MECHATRONICS DESIGN

Claudia Renero
Ambrose Liu
Angelo Braganza

Autonomous Trash Manual

8th December, 2022

OVERVIEW

Opportunity

Trash is one of the biggest problems in present times, with the United States being responsible for 12% of the trash produced in the world. We saw the opportunity to help solve this problem through trash storing.

High-level Strategy

We address the problem by wanting to reach a greater audience implementing something easy to use. The logic in our design was that first we wanted to place an object into a small basket, then sensing began, next sorting into the right basket and finally returning to the original state. Initially we wanted to only rely on gravity to sort the trash, but we were not able to come up with an efficient way to do it. We also wanted to have 80% sorting accuracy and minimal failure/jamming of motorized components, which ended up being our biggest problem.

Function-Critical Decisions & Calculations

We had to come up with solutions to how we were going to attach the motor to the housing.

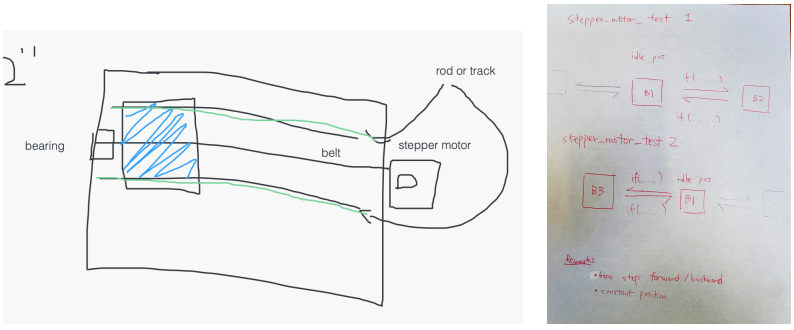


Fig. 1 [Left] Sketch Design. [Right] Picture of motor calculations

INTEGRATED DESIGN

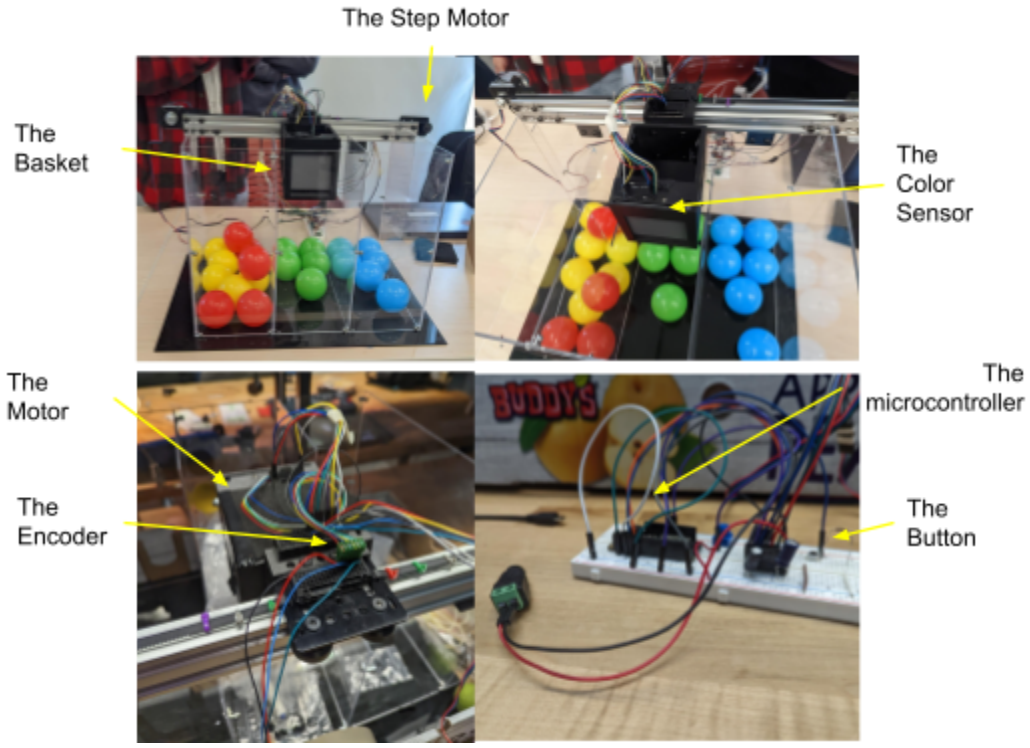


Fig. 2 Part and names of our project

DIAGRAMS

Circuit Diagram

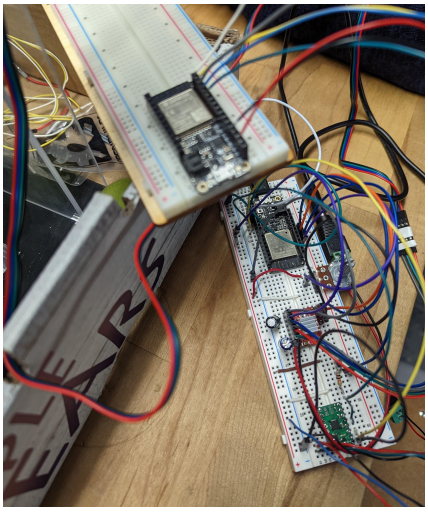


Fig. 3 Circuit diagram

Diagram

We addres

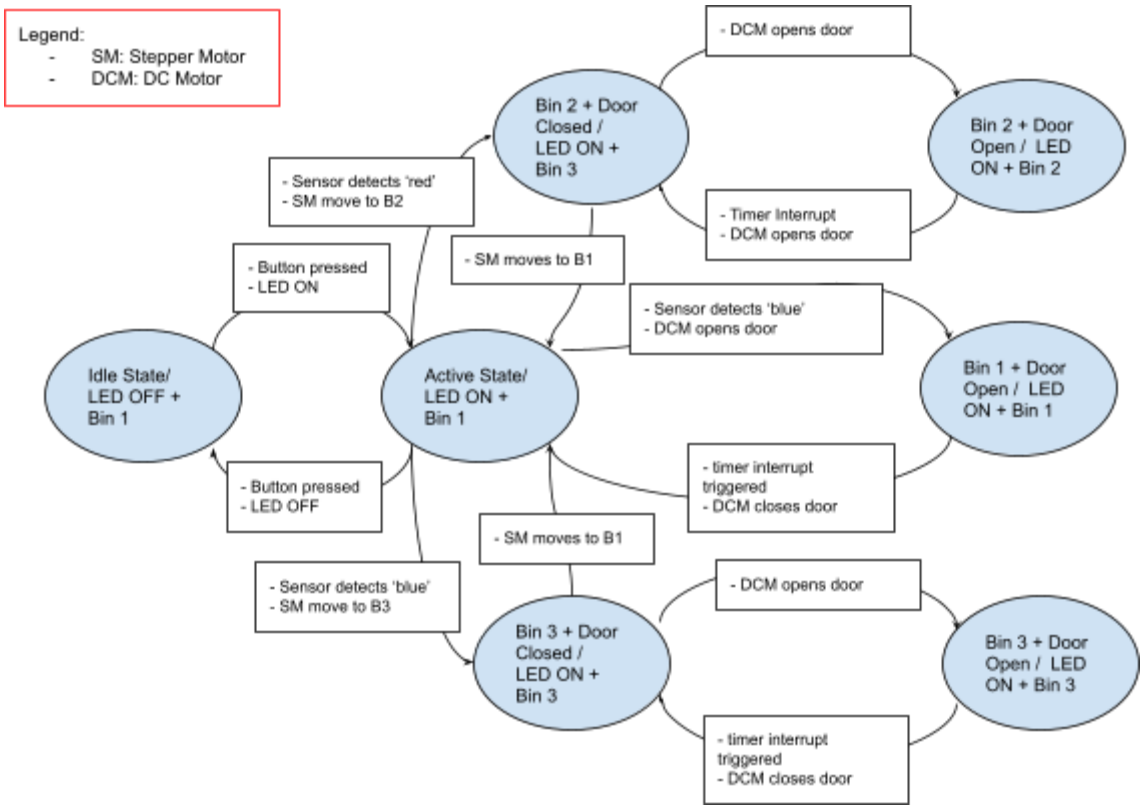


Fig. 4 Circuit diagram

FUTURE STUDENTS

Start early! & Simple works best!

Once you simplify the problem, it's easy for everyone to be on the page, remembering everyone's strengths and weaknesses since the beginning of the project! It will be a stressful process, but the staff are your allies! Use your resources to maximize efficiency.

