Mechatronic Design
Fall 2021 – Project Assignment #4
Final class deliverables

By: Maria Padilla, Levi Evans, Maikel Masoud

Opportunity

Who didn't like candy as an award for getting a star in class? Our simple candy dispensing device for kids is a perfect reward for the amazing kids. Dispenses just the right amount of candy and can fit in your cabinet.

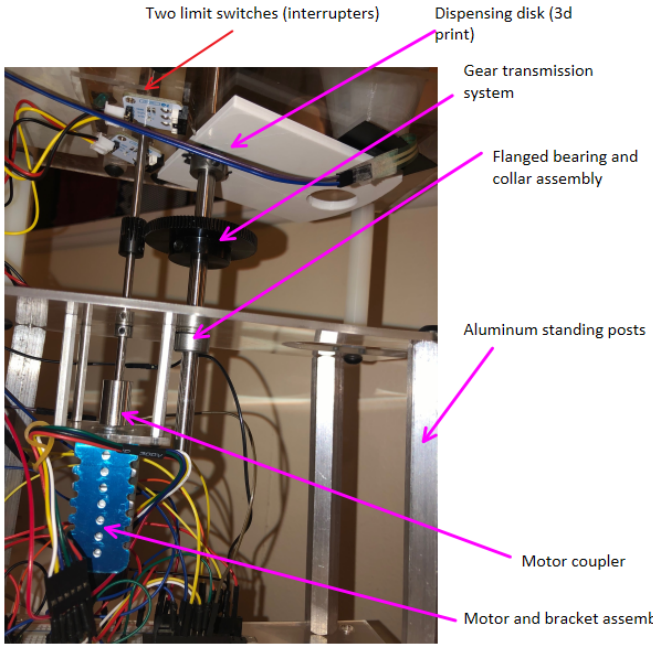
High Level Strategy

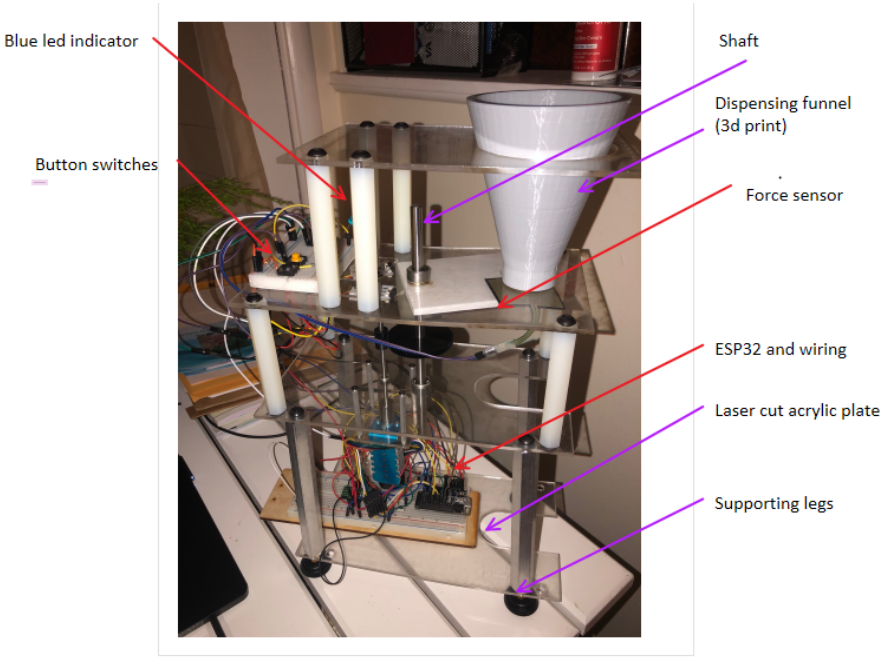
Plug into the power source.
Press button 1 when you are ready.
The device will check for refills and notify you.
Press button 2 for a special treat.
Hold the cup under the curved notch.
Enjoy!
Press button 2 again for a second round or button 1 to turn it off.

Comparison Between Initial and Final Design

	Initial Design	Final Design
PI control	PI to control the amount dispensed by controlling the motor speed.	Open loop and using time to control the dispensing amount (motor issue with starting speed)
Message displaying	LCD using I2c pins but not enough pins in ESP32	Messages are displayed on the computer screen (Arduino IDE)

Photos





Function-Critical Decisions

The key calculation for our system as it relates to the function of the mechanism is whether our motor has the necessary torque to rotate the gear train and the disk that rotates to dispense the candy. We took into account the weight of the candy that is applied as a load to the disk when it is closed. We also took into account the forces applied to the bearings by the system and vice versa. We also included the force propagation through the gear train to the motor. Using this model and Newton's Laws to derive the force and torque equations that characterize the system, we were able to determine the maximum radial forces applied to the bearing and confirm that the motor we considered using could supply the needed torque and not stall out. We did this by solving the system of equations that we derived in Matlab. In these calculations, we made the assumption that there would be a maximum of 150 grams of candy. We found that the maximum radial load on the bearings would be well below the rating found in the specification sheets for the ball bearings. Further, we found that half the motor stall torque given by the motor specification sheet would be sufficient to drive the system under the given load. Thus, any motor more powerful than that motor would be sufficient. We ultimately ended up using a stronger version of the same motor from Pololu (https://www.pololu.com/product/4802). The stall torque of the motor we used for the calculations is 0.127kg*cm at 6V. The stall torque of the motor we ended up using in our bill of materials has a stall torque of 2.3 kg*cm at 6V, which exceeds the requirement set out in our calculations. The specifications of the final motor used in the mechanism is given in Appendix B. The Matlab code used and the results as well as our force/torque diagram are given in the Appendix A.

