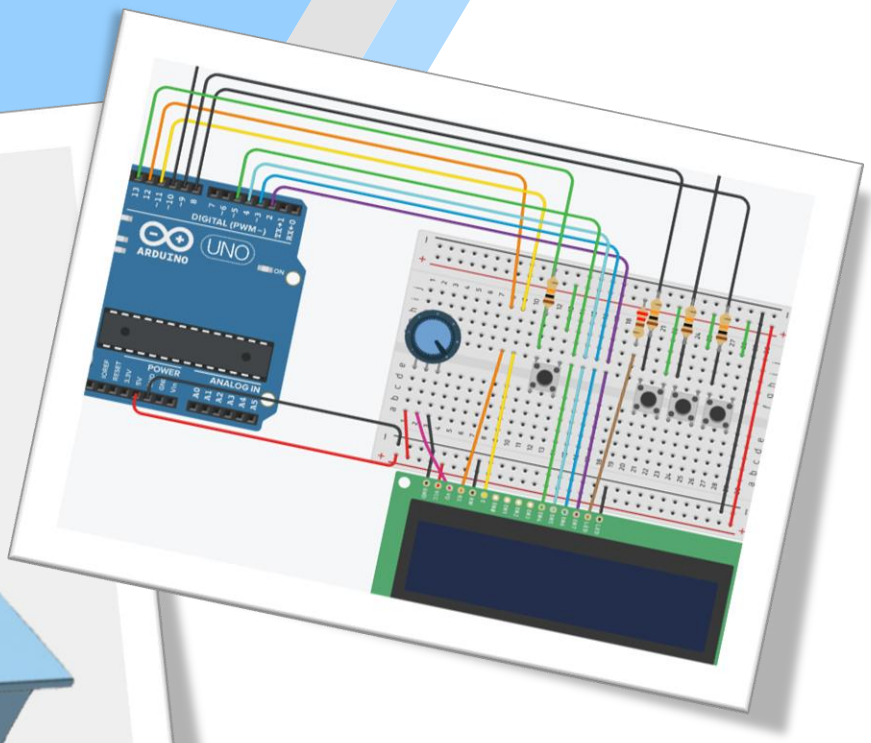
EGN2020C-SF02

Andrea Goncher

Spring 2022

What's For Dinner?