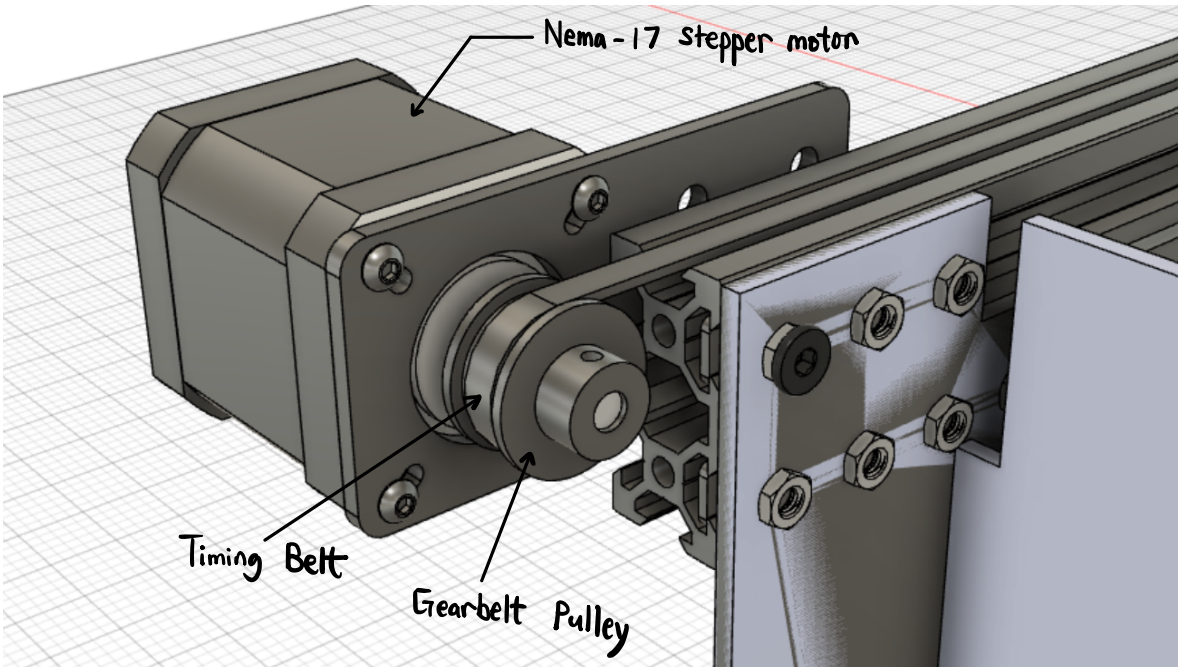
Time is key!

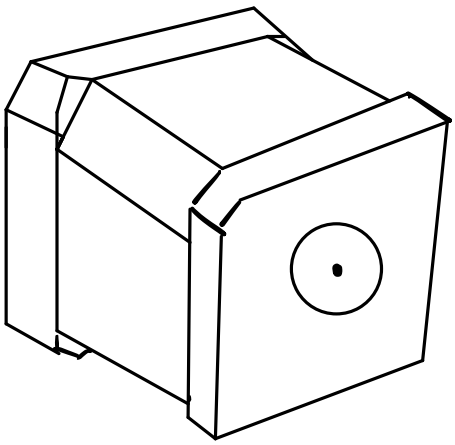
What worked well was doing a simple enough project rather than what we originally had in mind. There were a lot of different ideas for the implementations that ended up negatively affecting the outcome of our project.

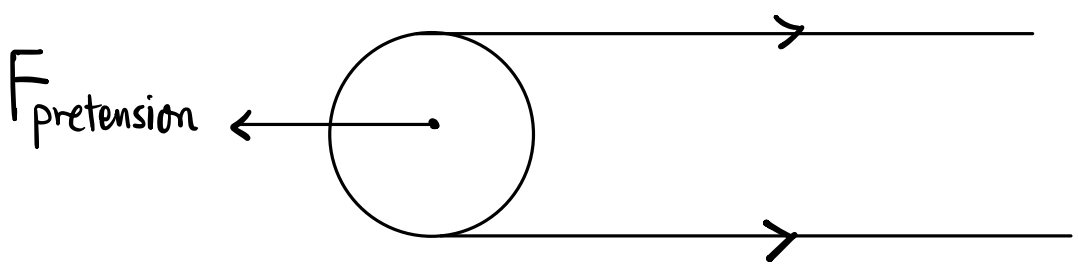
REFERENCES

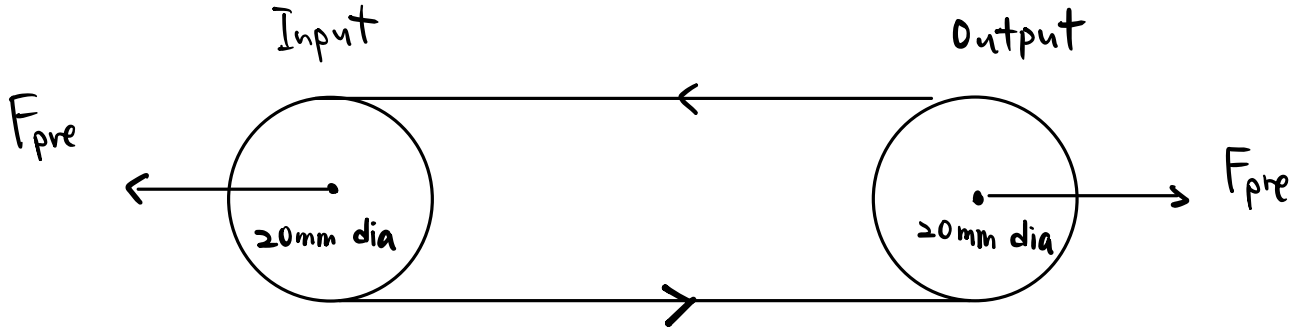
PFORZHEIMER, ADRIAN, and ALEXANDER TRUELOVE. “Trash in America: Moving from Destructive Consumption towards a Zero ...” Trash in America: Moving from Destructive Consumption towards a Zero-Waste System, https://publicinterestnetwork.org/wp-content/uploads/2021/09/US_Trash-in-America-2021-SCRN.pdf

Part #	Part Name	Supplier	Description	Quantity (by Pack)	Price Per Unit (no taxes)	Total Price	Link	Notes	Who's purchasing	Purchased?	Delivered ?	Invoiced	Returned
1334	RGB Color Sensor with IR filter and White LED - TCS34725	Adafruit	Sensor	1	7.95	7.95	Adafruit		Angelo (Purchased)	Y	Y		
324	Stepper motor - NEMA-17 size - 200 steps/rev, 12V 350mA	Adafruit	Motor	1	14		Adafruit	Already Have One	Ambrose (Already Have)	Y	Y	--	N
2215	75:1 Micro Metal Gearmotor HP 6V with Extended Motor Shaft	Pololu	Motor	1	20.95		Pololu	Same as Kit	Ambrose (Already Have)	Y	Y	--	N
1070-Bundle	V-Slot NEMA 17 Linear Actuator Bundle (Belt Driven)	Open Builds Part Store	Transmission System	1	93.99	93.99	Open Builds		Angelo	Y	Y		
	Glaciart One Felt Balls, Felt Pom Poms (5 Pieces) 4 Centimeters - 1.6 Inch	Amazon	Color balls (trash symbol)	1	9.73	9.73	Amazon	might switch	Claudia	Y	Y		
536-Pack	Tee Nuts - M5 (10 Pack)	Open Builds Part Store	Fastener	2	2.99		Open Builds		Ambrose	Y	Y		
946-pack	Low Profile Screws M5 (10 Pack) - 8mm	Open Builds Part Store	Fastener	2	0.99		Open Builds	Everything come with the package is okay for the rail assembly, universal 8mm	Ambrose	Y	Y		
856-pack	Button Head Screws M3 (10 Pack)	Open Builds Part Store	Fastener	1	0.99	15.89	Open Builds			Y	Y	Y	
2190-pack	Low Profile Screws M5 (10 Pack) - 12mm	Open Builds Part Store	Fastener	3	1.09	3.27	Open Builds	for other part exclude the rail assembly	Ambrose	Y	N	--	
36-621-ND	MOUNT BRACKET SHORT L 4-40 STEEL	Digi-Key	Fastener	25	0.3704		Digi-Key	total: $26.80 for both items including tax + shipping	Ambrose	Y	Y		
36-9900-ND	MACHINE SCREW PAN PHILLIPS 4-40	Digi-Key	Fastener	25	0.1272	26.8	Digi-Key		Ambrose	Y	Y	Y	
	3D-printed part - Basket	Craftcloud by All3DP	Hardware	1	47.22		Craftcloud	Material — Carbon Fiber PETG	Ambrose	Y	Y		
	3D-printed part - Custom br	Craftcloud by All3DP	Hardware	1	9.26	60.64	Craftcloud	Material — Carbon Fiber PETG	Ambrose	Y	Y	Y	
	3D-printed part - customerized reinforced br	Craftcloud by All3DP	Hardware	3	4.08		Craftcloud	Material — PLA	Ambrose	Y	Y		
	3D-printed part - Gate	Craftcloud by All3DP	Hardware	1	3.78	17.97	Craftcloud	Material — PLA	Ambrose	Y	Y	Y	
	Laser cutting part - 6 piece of walls	CITRIS INVENTION LAB	Hardware	1		$80	N.A	6x Clear Acrylic 12" x 24"	Ambroes	Y	Y	Y	
					Total:	316.24	**Might not include the shipping cost yet !!!**						


