Design Overview

Our bop it style game is called Gamble It. Gamble it, as seen in the name, is centered around gambling where we implemented three different casino games. We use slots, blackjack, and roulette. For slots, we implemented a push button and an OLED screen that will spin a virtual slot machine. For blackjack, we used a touch sensor that when it detects a touch, a message will be displayed to the OLED that says “Blackjack”. For roulette, we used a sound sensor that when a clap is detected, it spins the LED roulette wheel. To start our game, you first turn on the on/off switch and then you MUST insert money into the money slot where an IR sensor will sense once the money has been detected. Once the money is detected, then you can press the start button. You are not able to start the game unless you insert money into the game. This includes when you mess up or beat the game. You must insert money into the game every time you want to play. This helps to simulate a real like casino experience. Once you press the start button, you will be given one of three distinct sounds. Each sound corresponds to a certain game (blackjack, slots, and roulette). For first time users that don’t know the sounds, we also have an OLED screen that will display what to do, it will either display “touch now”, “press now”, or “clap now”. As you are playing the game, your score is displayed in the top left of the OLED and is updated after each successful input. We added a cool feature to slots portion of the game too where if you spin the slot machine and hit the jackpot, which is “777”, 50 points get added to your score. We specifically coded the slots game so that you have a 20% chance of hitting the jackpot when the slots games is the input and is played within time. When someone either runs out of time, presses the wrong input, or wins the game, a game over sound is played and a message is displayed to the OLED that says game over, tells you your score, and tells you to insert a coin and press the start button to play again.

References:

Touch sensor- https://docs.sunfounder.com/projects/umsk/en/latest/02_arduino/uno_lesson22_touch_sensor.html

Sound sensor detection - https://www.youtube.com/watch?v=Mx_daAYlHvs

Slot machine - https://www.youtube.com/watch?v=1SCmmTmyk-I

OLED - https://randomnerdtutorials.com/guide-for-oled-display-with-arduino/

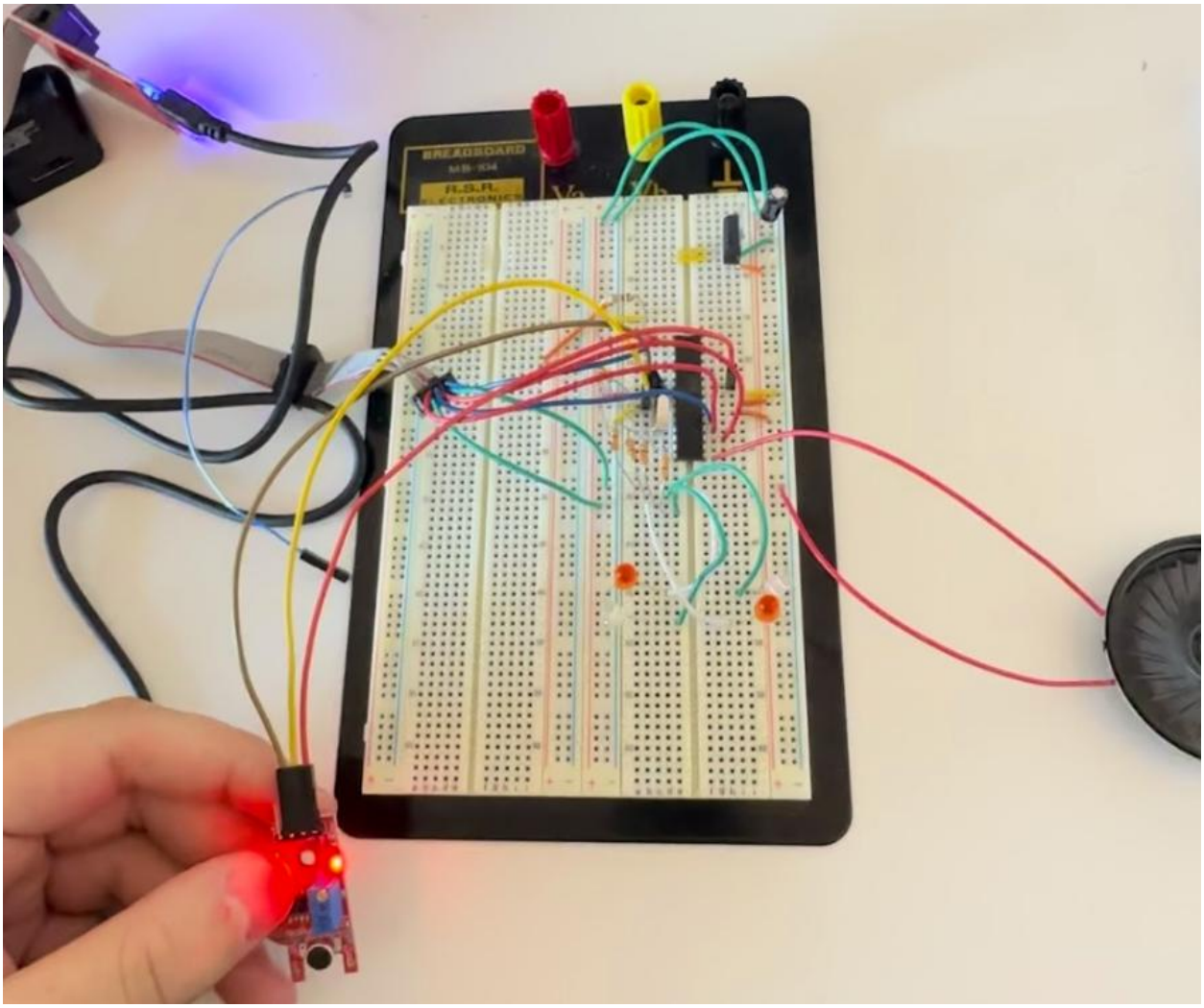
IR sensor - https://www.youtube.com/watch?v=rr8p2wtrqXw

We mainly stuck with everything from our original design idea. We knew that we wanted to do blackjack, slots, and roulette. One thing we were questioning about doing was having a money gun shoot out money if you win a jackpot or win the game. We decided not to do this early on because we thought the rest of the features would be hard enough to implement.

