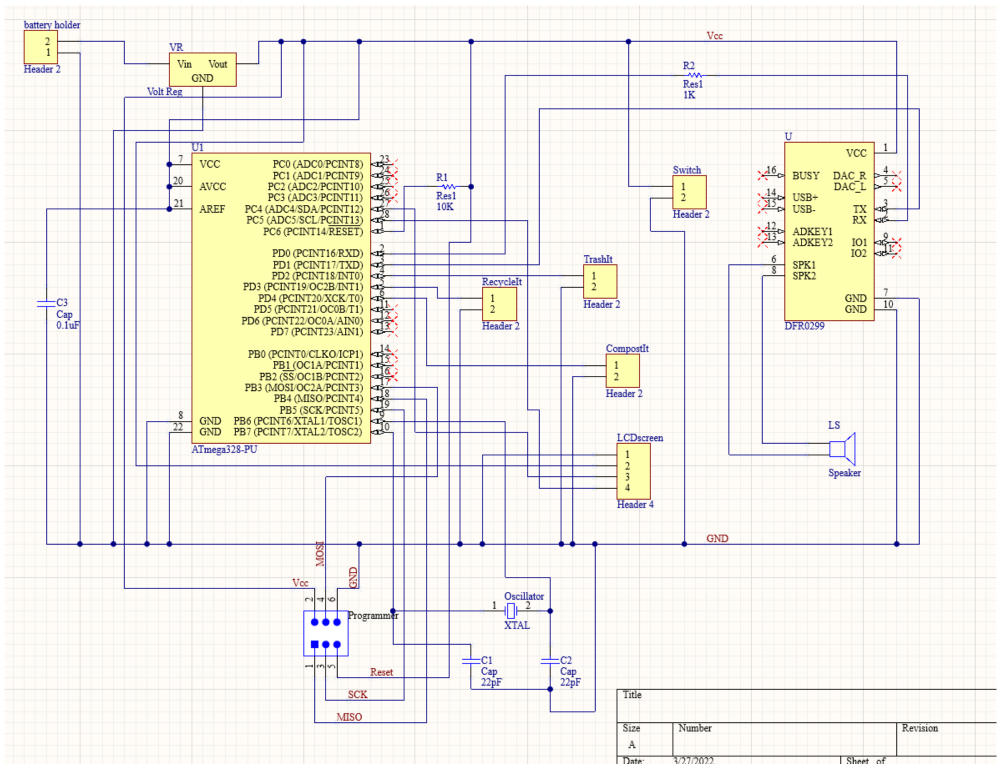


Figure 5

Updated PCB

Schematics

Although this version looked better, it still needed a lot of work. In fact, after testing all the components on the breadboard and verifying all the connections, we figured that the buttons and the switch should be connected differently. Also, we wanted to use a rotary encoder for the Recycle It command, which required a 3 pin header instead of a 2 pin one. We followed that particular scheme shown below to connect all these commands.

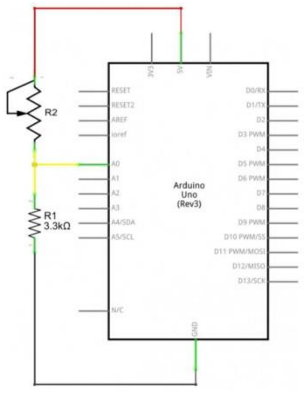


Figure 6

Connection for header pins

The rotary encoder (RecycleIt) should be connected to 2 digital pins (PD5 and PD6) and ground. The sensitive resistor's (CompostIt) first pin to VCC and second pin to an analog pin (PC3) and ground via a 3.3k resistor (R5). The same connections were needed for the push button corresponding to the reset button and TrashIt, but those connected to digital pins (PB1 and PD2 respectively). Finally, we figured we also needed LEDs to indicate whether the answer is correct or wrong. So we connected 2 LEDs to digital pins PD3 and PD7 respectively, with small resistors. The final PCB schematics is shown below.

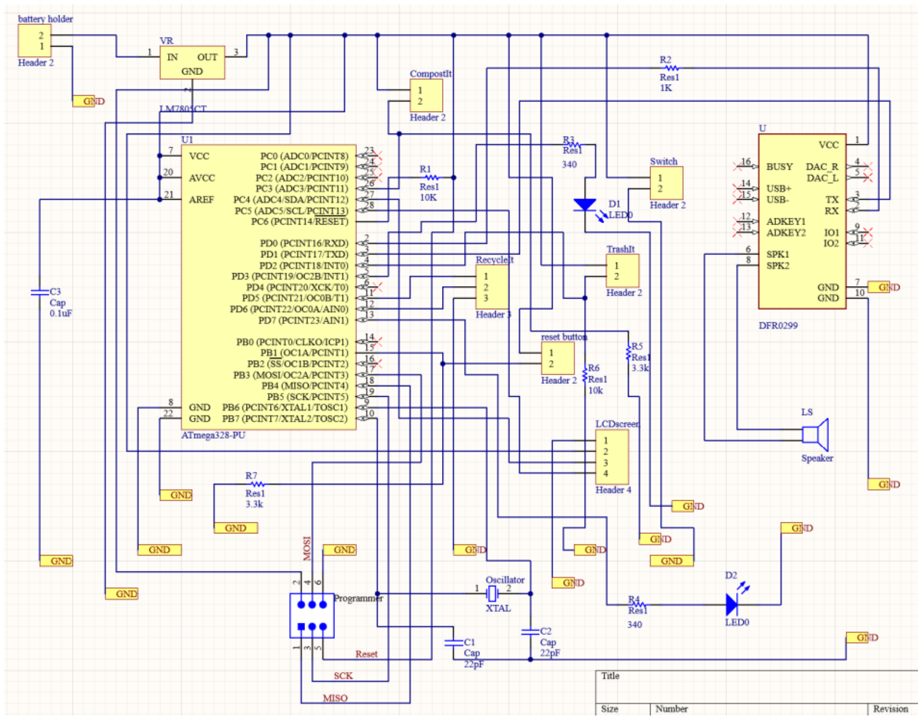


Figure 7

Final PCB Schematics

PCB layout

For the PCB design, we followed multiple practices that made it easier for us to add traces and visualize the entire board. We started by placing all edge components first, mainly the battery holder's pin header, the voltage regulator and the programmer. This was followed by the two chips. The first traces added were the ones we thought created the most hassle. Mainly, the programmer's, the LCD screen's and the rotary encoder's traces were added first. After that, it made it easier to add the other ones. In general, we tried to keep similar components in the same direction, like the capacitors, the LED, the header pins and the chips. Additionally, we tried to follow the same design as the schematics' in certain parts, like the programmer's position, the oscilloscope's connections, the voltage regulator, the chips and some of the other header pins. This made things easier to visualize and conceptualize the final board. We tried to keep most components on the same layer, mainly the bottom one, even though it didn't really matter since with these new professionally manufactured boards, the solder was going to stick on both layers anyway. All the components used were through holes. The final dimensions of the board were 105.3mmX87mm, making sure it fit inside the enclosure. Additionally, we added 4 holes in each corner of the board with a hole size of 2.5mm, in order for us to drill the PCB onto the final box during the enclosure. The only difficulties were adding the traces since we used several pins from both the chips, which made it harder to connect everything together, even with all the practices used. Therefore, we had to rotate each component and figure out which direction worked best with the rest of the circuit. Finally, we added the board outline in the keep-out layer. Here is what our final PCB layout looked like.

