Project: Stack Attack

Instructor: Colleen Josephson
Project Engineers:
Diya Saha
Davin Muramoto
Kenny Kim
Robert Gaines

Table of Contents

1 Introduction

Our goal for this final project is to create a captivating arcade experience with a thrilling rendition of the game Stacker. Our ultimate aim is to transport users into the nostalgic ambiance of an arcade, providing them with an exceptional and enjoyable time.

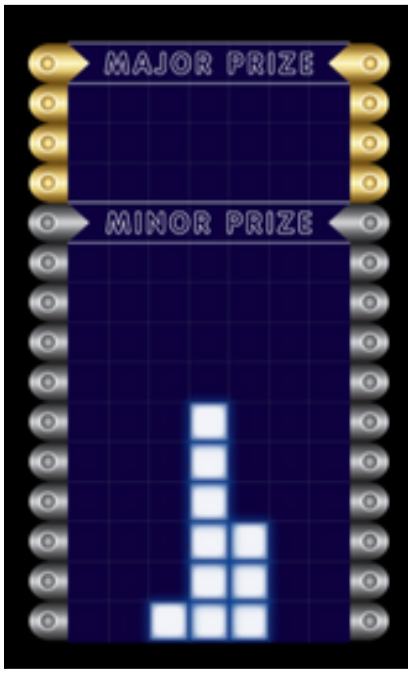

Fig.1. The arcade game Stacker

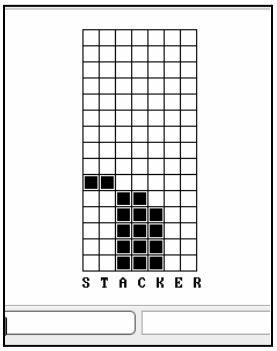

Fig.2. Demo of the gameplay of Stacker

The game starts with a blank screen and no squares. A row of blocks (3x1) appears and oscillates left and right. Every time the block hits a wall, a different tone is played out. The player's goal is to press the input button, the PING sensor, to stack the blocks at the correct position on top of the previous row. The next row above it then appears until the player reaches the topmost row. If any blocks are not aligned perfectly, they will fall and be lost, accompanied by a falling tone. The tower's width adjusts according to the lost blocks, requiring the player to match the new width. As the tower grows taller, the speed of the oscillating rows increases to increase the difficulty.

To summarize the basics of our game,

1. **Objective**: Get to the top of the screen.

2. **Challenge**: speed of oscillations increases as height does, blocks with no support will fall and be lost, and players will need to continue with fewer blocks for each subsequent level.
3. **Inputs**: PING sensor
4. **Outputs**: Arcade sounds and light animation with the LED array

2 Background

Using the PIC32 microcontroller used in class along with the capacitive breakout touch board, PING sensor, and a custom-built LED array and Game Rig, we built our physical arcade machine. To explain the basic mechanics of our arcade machine we will break it down into different sections.

Hardware

For the hardware, we essentially started by making the LED arrays. We chose to use RGB LEDs, but only used the red pin as we didn't have the time to implement all the colors. By soldering all the grounds together for an array, we could connect all the grounds for the entire game and connect it all together. From there, we soldered all the red LED pins to a long wire which was soldered to a connector header, and that was then connected to the shift registers for each array.

For each of the sensors, we mostly just used the same circuit from the labs. For example, the ultrasonic distance sensor was connected to the OC pin 34 for the echo, as it was in the lab that we used it. For the capacitive touch sensor, we connected it to an AD pin, as it was exactly in a prior lab. We wanted to use the same circuit as it would just streamline it and make it easy for us to use. For the capacitive touch sensor, this involved using the 1MΩ resistor, as it was in the lab.