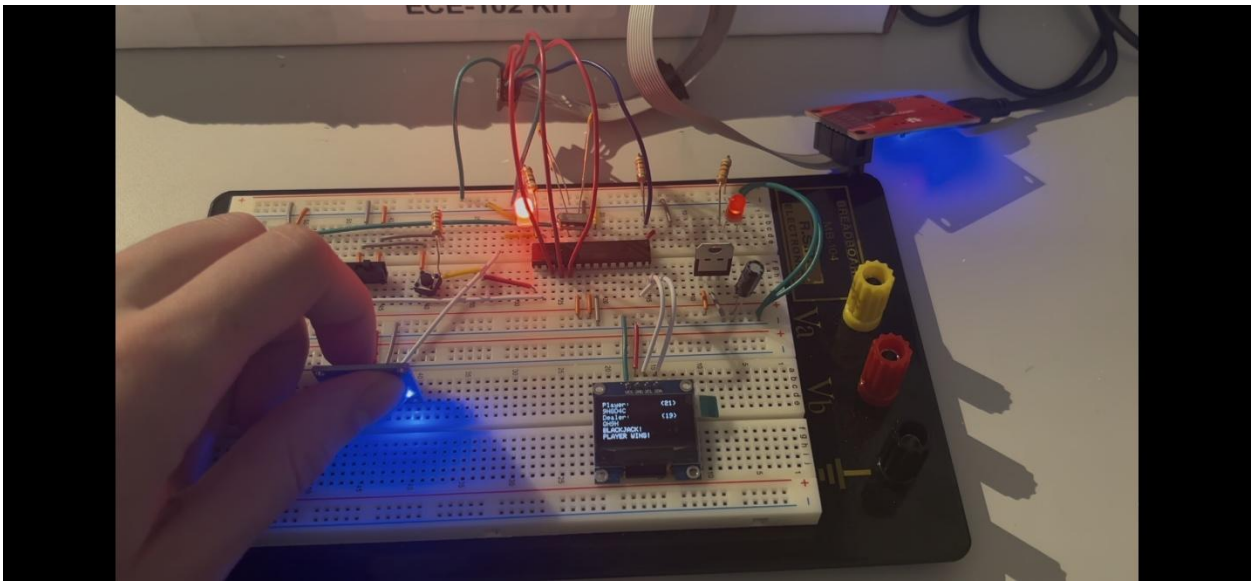
Design Verification

We decided to individually test each component out separately and then add them all together. Joey oversaw testing the sound sensor for roulette as well as the speaker. Philip oversaw the slots and tested the slot machine with an OLED. Seth tested the blackjack portion with the OLED as well as the IR sensor.

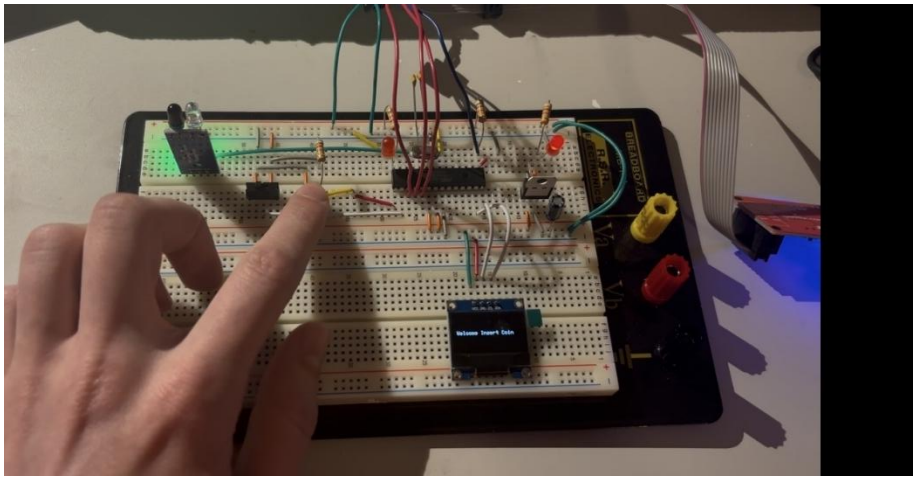
For the actual sound sensor itself, testing was really difficult. It was hard to find the perfect sound threshold to detect only a clap and not other noise such as voice or the speaker playing. After about 4 days of messing with different thresholds and not knowing why anything was showing up in the serial monitor section in Arduino, it came to our attention that we needed the RX and TX pins connected. To do this, Jacob let us use one of his Arduino boards to see what the sound threshold should be set to. After figuring that out, it made testing the clap detection much simpler. The sound sensor that we also started with had a smaller mic for detecting sound, so we switched it to a bigger one and that worked very well. The sound threshold was set to 540 for the clap detection and this was the perfect number. (Picture of sound sensor design below).



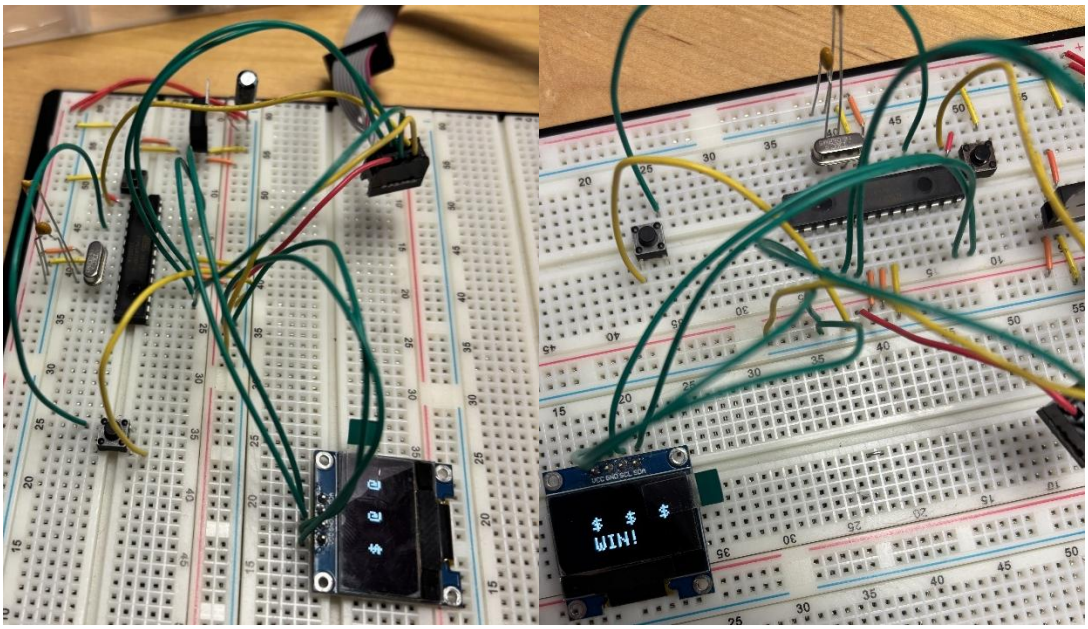
For the touch sensor it was relatively straightforward. The sensor has three pins: Vcc, GND, and Signal. So, the signal pin just had to be connected to an open digital pin. Before connecting the OLED screen an LED was connected to another digital pin as an output to just test to see if the touch sensor was working correctly. Once verified that every time the sensor was touched the LED would light up, coding the OLED screen was the next step. For Blackjack the screen would show the players' cards and the dealers' cards that were dealt. Then once the sensor detected an input, the screen would change adding an additional card to the player's hand; that would add to 21, and the screen would flash with a message saying “BLACKJACK, you won”. (Picture of Blackjack breadboard design below).



