Push & Go
Cereal Dispenser

Developed by: Semih Akyuz, Raymond Pagan Jr

Problem Statement

Long lines form around the most common breakfast items in dining halls causing a potential increase in wasted time and accidents.

Mission Statement

We provide an efficient and effective experience to customers and employees in cafeterias through automation of the repetitive tasks.

Hotel Services

Corporate Breakrooms

Dining Halls

Food Industry Automation has a market size of 10.3 Billion dollars as of 2022 and is projected to reach 15.3 billion by 2030

User Persona’s

Jake

A college student who lives on campus and relies on dining services. Jake typically has a quick bowl of cereal for breakfast before rushing to classes. He likes things that are efficient and effective due to his busy schedule.

Ralph

A hardworking engineer who relies on a good snack to give him a boost of energy in his day. Ralph typically wants something sweet and quick without the need to leave the breakroom due to his busy schedule.

Paul

A hotel manager at Hilton Suites. He wants to ensure that all his guests get a quick and satisfying breakfast before they depart on there journey.

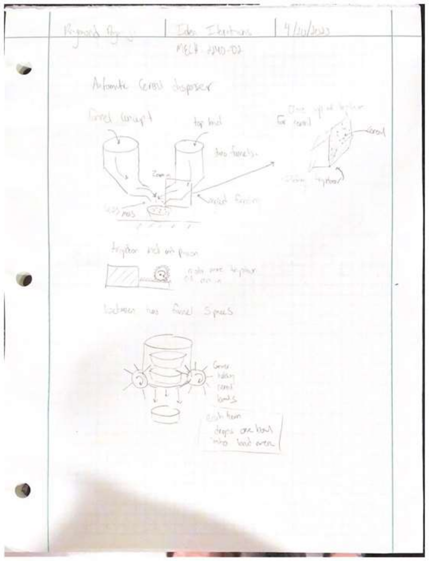
What do all three have in common?

Value Proposition

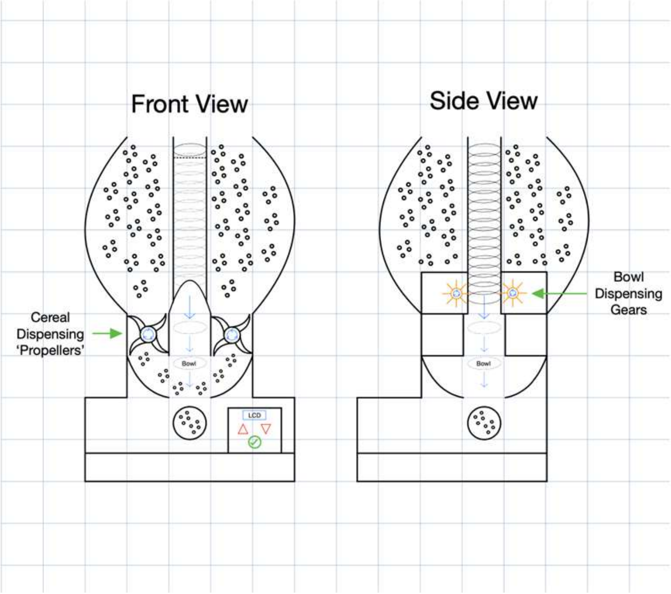
Our product will provide a better experience for customers looking to have an efficient breakfast while keeping employees happier due to reduced housekeeping tasks by automating a common process.

Concept Generation and Refinements

Sketched designs prioritized ease of use and efficiency, featuring an automated system for both bowl and cereal dispensing.


Idea Iterations
4/11/2023
MECH 2240-02
Automatic Cereal dispenser
top bowl
two funnels
cereal
sliding trapdoor
trapdoor rack and pinion
between two funnel spaces
Gears holding cereal bowls
each turn drops one bowl into bowl area

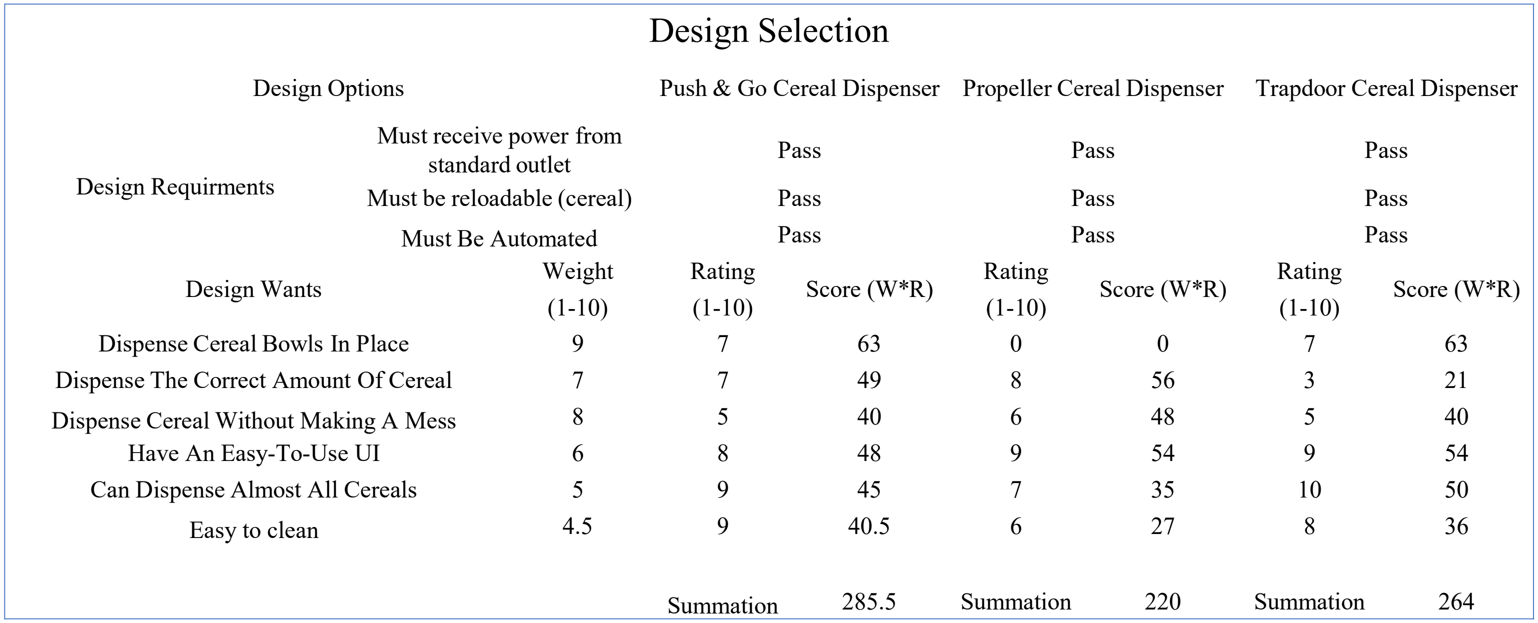
This design features two separate bodies that house the cereal and bowls. It uses a rack and pinion design to open a trapdoor that allows cereal to flow through.



This design features a sleek look by confining the bowl compartment within the cereal compartment. It makes use of gears and propellers to dispense the bowl and cereal.

Design Selection and Grading

Design considerations were determined based on user needs and wants.