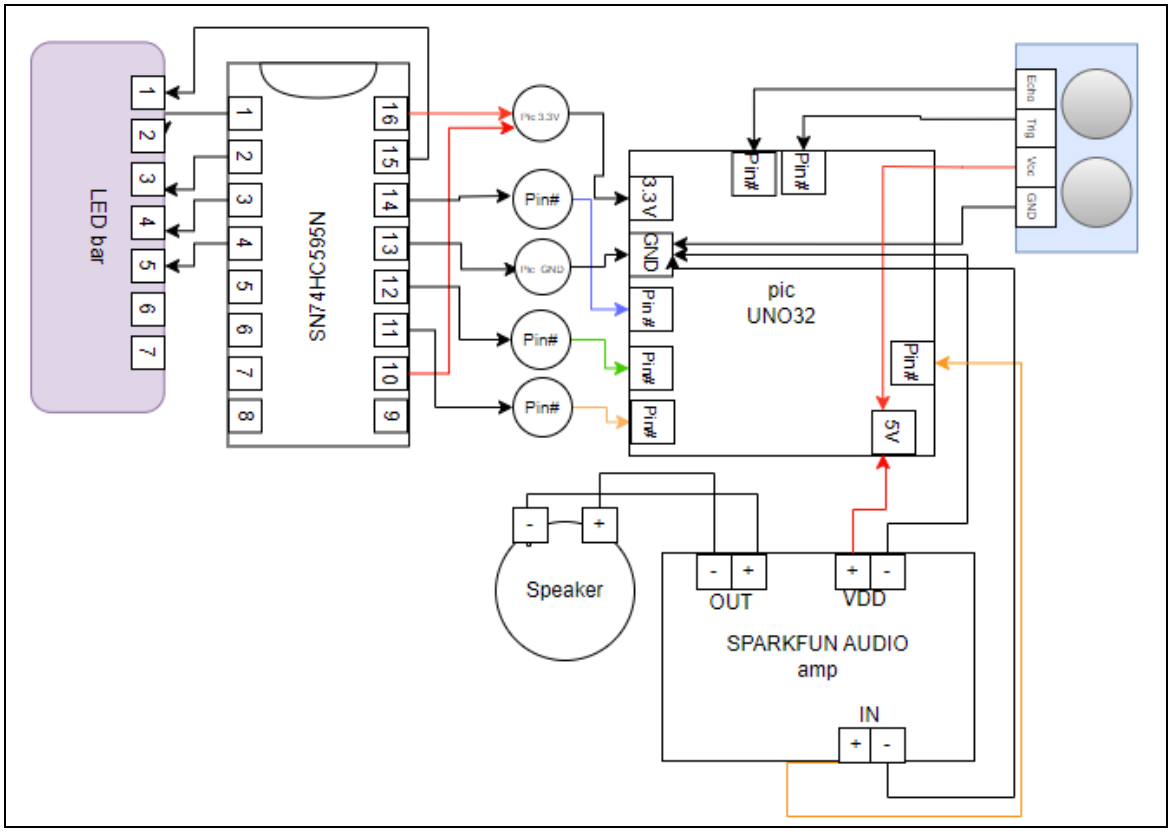

Fig. 3. Wiring diagram of the circuit

Software

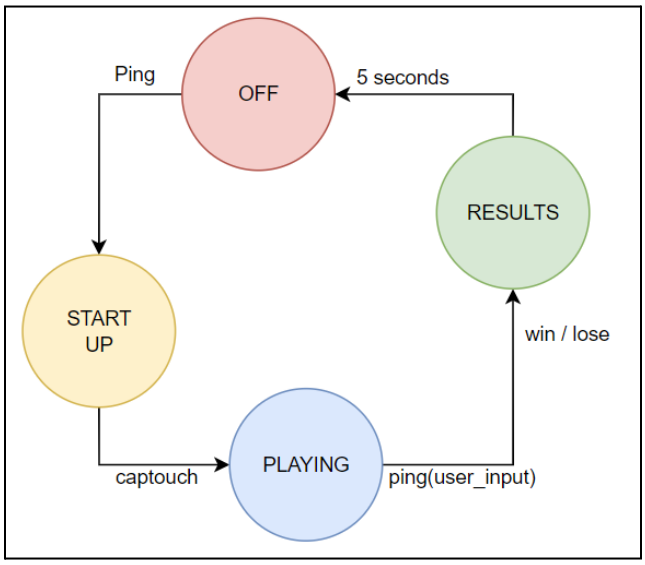

Fig.4. State Machine for the gameplay

The figure above displays a state machine representing the entire code that will be flashed onto the microcontroller. The initial state of this machine is the OFF state, where the user must walk in front of the ping sensor to trigger a state change to the START_UP state. In this state, the

code plays a song and displays "ECE 167" on the LED game board. An arrow is then shown on the LEDs, indicating to the user to touch the capacitive touch sensor to begin playing the game. Once the user touches the sensor, a state transition occurs to the playing state.

The playing state is extensive and functions as a state machine within itself. It controls the game flow, managing factors such as the number of lives (blocks bouncing back and forth) the user has and the number of levels completed. Additionally, this state determines whether the game board should display blocks stopping, falling, or bouncing, based on user inputs and specific conditions. Ultimately, this state runs the logic for the simple brick stacker arcade game.

The playing state is left once a win or loss is detected. A win is detected if the user reaches the maximum level, while a loss occurs if all the lives (blocks) have fallen. The transitioned state is the RESULT state, where the LEDs display either a "W" with a happy song for a win or an "L" with a sad song for a loss. Afterward, the player is returned to the OFF state, requiring the ping sensor to be triggered again in order to start a new game.

Overall, this state machine structure ensures the proper sequencing and control of game states, enabling a smooth and interactive gameplay experience.