For the IR sensor, on/off switch, and start game button the hardest part was the software aspect. We coded it specifically so that only when money, movement, was detected then you could hit the start button and the game starts. Each component was tested individually and very similarly to the touch sensor. Before combining each component was connected to an LED and only when the desired input was met would the LED turn on. For example, only when an object was detected would the LED turn on. Once all components were verified that they work, then came the task of assembling them. The correct sequence to light up the LED and get it to stay on was turn game on, detect money, press button. Each correct input would flash the LED, if you tried to press the button before an object was detected nothing would light up. (Picture below shows trying to press the start button but no LED is lit because money/object was not detected).



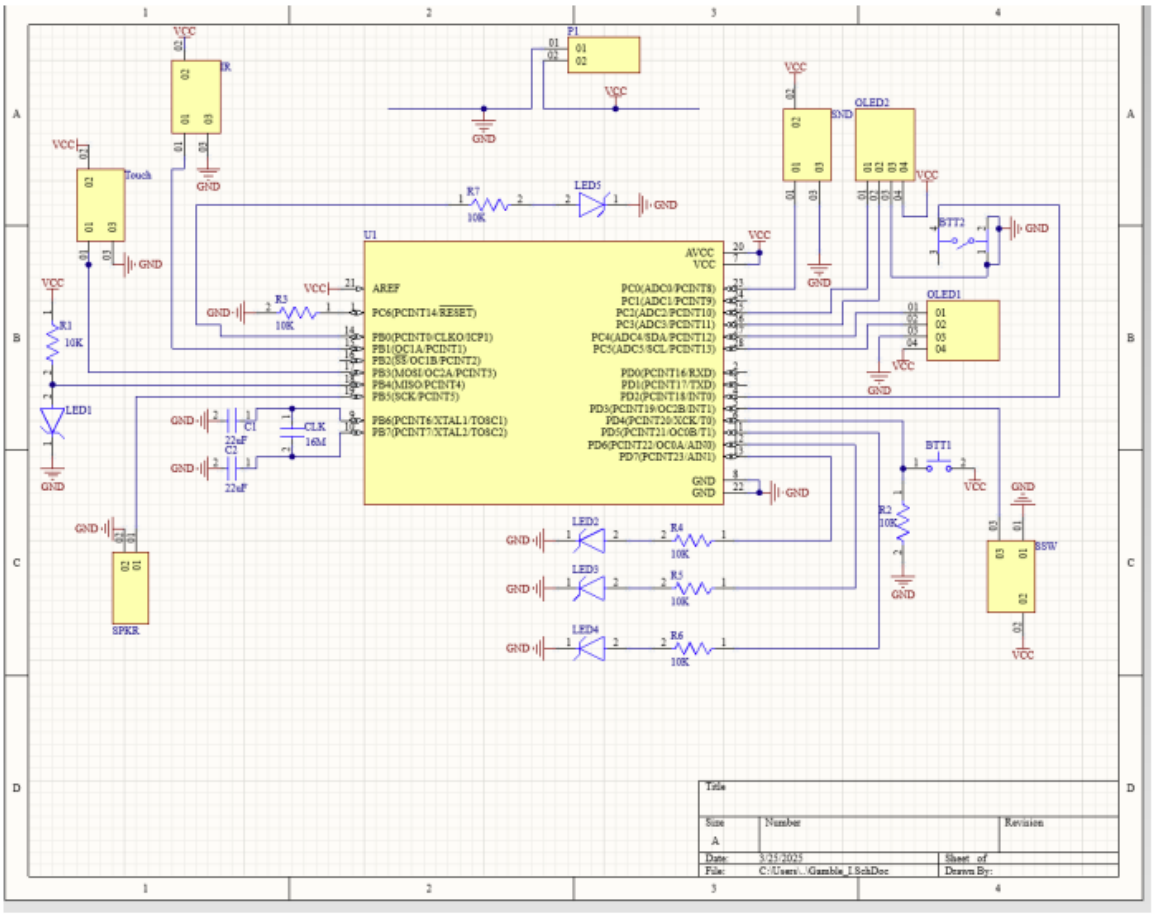
For the slots code, there wasn't a whole lot of troubleshooting to be done. We started out with a program that would cycle through 8 special characters in 3 slots, with "$, $, $," winning the mini game. With this first draft, the button basically worked as a debouncer; the user would initialize the game by pressing and holding a button, while the program cycles through the special characters, and once the user let go of the button, the slots would stop. One of the first challenges we had was we kept getting errors with our ATMega chip. After troubleshooting, we realized that the chip was burnt, and once we replaced it, the program worked pretty well. After we had this first draft, we decided that we wanted to move in the direction of wanting the button to activate the slots on press, and then have the slots spin for a set amount of time before slowing. We set the total spin time to be 2000 milliseconds, and a delay command that runs during every cycle of the while loop, and it also increases the delay until the slots stop at their designated values. Finally, when we started integrating the 3 mini games, we realized that it was very difficult to win at slots. While that may be good for an actual casino, we wanted to give the user an actual chance to win. To do this, we designed a program that would cycle through different arrays of possible 3-symbol combinations. This would allow us to manually tune the odds of the slot machine, and we decided to tune it to a 20%-win rate. In addition, if the user hits the jackpot, we add 50 points to their score. We also changed from special characters to numbers, with "7 7 7" equal to winning.



We then all met up together and combined all our components into one board and made sure that they all fit into one chip. They did, so we next ran a simple code to see if it would run with all our components connected. We simplified our individual code for our components and ran it. We had a code that when a clap was detected, a led turned on and

the speaker made a noise. We tested another code to make sure that the touch sensor would detect when someone touched it and that worked. We lastly tested to make sure that when the roulette wheel button was pressed, the OLED spun the virtual wheel. All of this worked, so we then decided to get started on our PCB and make the schematic for our design.

