Group Name: BUILD -A- BURGER

Group Mates: Elizabeth Villanueva, Conrad Ptasznik, Nicole Laczny

The Arduino project will consist of:

- 3 Arduino Uno's
- ~~21~~ 28 LEDs
  - Red, White, Yellow, Green, Orange (from special bought set)
- 7 Buttons for "Ingredients" + 1 Button for "Start Game"
- 1 Four digit eight segment tube (Keep score)
- Wires (Lots)
- Inactive Buzzer (new) (For music)
- Active Buzzer (Losing sound)
- 3 Color LED Module (new)
- ~~Speaker~~
- Resistors
- 16x2 Display (Displays info to the player + high score)
- Breadboards

Project Idea:

We will doing a puzzle using buttons, sounds, LED, and possibly the 4 7 segment. The idea is to build a "burger" in the correct order before the time runs out.

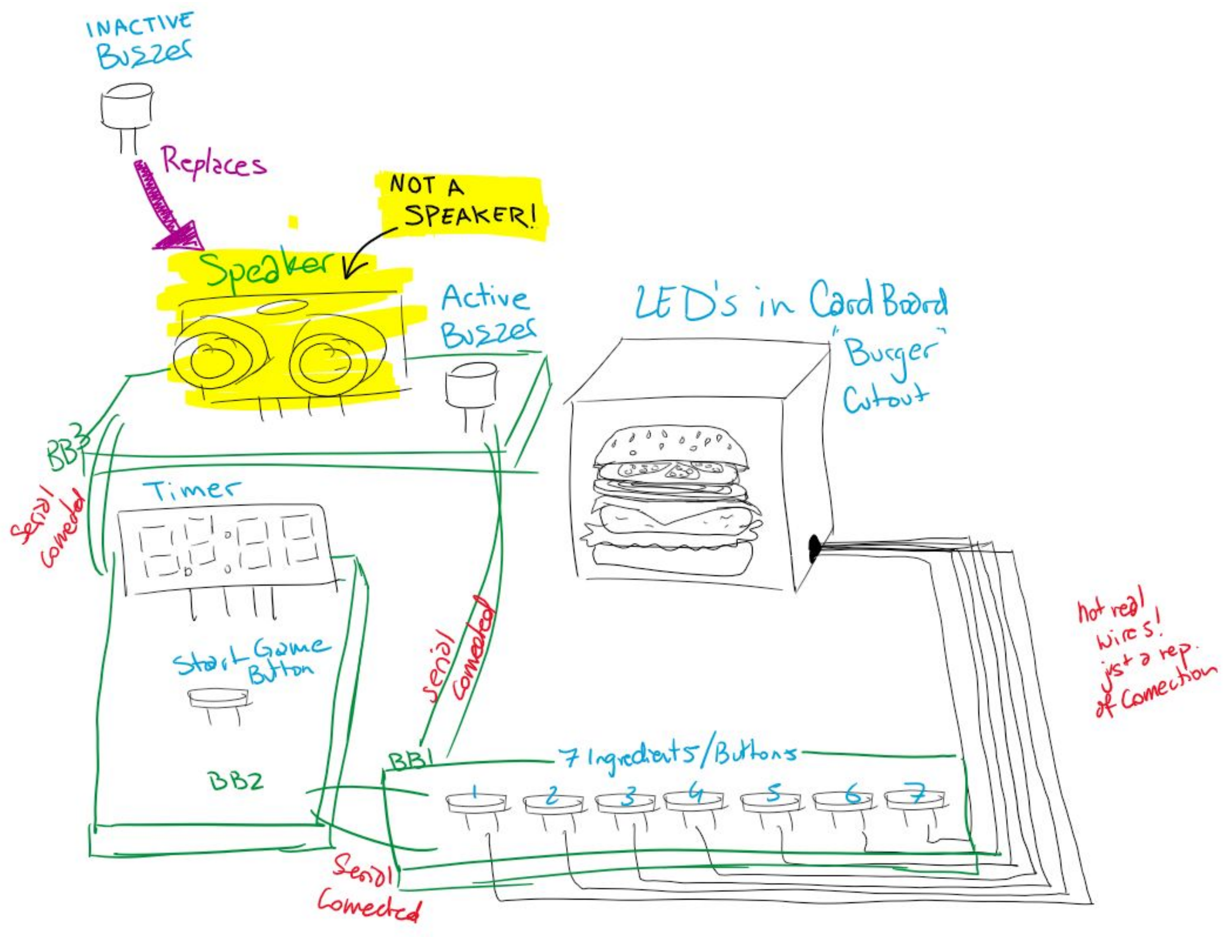
How it Works:

Each button adds a different ingredient/topping and for every correct ingredient added, the 4 LEDs, which are in a ~~double~~ row, a "layer" of LEDs will light up, and a positive sound plays (sound will be different for each ingredient, this will help clue in the player what button is what). If the wrong ingredient is added, a "negative" sound plays and the LEDs reset and the burger has to be rebuilt. To make this a more challenging puzzle, We will be making this a time challenge type of game where the 4 7 segment gives the user x minutes to solve the puzzle, and if the user doesn't finish by the time the timer goes down, the buttons/ingredients randomize, a fail melody will be played, and a new game starts. To make this fair there will be an extra button used to start the game.