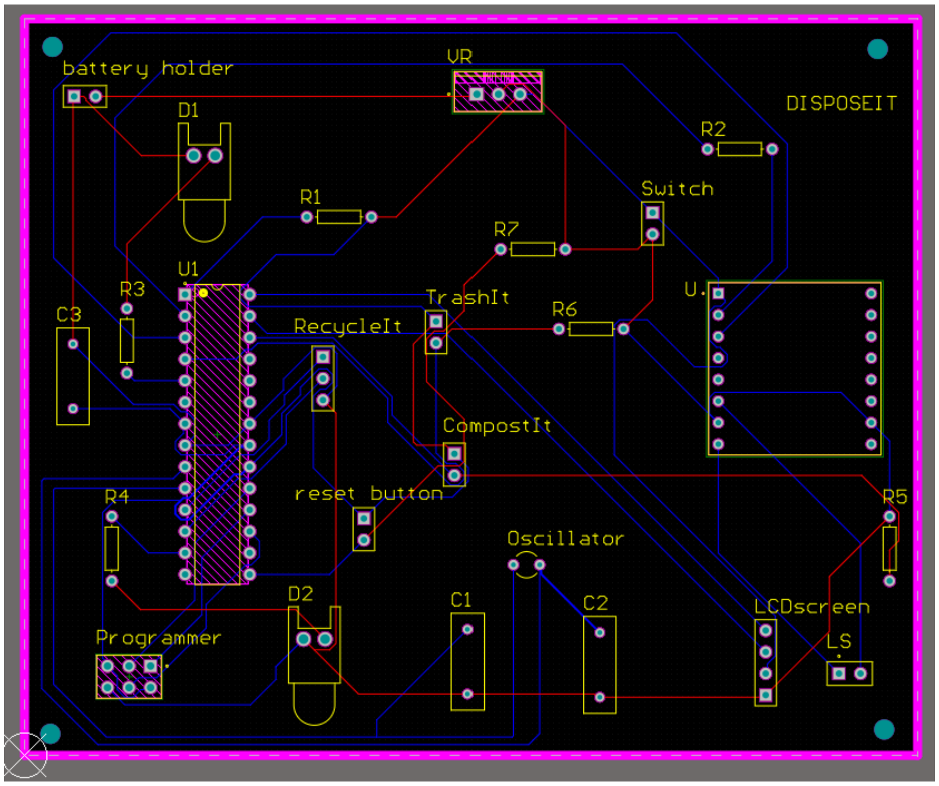
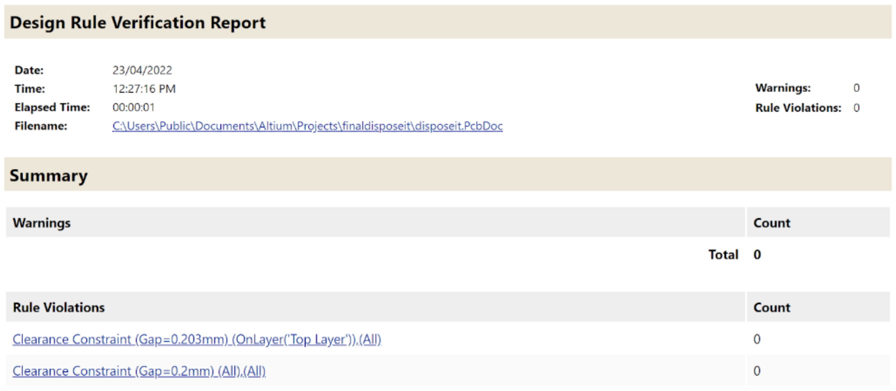


Figure 8

Final PCB layout

Finally, we imported the specified design rules. After running the Design Rule Check, we didn't receive any error, as shown in the screenshot below.



Design Rule Verification Report

Date:	23/04/2022
Time:	12:27:16 PM
Elapsed Time:	00:00:01
Filename:	C:\Users\Public\Documents\Altium\Projects\finaldisposeit\disposeit.PcbDoc

Warnings:	0
Rule Violations:	0

Summary

Warnings	**Count**
Total	**0**

Rule Violations	**Count**
Clearance Constraint (Gap=0.203mm) (OnLayer('Top Layer')),(All)	0
Clearance Constraint (Gap=0.2mm) (All),(All)	0

Figure 9
Design Rule Verification Report

Later, we checked our design in OSH park, and here is the preview of our board

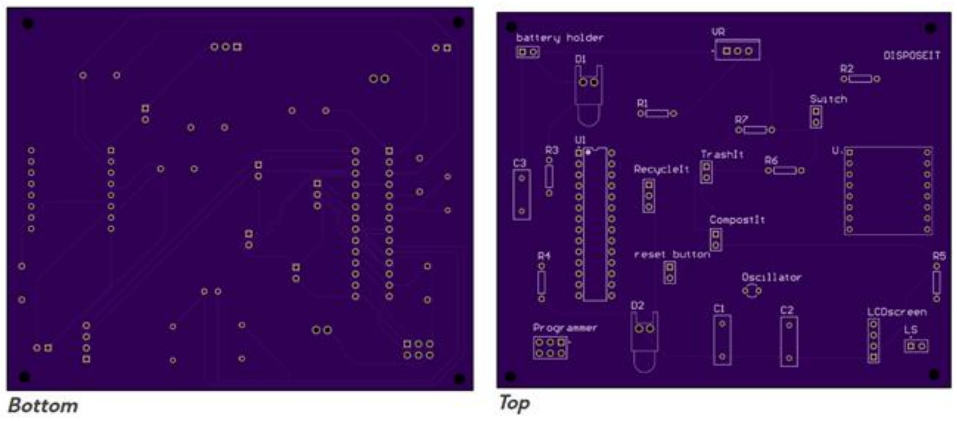


Figure 10
Preview of OSH Park

<u>Software Implementation</u>

Overview of code structure:

The software aspect of Dispose-It is a modular design dividing the different aspects of the game into classes and utilizing the main loop to integrate them. Bop-It has distinct, independent functions that can be divided and modularized such as: audio, the various inputs, a timer, game commands, and the game functionality (start, end, reset, score keeping).

For Dispose-It, each of these functions are handled separately.

Audio

For audio, we selected 4 sound effects to implement. The audio class has a function for each of the sound effects and it outputs sounds on the speaker pin by playing a tone on a pin for a fixed amount of time (which is determined by how long we delay the tone before changing the output on the speaker pin). To ease the process of playing sounds directly onto the speaker pin via frequency values, we used a “pitches” library. The pitches library allowed us to specify the specific note to play on the speaker rather than the exact frequency value.

Commands

The commands class is responsible for generating commands, checking if the user completed the correct action for a command, and telling the user the correct answer to a command if they got an answer wrong. We used an array to store our commands and their answers. We generated a random number to randomize command generation, and we also made it so the same command is not generated twice in a row.

