T03 Project Part 2 Report

ECE 102

Dmitrii Fotin (dfotin@pdx.edu)

Ryan Blackwood (rblack2@pdx.edu)

Ghanem Taher (gtaher@pdx.edu)

Celina Wong (wcelina@pdx.edu)

Introduction

The goal of the projects is to design a memory skill Simon game, which plays a sequence of tones and lights and prompts the player to repeat the pattern by pressing corresponding buttons of different colors. The game uses the Trinket M0 as the microcontroller for all aspects of gameplay.

Final Design

The final design of the game includes a 3D-printed case that will contain all circuit components as listed in *Table 1.1*. The 3D-printed buttons are custom colored and made from clear, red, blue, yellow, and black resin dye. The case features a RESET button that will start or end the game and an ON/OFF switch that can cut the power from the Trinket M0 to the breadboard to prevent a constant draw of power through the circuit. Software features include a timeout function that would turn the game off if a player is inactive at any point, level of difficulty selected directly with the buttons as opposed to serial console, voltage divider to produce unique analog inputs for button presses and a demultiplexer to help control the LEDs in spit of the limit of the I/O pins.

Figure 1.1. 3-D models illustrating the initial vs final designs & final physical model.

Hardware

Circuit

The critical components of this device include a demultiplexer, a voltage divider and a microcontroller compatible with CircuitPython, M0 Trinket. As shown in *Figure 1.2*, The SN74HC138N IC demultiplexer is used to switch from three common data input lines (signals from microcontroller pins) to four output data lines (LEDs). A voltage divider consisting of six different resistors allows to bridge all push buttons to one pin on the microcontroller with a unique analog input per resistor/button, which is then processed by the microcontroller to

determine which resistor lets the signal through and returns a corresponding variable value for further use within the code.

The full list of materials are shown in *Table 1.1* below.

Bill of Materials		
	Quantity	**Item Name**
1.	4	Push buttons
2.	1	SN74HC138N IC
3.	4	LEDs (red, blue, yellow, and green)
4.	1	Adafruit Trinket M0 microcontroller
5.	1	Piezoelectric speaker
6.	10	Resistors: 20 kΩ (x1), 10 kΩ (x1), 5 kΩ (x1), 2 kΩ (x1), 1 kΩ (x1), 327 Ω (x4), 10 Ω (x1)

Table 1.1. List of parts used.

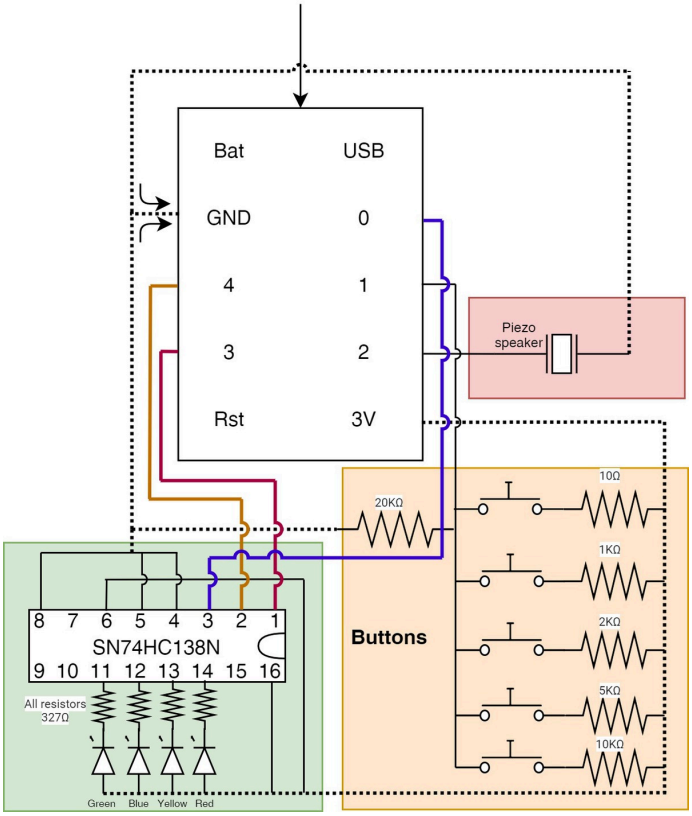

Figure 1.2. Diagram of the circuit and connections to Trinket M0.

Housing

The housing design was based off of the “pocket” version of the game [1].

