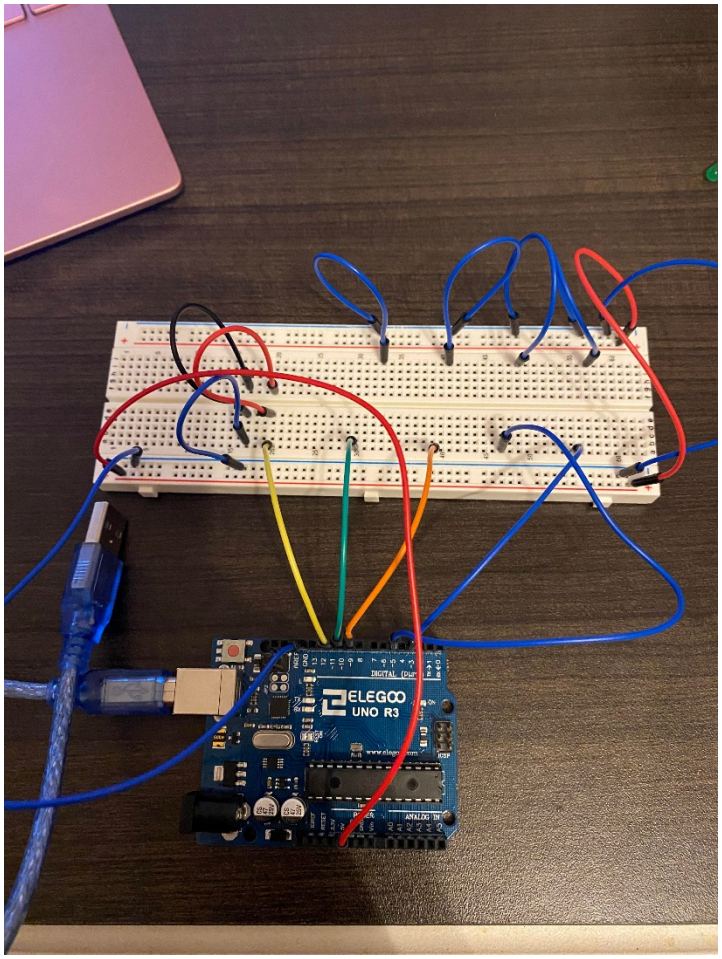
Cassandra Rousseau
Student ID: 40177594

November 19th, 2021
CART360— Tangible Media and Physical Computing

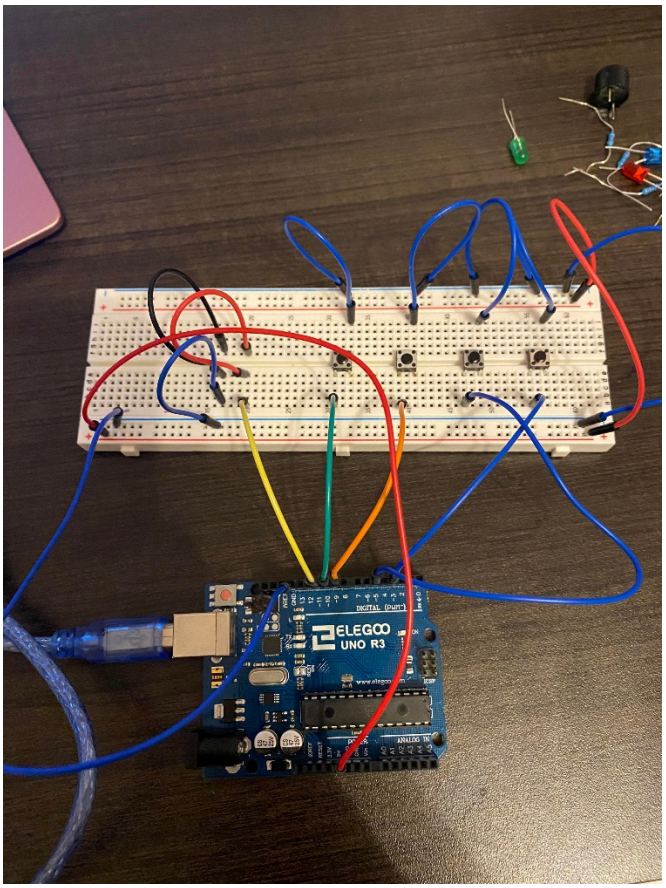
Etude 03 – Mind- It

Part One:

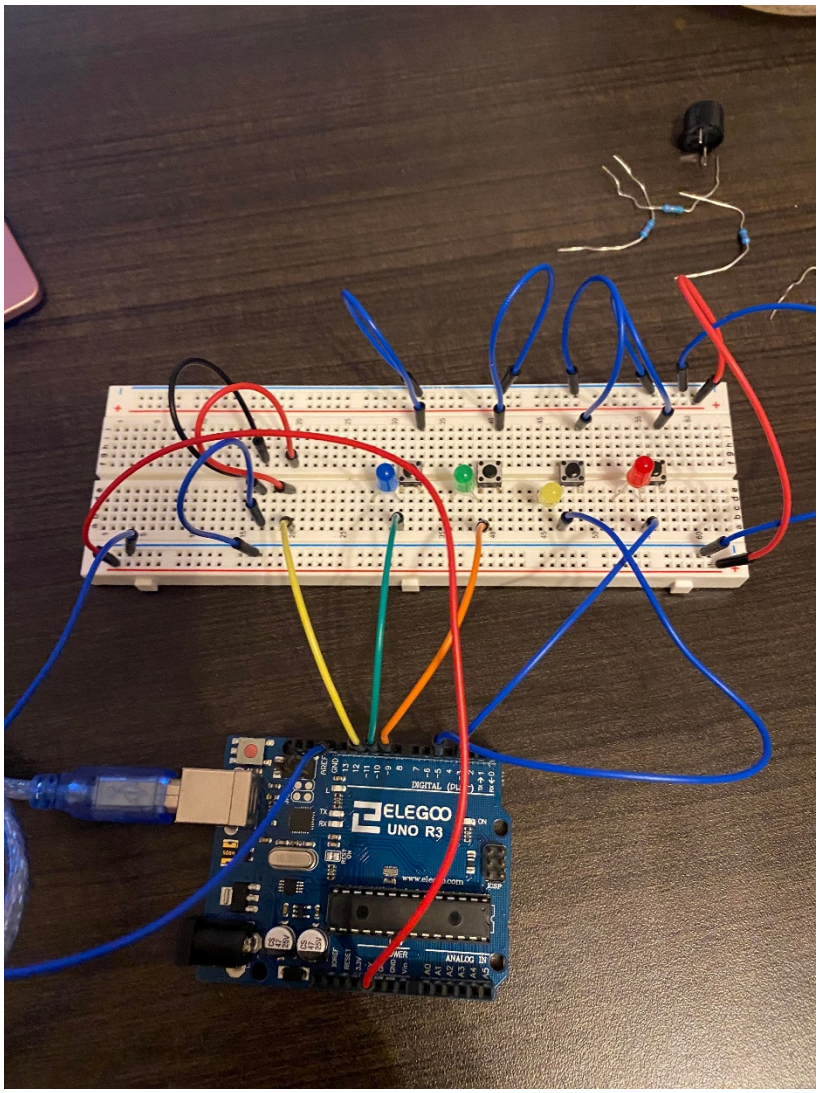
For this etude, it was not long to build the circuit. As a first step, I grouped the required components. I started to build the circuit based on both the Fritzing schematic and the picture of the final result of the circuit. I began by adding wires. Then I placed the buttons on the breadboard; I added the LEDs right after. I put the resistors. And finally, I finished by placing the Piezo buzzer on the breadboard. At the first trial, I noticed the Piezo buzzer did not work. After some checkup, I found out the wires were not connected to the buzzer, so I changed their positions. It worked at the second trial. However, the red LED connected to the Piezo buzzer only when I press its button but does not light up when the music sequence of the game is playing. I tried to change the components around this section, and it did not solve the issue. I look if this part of the circuit is different from the others, and there is no difference with other LEDs setups. I decided to start again by deconstructing the circuit and building it again, following the same steps. While analyzing the *Fritzing* circuit, I found out that I put my breadboard in the wrong orientation. After making it a second time and testing it, everything worked perfectly.