Additionally, we will add a cardboard cutout to create a retro-burger that lights up. The ~~speaker~~ buzzer will be used to play out music. When the timer gets lower and lower the music gets faster and faster. A LCD board to display messages to the player such as if they got something wrong and tell the player "well done" if they got it right.

Resources:

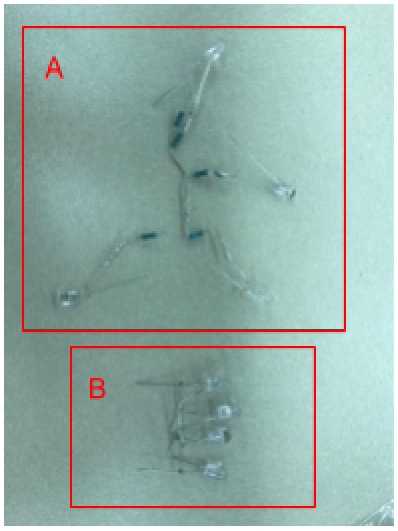
- The 3 unos will be connected using serial connections
  - https://www.youtube.com/watch?v=yQ15Hi1E7I4, - this is the main idea we used to connect our ardinos
  - https://www.arduino.cc/reference/en/language/functions/communication/serial/
- We will also need to buy different LEDs as well as build a case for the arduino. This can be done by either building one ourselves using cardboard or using a 3D printer to make it.
- Soldered together 3 LEDs to create a "level" that will go into the cardboard
  - ~~https://learn.adafruit.com/lets-put-leds-in-things/soldering~~ (not the correct idea for what our project needed). After our fail (in depth later), we used the idea of connecting LEDs from an LED cube: https://www.instructables.com/id/How-to-build-a-8x8x8-led-cube-English-version/
- Create music/buzzer sounds
  - https://www.arduino.cc/en/Tutorial/toneMelody



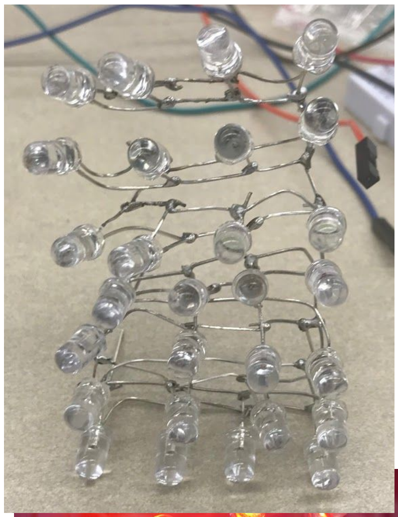
Future Plan:

WEEKS		PROGRESS
10/12	Two Page Write up is Due	complete
10/19	Have buttons working and able to shuffle	Incomplete (moved to later week)
10/26	Solder and Incorporate LEDs to projects	complete
11/5	4 Page Write up (Due 11/9)	Current - complete
11/12	Get individual parts working Connect arduinos	complete
11/19	Build "burger case"	complete
11/26	Error Testing (moved from 11/19) Video	complete

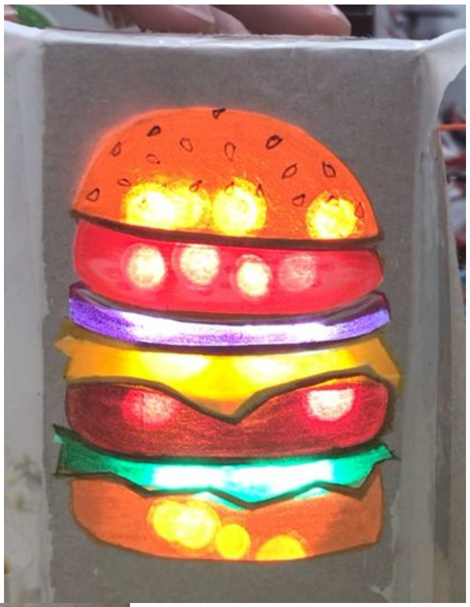
Job	**Progress**
Buttons	Done
Sounds	Done
Buzzer	Done
Timer	Done
LED’s (soldering)	Done
Game cutout/3D print design	Done
Connecting 3 Arduino’s	Done



We have soldered the LEDs into the shape and style of a burger, from this layout we can easily put it in a case to cleanly display the “layers” of the burger. Our first attempt was a mess since we attached resistors to each LED before soldering them to another LED (A) . This caused each LED to be different brightness which is not what we wanted. We then tried to solder all the positive sides together as well as all negatives together to see if that would work and give us the same level of brightness (B). This new design allowed all the connected LEDs to have the same brightness regardless of the amount of LEDs. So we replicated this design for all the “layers” of the burger (7 times). We then realized