Design Selection

Design Options			Push & Go Cereal Dispenser		Propeller Cereal Dispenser		Trapdoor Cereal Dispenser	
Design Requirments	Must receive power from standard outlet		Pass		Pass		Pass	
	Must be reloadable (cereal)		Pass		Pass		Pass	
	Must Be Automated		Pass		Pass		Pass	
Design Wants		Weight (1-10)	Rating (1-10)	Score (W*R)	Rating (1-10)	Score (W*R)	Rating (1-10)	Score (W*R)
Dispense Cereal Bowls In Place		9	7	63	0	0	7	63
Dispense The Correct Amount Of Cereal		7	7	49	8	56	3	21
Dispense Cereal Without Making A Mess		8	5	40	6	48	5	40
Have An Easy-To-Use UI		6	8	48	9	54	9	54
Can Dispense Almost All Cereals		5	9	45	7	35	10	50
Easy to clean		4.5	9	40.5	6	27	8	36
			Summation	285.5	Summation	220	Summation	264

Winner

User Requirements

End User

User shall be able to…

- Understand the purpose of the machine.
- Understand how to use the machine
- Receive a bowl
- Receive cereal in the bowl
- Select the amount of cereal they receive
- Tell when their serving is ready

Maintainer & Purchaser

User shall be able to…

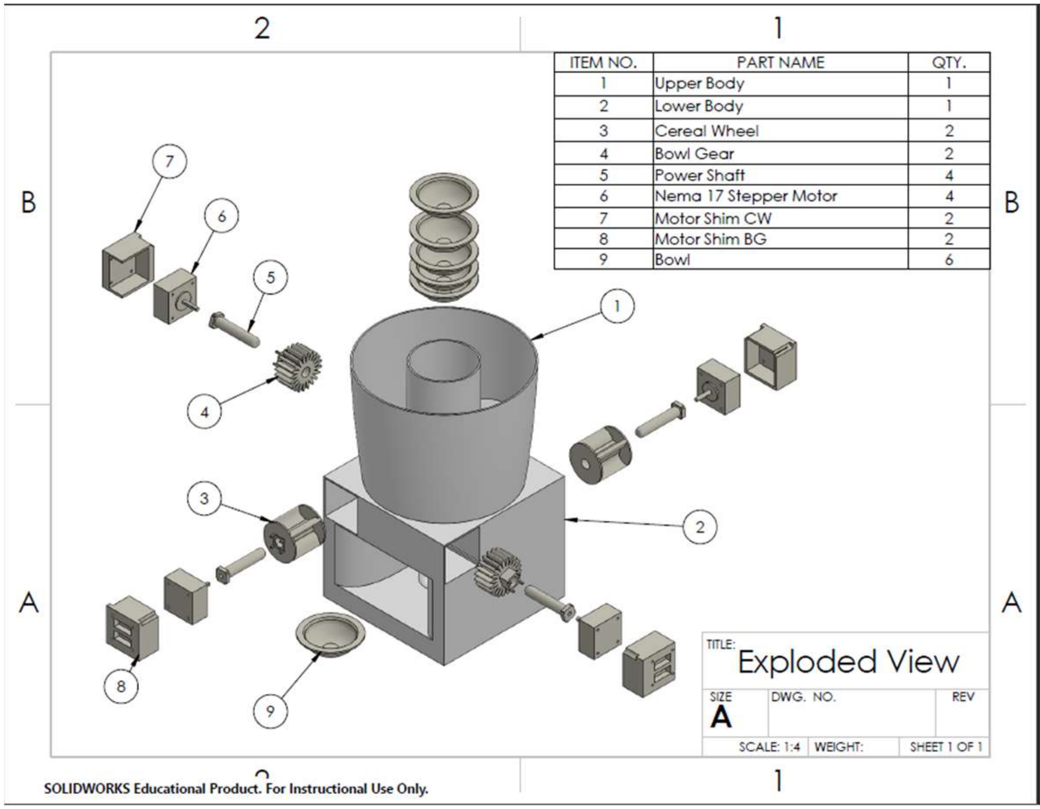
- Clean the machine
- Reload the machine (Bowls & Cereal)
- Power the machine

Functional Requirements

The device must be able to…

- Receive power from standard US outlet
- Convey how to use the machine
- Dispense bowls
- Dispense cereal only into the bowls
- Modify the amount of cereal to be dispensed
- Receive communication/inputs from the user

Function and Feel



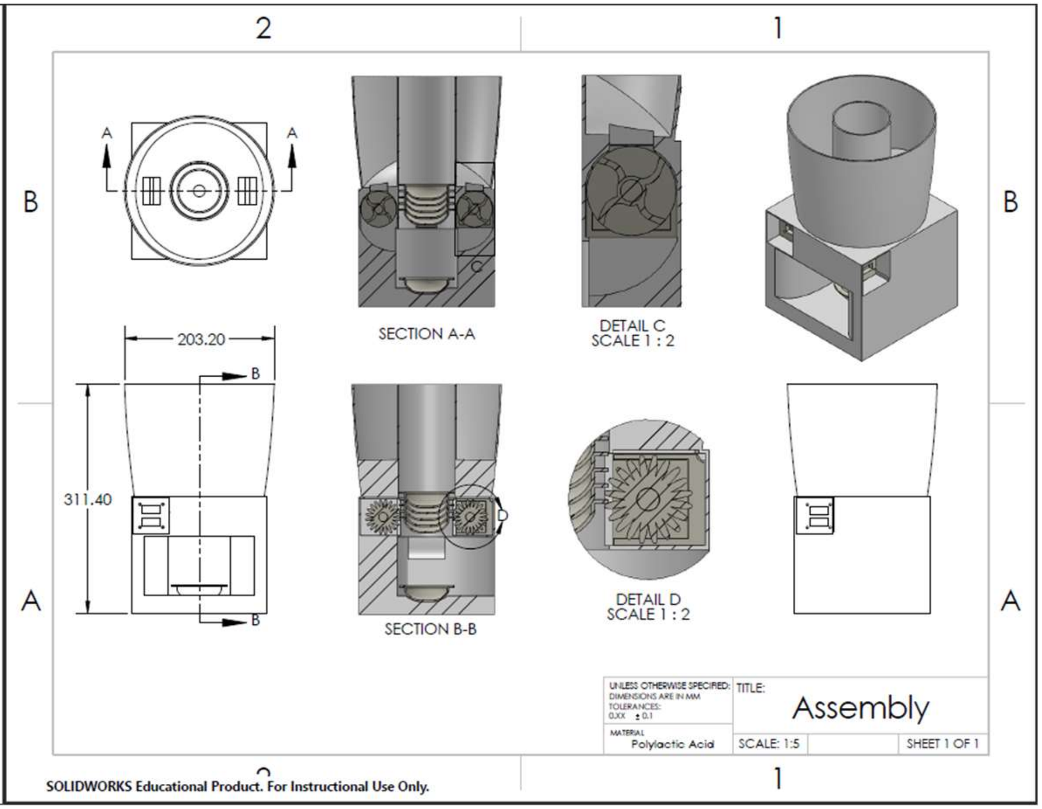
ITEM NO.	PART NAME	QTY.
1	Upper Body	1
2	Lower Body	1
3	Cereal Wheel	2
4	Bowl Gear	2
5	Power Shaft	4
6	Nema 17 Stepper Motor	4
7	Motor Shim CW	2
8	Motor Shim BG	2
9	Bowl	6

TITLE: Exploded View

SIZE A DWG. NO. REV

SCALE: 1:4 WEIGHT: SHEET 1 OF 1

SOLIDWORKS Educational Product. For Instructional Use Only.