Game Functions

The game functions class handles scorekeeping, telling the user they got an answer correct or incorrect, ending a game, starting a game, and restarting a game. There are several scenarios in which a game would end such as: user reaches max score of 99, user answers incorrectly, and user does not answer in time. We use if statements to check why the game is being ended and prompt the user via the LCD screen, speaker, and external LEDS accordingly.

Input Classes

The three input classes compostIt, trashIt, and recycleIt behave similarly in that they check for action on their corresponding inputs, and return a boolean value TRUE if it registered user input. The classes differ in that each of them utilizes a different input sensor. CompostIt is a force sensitive resistor. The resistor is attached to an analog pin is polled for changes in pressure. TrashIt is a push button. The button is attached to a digital pin. We poll the pin for a button press, and also debounce the button to eliminate button presses from mechanical malfunctions. Finally, recycleIt is implemented with a rotary encoder. The rotary encoder is polled for any change in the position of the knob, and is also attached to a digital pin.

Timer

The timer is implemented by using Arduino's millis() function. We selected a starting time limit and end time limit, and calculated how much to decrement the interval by based on the max score of 99 (which means the user could play ~100 rounds). The millis() function "returns the number of milliseconds passed since the Arduino board began running the current program". Every time a command is generated we check the time with the millis function. We then keep track of the time passed since we last checked the time with the millis() function and if the difference of the current millis check and the millis check at command generation is greater than the current time limit, the timer has expired. We return a boolean HIGH on the timer expiration. Prior to generating a new command, the timer variables are reset.

Main

The functionality of the game and integration of the classes occurs in the main file. There is a gameplay function, a function for the reset button, the setup function, and the loop function. The loop function directs the game from a high level. It checks if a game is active, and if so, it goes to the gameplay function. If there is no game active, it will wait for the user to press the reset button to start a game. It is also responsible for calling the gameFunctions.restartGame() if the user presses reset in the middle of a game. The gameplay function calls the function to generate a command, and waits for either user input, timer expiration, or reset game request. When one of these occurs, the gameplay function responds accordingly. The setup function initializes the mode of all the pins. The reset function is responsible for checking if the user pressed reset, and also denouncing the input.

Software engineering design practices used:

Modular Design

We used a modular software design to implement our Bop It. The separate functions of the Bop It were separated into classes, so that they could be implemented individually, tested individually, and make the code easier to read, debug, and develop.

Code planning:

We used flowcharts and diagrams to define the interactions between classes, and to map out the flow of the game. It was important to create these diagrams so that we had an idea of the game functionality, and so there would be fewer logic errors.

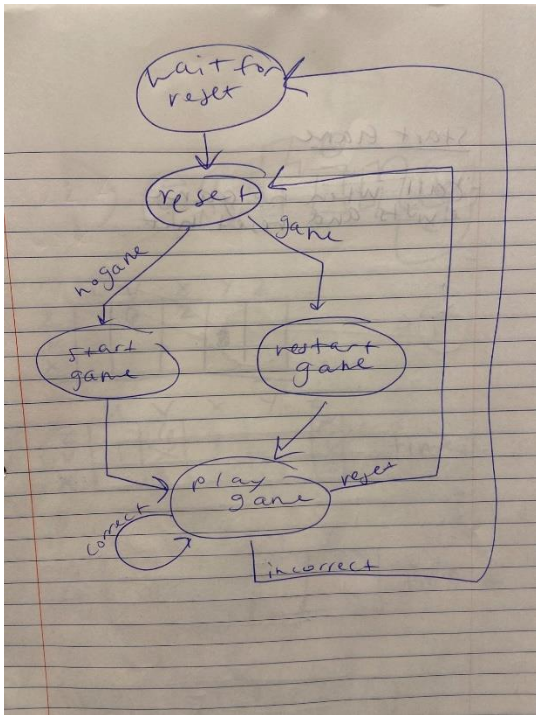


Figure 11
Flowchart for reset button

Challenges Encountered:

1. A software challenge we encountered was logic errors. Many times we did not properly map out the logic for a function, or aspect of the game, due to misunderstanding the functionality, and the response on the board was unexpected. For example, when implementing the reset button and game ending functionality, we did not account for the different scenarios in which a game could end and be reset. A game can be reset in the middle of a game, the reset button is pressed to start a game, and a game ends upon timer expiration. All of these prompt the user differently. If the user presses the reset button to start a game, the user does not need to see their score, or see output on the incorrect LED. Alternatively, if the user presses the reset in the middle of a game, they do not need to see the "Press reset to start a game." command and the game can start immediately. It took us a long time to iron out all the different scenarios under which user input can occur.

2. Another challenge we encountered was rogue input on the rotary encoder. It should be noted that the components used on our Bop-It are not industrial grade and they experienced some wear. At some point during our development of our Bop-It, our rotary encoder displayed signs of wear by getting stuck in a state of constantly triggering input on its pin. This was interesting for us to debug because we initially assumed the logic of the game was wrong. The second a command was generated, the game would end because it registered an action on the rotary encoder. We realized it was the rotary encoder behaving erratically when there were correct actions registered on recyclable items. To fix this, we twisted it and

pressed it until it stopped producing false actions. This happened several more times during the development of our Bop-It, but the other times we knew the software was not at fault, but rather a component was displaying symptoms of wear.

Classes and subroutines implemented:

- **Audio**
  - void everyTenPoints()
    - Played every 10 points
  - void gameWon()
    - Played when user reaches score = 99
  - void trombone()
    - Played when user loses
  - void everyPoint()
    - Played every correct answer when score is not multiple of 10 or equal to 99
- **Command**
  - void generateCommand(LiquidCrystal_I2C lcd)
    - Generates a command, prints it to LCD
  - boolean checkCommand(int actionTaken)
    - Check correctness of user's action
      - return high if true, low if false
  - char* getCorrectString()
    - Returns string of correct answer
- **Compost**
  - boolean action()
    - Returns true if user interacted with the FSR
- **GameFunctions**
  - void correctAction(LiquidCrystal_I2C lcd, int action)
    - The user answered correctly if this is called
    - Tells user they answered correctly
    - Increments score
    - Lights up correct answer LED
  - boolean endGame(LiquidCrystal_I2C lcd, char* correctAnswer, boolean timeUp, boolean maxScore)
    - If this is called the user:
      - Ran out of time
      - Answered incorrectly
      - Reached the max score of 99
    - Tells user why the game ended
    - Tells them the correct answer to the command if they answered incorrectly or if they ran out of time
      - Also lights up incorrect LED in both of these cases
    - Lights up correct LED if max score reached
    - Plays sound corresponding to case that called subroutine
  - void restartGame(LiquidCrystal_I2C lcd)
  - void incrementScore()
  - void checkScore()
  - int getScore()
- **RecycleIt**
  - boolean action()

        - Returns true if user interacted with the rotary encoder
- **TrashIt**
    - boolean action()
        - Returns true if user interacted with the push button
- **Timer**
    - boolean timeLimitCheck()
        - Check time since command generation
        - Return true if time since command generation is greater than time limit
    - void resetTimer()
        - Resets timer by using millis() to get time the program has been running at command generation
    - void speedUpGame(int score)
        - Decrease the time limit based on the user's progress in the game
    - void resetInterval()
        - Set the time limit to the maximum value
- **Main File**
    - void gameplay()
        - Generates command
        - Polls for actions
        - Reacts to user action or timer expiration
    - boolean checkReset()
        - Polls reset button
        - Returns high if pressed
    - void loop()
        - Checks reset
        - Checks if there is a game active
        - Calls gameplay if game is active
        - Does nothing if no game
            - Waits for reset selection to start game

<u>Enclosure Design</u>

Explanation on final enclosure design:

Fortunately, we had been happy with our initial sketch of our enclosure from the project proposal, so our ultimate goal was to construct an enclosure to match the sketch as much as possible. We had decided to stick with the cube design since we had never manufactured an enclosure like this before so we wanted to keep the design as simple as possible. Here we will discuss the entire design process and major milestones reached that allowed us to produce the final version of our enclosure.

