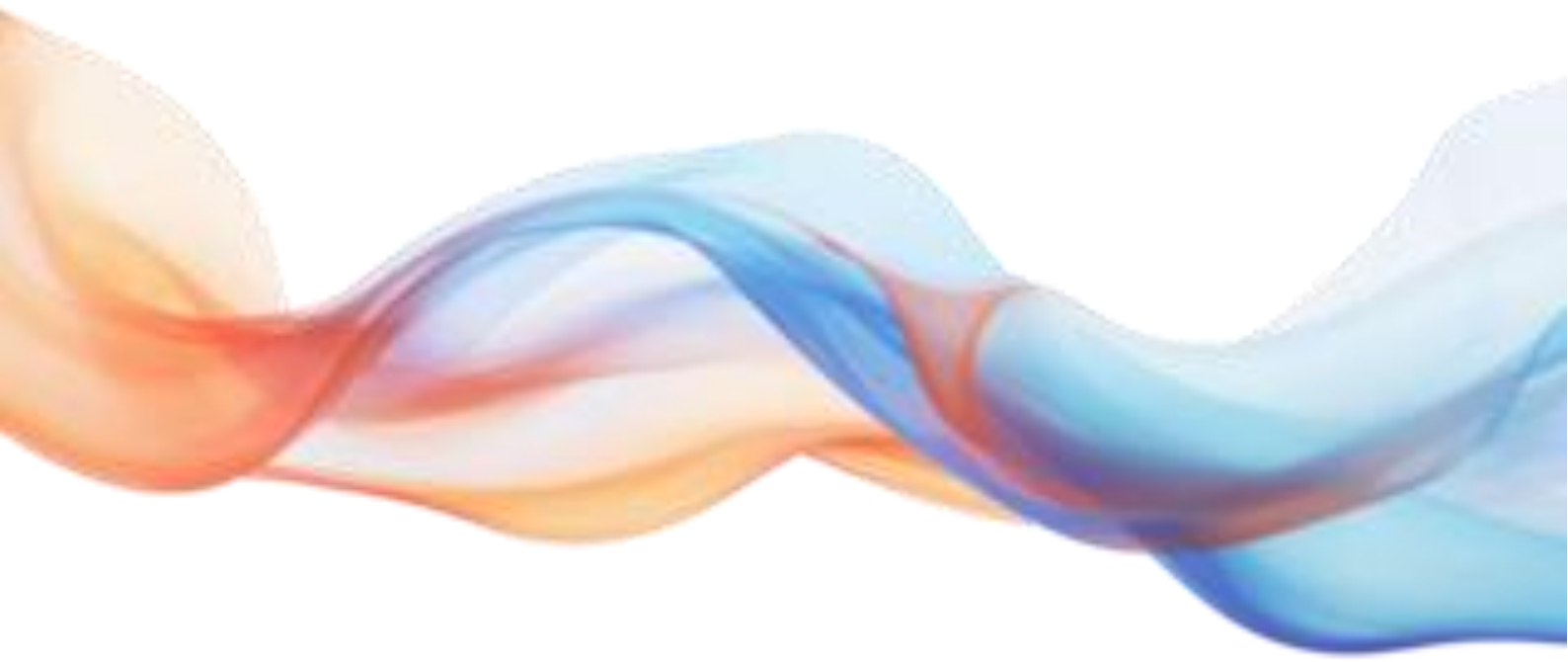
UNIVERSITY OF SUSSEX

University of Sussex School of Engineering and Informatics

Design of a 5-Bar Planar Parallel Robotic Solution for Personalised Chocolate Trays

H7123 - Robot Design and Implementation



Word Count: 3077

Candidate Number: 233977

02/05/2024

Table of Contents

Abstract

The following report outlines a robotic solution to a personalised chocolate box gift manufacturing process for given specifications and made assumptions highlighting kinematic design, mechanical design, end-effector design and machine vision implementation.

1. Introduction

In an innovative venture into personalised gifting, a chocolate manufacturing company introduced a product that transforms customer drawings into edible masterpieces using chocolates of various colours. This project revolves around designing a robotic system capable of assembling these unique chocolate configurations on a specifically designed tray.

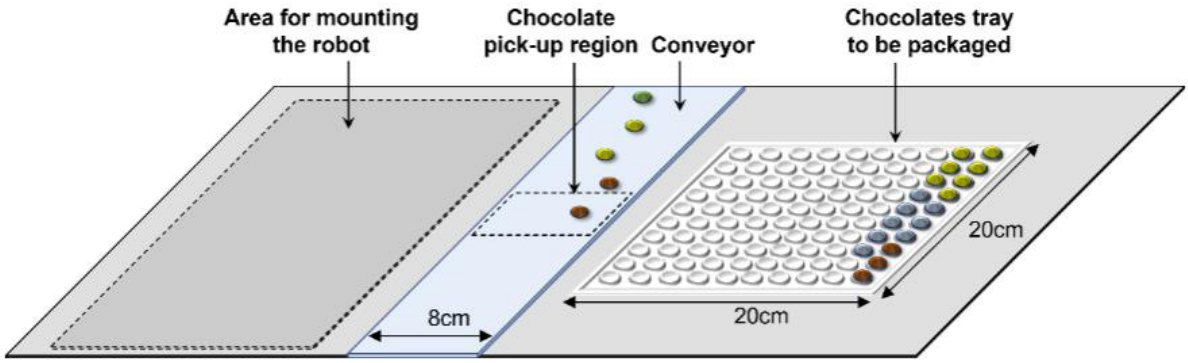


Figure 1. Illustration of chocolate picking and packing environment and application scenario

The robotic solution for this system is based on a 5-bar planar parallel robot equipped with a sophisticated end-effector, tasked with arranging ellipsoid-shaped chocolates into a pre-defined 10 by 10 grid on a tray. The task outlines that the system should make use of cameras to detect the colours of the incoming chocolates via conveyor and converting this information to gain the desired target tray position of this chocolate.

This document outlines an approach to developing a robotic solution meeting the creative demands of customised chocolate artwork but also addressing the technical challenges associated with the automations of such process.

Key features of the system include the integration of high-resolution cameras and sensors for accurate chocolate positioning, the design of an agile end-effector for delicate handling of the chocolates and the development of kinematic models to ensue flawless execution of the chocolate placements.

The subsequent sections will detail the technical designs, mechanical designs and operation protocols for the proposed robotic assembly line.

2. Kinematic Design

This section details the kinematic design solution to this chocolate picking and packing problem. Demonstrating how the required lengths of the linkages were determined, ensuring these fit within the constraints of the robot's workspace, before modelling the forward and inverse kinematic solutions to this 5-bar configuration.

2.1 Workspace and Link Lengths

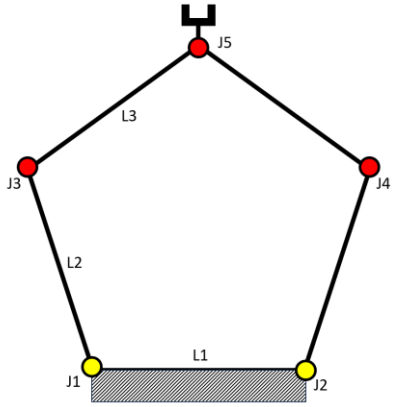


Figure 2. 5-bar planar parallel robot configuration with labelled joints and links

To determine the link lengths, the most suitable approach was to begin by modelling the chocolate picking and packing environment as given in the brief in combination with assumptions of the lengths not explicitly stated to be 2cm. This model of the environment was created using GeoGebra (figure 3), a dynamic mathematics tool for creating mathematical and geometric models. For the purpose of design criteria and consequent forward and inverse kinematics, the origin of the workspace (0, 0) is set at the halfway point between the servos. As an initial design choice, it was agreed to mount the servo motors such that their output shaft lies 10cm into the robot mounting area and 20cm apart, thus establishing the length of link L1.