Sean Lewertow
Michael Garcia
Ethan Laws
Chase Watson

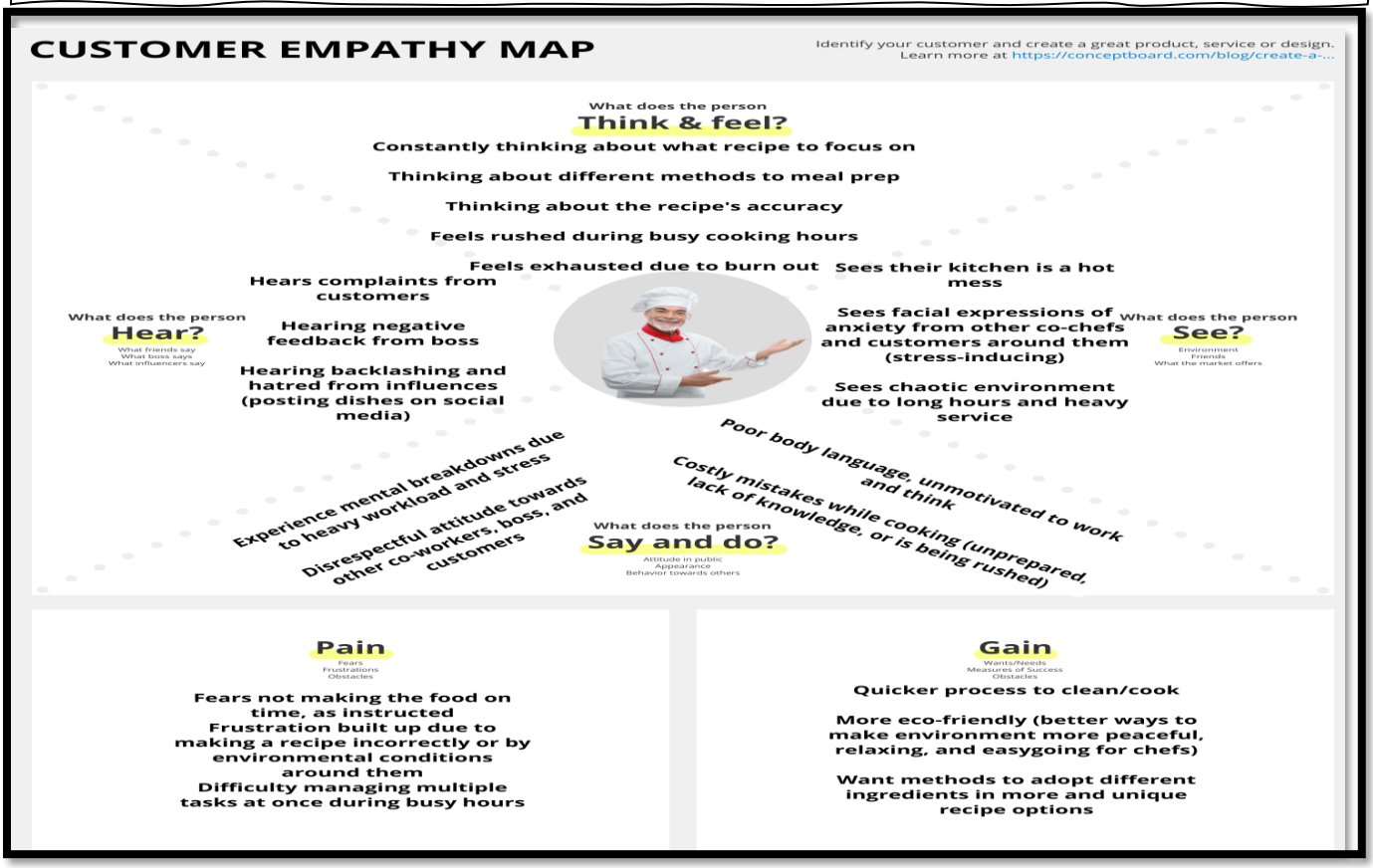
GATOR Engineering
UNIVERSITY OF FLORIDA

TABLE OF CONTENTS

User Group and Needs Statement

Human-Centered Design

Chefs were picked as our targeted user group. Being a chef is physically and psychologically straining since you must stand in a hot kitchen for hours on end while also putting in the long hours and attention required to get the food out on time. They have a lot of tasks to fulfill as well as a lot of hurdles to overcome. Chefs must make specific types of meals in a short period of time. They must also accomplish several difficult tasks; as a result, we are designing new and more efficient technologies to aid them. To satisfy our users, we aim to produce a cost-effective, human-centered design project for our user group that is successful enough to increase quality of life while also being environmentally friendly. To do this, we electronically revamped the cookbook to reduce the burden involved with deciding what to prepare. Flipping through a cookbook takes time and prohibits a chef from serving diners at an astonishing rate, thus, to save time before preparing dishes, the chef may swiftly flick between the display on the What's For Dinner gadget. Based on the components, the chef will be able to view the best potential recipe to cook right now. Because a chef's goal is to prepare food that looks great, the device will allow them to spend more time in the kitchen preparing meals. It can be difficult to stop and think in the kitchen, so the device eliminates the need to think about recipes, which contain a lot of information on paper. Chefs would rather have a mental image of how to prepare their meals than try to read large blocks of text. When time is of the essence, many people today prefer not to read. We believe utilizing this technological design is superior to using any other guidebook or way to obtain recipes that a chef may have difficulty finding. This is What's For Dinner.



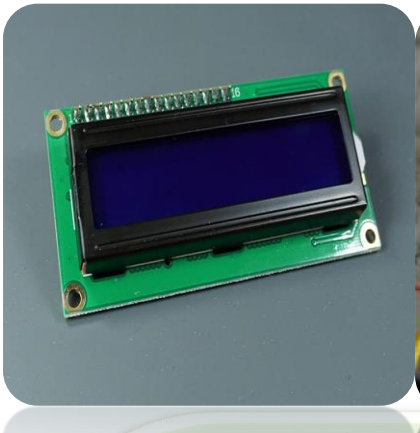
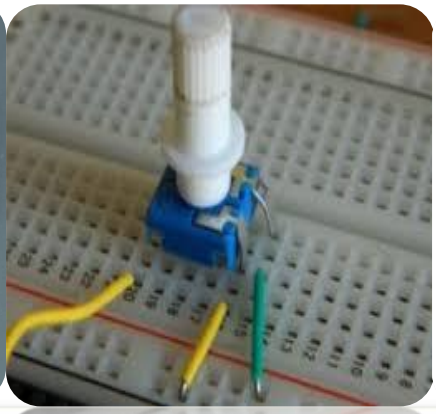
Design Justifications

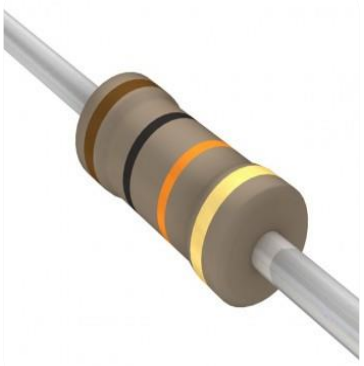
Requirements	weight(%)	Normalized score (Stovetop)	Weighted Score (Stovetop)	Normalized Score (What's for Dinner)	Weighted Score (What's for Dinner)
cost	**15%**	**0.800**	**0.12000**	**0.6250**	**0.09375**
volume	**5%**	**0.525**	**0.02625**	**0.7140**	**0.0357**
Ease of use	**20%**	**0.666**	**0.1332**	**0.3330**	**0.0666**
Practicality	**60%**	**0.444**	**0.2667**	**0.75**	**0.45**
	=		**=**		**=**
total	**100%**		**0.5462**		**0.6461**