SECTION A-A

DETAIL C
SCALE 1 : 2

SECTION B-B

DETAIL D
SCALE 1 : 2

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MM
TOLERANCES:
0.XX ± 0.1

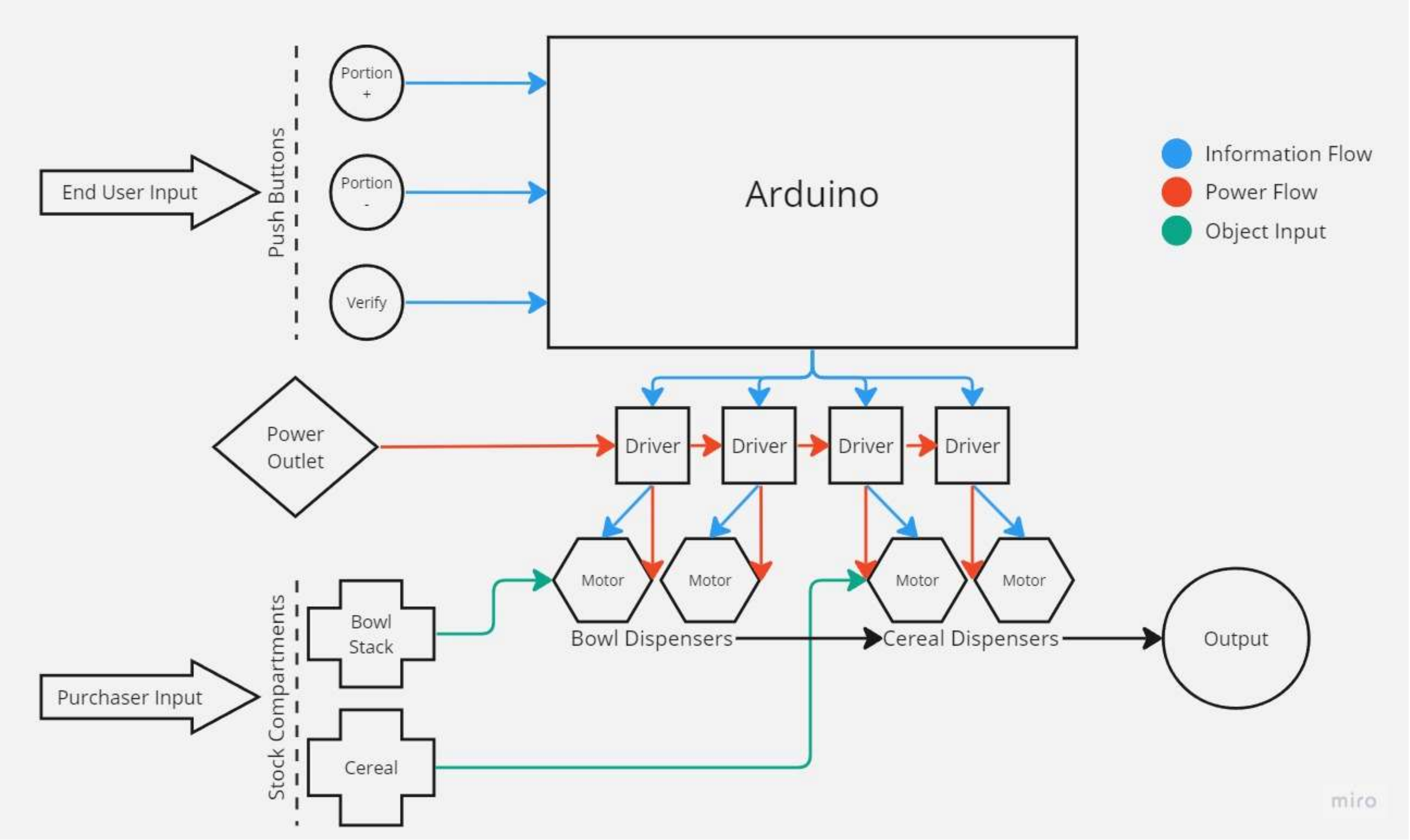
MATERIAL
Polylactic Acid

TITLE: Assembly

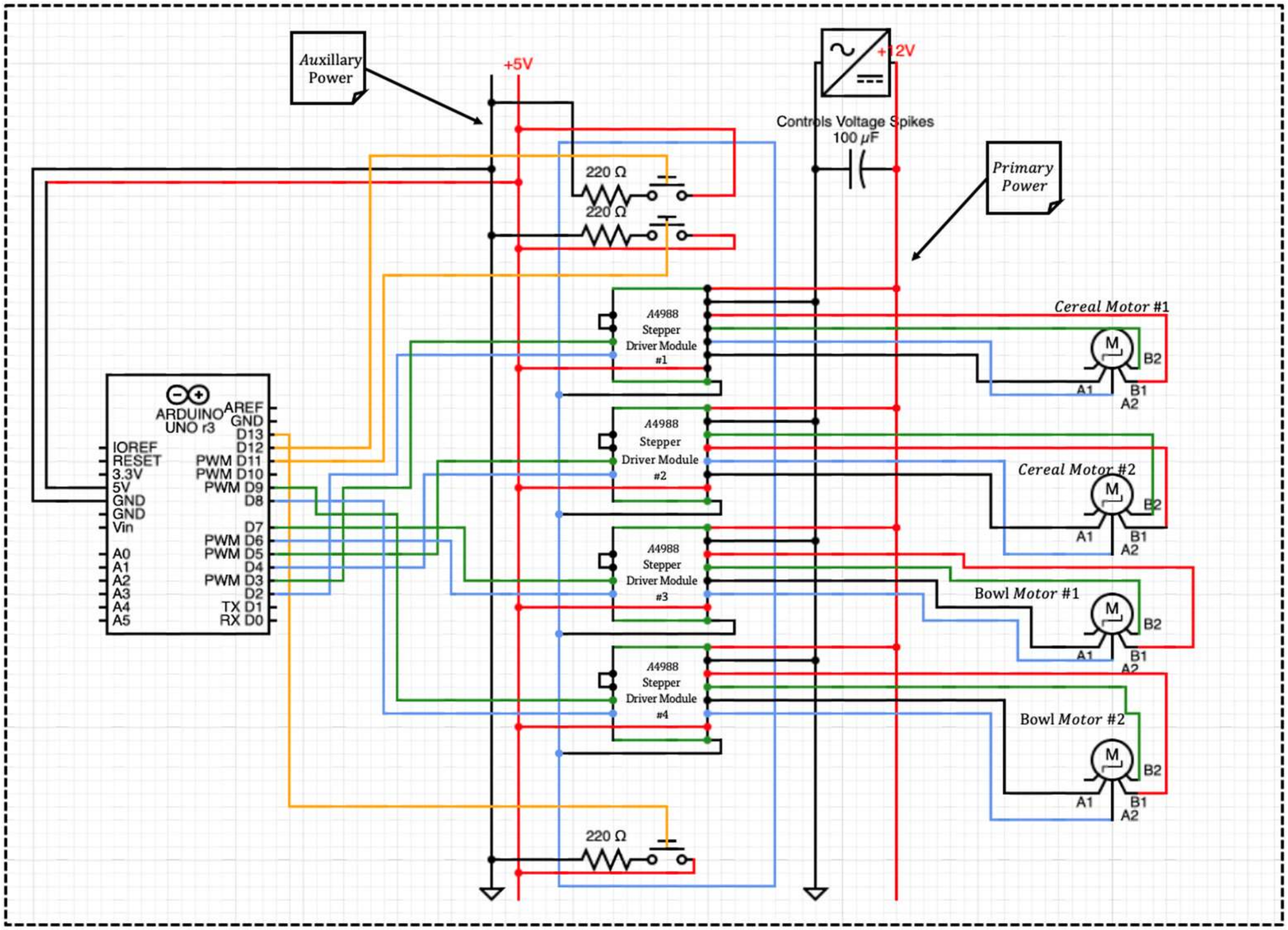
SCALE: 1:5 SHEET 1 OF 1

SOLIDWORKS Educational Product. For Instructional Use Only.