Nema-17 stepper motor
Timing Belt
Gearbelt Pulley




F pretension



To prevent slackness

$\tau = 2.2\ N \cdot cm \qquad r = 0.1\ m$

$$\Sigma M = \tau + r(T_i - \Delta T) - r(T_i + \Delta T)$$

$$\Sigma M = \tau - 2r\Delta T$$

$$0 = (2.2) - 2(0.1)\,\Delta T$$

$$\Delta T = 11\ N$$

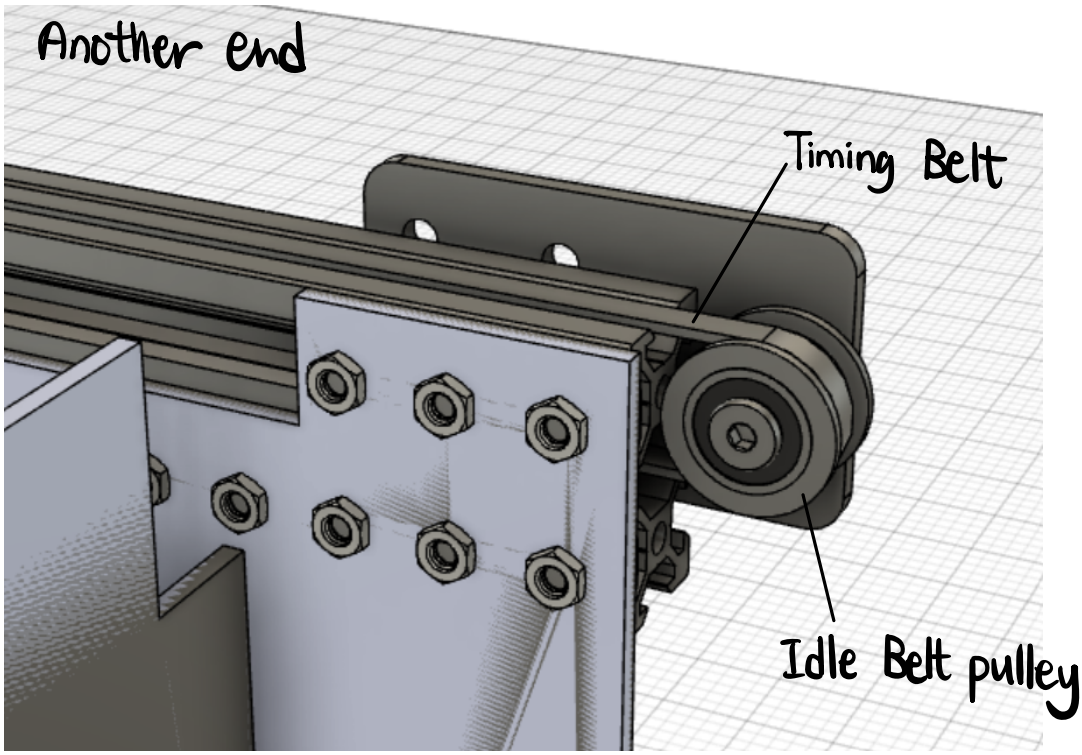
$$2T_i = F_{pretension}$$

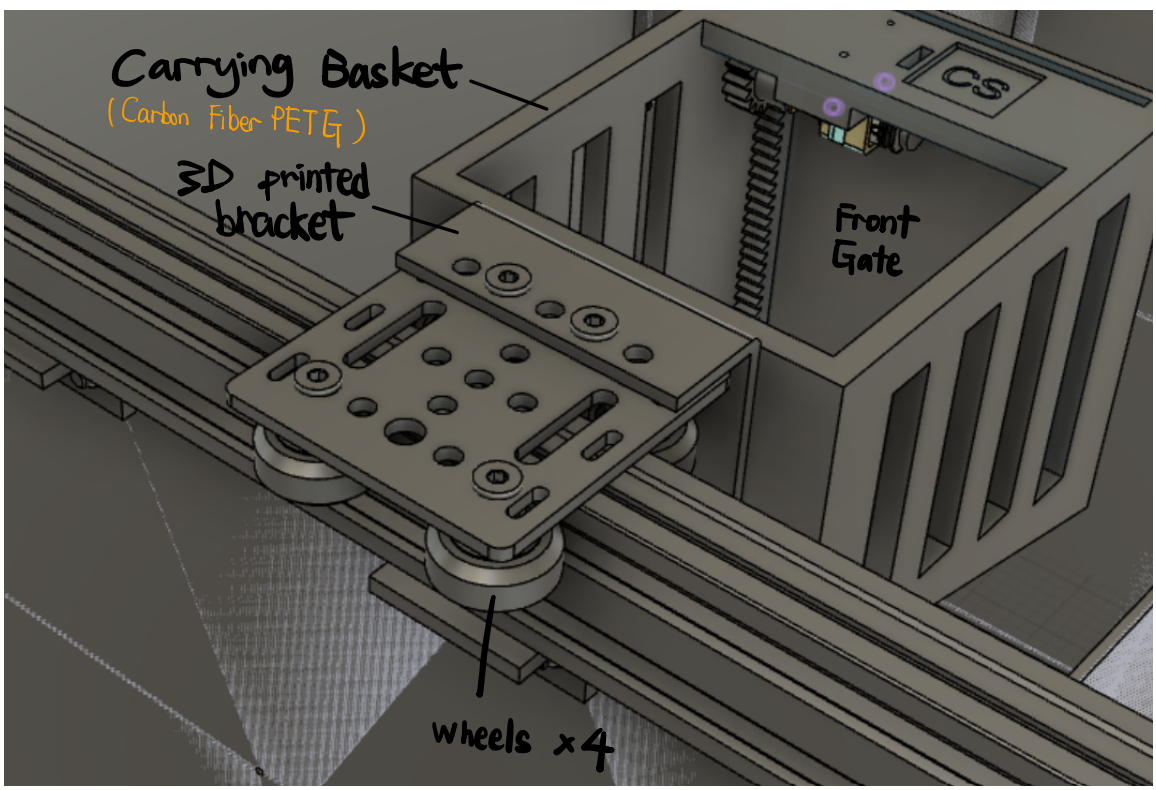
$T_i - \Delta T \geq 0$ to prevent slackness