What We Wish We Had Done Differently

We were able to get our machine to work consistently which was a great accomplishment. In hindsight there are a few things that we would have done differently. After working with the limit switches we thought it might have been a better solution to use a stepper motor to control the aperture. We also wanted to incorporate an LCD instead of the IDE so that the user would know the status of their dispense. The motor was mounted upright and attached to the bottom of one of the plates. We would have liked to have built a better housing for the motor to make it more secure. Additionally we used a rigid shaft coupler which worked fine for our machine, because the machine was quite flexible, but in a future project, a flexible shaft coupler is ideal.

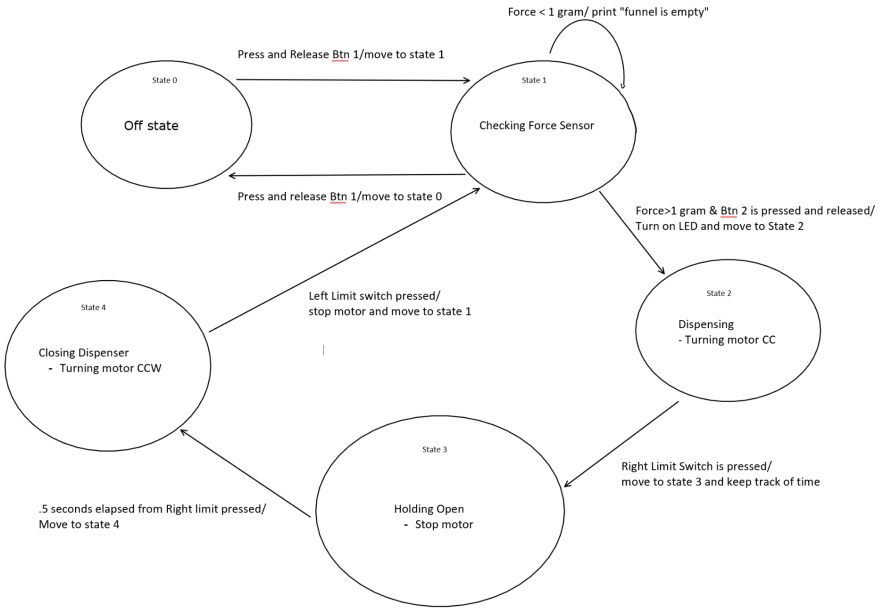


Figure 2: State Diagram

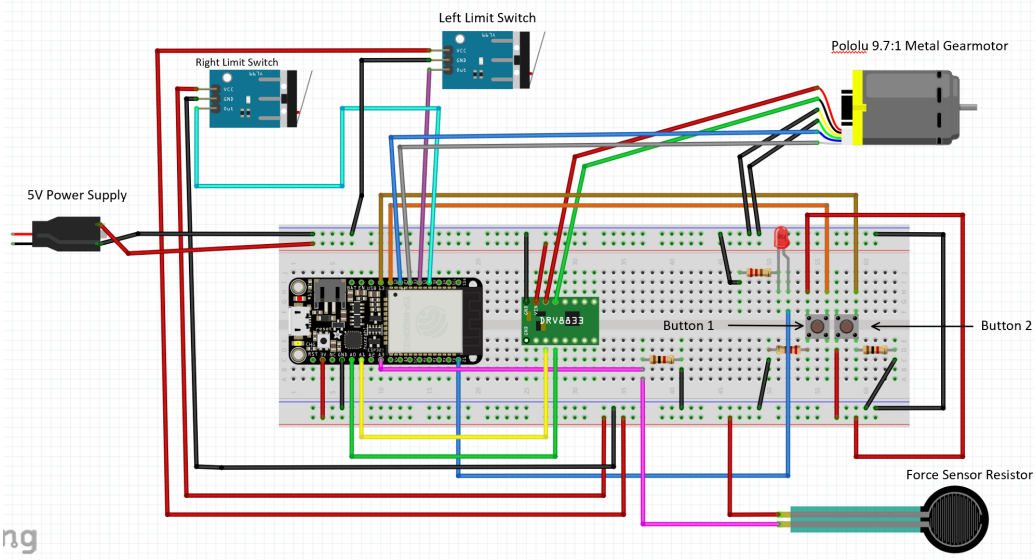
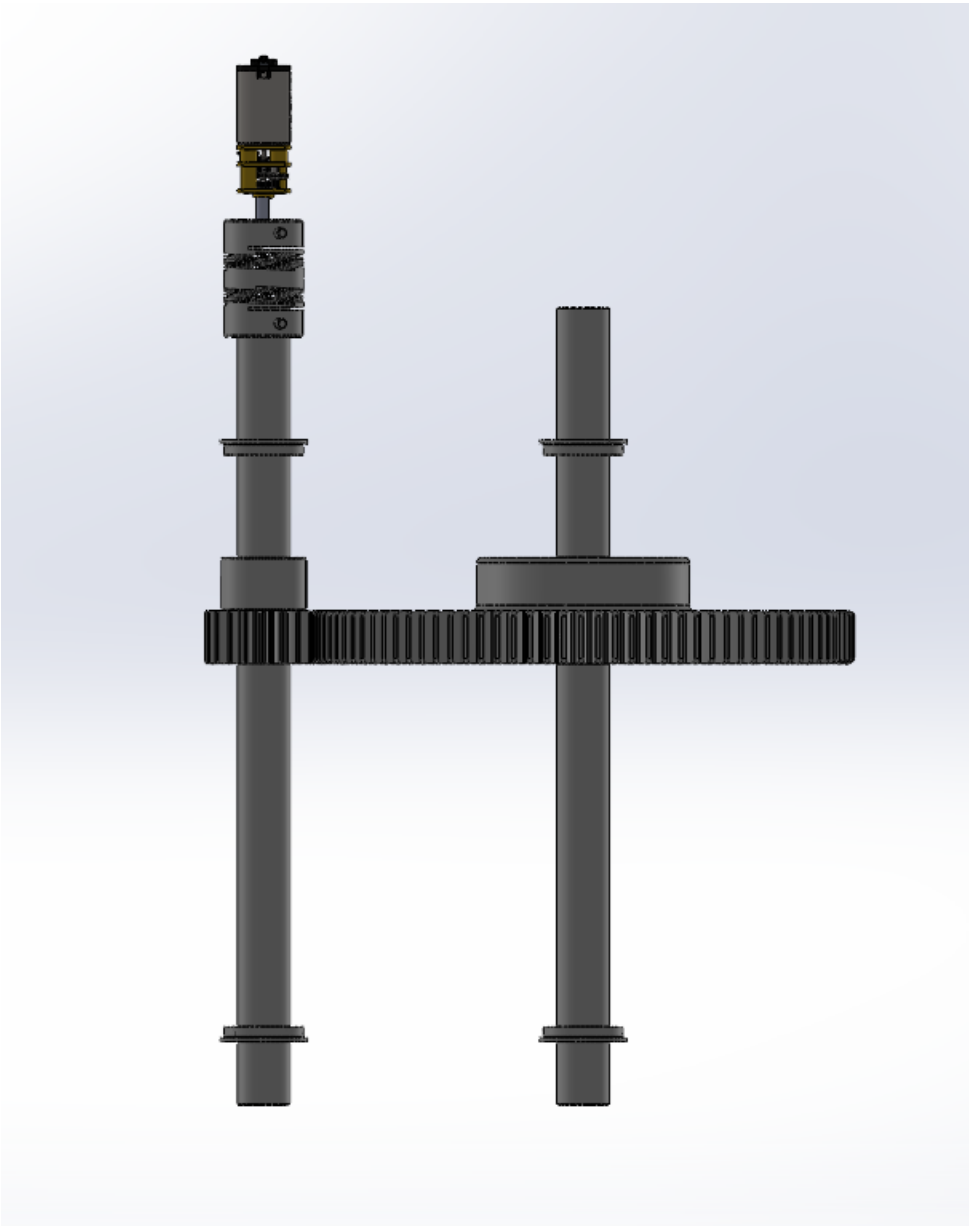