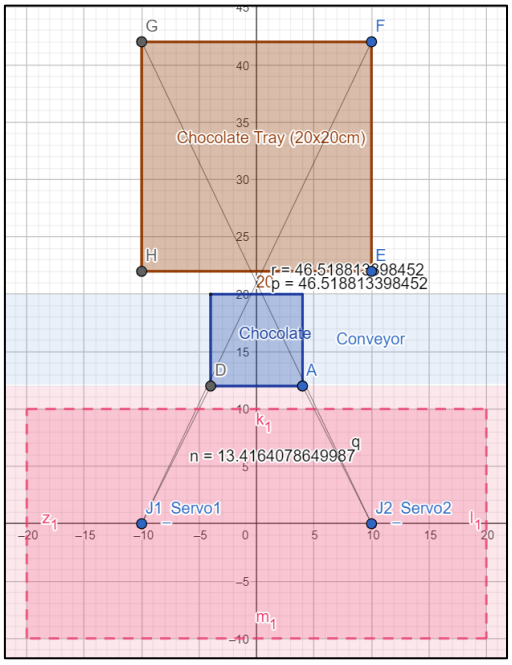


Figure 3. GeoGebra model of the chocolate picking and placing environment with Euclidean distances

In order to ensure the proximal links, connected to the servos, refrain from colliding with the servo motors, the first condition of L2 being less than 20cm. Next, to ensure the robot can reach the furthest corners of the chocolate tray, we must create a second condition: the combined lengths of proximal and distal links (L2 and L3 respectively) must be greater than the Euclidean distance between servos and opposing furthest corner. Finally, in order to ensure the nearest points are equally reachable, a third condition must be met: the difference between proximal and distal links must be less than the Euclidean distance between servos and nearest bottom corner of the chocolate pick-up region, which was assumed to be 8cm^2.

$$L1 \text{ set as } 20\text{cm}$$

$$L2 < 20cm$$

$$L3 - L2 < 13.41cm$$

$$L3 + L2 > 46.52cm$$

Figure 4. Link length conditions derived from project environment specifications

In accordance to the conditional statements in figure 4, the proximal link length (L2) was set as 17cm and the distal link length (L3) set as 30cm. Once these link lengths selected, the complete workspace area of the robot in the '+ +' configuration is modelled using the MATLAB script snippets from the 'Main.m' (Appendix 1.8.2) and 'PlotLinks.m' (Appendix 1.3). The resulting workspace model can be seen plotted in figure 7.

```
% Workspace
workspace=zeros(100,2);
workspace_Angles_J2 = linspace(0,180,100);
workspace_Angles_J1 = linspace(180,0,100);
index = 1;

for i=1:length(workspace_Angles_J2)
    for j=1:length(workspace_Angles_J1)
        A2 = workspace_Angles_J2(i);
        A1 = workspace_Angles_J1(j);
        A = [A1,A2];
        [J1_w, J2_w, J3_w, J4_w, J5_w] = ForwardKinematics(A, L);
        workspace(index, :) = J5_w;
        index = index + 1;
    end
end

PlotLinks(Joints, workspace);
```

Figure 5. 'Main.m' snippet for workspace calculations for robot in '+ +' configuration

```
% Plotting Workspace
plot(workspace(:,1),workspace(:,2),'LineStyle','None','Marker','.','Color','g');
```

Figure 6. 'PlotLink.m' snippet for plotting of the workspace

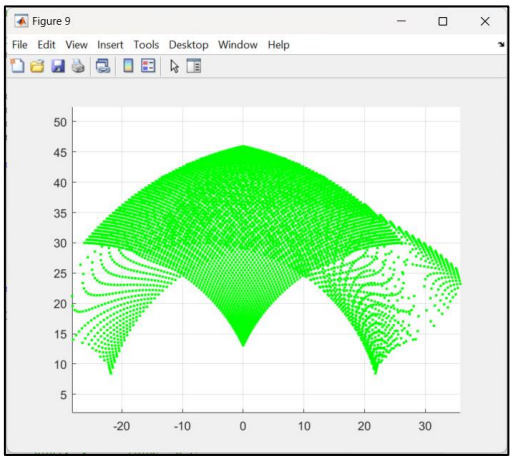


Figure 7. Modelled workspace of robot for established link lengths

2.2 Forward Kinematics

This solution outlines the geometric approach for solving the forward kinematics of the 5-bar planar parallel robot.

Taking in the established link lengths and input servo motor, forward kinematics can be applied to compute the end-effector position and model the 5-bar robot configuration for given input angles.

Let the midpoint between the Servos serve as the origin. From there we can calculate the [X, Y] positions of the servos (J1 and J2) as:

$$J1 = \left[-\frac{L1}{2}, 0\right] \tag{1}$$

$$J2 = \left[\frac{L1}{2}, 0\right] \tag{2}$$

The positions of passive Joints J3 and J4 are then calculated using basic trigonometry:

$$J3 = \left[X_{J1} + L2 * \cos(\theta_1) \quad , \quad Y_{J1} + L2 * \sin(\theta_1)\right] \tag{3}$$

$$J4 = \left[X_{J2} + L2 * \cos(\theta_2) \quad , \quad Y_{J2} + L2 * \sin(\theta_2)\right] \tag{4}$$

From these coordinates we can then get the end effector position or J5. We begin by creating a triangle by drawing a line between joints J3 and J4 that we are going to call H as in figure 8.1, creating two triangles used to find theta H and theta C as in figure 8.2.

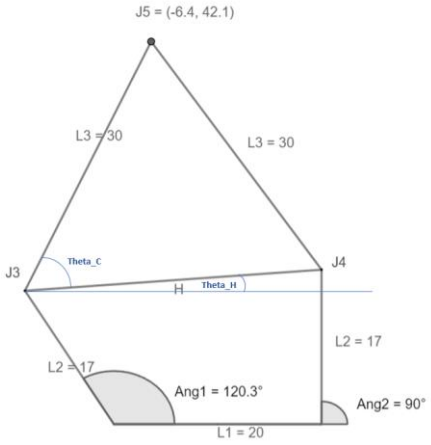


Figure 8.1. Segment H between passive joints J3 and J4 & angles needed to calculate EE position.

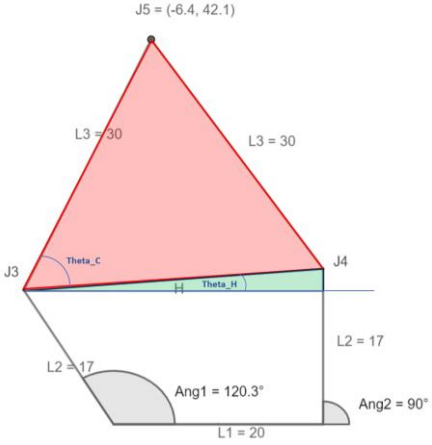


Figure 8.2. Green triangle used to find theta H, red triangle used to find theta C using cosine rule

From this we can calculate the angle of H from parallel to the x-axis using basic trigonometry:

$$\theta_H = \tan^{-1}\left(\frac{\Delta Y\ (between\ J3\ and\ J4)}{\Delta X\ (between\ J3\ and\ J4)}\right) \tag{5}$$

Then we need to calculate the angle theta C using the cosine rule:

$$\theta_C = \cos^{-1}\left(\frac{L3^2 + H^2 - L3^2}{2 * L3 * H}\right) \quad (6)$$

$$\theta_C = \cos^{-1}\left(\frac{H}{2 * L3}\right) \quad (7)$$

From this we can get the angle of J3 relative to the x-axis by adding Theta C and Theta H which we can then use to compute the position of the end-effect (or joint J5) [X, Y].

$$X_{EE} = X_{J3} + L3 * \cos(\theta_H + \theta_C) \quad (8)$$

$$Y_{EE} = Y_{J3} + L3 * \sin(\theta_H + \theta_C) \quad (9)$$

Whilst this approach works, we must ensure that this still meets the constraint of the links between J3 and end-effector are the same size as the link from J4 to end-effector. This is expressed in the 'ForwardKinematics.m' (Appendix 1.1) function snippet in figure 9.

```
% Compare link lengths from J3 to EE and J4 to EE
dist1 = norm(J3(1,:) - J5(1,:));
dist2 = norm(J4(1,:) - J5(1,:));
tolerance = 1e-6; % Define a tolerance level

%Checking Link J3 to EE and J4 to EE are same length
if abs(dist1-dist2) > tolerance
    disp('J3 to EE and J4 to EE are not the same.')
    % Flip angles to run ForwardKinematicsRight for corrected EE
    Left = 180 - Ang_J2; % New S.A left = 180 - S.A right
    Right = 180 - Ang_J1; % New S.A right = 180 - S.A left

    J5 = ForwardKinematicsRight([Left,Right],L);

else
    disp('J3 to EE and J4 to EE are the same.')
end
```

Figure 9. Snippet from 'ForwardKinematics.m' (Appendix 1.1) function checking end-effector position is possible and within model constraints.