Figure 2.1. Classic Simon Pocket

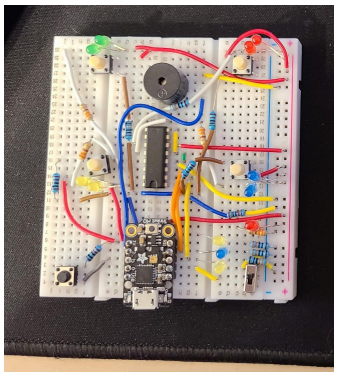

Figure 2.2. (2) Half Breadboards

This was designed around (2) half breadboards with a single power rail to keep the size small enough to print in a single piece, and to cut down on some of the print time and material cost. This ended up making it almost the perfect size for a handheld game, and is comfortable in the hand as the buttons are well sized and laid out.

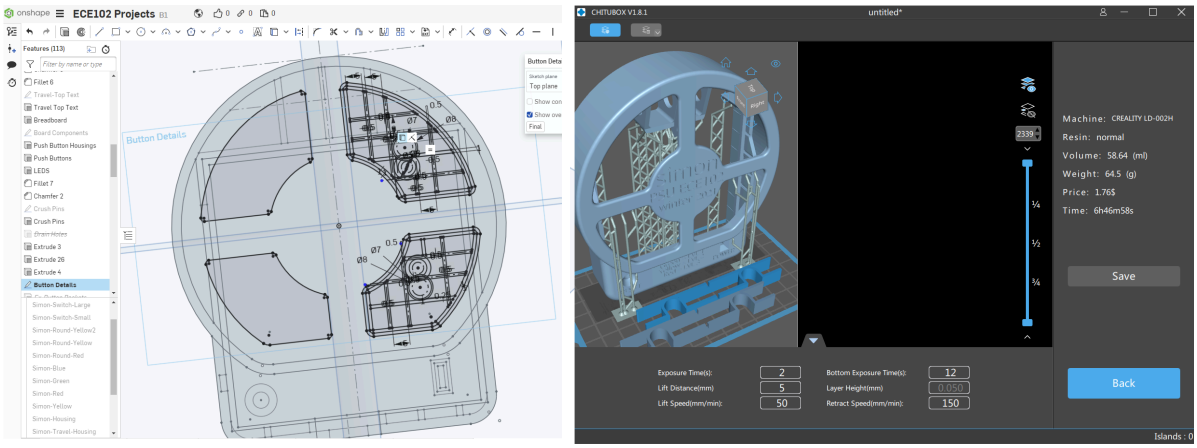

Figure 2.3. 3-D modeling & printing software

Once the main shape and size were determined, a recess was designed into the bottom to hold the breadboard in place, allowing all of the components to fit inside the housing. The breadboard was modeled, along with its main components (switches, LEDs, trinket, and buzzer), while designing its recess. This allowed the design to continue around those components and account for their positions for all components and revisions.

The next step was to design the buttons. Initially it was thought that they would need a spring mechanism (either 3D Printed, or an actual metal spring). However, after designing the buttons fairly thin and light, the actual mechanical buttons were strong enough to not need another

spring mechanism. Along with the buttons being thin to reduce weight and cost, it allowed the LEDs to shine through the material much brighter.

The buttons are held in place with 2 removable pieces, which do not interfere with the button operation, even when the breadboard is not installed. The pieces were designed to remain clear of the buttons, LEDs, and buzzer.

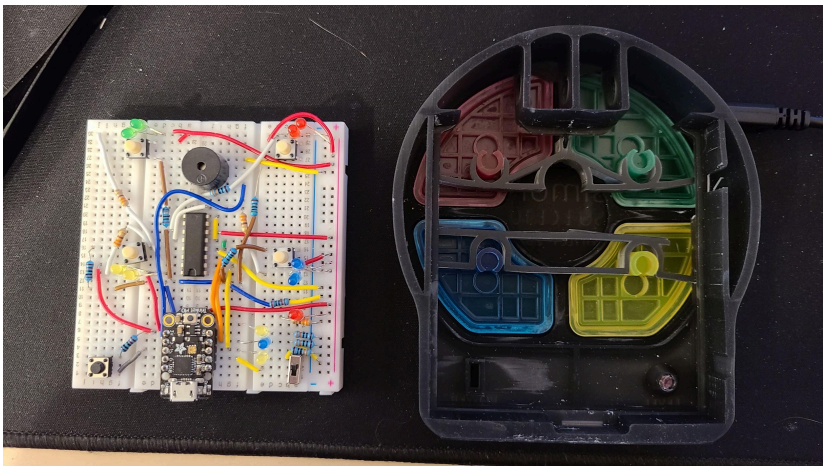

Figure 2.4. Final printed model with buttons & clips and final Breadboard circuit.