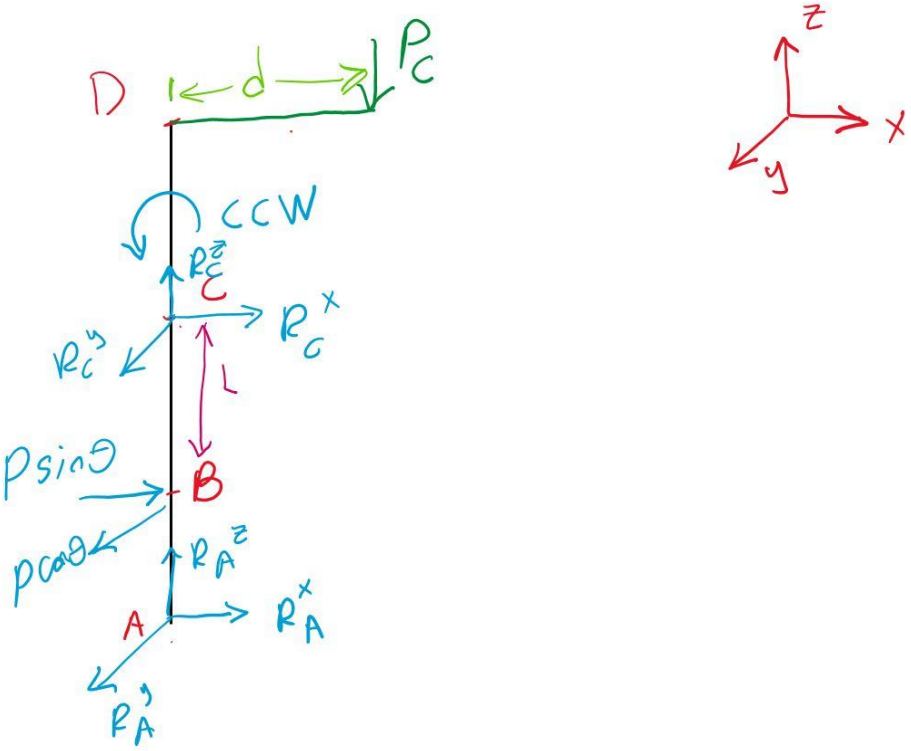
Figure 3: Wiring Diagram

Appendices

Appendix A: Function Critical Calculations



Profile View of Assembly



A: Bottom bearing

B: First stage gear

C: Top bearing

D: Disk

Force Diagram of Assembly

Project Specification:

```
%Assumptions:
% All components including gears, bearing, disk and shaft have
 negligable
% weight and the thrust bearing can withstand all the vertical loads.


d=0.075; %Distance from axis of rotation to center mass of the candy m
Pc=1.47; %Weight of coulmn of candy above the dispensing hole N
R=5; %Ger ratio
L=0.05; %Distance between A, B, C and D in m
Tm=0.127/2; % half of Torque from motor N.M
theta=20; %Gear pressure angle in deg
Rp=20/1000; %Pinion gear diameter m



% Calculations:
P=Tm/(Rp*cosd(theta)); %Gear force from the motor N

% Moments about point A = 0:
syms Rc_x Rc_y RA_x RA_y

eq1=(P*sind(theta)*L)+(Rc_x*2*L)+(Pc*d)==0;
eq2=(P*cosd(theta)*L)+(Rc_y*2*L)==0;

Rc_X=double(solve(eq1,Rc_x))%Reaction in x-dir at C
Rc_Y=double(solve(eq2,Rc_y))%Reaction in y-dir at C

%Net forces equilibrium:
eq3=Rc_X+RA_x+(P*sind(theta))==0;
eq4=Rc_Y+RA_y+(P*cosd(theta))==0;

RA_X=double(solve(eq3,RA_x))
RA_Y=double(solve(eq4,RA_y))


Rc_X =

   -1.6803


Rc_Y =

   -1.5875


RA_X =
```

Matlab Code to Calculate Forces

Motor Specifications

https://www.pololu.com/product/4802

Dimensions

Size:	25D x 63L mm[1]
Weight:	95 g
Shaft diameter:	4 mm[2]

General specifications

Gear ratio:	9.68:1
No-load speed @ 6V:	1000 rpm[3]
No-load current @ 6V:	0.50 A[4]
Stall current @ 6V:	6.0 A[5]
Stall torque @ 6V:	2.3 kg·cm[5]
Max output power @ 6V:	5.9 W
Motor type:	6V, 6.0A stall (HP 6V)