Following the same geometric approach the 'ForwardKinematicsRight.m' function was created to return only the end-effector position but with negative X coordinate and the re-calculated input angles. This symmetrical geometry is showcased in the figures below.

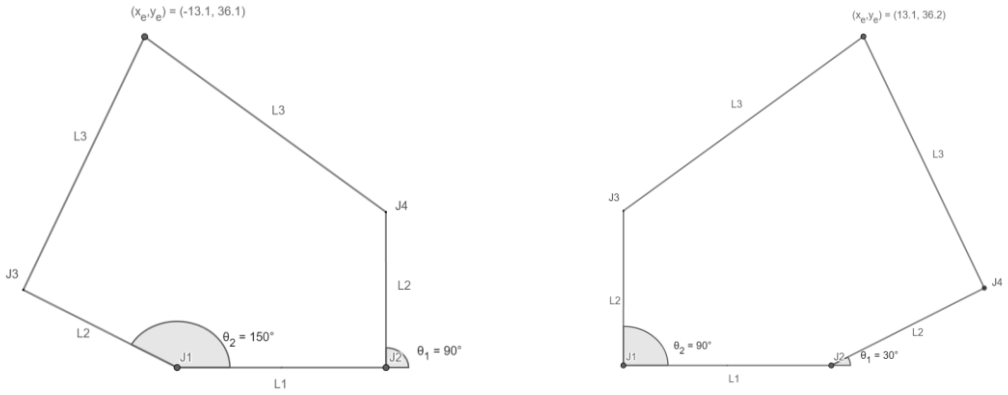


Figure 10. Justification of 'ForwardKinematics' and 'ForwardKinematicsRight' symmetry

2.2.1 Forward Kinematic Simulation Results

In verification of the derived geometric approach to solving forward kinematics, a task was created to obtain simulation results for six different input servo angle configurations as seen in the following figures.

Input Angles		Forward Kinematic Solution (EE position)	Figure
Left Servo Angle (θ_1) in degrees	**Right Servo Angle (θ_2) in degrees**		
30	30	(14.7 , 36.8)	11.1
30	45	(7.7 , 38.4)	11.2
120	90	(-6.4 , 42.2)	11.3
135	-30	(7.7 , 16.2)	11.4
-30	-45	(19.1 , 17.8)	11.5
120	-45	(11.5 , 16.1)	11.6

Table 1. Input servo angle combinations for forward kinematic simulated results

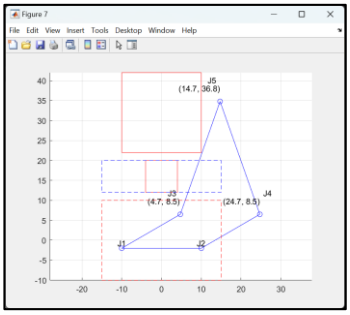


Figure 11.1. Simulation result input angles 1

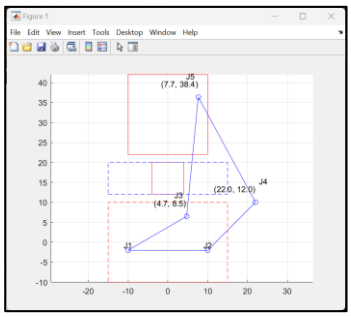


Figure 11.2. Simulation result input angles 2

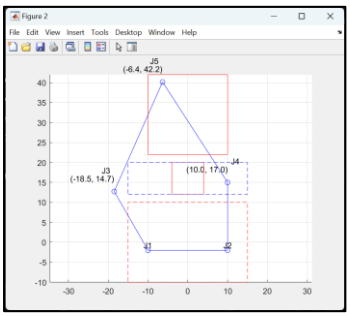


Figure 11.3. Simulation result input angles 3

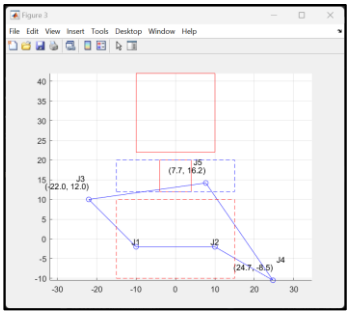


Figure 11.4. Simulation result input angles 4

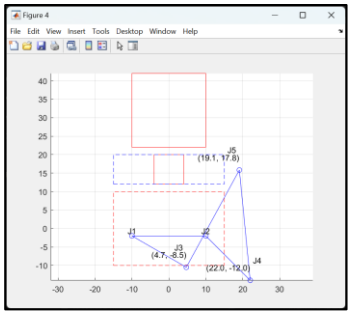


Figure 11.5. Simulation result input angles 5

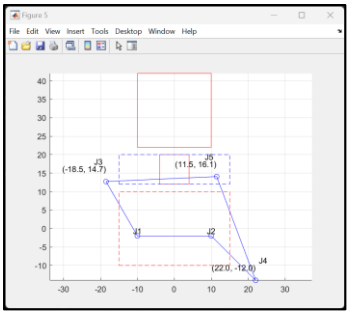


Figure 11.6. Simulation result input angles 6

2.3 Inverse Kinematics

The proposed solution to inverse kinematics uses a geometric approach to compute the combination of servo angles needed to reach a pre-defined end-effector position taking in only the end-effector position [X, Y] and the system link lengths (L1, L2 and L3).

The servo positions (joints J1 and J2) are derived in the same way as in the forward kinematics, equations 1 and 2.

To solve the inverse kinematic problem geometrically, the 5 bar-robot can be broken down into four individual triangles, two triangles used to calculate the angle of each servo. These triangles are derived from the hypotenuse length between the end-effector and each respective servo as depicted in figure 12.

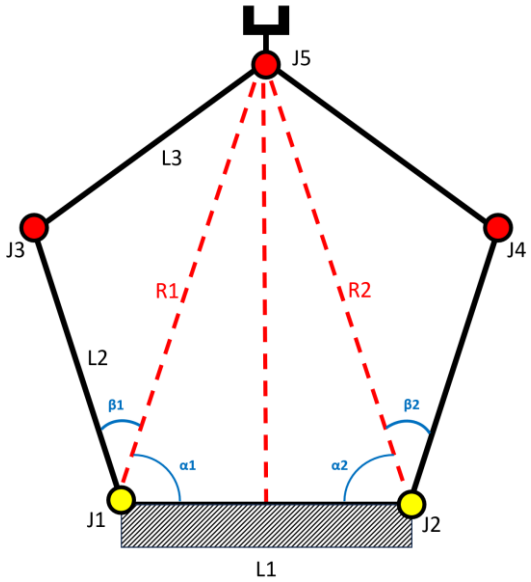


Figure 12. Illustration of geometric approach to solving inverse kinematic problem

As the positions of joints J5, J1 and J2 are known, we can use the Pythagorean theorem to calculate the length of R1 and R2 as below.

$$R1 = \sqrt{\left(\Delta X_{EE}\ to\ X_{J1}\right)^2 + \left(\Delta Y_{EE}\ to\ Y_{J1}\right)^2} \tag{10}$$

$$R2 = \sqrt{\left(\Delta X_{EE}\ to\ X_{J2}\right)^2 + \left(\Delta Y_{EE}\ to\ Y_{J2}\right)^2} \tag{11}$$

Using the MATLAB norm function we can get the absolute value of these hypotenuse lengths (see figure 13.).

```
14        % Hypotenuse from EE to Servos (R1 - LEFT servo, R2 - RIGHT servo)
15        R1 = norm([XE,YE] - Servos(1,:));
16        R2 = norm([XE,YE] - Servos(2,:));
```

Figure 13. Using MATLAB norm function to calculate R1 and R2 from 'InverseKinematics.m' function (Appendix 1.4)

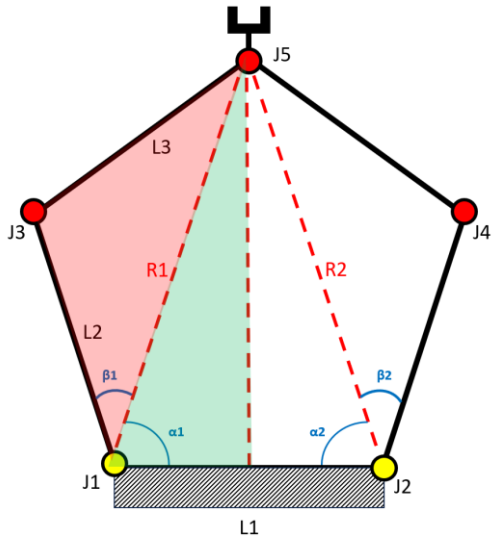


Figure 14. Triangles used to compute angles alpha and beta.

