BOB: Balanced Organizing Bot

Organizing Superstar

Presented to: Dr. Carine Habchi

Presented by:

Ahmad Kaiss - 201903357

Cynthia Salha - 202004382

Hadi Mazloum - 201904308

Maher Abou Dargham - 202001751

Razan Hmede - 202103291

Presented on: Tuesday, May 14th, 2024

Table of Contents

Table of Figures

Table of Tables

Table of Equations

Introduction

BOB: Balanced Organizing Bot is a conveyer belt-based system that deals with sorting different materials that are placed by the gripper at the beginning of the belt. There is constant communication between the gripper system and the conveyer belt for optimal coordination.

The idea behind this project arises from the urgent need for resource efficiency by sorting materials based on their properties. Therefore, different techniques specific for each property are used for detection.

This project is a demonstration for an environmental sustainability design that can be used on a wider scale to minimize pollution and facilitate recycling.

We aim in this project to demonstrate the working of mechanical based design such as the conveyer belt, gears, pulleys and gripper with software-based applications such as vs-code for computer vision and Arduino IDE for coding.

Description

Conveyer belt sorting system- Detecting plastic, wood, metal –Giving the user output as form of ramp movement based on the result of detection – gripper to automate the placement of the objects on the conveyer belt.

Purpose

Sorting-based projects, can be used for recycling and manufacturing purposes.

Robotic Arm Design I

The robotic arm is a key component of the balanced organizing bot, designed specifically for small-scale object manipulation and organization tasks. It consists of a base, multiple arm segments, joints, and a gripper at the end, each contributing significantly to the arm's overall functionality and precision.

Research

The robotic arm's design draws inspiration from industrial automation, particularly the pick-and-place systems used in manufacturing. In-depth research into existing robotic arm implementations highlighted the importance of efficiency and reliability in repetitive tasks. This research was supplemented by findings from the latest robotics journals and industry reports, which emphasized the need for advancements in lightweight yet strong materials.

Design Description

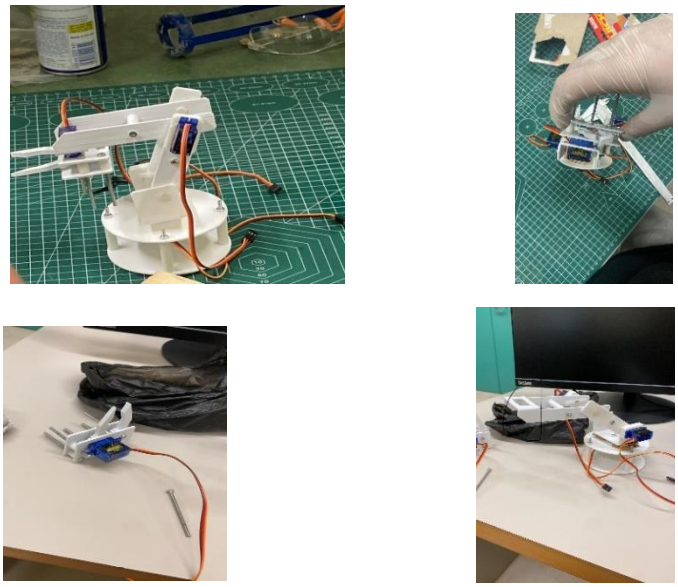

Figure 1 - Robotic Arm I Assembly

The base of the robotic arm serves as the primary mounting point and includes a rotational servo motor that enables it to pivot 360 degrees, as seen in Image 1. The arm segments and joints are designed to provide articulated movement, with each joint being equipped with a servo motor to ensure precise angular control, detailed in Image 2. At the tip of the arm is the gripper, equipped with two fingers and a dedicated servo motor that allows it to grasp and release objects of various sizes, showcased in Image 3.

Manufacturing

The manufacturing process involved 3D printing all components using ABS plastic, a material chosen for its balance of durability and ease of printing. The parts were designed using Solidworks, facilitating rapid prototyping and simplifications in the adjustment process.