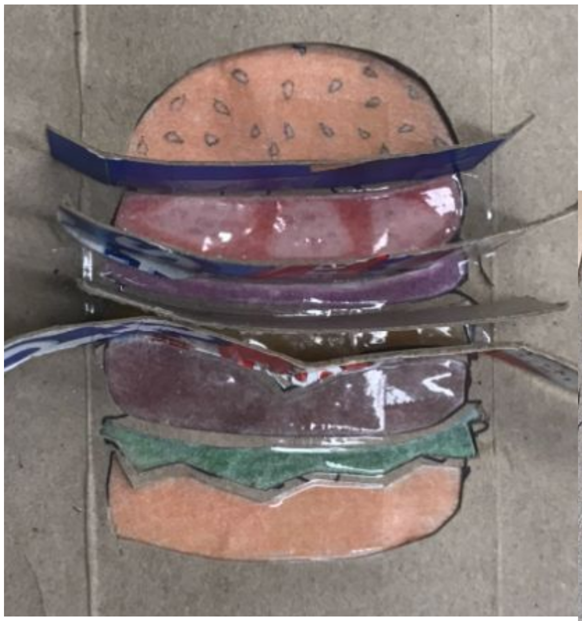


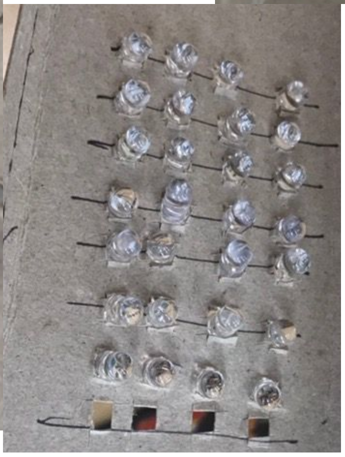
that each row would have a input and ground, causing there to lots of ground wires, so we soldered all the ground wires to each other leaving us with only 7 input wires and 1 ground wire. (A/N: We bought these clear LED's on Amazon thinking they had good color choices, but most of the colors were a disappointment - totally different than what was being advertised).



This was our final result:

To get this set up we had to get a good solid hold on the LEDs so that they would be held in place in the casing using a honeycomb style setup:

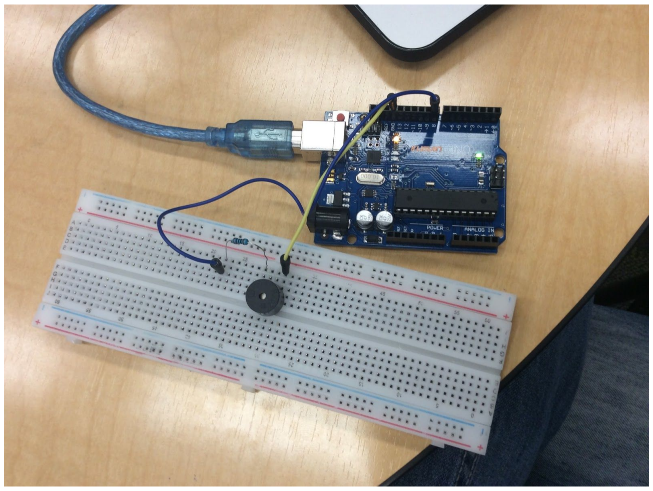




The burger cutout was the first part done to create our light up Burger, to do this we cut out a burger in a tea box, then we created layers from more cardboard to make sure that layers of light don't bleed through to each other.

The cardboard is used to hold the LEDs in place, using square cutouts instead of circles allowed us to "pop" the LEDs into the cardboard ensuring they do not move while the game is played, as you can see there are 7 LED Rows but 8 row cutouts, the reason behind this because a mistake was made while cutting out the squares - they were too far apart and the soldered LEDs could not reach the cutouts. Once all the LEDs were in position we just attached the two pieces together using tape. We also made a mistake here where we taped the LEDs the wrong order, thus having the burger be built upside-down, so we had to untape and flip the LEDs. we also had to trip the sides of the cardboard holding the LEDs since when the box was folded, the cardboard bulged and pulled the LEDs away from the separators for the layers. Once the LEDs were in place, then we can refold the tea box back into its box shape, with only the 7 pin wires and the ground wire coming out/into it.

For creating music we first thought that components with two circular tubes one it was a speaker that came with arduino kit. It was instead a ultrasonic sensor used to detect how far an object. We were able to find code that would take the tones that the inactive buzzer can take and make them into musical notes that Elizabeth can used to create a song for when the game is running. Adding a ~~library~~ header file called pitches.h will convert each bit value the buzzer can make into a musical note. This code can also set how long each note needs to be, whether it full, ¼, or ⅛ duration, and



https://www.arduino.cc/en/Tutorial/toneMelody

The main melody played while the game is active is based off of the main Tetris theme, but with different note durations and a few notes changed to different pitches. The melody played when the game is won is based off the Final Fantasy Victory Fanfare, but also with a few notes changed to different pitches. When the wrong button is pressed, 2 notes play a monotonous beeping noise to let the player know they selected incorrectly. When the game times out, 4 notes play a similar beeping noise, to let the player know it's game over. The buzzer is set as an output and has two wires connected to it. One wire is the ground and the other is connected to a pin on the arduino to receive the value of pitch that is to be made into sound by the buzzer.