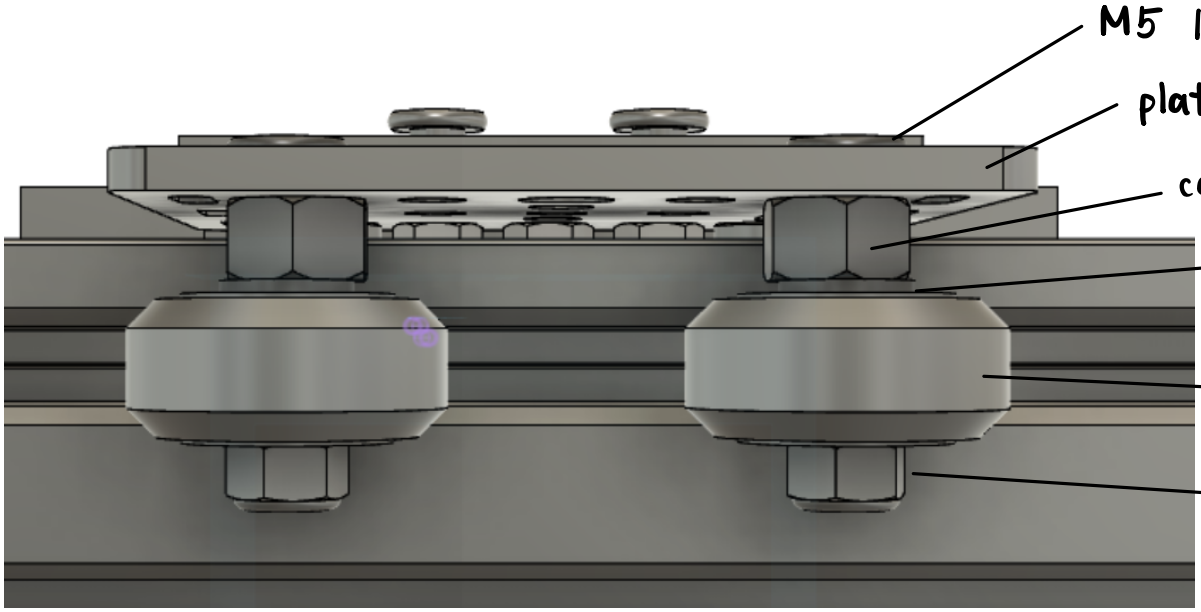
$$T_i \geq \Delta T$$

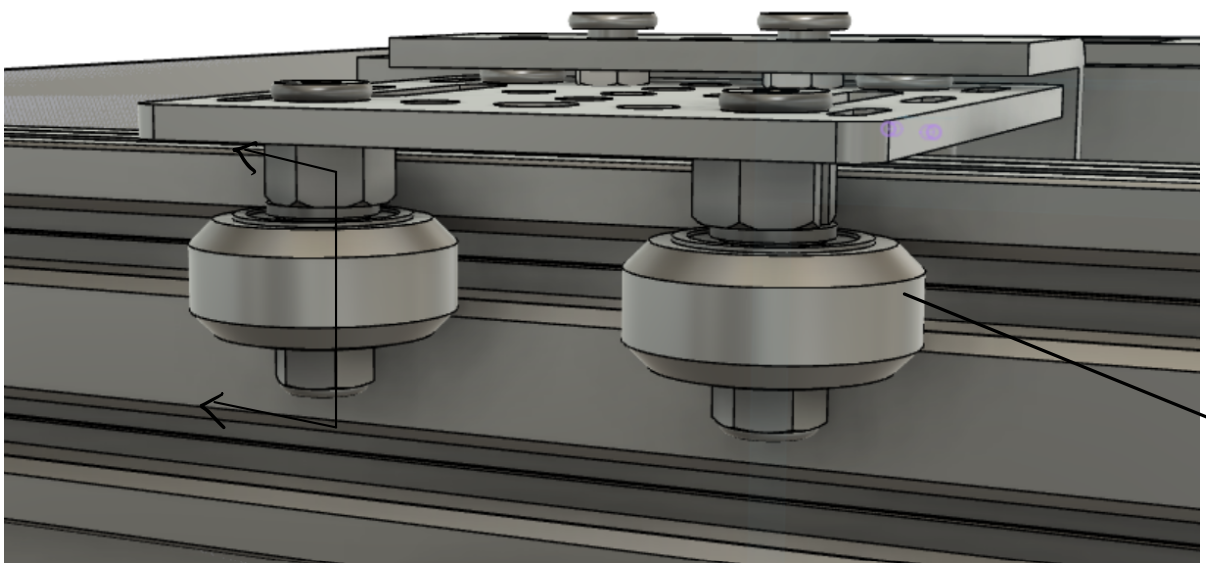
$$2T_i = F_{pretension} \geq 22\ N$$

$$\boxed{F_{pretension} \geq 22\ N}$$

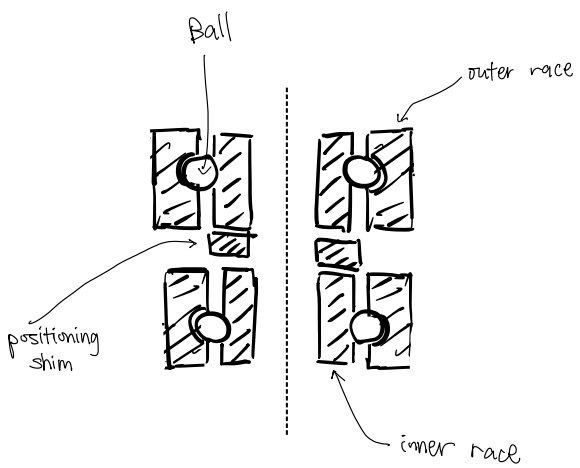

Another end
Timing Belt
Idle Belt pulley

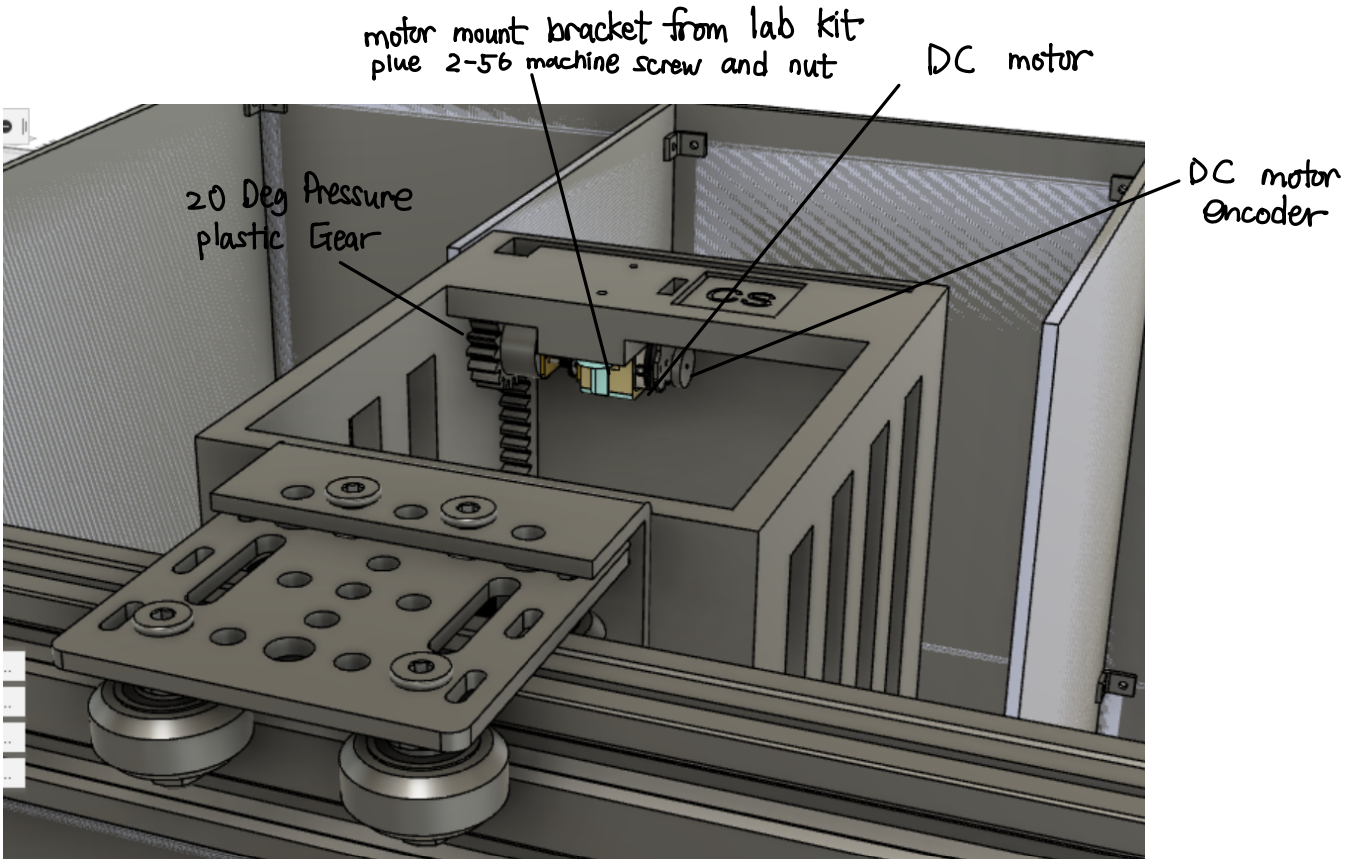

Carrying Basket
(Carbon Fiber PETG)
3D printed bracket
Front Gate
Wheels x4