We ultimately chose "What's for Dinner" (WFD) over "Stovetop" because we felt it better addressed the needs of our target user demographic. Chefs, or those who cook food, were our target audience. The ultimate design was determined by four criteria: cost, volume, simplicity of use, and practicality. We discovered that WFD was more feasible for cooks to utilize. The WFD's final design was created to sit on a kitchen counter. It doesn't take up much room and is simple to use for any cook. "Stovetop" detects if something is dangling from a kitchen stove, which not everyone will need. The WFD can assist anybody locate culinary recipes, but "Stovetop" is only beneficial in certain situations. WFD is more beneficial overall than the Stovetop design. I now disagree with our previous "Ease of Use" ratings for both WFD and Stovetop; the stovetop design must be set up by connecting it to something and then accurately directing it towards a burner. The sensor on the stovetop cannot be concealed during cooking, which is something a chef is likely to do. The WFD is simple to use and does not require any setup. Stovetop is most likely less expensive to manufacture than WFD. The stovetop just requires a sensor, but the WFD requires four buttons wired to a breadboard as well as an LCD screen. The Stovetop would most likely take up less space than the WFD since the WFD requires a different layout of an LCD screen, a breadboard, and an Arduino circuit board than the Stovetop. The WFD has the appearance of a book, which I think is an excellent design decision because it serves as a cookbook. Chefs or folks who enjoy cooking would probably prefer "What's for Dinner" to "Stovetop."

Ethical & Environmental Considerations

The non-circuitry parts “What’s for Dinner” are made from 3d printing filament, its’ 3d printed parts can be printed again and replaced. The 3d printed filament is relatively durable, it’s hard to break. The screen was placed to be easily readable; the LCD screen is connected to the breadboard using male to female cables. The WFD was made to take up little space; there isn’t much extra space between the circuitry and the 3d printed parts. The WFD is slightly water resistant in case of spills during cooking. The WFD was made for people you cook food; it is not hard for the average chef to use. It can give you lists of recipes for you to use. The WFD is easily usable for the average person; it only has four buttons to navigate the menu and recipes. You might be wondering if parts of the WFD can be recycled after it breaks down? When the WFD breaks down its’ parts can’t be recycled. You could theoretically use the 3d printed parts for other purposes after it breaks down, but the printed parts will likely have to be taken to a landfill. If you decide to recycle the 3d printed parts you could use them as a: cool looking box to hold various things, a miniature doll house, weird looking small picture frame, or even a box cats can play with. The 3d printed parts, buttons, and LCD screen can be broken and the WFD could still be repaired, however if the circuitry is damaged it may be significantly harder to repair. The manufacturing of the 3d printed parts can be done mostly automatically considering they’re 3d printed. Parts that are 3d printed are relatively easy to produce considering you only need a 3d printer and some filament to make them.



LIQUID CRYSTAL DISPLAY (LCD)

To offer a user interface, you may easily connect a liquid crystal display (LCD) to an Arduino. LCDs are utilized in embedded system applications to show different system data and status. Soldering a pin header strip to different pin count connections of the LCD screen would be the best way to verify that the lcd works properly. Recipes are presented on the LCD based on the digital pin arrangement in the Arduino, which is then coded in code to display a user interface via recipe options.

POTENTIOMETER

A potentiometer is a basic mechanical device that generates variable resistance when its shaft is rotated. It is feasible to measure the amount of resistance generated by a potentiometer (or pot for short) as an analog value by feeding voltage through a potentiometer and onto an analog input on your board. A potentiometer was used on an LCD to control the bias level of the LCD, which is the contrast. We needed to set a voltage between Vcc and Vee, which we sent into Vo. That is, a voltage ranging from +5V to -5V.