Function Flow Diagram



Electrical Diagram



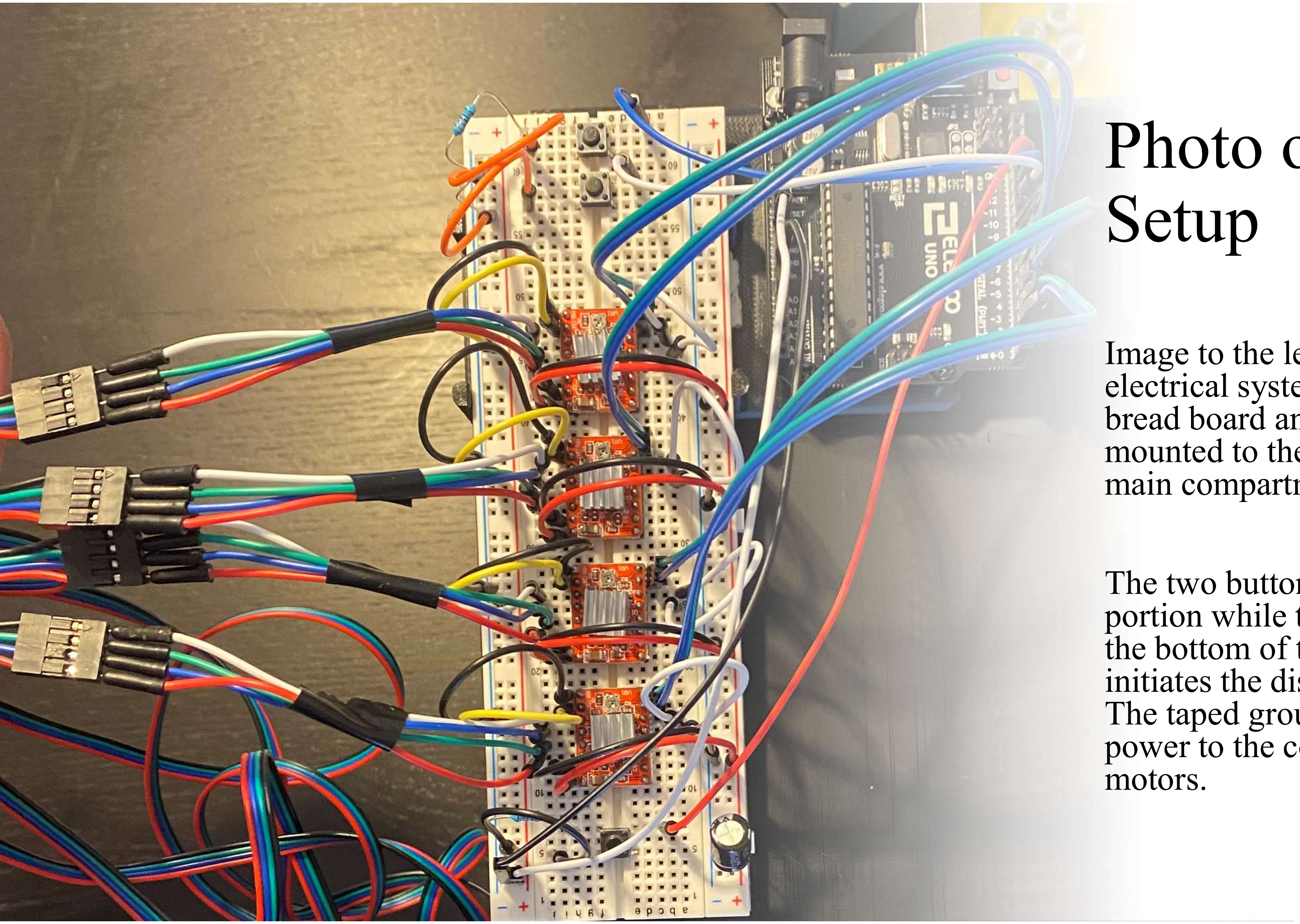
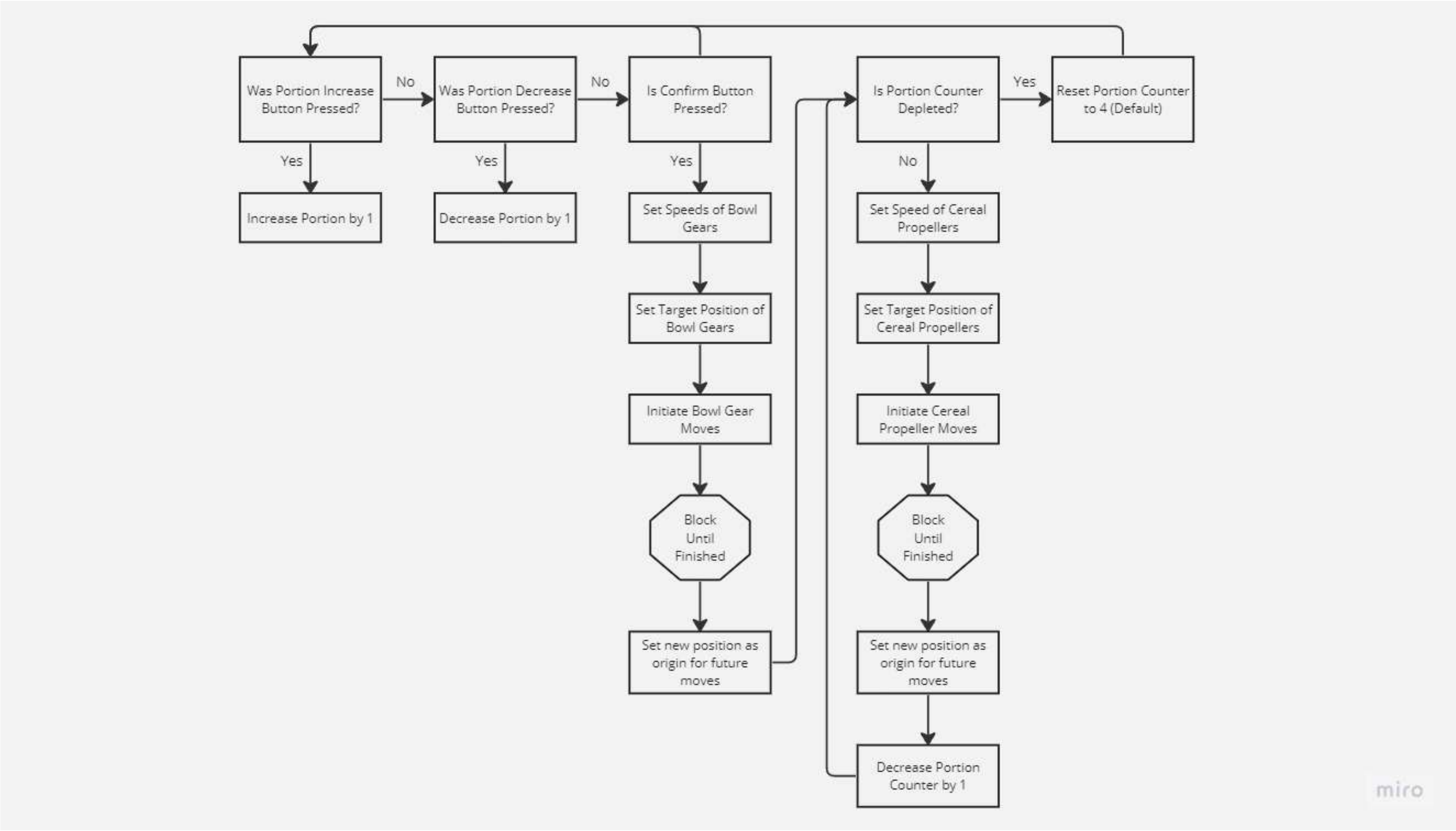


Photo of Current Setup

Image to the left shows how the electrical system is wired. The bread board and Arduino are both mounted to the backside of the main compartment by a bracket.