Random Ref and 3 color LED:

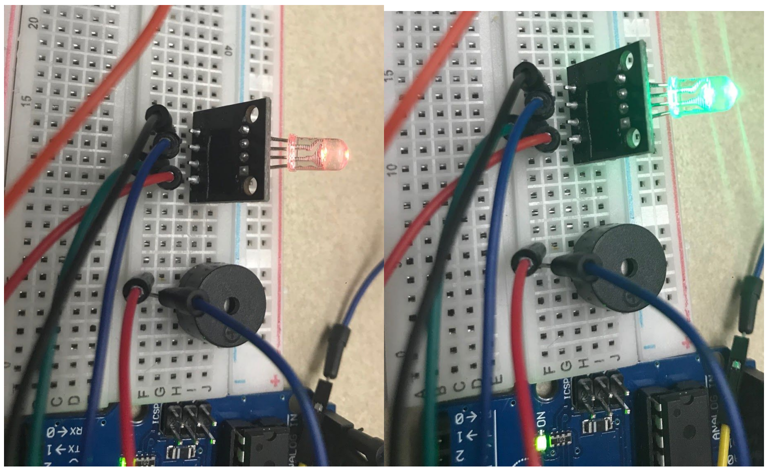
https://www.arduino.cc/reference/en/language/functions/random-numbers/random/

The idea is to have the code randomly assign ingredients to buttons, error checking to make sure that the same button isn't assigned twice. When the timer runs out the game stops and the buttons reshuffle when a person clicks the "start" button. Once the start button is clicked we will use the 3 Color LED Module

The plan was to use this code:

(http://www.learningaboutelectronics.com/Articles/RGB-full-color-LED-module-circuit.php)

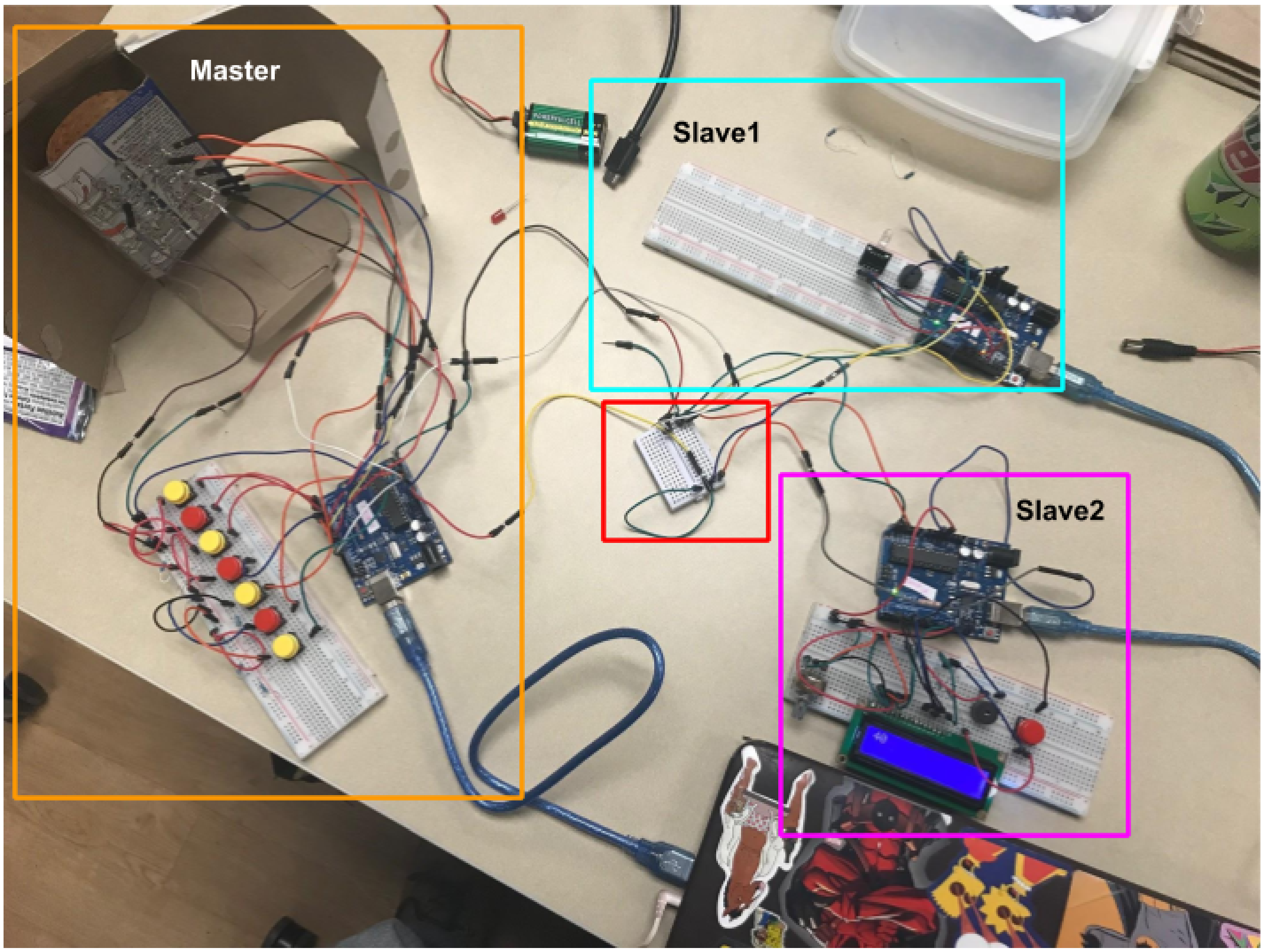
to do a "ready, set, go" style of "red yellow, green", giving the user some time to get ready before they play, but there were some technical difficulties (explanation on that later) that prevented the completion of this part. What we do have is the RGB LED displaying red when the