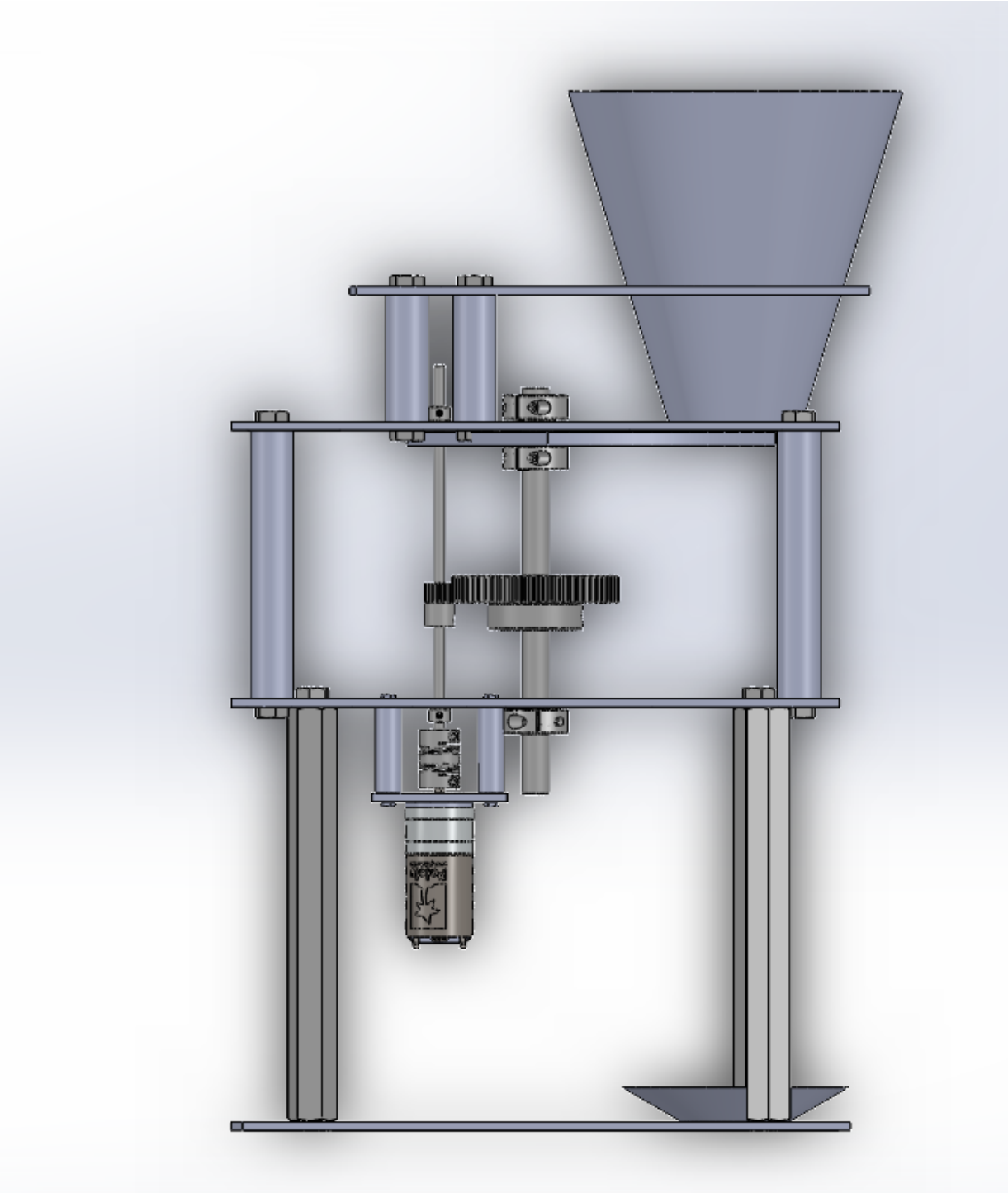
Performance at maximum efficiency

Max efficiency @ 6V:	44 %
Speed at max efficiency:	810 rpm
Torque at max efficiency:	0.45 kg·cm
Current at max efficiency:	1.4 A
Output power at max efficiency:	3.8 W