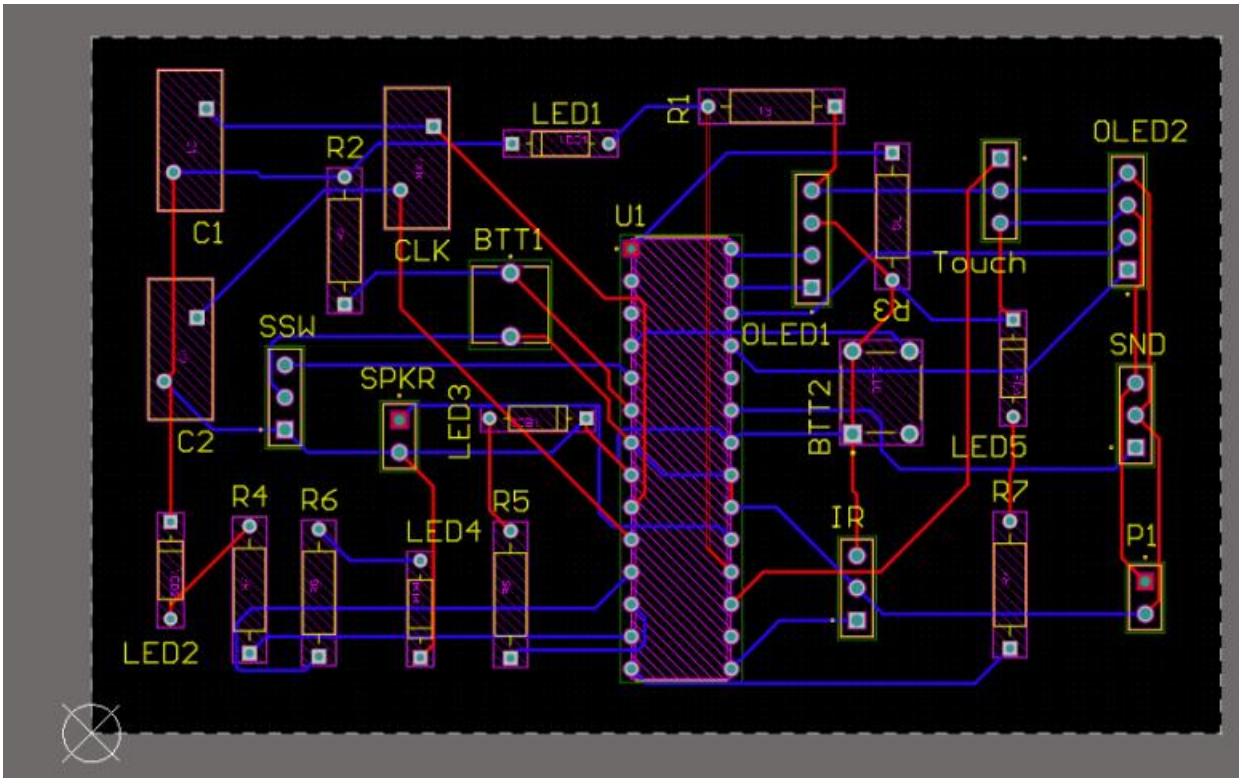
Electronic Design Implementation



The sound sensor was connected to pin A0, the LEDs were connected to digital pins 5,6,7, and 8. The touch pad sensor was connected to digital pin 11. The push button for the slots machine was connected to pin 2. The buzzer was connected to pin 13. The IR sensor was connected to digital pin 9. The reset button was connected to pin 4. In the actual

schematic, we accidentally connected the reset pin to ground instead of connecting it to power.

Our schematic has four 3 pin headers (one for the touch sensor, one for the on/off switch, one for the sound sensor, and one for the IR sensor). We then had two 4 pin headers for the two OLEDs (we ended up only using OLED1). We then also had two 2 pin headers (one for the speaker and one for the power/ground). Using these pin headers saved us time where we didn't have to get each component and load them into Altium. They were also very easy to use. Instead, we were able to use simple pin headers. Other than that, the other components were the exact components that we were using. There are 5 LEDs, 7 resistors, 3 capacitors, and two push buttons (one 4 pin and one 2 pin).



For the actual PCB, we tried to make the board as small as possible so that it would fit inside our enclosure. For making the connections, we used the autoroute feature and this was super helpful. For the layout of our PCB, we have the ATMega chip right in the middle and scattered the rest of the components throughout. We tried to make the board as small as possible, so the rest of the components were placed to try and make the board as small

as we could. We didn't experience any challenges with designing the schematic or PCB. It was thankfully and smooth process.

Software Implementation

Here is a list of our function protocols:

```
void turnOffAllLEDs();

void playCommandSound();

void playFailSound();

void playSuccessSound();

void playStartupSound();

void issueNewCommand();

void displaySlotMachine();

void displayBlackjack();

void displayGameOver(bool isWin);

void displayStartScreen();

void runSimpleLEDRoulette();
```

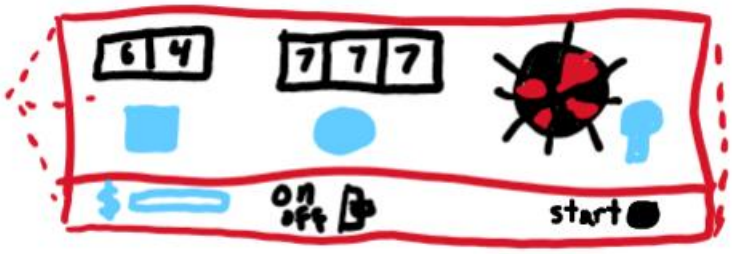
Our code is designed and organized in a loop structure where we initialized the components in setup() and the actual game runs in loop(). We implemented a lot of commands that manage different parts of the game. For example, our issueNewCommand() function will randomly generate one of the three commands. Once it does this, the CheckplayerResponse() function checks to see if the user responds to the right command in time. It also checks to see if the user inputs the incorrect command. We have a command for each game (displaySlotMachine(), displayBlackjack(), and runSimpleLEDRoulette()) that runs when the user does the correct command within time. The displaySlotMachine() function simulates a basic slot machine by cycling through random numbers on the OLED before showing the result. We included a cool feature for this where if you get "777", 50 points get added to the score. The displayBlackjack() shows a simple blackjack screen to the OLED where the dealer has 19 and the player has 21, so the player wins. "Blackjack" is also displayed. The runSimpleLEDRoulette() function lights up a series of LEDs in a loop to simulate a roulette effect of the ball going around the

roulette wheel. Once you mess up or win the game, the DisplayGameOver() function runs and displays a message to the OLED. On the OLED you see a message that displays game over and the final score. It also mentions that you need to insert a coin and press start. You are then able to insert a coin and start again.

A cool feature that we added to this project was the IR sensor. Before you can start the game, you must insert a coin to start. We did this by constantly checking the irSensorReading and checking to see if it goes to LOW, which indicates that something was inserted. Once this reading becomes low, and the button state then goes to high (the start button was pressed), the game starts again.