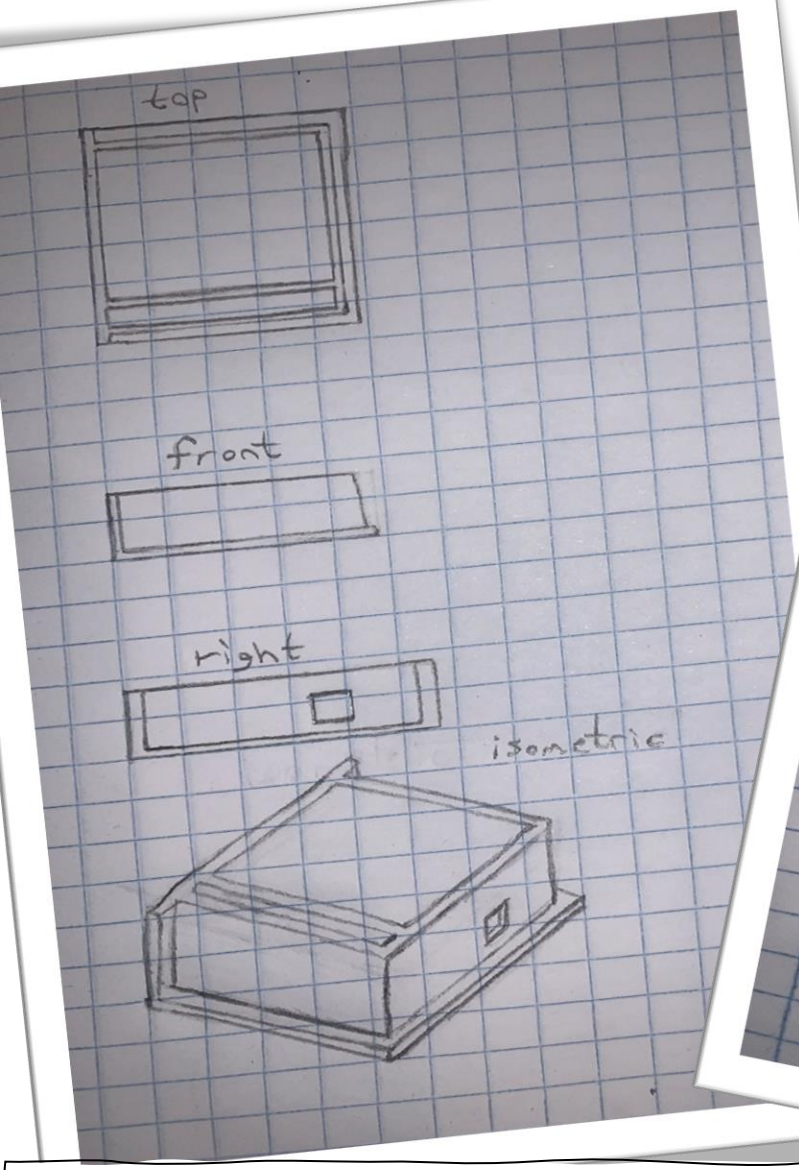
PUSH BUTTONS

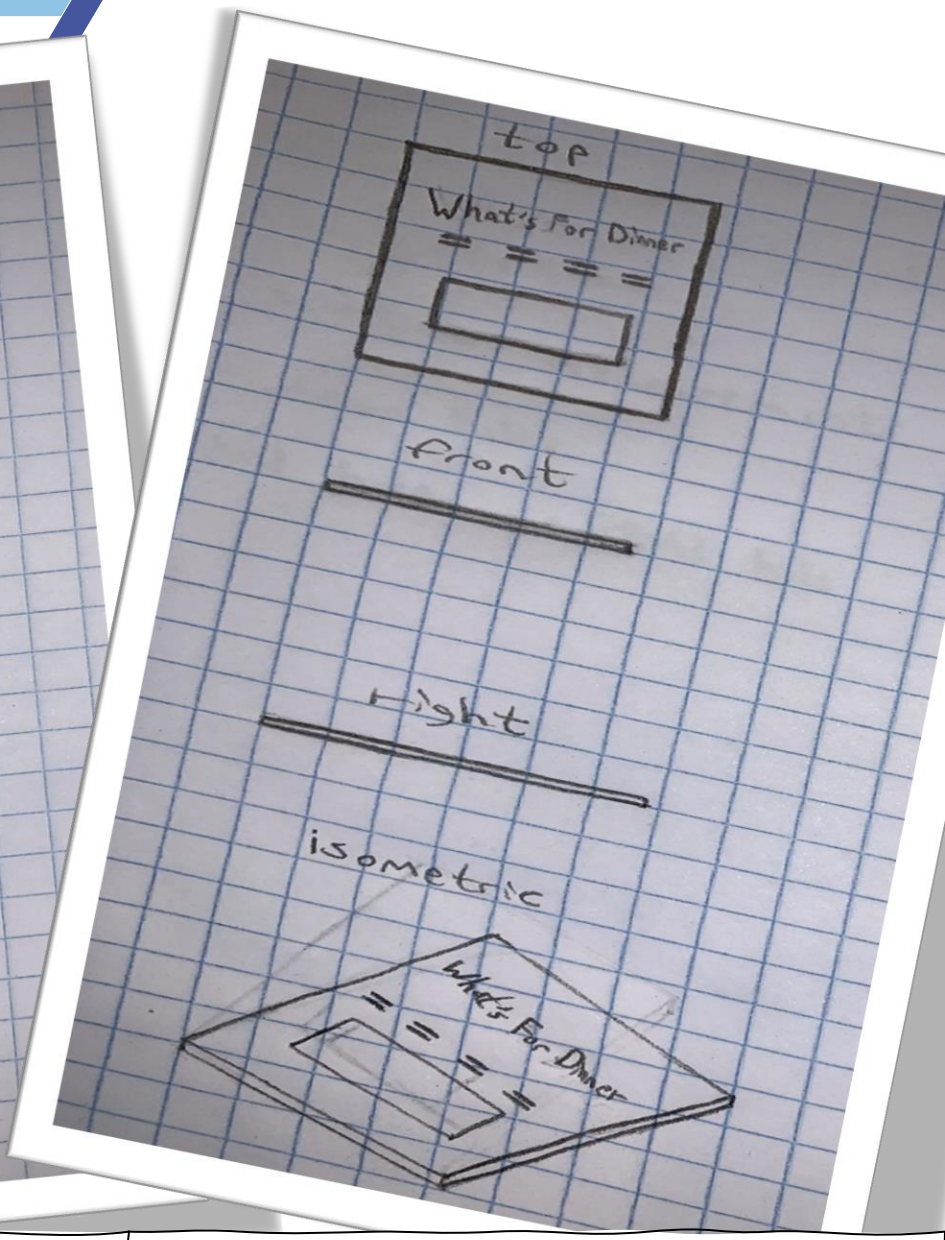
When an user presses a pushbutton or switch, it joins two points in a circuit that the tinkercad analyzes. A resistor is required when a push button is used as an input by a microprocessor or microcontroller. Following this, ensure that the micrcontroller's input has a stated state (0 or 1); otherwise, the input may be left floating (which is unwanted). The function is digitally read through the digital read of the function, in this case digital 8.

