Denver Chikokonya

Nadine Iradukunda

Opanin Akuffo

Joe Amarteifio

Final Project: A 3 axis Robotic Arm with a Conveyor Belt

Introduction to Engineering

Group 12

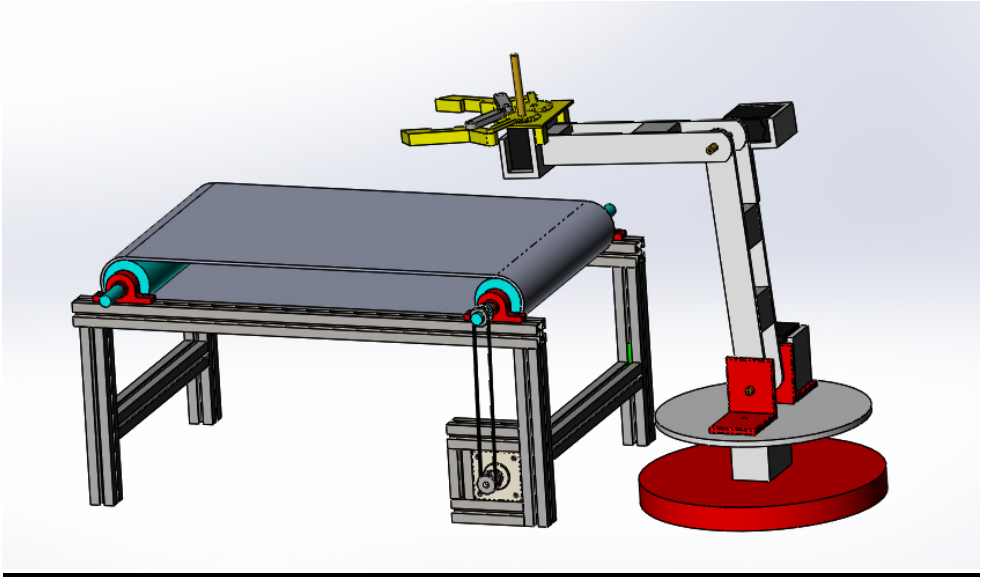
July 23, 2018

Background

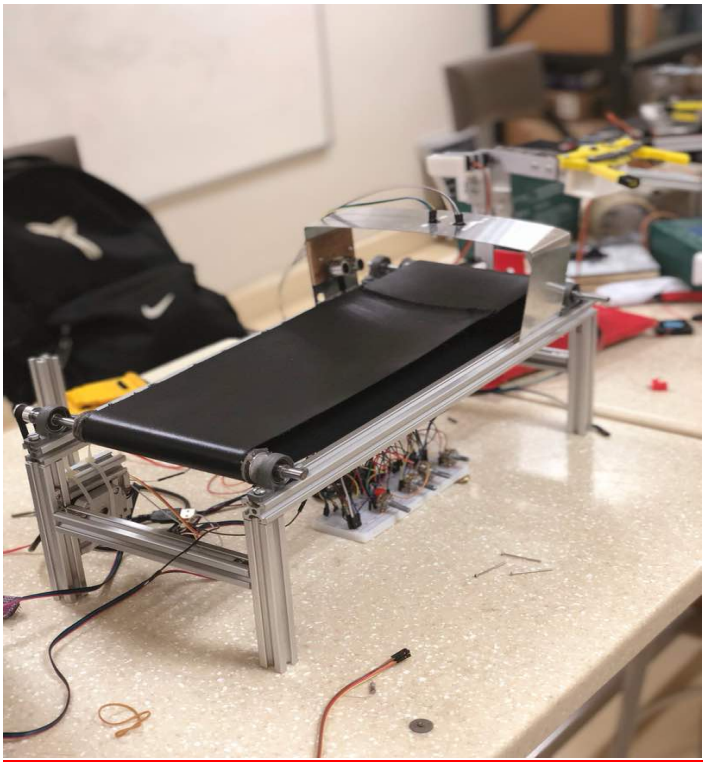
This project is a design, and a construction of a 3-axis robotic arm with a conveyor belt to be used to perform a task of picking objects from a conveyor belt to a certain place, it can also pick objects based on colors, height, weight. To perform this project, we used Solidworks to design CAD parts of the arm and conveyor belt, Arduino programming knowledge in controlling the activities that the robot performs, electronic circuit components such as 4 servo motors, 1 stepper motor and driver, resistors, color sensors, and push buttons.

In the modern world, human driven operation of producing, moving, and picking products from one place to another in industries should be reduced because the process takes time, slows the production channel, comparing to the industries that use automated technological approaches of performing the same tasks because the automated method improves efficiency, reduce the labor, increase the speed of production. Thus, our project can be used in breweries industries when packing drinks in their bottles. For instance, soft drinks like coca cola, sprite, are packed based on the color of bottles, some based on the color of the drink. In this case, the robotic arm pick and place drinks in their respective trays based on the colors.

In addition, robotic arm and conveyor belt can be used in construction of tall buildings, in picking block from the basement to a certain level. In this case, for instance, if the user need painted block the sensor would detect the one which painted the one which is not, and the robot can remove it in the line of the blocks to be used in the construction.