The two buttons at the top control portion while the button located at the bottom of the bread-board initiates the dispensing process. The taped groups of 4 wires supply power to the coils of the 4 stepper motors.

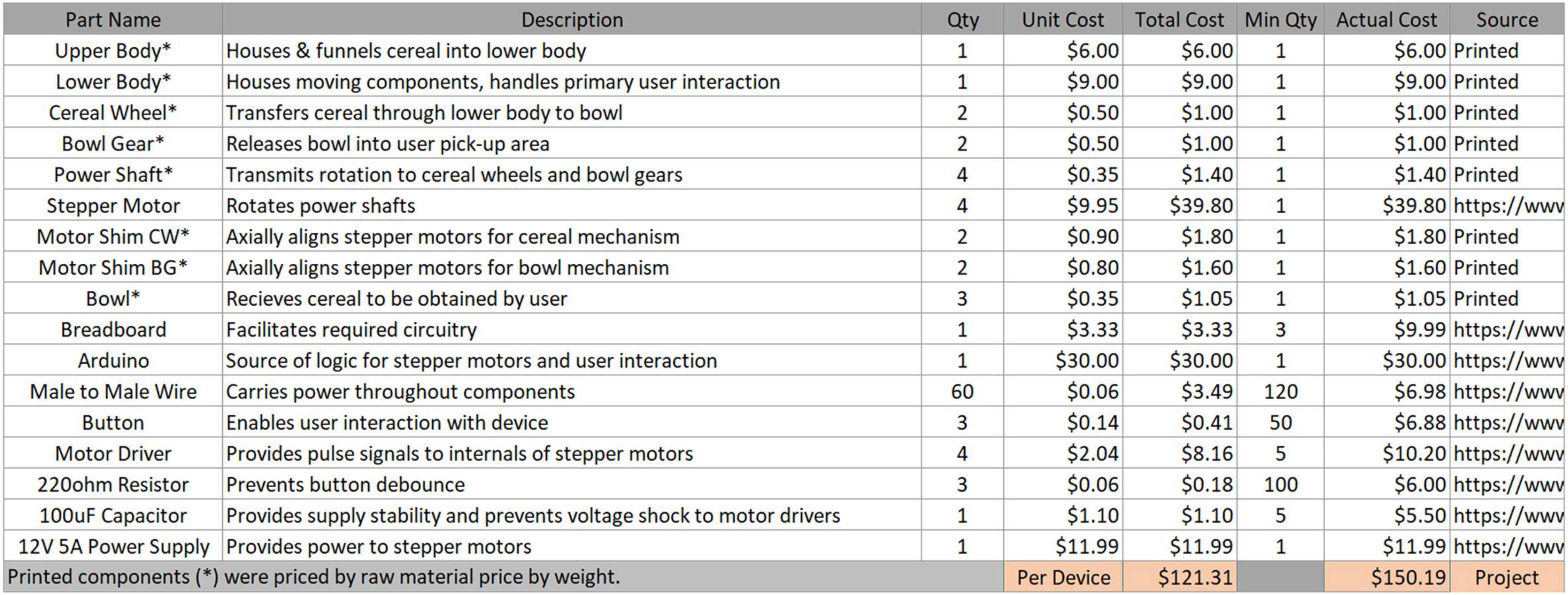
Code Flow Diagram


Was Portion Increase Button Pressed?
No
Was Portion Decrease Button Pressed?
No
Is Confirm Button Pressed?
Is Portion Counter Depleted?
Yes
Reset Portion Counter to 4 (Default)
Yes
Increase Portion by 1
Yes
Decrease Portion by 1
Yes
Set Speeds of Bowl Gears
Set Target Position of Bowl Gears
Initiate Bowl Gear Moves
Block Until Finished
Set new position as origin for future moves
No
Set Speed of Cereal Propellers
Set Target Position of Cereal Propellers
Initiate Cereal Propeller Moves
Block Until Finished
Set new position as origin for future moves
Decrease Portion Counter by 1
miro