RESISTORS

The resistor's purpose is to limit the amount of current that travels through it into the LED to a level that the Arduino can provide. When the pushbutton is open (no connection), the input has an uncertain value, therefore some resistance is required. We had to determine that the pullup causes the pin to be pulled to a high voltage level (5 volts). The voltage on the pin lowers to a low level when the user presses the button (0 volts).

Engineering Drawings

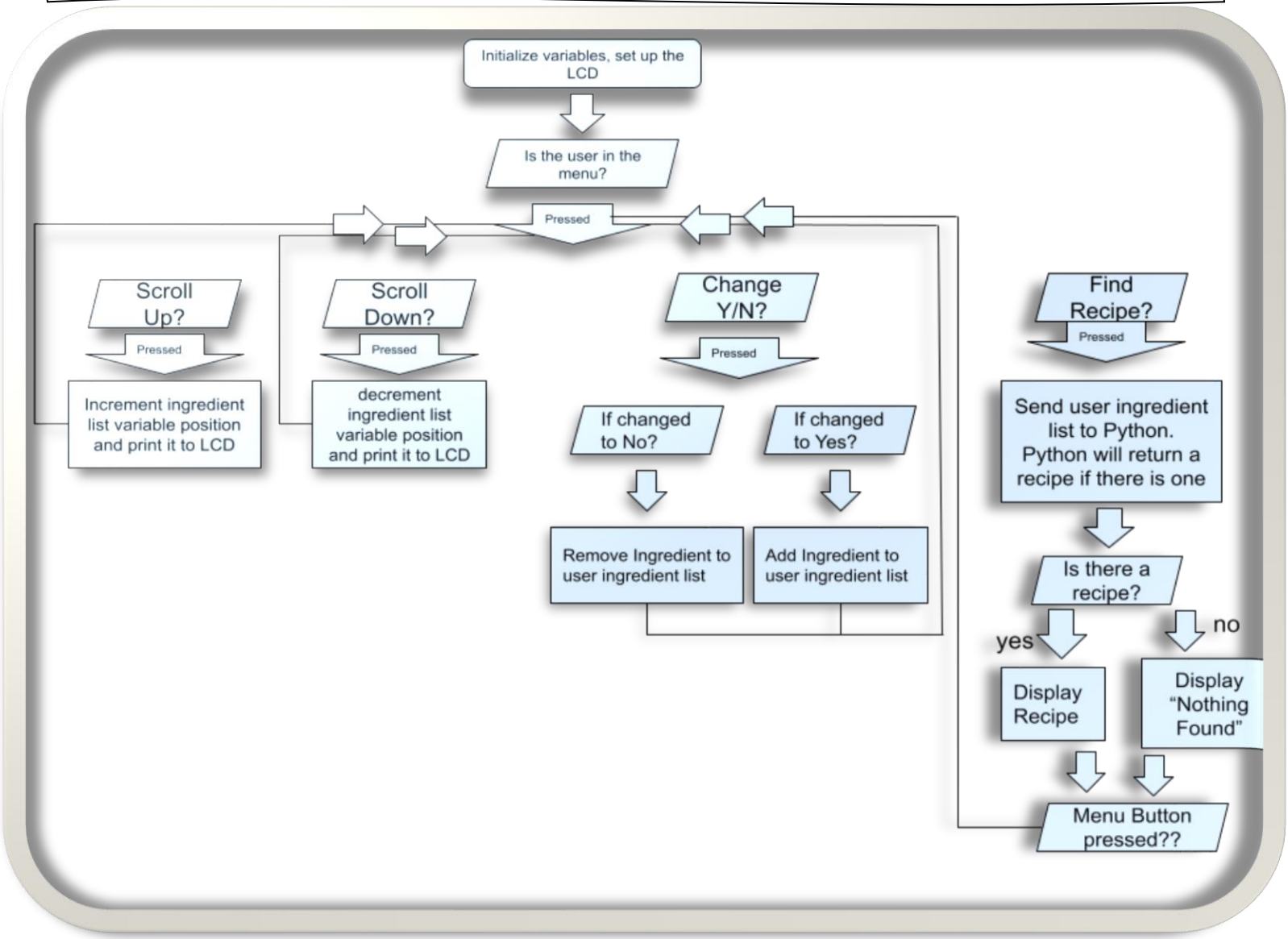




This portrayal is the base of the WFD, it holds the breadboard, circuit board, and has a HDMI port.	**This diagram shows the cover of the WFD, it has the logo, holes for buttons, and an LCD screen.**

Flowchart

The code begins by displaying a start screen to the LCD. The user presses the menu button, it brings the user to the ingredient menu. The user can then decide to press one of four buttons. If the user scrolls down, then decrement the position of the ingredients list and display the new ingredients. if the user scrolls up, increment the ingredients list and then display the new ingredients. If the user selects the y/n button, and if the user's currently selected ingredient is N, then change it to Y, and if it is Y then change it to N. After any of these three options are selected, return to the original menu state, and wait for the next user input. The final option is if the user presses the change menu button again. This would send the lcd into the recipe loading screen which displays a “Please Wait…” text as a python script processes the users selected ingredients list and tries to search for if the users’ available ingredients is enough to make any of the recipes in the recipe database. If nothing is found, print out that nothing is found. If something is found, print out the recipe that is found. After this, If the user pressed the menu button again, bring them to the ingredient menu screen.



Commented Code

Arduino Code

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
const unsigned int MaxMessageLength = 20;

//if rue, the ingrediant menue will be displayed if not the random food recipie will be displayed
bool inMenue = false;
bool backToIngredients = false;

//if the button is not being pressed set to true. if not it will be false
bool buttonPressed;

//we use 'n' here to determine how far down the list of ingrediants we are. this number can be used for a few things