The green and red triangles on the diagram above are used to compute the angles alpha and beta respectively. For angle alpha, we can apply basic trigonometric principles using the inverse tangent of the opposite and adjacent sides as equation 12 – as the opposite is simply the Y end-effector coordinate and the adjacent side is the difference in the X coordinate of the end-effector and servo motor.

$$\alpha_1 = \tan^{-1}\left(\frac{Y_{EE}}{\Delta X_{EE}\ and\ X_{J1}}\right) \tag{12}$$

Angle beta is calculated using the cosine rule as per equation 13.

$$\beta_1 = \cos^{-1}\left(\frac{R1^2 + L2^2 - L3^2}{2 * R1 * L2}\right) \tag{13}$$

From these angles we can then calculate input servo angles one and two as shown below:

```
Ang1 = Alpha1 + Beta1;
Ang2 = 180 - Alpha2 - Beta2;
```

Figure 15. Computing the respective servo angles 1 and 2 in the '+ +' configuration

It is important to note that whilst this works in the '+ +' configuration it is not a solution for all possible robot configurations. This problem can be resolved through a series of conditional '*if*' statements to check which configuration the end-effector might be in such that the equation for servo angles can be set to the correct combination of angles alpha and beta.

2.3.1 Inverse Kinematics Simulation Circle and Square

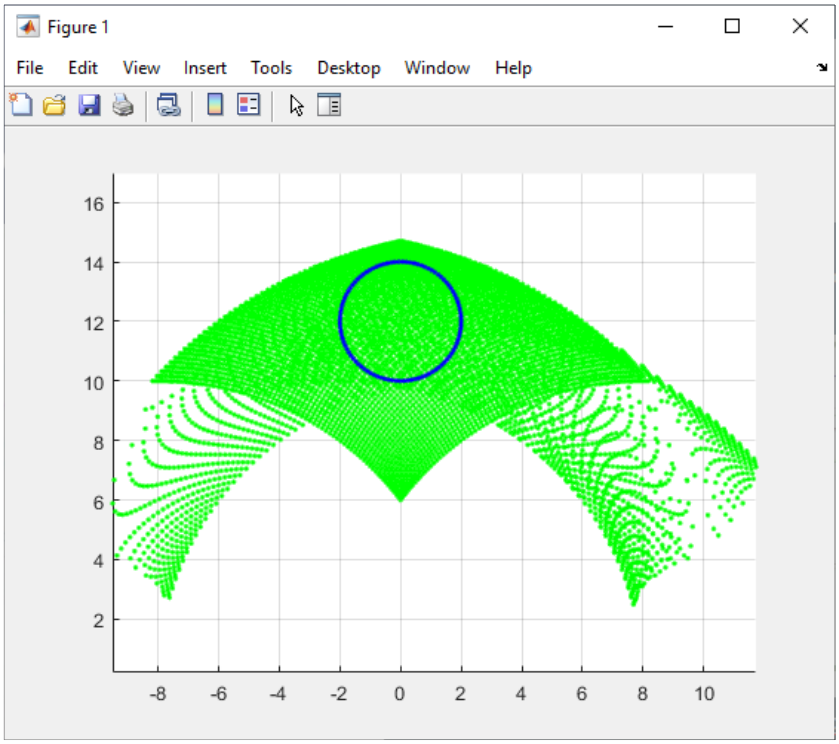


Figure 16. 4cm diameter circle plotted within workspace for live demonstration (created using CircleXY.m appendix A.1.5)

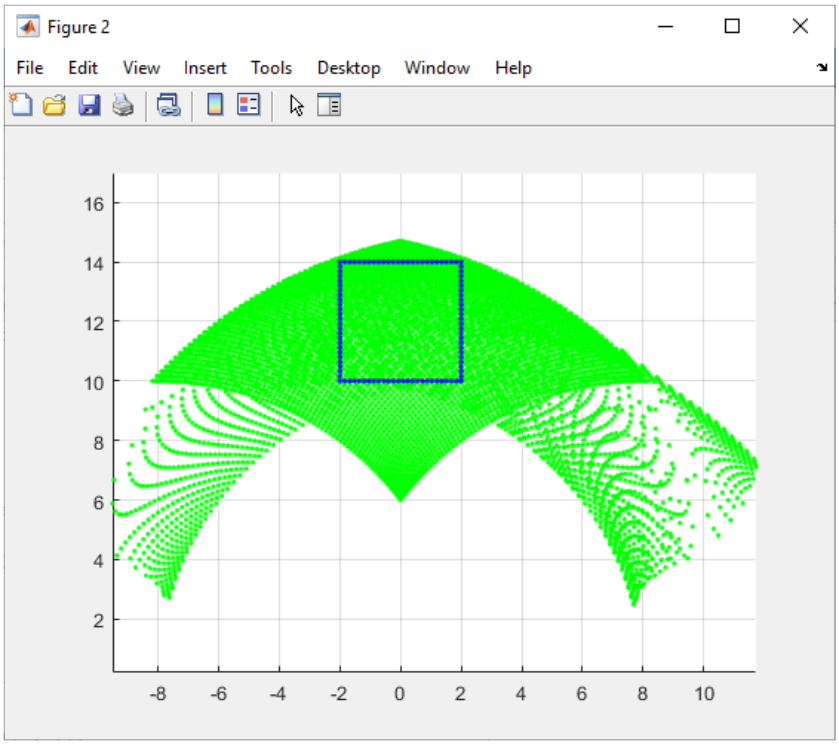


Figure 17. 4cm square plotted within workspace for live demonstration (created using SquareXY.m appendix A.1.6)

3. Robot Design, Fabrication and Assembly

3.1 Initial Robot Design – excluding end-effector

The first task to the robot design and assembly details the chosen approach to assembling the 5-bar planar parallel robot with the pre-determined link lengths as per section 1.1. It must be noted that due to the nature of the kinematic solutions, the links must be longer than the defined lengths as the link lengths are in fact measure from the pivotal axis of each individual joint which can be seen in the technical drawings below.

3.1.1 Design of Robot Links

Figure 18. Technical drawing of the proximal link (L2) measuring 17cm from centre of axis of rotation

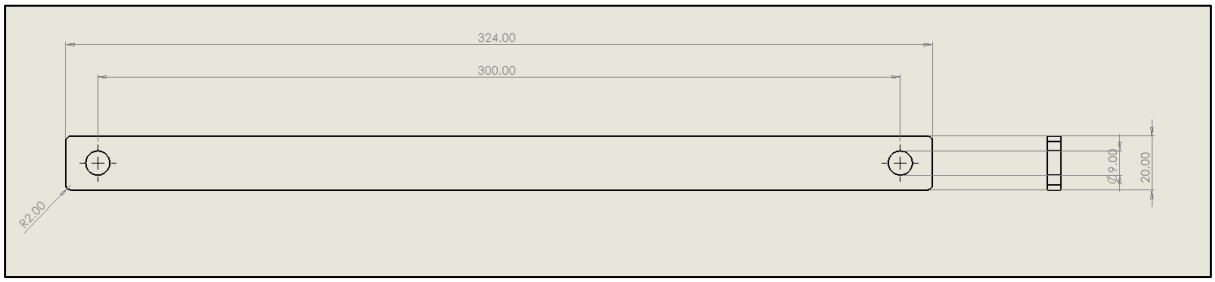


Figure 19. Technical drawing of the distal link (L3) measuring 30cm from centre of axis of rotation

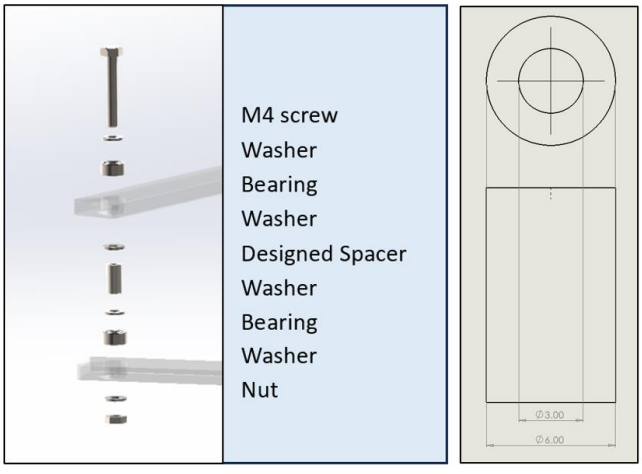


Figure 20. Exploded view of the joints connecting links and respective technical drawing of the designed spacer

3.1.2 Robot Assembly – excluding end-effector

As stated in the project specifications, the robot is to be mounted on a horizontal platform, and must be able to manoeuvre freely whilst also ensuring there is a designated space for a camera to be mounted for chocolate detection.