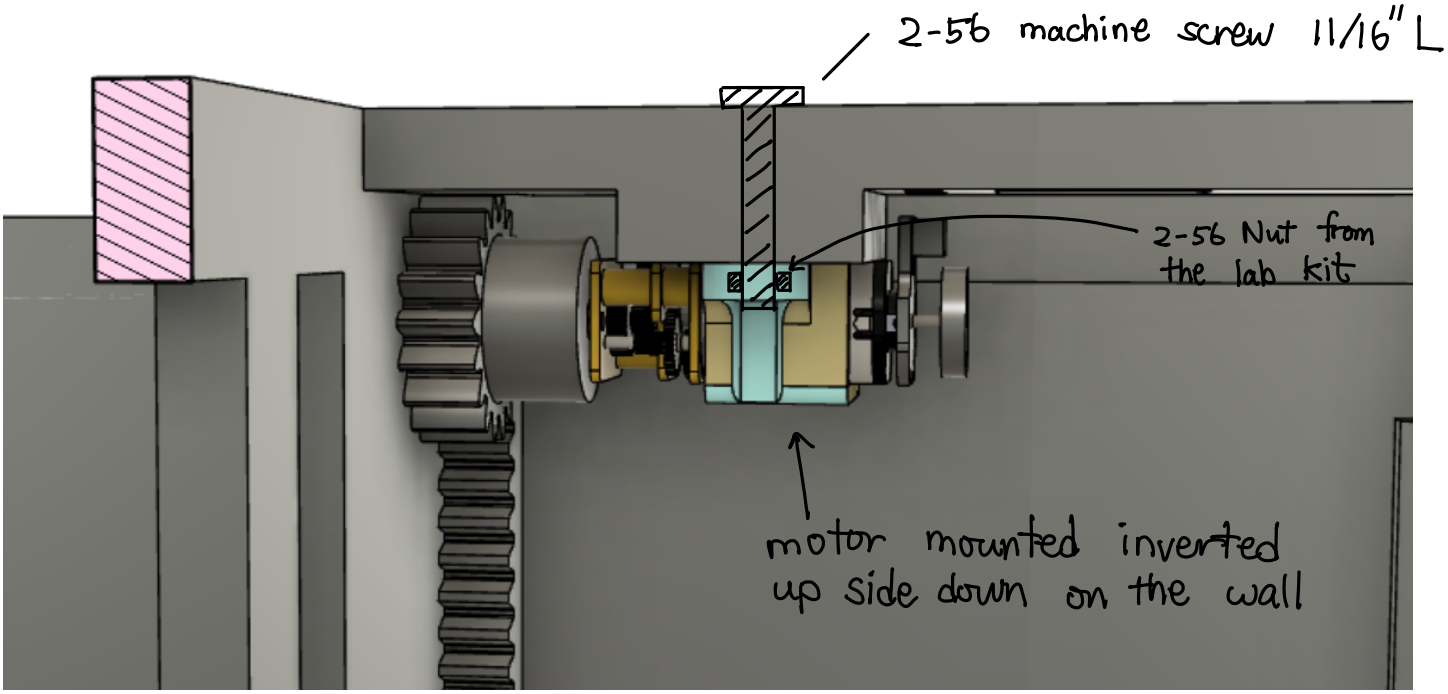




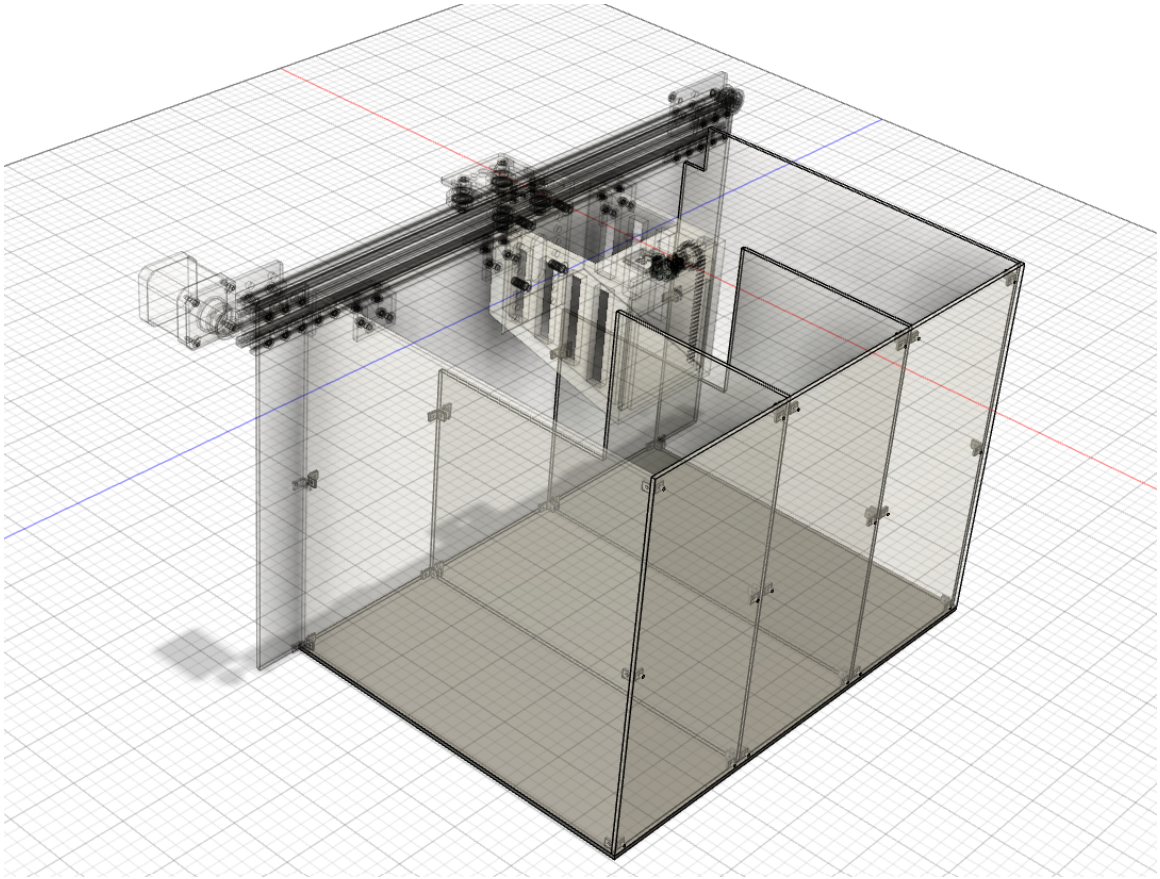
Cross-section view of wheel

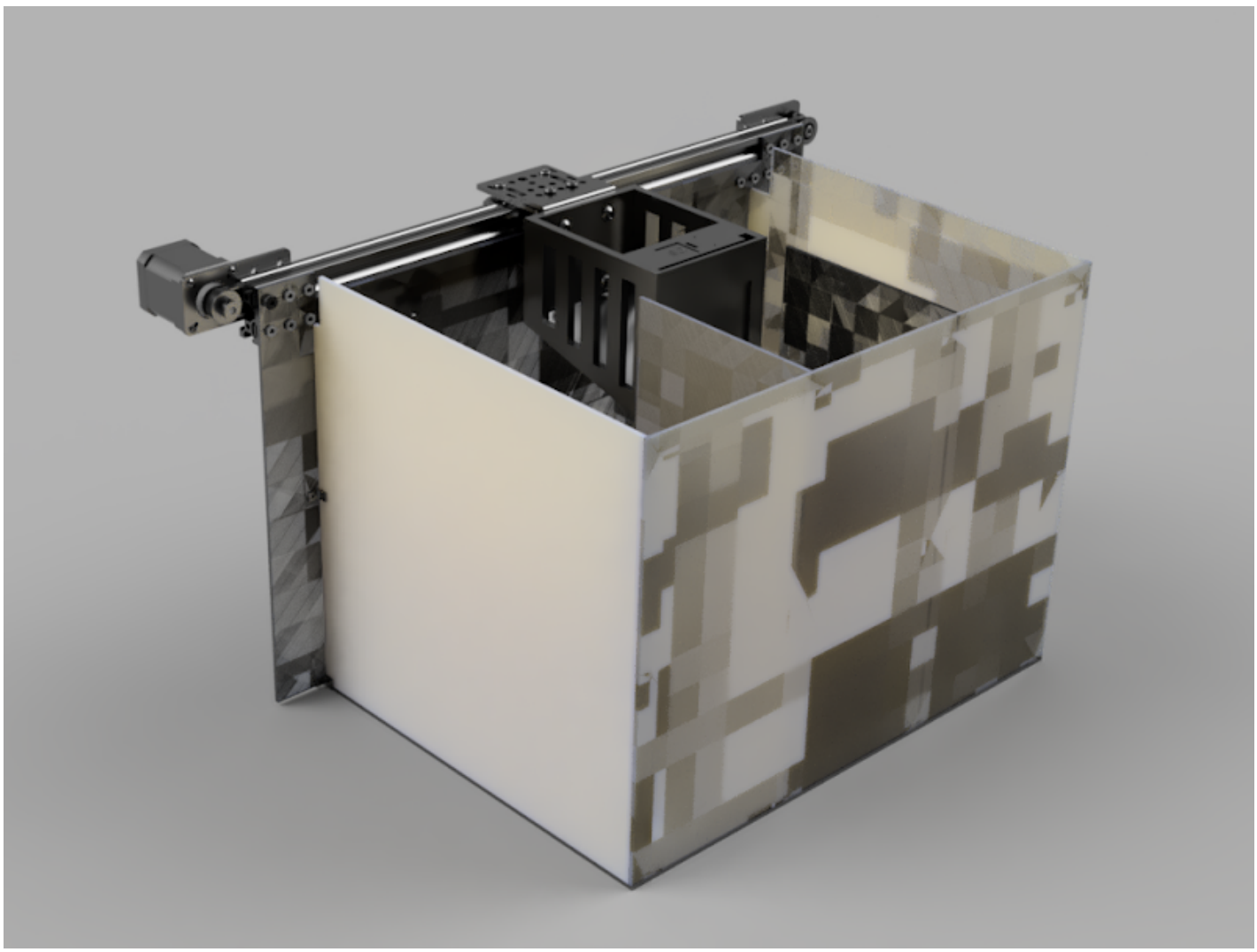



motor mount bracket from lab kit
plue 2-56 machine screw and nut
DC motor
DC motor
encoder
20 Deg Pressure
plastic Gear

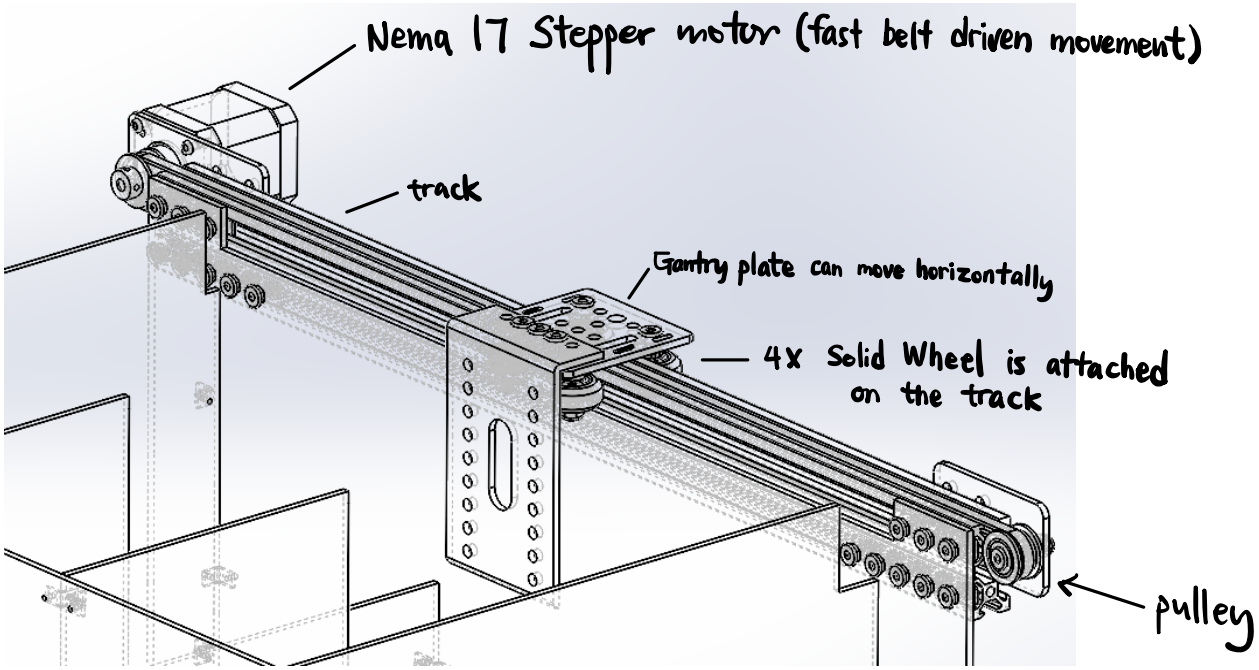

2-56 machine screw 11/16" L
2-56 Nut from
the lab kit
motor mounted inverted
up side down on the wall