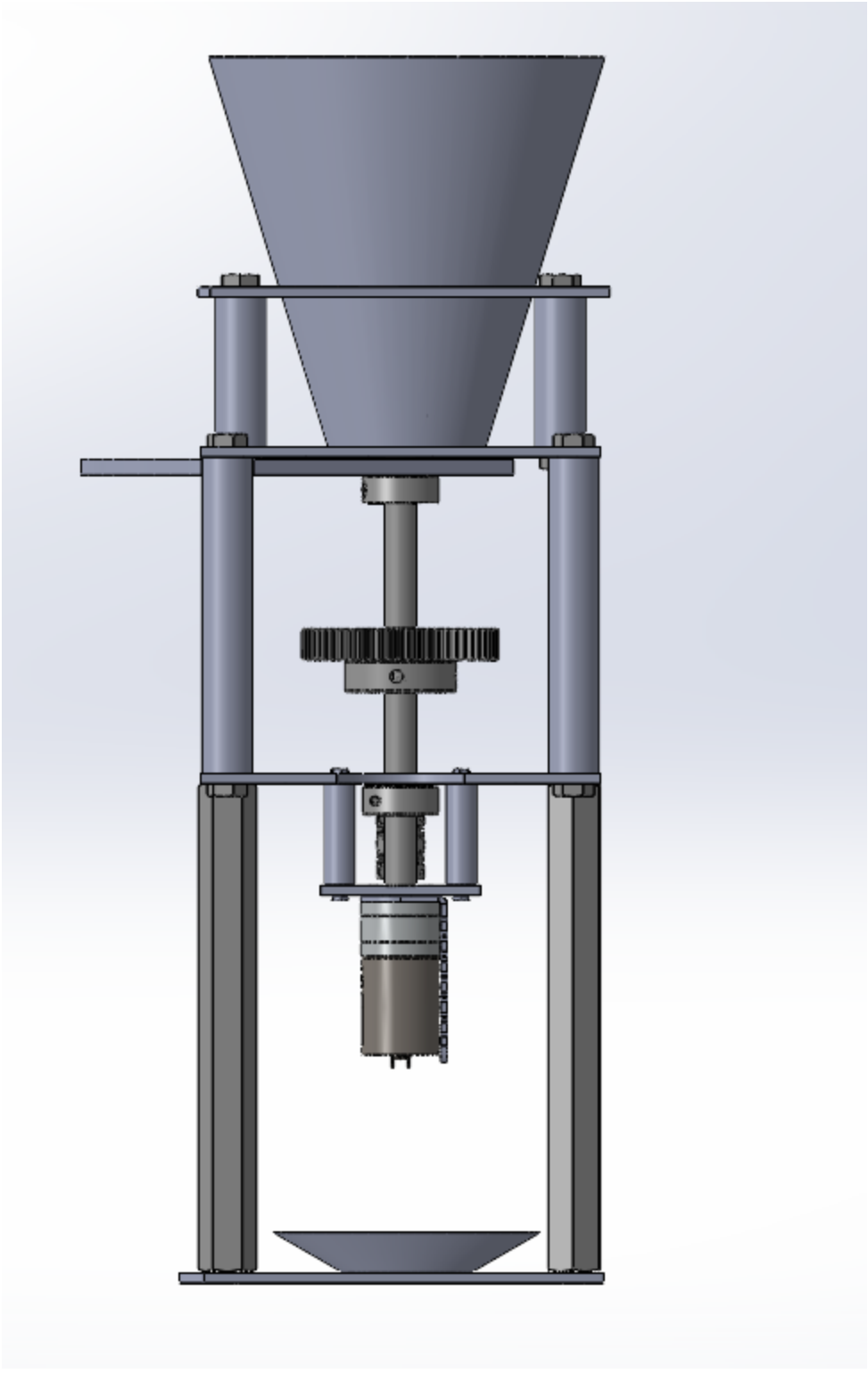
Appendix B: Bill Of Materials

Item	Link	Cost	Quantity
Leveling Mounts	https://www.mcmaster.com/2284T52/	$3.23	1.0
Nylon 6/6 Plastic Hex Standoff	https://www.mcmaster.com/92319A079/	$8.74	8.0
Nylon 6/6 Plastic Hex Standoff	https://www.mcmaster.com/92319A776/	$5.60	4.0
Aluminum Female Threaded Hex Standoff	https://www.mcmaster.com/95947A501/	$1.77	4.0
Female Threaded Hex Standoff	https://www.mcmaster.com/91780A266/	$6.33	4.0
Clear High-Strength UV-Resistant Acrylic	https://www.mcmaster.com/4615T93/	$8.00	5.0
18-8 Stainless Steel Button Head Hex Drive Screws	https://www.mcmaster.com/97763A811/	$6.67	2.0
Heat-Set Inserts for Plastic	https://www.mcmaster.com/94459A421/	$6.93	1.0
18-8 Stainless Steel Button Head Hex Drive Screws	https://www.mcmaster.com/97763A262/	$7.82	1.0
Heat-Set Inserts for Plastic	https://www.mcmaster.com/94459A380/	$9.69	1.0
Metal Gear - 8mm	https://www.mcmaster.com/2664N333/	$38.10	1.0
Rotary Shaft 200 mm	https://www.mcmaster.com/1265K37/	$15.50	1.0
Carbon Steel Set Screw Collar 4mm	https://www.mcmaster.com/6056N12/	$1.75	3.0
Belleville Disc Spring 4mm	https://www.mcmaster.com/96445K211/	$2.70	1.0
Belleville Disc Springs 2mm	https://www.mcmaster.com/94065K34/	$4.00	1.0
Carbon Steel Set Screw Collar 8mm	https://www.mcmaster.com/6056N16/	$1.94	3.0
Flanged Ball Bearing 8mm	https://www.mcmaster.com/57155K496/	$8.47	3.0
Flanged Ball Bearing 4mm	https://www.mcmaster.com/57155K437/	$9.53	3.0
Metal Gear -4mm	https://www.mcmaster.com/2664N314/	$14.80	1.0
Pololu 25D mm Metal Gearmotor Bracket Pair	https://www.pololu.com/product/2676	$7.45	1.0
9.7:1 Metal Gearmotor 25Dx63L mm HP 6V with 48 CPR Encoder	https://www.pololu.com/product/4802	$36.95	1.0
Double Sided Tape	https://www.amazon.com/dp/B091DNKSTV?psc=1&ref=ppx_yo2_dt_b_product_details	$12.98	1.0
FORCE SENSING RESISTOR	https://www.amazon.com/dp/B00B887DBC?psc=1&ref=ppx_yo2_dt_b_product_details	$12.99	1.0
End-Stop Switch Module	https://www.amazon.com/dp/B07HGCVZ1W?psc=1&ref=ppx_yo2_dt_b_product_details	$10.95	1.0
Total		$427.22	