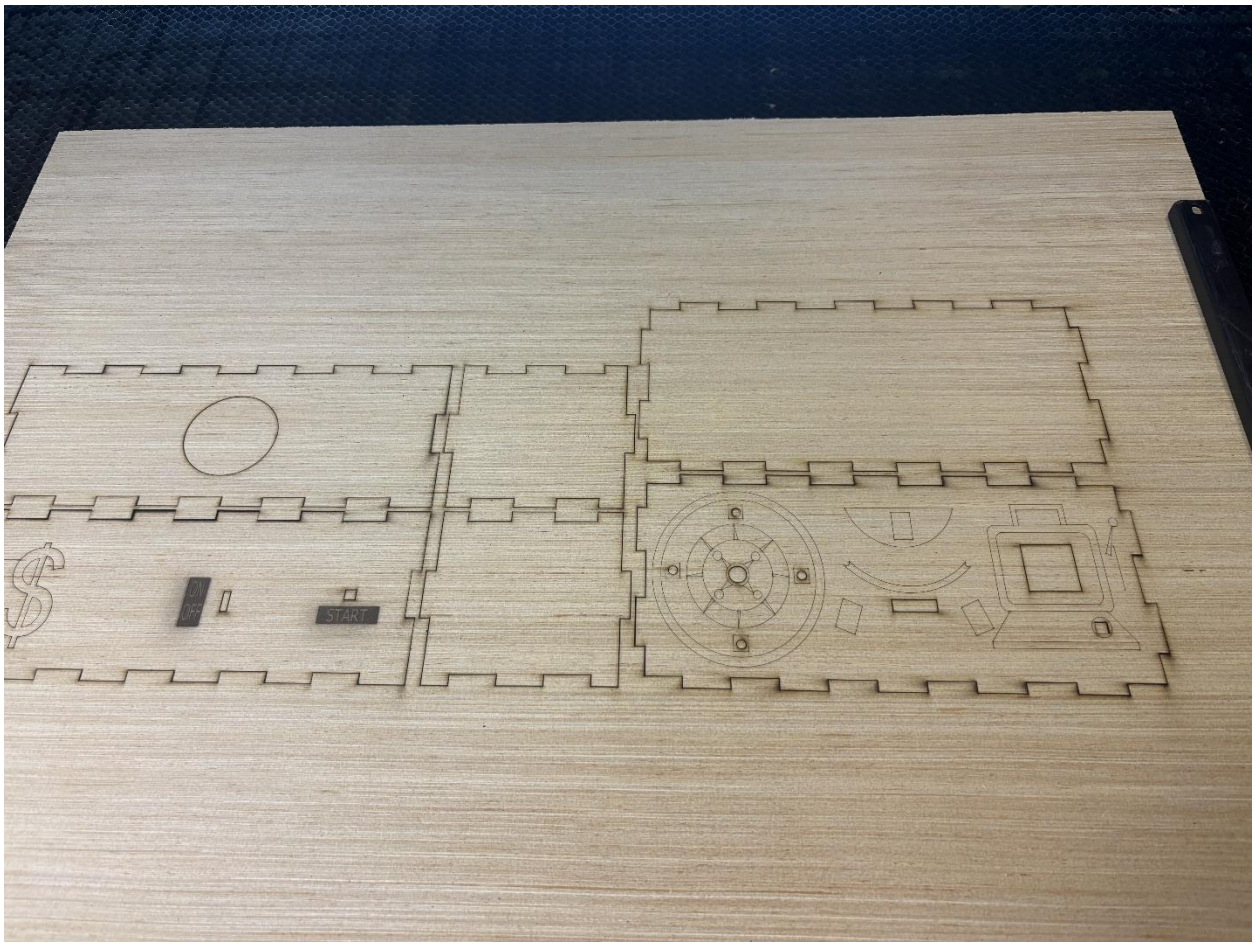
Getting the actual code to run all together with the components connected to the PCB was very challenging. I kept getting errors where neither the speaker would sound, and the OLED wouldn't turn on. After debugging for a couple hours, the issue was because of delays. Once the delay (1000) was removed, it finally started functioning properly. Combining all the code together into one final code took a lot of time and we finally got it all working.