Assembly

During assembly, servo motors were attached at each joint, including the base and the gripper. The arm segments were connected using screws and bolts, ensuring robustness while allowing for easy disassembly for maintenance. Additionally, careful integration of the wiring for power and control was carried out to ensure that cables were securely fixed and did not interfere with the operation of the arm.

Limitations

The current design has several limitations. The strength and load capacity are only sufficient for lighter objects, which may restrict the arm's utility with heavier or bulkier items. That, is because of the deficiency of the supply of screws and bearings. Additionally, the range of motion and speed are somewhat limited by the lengths and pivot points of the arm segments and the type of servo motors used.

Improvements

We can make some good improvements to the robotic arm. Using composite materials would make the arm stronger without making it heavier, so it could handle heavier objects. Upgrading to high-performance servo motors would help the arm move faster and more accurately. Also, by adjusting the length of the arm and the design of the joints, we could increase its range of motion and make it more flexible.

Robotic Arm Design II

The gripper was design on SolidWorks and consists of four major parts:

- Base
- Elbow
- Shoulder
- Gripper

It has 4 degrees of freedom.

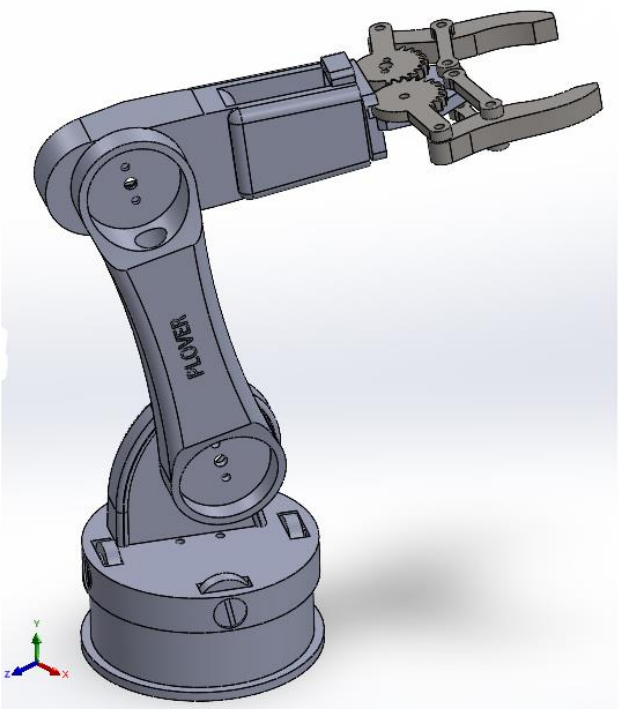

Figure 2 - Robotic Arm Design on SolidWorks

MG996R servo motors were used for the Base and the Elbow since they need more power and torque, while SG90 servo motors were used for the shoulder and the gripper.

The servo motors move based on the coordinates of the target object (x,y), which is the object that will be picked and placed on the belt.

The reference coordinate system was chosen to be at the servo motor of the elbow (just above the base).

The gripper always starts from the initial position as shown in the figure above.

- Base at 0 degrees: facing forward.
- Elbow at 90 degrees.
- Shoulder at 0 degrees (90 degrees w.r.t the elbow).
- Gripper fully open.

Inverse Kinematics

Once we have x and y, inverse kinematics equations are used to calculate the angles needed to move the servo motors and move the gripper to the desired location.