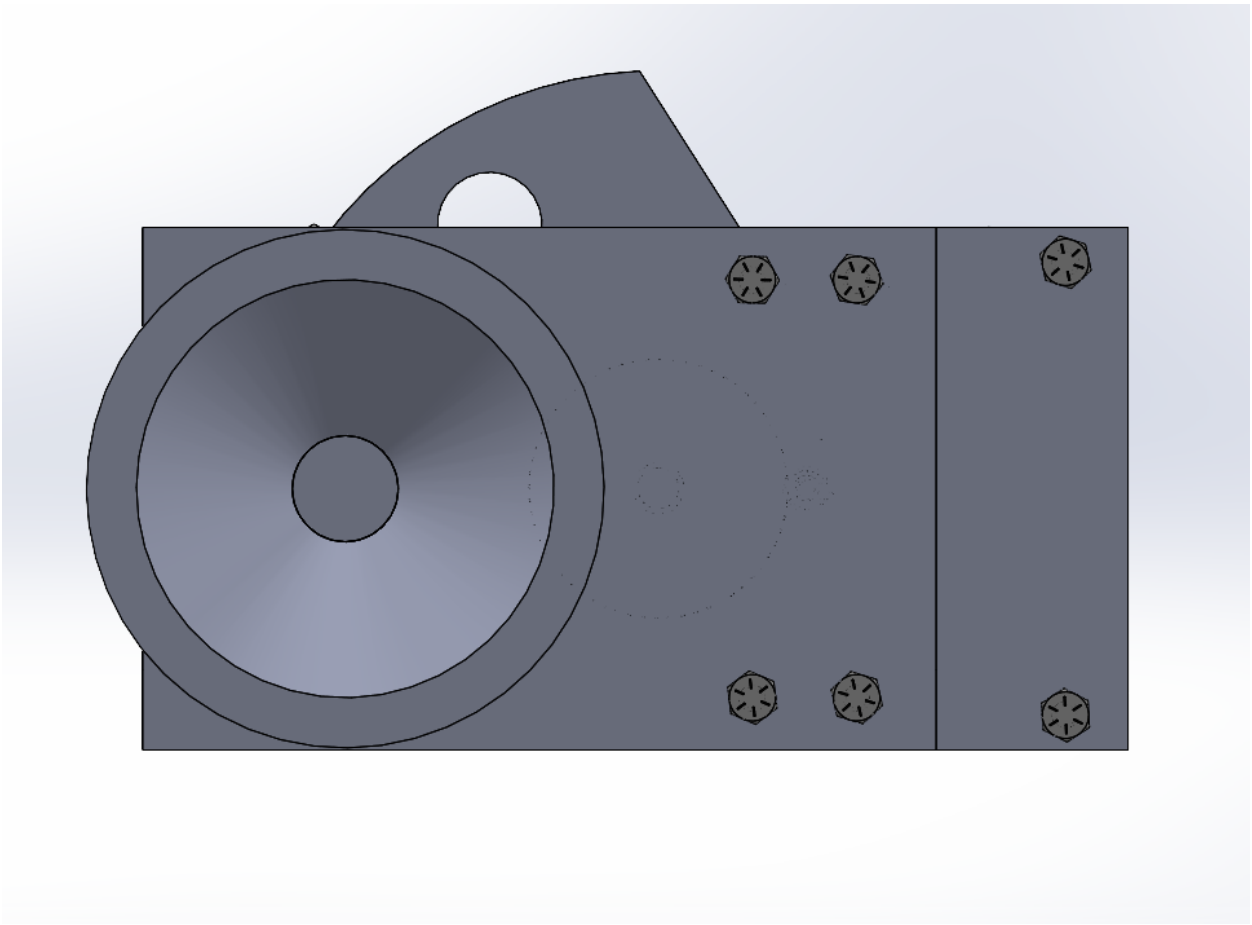
Appendix C: CAD Images



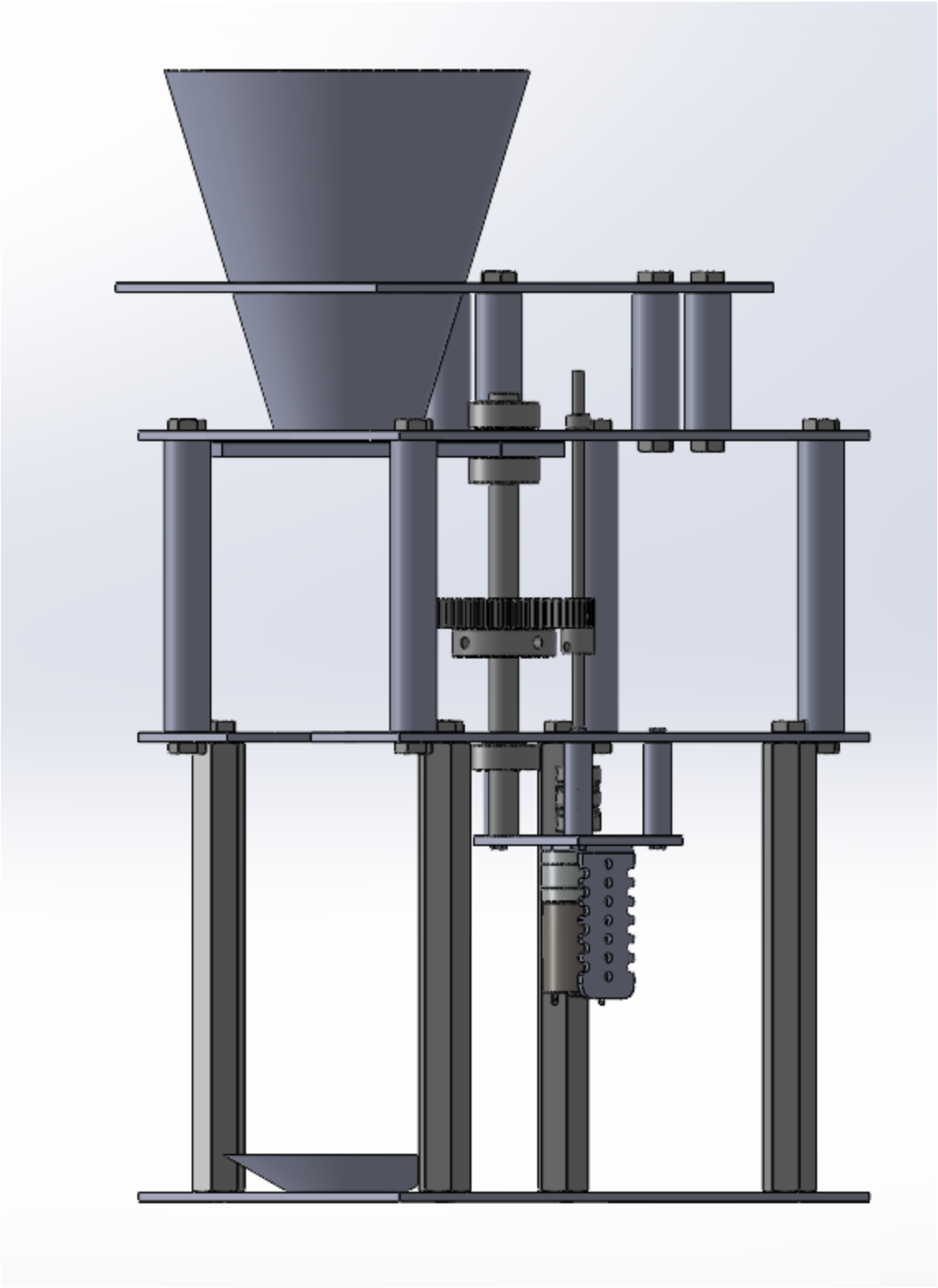
Profile View of Our Candy Dispenser



Front View of Candy Dispenser



Top View



Isometric View of Candy Dispenser

Appendix D: Code

```
1
2
3  #include <ESP32Encoder.h>
4
5  #define BTN_1 12
6  #define BTN_2 13
7
8  #define BIN_1 26
9  #define BIN_2 25
10 #define LED_blue 21
11 //#define LED_red 13
12 #define force  39
13 #define leftSwitch  15
14 #define rightSwitch  32
15
16 ESP32Encoder encoder;
17
18
19 int state = 0;
20
21 int holdOpenDelay = 500;
22 int holdOpenStartTime = 0;
23
24 int lastlimitside = 1;
25
26 volatile bool buttonIsPressed1 = false;
27 volatile bool buttonIsPressed2 = false;
28 volatile int LswitchIsPressed = false;
29 volatile int RswitchIsPressed = false;
30
31
32
33 // setting PWM properties ----------------------------
34 const int freq = 5000;
35 const int ledChannel_1 = 1;
```

```
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int resolution = 8;
const int MAX_PWM_VOLTAGE = 255;


//Initialization -------------------------------------


void IRAM_ATTR isrLeftSwitch() {  // the function to be called when interrupt is triggered
  LswitchIsPressed = true;
}

void IRAM_ATTR isrRightSwitch() {  // the function to be called when interrupt is triggered
  RswitchIsPressed = true;
}

void IRAM_ATTR isr1() {  // the function to be called when interrupt is triggered
  buttonIsPressed1 = true;
}
void IRAM_ATTR isr2() {  // the function to be called when interrupt is triggered
  buttonIsPressed2 = true;
}
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);

  pinMode(BTN_1, INPUT); // configures the specified pin to behave either as an input or an output
  pinMode(BTN_2, INPUT); // configures the specified pin to behave either as an input or an output