Initially there was an opening for the reset button on the Trinket, which could be pressed with a paperclip to reset the trinket itself, as well as an opening for a power switch. The power switch was to turn the power from a battery on and off. To save time and reduce complexity, the team decided against using a battery, opting to use a physical switch to disconnect power flow between the trinket and the rest of the circuit. This keeps the LEDs off, and reduces the power consumption. The power button may not be easily accessible, but it's possible to design a part to resolve that.

The team determined there was no need for the reset button on the trinket itself, and instead added a separate reset button to end and restart the main program. As the reset button is needed to play the game, a pin was printed and glued into the opening in the housing with flexible glue, so it is attached to the housing, but would move when pressed, and press down on the reset button on the breadboard below. Other than that, the entire assembly goes together with no fasteners or adhesives.

Initial Print: Standard grey prototyping resin for fit testing of buttons and breadboard. The buttons ended up being too tight. (illustrated in Figure 2.5, which has no internal components)

Figure 2.5. Initial printed model for fit testing

Revision 1: More gap around the 4 colored buttons, the first set to print ended up being too tight and would get stuck when pushed down.

Revision 2: Added a space in the clip that holds the Red & Green Buttons, to allow for the buzzer. A smaller buzzer had to be used, as the one that came with the Trinket kit would have required the housing to be almost 1” thicker.

Revision 2.5: Adjusted some internal dimensions and printed just the bottom quarter to check the fit of the breadboard and Trinket.

Revision 3: Initially the game had a level button, but it seemed better to have the colored buttons select the level, rather than press the same button multiple times. So LEVEL was changed to RESET, some internal dimensions were also tweaked to hold the breadboard tighter.

Software

The program is designed to produce a Simon game logic to work with the circuit diagram in *Figure 1.4*. The game’s difficulty selection loop starts when the reset button is pressed. Then, the program waits for a level difficulty signal to proceed into the game based on the level selected. The game ends if a level is not selected in ten seconds.

When the game starts, a timer is set so that the game ends if there is no activity from the user in three seconds after the generated sequence is played, or after the last button was pressed. The game then generates a random number, which prompts a respective LED to light up and the Piezo to play an assigned tone. This number is appended to the generated sequence list. Once the game finishes playing the sequence, the program initiates a user input loop that listens to the analog signal from a button press and assigns a number corresponding to each analog value/button press. This is then appended to the player sequence list.

If both the input sequence and the generated sequence match and the level of difficulty is not beaten yet, then the game proceeds to the next round of generating the next sequence. If the

sequence doesn't match, the user loses the game, the Piezo plays a “buzzer” sound, and the program breaks and returns to the beginning of the main code for the player to press the “restart” start button. If the length of the sequence reaches the limit depending on the selected level, the program produces a victory sequence of LED and the Piezo plays a “winning” tune. The flowchart in *Figure 1.5* below illustrates the overall structure of the software.

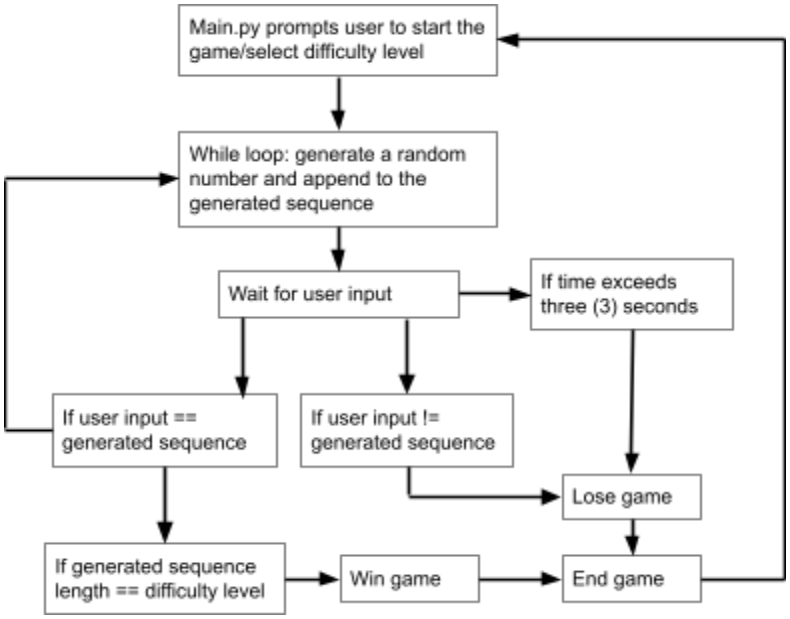

Figure 1.5. Flowchart of overall structure of program.