int n = 0;

//UI variables
String up = " ";
String down = "v";

//all the ingrediants used in the recipies(In final product, this would be obtained by sorting through recipes list)
String ingredients[] = {"Bread","Jelly","Ham","Cheese","Mayonnaise",
                        "Meatball","PB","Tomato Sus","Tomatoe","Lettus",
                        "Croutons","Rice","Broth","Muchrooms","Garlic","Parcly",
                        "Mchd ptatos","Gravy","Corn","Ground Beef"};

//Y / N UI element for each potential ingrediant.
String hasIngredients[] = {"N","N","N","N","N","N","N","N","N","N","N","N","N","N","N","N","N","N","N","N","N"};

//what ingrediants does the user have
String userIngredientsList[] = {"End"};
String userIngredients[] = {};

//counter value for the position of the userIngredientsList
int counter;


//digital pins
const int Dig8 = 8;
const int Dig9 = 9;
const int Dig7 = 7;
const int Dig13 = 13;
```

```
void setup() {
  //set baudrate to 9600
  Serial.begin(9600);

  //set up lcd
  lcd.begin(16, 2);
  lcd.print("   What's For");
  lcd.setCursor(0,1);
  lcd.print("    Dinner?");

  //set up the digital inputs
  pinMode(Dig8, INPUT);
  pinMode(Dig9, INPUT);
  pinMode(Dig7, INPUT);
  pinMode(Dig13, INPUT);
  counter = 0;
}
```

```
//main loop where everything happens
void loop() {
  //if the last position of UserIngredientsList is not "END" then call the recipe fuction at position of counter
  if(userIngredientsList[counter] != userIngredientsList[-1]){
    recipe(userIngredientsList[counter]);
    counter++;
  }
  //determins if no button is being pressed
  if(buttonPressed == false && digitalRead(Dig13) == LOW && digitalRead(Dig9) == LOW && digitalRead(Dig8) == LOW && digitalRead(Dig7) == LOW && inMenue == tr
    buttonPressed = true;
    backToIngredients = false;
    lcd.clear();
  }
  //displays dinner idea
  if(digitalRead(Dig7) == HIGH && inMenue == false && buttonPressed == false){
    if(backToIngredients == true){
      lcd.print(" Random Dinner");
      String dinnerIdea;
      int pos = 0;
      //get the recipe from the serial buffer from python
      while(Serial.available() > 0){
        String[pos] = Serial.read();
        pos++;
      }
      lcd.setCursor(0,1);
      //print recipe
      lcd.print(dinnerIdea);
      buttonPressed = false;
    }
    //else go back to ingredient menue
    else{
      inMenue = true;
      buttonPressed = false;
      lcd.clear();
    }
  }
```