Inverse Kinematics Equations:

1. The distance from the reference to the target is calculated:

$$d = \sqrt{(x^2 + y^2)}$$

Equation 1 – distance from reference to target

2. The angle of the object from the reference is calculated:

$$\theta = \tan^{-1}\left(\frac{x}{y}\right)$$

Equation 2 – angle of the object from the reference

Referring to the following diagram, we can calculate theta 1 and theta 2 using the cosine law and some angle manipulations:

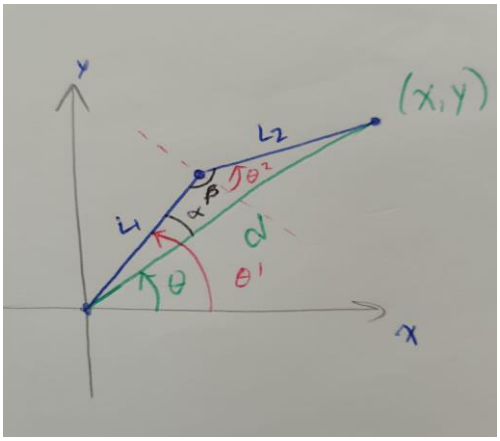

Figure 3 - Inverse Kinematics Diagram

Theta 1 is the angle of the elbow and theta 2 is the angle of shoulder with respect to the elbow.

To get those angles, we must calculate alpha and beta first:

$$\beta = \cos^{-1}\left(\frac{(L1^2 + L2^2 - d^2)}{2L1L2}\right)$$

Equation 3 – beta

$$\alpha = \cos^{-1}\left(\frac{(L1^2 + d^2 - L2^2)}{2L1d}\right)$$

Equation 4 – alpha

This gives us:

$$\theta_1 = \theta + \alpha$$

$\theta_2 = |90 - \beta|$ (Since the initial position of the shoulder is 90 degrees w.r.t the elbow).

Since the gripper will always pick the object from the same position. The x and y coordinates are fixed.

X = 10 and Y = 15 cm.

The links (elbow and shoulder) were measured to be 12 and 17 cm respectively.

The main loop of the code is shown below:

```
void loop() {
  // Coordinates of the target point
  float x = 10, y = 15;
  initialPosition();
  // Ensure gripper is open before moving to the target
  openGripper();

  // Compute inverse kinematics for the given target point
  inv_kinematics(x, y);

  // Wait for the arm to reach the target position
  delay(1000);

  // Close the gripper once the arm has reached the target position
  closeGripper();

  // Rotate base to place object in container behind
  rotateBase();
}
```

Figure 4 - Main Loop of Gripper code

The inverse kinematics method is:

```
void inv_kinematics(float x, float y) {
  // Compute the distance from the origin to the point
  float d = sqrt(pow(x, 2) + pow(y, 2));
  Serial.print("d: ");
  Serial.println(d);

  // Compute angle to point (in degrees) using atan2 for full quadrant coverage
  float theta = degrees(atan2(y, x));
  Serial.print("theta: ");
  Serial.println(theta);

  // Check if the point is within reach
  if ((L1 + L2) >= d && abs(L1 - L2) <= d) {
    // Calculate joint angles using the cosine law
    float beta = degrees(acos((pow(L1, 2) + pow(L2, 2) - pow(d, 2)) / (2 * L1 * L2)));
    float alpha = degrees(acos((pow(L1, 2) + pow(d, 2) - pow(L2, 2)) / (2 * L1 * d)));

    float theta1 = theta + alpha;
    float theta2 = abs(90-beta);

    servo1.write(theta1);
    servo2.write(theta2);

    Serial.println("Arm moved to position.");
  } else {
    Serial.println("Position out of reach.");
    // Move servos to initial position if target is out of reach
    initialPosition();
  }
}
```

Figure 5 - Inverse kinematics code

Note that we included an if statement to ensure that the position is reachable, in other words if its distance is less than the sum of the two links of the robotic arm.