Enclosure Design

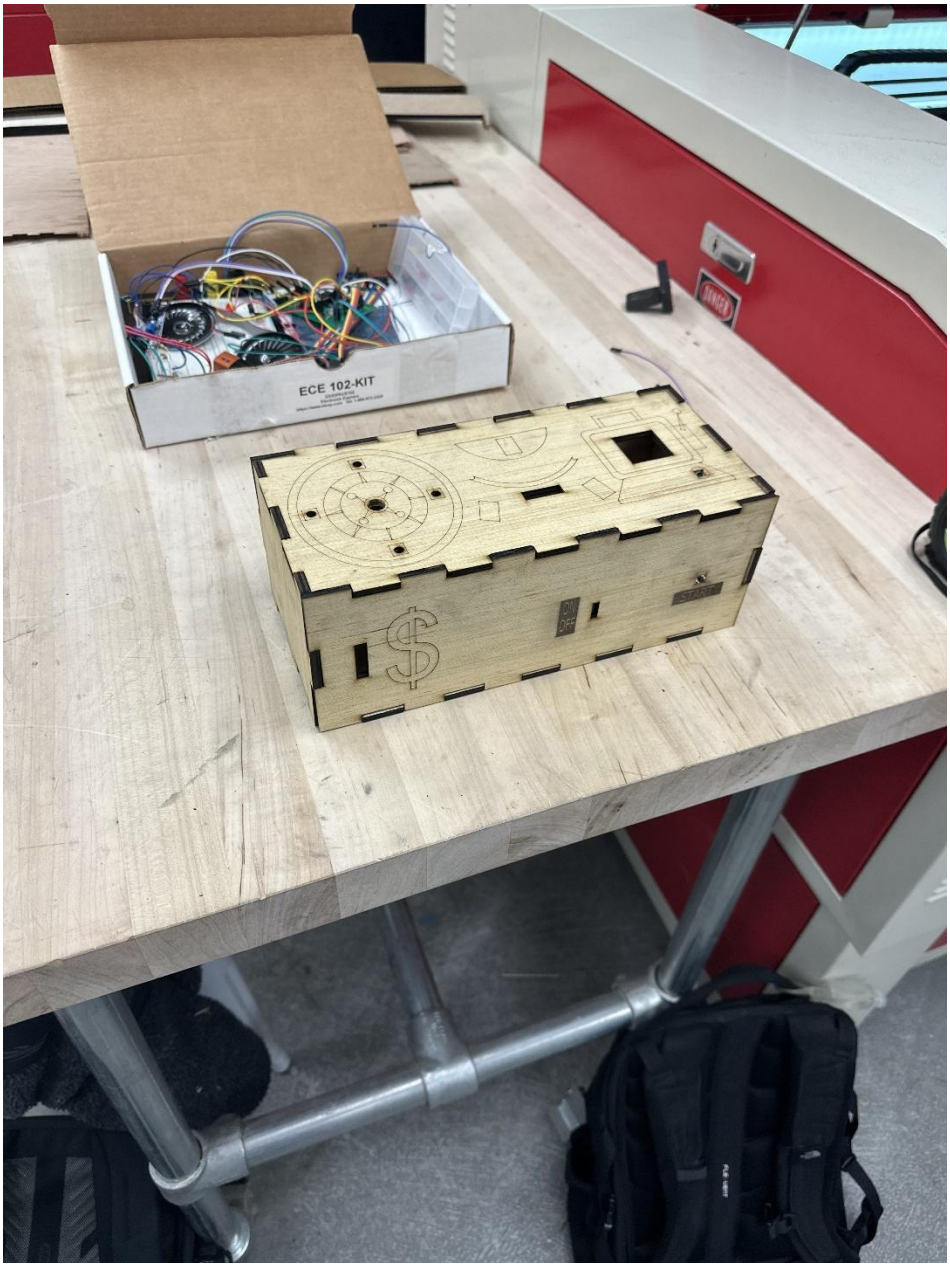


The Enclosure was decided to just be a box to keep it simple and straightforward. We wanted Blackjack, Roulette, and Slots to be on top for easy playing. Initially we did have it designed where we had two OLEDs, one for slots and one for blackjack. Slots would be in the middle, blackjack on the left, and roulette on the right Then we found out that we can't have two OLED screens on one ATMega chip, so the design had to be changed a little bit. Slots with the OLED screen and button were on the right, blackjack with the touch sensor was in the middle, and roulette with the four LEDs and sound sensor were on the left. On the back of the box was a cutout for the speaker. On the front we had cutouts for the money

slot, on/off switch, and start button. The money slot would be on the left, the on/off switch in the middle, and the start button on the right. In addition to the cutouts, the board was also designed with vector drawings so that the actual components were integrated with the drawings to make it look like the actual gambling games. Vector drawings for the front were also done. A money sign for the money slot, “on/off” lettering for the switch, and “start” lettering for the button.



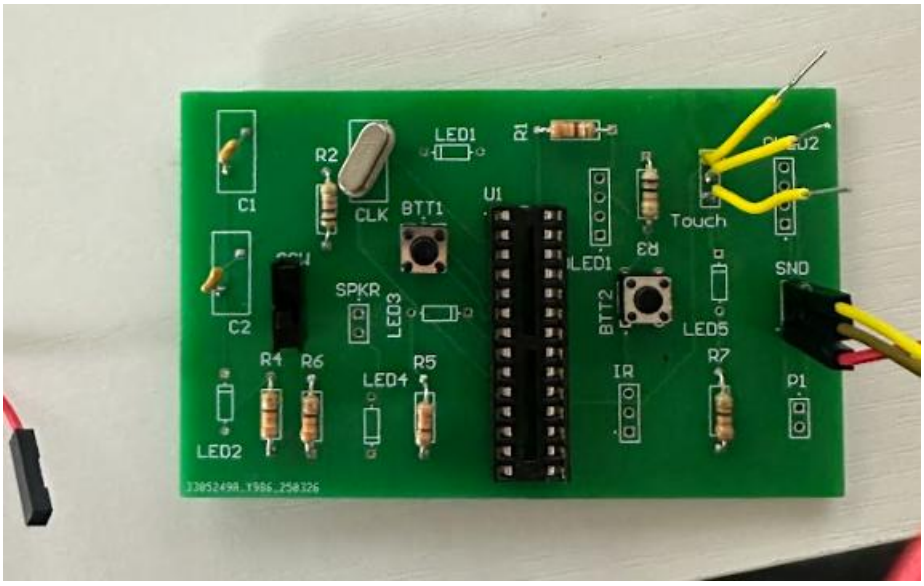
Designing the enclosure and vectors was all done in Fusion360. We decided to use wood as our material and to use a laser cutter to cut out the shape and design the drawings on the box. After receiving makerspace training, we downloaded the Fusion360 file into a dxf vector drawing and uploaded that into the laser cutter computer. Quarter inch thick wood was used as the thickness for each part of the box.



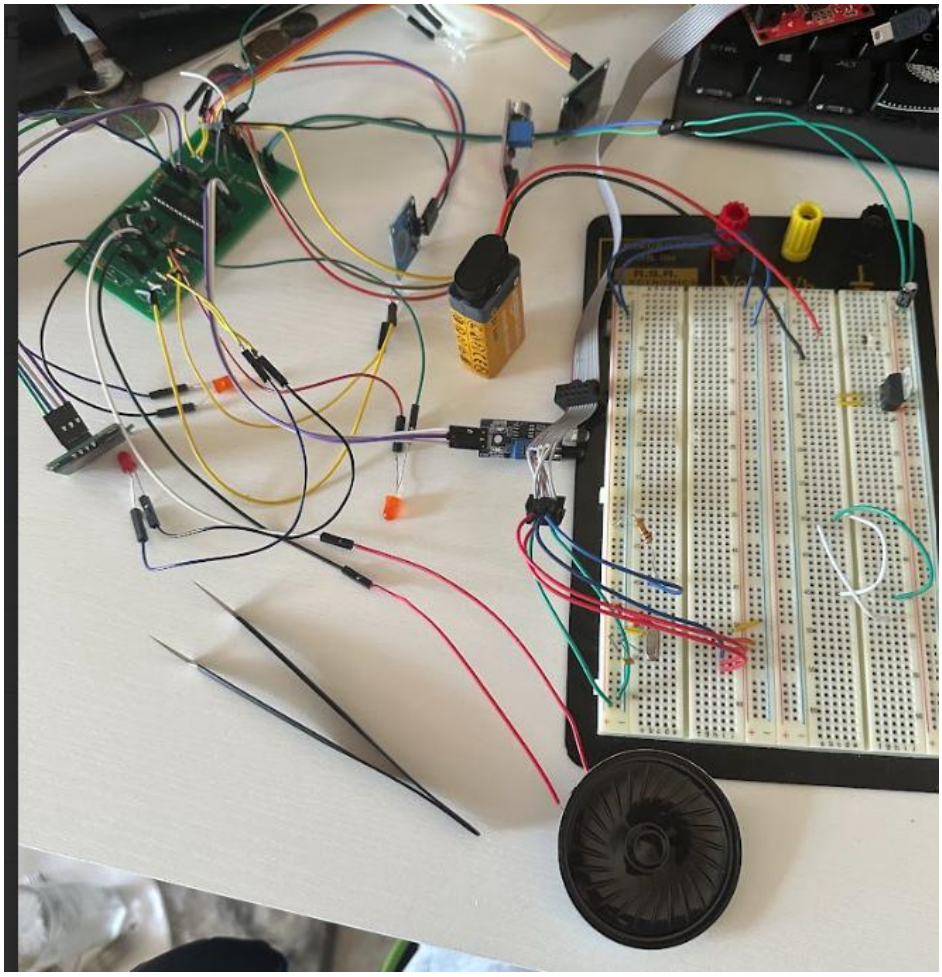
Assembly and System Integration

Soldering the board gave us no issues. It was simple and took no more than 1 hour. Below is an image of the board partially soldered. As you can see in this, the buttons and switch were soldered on the board. However, the components needed to be connected to wires because the buttons and switch need to be on the actual encloser and not on the PCB. So, we had to desolder and resolder wires for these components. Since in our design, we accidentally grounded the reset pin. We fixed this by desoldering the reset pin and resistor

3 that was connected to ground. Once we did this, our board started functioning properly. (Picture of the PCB partially soldered below).