Housing

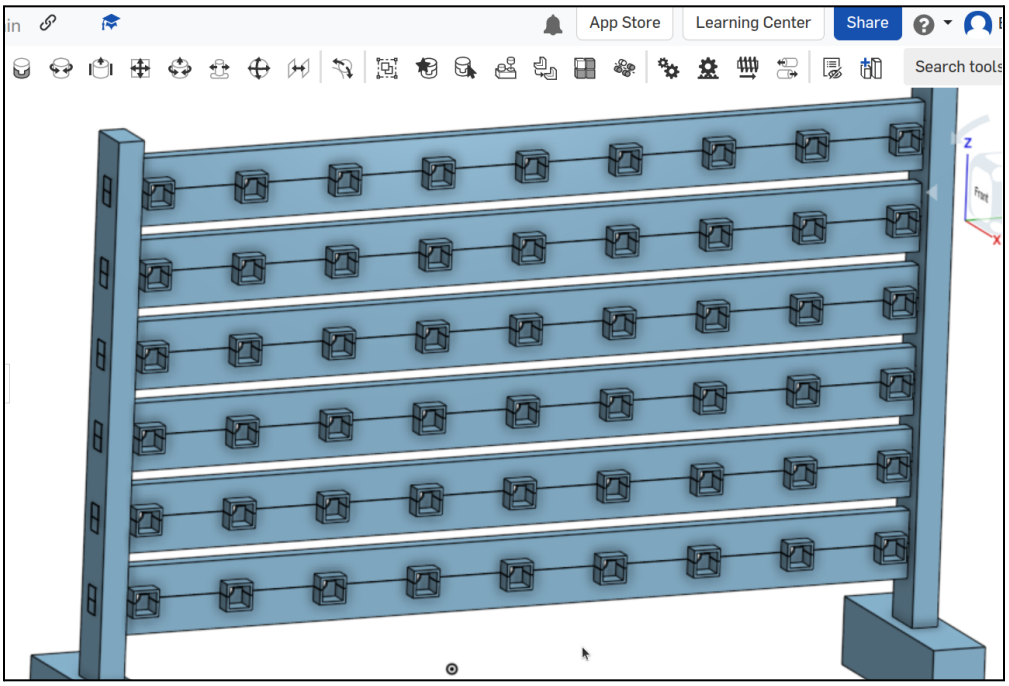

Fig. 5. CAD diagram of the 3D housing - 1st design

For the mechanical design of the project we started with a design that used friction fit pegs to hold the led array bars in place. After initial print and assembly of this design, structural support

and practical use was low, the prints would break along the weak vertical walls of the print, which is attributed to the design choice of speeding up print time while sacrificing support. Instead, for the second iteration the design was a dual T-Channel slot for the led bars to slide into. Using One solid print as the base with the T-Channels Provided greater support and ease of access, which proved useful because taking out rows and debugging specific leds was a task we needed to complete multiple times.