Overall View

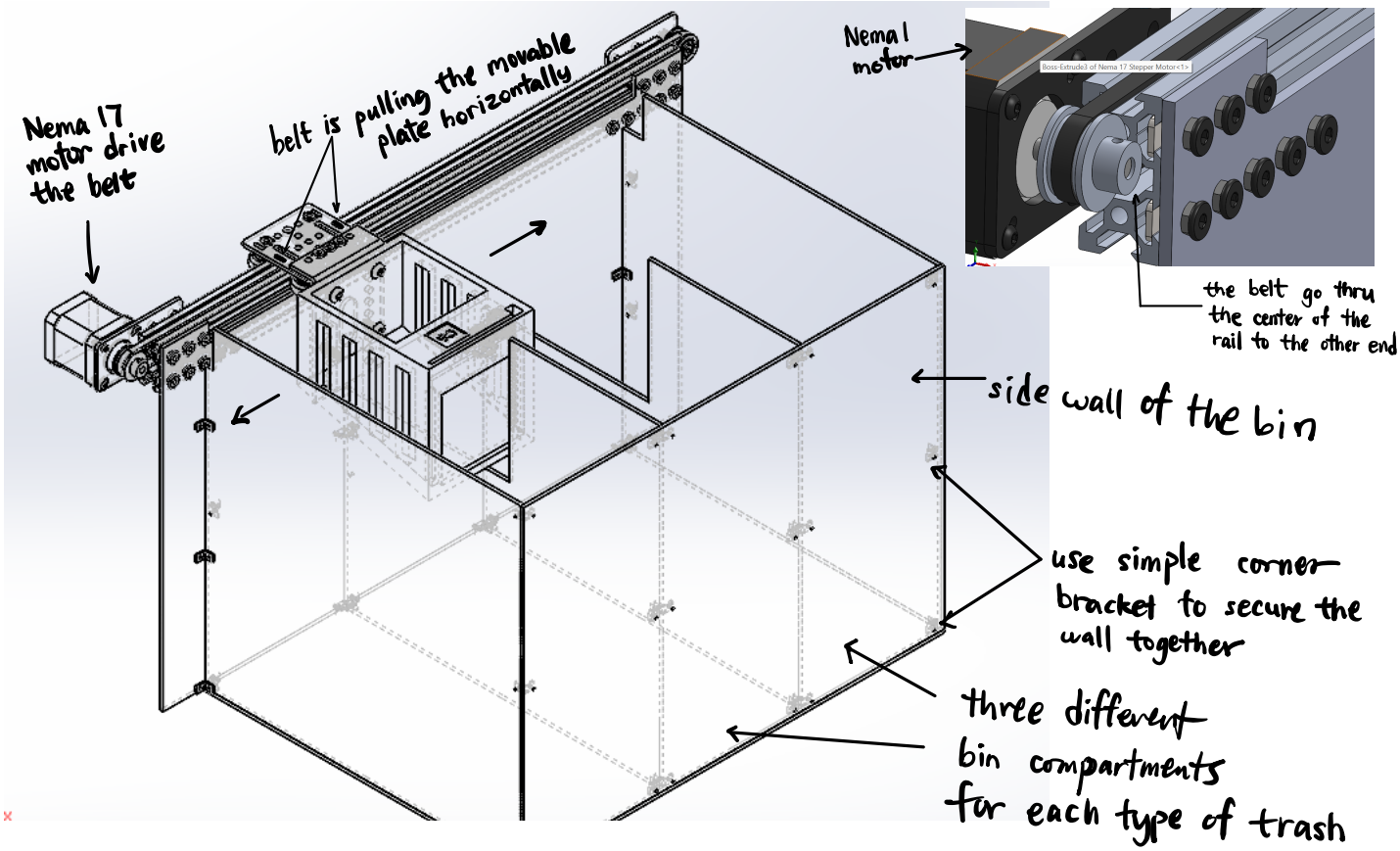


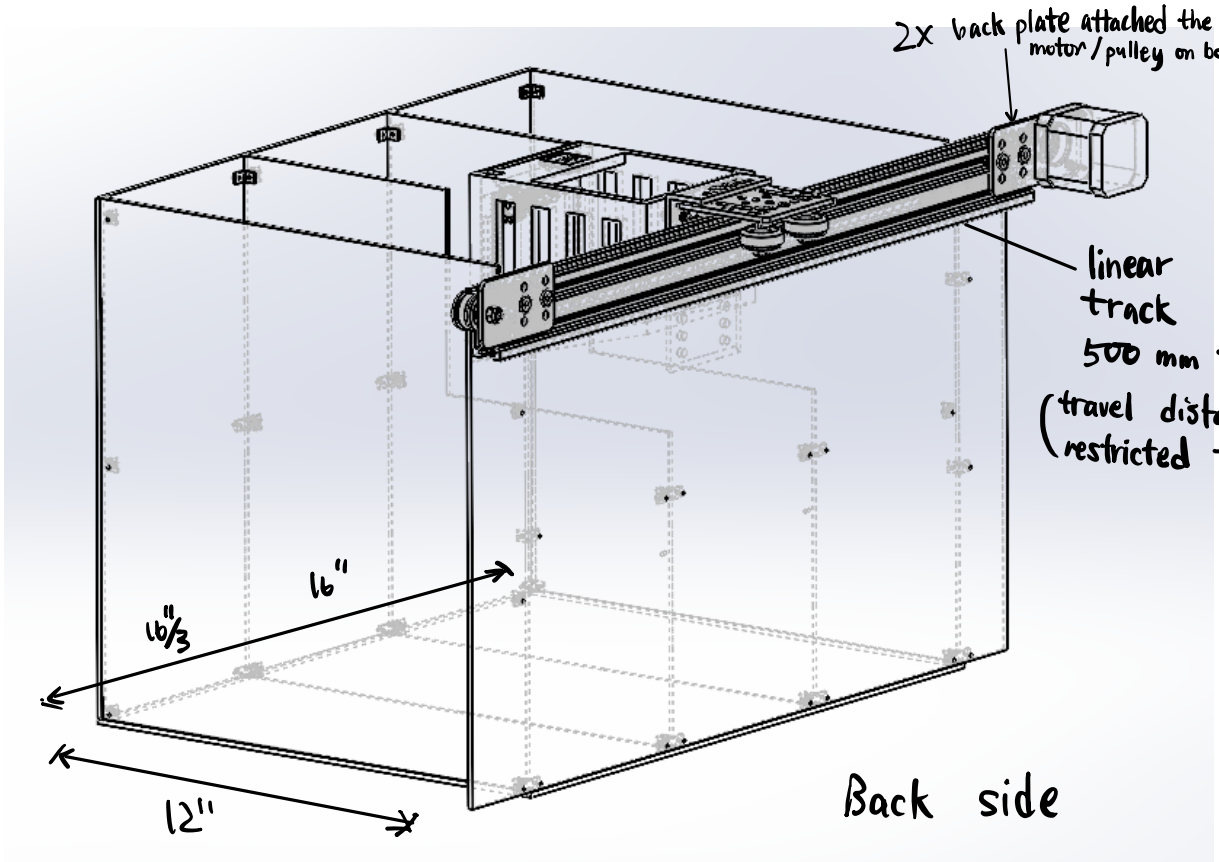


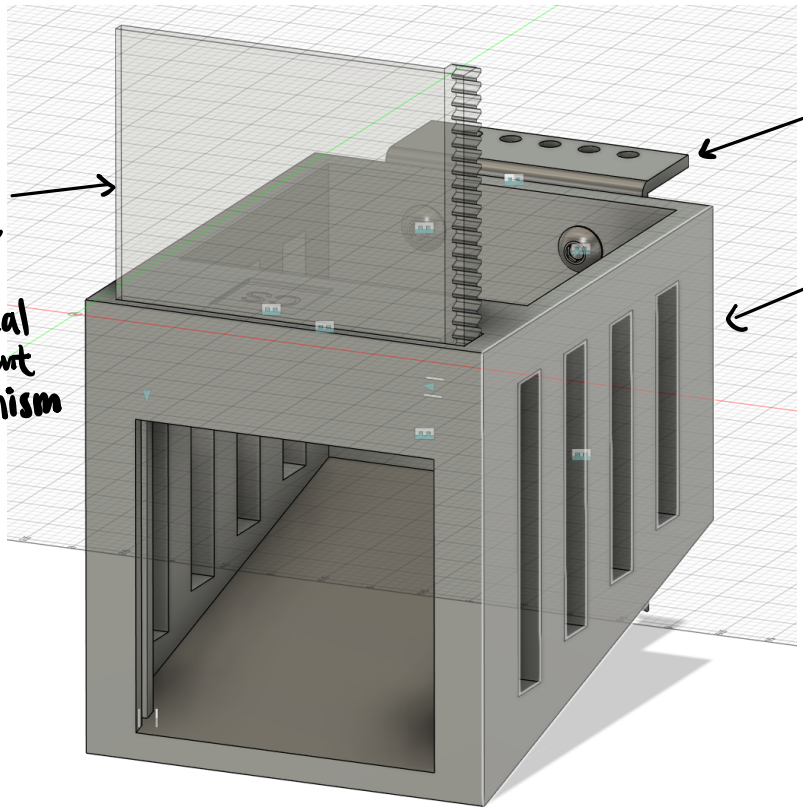
Design model



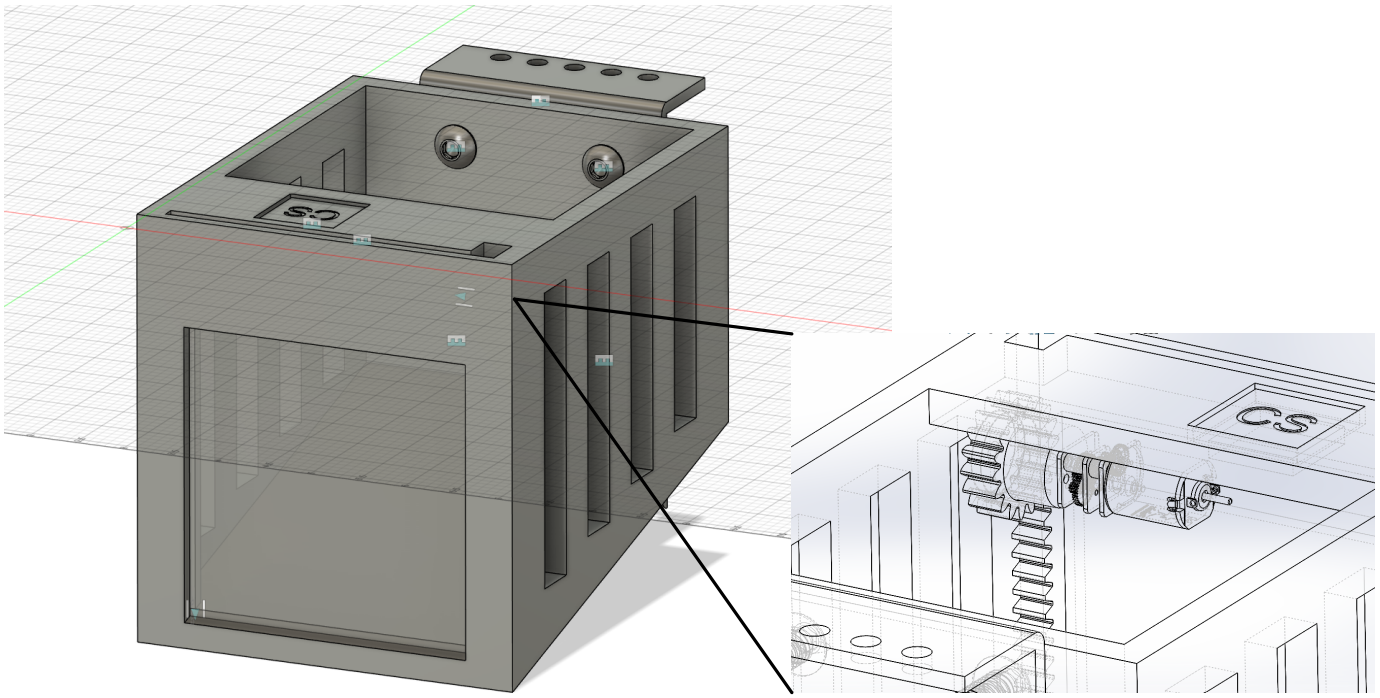
In our design, we use V-slot belt drive linear actuator to provide a linear motion for the bracket mounted with our carriage basket within a travel distance of 14” horizontally