Four Bar Linkage – Grashoff

1- Ground = 22 mm
2- Input = 31 mm
3- Coupler = 22 mm
4- Output = 31 mm

Grashoff Condition:

$$S + L = 53\ mm$$

$$P + Q = 53\ mm$$

$$\rightarrow Grashoff\ Type\ 3\ and\ since\ the\ smallest\ link\ is\ the\ ground \rightarrow Double\ Crank$$

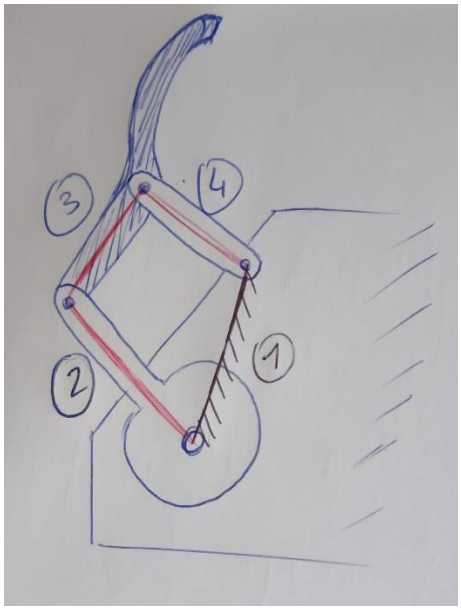

Figure 6 - Four Bar Linkage

Belt Design

To perform the different tests on the objects to be able to identify their material properties, it was decided to run the objects on a conveyor belt that contains multiple sensors that could identify those properties.

Belt Kinematics

The belts used in this project were the only ones available in the market and thus the only viable option for the design and assembly of this project. The belt characteristics are the following:

Type: Timing Belt

Length: 610 mm

Depth: 6 mm

Thickness: 6 mm

The belts were bought with pulleys that are compatible to translate the rotational motion of the DC motors into the linear motion of the belt. The pulleys used have the following characteristics:

Pulley Diameter: 12 mm

Number of Teeth: 20

Diametral Pitch: 42.33

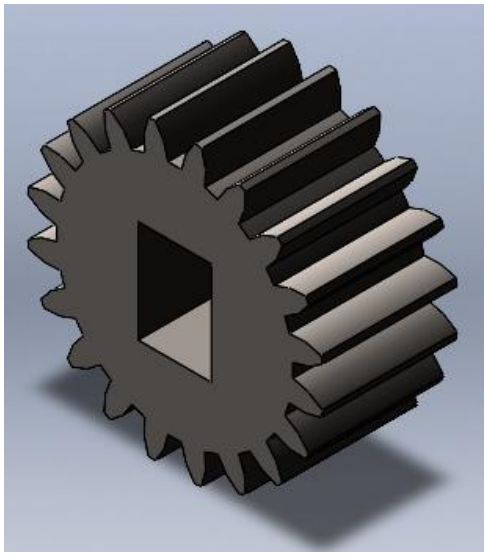

Figure 7 - 3D Model of 3D printed Gear

Columns that consisted of an elevated fitting were 3D printed and fixed to a support, and the pulleys were secured on the fittings with a center distance that suits the acquired belt.

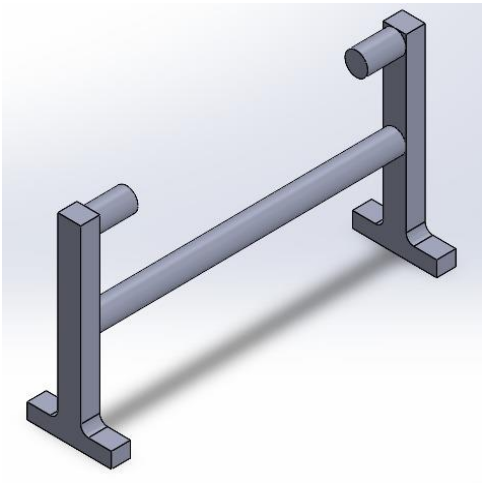

Figure 8 - 3D Model of Pulley Frame

To calculate the center distance first B was obtained:

$$B = 4L - 2\pi(d_2 + d_1)$$

Equation 5 – Belt variable B equation

$$C = \frac{B + \sqrt{B^2 - 32(d_2 - d_1)^2}}{16}$$

Equation 6 – Belt variable C equation

From the previous equations the following values were obtained:

B = 2.2892 cm C = 28.615 cm

With the current structure the rated power is at 100%, since all the pulleys and gears have the same outer diameter.