Test and Validation Video

*Demo Video May Not Work Depending on Viewing Platform

Bill of Materials

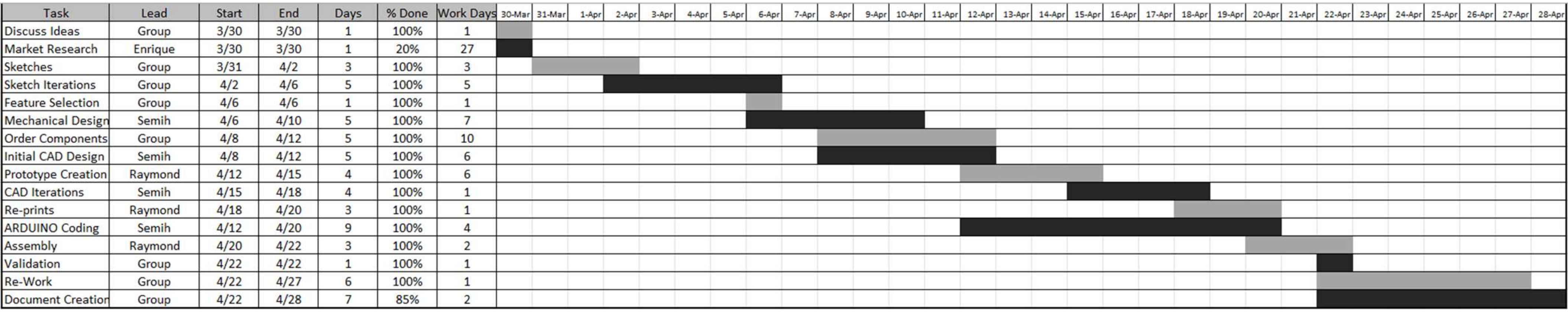


Part Name	Description	Qty	Unit Cost	Total Cost	Min Qty	Actual Cost	Source
Upper Body*	Houses & funnels cereal into lower body	1	$6.00	$6.00	1	$6.00	Printed
Lower Body*	Houses moving components, handles primary user interaction	1	$9.00	$9.00	1	$9.00	Printed
Cereal Wheel*	Transfers cereal through lower body to bowl	2	$0.50	$1.00	1	$1.00	Printed
Bowl Gear*	Releases bowl into user pick-up area	2	$0.50	$1.00	1	$1.00	Printed
Power Shaft*	Transmits rotation to cereal wheels and bowl gears	4	$0.35	$1.40	1	$1.40	Printed
Stepper Motor	Rotates power shafts	4	$9.95	$39.80	1	$39.80	https://wwv
Motor Shim CW*	Axially aligns stepper motors for cereal mechanism	2	$0.90	$1.80	1	$1.80	Printed
Motor Shim BG*	Axially aligns stepper motors for bowl mechanism	2	$0.80	$1.60	1	$1.60	Printed
Bowl*	Recieves cereal to be obtained by user	3	$0.35	$1.05	1	$1.05	Printed
Breadboard	Facilitates required circuitry	1	$3.33	$3.33	3	$9.99	https://wwv
Arduino	Source of logic for stepper motors and user interaction	1	$30.00	$30.00	1	$30.00	https://wwv
Male to Male Wire	Carries power throughout components	60	$0.06	$3.49	120	$6.98	https://wwv
Button	Enables user interaction with device	3	$0.14	$0.41	50	$6.88	https://wwv
Motor Driver	Provides pulse signals to internals of stepper motors	4	$2.04	$8.16	5	$10.20	https://wwv
220ohm Resistor	Prevents button debounce	3	$0.06	$0.18	100	$6.00	https://wwv
100uF Capacitor	Provides supply stability and prevents voltage shock to motor drivers	1	$1.10	$1.10	5	$5.50	https://wwv
12V 5A Power Supply	Provides power to stepper motors	1	$11.99	$11.99	1	$11.99	https://wwv
Printed components (*) were priced by raw material price by weight.			Per Device	$121.31		$150.19	Project

Future Development Goals

- Increase clearance of bowl gear compartments in lower body
- Increase bowl gear obstruction into bowl stack compartment
- Combine motor and logic power supplies into one source
- Replace cereal compartment with see-through material
- Add access panel and move circuitry into the lower body
- Upgrade motor drivers for silent operation
- Redesign cereal to propeller funnel to minimize jamming
- Add single digit display for portion counter feedback

Project Timeline

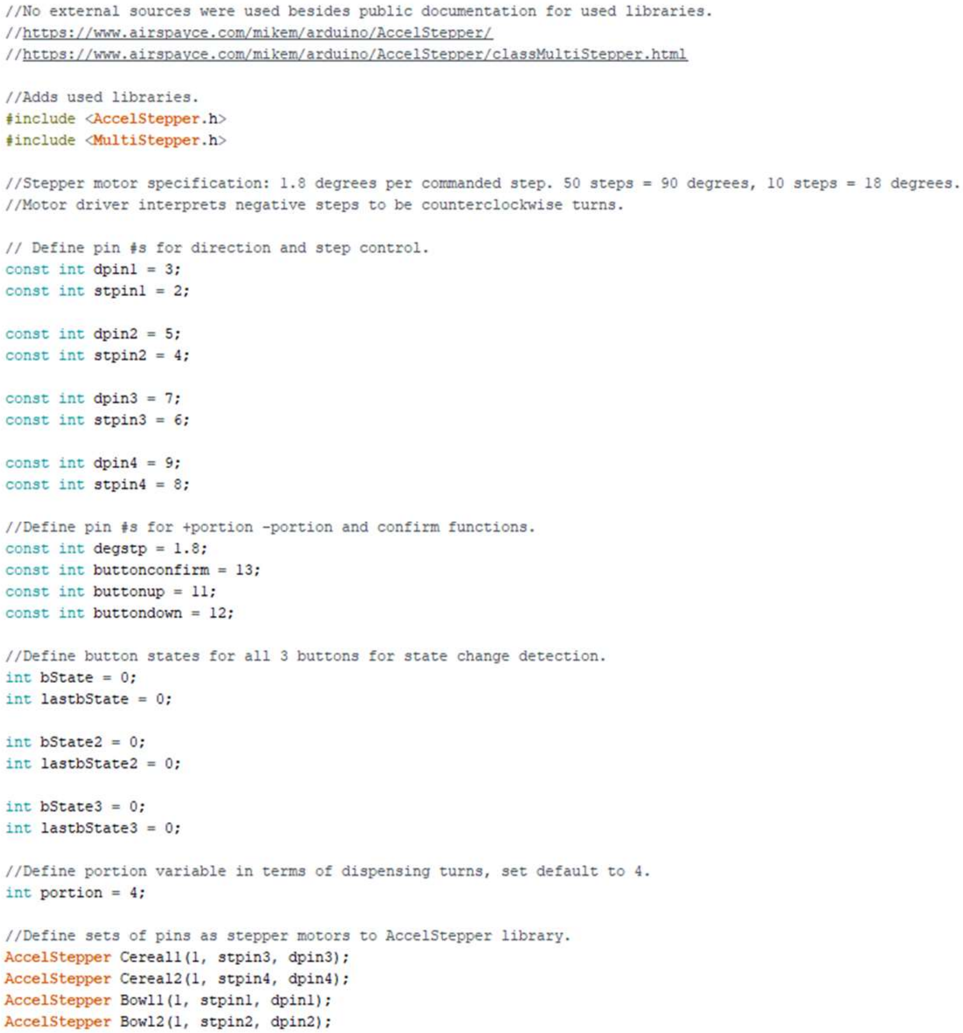


Task	Lead	Start	End	Days	% Done	Work Days	30-Mar	31-Mar	1-Apr	2-Apr	3-Apr	4-Apr	5-Apr	6-Apr	7-Apr	8-Apr	9-Apr	10-Apr	11-Apr	12-Apr	13-Apr	14-Apr	15-Apr	16-Apr	17-Apr	18-Apr	19-Apr	20-Apr	21-Apr	22-Apr	23-Apr	24-Apr	25-Apr	26-Apr	27-Apr	28-Apr
Discuss Ideas	Group	3/30	3/30	1	100%	1	▨																													
Market Research	Enrique	3/30	3/30	1	20%	27	■																													
Sketches	Group	3/31	4/2	3	100%	3		▨	▨	▨																										
Sketch Iterations	Group	4/2	4/6	5	100%	5				■	■	■	■	■																						
Feature Selection	Group	4/6	4/6	1	100%	1								▨																						
Mechanical Design	Semih	4/6	4/10	5	100%	7								■	■	■	■	■																		
Order Components	Group	4/8	4/12	5	100%	10										▨	▨	▨	▨	▨																
Initial CAD Design	Semih	4/8	4/12	5	100%	6										■	■	■	■	■																
Prototype Creation	Raymond	4/12	4/15	4	100%	6														▨	▨	▨	▨													
CAD Iterations	Semih	4/15	4/18	4	100%	1																	■	■	■	■										
Re-prints	Raymond	4/18	4/20	3	100%	1																				▨	▨	▨								
ARDUINO Coding	Semih	4/12	4/20	9	100%	4														■	■	■	■	■	■	■	■	■								
Assembly	Raymond	4/20	4/22	3	100%	2																						▨	▨	▨						
Validation	Group	4/22	4/22	1	100%	1																								■						
Re-Work	Group	4/22	4/27	6	100%	1																								▨	▨	▨	▨	▨	▨	
Document Creatior	Group	4/22	4/28	7	85%	2																								■	■	■	■	■	■	■