Initial steps of design

1. <u>Modeling the shape of the enclosure</u>

    Prior to this course, we had very little experience working with Fusion 360 and CAD modeling tools. Using Fusion 360 did have a bit of a learning curve, however thanks to the various tutorials listed online, we were able to make a hollowed box that became the basis/foundation of our design. The next step was to adjust the size of the box to height and width we were comfortable with. In the beginning, the box was set to be about 3.5 inches all around *(refer to figure 12 for reference)*, however after designing the PCB, we realized we needed more room for the board to fit comfortable so we increased the size of the box to about 6 inches, *(refer to figure 13 for reference)*

Figure 12

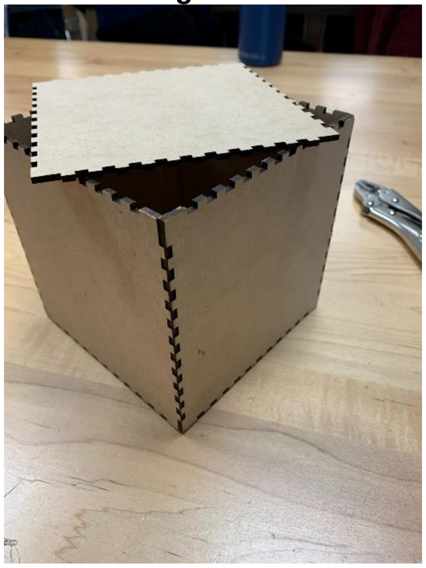


Figure 13

2. <u>Acquiring Measurements</u>

After the body of the enclosure was created, it was time to start adding cutouts to the faces so that the users would have access to the sensors to complete the required actions. We quickly realized that the measurements for all the cutouts would need to be exact in order for the components to fit correctly. In order to do this, we referred to the datasheets for each part and sized each measurement up by 2 mm to account for scaling issues that might occur during manufacturing. There were some issues that occurred with these measurements that we will talk about in more detail in the manufacturing process portion of this section, but this was a general process used to further customize the enclosure.

Makeup and structure

Front face

The front wall of the cube will be what the user will utilize the most. It contains a slot for the LCD screen along with 4 small holes in the corner to be used to drill the LCD screen flush with the wall. Beneath the LCD screen are two more holes used LEDs that will indicate to the user if their response to the prompt was correct. And finally, towards the bottom of this wall is a slot for the speaker with ventilation holes so that audio could be clearly heard.

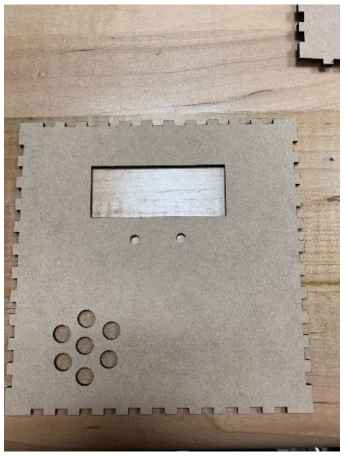


Figure 14

Top face

The top face contains 3 components; 3 buttons and one power switch. The button in the center will house the trash it button that the user will interact with the most. The leftmost button will be used as the reset button that the user can press to restart the game. And lastly, the rightmost cutout will be the power switch button to turn on and off the device.

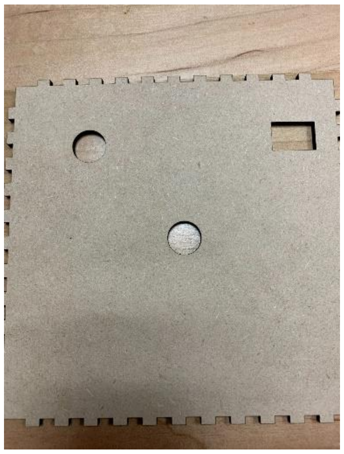


Figure 15

Recycle-It face

This side of the enclosure only contains one hole meant for the rotary encoder that is used for the Recycle it action.

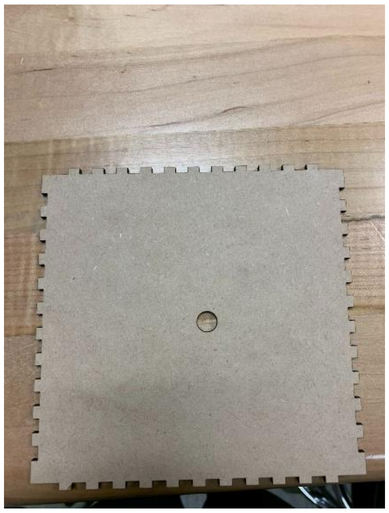


Figure 16

Compost-It face

After our decision to change the sensor used for the compost-it action to a force sensitive resistor, we decided to make this side of the enclosure be a cutout of a leaf (the symbol used for composting) where the force sensitive resistor pad will sit.

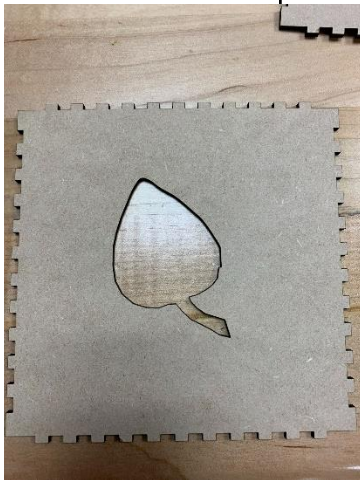


Figure 17

Back face

And lastly, this side of the enclosure houses a lid meant to give the user access to the battery that is powering the enclosure.