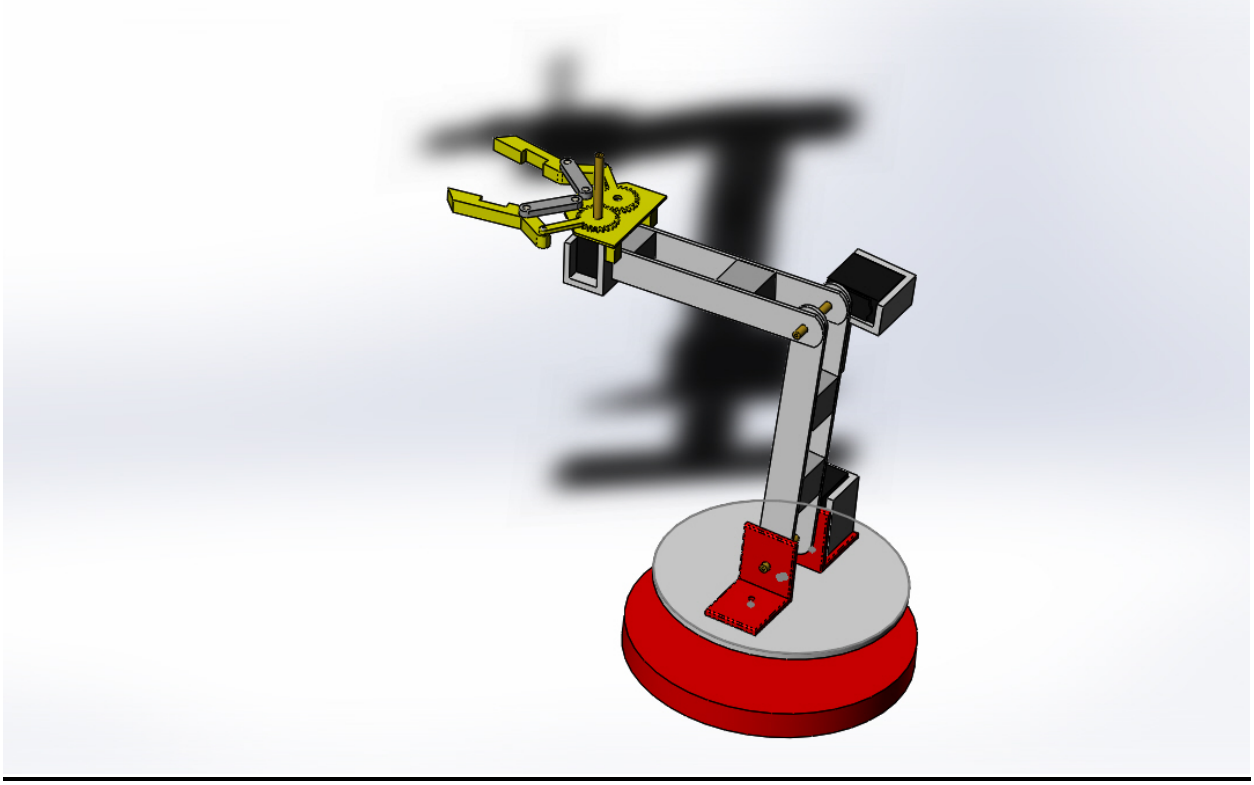


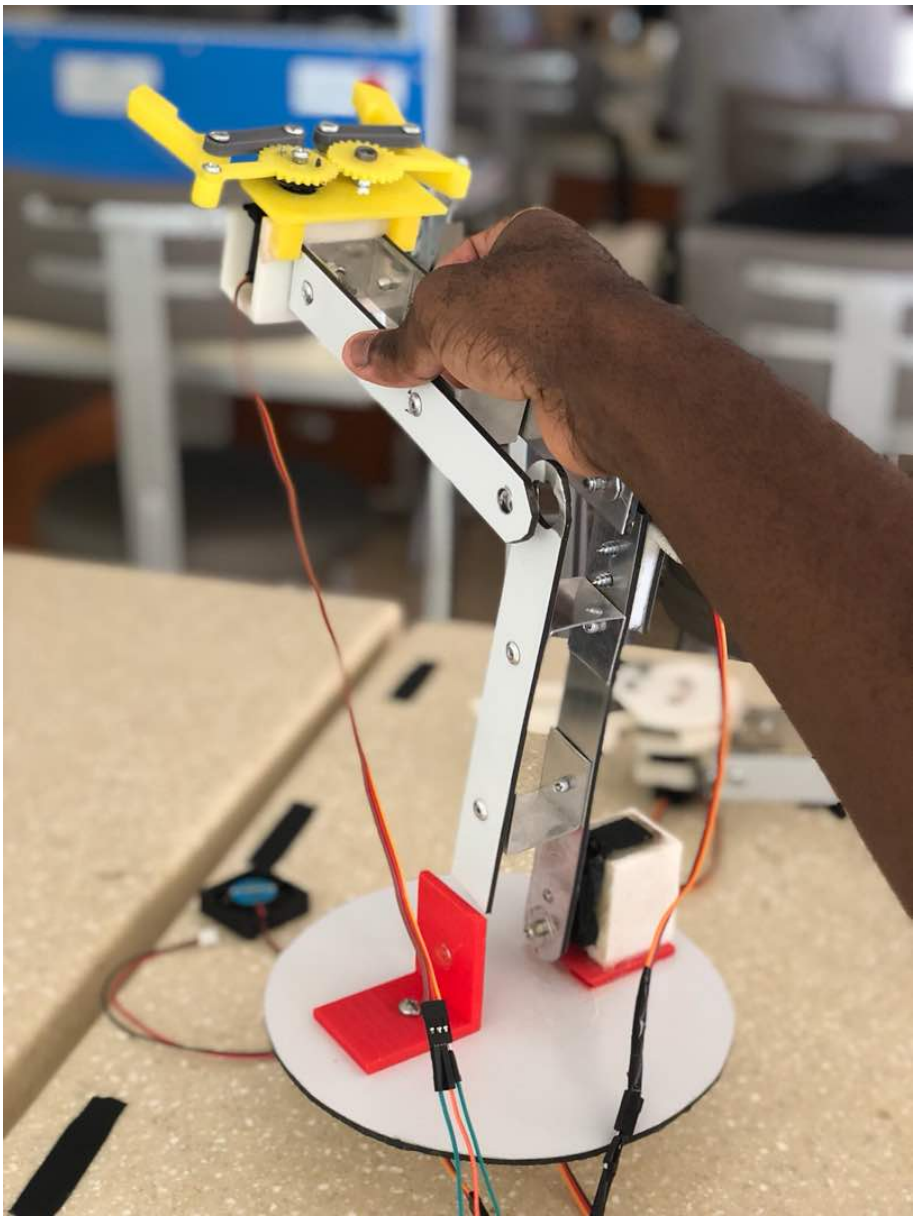
The Conveyor Belt



The conveyor belt was made using, 1 stepper motor which will be controlled by the Arduino code. Timing pulley, shafts, bearings, frame, bolts, nuts and connectors. A push button was used to control the belt this code allows the user to start, stop and reverse the conveyor belt when the object is being rolled on it. The direction of the belt can be controlled manually by pushing the button on the breadboard, once the button the power supply is on the belt starts, and when the button is pushed the belt stops again when it is pushed again it reverses the direction. Automatically, once the Arduino code has been verified and uploaded the stepper motor rotates with the timing pulley and timing belt causing the conveyor belt to move in a certain direction. When the object has been the detected by ultrasonic and color sensor, the belt stops for a while and then the robot gets ready to pick the object depending on the color of it.