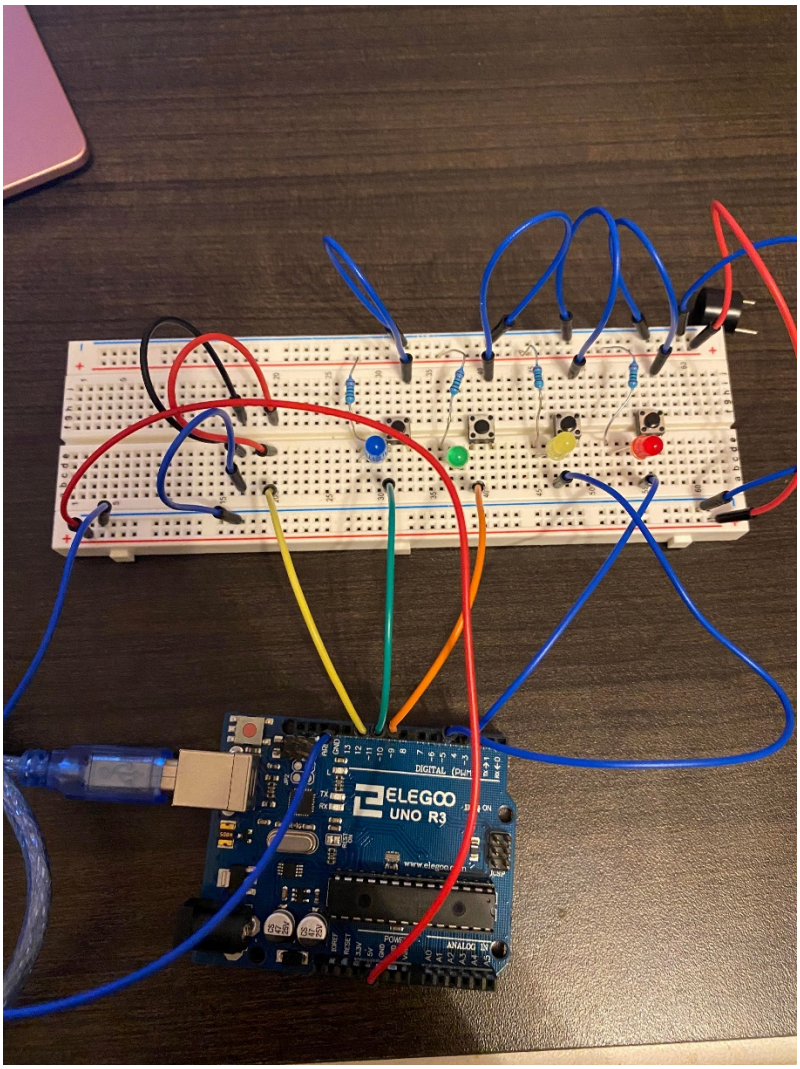
A picture of the second step with the working circuit: connecting wires to the breadboard.



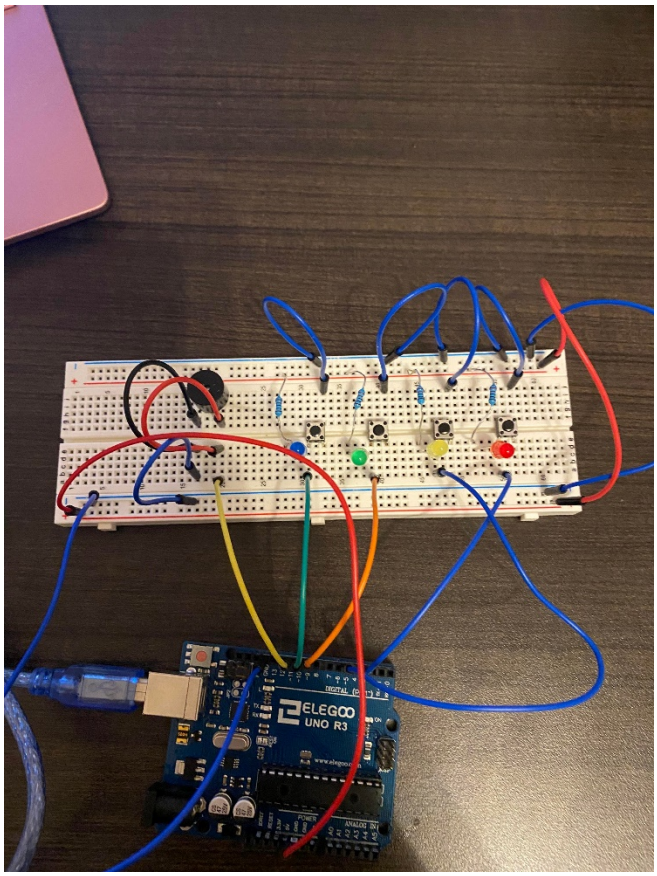
A picture of the third step with the working circuit: adding buttons to the breadboard.