Lastly, a leather belt was tailored to be fitted on the timing belts with a chosen width of 8 cm.

Following this analysis, it was anticipated that the belt would bend around the middle of the center distance, thus another column was added halfway through the center distance and was fitted with the same pulleys. This reduced its rated power to about 95% (obtained through testing) but was essential for proper functionality, as with the added load of an object the belt could fold on itself near the middle and the belt would fail.

Belt Accessories

The robotic arm, already discussed, is placed before the belt entry.

Belt Drive

At the beginning of the belt, the DC motors were placed as belt drives, and attached to the output shafts of the motors were 3D printed gears that followed the same characteristics as the purchased pulleys. The DC motors were connected to the Arduino through an H-Bridge, which is the proper way of connecting DC motors even though it is not required to change the direction or speed of the motors. They operated on an external 6V battery source to avoid overloading the Aduino.

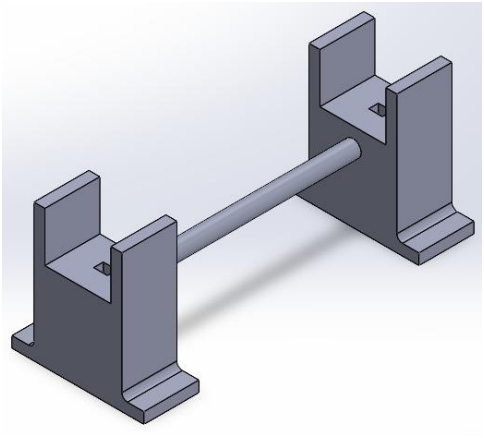

Figure 9 - 3D Model of 3D printed Motor Frame

Inductive Sensor

Also close to the start of the belt, an inductive sensor was attached to detect whether the object is metallic or not. The sensor only has three pins, Vcc and Ground were connected to the Arduino and the excitation pin was connected to a digital pin of the Arduino.

Figure 10 - Induction Sensor

Camera

To be able to operate the computer vision (described in detail in the following section) a camera was placed in a way that it could detect the entirety of the belt to ensure that it captures the object and output an answer before the object passes off the belt.

Rotating Ramp

The ramp was attached to a servo motor so it could rotate to sort the different materials in different boxes. The ramp had a 59° angle with a height of 5cm, length of 3cm, and a depth of 8cm. It was placed 2 cm in front of the end of the belt to avoid coliding with the belt when it rotates.

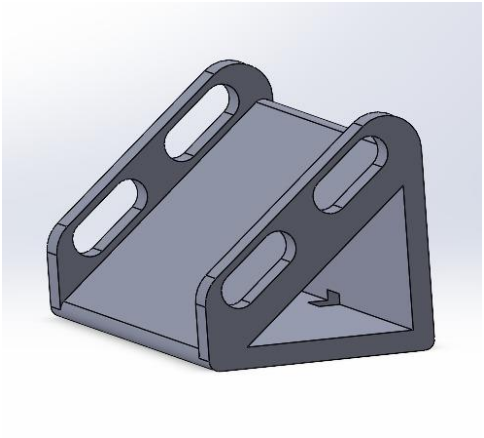

Figure 11 – 3D Model of 3D printed Ramp

Computer Vision

The computer vision part of this project aims to detect a wood block based on the color array specific to a range of wood.

We picked this range for accuracy to be:

lower_brown = np.array([10, 50, 50])