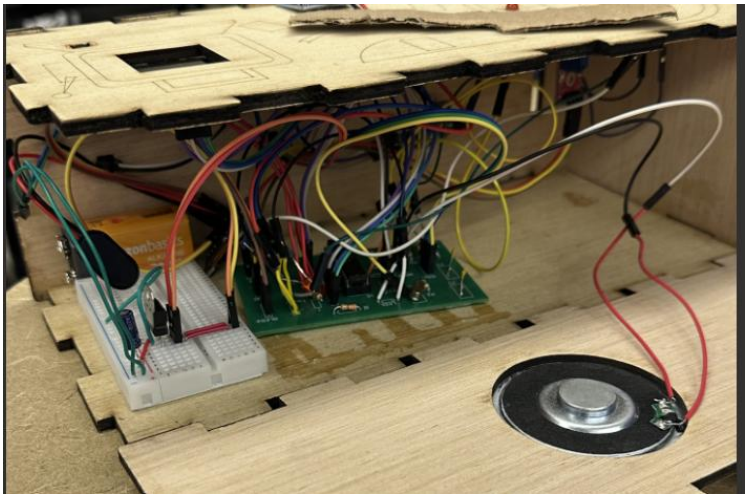
We connected all the components up to the PCB and tested to see if each component worked. We first started testing each component individually and they worked properly. When we combined everything together was when we started getting errors. Having two OLEDs was tricky and we couldn't figure out how to get the 2nd OLED to work, so we ultimately put everything onto one OLED. Once we did that, we started combining all the code and testing everything together. After hours of testing, we finally got it working and all functions worked properly. (Picture of the PCB and components connected below).



After the laser cutter cut out individual pieces of the box, it was realized that the sides were not the correct dimensions and would not fit with the box. So, we had to redo the design on the laser cutter and print out the side pieces again. Once confirmed that all the pieces correctly fit, we could now fix those in place. Wood glue was used to fix the bottom, front, top, and one side together. The back and the other side were left free because we still needed to insert the board and components. While trying to fixate the components into the desired cutouts, the holes had to be sanded a bit to allow easy placement. Then the components were fixed in place with super glue.

There was one major problem after we fixed all the components in place. When super glueing the PCB to the inside of the box the glue must have dripped across a couple of solders because once it was glued and we tried to play the game, the switch was not working, and the game was just constantly on. To fix this we moved the ground and voltage wires from the switch and PCB and connected it between the switch and battery. So that once power is connected, you have to turn the switch on in order for the game to turn on.

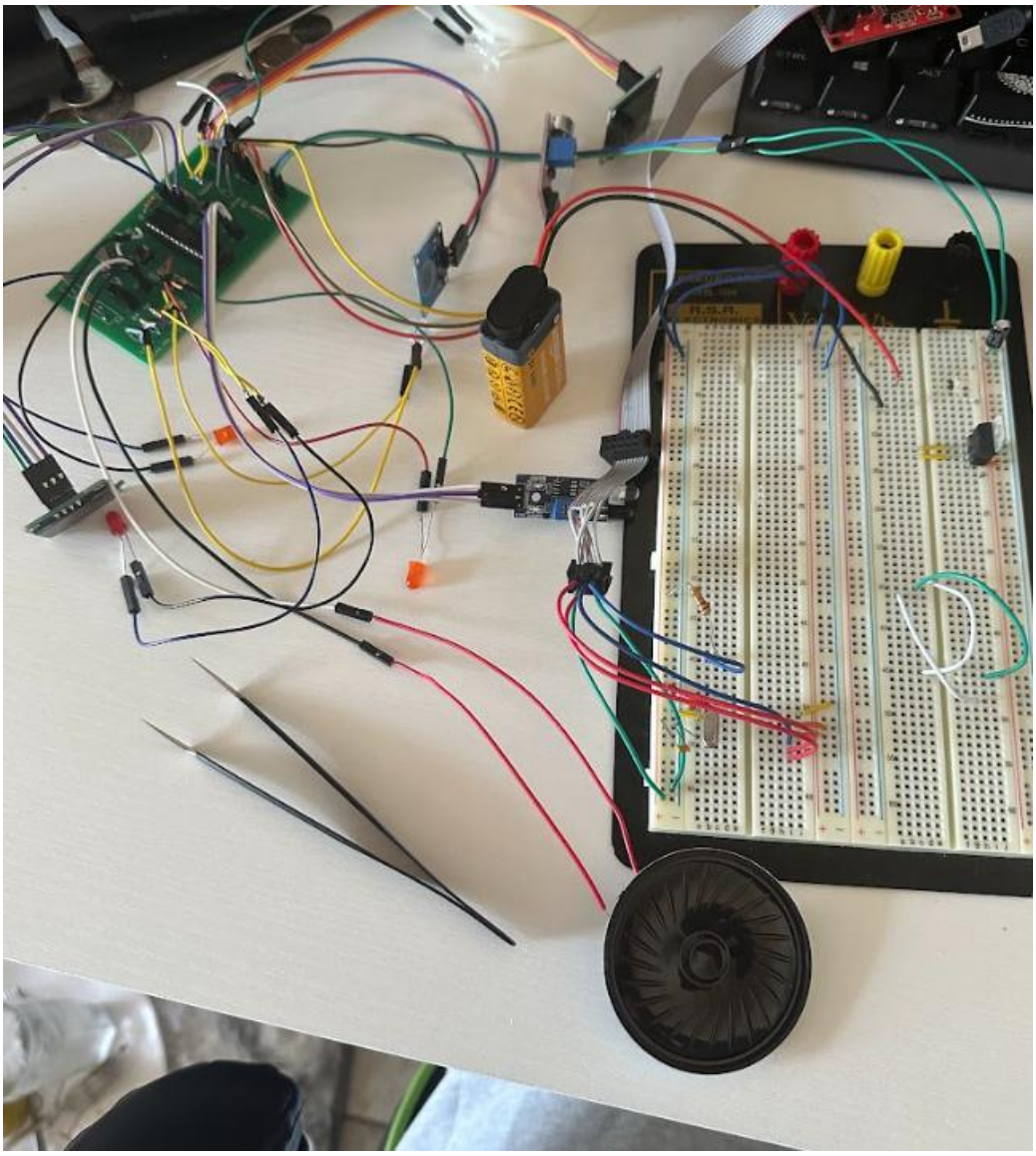
We also ran into another problem where the game wouldn't turn on at all, and that was due to voltage and ground wires not being in the right location for the voltage regulator. That was a simple fix of just moving the wires over to their correct position. (Picture of the PCB inside of the enclosure below).