The following technical drawings in figures 21.1 and 21.2 illustrate the base plate design to house the servo motors as well as the Raspberry Pi 4 and a mount to attach the selected Raspberry Pi camera module 3 to detect oncoming chocolates.

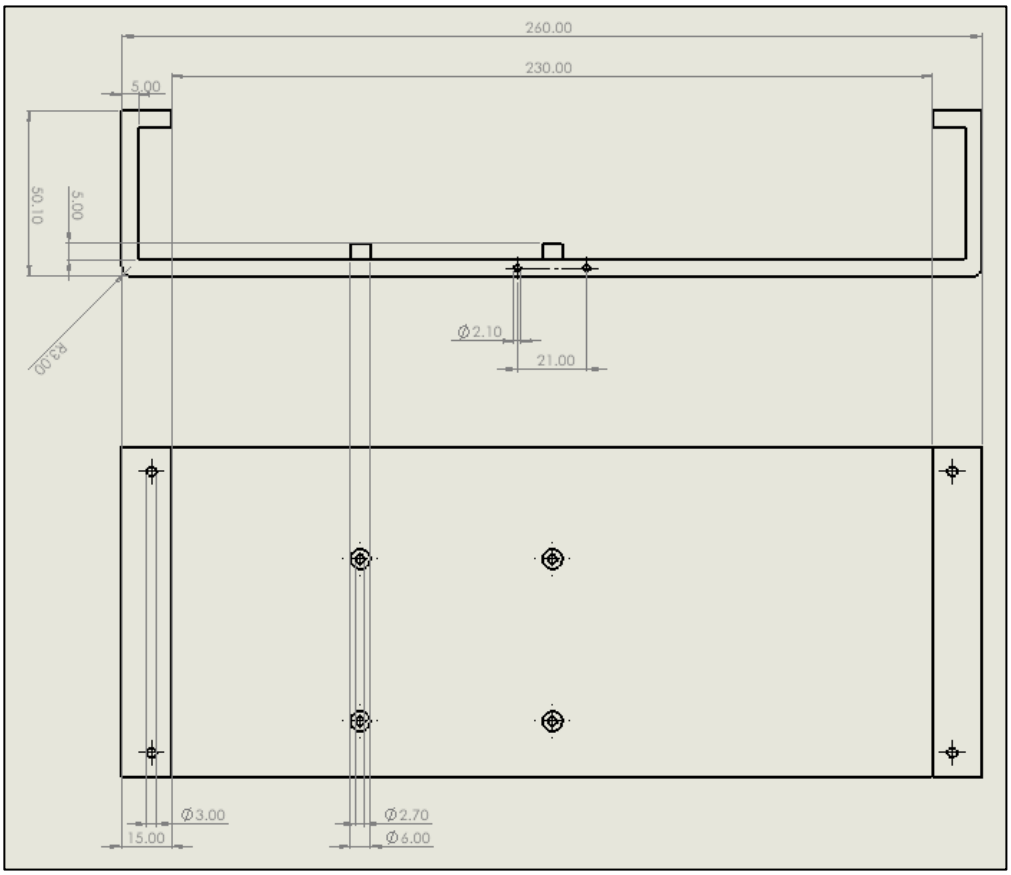


Figure 21.1. Technical drawing of the base plate for mounting servo motors and housing Raspberry Pi 4 and camera module

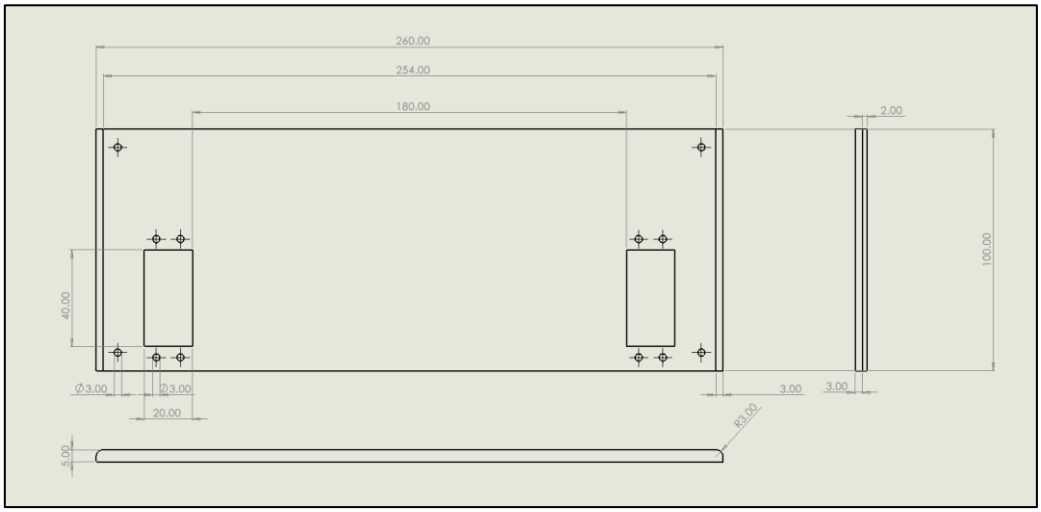


Figure 21.2. Technical drawing of the upper base plate casing

The selected Raspberry Pi 4 (4GB model) and Raspberry Pi camera SolidWorks models have been obtained from GrabCAD. The authors of the model designs are not affiliated with the company; however, the dimensions have been verified with self-owned boards and camera and are correct thus the assembly model effectively represents how the components can be mounted [1][2].

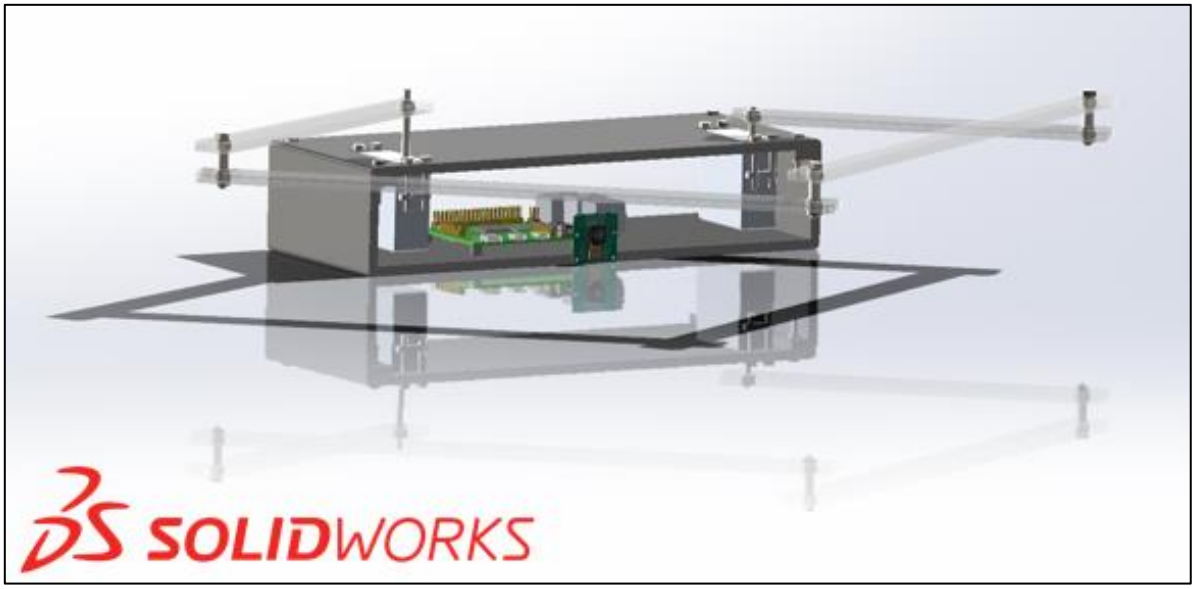


Figure 22. CAD assembly model of robotic design solution for 5-bar planar parallel robot excluding end-effector created using SOLIDWORKS.