The Robotic Arm

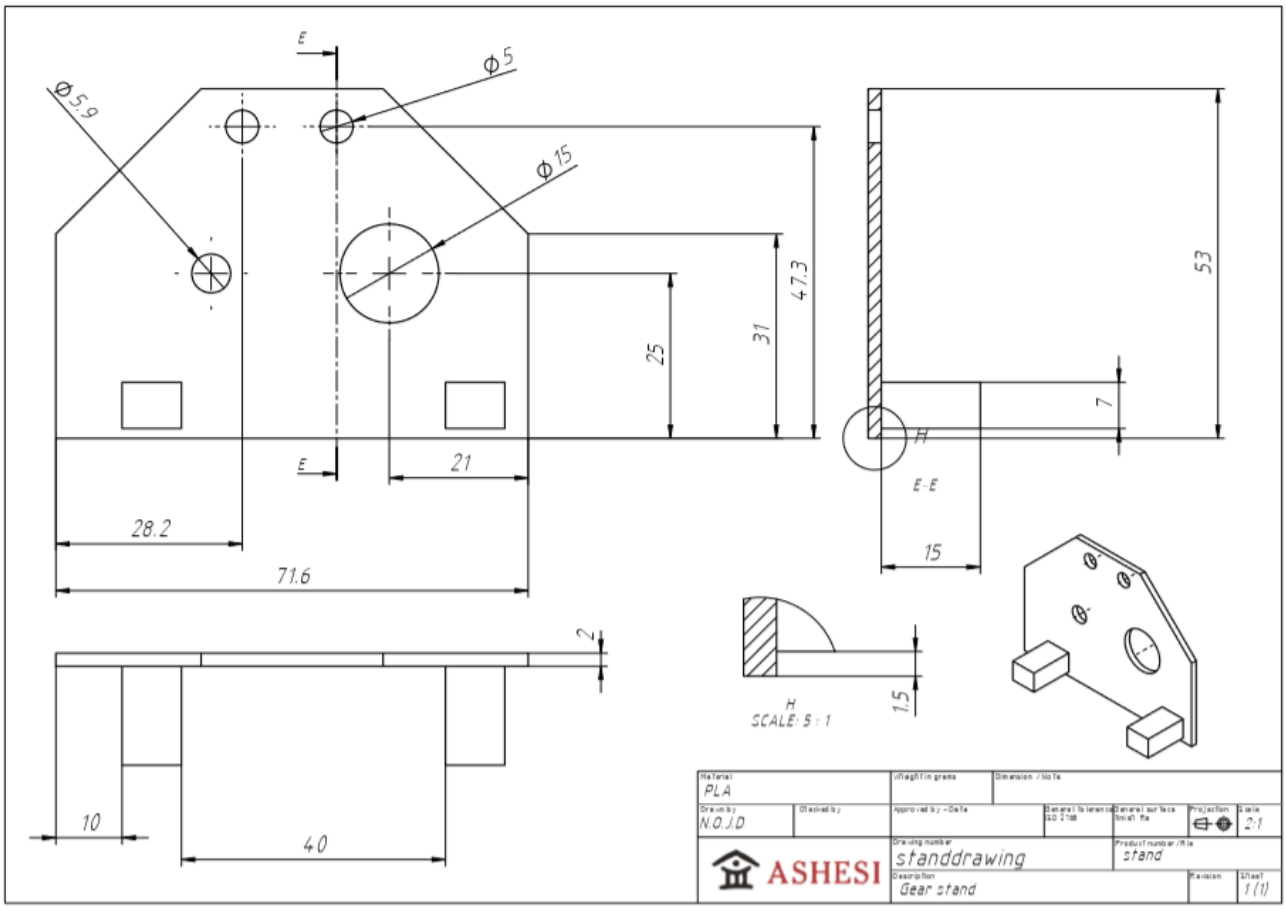


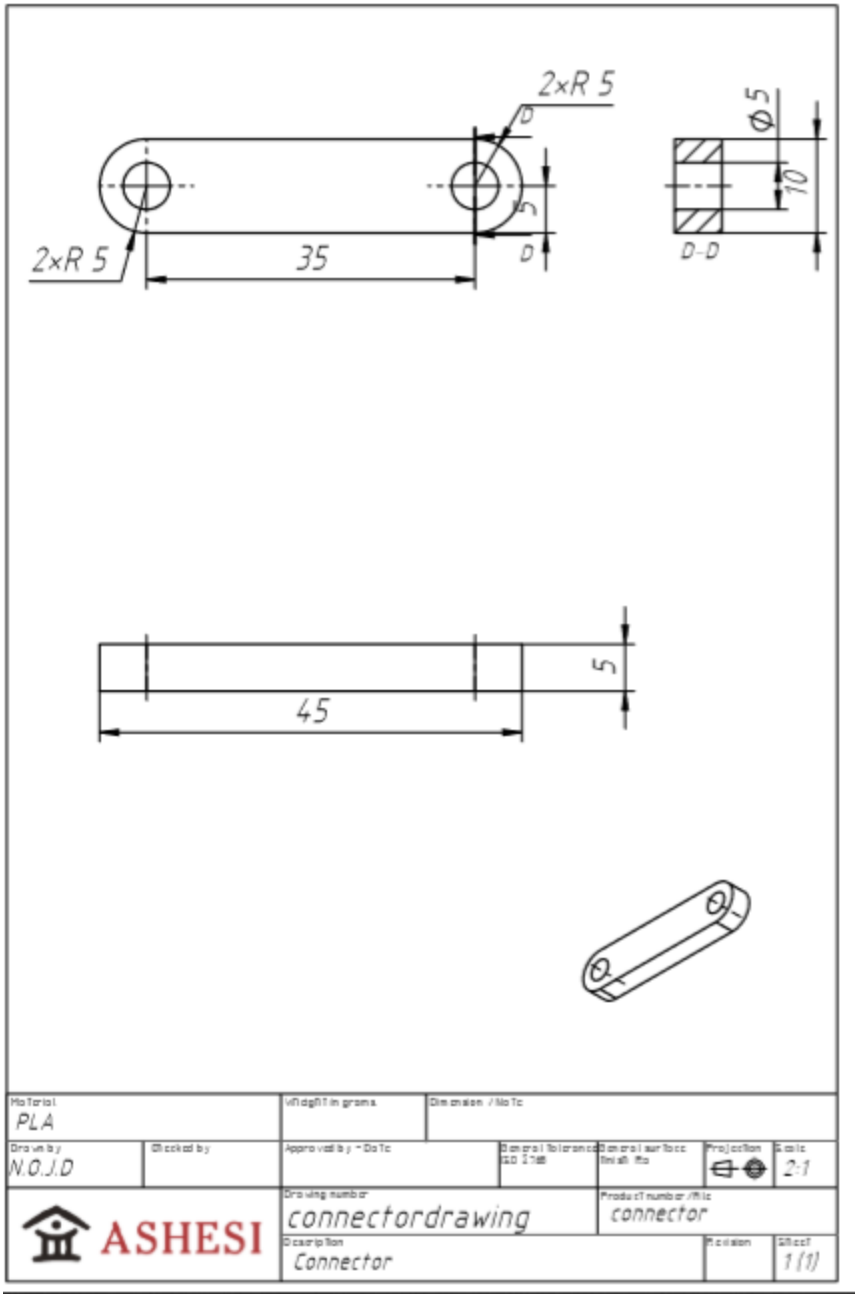


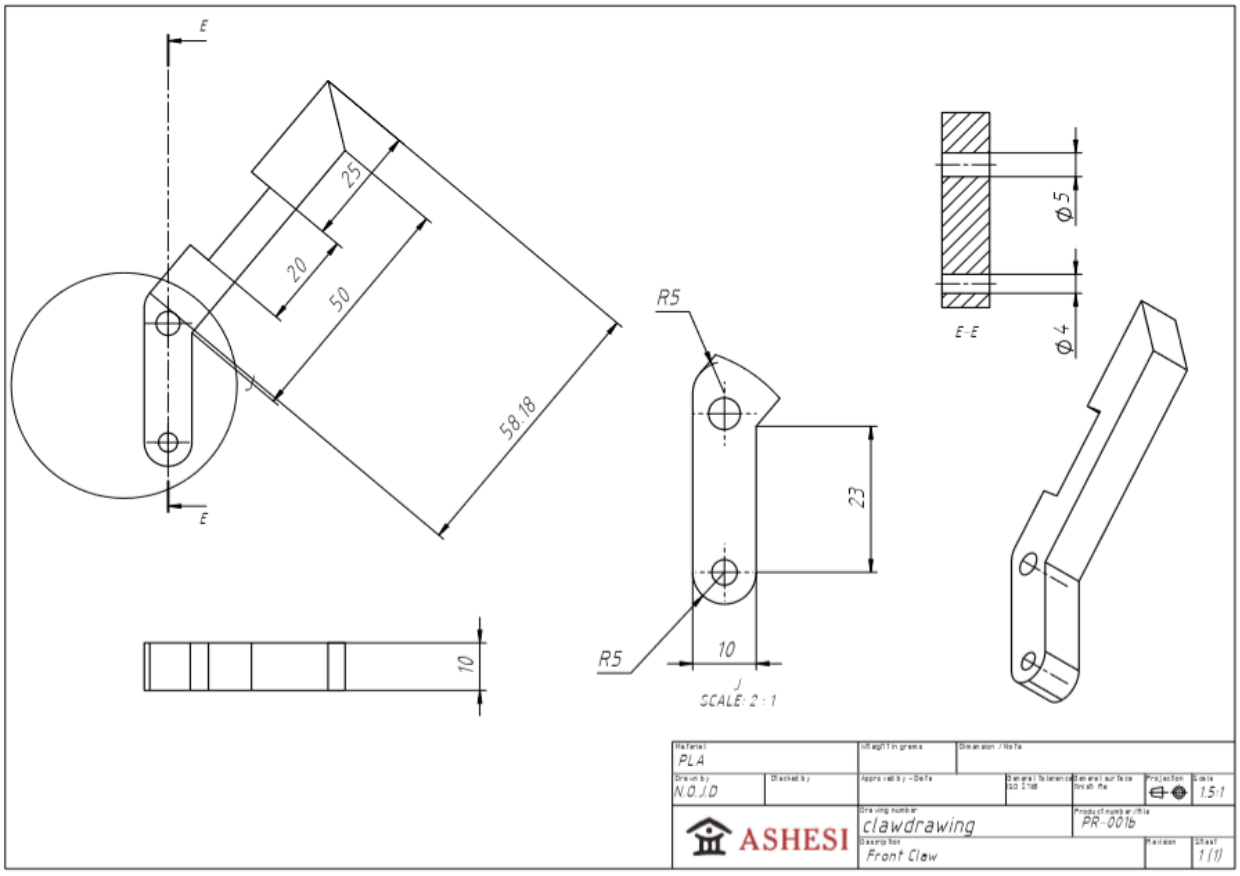
Our robot has a shoulder, elbow, and wrist joints and a grip which is capable of picking objects of size up to 4 cm in width, and this robot is controlled using an Arduino code and can be controlled manually. In addition, it can pick and place objects repeatedly. The robot pick object moving on the conveyor belt based on their colors and then place them in the allocated place or bin. Once the