upper_brown = np.array([30, 255, 255])

```
import cv2 as cv
import numpy as np
import sys
import serial
import time

# Initialize Serial communication
ser = serial.Serial('COM7', 9600)
ser.flush()

video = cv.VideoCapture(0)

if not video.isOpened():
    sys.exit("No camera detected")

while True:
    ret, frame = video.read()

    if not ret:
        print("Not all frames captured")
        break  # Exit the loop if no frames are captured

    cvt = cv.cvtColor(frame, cv.COLOR_BGR2HSV)

    lower_brown = np.array([10, 50, 50])
    upper_brown = np.array([30, 255, 255])

    mask = cv.inRange(cvt, lower_brown, upper_brown)

    kernel = np.ones((5,5), np.uint8)
    reduce = cv.morphologyEx(mask, cv.MORPH_OPEN, kernel)

    cv.imshow("reduce", reduce)
    cv.imshow("frame", frame)
```

```
    key = cv.waitKey(1)

    if key == ord("q"):
        break

    brown_area = cv.countNonZero(reduce)
    if brown_area > 5000:  # Adjust threshold as needed
        ser.write(b'b')    # Send command to Arduino
        ser.flush()            # Flush serial buffer to ensure immediate data
transmission

# Close OpenCV windows
cv.destroyAllWindows()

# Close the video capture device
if video.isOpened():
    video.release()

# Close Serial connection
ser.close()
```

A serial communication is established between python code in VScode and the method of detecting wood in the main code in Arduino IDE.

```
void computerVision(){
  if (Serial.available() > 0) {
    char command = Serial.read();

    if (command == 'b') {
      // Read and discard any remaining characters in the serial buffer
      while (Serial.available() > 0) {
        Serial.read();
      }

      Serial.println("Wood is detected");
      isWood = true;
      isMetal = false;
      isPlastic = false;
    } else {
      // Discard any remaining characters in the serial buffer
      while (Serial.available() > 0) {
        Serial.read();
      }
```

```
    Serial.println("Wood not detected");
  }

  delay(500); // Add a small delay to prevent rapid polling of serial buffer
}
}
```

Limitation: Due to the low processing power of the Arduino IDE, the delay in serial communication affected the performance of this part of the detection system.

Results

As we can see below after we run the code, the block of wood is detected in the region marked in white while everything else is ignored.
The test is accurate for detection of wood.

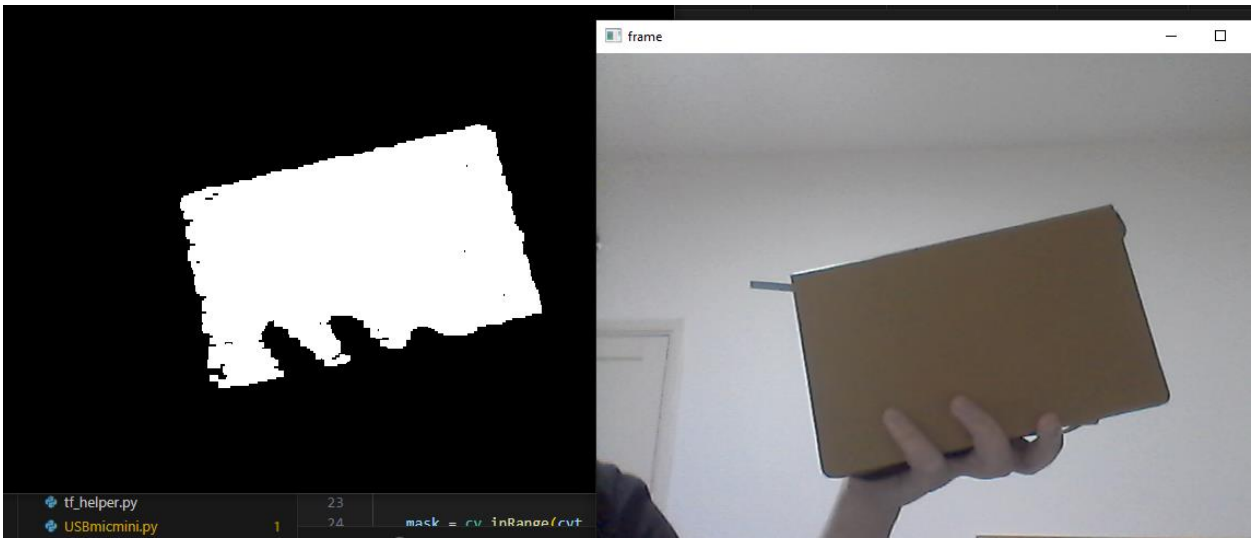

Figure 12 - Results of Computer Vision

Full Code

In this section the full code of the machine will be discussed in detail, the code is all in a single ino file, but it was separated into segments to explain it thoroughly (the code is written in another font). the gripper code was explained previously in the robotic arm design.