```
if(inMenue == true){
  lcd.setCursor(0,0);
  //an ingredient name can nor be too long or it will mess with LCD UI Layout
  //set length to 10 to solve that prbluem for now
  while(ingredients[n].length() > 10){
    int shift = 10 - ingredients[n].length();
    ingredients[n].remove(10,shift);
  }
  while(ingredients[n + 1].length() > 10){
    int shift = 10 - ingredients[n + 1].length();
    ingredients[n + 1].remove(10,shift);
  }
  //display text(always make sure text is updated in the menue
  lcd.print("> " + ingredients[n]);
  lcd.setCursor(12,0);
  lcd.print(" " + hasIngredients[n] + " " +  up);
  lcd.setCursor(0,1);
  lcd.print("  " + ingredients[n + 1]);
  lcd.setCursor(12,1);
  lcd.print(" " + hasIngredients[n + 1] + " " + down);
  if(digitalRead(Dig13) == HIGH){
    if(hasIngredients[n] == "Y" && buttonPressed == true){
      hasIngredients[n] = "N";
      for(int i=0;i <= sizeof(userIngredientsList)/6; i++){
        if(userIngredientsList[i] == ingredients[n]){
          userIngredientsList[i].remove(0, -1);
        }
      }
      buttonPressed = false;
    }
    else if(hasIngredients[n] == "N" && buttonPressed == true){
      hasIngredients[n] = "Y";
      userIngredientsList[-1] = ingredients[n];
      Serial.print(userIngredientsList[-1]);
      buttonPressed = false;
    }
```

```
    //if the user goes up,
    if(digitalRead(Dig8) == HIGH && buttonPressed == true){
      n++;
      //if the user is at the top of the list
      if(n == sizeof(ingredients)/6){
        n--;
      }
      //fix UI accordingly
      if(n == 0){
        up = " ";
      }
      else{
        up = "^";
      }
      down = "v";
      buttonPressed = false;
    }
    //If the user goes down,
    if(digitalRead(Dig9) == HIGH && buttonPressed == true){
      n--;
      //if the user is at the bottom of the list
      if(n < 0){
        n++;
      }
      //fix UI accordingly
      if(n == ((sizeof(ingredients)/6) - 1)){
        down = " ";
      }
      else{
        down = "v";
      }
      up = "^";
      buttonPressed = false;
    }
    //if the user presses the menue button again send the user ingredients list to python
    if(digitalRead(7) == HIGH && buttonPressed == true && backToIngredients == false){
      buttonPressed = false;
      inMenue = false;
      for(int i = 0; i < sizeof(ingredients)/6;i++){
        Serial.println(ingredients[i]);
      }
    }
  }
}
```

```
//recipe finder function!!!
String recipe(String ingredient){
  //send user ingredient at the counter's position to python
  Serial.println(ingredient);
  char message[MaxMessageLength];
  unsigned int message_pos = 0;
  while (Serial.available() > 0){
    //while serial buffer has bytes in it,
    //take out a char from the buffer
    char recipeInByte = Serial.read();
    //if its the last character or that max character length is met,
    if(recipeInByte != '\n' && message_pos < MaxMessageLength - 1){
      //set it in the message char array
      message[message_pos] = recipeInByte;
      //go to next char array pos
      message_pos++;
    }
    else{
      //make the last null character nothing
      message[message_pos-1] = ' ';
      //char string over so reset message pos
      message_pos = 0;
      lcd.print(" !Dinner Found!");
      lcd.setCursor(0,1);
      lcd.print(message);
      //userIngredientsList counter reset
      counter = 0;
    }
  }
}
```

Commented Code (cont.)

Python Code

```
import time

#list initializations
user_ingredients = []
user_ingredients_bytes = []

#Serial port initialization for python
ser = serial.Serial(port='COM3', baudrate = 9600, bytesize = 8, timeout = 5, stopbits = serial.STOPBITS_ONE)  # open serial port
time.sleep(2)



#Loop through thre recipes and return a recipe if the user has the ingredients for a recipe
def Loop1(userIngredients):
    recipeList = [
                ["PBJ","Bread","PB","Jelly"],
                ["Fluff N` Nutter","PB","Bread","Fluff"],
                ["Ham and Cheese","Ham","Cheese","Bread","Mayonnaise"],
                ["MeatBall Sub", "Meatball","Bread","Tomato Sauce","Cheese"],
                ["Hamburger","Bread","Ground Beef","Cheese","Lettus","Tomatoe"],
                ["Caeser Salad","Cheese","Lettus","Croutons"],
                ["Risotto","Rice","Broth","Muchrooms","Garlic","Parcly"],
                ["Famous Bowl","Mached Potatoes","Gravy","Corn"]
                ]
    #list containing all available recipes.(a final product would obviously contain more recipes)
    #recipes are indexed accordingly: [recipe name, ingredient1, ingredient2,...ingredient*)

    for n in range(0, len(recipeList)):
        recipeName = ""
        recipeName = Loop2(recipeList[n], userIngredients)
        if(n == len(recipeList)-1 and recipeName == "NO"):
            return "None Found"
        elif(recipeName != "NO" and recipeName != ""):
            return recipeName
        elif(recipeName == "NO"):
            continue