A picture of the fourth step with the working circuit: adding LEDs to the breadboard.



A picture of the fifth step with the working circuit: connecting resistors to the breadboard.



A picture of the seventh step with the working circuit: connecting Piezo buzzer to the breadboard

I tested the circuit multiple times to figure out how the game works. After a few trials, I finally understood how the game works. This game aims to reproduce the asked music sequence generated by the Arduino on the circuit. The user needs to press the buttons connected to a specific LED to produce the generated sequence. The game works by rounds. A new note is added to the music sequence that you need to remember each round. If the user fails, the game starts all over again. As long as you get the sequence right, the game never ends.

Part Two A:

i)

1- The void setup sets all the components and the jingle in the circuit.
2- The game starts with a jingle implemented with the void *startUpLightAndSound.* This void plays each note implemented into each LED after a delay of 100 milliseconds.
3- In the void loop, a random music sequence is generated by a for loop.
4- Once the game sequence is set, the user needs to press a button to start the game. Once a button is pushed, the game starts. The first note plays, and the selected LED lights up; this action is implemented by the void *displayLightAndSound*, this void activates the LEDs and the Piezo buzzer. A timer generated by a for loop in the void loop will start too.

5- During the rounds, an if statement in the void loop checks whether or not the correct button sequence has been pressed before the timer ends. A Boolean called *checkButtonPush* will also check whether the right buttons have been pressed or not. Each time a button is pressed, the selected LED lights up, and a sound will come out of the Piezo buzzer.
6- If the timer ends before the user makes an action or presses the wrong button, the if statement in the void loop will activate a game over sound and reset the game, playing the introduction jingle again.
7- If the player made the correct music sequence, the music sequence repeats with a new note added to the melody. This new note is added by the integer *getButtonPush*.
8- The game will continue like this until the player fails. This endless gameplay is caused by the static integer *sequenceLength*.

ii)

The *startUpLightAndSound* void introduces the player to the game by playing a jingle. The void loop establishes the gameplay. It generates the random music sequence that the player will need to figure out, the rounds, timer, and the game over statement. The *checkButtonPush* Boolean verifies if the user's pressed button sequence is correct and gives the result of the current round. The integer *getButtonPush* plays the music sequence generated while adding a new note each round.

Part Two B:

i)

Three main structures are missing in the code: adding a high pitch display *LightAndSound* while the timer works, adding an integer that limits the number of rounds in the game and letting the first note of the generated game sequence plays right after *startUpLightAndSound* void ends.

ii)

At the beginning of the gameplay, nothing happens after the jingle plays, which may confuse the player. Making the first note plays after the jingle will help the player understands better how the game works. Adding a high-pitched *displayLightAndSound* during the rounds allows the user to understand better that the amount of time to repeat the asked sequence is limited. Creating a fixed number of rounds prevents the player from getting bored due to endless gameplay.

Part Three:

i)

The structure that maintains the gameplay is the void loop and the static *gameSeed*. The void loop activates the whole gameplay (i.e., generates the music sequence; sets the rounds states, the timer,

and the game over state.) The static *gameSeed* generates the new music sequence for each gameplay and resets the games if the user fails.

ii)

The void loop runs the program continuously, allowing the program to change and respond constantly. It sets and initializes the values in the program. Due to *gameSeed* being a long static variable, the generated values and sequence will stay the same as long as the gameplay goes. The amount of values stored in this variable is significant, allowing multiple possibilities for the music sequences.