3.2 Components Selection and Fabrication Costs

The following robotic design assembly of the links, mounting platform and servo motors aims to utilise readily available components to create a functional 5-bar chocolate picking and packing robot.

3.2.1 Mechanical Components

As a fabrication choice, we opted 3D printed links using PC/ABS (polycarbonate/ acrylonitrile-butadiene-styrene terpolymer blend) due to its increased processability and relative strength compared to standard ABS [3]. The material per kilogram cost of PC/ABS filament for 3D printing is £33.29 (excluding 20% VAT, all further pricings assume that VAT is excluded) [4], whilst assuming no printing faults, one roll of filament is sufficient to print the four links in the system.

To secure the servo motors to the base M3 x 40mm screws are used, a pack of 25 from Screwfix is £1.35 [5]. Similarly, M4 x 30mm screws are used to secure the links together, a pack of 25 from Screwfix is £2.23 [6]. Between each axis of rotation of the links SKF miniature ball bearings (open) are used (ERIKS item#: 618/4-SKF), in total there are 8 bearings – per unit the cost is £7.94, totalling £63.52 for 8 units [7]. To minimise the contact between joints and to protect the bearings when tightening the screws washers are required. M4 washers can be purchased from Screwfix in bulk, 100 units costs £2.87 [8], for the M3 screw contacts M3 washers are required, a pack of 100 costs £2.39 [9]. To separate the links, minimising the risk of collisions, spacers have been created using cast alloy steel – casting of the 5 spacer units, including the required machining was found using an online calculator, costing an average of £2.51 per piece, totalling £12.55 [10]. For the base plate and "lid" a standard PLA 1.75mm filament can be used for 3D printing – costing £14.39 per roll 1kg roll of filament. To ensure both parts are printed successfully 2 rolls of filament are required, costing a total of £28.78 [11]. Finally, both M3 and M4 nuts are required, which costs £3.03 [12] and £3.75 [13] for a pack of 100 respectively.

3.2.2 Electrical Components

To adjust the joint angles and operate the link, two TowerPro Servo Motors – MG996R (metal gear) units are used, per unit the cost is £7.20, totalling £14.40 [14]. A Raspberry Pi 4 (4GB) (£42.24 per unit [15]) has been selected in conjunction with a Raspberry Pi Camera Module 3 (wide angled model) (£26.88 per unit [16]) to enable machine vision. This is assuming that all necessary cables are provided with the components and the user has access to an external power supply. To mount the Raspberry Pi and camera 2mm slot screws are used, costing £3.10 per 100 units [17].

Material	Cost
PC/ABS filament	33.29
PLA filament	28.78
M3 x 40mm screws	1.35
M4 x 30mm screws	2.23
SKF miniature ball bearings	63.52
M4 washers	2.87
M3 washers	2.39
Raspberry Pi 4 (4GB)	42.24
Raspberry Pi Camera Module 3	19.2
Spacers	12.55
M2 slot screw	3.1
MG996R servo motors	£14.40
Total cost	£225.92

Figure 23. Full price list of components for 5-bar robot assembly and mounting platform excluding end-effector

4. End-effector Design and Analysis

The end-effector design solution selected was a four-fingered gripper design. Due to the robot operating in a food environment, the design must be sanitary and ensure that it obeys food health and safety guidelines. Therefore, the majority of the end-effector is fabricated from stainless steel and equipped with silicon rubber grippers at the tips used for picking up the chocolates.

To further avoid exerting unnecessary force on the chocolate and ensure they maintain their integrity, a strain gauge [Single Sensor CP 149] has been inserted into the gripping pads of the gripper fingers. This in turn allows the robot to sense the exerted force and make necessary adjustments to ensure chocolate integrity.

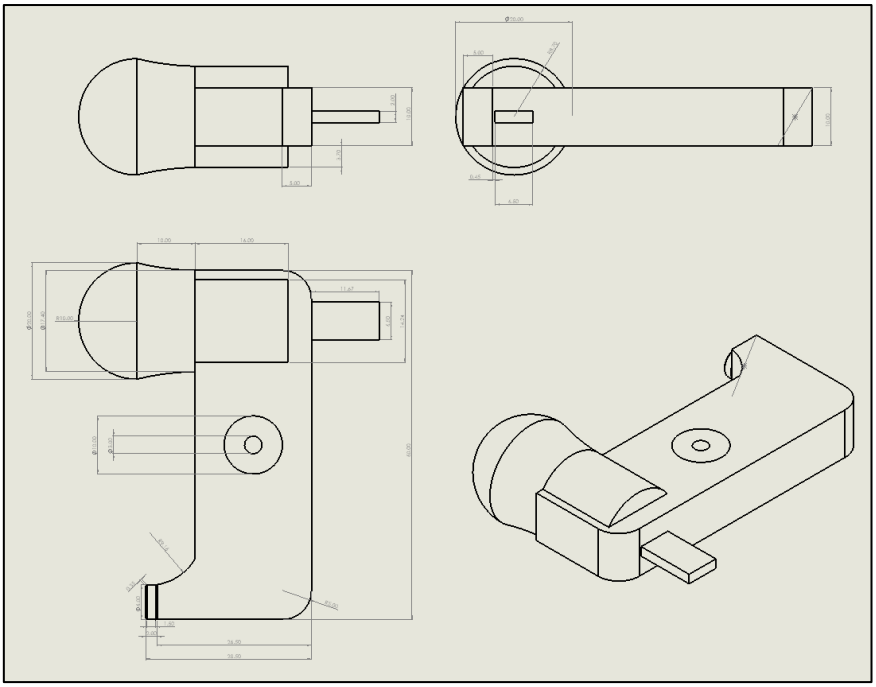


Figure 24. Technical drawing of end-effector gripper finger

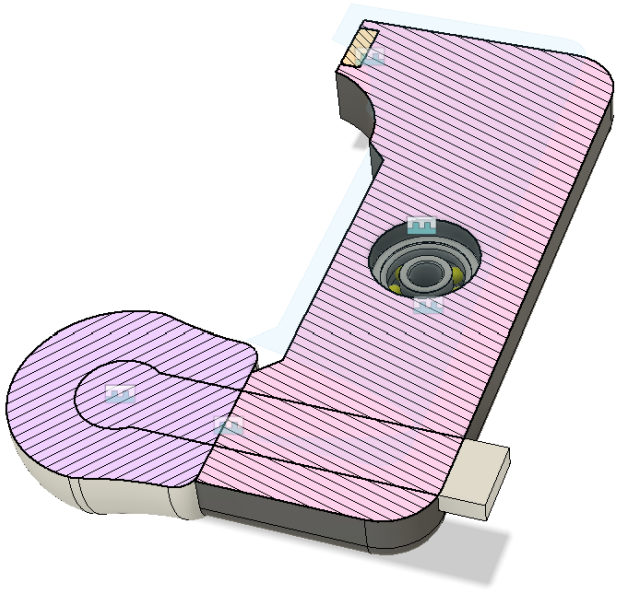


Figure 25. Cross section of individual gripper finger showing built-in strain gauge

The choice of using a gripper over pneumatics stemmed from the chocolates being gripped assumed to be soft, meaning being picked up with suction could easily damage the chocolate or have chocolate pieces enter the pneumatic system causing damage. Additionally, the pneumatic system may cause further constraints to the movement of the robot and would need to consider an efficient way to incorporate the pneumatic tubing without interfering with the robots range of motion.