#Loop through the recipe's ingredients and check if it is the user has all the ingredients.
def Loop2(ingredientList, userIngredients):
    for n in range(1, len(ingredientList)):
        hasIngredient = Loop3(ingredientList[n], userIngredients)
        if(hasIngredient == False):
            return "NO"
        elif(n == len(ingredientList)-1 and hasIngredient == True):
            return ingredientList[0]
        elif(hasIngredient == True):
            continue
```

```
#loop through the user ingredients and check if the selected recipe ingredient was a user ingredient
def Loop3(ingredient, userIngredients):
    for n in range(0, len(userIngredients)):
        if(userIngredients[n] == ingredient):
            return True
        elif(n == len(userIngredients)-1 and userIngredients[n] != ingredient):
            return False
        elif(userIngredients[n] != ingredient):
            continue



#This function will automatically run the instance the user decides to check for a recipe.
#This funtion will add whatever ingredients the user has to a python list which will then be used to find a recipe

while(ser.inWaiting() > 0):
    recieve = ser.readline()
    #find the first byte in the serial buffer
    recieved = recieve.decode("ISO-8859-1").strip('\n').strip('\r')
    #"ISO-8859-1" is a form of bytes-text conversion between arduino and python
    if(str(recieved) == "End"):
        ser.flush()
        #remove everthing else in the serial buffer
        recipe = Loop1(user_ingredients)
        #Call for loop 1 function to start up
        ser.write(recipe.encode('utf'))
        #send the recipe to the arduino code
        break
        #get out of the while loop
    user_ingredients.append(recieved)
    user_ingredients_bytes.append(recieve)

ser.close()
#close the serial port
```

Design Limitations

The appearance of our design makes it look like a cookbook, so it can easily be stored away and can just as easily be identified among smaller kitchen objects. The cubic shape gives it rigidity so that it can stay in one piece in the kitchen. The design will also be waterproof, so any type of spill in the kitchen will not affect the operation of the electronics. The major limitation of our design was the LCD display because it is only able to show two lines of text at a time. With a limited number of characters, chefs will have to toggle through more information. This is an aspect of the design that could be improved because it would both save chefs time and make the experience more enjoyable with a larger and more interactive display. Users will have to give the screen time to display all of the data based on the recipes that they are planning to put together. Scrolling through the data on the LCD display will be simple because the design incorporates four buttons. Two of the buttons will be for scrolling up and down and the other two buttons will be for making selections. The buttons are not very large, which suggests a minor limitation, but the compactness of the Arduino kit did not provide us with great options. If we could increase the size of our design, we would be more concerned about ergonomics and maximizing efficiency. Since we were limited to what the 3D printers could produce, we stuck with a design that could integrate Arduino parts.

References

- ***Chef download transparent png***. PNGHunter. (n.d.). Retrieved April 26, 2022, from https://pnghunter.com/png/chef-18/
- Programming Electronics Academy. "Using Serial.read() with Arduino | Part 1." ***YouTube*** video, 10:29. May 1, 2021. https://www.youtube.com/watch?v=qCjCRBLv_VM
- Experimentalist. "How to program Arduino using python | run Arduino through python | Experimentalist." ***YouTube*** video, 11:55. December 26, 2020. https://www.youtube.com/watch?v=sHzVZ19h_0I
- ***Tinkercad curcuit***. Tinkercad. (n.d.). Retrieved April 26, 2022, from https://www.tinkercad.com/things/8c8PmJeSWDx-start-simulating/editel?lessonid=EHD2303J3YPUS5Z&projectid=OIYJ88OJ3OPN3EA&collectionid=OIYJ88OJ3OPN3EA&sharecode=yQOH4qTu5f_qOBzijcpiIDRZ3lAtWbMR-deGY_A4UYY
- ***For***. for - Arduino Reference. (n.d.). Retrieved April 26, 2022, from https://www.arduino.cc/reference/en/language/structure/control-structure/for/
- Team, T. A. (n.d.). ***Digital Pins: Arduino documentation***. Arduino Documentation | Arduino Documentation. Retrieved April 26, 2022, from https://docs.arduino.cc/learn/microcontrollers/digital-pins#pullup-resistors-with-pins-configured-as-input

Appendix

Team Charter:
https://docs.google.com/document/d/1KvLwAF26kUU29vgcnHpY0nt47usK3EaLrsNr_GOaizU/edit

Tinkercad Link:
https://www.tinkercad.com/things/8c8PmJeSWDx-start-simulating/editel?lessonid=EHD2303J3YPUS5Z&projectid=OIYJ88OJ3OPN3EA&collectionid=OIYJ88OJ3OPN3EA&sharecode=yQOH4qTu5f_qOBzijcpiIDRZ3lAtWbMR-deGY_A4UYY