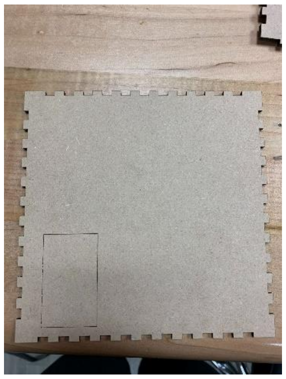


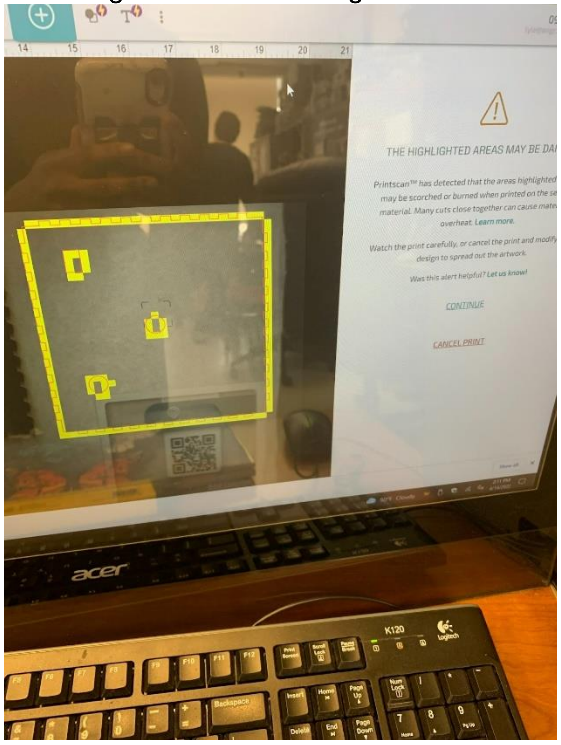
Figure 18

Manufacturing process:

After finalizing the Fusion 360 model of the design, it was time to start printing. After learning how to use the machine at SERC, we got started on our first print by using an unmodified file from Boxes.py and using medium draft board as the material we would be printing with. The print turned out very nice and was put together relatively quickly with the help of the TA's and a hammer. Our original plan was to take this first print and make the necessary cut outs ourselves, however we realized that would be difficult considering that certain cutouts had to be precise shapes (for example the leaf on the Compost-it face) and not just holes. We then started to work on translating the sketches from our fusion 360 model to the svg file required for laser cutting.

<u>Using Inkscape</u>

In order to add these edits to the boxes.py template, we utilized Inkscape which was a user-friendly platform that allowed us to add many different types of shapes and drawings to our svg *(refer to figure 19 for visual)*. After saving the file and uploading it to glowforge, we ran into our first major roadblock with the enclosure- the file wasn't readable by the software. Even after resaving and reuploading the design into the computer multiple times, we kept getting the same error message saying that our design would be damaged if we went ahead with the print.



THE HIGHLIGHTED AREAS MAY BE DA

Printscan™ has detected that the areas highlighted may be scorched or burned when printed on the se material. Many cuts close together can cause mate overheat. Learn more.

Watch the print carefully, or cancel the print and modify design to spread out the artwork.

Was this alert helpful? Let us know!

Figure 19

We decided to just try and print despite the warning messages, however as soon as the machine started we noticed the machine was sparking into flames a lot more than normal so we canceled the print to avoid causing any permanent damage to the laser cutter. We inspected the cuts that were made prior to cancellation to see if they were usable, but the lines were not as precise and were severely burned (image shown below). After consulting the internet to see if other people had a similar issue, we realized our problem was that the svg was being saved as an inkscape svg instead of a plain svg so glowforge was unable to properly read it. Once we re-exported the file as a plain svg, we no longer got any warnings and were finally able to print all the faces.

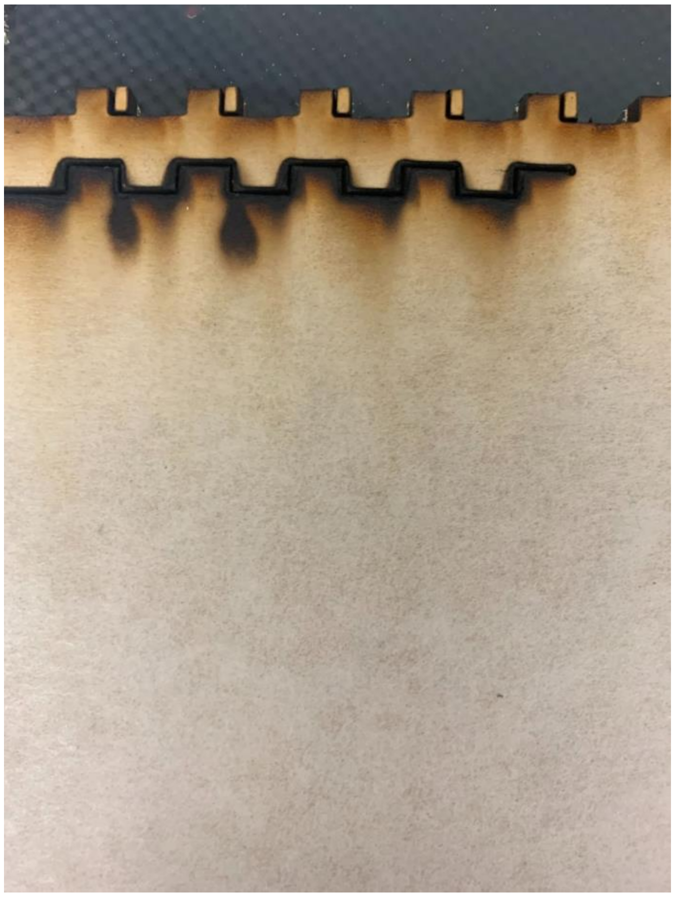


Figure 20

Prints

After the minor setback with the inkscape, we were anxious to get our second set of prints. It wasn't long before the walls of the enclosure were printed and we were finally ready to start assembling. Yet again, however, we were stopped after realizing that our components were not properly fitting in the cutouts.

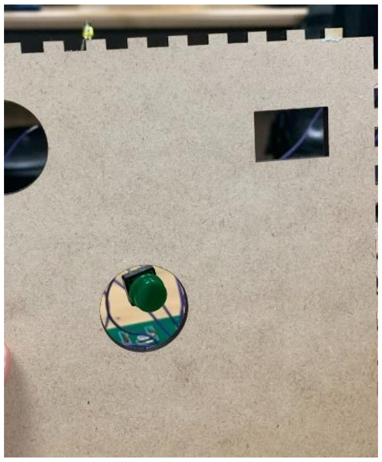


Figure 21

Despite using direct measurements from the datasheets of our components, there were still some components that were too large or small for their slot so, we concluded that there must have been a problem translating these measurements from Fusion 360 over to inkscape. After many more trials and errors (it took approximately 4 more reprints since we also ran into the issue of the material being hard to cut through which led to more reprints) we finally got the right sized cutouts for all 6 walls of the enclosure and were ready to start assembling.

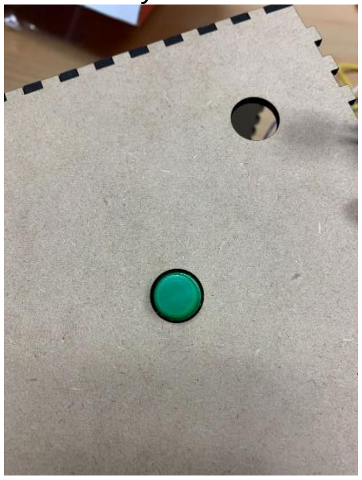


Figure 22

Assembly and System Integration

Enclosure:

Methods used for assembly:

The main method employed for constructing the enclosure was by using a hammer/wrench to hammer the walls together so that they fit in with each other snuggly and wouldn't come apart easily.