The grippers are mounted by two small sealed deep groove ball bearings [SKF_623-Z] , these are pre-lubricated from the factory and sealed in to ensure no harmful fluids leak. The grippers are actuated by a centrally mounted electromagnet [Seed Studio Grove-Electromagnet] which magnetises neodymium magnets located at the end of the grippers, using the feedback from the strain gauges the magnet can then be controlled to change the output current to maximise effective gripping of the chocolates.

The end-effector is then mounted onto the 5-bar robot assembly through a pneumatic linear actuator. Pneumatics was a design choice made to ensure food safety, this choice also allows the robot to move in the Z direction, allowing it to avoid possible obstacles but also ensure the chocolates are gently placed and to eliminate the risk of it being dragged by the conveyor.

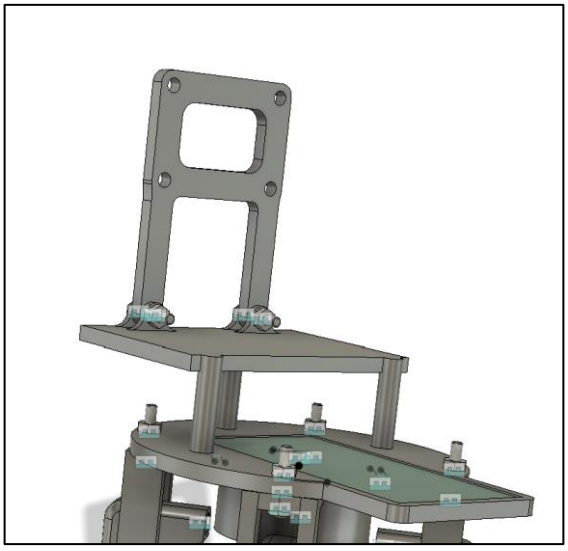


Figure 26. CAD model showing electromagnet mounted at the centre of end-effector assembly

Figure 27. CAD model showing linear pneumatic actuator mount for end-effector

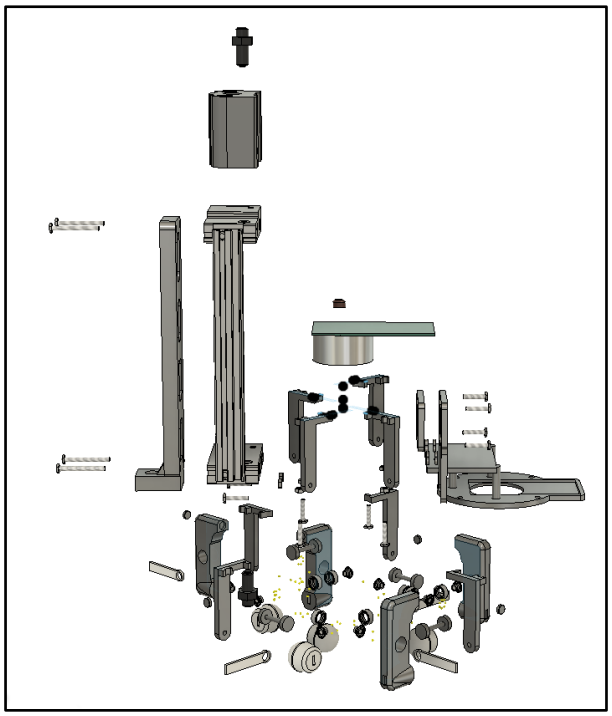


Figure 28. Exploded view of final end-effector and carriage design

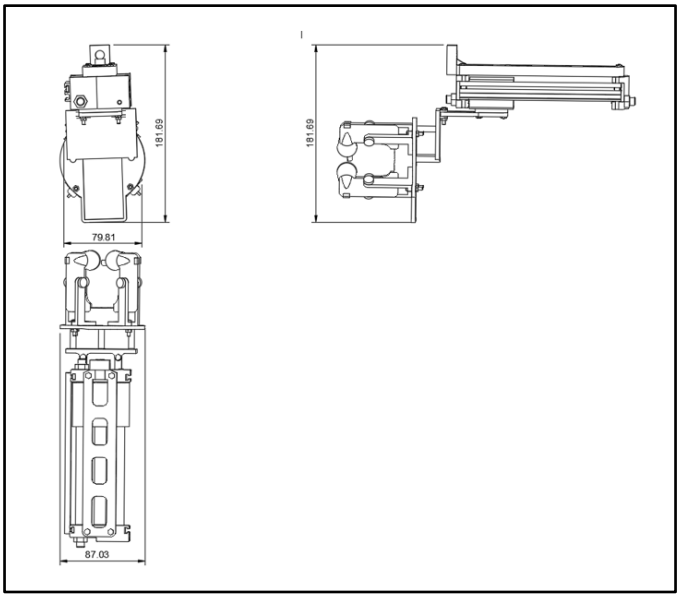


Figure 29. Technical drawing of end-effector and carriage assembly

4.1 Full Robot Design Assembly

When attaching the end-effector to the 5-bar robot linkages, we are able to rigidly fix the end-effector meaning the offset in the Y domain remains constant and is therefore negligible as we can simply subtract it from the end-effector position needed in inverse kinematics.

To support the large height of the electromagnetic gripper and pneumatic actuator mount, a second 5-bar mechanism has been attached above the original one using lead screws encased by a crossbar at the top to enforce the system's rigidity.