wrong button is pressed or the game times out. We also have the displaying random colors in a random order when the player wins the game. The RGB LED is set as an output and has four wires connected to it. There is one for each of the colors red, green, blue so that a value may be sent to it to by analogWrite() in the code uploaded and that will determine the amount of that color being outputted by the LED. There is also a ground connection. No resistors wereaddedso that the color could be strong.

We will be trying to get the 3 Arduinos to communicate using the tutorial from this link: https://www.youtube.com/watch?v=yQ15Hi1E7I4, should this way not work (or cause too many errors for our project) we will buy an additional Arduino Mega. After speaking to the TA Alex, an Arduino Mega would reduce a lot of the issues created when trying to connect multiple Arduinos, also with his recommendation we will refrain from using other connections such as bluetooth or wireless connections. - This idea was what we originally had, but we encountered a few issues along the way: first we attempted to connect too many parts together at once, therefore making it almost impossible to get the 3 arduinos to communicate. We also bought a Arduino mega that we ended up not using, the reason for that is simple… we didn't know exactly how to use it with the hardware serial communication.

So we had to "start" from scratch, as in literally a stripped empty skeleton of a code, from there we simply there we started off by having 2 arduinos communicate where if a button was pressed on the master Slave1 would turn on an LED, from there we exchanged the LED for a passive buzzer that would make a sound when the button was pressed. From there we kept adding parts of code to the skeleton until we had our game working.



The arduino in the ORANGE is the Master, it is connected to the LEDs, the buttons and it sends serial signals to the green and pink arduinos. One of the most difficult part of this project was randomizing the LEDs. The way that the randomization works with this project is that it takes in and stores what pins the LEDs are in and stores their pin number in an array. Since we can not switch wiring on the pins to the LEDs every time a new game begins, putting them into an array would be the most logical thing to do. From there, instead of the lighting up the pin manually, the button accesses the array and turns on the LED that is with that pin.

In order to randomize an array, it loops through the array, element by element, and swaps it with a random element within that array. That takes care of the randomization. In order to see if the LEDs were lit up in the right order, we had a second array that stores the correct order of the pins. When a button is pressed it accesses the array of the random array element based the number of the button pressed. It then compares the current sequence of buttons to the ordered array of the LEDs based on the Counter of the number of button presses. If they match, it keeps the LED on and increases the counter. If any part of the two do not match, the master resets the counter and turns off all the LEDs that are currently on the burger.

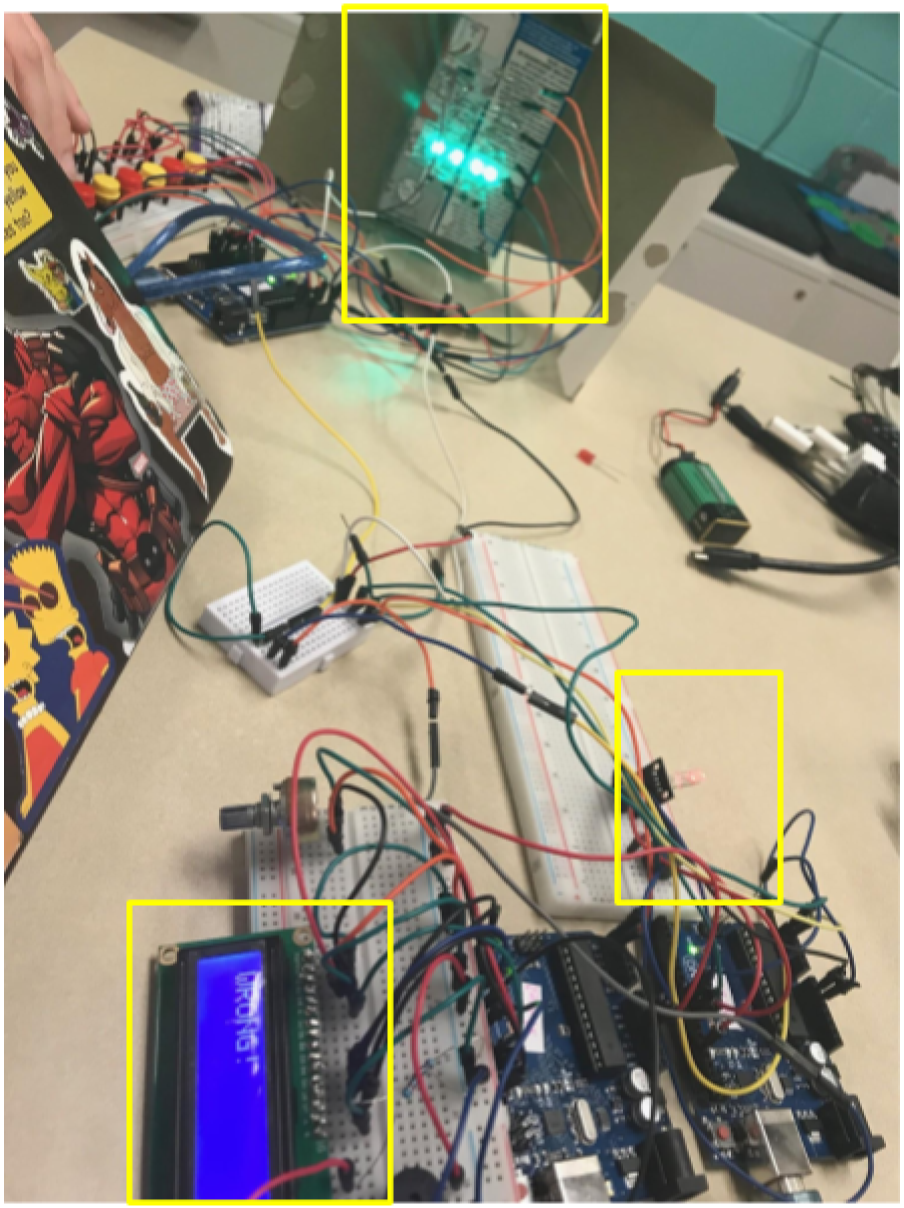
When the player hits the wrong button, the layer connected to that particular button lights up (so that the player can have a hit of what ingredient was assigned to that button), then the