color sensor has detected the object color, LCD print the distance at which the object is from the sensor and then print the color name either Red, Green, and Blue. If the other color is passed the sensor detect it but it and the robotic arm pick it form the belt to the destination. In addition, the

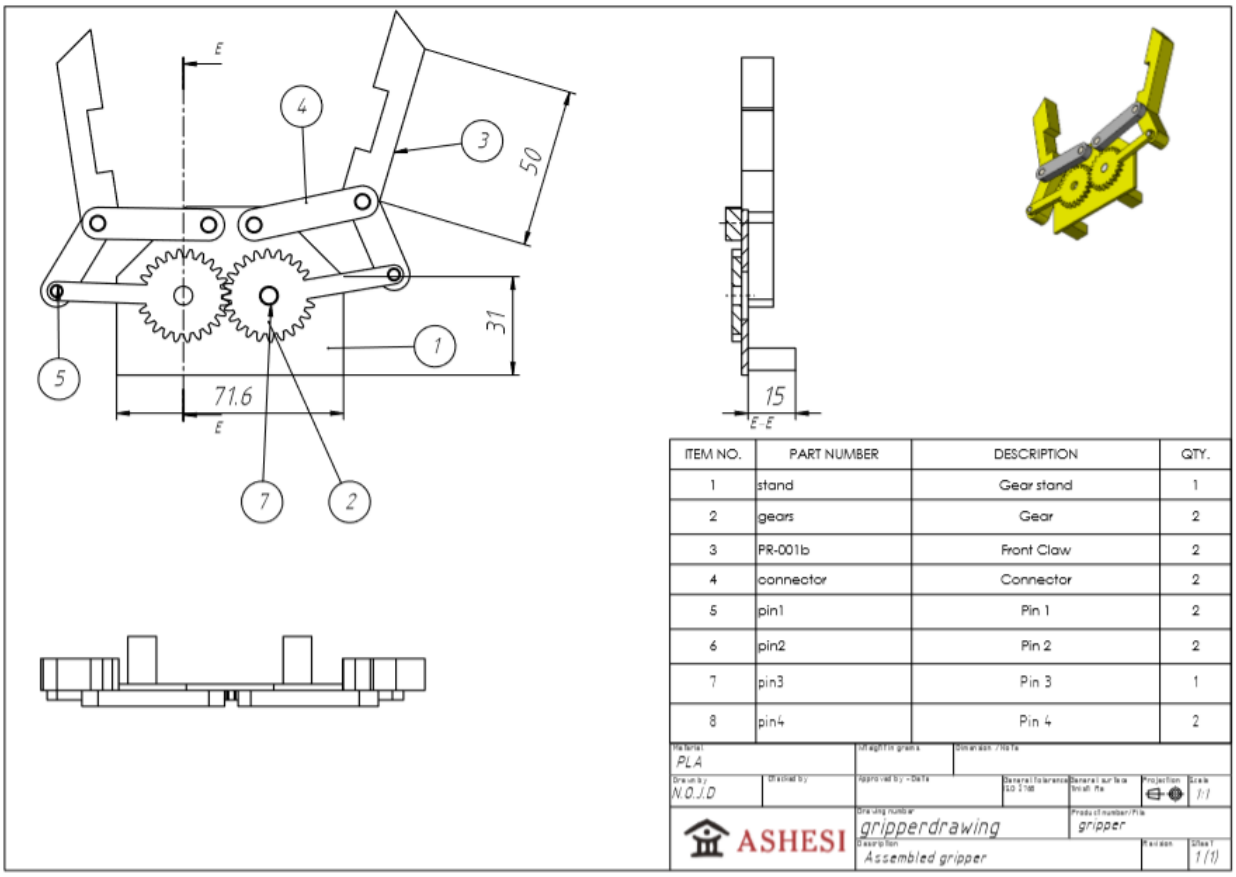
robotic can pick object from a given position and put it to the conveyor belt repeatedly. The robot can perform all these tasks automatically and manually by controlling it using potentiometers.