Setup

First the servo motor library must be imported, and all variables and pins must be initialized. Then the pins must be specified if they are used as inputs or outputs. And the motors must be off at the beginning of the initialization.

```
#include <Servo.h>
Servo servo;

// Motor A connections
int enA_B = 9;
int in1 = 8;
int in2 = 7;
// Motor B connections
int in3 = 5;
int in4 = 4;

const int metal = 3; //pin for inductive proximity sensor
bool isMetal = false, isPlastic = true, isWood = false; // the servo motor angles are 30 ; 90 ; 150 respectively
bool systemOn = false, firstIteration = true;
const int pushButton = 2; //pin for push button
int trigPin = 13;    // TRIG pin
int echoPin = 12;    // ECHO pin
float duration_us, distance_cm; //values for ultrasonic

void setup() {
 pinMode(metal, INPUT);
 pinMode(pushButton, INPUT);
 Serial.begin(9600);
 servo.attach(6);
```

```
 servo.write(90);


 // configure the trigger pin to output mode
 pinMode(trigPin, OUTPUT);
 // configure the echo pin to input mode
 pinMode(echoPin, INPUT);


 pinMode(enA_B, OUTPUT);
 pinMode(in1, OUTPUT);
 pinMode(in2, OUTPUT);
 pinMode(in3, OUTPUT);
 pinMode(in4, OUTPUT);


 // Turn off motors - Initial state
 digitalWrite(in1, LOW);
 digitalWrite(in2, LOW);
 digitalWrite(in3, LOW);
 digitalWrite(in4, LOW);
}
```

Functions

Detect Material is called whenever a new object is placed on the belt and its material hasn't been detected yet. Its purpose is to call the following two methods responsible for detecting the materials.

```
void detectMaterial(){
  detectMetal();
  computerVision();
}
```

Detect Metal is called to get feedback from the inductive sensor which will update the boolean variables of materials in case the input pin was HIGH.

```
void detectMetal(){
 int object = digitalRead(metal);
  if(object==HIGH) {
    Serial.println("Metal Object Not Detected");
```

```
    delay(50);
  }
  else {
    isMetal = true;
    isWood = false;
    isPlastic = false;
    Serial.println("Metal Object Detected");
    delay(50);
  }
}
```

Computer Vision is called to get feedback from the computer vision which will update the boolean variables of materials in case the ‘Wood is detected” was sent from the python code.

```
void computerVision(){
 if (Serial.available() > 0) {
  char command = Serial.read();

  if (command == 'b') {
   // Read and discard any remaining characters in the serial buffer
   while (Serial.available() > 0) {
    Serial.read();
   }

   Serial.println("Wood is detected");
   isWood = true;
   isMetal = false;
   isPlastic = false;
  } else {
   // Discard any remaining characters in the serial buffer
   while (Serial.available() > 0) {
    Serial.read();
   }
   Serial.println("Wood not detected");
  }
```

```
  delay(500); // Add a small delay to prevent rapid polling of serial buffer
 }
 }
```

Ramp Position is called to adjust the angle of the ramp depending on the type of material that was detcted in this iteration.

```
void rampPosition(){
 if(isMetal){
  servo.write(70);
 }
 else if(isPlastic){
  servo.write(90);
 }
 else{
  servo.write(110);
 }
}
```