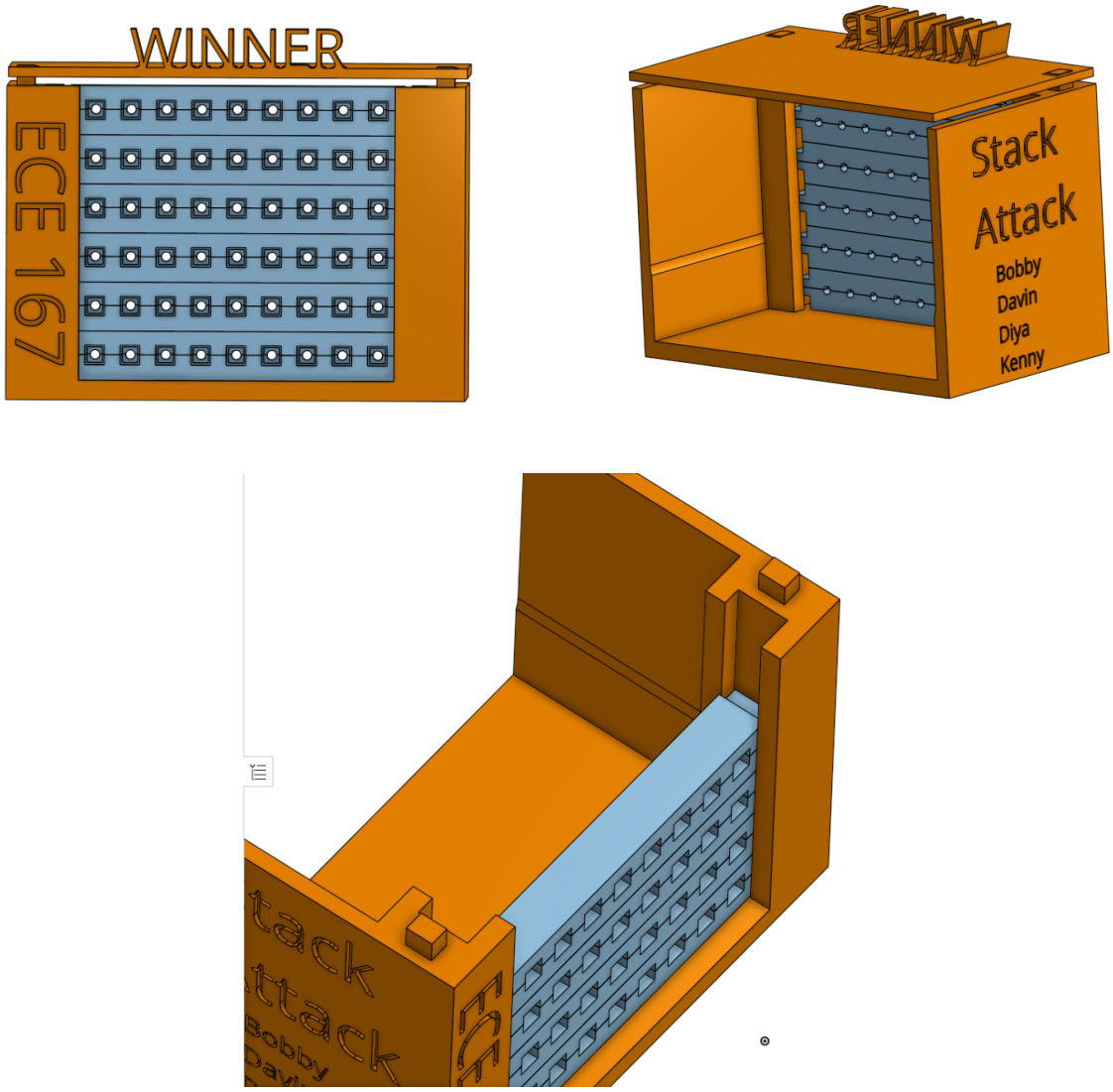

Fig.5.The final rendition of the 3D housing

For the first draft of the LED game housing, we created rectangular housings where the LEDs would fit, giving it a square appearance when illuminated. However, we encountered a challenge regarding the limited number of IO pins available to control the LEDs. To overcome this issue, we needed to design a simplification method to minimize the number of pins required for each

row. We decided to reduce the number of squares to 8 and utilized a shift register for assistance. By employing a SN74HC595N shift register, we managed to control each LED bar using only 3 bits (IO signals). The pinout for a single shift register is illustrated in Figure 6, and the wiring diagram for a single LED bar with the shift register is shown in Figure 7.