Gripper CAD Drawings




2×R 5
D
Ø5
10
5
2×R 5
35
D
D-D
45
5
Material
PLA
Weight in grams
Dimension / No Tc
Drawn by
N.O.J.D
Checked by
Approved by - Da Tc
General Tolerance
ISO 2768
General surface finish ffo
Projection
Scale
2:1
ASHESI
Drawing number
connectordrawing
Product number / title
connector
Description
Connector
Revision
Sheet
1 (1)

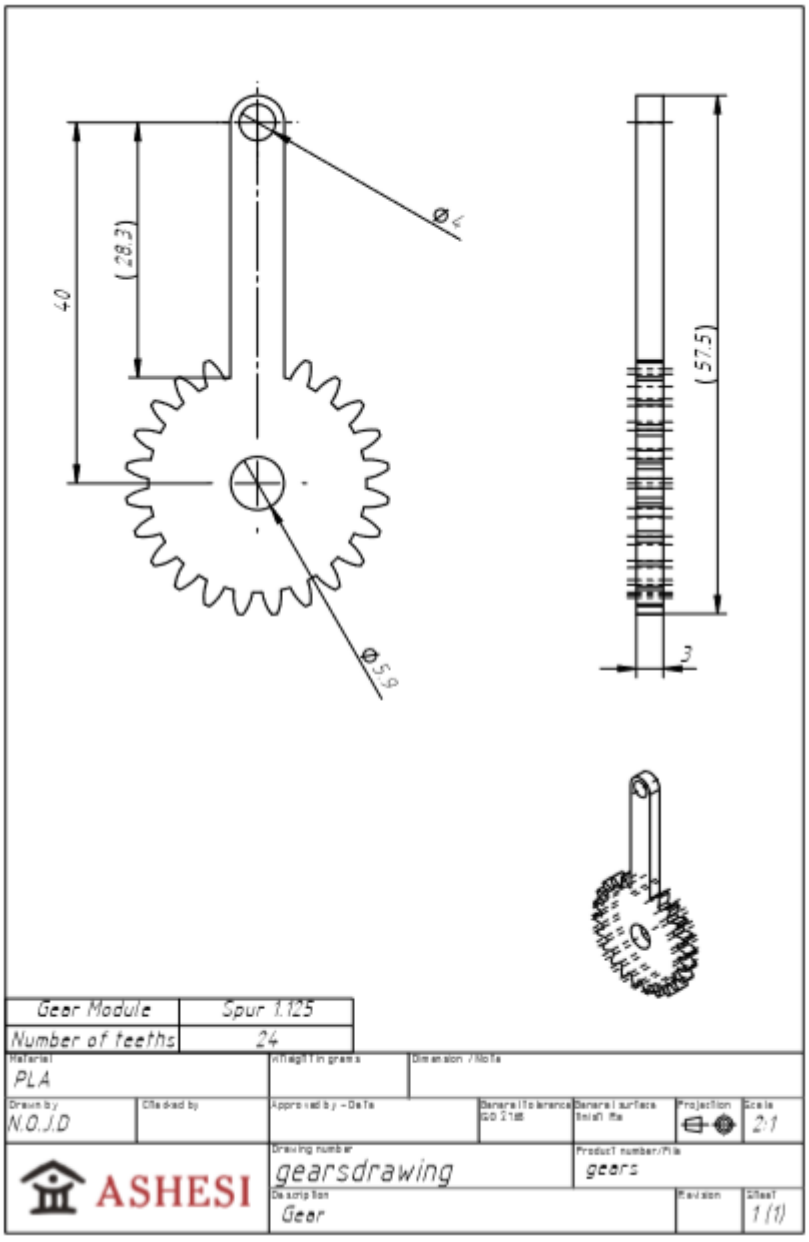

E
25
20
50
58.18
E
10
R5
23
10
R5
SCALE: 2 : 1
Ø5
Ø4
E-E
PLA
N.O.J.D
ASHESI
clawdrawing
Front Claw
PR-001b
1:1
1 (1)