Problems & Solutions

Circuit

Connecting LEDs to the SN74HC138N demux presented a challenge in the early design, as the demux selected output is Low. The problem was resolved by connecting the cathode of the LEDs to 3 V instead, and connecting the anode to the SN74HC138N demux, which allowed the 3 V flow through to the pin that was selected Low on the demux, lighting up a respective LED on the way.

Code

The most complicated parts of the code were ensuring proper overall logic of win/lose conditions and organizing the steps in a way that would execute per project specifications, as well as implementing the timeout function, as the control points needed to be set up right before entering input loops but had to be updated after each input to avoid timing out prematurely. Implementing functions to minimize the main loop code, naming variables clearly and adding annotations helped with organizing the code. Frequent testing after any minor changes to the code was critical for resolving logic and sequence errors without introducing new bugs.

The implementation of the timer first required extensive research on how the time functions operated specifically in CircuitPython. Through collecting information bit-by-bit from numerous websites [2] and videos [5], the understanding formed was that initiating a variable equal to time.monotonic() collects the current time in seconds. To stop the program after a certain amount of time, we can set the time function in the function that awaits user input then subtract it from its initial time collected at the start. From here, we can “point” the program to go where we would like, and in this case, “lose” the game.

Housing and 3D model

The main problem of the housing was that we needed to have a pretty good idea of the final design before making a housing. If any additional switches, lights, or large components are added to the project it would require modifying the design and printing it again. Producing the housing also took more time than anticipated, and ended up taking 3 revisions in order to get everything to fit properly, each revision taking close to 12 hours to print. Amazingly, there wasn’t a single print failure during this process. Usually there is close to a 30% failure rate, but there were a total of 12 print jobs for this project, and they were all successful.

Discussion

The biggest takeaway was learning how to approach the project design from determining project specifications, such as the limitations on I/O pins, analyzing key components, in particular, the resistors to use for the voltage divider, to coming up with ideas, identifying solutions, and organizing project tasks.

Another important lesson from the second part of the project was efficient project management, effective communication and team work to ensure project success. Organizing tasks with Trello provided a clear visual reference for project status and helped keep the project on track. Weekly team meetings allowed team members to brainstorm ideas, debug issues, check on progress, and to help each other if anyone had any issues or concerns.

From the technical perspective, learning how to create complex circuits with a limited number of I/O pins, implementing voltage dividers and demultiplexers, reverting demultiplexer logic to light up a specific LED (as opposed to turning it off due to default LOW signal), implementing functions to minimize repetitive code and 3D modeling the housing were some of the most enlightening aspects of the project.

Signature

Project Lead: Celina Wong
Software Lead: Dmitrii Fotin & Ghanem Taher
Hardware Lead: Ryan Blackwood
All of the students listed on this report have read it and agree with its content.

References

[1] “1980 Vintage Pocket Travel Simon Milton Bradley Handheld Game,” *Etsy*. [Online]. Available: https://www.etsy.com/listing/934563538/1980-vintage-pocket-travel-simon-milton. [Accessed: 15 Mar 2021].

[2] L. Ada, “Adafruit Trinket M0,” *Adafruit Learning System*. [Online]. Available: https://learn.adafruit.com/adafruit-trinket-m0-circuitpython-arduino/overview. [Accessed: 15 Mar 2021].

[3] D. Astels, “Arduino to CircuitPython,” *Adafruit Learning System*, 22-Oct-2018. [Online]. Available: https://learn.adafruit.com/arduino-to-circuitpython/time. [Accessed: 15-Mar-2021].

[4] Texas Instruments, “SNxHC138 3-Line To 8-Line Decoders/Demultiplexers,” [Online]. Available: https://www.ti.com/lit/ds/symlink/sn74hc138.pdf?ts=1615839004944&ref_url=https%253A%252F%252Fwww.google.com%252F. [Accessed 15 Mar 2021].

[5] TokyoEdtech, “Python Game Coding: How to Make a Game Timer,” *YouTube*, 20-May-2020. [Online]. Available: https://www.youtube.com/watch?v=juSH7hmYUGA&ab_channel=TokyoEdtech. [Accessed: 15-Mar-2021].