  pinMode(leftSwitch, INPUT);
  pinMode(rightSwitch, INPUT);
  pinMode(force, INPUT);
  pinMode(LED_blue, OUTPUT);
  digitalWrite(LED_blue, LOW);

```

```
69
70    ESP32Encoder::useInternalWeakPullResistors = UP; // Enable the weak pull up resistors
71    encoder.attachHalfQuad(33, 27); // Attache pins for use as encoder pins
72    encoder.setCount(0);  // set starting count value after attaching
73
74
75
76    attachInterrupt(leftSwitch, isrLeftSwitch, RISING);
77    attachInterrupt(rightSwitch, isrRightSwitch, RISING);
78    attachInterrupt(BTN_1, isr1, RISING);  // set the "BTN" pin as the interrupt pin; call fu
79    attachInterrupt(BTN_2, isr2, RISING);  // set the "BTN" pin as the interrupt pin; call fu
80
81    // configure LED PWM functionalitites
82    ledcSetup(ledChannel_1, freq, resolution);
83    ledcSetup(ledChannel_2, freq, resolution);
84
85    // attach the channel to the GPIO to be controlled
86    ledcAttachPin(BIN_1, ledChannel_1);
87    ledcAttachPin(BIN_2, ledChannel_2);
88  }
89
90
91
92  void loop() {
93    Serial.println(state);
94
95    switch (state) {
96
97      case 0:
98        stopMotorResponse();
99        Serial.println("Press BTN 1 to start");
.00
.01       if  (b_1()) {
.02         state = 1;
```

```
103       }
104       break;
105
106
107     case 1:
108       stopMotorResponse();
109
110       //Check to see if we should move back to state 0
111       if (b_1()) {
112         state = 0;
113       }
114
115       if (analogRead(force) <= 1) {
116         Serial.println("Funnel is empty, Please refill it then press BTN 2");
117         digitalWrite(LED_blue, LOW);
118       }
119       else {
120         Serial.println("Funnel is Full, press BTN 2 to dispence");
121         digitalWrite(LED_blue, HIGH);