master needs to show with the LEDs that it's in the wrong order by resetting the current lit LEDs. This will then send a signal to slave 1 from the master to change the color of the 3 color LED to red. If the player picks the right button, it will change the 3 color LED to green.

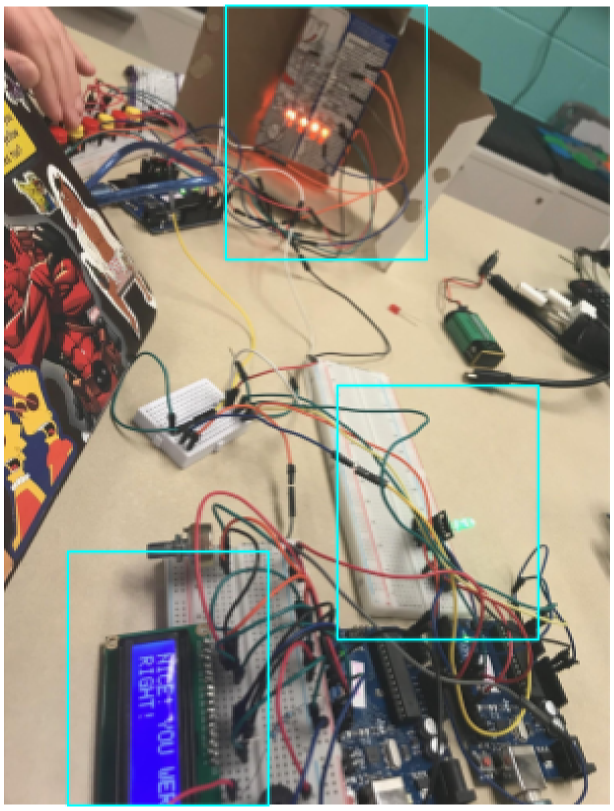
The way that sees that the player won, it just check it see if all the LEDs are turned on. When that happens they will causes the LEDs to blink on and off in a loop. The master then sends a signal to slave 1 to turn the 3 colored LED to randomly change color a few times. After that it makes a new game with a different random order then the last game.

The RED box is a mini breadboard that connects the 3 arduinos to each other, all three arduinos connect to the same ground, the same A4, and A5. this way in the code we have to specify which arduino is receiving the signal from the master.

The BLUE is referred to as Slave1, Slave1 is in charge of the game music and wrong button, time out, won sounds using the passive buzzer, and the RGB LED. When the start button is pressed(connected to Slave2) there will be a color count down to indicate that the game has begun. Every time a button is pressed, whether it is wrong or correct the main melody music will pause as the signal received from the master, all activity in Slave1 is paused in order to complete the action triggered, in this case the main melody music is stopped in order to light up the RGB LED based on whether the button was correct or not. The RGB LED will light up red if the wrong button is pressed. The RGB LED will light up green if the correct button was pressed.



The PINK box is Slave2, it is connected to the LCD, the active buzzer and the start button. The active buzzer makes a sound when the player presses the wrong button. The red button is also the start button for the next game, since we couldn't figure out how to make the master wait for it's slave input, so the first game is automatically started, with the timer also starting. The reason the timer starts is because it starts to run the moment Slave2 receives a signal from the master. Once the start button is pressed to start the second game the timer is reset, a signal is sent to Slave1 to start the RGB LED color countdown and the master arduino is able to start playing the game again.



Slave2's LCD display (while the game is being played) displays the playing time, and Slave2 displays messages when a button is pressed, "Nice, you were right" if the player presses the correct button order, and "WRONG!" if they got the wrong button. Then the LCD goes back to displaying the timer The top half of the LCD display will be used to display the quickest time spent solving the puzzle. This can be simply done by using a static global variable. A conditional statement will check to see if the most recent completed time is smaller than what that global variable is holding. If so, change it and display it. The bottom half will display messages to the player (this will be a very similar idea to Lab 3 where we had a scrolling text).