Figure 23

Image is from our very first print but the same method was used for the final rendition

To join the enclosure and PCB together, we used a mix of hot glue (mainly to stick the components to their designated slots) and a drill (for components like the LCD and PCB so they would be secure in place) Below is an image of the final product.

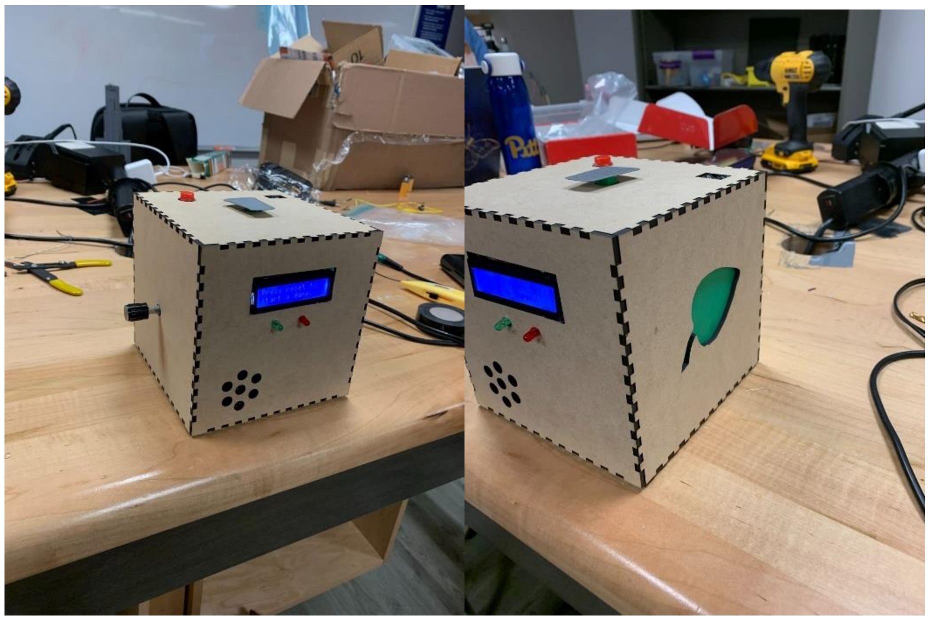


Figure 24

<u>Challenges Encountered:</u>

1. The major issue we ran into during assembly was trying to fit the PCB into the enclosure. Because the components had already been soldered to the board, it was difficult trying to fit some pieces through their slots (some of the cutouts were designed so that the component would stick out on top and wouldn't be able to fit all the way through their holes).
2. Another thing that presented an issue was figuring out how to prop up the force sensitive resistor in the leaf cutout. We wanted to make sure that the user's input would still be registered by the sensor while, at the same time, concealing the actual component.
3. The final challenge we encountered was placement of the battery. Based on the project specification, we are supposed to have all things that are not user inputs contained in the enclosure. However, we wanted to give users the option to access the battery in the event that the current one is faulty and needs to be switched out.

<u>Modifications/Solutions made:</u>

1. To resolve this problem, we simply unsoldered the wires for the bigger components, stuck them to the enclosure using glue, and then re-soldered them back onto their respective wires.
2. We decided to try and "sandwich" the force sensitive resistor *(labeled 2 in Figure 25)* between a piece of green construction paper *(labeled 3 in Figure 25)* and a piece of cardboard *(labeled 1 in Figure 25)*. This allowed us to cover the actual resistor and keep it propped up in its slot without compromising its function.

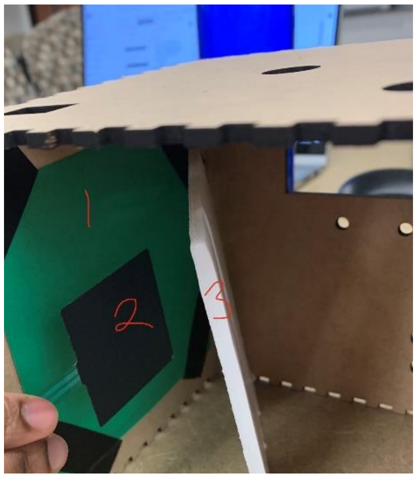


Figure 25

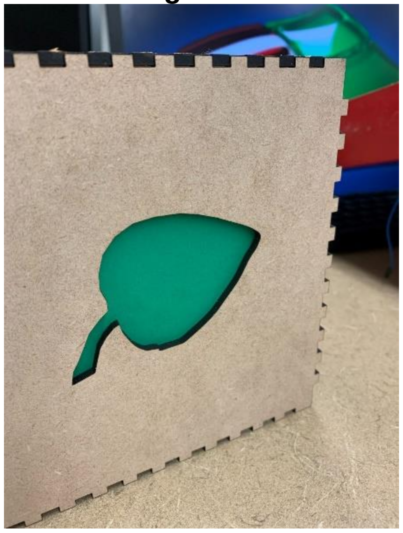


Figure 26

Outside view of Compost-it side

3. After some deliberation, we decided to attach the battery and battery holder to the back wall of the enclosure with a cutout that the user could open and close to get access to the battery. The cutout was glued down on one side but left free on the other side so that the flap would act as a latch that would remain attached to the enclosure (so that it wouldn't get lost) while still having the ability to open and close.

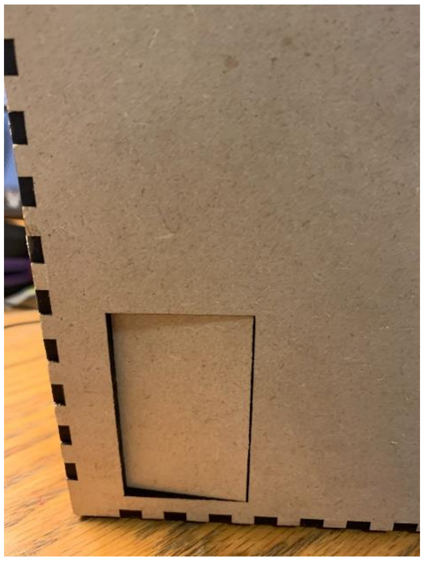


Figure 27

Hardware:

<u>Methods used for assembly:</u>

1. First, we soldered all the components that weren't going to pop out of the enclosure. This included all the resistors, capacitors, and the oscilloscope. Also, we used header pins instead of jumper wires for the programmer and the battery holder, which made it easier to connect the battery later on, and upload the code to our chip as we proceeded through the testing step. Additionally, we used a header pin for the chip as we were going to take it off at multiple instances to upload an updated version of the code.
2. At that time we realized that the MP3 player chip was unnecessary, after testing the speaker. Therefore, we didn't have to solder it on the board.
3. After that, we started placing the wires that allowed all the other components to reach the front face of the enclosure. This permitted us to glue them onto its surface. The components were the LCD screen, the rotary encoder, the force sensitive resistor, the push buttons and the LEDs, which were later soldered onto the wires.