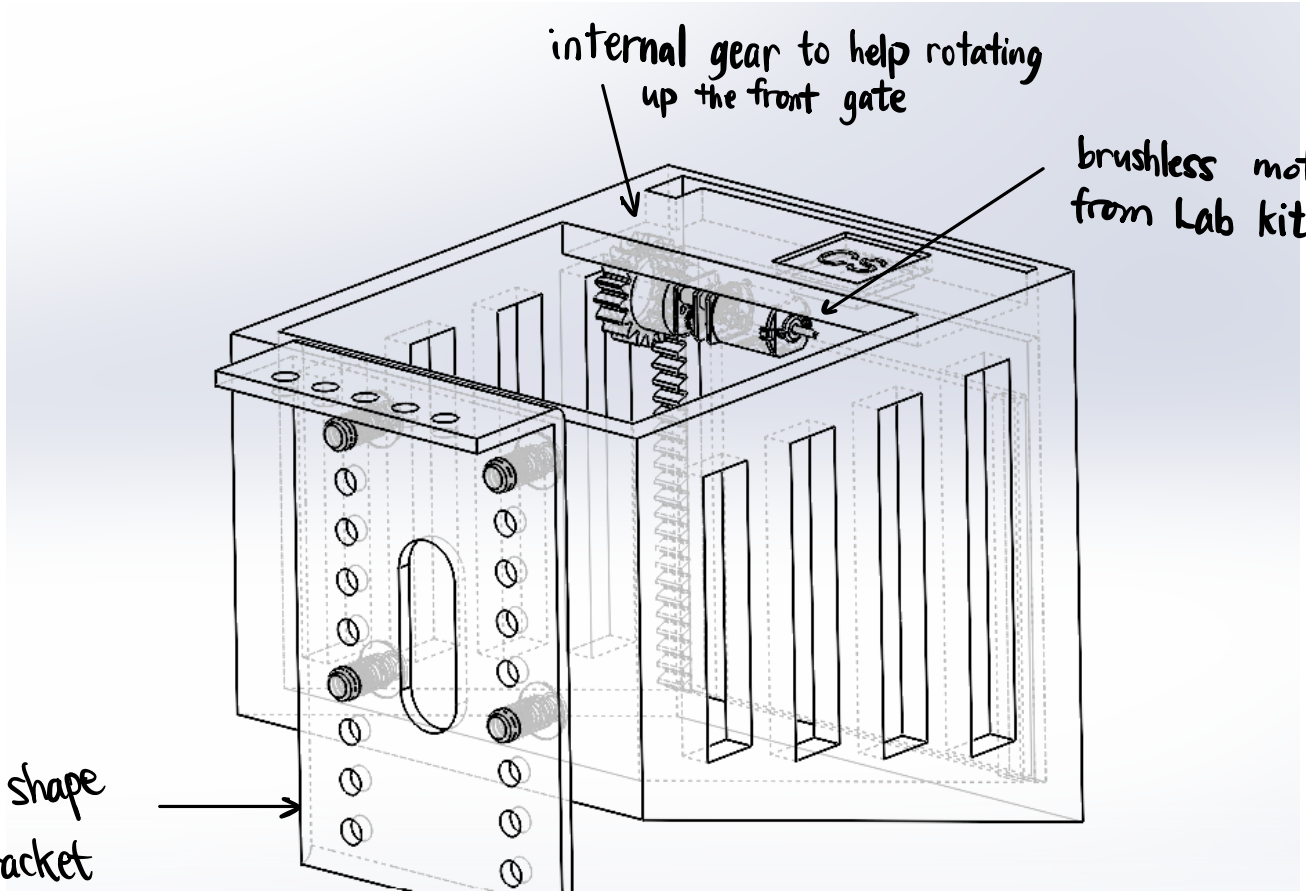




status: open → trash can fall down to each compartement at different location



status : closed → trash inside



```
#include <Arduino.h>
#include "A4988.h"              // Copyright (C)2015-2018
Laurentiu Badea (MIT)

#define Motor_steps 200         // Used a 200-microstep
motor (Nema-17)--> 200 step/rev
#define DIR 27                   // Declared the motor
driver directional pin (ORIGINLLY PIN 26)
#define STEP 33                 // Declared the motor
driver step pin (ORIGINLLY PIN 25)
#define BTN 15                  // Declared the button ED
pin number
#define LED_PIN 13              // Declare the builtin
LED pin number
#define POT 32                  //

#define BIN_1 26               // DC motor pin
#define BIN_2 25               //  DC motor pin

//pin for the color sensor
#define S0 21
#define S1 17
#define S2 18
#define S3 19
#define sensorOut 16

//Setup the stepper motor
 A4988 stepper(Motor_steps, DIR, STEP);

 //Setup variables
-------------------------------------------------------
```

```
-
  int step_count ;
  int des_step ;
  volatile bool buttonIsPressed = false ;
  volatile bool buttonstatehaschanged = false ;
  int state = 1 ;
  int buttonState ;
  int value ;
  int threshold = 1250;
  int microstep ;
  int back_Step ;
  int inner_Step = 0;
  int des_pos;
  int x = 0 ;
  int dc_motor_dist = 60;
  int color_number ;

// setting PWM properties ----------------------------
 const int freq = 5000;
 const int ledChannel_1 = 1;
 const int ledChannel_2 = 2;
 const int resolution = 8;
 int MAX_PWM_VOLTAGE = 140;

//Setting up the color sensor variable
----------------------
 int red_f = 0;
 int blue_f = 0;
 int green_f = 0;

//Initialization  ----------------------------------
```

```
 void IRAM_ATTR isr() {    // the function to be called
when interrupt is triggered
   buttonIsPressed = true;
   buttonstatehaschanged = true;
  }

//please put your code
here----------------------------------------------------
---------------

 void setup() {

    pinMode(BTN, INPUT);
    pinMode(LED_PIN, OUTPUT);
    pinMode(POT, INPUT);
    attachInterrupt(BTN, isr, CHANGE);
    stepper.begin(30, 1);
    Serial.begin(115200);

    // configure LED PWM functionalitites
    ledcSetup(ledChannel_1, freq, resolution);
    ledcSetup(ledChannel_2, freq, resolution);

    // attach the channel to the GPIO to be controlled
    ledcAttachPin(BIN_1, ledChannel_1);
    ledcAttachPin(BIN_2, ledChannel_2);


    pinMode(S0, OUTPUT);
    pinMode(S1, OUTPUT);
    pinMode(S2, OUTPUT);
```

```
    pinMode(S3, OUTPUT);
    pinMode(sensorOut, INPUT);

   // Setting frequency-scaling to 20%
   digitalWrite(S0, HIGH);
   digitalWrite(S1, HIGH);

  }

 void loop() {
   buttonState = digitalRead(BTN);
   value = analogRead(POT);
   printing();
   colorsensor();

   switch (state){
     case 1:
       if (checkForButtonPress() &&
checkForButtonStatehaschanged() ){
         led_on();
         state = 2;
         //colorsensor();
         }
         break;

    case 2:
       //colorsensor();
       if (checkForButtonPress() &&
checkForButtonStatehaschanged() ) {
         led_off();
         state = 1;
```

```
    }
    else {
        //colorsensor();
        if (color_number == 300){  //blue ball
                  led_on();
                  Movethemotortotheright();
                  state = 3;
                }

        else if (color_number == 200){  //green ball
                  led_on();
                  state = 5;
                }

        else if (color_number == 100){//red ball
                  led_on();
                  Movethemotortotheleft();
                  state = 7;
                }
         else {
                  led_on();
                  state = 2;
                  //do nothing
                  }
    }
      break;

case 3:
 led_on();
 //colorsensor();
 if (color_number == 300){
```

```
      if ( x < dc_motor_dist ){
          ledcWrite(ledChannel_1, LOW);
          ledcWrite(ledChannel_2, MAX_PWM_VOLTAGE);
         x++ ;
      }
      else {
         ledcWrite(ledChannel_1, LOW);
         ledcWrite(ledChannel_2, LOW);
        state = 4;
      }
   }
  else {
      Movethemotortotheleft();
      state = 2;
      }

   break;

   case 4:
    led_on();
    if ( color_number == 0 ){
         if ( x != 0 ){ //turning on the motor to
close the gate
            ledcWrite(ledChannel_2, LOW);
            ledcWrite(ledChannel_1, MAX_PWM_VOLTAGE);
           x-- ;
           }
          else {
           ledcWrite(ledChannel_2, LOW);
           ledcWrite(ledChannel_1, LOW);
           state = 3;
```

```
          }
        }
  else {
      ledcWrite(ledChannel_1, LOW);
      ledcWrite(ledChannel_2, LOW);
     state = 4;
     }

 break;

case 5:
 led_on();
 if ( color_number == 200 ){
    if ( x < dc_motor_dist ){