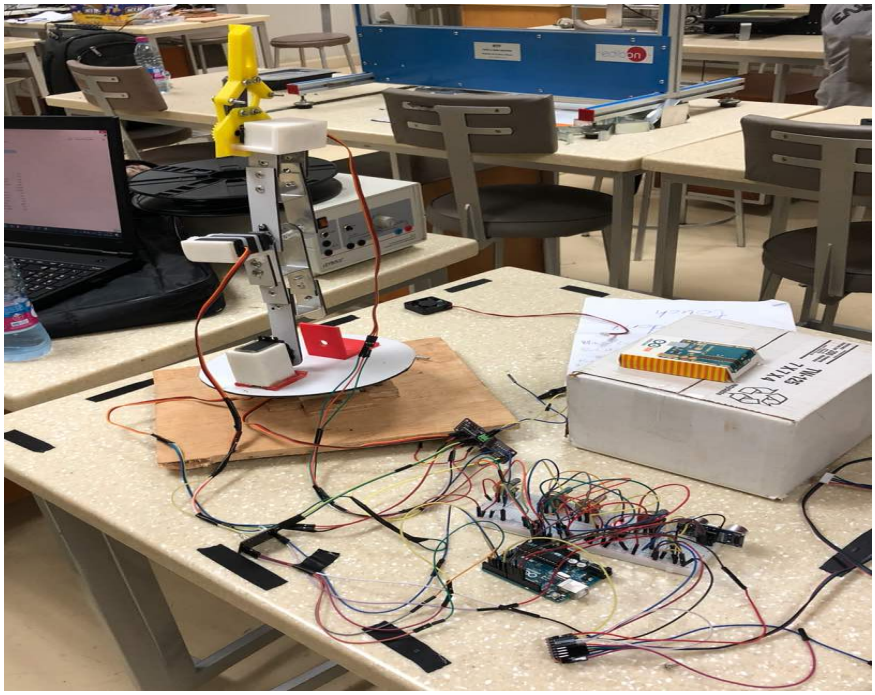
ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	stand	Gear stand	1
2	gears	Gear	2
3	PR-001b	Front Claw	2
4	connector	Connector	2
5	pin1	Pin 1	2
6	pin2	Pin 2	2
7	pin3	Pin 3	1
8	pin4	Pin 4	2

Material: PLA
Drawn by: N.O.J.D
ASHESI
Drawing number: gripperdrawing
Product number/Pla: gripper
Description: Assembled gripper
Sheet: 1 (1)

SOLIDWORKS Educational Product. For Instructional Use Only.


Ø4
(28.3)
40
(57.5)
3
Ø59
Gear Module
Spur 1.125
Number of teeths
24
Material
PLA
Weight in grams
Dimension / Note
Drawn by
N.O.J.D
Checked by
Approved by – Date
General Tolerance
ISO 2768
General surface
finish Ra
Projection
Scale
2:1
Drawing number
gearsdrawing
Product number/File
gears
ASHESI
Description
Gear
Revision
Sheet
1 (1)

Circuitry



Following is the Arduino code that controls system,

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <LiquidCrystal_I2C.h>

// called this way, it uses the default address 0x40
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

#define SERVOMIN  115 // this is the 'minimum' pulse length count (out of 4096)
#define SERVOMAX  560 // this is the 'maximum' pulse length count (out of 4096)
#define S0 6
#define S1 5
#define S2 3
#define S3 4
```

```
#define sensorOut 2

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); //initializing the LCD display

int Rw = 35; // color sensor red diodes low val
int Rb = 300; // color sensor red diodes high val
int Gw = 40; // color sensor green diodes low val
int Gb = 320; // color sensor green diodes high val
int Bw = 35; // color sensor blue diodes low val
int Bb = 255; // color sensor blue diodes high val
unsigned long interval = 4000;
unsigned long previousMillis = 0;
unsigned long currentMillis;
int buttonPushCounter = 0; // counter for the number of mode button presses
int buttonState = 0;         // current state of the mode button
int lastButtonState = 0;     // previous state of the mode button
int buttonStepperPushCounter = 0;   // counter for the number of stepper button presses
int buttonStepperState = 0;         // current state of the stepper button
int lastButtonStepperState = 0;     // previous state of the stepper button
int potpinwaist = A0;  // analog pin used to connect the waist potentiometer
int potvalwaist;    // variable to read the value from the waist potentiometer
int potpinshoulder = A1;  // analog pin used to connect the shoulder potentiometer
int potvalshoulder;    // variable to read the value from the shoulder potentiometer
int potpinelbow = A2; // analog pin used to connect the elbow potentiometer
int potvalelbow; // variable to read the value from the elbow potentiometer
int potpingripper = A3; // analog pin used to connect the gripper potentiometer
int potvalgripper; // variable to read the value from the gripper potentiometer
const int buttonPin = 8;    // the pin that the mode pushbutton is attached to
const int buttonStepperPin = 13;    // the pin that the stepper pushbutton is attached to
const int stepPin = 9; //stepper motor step pin number
```

```
const int dirPin = 10; //stepper motor dir pin number
const int trigPin = 12; //ultrasonic sensor trig pin number
const int echoPin = 11; //ultrasonic sensor echo pin number
const int ultrasonicLED = 7; //ultrasonic sensor LED pin number
const int waist = 1; // Waist(Round) servo at port 1 on PCA9685
const int shoulder = 3; // Shoulder(Front_Back) servo at port 3 on PCA9685
const int elbow = 5;  // Elbow(Up_Down) servo at port 5 on PCA9685
const int gripper = 6; // Gripper(Open_Close) servo at port 6 on PCA9685
int i, distance, R, G, B; //initializing some variables that would be used later
int frequency = 0;
long duration;
static int mode = 0;
String color;

void setup() {
  //PinModes
  pinMode(S0, OUTPUT); //color sensor
  pinMode(S1, OUTPUT); //color sensor
  pinMode(S2, OUTPUT); //color sensor
  pinMode(S3, OUTPUT); //color sensor
  pinMode(sensorOut, INPUT); //color sensor
  digitalWrite(S0, HIGH); // color sensor
  digitalWrite(S1, LOW); //color sensor
  pinMode(ultrasonicLED, OUTPUT); //ultrasonic sensor
  pinMode(buttonPin, INPUT); // initialize the button pin as a input
  pinMode(buttonStepperPin, INPUT); // initialize the button pin as a input
  pinMode(stepPin, OUTPUT); //stpper motor
  pinMode(dirPin, OUTPUT); //stepper motor
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
```

```
  lcd.begin(16, 2);
  lcd.setBacklightPin(3, POSITIVE);
  lcd.setBacklight(1);
  Serial.begin(9600);     // Set the boud rate to 9600

  mode = 1;  // Default state of operation

  pwm.begin();
  pwm.setPWMFreq(60);  // Analog servos run at ~60 Hz updates
  delay(10);
}

void loop() {
  buttonState = digitalRead(buttonPin); // read the pushbutton input pin
  // compare the buttonState to its previous state
  if (buttonState != lastButtonState) {
    // if the state has changed, increment the counter
    if (buttonState == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      buttonPushCounter++;
      Serial.print("number of mode button pushes: ");
      Serial.println(buttonPushCounter);
    }
    // Delay a little bit to avoid bouncing
    delay(50);
  }
  // save the current state as the last state, for next time through the loop
  lastButtonState = buttonState;
```

```
// changes robotic arm mode every button push by checking the modulo of the
// button push counter. the modulo function gives you the remainder of the
// division of two numbers:
if (buttonPushCounter % 2 == 0) {
  mode = 1;
  Serial.println("Manual mode activated");
} else {
  mode = 2;
  Serial.println("Automatic mode activated");
}