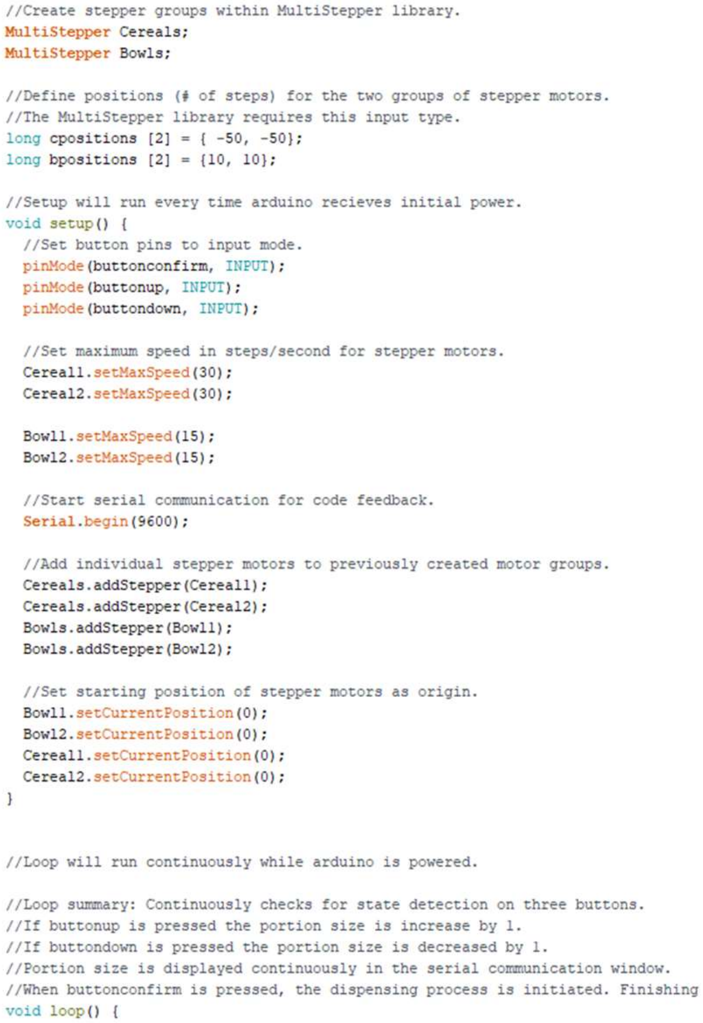
Thank you !

Any Questions?

Arduino Code

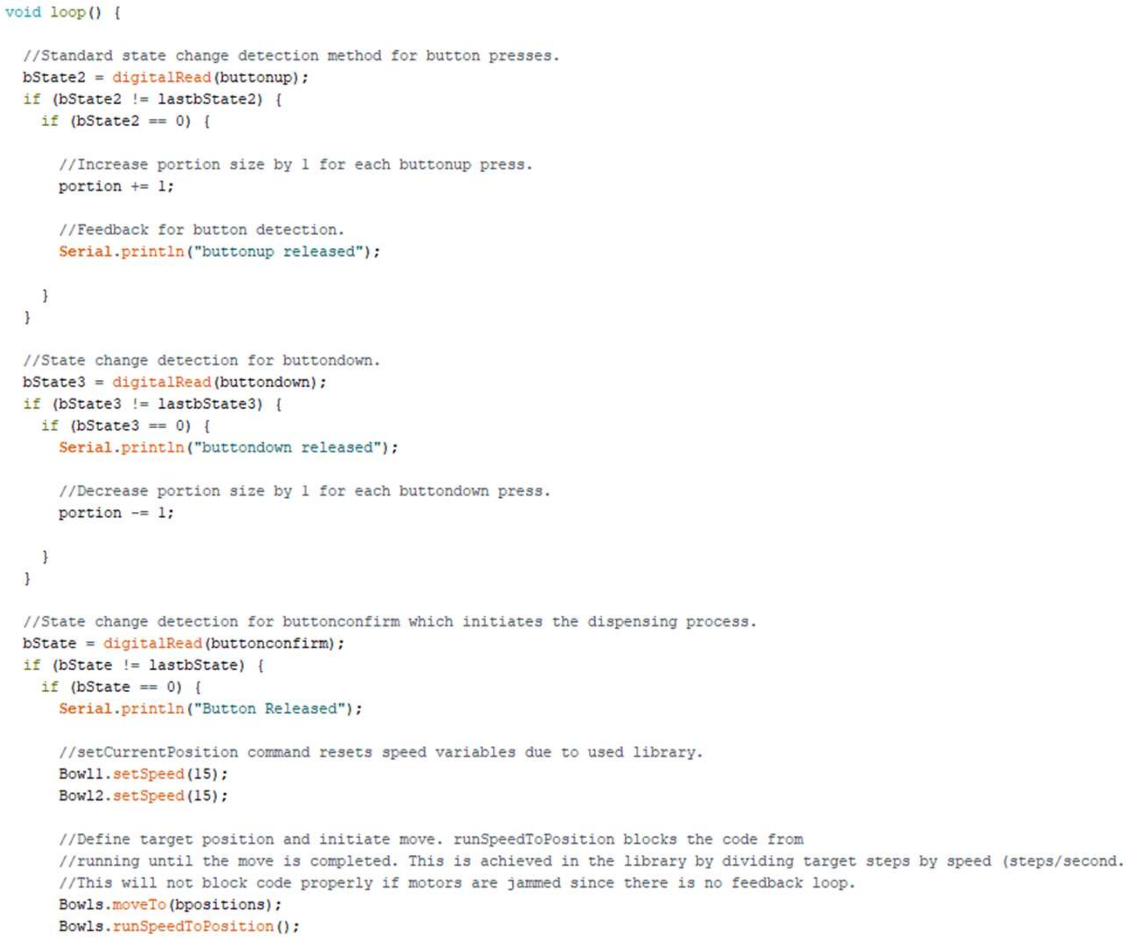


```
//No external sources were used besides public documentation for used libraries.
//https://www.airspayce.com/mikem/arduino/AccelStepper/
//https://www.airspayce.com/mikem/arduino/AccelStepper/classMultiStepper.html

//Adds used libraries.
#include <AccelStepper.h>
#include <MultiStepper.h>

//Stepper motor specification: 1.8 degrees per commanded step. 50 steps = 90 degrees, 10 steps = 18 degrees.
//Motor driver interprets negative steps to be counterclockwise turns.

// Define pin #s for direction and step control.
const int dpin1 = 3;
const int stpin1 = 2;

const int dpin2 = 5;
const int stpin2 = 4;

const int dpin3 = 7;
const int stpin3 = 6;

const int dpin4 = 9;
const int stpin4 = 8;

//Define pin #s for +portion -portion and confirm functions.
const int degstp = 1.8;
const int buttonconfirm = 13;
const int buttonup = 11;
const int buttondown = 12;

//Define button states for all 3 buttons for state change detection.
int bState = 0;
int lastbState = 0;

int bState2 = 0;
int lastbState2 = 0;

int bState3 = 0;
int lastbState3 = 0;

//Define portion variable in terms of dispensing turns, set default to 4.
int portion = 4;

//Define sets of pins as stepper motors to AccelStepper library.
AccelStepper Cereal1(1, stpin3, dpin3);
AccelStepper Cereal2(1, stpin4, dpin4);
AccelStepper Bowl1(1, stpin1, dpin1);
AccelStepper Bowl2(1, stpin2, dpin2);
```