<u>Challenges Encountered:</u>

1. The main difficulty was the soldering itself, as some of the traces were very close to each other. However, the solder was sticking easily to the holes so it wasn't as hard as we thought.
2. The challenge here was to figure out what pins were still needed on the DFR0299 chip in order for the circuit to work.
3. We encountered one primary problem during that step, which consisted of the components getting damaged as we tried to solder them on the wires. In fact, one of the force sensitive resistor's pins got ripped, as shown in the following picture. Also, it was hard to manipulate the board because of how many wires were sticking out of it, especially during the gluing part.

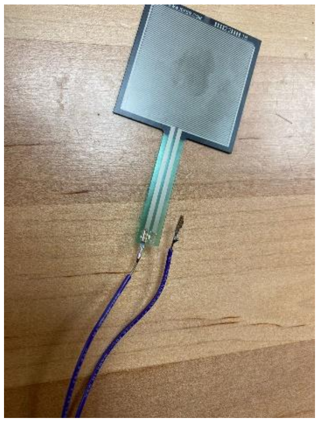


Figure 28
Force sensitive resistor damaged

<u>Modifications made:</u>

1. We just had to be extra careful and patient while soldering, and make sure that all the connections were correct using a multimeter to run a connectivity check.
2. We traced every pin and noticed that the speaker should connect to the 3rd pin of the chip (TX) on one hand, which connects to the TXD pin of the ATmega238P chip, and ground (pin 10) on the other hand. Finally, we soldered the 7th pin of the chip, which should also go to ground, since it affects the rest of the circuit.
3. We had ordered another force sensitive resistor just in case, and so we used it instead of the other one. Also, we tried to separate the wires and untie them as we proceeded. Here is a picture of our final board (however, we replaced the yellow LED with a red one later on)

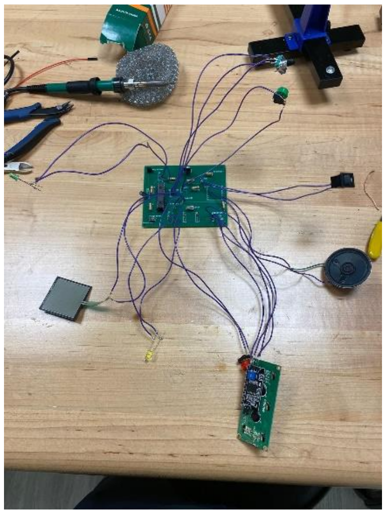


Figure 29
Overview of the board (Top layer)

Design Testing

Enclosure:

After assembling the enclosure, we composed two different tests to ensure that the enclosure was an adequate environment for the game and that the project requirements were fulfilled. The test cases are listed below.

Test 1: Enclosure is sturdy
Test 2: User only has access to inputs of the device
Test 3: User can visibly see and hear outputs

Results

Test 1: Enclosure is sturdy

To test the enclosure's sturdiness, we each tried pulling and pushing on the walls of the enclosure to see if it would break apart or crack during use. After each of our attempts, the enclosure remained intact and undamaged indicating that the structure was well built.

We wanted to make sure that the PCB was securely in place so that it wouldn't get damaged if the user were to shake or jerk the enclosure. To do this we decided to drill down the PCB to the bottom face of the enclosure (instead of gluing it down like a majority of our other components) so that it was less likely to move. After drilling, we tested this by moving around the bottom face of the enclosure in different directions to simulate the user playing the game. Since the PCB stayed in place the entire time, we were finally ready to attack the bottom face to the rest of the enclosure

Test 2: User only has access to inputs of the device

It was also important for us to test that the user was still able to reach all the necessary components of the game since this would determine the success of the enclosure. We asked a few people to test our game once it was completely finished to see how easy it was for them to access the three sensors used in the game. We initially had problems with the Trash-it button getting stuck due to the glue, but after removing some and repositioning the button, we no longer ran into this problem. As for the rest of the sensors, users seemed to be able to easily access and use them in the given time frame.

Test 3:User can visibly see and hear outputs

The main goal of this test was to make sure the user could see the directions, see their overall score, and hear the audio cues clearly. Again we asked a few people to play the game to let us know if this was possible for them, which they all agreed that it was. We did receive a request to add sound effects for every answer that the user gives (whether that be a correct or incorrect answer) so we did end up updating our game to include this feature.

Hardware:

Problem encountered:

As we tested our game for the first time, we noticed something usual. The voltage regulator would get extremely hot, even after the switch was turned OFF. Therefore, we looked more into it and realized that the switch was placed in the wrong spot. In fact, whether it was ON or OFF, current was still being delivered to the rest of the circuit. We checked our PCB schematics and figured it out. Turns out the switch should have been connected between the battery holder and the VIN pin of the voltage regulator. Here is a picture that shows the change that needed to be done.

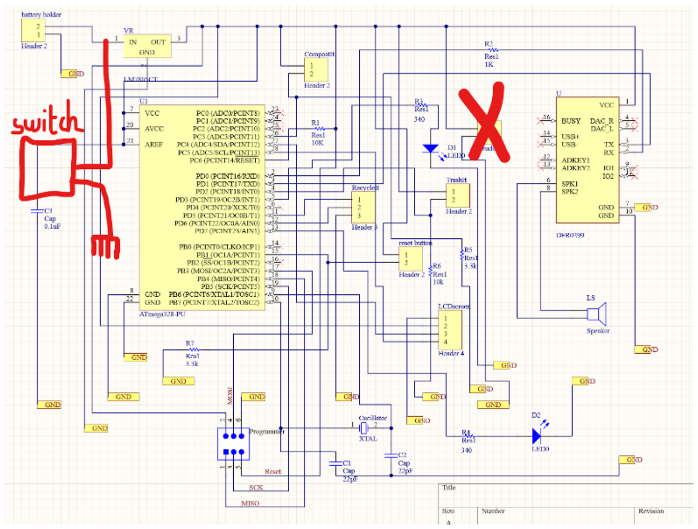


Figure 30
Modification needed visualized on the PCB schematics

Solution:
To resolve that issue, we had to cut the trace that connected the positive terminal of the battery to the VIN pin of the voltage regulator, and make it so that the current passes through the switch (when it's ON) before touching the voltage regulator. For that, we used a metal cutter to scrape the corresponding trace and therefore cut the connection. To verify that it was indeed cut, we used a connectivity check. After that, we connected the switch to the positive terminal of the battery and the VIN pin of the voltage regulator using wires of course. Here is a picture that shows the cut made to the trace.

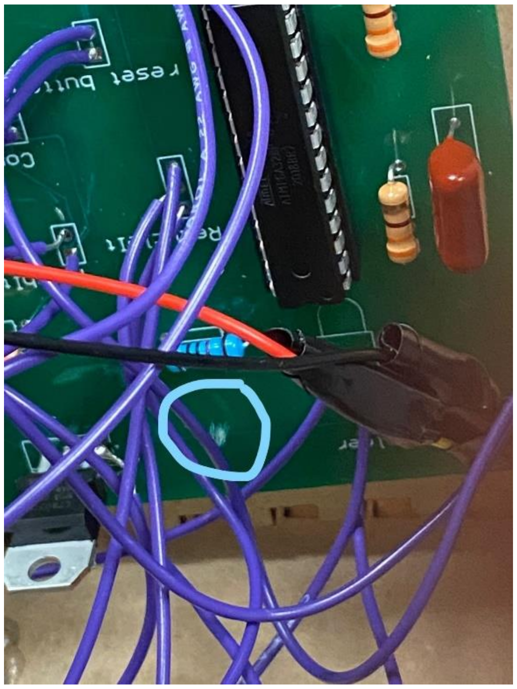


Figure 31
Picture showing the cut made on the trace between the battery and the voltage regulator

Software

The game software had many requirements to ensure the game functioned similarly to the classic Bop It game. Each of these requirements needed to be tested to verify their functionality.