Conveyor is called to turn the belt drive on or off depending on the boolean parameter that it resceives.

```
void conveyor(bool conveyorOn){
if (conveyorOn) {
  // Set motors to maximum speed
        // For PWM maximum possible values are 0 to 255
  analogWrite(enA_B, 255);

 // Turn on motor A & B
 digitalWrite(in1, LOW);
 digitalWrite(in2, HIGH);
 digitalWrite(in3, LOW);
 digitalWrite(in4, HIGH);
 delay(2000);
 }
 else {
```

```
  analogWrite(enA_B, 0);

 // Turn on motor A & B
 digitalWrite(in1, HIGH);
 digitalWrite(in2, HIGH);
 digitalWrite(in3, HIGH);
 digitalWrite(in4, HIGH);
 }
}
```

Ultrasonic is called every few milliseconds to detect whether an object has completely passed through the belt and has been sorted to reset the sensors and ramp position and send a signal to the robotic arm to place another object on the belt.

```
void ultrasonic(){
 // generate 10-microsecond pulse to TRIG pin
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);

 // measure duration of pulse from ECHO pin
 duration_us = pulseIn(echoPin, HIGH);

 // calculate the distance
 distance_cm = 0.017 * duration_us;

 // print the value to Serial Monitor
 Serial.print("distance: ");
 Serial.print(distance_cm);
 Serial.println(" cm");

 if(distance_cm<7){
  isMetal = false;
  isPlastic = true;
  isWood = false;
  delay(1000);
```

```
  Serial.println("An Object has been Sorted");
  delay(2000);
  //gripper();
 }
}
```

Main Loop

The following code is the main loop of the system, that keeps on iterating. First the push button state is read and updated. Next is a conditional statement that checks if the system is on or not, if the system is off the belt drive is off and the code will wait until the push button is pressed to turn the system on. If the system is on then it will check if it is the first iteration, if it is then it will activate the robotic arm, if not it will turn on the belt drive (or keep it on). Then the detect material function is called to check if any metal or wood is currently present on the conveyor belt, the ramp position is updated depending on the type of material that is detected, and lastly the ultrasonic function is called to check if the item was sorted or not. Finally, while the system is on, if the push button was pressed it will turn off the motor.

```
void loop() {
 delay(200);
 int PBstate = digitalRead(pushButton); //Standby button
 Serial.println("Pushbutton state: " + PBstate);
 //Serial.println("Plastic state: " + isPlastic);
 //Serial.println("Wood state: " + isWood);
 //Serial.println("Metal state: " + isMetal);
 //detectMaterial();

 if(systemOn){
  Serial.println("System is ON!");
  //Intializing system

  if(firstIteration){
   firstIteration = false;
   delay(10000);
   Serial.println("Gripper Activated");//gripper();
  }
  //turn on conveyor belt
  conveyor(true);
 //detect material
```

```
  detectMaterial();

  rampPosition(); //adjust the ramp angle
  ultrasonic(); //detect presence of object at the last position and activate gripper

  if(PBstate == HIGH){ //if standby, turn off
    systemOn = false;
  }
 }
 else {
  Serial.println("System is OFF!");
  conveyor(false); //turn off motors
  //delay(500);
  if(PBstate == HIGH){ //if standby turn on
    systemOn = true;
  }
 }
}
```

Conclusion

Overall, we demonstrated in this multidisciplinary project the combination between mechanical, electrical and computer-based design that work all together to give a working sorting system that can have many applications including manufacturing and environmental. As we faced some limitations regarding the delay of the serial communication with Arduino and its low processing power, the rupture of the belt that we attached to the timing belts and some other design problems, the project demonstrated good overall performance in the tasks we first highlighted.

Appendix

BOQ

Serial Number	Name	Quantity	Unit	Rate ($)	Amount ($)
1	Ultrasonic Sensor	1	--	1.5	1.5
2	DC Motor	2	--	0.9	1.8
3	Servo Motor	4	--	3	12
4	Inductive Sensor	1	--	3.75	3.75
5	Timing Belt	2	--	3.4	6.8
6	Pulley	4	--	2.25	9
7	Plastic	1.5	Kg	17	25.5
8	Shipping	1	--	3.5	3.5
				Total Cost:	63.85

Table 1 - Bill of Quantities