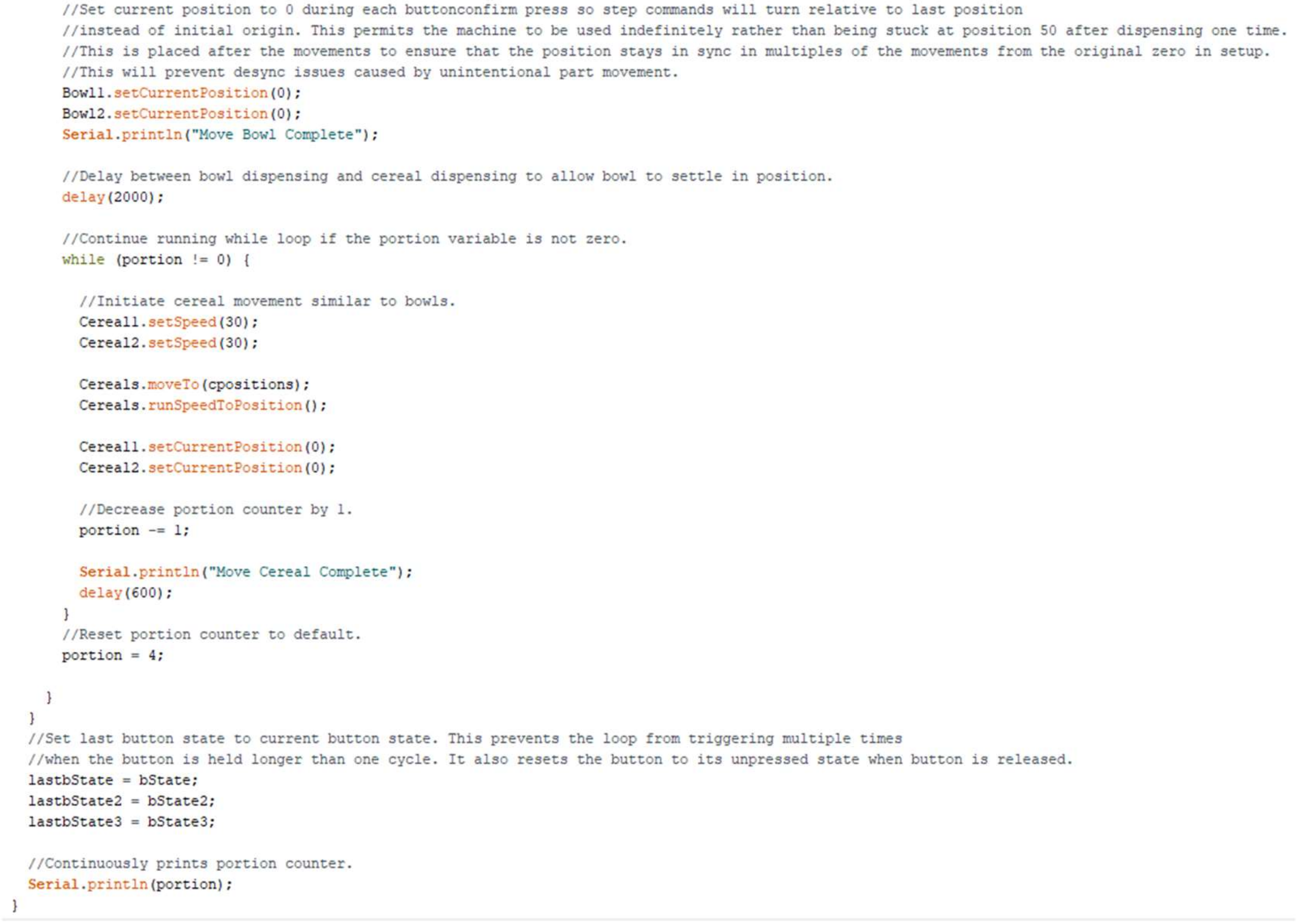
```
//Create stepper groups within MultiStepper library.
MultiStepper Cereals;
MultiStepper Bowls;

//Define positions (# of steps) for the two groups of stepper motors.
//The MultiStepper library requires this input type.
long cpositions [2] = { -50, -50};
long bpositions [2] = {10, 10};

//Setup will run every time arduino recieves initial power.
void setup() {
  //Set button pins to input mode.
  pinMode(buttonconfirm, INPUT);
  pinMode(buttonup, INPUT);
  pinMode(buttondown, INPUT);

  //Set maximum speed in steps/second for stepper motors.
  Cereal1.setMaxSpeed(30);
  Cereal2.setMaxSpeed(30);

  Bowl1.setMaxSpeed(15);
  Bowl2.setMaxSpeed(15);

  //Start serial communication for code feedback.
  Serial.begin(9600);

  //Add individual stepper motors to previously created motor groups.
  Cereals.addStepper(Cereal1);
  Cereals.addStepper(Cereal2);
  Bowls.addStepper(Bowl1);
  Bowls.addStepper(Bowl2);

  //Set starting position of stepper motors as origin.
  Bowl1.setCurrentPosition(0);
  Bowl2.setCurrentPosition(0);
  Cereal1.setCurrentPosition(0);
  Cereal2.setCurrentPosition(0);
}


//Loop will run continuously while arduino is powered.

//Loop summary: Continuously checks for state detection on three buttons.
//If buttonup is pressed the portion size is increase by 1.
//If buttondown is pressed the portion size is decreased by 1.
//Portion size is displayed continuously in the serial communication window.
//When buttonconfirm is pressed, the dispensing process is initiated. Finishing
void loop() {
```

Arduino Code Continued



```
void loop() {

  //Standard state change detection method for button presses.
  bState2 = digitalRead(buttonup);
  if (bState2 != lastbState2) {
    if (bState2 == 0) {

      //Increase portion size by 1 for each buttonup press.
      portion += 1;

      //Feedback for button detection.
      Serial.println("buttonup released");

    }
  }

  //State change detection for buttondown.
  bState3 = digitalRead(buttondown);
  if (bState3 != lastbState3) {
    if (bState3 == 0) {
      Serial.println("buttondown released");

      //Decrease portion size by 1 for each buttondown press.
      portion -= 1;

    }
  }

  //State change detection for buttonconfirm which initiates the dispensing process.
  bState = digitalRead(buttonconfirm);
  if (bState != lastbState) {
    if (bState == 0) {
      Serial.println("Button Released");

      //setCurrentPosition command resets speed variables due to used library.
      Bowl1.setSpeed(15);
      Bowl2.setSpeed(15);

      //Define target position and initiate move. runSpeedToPosition blocks the code from
      //running until the move is completed. This is achieved in the library by dividing target steps by speed (steps/second.
      //This will not block code properly if motors are jammed since there is no feedback loop.
      Bowls.moveTo(bpositions);
      Bowls.runSpeedToPosition();
```

Arduino Code Continued



```
      //Set current position to 0 during each buttonconfirm press so step commands will turn relative to last position
      //instead of initial origin. This permits the machine to be used indefinitely rather than being stuck at position 50 after dispensing one time.
      //This is placed after the movements to ensure that the position stays in sync in multiples of the movements from the original zero in setup.
      //This will prevent desync issues caused by unintentional part movement.
      Bowl1.setCurrentPosition(0);
      Bowl2.setCurrentPosition(0);
      Serial.println("Move Bowl Complete");

      //Delay between bowl dispensing and cereal dispensing to allow bowl to settle in position.
      delay(2000);

      //Continue running while loop if the portion variable is not zero.
      while (portion != 0) {

        //Initiate cereal movement similar to bowls.
        Cereal1.setSpeed(30);
        Cereal2.setSpeed(30);

        Cereals.moveTo(cpositions);
        Cereals.runSpeedToPosition();

        Cereal1.setCurrentPosition(0);
        Cereal2.setCurrentPosition(0);

        //Decrease portion counter by 1.
        portion -= 1;

        Serial.println("Move Cereal Complete");
        delay(600);
      }
      //Reset portion counter to default.
      portion = 4;

    }
  }
  //Set last button state to current button state. This prevents the loop from triggering multiple times
  //when the button is held longer than one cycle. It also resets the button to its unpressed state when button is released.
  lastbState = bState;
  lastbState2 = bState2;
  lastbState3 = bState3;

  //Continuously prints portion counter.
  Serial.println(portion);
}
```

Additional Pictures