       ledcWrite(ledChannel_1, LOW);
       ledcWrite(ledChannel_2, MAX_PWM_VOLTAGE);
      x++ ;
      }
     else{
      ledcWrite(ledChannel_1, LOW);
      ledcWrite(ledChannel_2, LOW);
      state = 6;
     }
 }
else {
    state = 2;
    }
 break;

case 6:
  led_on();
```

```
    if ( color_number == 0 ){
        if ( x != 0 ){ //turning on the motor to close
the gate
           ledcWrite(ledChannel_2, LOW);
           ledcWrite(ledChannel_1, MAX_PWM_VOLTAGE);
          x-- ;
          }
        else {
           ledcWrite(ledChannel_2, LOW);
           ledcWrite(ledChannel_1, LOW);
           state = 5;
         }
       }
    else {
        ledcWrite(ledChannel_1, LOW);
        ledcWrite(ledChannel_2, LOW);
       state = 6;
       }
   break;

  case 7:
  led_on();
  if ( color_number == 100 ){
      if ( x < dc_motor_dist ){
         ledcWrite(ledChannel_1, LOW);
         ledcWrite(ledChannel_2, MAX_PWM_VOLTAGE);
        x++ ;
         }
       else {
           ledcWrite(ledChannel_1, LOW);
           ledcWrite(ledChannel_2, LOW);
```

```
            state = 8;
           }
          }
  else {
       Movethemotortotheright();
       state = 2;
       }
    break;


  case 8:
    led_on();
    if ( color_number == 0 ){
          if ( x != 0 ){ //turning on the motor to
close the gate
               ledcWrite(ledChannel_2, LOW);
               ledcWrite(ledChannel_1, MAX_PWM_VOLTAGE);
              x-- ;
              }
         else {
            ledcWrite(ledChannel_2, LOW);
            ledcWrite(ledChannel_1, LOW);
            state = 7;
            }
       }
    else {
         ledcWrite(ledChannel_1, LOW);
         ledcWrite(ledChannel_2, LOW);
        state = 8;
        }
    break;
```

```
  }
}

// Event checker
-----------------------------------------
 bool checkForButtonPress() {
  if (buttonIsPressed == true){
    buttonIsPressed = false ;
    return true;
    }
    else {
    return false;
    }
  }

 bool checkForButtonStatehaschanged() {
  if (buttonstatehaschanged != buttonState){
    return true ;
    }
   else {
    return false ;
    }
 }

// Event service
-----------------------------------------
 void led_on() {
  digitalWrite(LED_PIN,HIGH);
  }
```

```
 void led_off() {
  digitalWrite(LED_PIN,LOW);
 }

  void printing(){
    char statement[300];
    sprintf(statement, " State: %i    Potentionemter:
%i    Desired_position: %i    step_count %i
DC_motor:  %i   Color: %i", state, value,  des_pos,
step_count, x, color_number);
    Serial.println(statement);
  }

void Movethemotortotheright() {
   des_step = 450;
   des_pos = step_count + des_step;
   inner_Step = 0;
   stepper.startMove(des_step);
   while (inner_Step < des_step){
    microstep = stepper.getStepsCompleted();
    stepper.nextAction();
    printing();
    if (microstep = 1){
      inner_Step ++ ;
      step_count += microstep;
      }
     printing();
     }
    stepper.stop();
    des_pos = step_count ;
  }
```

```
void Movethemotortotheleft() {
   des_step = -450;
   des_pos = step_count + des_step;
   inner_Step = 0;
   stepper.startMove(des_step);  //Initiate a move over
known distance
   while (inner_Step > des_step){
       microstep = stepper.getStepsCompleted();
       stepper.nextAction();
       printing();
          if (microstep = 1){
            inner_Step -- ;
            step_count -= microstep;
            }
           }
       stepper.stop();
       des_pos = step_count ;
     }

void colorsensor(){
    // put your main code here, to run repeatedly:
    // put your main code here, to run repeatedly:
    // Setting red filtered photodiodes to be read
    digitalWrite(S2, LOW);
    digitalWrite(S3, LOW);
    // Reading the output frequency
    red_f = pulseIn(sensorOut, LOW);
    // Setting Green filtered photodiodes to be read
    digitalWrite(S2, HIGH);
    digitalWrite(S3, HIGH);
```

```
    // Reading the output frequency
    green_f = pulseIn(sensorOut, LOW);
    // Setting Blue filtered photodiodes to be read
    digitalWrite(S2, LOW);
    digitalWrite(S3, HIGH);
    // Reading the output frequency
    blue_f = pulseIn(sensorOut, LOW);

    if (blue_f > 40 && blue_f < 110 && green_f > 60 &&
green_f < 140 && red_f > 40 && red_f < 130){ //replace
with upper bounds of room lighting
            //ex: ME102B Room: Black - 140, 153, 118;
ME100 Room: Black - 134, 146,. 112
            if (green_f < red_f && green_f < blue_f){
            color_number = 200;
            //delay(100);
            }
            else if (blue_f < red_f && blue_f <
green_f){
            color_number = 300;
            //delay(100);
            }
            else if (red_f < blue_f && red_f < green_f){
            color_number = 100;
            //delay(100);
            }
       }
   else{
       color_number = 0;
       //delay(100);
     }
```

}