Test 1: Components are interfaced correctly.
Test 2: Commands are generated.
Test 3: User action on input is checked for correctness.
Test 4: Game keeps track of score.
Test 5: Reset button functions correctly.
Test 6: Audio outputs when expected.
Test 7: User is playing against a timer.

<u>Test 1: Components are interfaced correctly</u>

As mentioned in the Design Verification section, each of the components was tested and verified independently to ensure functionality prior to integration in the overall design. The components were initially tested and verified to function correctly, and were also tested when malfunctions were suspected, as happened with the rotary encoder.

<u>Test 2: Commands are generated</u>

To test random command generation, we displayed the commands on the screen and verified they appeared, meaning they had been generated.

<u>Test 3: User action on input is checked for correctness</u>

To test if the user's action on the input is checked for correctness, we displayed "Correct" or "Incorrect" to the LCD when action is registered in response to a command. This was further developed to fit our game specifications, but our initial test was a simple print statement on the LCD.

Test 4: Game keeps track of score

To test if the game was tracking the user's score, we started testing by displaying the score to the LCD screen after every user response to a command (whether incorrect or correct). After this test was successful, the score was only displayed when the user lost the game so that we did not have to delay the game to display the correct score for every turn.

Test 5: Reset button functions correctly

To test if the reset button functions correctly, we ran several tests. We pressed the reset button in the middle of a game, upon device startup, and after losing a game. All of these scenarios resulted in a new game starting which was the behavior we expected.

Test 6: Audio outputs when expected

We had 4 cases in which audio would play. We tested each of them by simulating each case during gameplay. For example, to test the correct answer sound, we got an answer right. To test the game over sound, we got an answer wrong. To test the sound that played every ten points, we played to at least a score of 10. To test the game-winning sound that occurs at a score of 99, we made the winning score 10, and checked that the game ended, and the corresponding sound played.

Test 7: User is playing against timer

To test that the user is playing against a timer that speeds up, we played the game and did not answer fast enough. The game ended, hence ensuring the timer's functionality. To check that the timer sped up, we printed the current time limit to the LCD after every correct answer.

Budget and Cost Analysis

Determine financial cost of manufacturing and assembling our design

Product	Price
Multicolored Tactile Buttons-4 pack	$1.00
Power Switch Rocker	$0.67
Rotary Encoder	$4.50
Rotary Encoder Knob	$0.80
Force Sensitive Resistor	$12.50
LCD Screen	~$5.00
LED (2 in total)	$1.26
Speaker	$3.26
9V Battery	$1.50

Battery Holder	$2.50
Solid Wire spool (jumper wires)	$2.80
ATMega328p Chip	$3.00
Voltage Regulator	$0.65
Crystal Oscillator	$0.32
Capacitors (3 in total)	~$2.25
Resistors (2 in total)	~$1.00
PCB	$2.00
Medium draft board panels (2 in total)	$20.00
PCB standoffs (4 in total)	$2.36
Screws (4 in total)	$0.64

Total Cost (for 1 dispose-it): $68.01

Therefore the cost to produce 10,000 copies would be:

$680,100

This number might be an overestimate of the manufacturing cost for Dispose-It since it does not consider the reduction of prices that comes with buying components in bulk.

Team

Team Roles

Ashley

Ashley was responsible for the enclosure. She handled designing the enclosure that houses the PCB, securing the PCB within the enclosure, and ensuring the user could easily and intuitively access all the inputs required for gameplay.

Destiny

Destiny handled the software. She developed software to meet the project requirements, and ensure seamless gameplay for the user.

Rayan

Rayan was responsible for the hardware. He did pin assignments for the ATmega328p, created the schematic, designed the PCB, and soldered/assembled the components for the final design.

Team Dynamic

Team Communication and Meetings

Our team had a meeting schedule that was not explicitly designated but happened to be the same every week. We like to have help from the TAs so we almost always had a team meeting after class during office hours. A few times during the semester, we had to meet deadlines, and we had a team meeting on a Sunday. Luckily, all team members were dependable so it was not pertinent to have a rigid meeting schedule. The team communicated over text messages outside of class. We were all communicative about any milestones we reached, or any questions we had for each other. All teammates were accessible and trustworthy, therefore we had no issues with team communication and our team dynamic was positive and effective.

Timeline

March 17th - Project Proposal, 1st parts order, and preliminary sketch
March 22nd - ATMega Pin assignments
March 24th - Deciding on 3rd sensor, First rendition of schematic
March 28th - LCD display working, basic command functionalities working, 2nd parts order
March 29th - Prototype of Enclosure Design created
March 31st - Final PCB schematics and PCB layout made
April 1st - New parts arrived for testing
April 4th - Force resistor code working
April 5th - First laser cut box
April 7th - Started soldering the PCB
April 11th - Audio cues working
April 15th - Final laser cut for enclosure printed, soldering for PCB complete
April 18th - Final version of the code uploaded to the chip
April 19th - Assembling the final version of Dispose-It

Summary

In conclusion, we were able to design a simple, yet fun educational game in a span of only a few weeks. The project consisted of 3 actions, RecycleIt, TrashIt and CompostIt, which the player had to choose from in order to dispose of an item that pops on the screen. A push button, rotary encoder and a force sensitive resistor were used for TrashIt, RecycleIt and CompostIt respectively. The design also included a reset button, an LCD screen, and LEDs with a speaker to indicate if the answers were correct or not. We ran a loop through the game and the code was divided into classes. The final enclosure consisted of a box with all the other components glued onto its surface. With great communication within the group and teamwork, we made sure that all of our parts (software, hardware, enclosure) worked well together. From breadboard prototyping to making sure the board fits in the final box, we had to be on track of what each one of us was doing. Although we encountered many problems, we were able to resolve them and make changes accordingly. However, some modifications could have been made to improve our final design. On the software level, more commands could have been added to give the player more options and expand their knowledge. Also, the time delay between every instruction could have been changed to make the game a bit more challenging. Moreover, it would have been more interactive if pictures of the corresponding items would pop on the screen instead of only displaying the name. When it comes to the PCB design, it would have been easier if traces weren't as close to one another, and holes were wider. Also, the switch should have been placed between the battery holder and the voltage regulator from the beginning, which would have allowed us to avoid cutting traces on the board and using extra jumper wires. On the enclosure level, the design would have looked better had we used 3D printing and made it look like a trash can or a dumpster. However, this would have taken more time and made it harder for the PCB to design, as it needs to fit inside.

All things considered, we are proud of what we accomplished in this project. As tough as it seemed at certain moments, we learned a great deal and overcame all the problems we faced.