122
123         if (b_2 ()) {
124           state = 2; //dispense state
125           Serial.println("Enjoy ;)");
126         }
127       }
128
129       break;
130
131
132     case 2:
133       Serial.println("Dispensing");
134
135       MotorClockwise();
136
137       if (Rswitch()) {
```

```
        holdOpenStartTime = millis();
        state = 3;//Hold open state
      }

      break;

    case 3:
      stopMotorResponse();
      if (millis() < holdOpenStartTime + holdOpenDelay) {

      } else {
        state = 4;
      }

      break;

    case 4:

      Serial.println("Done Dispensing");

      MotorCounterClockwise();

      if (Lswitch()) {
        stopMotorResponse();
        state = 1; //Idle ready state
      }
      break;

  }

}
```

```
//event checkers
bool b_1 () {
  if (buttonIsPressed1 == true) {
    buttonIsPressed1 == false;
    Serial.println("button 1 pressed");
    return true;
  }
  else {
    return false;
  }
}

bool b_2 () {
  if (buttonIsPressed2 == true) {
    buttonIsPressed2 == false;
    Serial.println("button 2 pressed");
    return true;
  }
  else {
    return false;
  }
}


bool Rswitch() {
  if (RswitchIsPressed == true) {
    RswitchIsPressed = false;
    return true;
  }
  else {
    return false;
  }
}

```

```
bool Lswitch() {
  if (LswitchIsPressed == true) {
    LswitchIsPressed = false;
    return true;
  }
  else {
    return false;
  }
}



// Service Responses
void MotorClockwise() {
  ledcWrite(ledChannel_2, LOW);
  ledcWrite(ledChannel_1, MAX_PWM_VOLTAGE);
}

void MotorCounterClockwise() {
  ledcWrite(ledChannel_1, LOW);
  ledcWrite(ledChannel_2, MAX_PWM_VOLTAGE);
}
void stopMotorResponse() {
  ledcWrite(ledChannel_2, LOW);
  ledcWrite(ledChannel_1, LOW);
  digitalWrite(LED_blue, LOW);
}
```