Problem we encountered.

1. We had Elizabeth’s laptop become unable to upload any sketches. Every time an upload was tried, we got this error message:

```
C:\Program Files (x86)\Arduino\hardware\tools\avr/bin/avrdude -CC:\Program Files
(x86)\Arduino\hardware\tools\avr/etc/avrdude.conf -v -patmega328p -cstk500v2 -Pusb
-Uflash:w:C:\Users\villa\AppData\Local\Temp\arduino_build_945740/FinalProj.ino.hex:i
        avrdude: Version 6.3-20171130
                Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
                Copyright (c) 2007-2014 Joerg Wunsch
                System wide configuration file is "C:\Program Files
(x86)\Arduino\hardware\tools\avr/etc/avrdude.conf"
                Using Port                    : usb
                Using Programmer              : stk500v2
        avrdude: usbdev_open(): did not find any USB device "usb" (0x03eb:0x2104)
        avrdude done.  Thank you.
        An error occurred while uploading the sketch
```

We figured out that the problem was on that laptop and not with the code, wiring, USB, or arduino by trying to upload from different laptops the exact same code on the arduino she was working with and used the same USB, IDE, and board/port/programmer settings. All uploads were successful, except on Elizabeth’s laptop. We tried all combinations of the following to find a solution so she could keep coding and testing: plugging in with different USB cables, different arduinos, uploading different kinds of sketches, changing the board/programmer settings, using different arduino applications (IDE, desktop app, online editor), shutting down the system, updating, uninstalling/reinstalling. Everything was unsuccessful. All those same combinations we tried to pinpoint the problem on the laptop worked on all other laptops. After trying for 24 hours, we gave up and moved on to focus on getting the arduinos to communicate, since most of the code for the buzzer and RGB LED was already written and the main part of this project is to get multiple arduinos to communicate.

2. We had a problem with soldering, at one point the LEDs stopped working. This was mostly due to the fact that our LEDs wouldn't be

3. One of the the biggest problems we ran into was connecting the three arduinos in order to make then all communicate. We first started by getting a arduino mega in order to connect the arduinos. However, because there was very little info online on how to connect and code for arduino mega. We were unable to use the MEGA. We indead found a youtube video on how to connect 3 arduinos using i2C to make them communicate with each other. Here we would have a master and 2 slaves. The master would be handling the buttons and LEDs while the slaves handle the rest of the external devices. The way this works is that the slaves will check to see if anything serial letter was sent through the wires. When it does it takes that letter to that slave and based on that letter it causes a trigger. Based on that trigger it will do the intended thing. However, it will stop whatever that arduino was currently doing, interrupting the current action. For example, if the trigger to turn on the 3 color LED comes, then that will stop (interrupt) the music in order to turn on the 3 color LED.

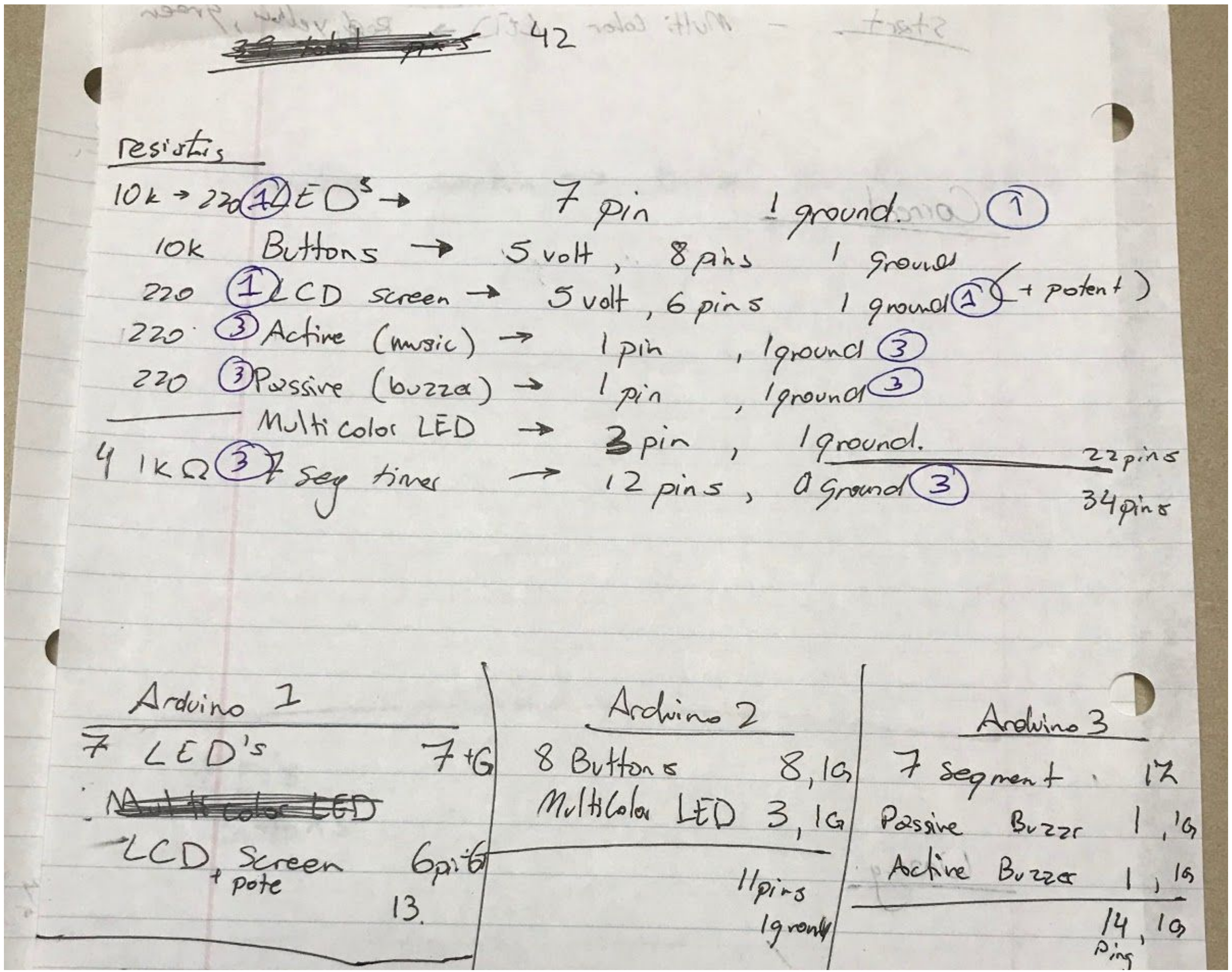
4. A major problem we had was that when we used the trigger for the communication, delays would not work. This was a problem because this is how our music worked, with delays.