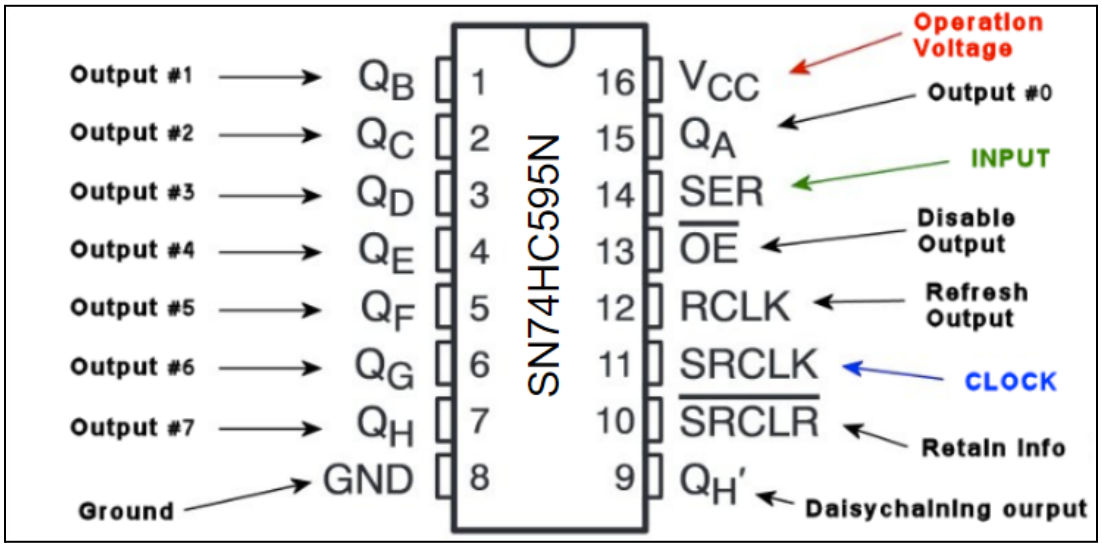

Fig. 6. Pinout for the shift register

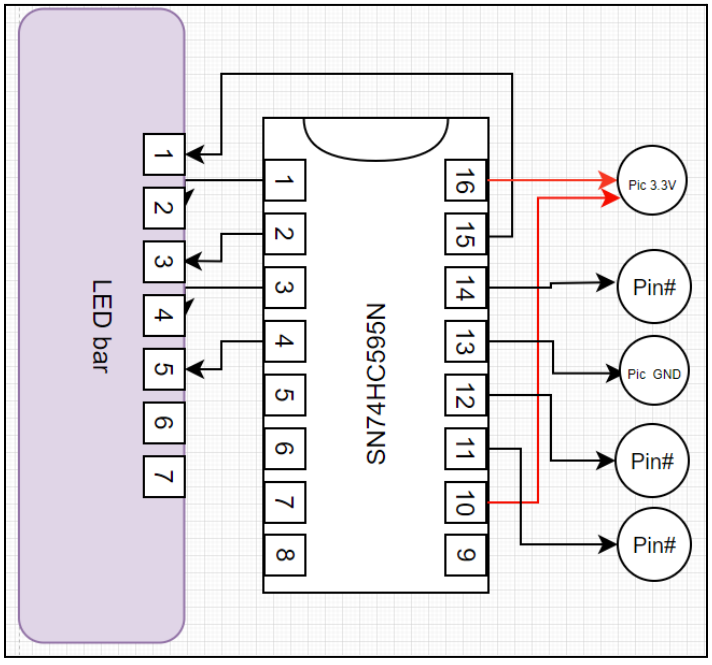

Fig. 7. Wiring diagram for each individual shift register

3 Evaluation

To test and verify the proper functioning of our machine, we conducted a series of small unit tests. Before initiating our project, we decided to test all of our different hardware components.

1. PING test

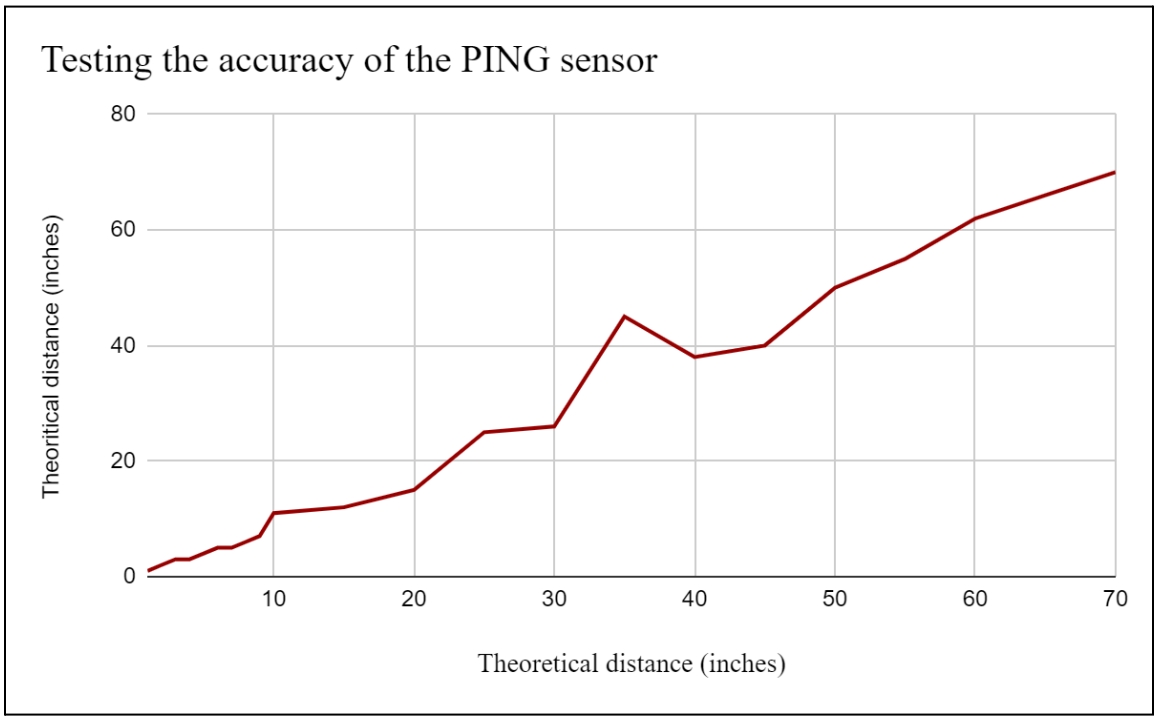

Fig.7. Graph to test the accuracy of the distance of the PING Sensor

To test the PING sensors, we utilized MPLabX to print out the distances, which helped us determine the appropriate range for user input. We conducted sensitivity tests at various distances and decided to establish two distinct ranges. One range was designated for starting up the arcade machine, encompassing larger distances, while the other range was intended for recording user input during the game mechanics, involving smaller distances.

To establish the two distances, we performed tests to assess the theoretical and experimental accuracy of the PING sensor. As depicted in Figure 7, the results showed that the measurements were highly accurate at closer distances. However, as the distances increased, a slight increase in error became evident in the graph. Consequently, we chose 70 inches as the starting distance for powering up the arcade machine and 10 inches for capturing user input during the game.