// If performing manual control use this
if (mode == 1) {
  buttonStepperState = digitalRead(buttonStepperPin); // read the stepper pushbutton input pin
  // compare the stepper buttonState to its previous state
  if (buttonStepperState != lastButtonStepperState) {
    // if the state has changed, increment the counter
    if (buttonStepperState == HIGH) {
      // if the current state is HIGH then the button went from off to on:
      buttonStepperPushCounter++;
      Serial.print("number of stepper button pushes: ");
      Serial.println(buttonStepperPushCounter);
    }
    // Delay a little bit to avoid bouncing
    delay(50);
  }
  // save the current state as the last stepper state, for next time through the loop
  lastButtonStepperState = buttonStepperState;
  // initially the stepper motor moves the belt forward
  if (buttonStepperPushCounter == 0) {
```

```
  digitalWrite(dirPin, HIGH);
  for (int x = 0; x < 200; x++) {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(2000);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(2000);
  }
}
//when it is clicked the belt stops
if (buttonStepperPushCounter == 1) {
  buttonStepperPushCounter = 1;
}
//when clicked again, the belt reverses
if (buttonStepperPushCounter == 2) {
  digitalWrite(dirPin, LOW);
  for (int x = 0; x < 200; x++) {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(2000);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(2000);
  }
}
//if the button has been pushed 3 times, the button resets to 0 and the loop continues
if (buttonStepperPushCounter > 2) {
  buttonStepperPushCounter = 0;
}
//Ultrasonic Sensor
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
```

```
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2;
  // Prints the distance on the Serial Monitor and LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Distance: ");
  lcd.print(distance);
  Serial.print("Distance: ");
  Serial.println(distance);

  if (distance > 15) {
    // If an object is not sensed, nothing happens, light is off
    digitalWrite(ultrasonicLED, LOW);
  }
  else {
    // If an object is sensed, the belt stops
    // LED turns on
    digitalWrite(ultrasonicLED, HIGH);
    // Objects color is read and printed on serial monitor and LCD
    color = readcolor();
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print(color);
    lcd.print(" ");
```

```
    Serial.print(color);
    Serial.println("  ");
  }
  potvalwaist = analogRead(potpinwaist);            // reads the value of the waist potentiometer (value between 0 and 1023)
  potvalwaist = map(potvalwaist, 0, 1023, 0, 180);     // scale it to use it with the servo (value between 0 and 180)
  pwm.setPWM(waist, 0, angletopulse(potvalwaist));              // sets the waist servo position according to the scaled value
  delay(15);                    // waits for the servo to get there
  potvalshoulder = analogRead(potpinshoulder);              // reads the value of the shoulder potentiometer (value between 0 and 1023)
  potvalshoulder = map(potvalshoulder, 0, 1023, 0, 180);     // scale it to use it with the servo (value between 0 and 180)
  pwm.setPWM(shoulder, 0, angletopulse(potvalshoulder));              // sets the shoulder servo position according to the scaled value
  delay(15);                    // waits for the servo to get there
  potvalelbow = analogRead(potpinelbow);            // reads the value of the elbow potentiometer (value between 0 and 1023)
  potvalelbow = map(potvalelbow, 0, 1023, 0, 180);     // scale it to use it with the servo (value between 0 and 180)
  pwm.setPWM(elbow, 0, angletopulse(potvalelbow));              // sets the elbow servo position according to the scaled value
  delay(15);
  potvalgripper = analogRead(potpingripper);        // reads the value of the grippert potentiometer (value between 0 and 1023)
  potvalgripper = map(potvalgripper, 0, 1023, 0, 180);    // scale it to use it with the servo (value between 0 and 180)
  pwm.setPWM(gripper, 0, angletopulse(potvalgripper));            // sets the gripper servo position according to the scaled value
  delay(15);
 }


 // If performing automatic control use this
```

```
  if (mode == 2) {
    Serial.println("Starting auto mode");
    //Conyeyor belt
    // Clears the trigPin
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    // Sets the trigPin on HIGH state for 10 micro seconds
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    // Reads the echoPin, returns the sound wave travel time in microseconds
    duration = pulseIn(echoPin, HIGH);
    // Calculating the distance
    distance = duration * 0.034 / 2;
    // Prints the distance on the Serial Monitor and LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Distance: ");
    lcd.print(distance);
    Serial.print("Distance: ");
    Serial.println(distance);

    if (distance > 15) {
      // If an object is not sensed, keep the conveyor belt in motion
      digitalWrite(ultrasonicLED, LOW);
      digitalWrite(dirPin, HIGH);
      for (int x = 0; x < 200; x++) {
        digitalWrite(stepPin, HIGH);
        delayMicroseconds(2000);
        digitalWrite(stepPin, LOW);
```

```
      delayMicroseconds(2000);
    }
  }
  else {
    // If an object is sensed, the belt stops
    // LED turns on
    delay(2000);
    digitalWrite(ultrasonicLED, HIGH);
    // Objects color is read and printed on serial monitor and LCD
    color = readcolor();
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print(color);
    lcd.print(" ");
    Serial.print(color);
    Serial.println("  ");
    currentMillis = millis();
    //Belt moves small distance (quater stepper motor cycle) to allow room for robotic arm to pick object
    if (currentMillis - previousMillis >= interval) {
      digitalWrite(dirPin, HIGH);
      for (int x = 0; x < 50; x++) {
        digitalWrite(stepPin, HIGH);
        delayMicroseconds(2000);
        digitalWrite(stepPin, LOW);
        delayMicroseconds(2000);
      }
      previousMillis = millis();
    }
    if (color == "Red") {
```

```
// Robot moves specific positions based on color sensor's output. For Red:
for (int i = 100; i < 110; i++) {
  pwm.setPWM(elbow, 0, angletopulse(i));
  delay(40);
}
for (int i = 100; i < 180; i++) {
  pwm.setPWM(gripper, 0, angletopulse(i));
  delay(40);
}
for (int i = 70; i < 110; i++) {
  pwm.setPWM(shoulder, 0, angletopulse(i));
  delay(40);
}
for (int i = 180; i > 100; i--) {
  pwm.setPWM(gripper, 0, angletopulse(i));
  delay(40);
}
for (int i = 110; i > 70; i--) {
  pwm.setPWM(shoulder, 0, angletopulse(i));
  delay(40);
}
for (int i = 110; i > 100; i--) {
  pwm.setPWM(elbow, 0, angletopulse(i));
  delay(40);
}
for (int i = 0; i < 170; i++) {
  pwm.setPWM(waist, 0, angletopulse(i));
  delay(40);
}
for (int i = 70; i < 100; i++) {
```