Design Testing

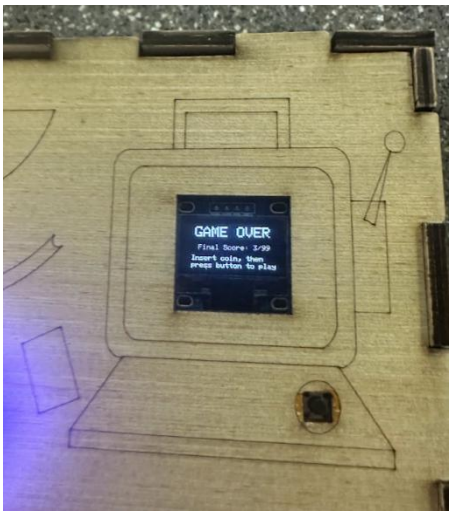
For testing our prototype, we first connected all the components to the PCB. Once this was done, we tested the individual components by themselves. We ran the code that each of us made for each component and they each worked separately, but once we started testing everything together, we got issues. We got the clap sound sensor, speaker, and IR sensor to all work together, but once we implemented the touch pad, slot machine, and OLED a lot of issues occurred. The OLED started to not turn on and the speaker stopped working. For the slot machine, we originally used characters instead of numbers to spin on the machine. Our machine wasn't spinning when the characters were supposed to spin, but when we changed the characters to numbers, it spun successfully. To get everything to work together, we had to add the components one at a time and that took a lot of time. Once we finally got the last component connect, which was the slot machine with the OLED, we got the same issue where nothing would turn on. It took a lot of time debugging and we realized that it was because of one delay(1000). Our theory is that the delay (1000) was somehow nonstop looping, which allowed nothing else to run. Once we removed the delay, everything came together and worked. So, after hours of coding a debugging, we got our code to properly work. After this, our project worked 100% the way we wanted it to.

Another issue that came up was that some of our ATMega chips stopped working after we kept uploading code to them. So, we had to change the ATMega chips about 5 times and use new ones. ((Picture of the PCB and components connected below for testing).



(Below are images of the components connected to the enclosure and the game on).

(below is an image of the OLED screen when the game is over)



Working Video of our project - https://drive.google.com/file/d/1F2KSDveOcy8vRROT3FhGZht06GuTFspm/view?usp=sharing

Budget and Cost Analysis

When we put our board onto JLCPCB's website, we found that it cost around $2 per board to get them fabricated. For the solder, we estimated that we would use around 2 inches of solder per toy and we found a 4-pack on amazon to cover this. For the enclosure, we only need a 22"x12" board per toy, and we found boards being sold that were 24"x12" a

piece. Obviously, we would hope there would be some sort of bulk discount when we order 10,000 of them, but this did simplify our cost process. For the PCB components, we estimated that we would spend $5 per board for miscellaneous resistors, capacitors, voltage regulators, and other elements. Finally, we have our prices from our bill of materials. In total, we found our total cost for 10,000 units to be $241,037.32, or about $24 per toy. This also does not take into consideration shipping costs or taxes.

PCB: $2 a board --> $1,945.80

Solder Iron + Solder: 22 packs of these : $1,671.78

2 inches of solder per pcb.

114 boards per 1 lbs. solder, 88 lbs. of solder total, 4-pack of solder for $75.99

Enclosure: $36,000

Wood – 2 12”x 5” boards, 2 12”x4” boards, 2 4”x5” boards = 22”x12” board of wood per enclosure. ¼" thick, found plywood on linked website which does sheets of 12”x24” sheets of plywood for $3.60 per piece.

PCB Components: ~$5 per board --> $50,000

BOM Total Price:

IR Sensor: 2-pack for $9.99 = **$49,950** for 5,000

Microphone Sensor: 3-pack for $6.98 = 3334 packs --> **$23,271.32**

OLED: 5-pack: 5-pack for $12.15 = 2000 packs --> **$24,300**

Touch Sensor: 1 for $5.39 --> 10,000 packs --> **$53,900**

Total: $241,037.32 (no shipping costs, untaxed), ~$24 per toy

Team

Joey- Oversaw the sound sensor component testing for roulette, soldering the PCB, helping combine all code to make the final product work, testing the PCB for proper functionality, and assisting with testing the final product test with the enclosure.

Philip- Oversaw the slots game component and OLED with the slot component, helped install the PCB into the enclosure, assisted Joey in writing the code and combining the code for the final product, and assisted with testing the final product test with the enclosure.

Seth- Oversaw the Blackjack component, IR sensor, On/Off switch, start button, designed the enclosure, built enclosure, and helped to put the PCB into the enclosure and test.

We all had our equal amount of work put out and would ask each other for help when we were struggling or needed help. We made a group chat and communicated over that and scheduled times to meet outside of class where we worked on the project. We met outside of class many times, combining all our components together, getting trained in the treehouse, building the enclosure, putting everything in the enclosure, and doing our testing with the completed design.

Timeline

Week 1: This was the week where we got the components for our project. We each grabbed our components and started testing on breadboards to get them to work separately.

Week 2: We got each of our components to work separately and combined them all together to see if they would work on one chip and fit on the chip as well. We found that they fit and worked, so we then started working on the PCB design.

Week 3: Around this time, we submitted our Gerber files and waiting for the PCB to come in. While waiting, we started on our enclosure and got training in the treehouse to use the laser cutter.

Week 4: Around this time, we received back our boards for testing. The board was soldered this week, and the code was uploaded onto the board. Not all the code worked, so we had to spend hours to finally get all the components to properly work on the PCB.

Week 5: We laser cut our enclosure this week and started to combine everything together. We built the enclosure and added the PCB inside of the enclosure. We had to do some debugging and then we got our final design to work.

Week 6: On Monday of this week, we met for our final time to record a video of our final design properly working. We submitted our design and are all proud of what we have accomplished.

Summary, Conclusions and Future Work

Our project was a lot of work, and we learned a whole lot while building it. We tested each of our components separately (blackjack, roulette, slots, IR sensor, speaker, start button, and on/off switch). Once we got those to work, we combined everything together and tested to make sure the components worked on one chip. We then made the schematic of the board, made the PCB, and sent the Gerber files to get the PCB. While waiting for the PCB to come back, we designed the encloser on Fusion360. Once we got the board back, we soldered and spent hours combining code all together and testing all components connected to the PCB. After countless hours of working, we got Gamble It game to finally worked and we are so proud of our work.

If we were to do another design iteration, we would do more testing before ordering the PCB. We would also check to make sure that our PCB was properly connected. We accidentally connected the reset pin to ground instead of Vcc and didn't realize until we already had the PCB. I would also spend time checking to make sure that the OLEDs work properly because when we did the testing, we couldn't get the 2nd OLED to work and had to use only 1 OLED. So, if there were a few things we could do differently, we would do much more testing before building the PCB to make sure everything works together, and we would check the PCB to make sure everything is connected properly.