After successfully implementing this mechanism, we printed out "slapped" or "not slapped" when the user interacted by slapping in front of the PING sensor. Once this test proved successful, we integrated this mechanism into our gameplay, eventually gaining control over falling bricks. The conducted tests and subsequent implementation of the PING sensor ensured accurate distance measurements and facilitated interactive gameplay for the arcade machine.

2. Capacitive touch breakout board test

One such component we tested was the cap touch, where our objective was to ensure that we could obtain a distinguishable reading each time the cap touch was touched. For this purpose, we connected the cap touch to an oscilloscope to obtain the readings.

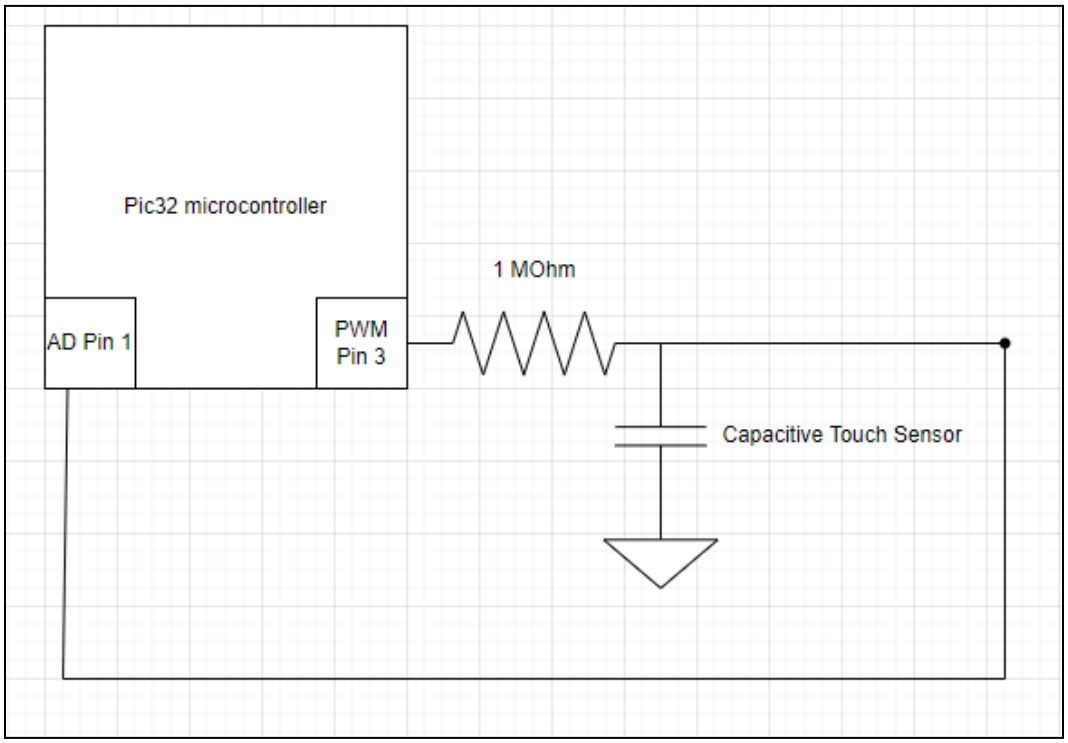

Fig. 8. Measuring the touch sensitivity of the cap touch

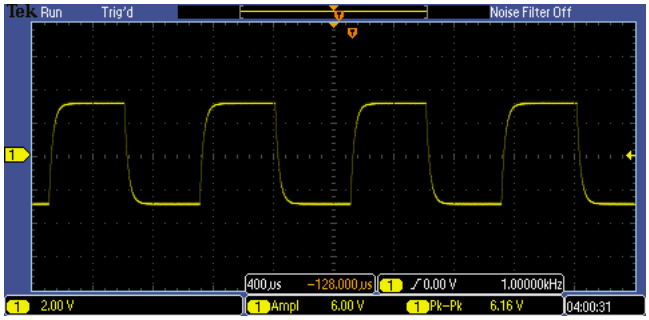

Fig.7. Captouch untouched tested with 1MOhms

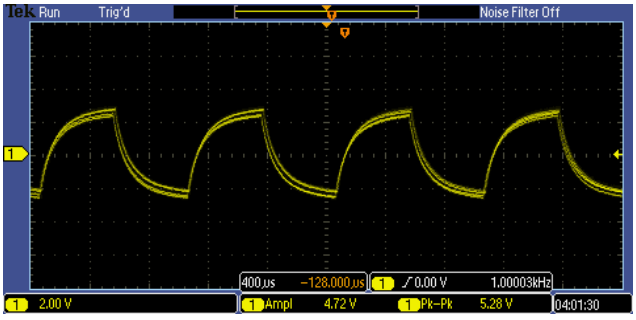

Fig.8. Captouch touched tested with 1MOhms

As you can see in Figure 7 the difference between the two oscilloscope readings was very distinguishable and we were able to get a distinct reading.