We then found out that the loop still works when inputting that music function in there. This caused the music to work and in order to make the winning song work we added a flag that can we activated by the triggers. Additionally, we eventually we found out the "receiveEvent" function used to communicate between master and slave when it runs through its slave code, it tries to go through its code/functions as fast as it can, therefore delays are done as quickly as possible so we had exaggerate the delay x10,000 in order to make any sort of delay visible.

5. We had a problem with the timer. When we try to send a signal to reset the timer it just would not do it. Instead we reset it to zero when we press the start button.

6. We also tried to get help from a TA (they will remain nameless), and we were told that our project was too complicated and that they could not help us.

7. We also had issues with planning which arduino controls with external devices. For that we spend some time creating a layout of arduinos and breadboards:



~~39 total pins~~ 42

resistors

10k → 220 (1) LED's → 7 pin 1 ground. (1)
10k Buttons → 5 volt, 8 pins 1 ground
220 (1) LCD screen → 5 volt, 6 pins 1 ground (1) (+ potent)
220 (3) Active (music) → 1 pin, 1 ground (3)
220 (3) Passive (buzzer) → 1 pin, 1 ground (3)
Multi color LED → 3 pin, 1 ground.
4 1kΩ (3) 7 seg timer → 12 pins, 0 Ground (3)

22 pins
34 pins

Arduino 1		Arduino 2		Arduino 3	
7 LED's	7 +G	8 Buttons	8, 1G	7 segment	12
~~Multi color LED~~		MultiColor LED	3, 1G	Passive Buzzr	1, 1G
LCD Screen + pote	6pi+G			Active Buzzer	1, 1G
	13.		11 pins 1 ground		14, 1G Pins

timer thing ⟶ 00:00

7 flags for layers

- → user press start button
- → multi color LED countdown Red, yellow, green.
- → Active Buzzer sound "(3,2,1)"
- → timer starts
- → [illegible] Active buzzer starts play music.

Correct button:

- ⟶ layer lights up
- → multi goes green
- → [illegible] music contines
- → LCD "Correct!"

if all 7 layers on:

- → win song plays
- → layers flash on/off
- → LCD "you won!"
- → timer stops
- → multi: random colors

Wrong Button:

- → All layers turn off (either all at once or lay layer)
- → Passive Buzzer Buzzer "wrong" tone
- → Active Buzzer Music Stops
- → LCD = "Wrong ingredient"
- → music re starts.

Contine Sequence untill 2 min up.
↳ LCD → "Game Over"

Future plans:

- Implement the 7- segment, we unfortunately were not able to implement it due to the fact that we ran into many complications with the software serial communication, and then later the issue with not being able to "use" delays in the slave codes. We would have used the 7-segment to keep time rather than have the LCD screen display/ keep track of the time.
- Cleaned up the wiring, while we do have the main burger display in a box, the rest of the our project is a mess of wires and it looks messy and over complicated. To fix this we could have put the RGB LED in the burger case along with creating a case for the buttons and the LCD screen. This would have reduced the amount of wires visible.
- We also wanted to the music speed up depending on how much time was left before the game was over, this would help the player know if they are running out of time. We were not able to implement this because this requires one slave -slave2 (the one that would have the 7-segment) to constantly send the time to the other slave- slave1 (the one with the passive buzzer) to notify it when to speed up the song. When sending instances/triggers to slaves, all activity is stopped of that arduino, thus making the LCD and active buzzer user useless since they will always be sending data to slave1.