```
    pwm.setPWM(shoulder, 0, angletopulse(i));
    delay(40);
   }
   for (int i = 100; i > 180; i++) {
    pwm.setPWM(gripper, 0, angletopulse(i));
    delay(40);
   }
   for (int i = 180; i > 100; i--) {
    pwm.setPWM(gripper, 0, angletopulse(i));
    delay(40);
   }
   for (int i = 100; i > 70; i--) {
    pwm.setPWM(shoulder, 0, angletopulse(i));
    delay(40);
   }
   for (int i = 170; i > 0; i--) {
    pwm.setPWM(waist, 0, angletopulse(i));
    delay(40);
   }
  }
  if (color == "Green") {
   // Robot moves specific positions based on color sensor's output. For Green:
   for (int i = 100; i < 110; i++) {
    pwm.setPWM(elbow, 0, angletopulse(i));
    delay(40);
   }
   for (int i = 100; i < 180; i++) {
    pwm.setPWM(gripper, 0, angletopulse(i));
    delay(40);
   }
```

```
for (int i = 70; i < 110; i++) {
  pwm.setPWM(shoulder, 0, angletopulse(i));
  delay(40);
}
for (int i = 180; i > 100; i--) {
  pwm.setPWM(gripper, 0, angletopulse(i));
  delay(40);
}
for (int i = 110; i > 70; i--) {
  pwm.setPWM(shoulder, 0, angletopulse(i));
  delay(40);
}
for (int i = 110; i > 100; i--) {
  pwm.setPWM(elbow, 0, angletopulse(i));
  delay(40);
}
for (int i = 0; i < 140; i++) {
  pwm.setPWM(waist, 0, angletopulse(i));
  delay(40);
}
for (int i = 70; i < 100; i++) {
  pwm.setPWM(shoulder, 0, angletopulse(i));
  delay(40);
}
for (int i = 100; i > 180; i++) {
  pwm.setPWM(gripper, 0, angletopulse(i));
  delay(40);
}
for (int i = 180; i > 100; i--) {
  pwm.setPWM(gripper, 0, angletopulse(i));
```

```
    delay(40);
   }
   for (int i = 100; i > 70; i--) {
    pwm.setPWM(shoulder, 0, angletopulse(i));
    delay(40);
   }
   for (int i = 140; i > 0; i--) {
    pwm.setPWM(waist, 0, angletopulse(i));
    delay(40);
   }
  }
  if (color == "Blue") {
   // Robot moves specific positions based on color sensor's output. For Blue:
   for (int i = 100; i < 110; i++) {
    pwm.setPWM(elbow, 0, angletopulse(i));
    delay(40);
   }
   for (int i = 100; i < 180; i++) {
    pwm.setPWM(gripper, 0, angletopulse(i));
    delay(40);
   }
   for (int i = 70; i < 110; i++) {
    pwm.setPWM(shoulder, 0, angletopulse(i));
    delay(40);
   }
   for (int i = 180; i > 100; i--) {
    pwm.setPWM(gripper, 0, angletopulse(i));
    delay(40);
   }
   for (int i = 110; i > 70; i--) {
```

```
  pwm.setPWM(shoulder, 0, angletopulse(i));
  delay(40);
 }
 for (int i = 110; i > 100; i--) {
  pwm.setPWM(elbow, 0, angletopulse(i));
  delay(40);
 }
 for (int i = 0; i < 100; i++) {
  pwm.setPWM(waist, 0, angletopulse(i));
  delay(40);
 }
 for (int i = 70; i < 100; i++) {
  pwm.setPWM(shoulder, 0, angletopulse(i));
  delay(40);
 }
 for (int i = 100; i > 180; i++) {
  pwm.setPWM(gripper, 0, angletopulse(i));
  delay(40);
 }
 for (int i = 180; i > 100; i--) {
  pwm.setPWM(gripper, 0, angletopulse(i));
  delay(40);
 }
 for (int i = 100; i > 70; i--) {
  pwm.setPWM(shoulder, 0, angletopulse(i));
  delay(40);
 }
 for (int i = 100; i > 0; i--) {
  pwm.setPWM(waist, 0, angletopulse(i));
  delay(40);
```

```
      }
     }
    }
  }
}
//Function that converts angle to pulses for PCA9685 motor driver
int angletopulse(int ang) {
  int pulse = map(ang, 0, 180, SERVOMIN, SERVOMAX);
  return pulse;
}

//Function that checks color of object using the color sensor
String readcolor() {
  // Setting red filtered photodiodes to be read
  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);
  // Reading the output frequency
  frequency = pulseIn(sensorOut, LOW);
  //mapping output frequency from 0 to 255
  R = map(frequency, Rw, Rb, 0, 255);
  //Prints mapped value to serial monitor
  Serial.print("R= ");
  Serial.print(R);
  Serial.println("  ");
  delay(50);
  // Setting green filtered photodiodes to be read
  digitalWrite(S2, HIGH);
  digitalWrite(S3, HIGH);
  // Reading the output frequency
  frequency = pulseIn(sensorOut, LOW);
```

```
  //mapping output frequency from 0 to 255
  G = map(frequency, Gw, Gb, 0, 255);
  //Prints mapped value to serial monitor
  Serial.print("G= ");
  Serial.print(G);
  Serial.println("  ");
  delay(50);
  // Setting green filtered photodiodes to be read
  digitalWrite(S2, LOW);
  digitalWrite(S3, HIGH);
  // Reading the output frequency
  frequency = pulseIn(sensorOut, LOW);
  //mapping output frequency from 0 to 255
  B = map(frequency, Bw, Bb, 0, 255);
  //Prints mapped value to serial monitor
  Serial.print("B= ");
  Serial.print(B);
  Serial.println("  ");
  delay(50);


  //Depending on different color frequencies, conditions are used to determine whether object is red, green or blue.
  //If red is less than both green and blue, color must be red
  if (R < G && R < B) {
    color = "Red"; // Red
  }
  //If green is less than both red and blue, color must be green
  if (G < R && G < B) {
    color = "Green"; // Green
  }
```

```
  //If blue is less than both green and red, color must be red
  if (B < R && B < G) {
    color = "Blue"; // Blue
  }
  return color;
}
```

Conclusion

Overall, working on this project was great. Despite facing challenges such as limited time, working in the workshop and coding the system to mimic factory production line operations, we were able to pull through and at the end, the system worked perfectly. This project proves the way Arduino can be used to control certain tasks in our life, and the way the robotic arm can be used in manufacturing sector where this automated method improves efficiency, reduce the labor, and increase the speed of production. Therefore, we recommend more projects to the next classes because it allows students to think critically, innovate, work in teams, and solve real life problems by applying the concepts learned in class.