3. Software Testing

```
0 0 0 1 1 1 0 0 0 0
0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 1 1 1 0 0
0 0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 1 1 1
0 0 0 0 0 0 1 1 1 0
0 0 0 0 0 1 1 1 0 0
0 0 0 0 1 1 1 0 0 0
0 0 0 1 1 1 0 0 0 0
0 0 1 1 1 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 0
0 0 0 1 1 1 0 0 0 0
0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 1 1 1 0 0
0 0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 1 1 1
0 0 0 0 0 0 1 1 1 0
```

Fig.9. Testing out the logic for the LEDs

One aspect of the software that we tested was developing logic for a simple oscillation of 1s in an integer array. In Figure 9, you can observe the illuminated LEDs represented by the ones in a row. Each row signifies a distinct step for the LEDs when a 3x1 block is in motion. The code operates by utilizing an integer array, the size of which depends on the number of LEDs in a single row. Within this array, we establish a starting point (in this case, its index 3, 4, and 5)

and a designated direction to initiate movement. The block is then shifted by one index, and we check if it encounters a boundary (either the left or right wall). If a wall is reached, the direction is altered. If no wall is hit, the block progresses in that particular direction. Regarding the implementation of this software into our game, we are currently deliberating between two options. The first option involves establishing a direct connection to each LED on the game board. The second option entails employing some form of logic to control a single input, which would subsequently control an entire block (either 2x1 or 3x1). We believe that the second option may be a more sophisticated and logical choice. However, if we are unable to make it work, we acknowledge that directly driving each LED based on the integer array will suffice.

Aside from just this simple oscillation. We needed to come up with a viable plan/test for how we were going to get multiple integer arrays to work in tandem and display them on the LED game board. We decided the most effective and simple way for going about this was to manage the game using a 2D array and then have a loadArray function that will load the 2D array at any time to visualize it on the LEDs.

The way we went around testing if this was a viable method was to check if the method first worked and if there was any latency that would make our game unplayable. To check if the method worked, we loaded various different 2D arrays onto the LED game screen using the created loadArray function. We knew the method would work if the correct array appeared in the LED screen given there are no potential hardware disturbances. To test latency we used the START UP state animation. Basically we displayed each letter/number with various delays in between to see how fast the loadArray function would work and if we can create a slow animation like the ECE167 animation and a fast animation like the overall gameplay of the game.

Overall, this plan turned out to be a success with no drawbacks from the start. At first, I was skeptical of how accurate and fast the LEDS would update and how smoothly the animations would work. In the end, this was the core thing that we needed to test on the software side as creating the game using just software and not integrating hardware is pretty straightforward.

4. Gameplay Testing

We conducted tests to assess the simplicity of gameplay, aiming to ensure that users could play the game without requiring a complex set of instructions. Several students were asked to play the game without prior knowledge as part of the testing process. Initially, many players were confused about whether the game had started, prompting us to incorporate LED animations that indicated the game was starting up. Another challenge encountered was that players were unsure about the location of the sensors. To address this, we hot glued the sensor to the side of the housing and displayed an arrow pointing towards it, providing clear visual cues for the players. Furthermore, during the early stages of testing, the fallen blocks were not being erased, leading to confusion among players regarding the stacking of blocks. To rectify this, we implemented the functionality for the fallen blocks to disappear, accompanied by a sound effect, ensuring that users were aware when blocks were properly stacked. Upon testing this out, as we can see in Figure 10, 14 out of 15 users were able to navigate the game play without

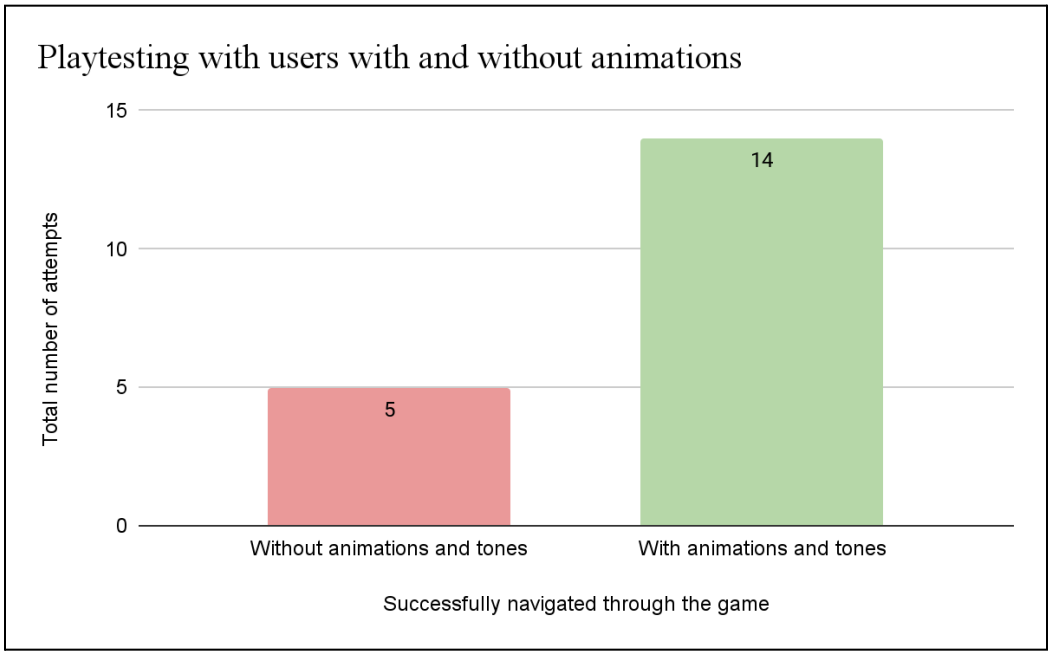

Fig.10. User feedback with and without LED animations and tones

Additionally, we wanted to ensure smooth transitions in the state machine. To achieve this, we ran through the gameplay, monitoring various bits of information to ensure that everything was functioning properly. Figure 3 depicts the thorough testing conducted on the state machine, indicating that transitions were smooth. Initially, the state machine encountered issues in transitioning correctly from the RESULTS state back to the OFF state, resulting in the array

not being reset. To resolve this, we implemented certain actions during the transition as well as within the state itself, ensuring comprehensive control over the game's behavior.

By conducting these tests and incorporating necessary improvements, we aimed to create a user-friendly gameplay experience and ensure the smooth operation of the state machine.

4 Discussion and Conclusion

In conclusion, we divided the tasks among team members to enable parallel work on different aspects of the project. We utilized a Gantt chart to outline our progress and created individual to-do lists for each person. This approach allowed us to stay on track and simultaneously test multiple functional elements. While the project initially appeared complex, we successfully broke it down into different modules, addressing small tasks incrementally. As a result, we were able to accomplish most of our intended goals for the user experience, including engaging visuals, responsive controls, difficulty progression, and a rewarding gameplay experience.

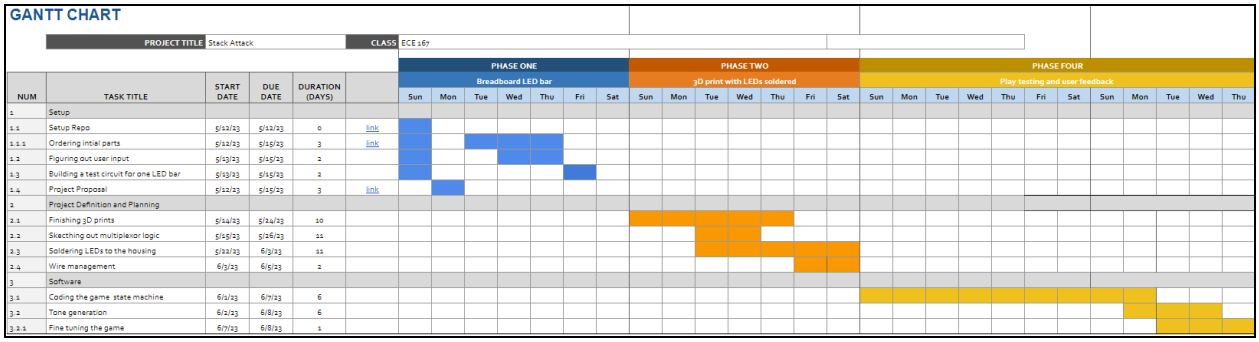

Fig.11. Testing out the logic for the LED

Some specific shortcomings and limitations that we ran into were mostly hardware based. We wanted to use certain sensors but they were giving us trouble as they wouldn't respond or wouldn't react to our inputs to transition to a new state. Another limitation was our prize dispensing. If given more time, we would've added a mechanic to dispense a prize based on whether or not the user wins. This would've been achieved through the use of solenoids or stepper motors. We initially wanted to use solenoids but we ran into an issue in that the solenoids we found required 12V DC, which we didn't have access to, so we just scrapped the idea. Another limitation was the number of LED bars. If we added more levels, the game would've been more engaging and fun to play, but we didn't have any more breadboards/shift registers. If given the opportunity, we'd probably add 6 more levels.

We achieved engaging visuals through the implementation of an LED bar and captivating light animations. Responsive controls were ensured by incorporating the PING sensor. As the user progressed through each row, we successfully increased the speed of the moving bricks to enhance the difficulty progression. Furthermore, we aimed to integrate solenoids and a prize dispensing mechanism using a flex sensor, allowing users to select their desired prize and creating a truly rewarding gameplay experience. Overall, our meticulous planning and collaborative efforts enabled us to deliver an immersive and enjoyable arcade experience, meeting the majority of our desired objectives.