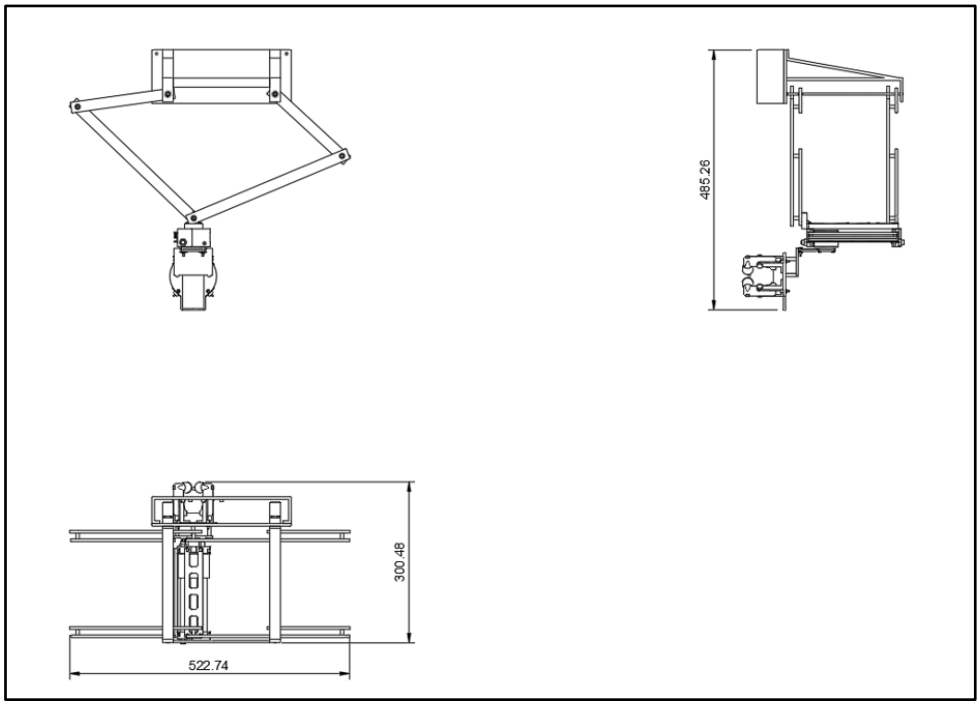


Figure 30. Technical drawing of complete robot assembly including end-effector

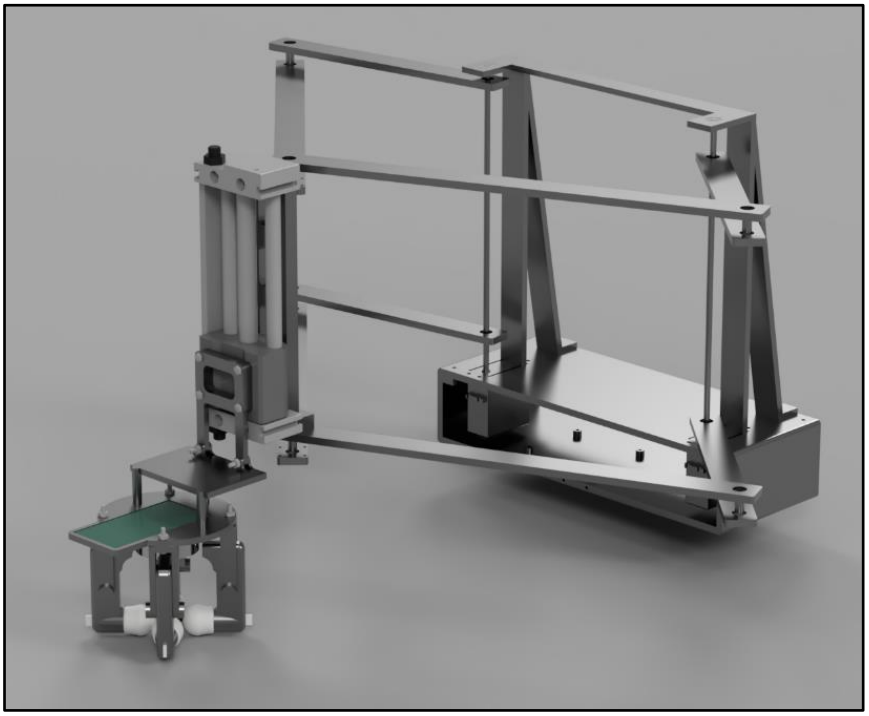


Figure 31. Complete render of the finalised robot assembly including end-effector design

4.2 Components Selection and Fabrication Costs

The following robotic design assembly of the links, mounting platform and servo motors aims to utilise readily available components to create a functional 5-bar chocolate picking and packing robot.

4.2.1 Mechanical Components

For the end-effector, the three different bearings are manufactured by SKF and are differing sizes depending on where they are being used to reduce friction in revolving components. The 2 619/4 2RS1 [19] is used to connect the lead screws to the frame for extra rigidity and the SKF 623 Z [23] is used to connect the fingers to the mount. Various fixtures are used to construct the assembly, these are mostly M3 bolts [20] [21], as the assembly is small, having a larger bolt would add unnecessary stresses into the system and could cause premature failure. Shipped in large, packed quantities they are a strong cost-effective fixing solution.

The rod-less pneumatic actuator [22] used to control the z direction of the end effector, has been chosen to have a large stroke, allowing for application flexibility. Finally, neodymium magnets [2] allow the actuation of the fingers in the grippers depending on the electromagnet, and are simply glued into place.

4.2.2 Electrical Components

For the end-effector, The low-profile strain gauge [24] is inserted into the gripper pads will be glued into the fingertips, which will be glued onto the fingers using a strong adhesive to keep the design compact. The electromagnet [26] is small and fits perfectly into the assembly, costing £10.56.

Material [Pack Quant]	Cost	Quant	Total cost
SKF W 619/4 2RS1	£ 14.53	2	£ 29.06
M3x30mm bolt [50]	£ 4.99	1	£ 4.99
M3x15 [20]	£ 3.62	1	£ 3.62
SMC Rodless Actuator	£ 421.90	1	£ 421.90
IEE low profile strain guage	£ 22.98	4	£ 91.92
SKF 623 Z	£ 3.24	8	£ 25.92
Seeed Studio Electromagnet	£ 10.56	1	£ 10.56
Neodymium magnet [100]	£ 7.20	1	£ 7.20
Total:			£ 595.17

Figure 32. Full price list of components for designed electromagnetic gripper end-effector

5. Implementation of Robotic Solution

In the implementation of this robotic solution we can combine all of the before mentioned solutions to implement these into the chocolate picking and packing environment.

This section of the report details the chocolate detection approach applied to this solution. Firstly, prior to beginning the process, we must begin by capturing images of the different chocolate colours and saving these to retrieve their respective histograms.

From these RGB histograms (see example histogram figure 33.), we are able to record the thresholding RGB value combinations to isolate each individual chocolate colour.

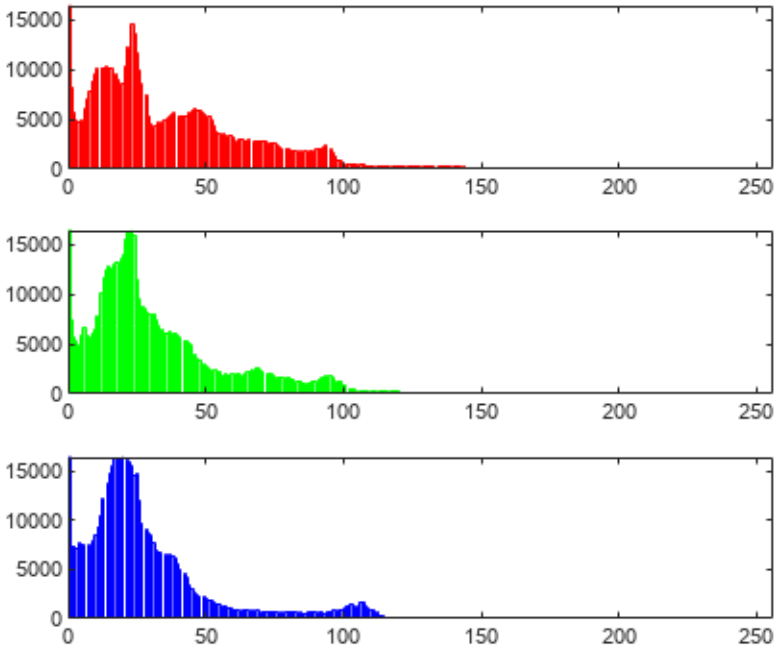


Figure 33. Example of MATLAB RGB histogram for thresholding

Once these individual thresholding values stored we can perform colour segmentation. This can be done by creating an image mask of thresholding values for each respective colour to be applied in real time as the chocolates are being fed in the conveyor belt system. This mask will detect the chosen colour making it appear white as seen in figure XX.

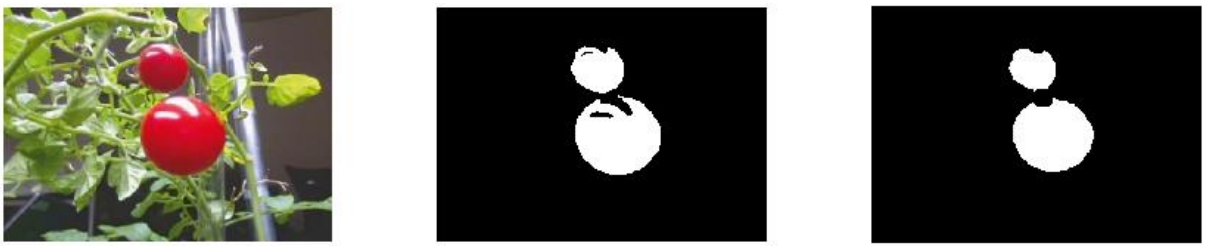


Figure 34. Colour segmentation using thresholding mask and applying post-processing to eliminate anomalies

In reflection, we encounter small issues regarding predicting the oncoming chocolate. Given the nature of applying a thresholding mask, there is an issue of knowing which mask needs to be applied. One solution that proves to be effective is the following: assuming the system fills the tray to paint the image line by line from left to right, we can assume that it already knows exactly which colour is to be placed in each individual dimple in the tray. With this in mind, rather than dealing with each chocolate as they come, we can set the system to filter for the next needed colour to fill the tray, meaning it would skip chocolates until the desired colour is detected and then pick it up and place it in the designate position.

Whilst this may not be the fastest or most efficient way to perform this task, it is the only plausible way to ensure it remains effective and efficient minimising the risk of error. This could be improved upon by considering implementing multiple cameras such as to apply multiple colour segmenting masks and act based upon the feed from the camera that successfully detects the oncoming chocolate.

6. Conclusion

In conclusion, this development of a robotic solution for the personalized chocolate box gift manufacturing details an advancement in the automation of food customisation. The design and implementation of the 5-bar planar parallel robot, equipped with a sophisticated end-effector and high-resolution machine vision, have successfully addressed the complex requirements of transforming customer drawings into chocolate arrangements. This project demonstrates the feasibility of precise, automated chocolate placement based on customer specifications.

The successful integration of kinematic and mechanical designs with advanced sensors and imaging technology has proven effective in ensuring the accuracy and delicacy required for handling food products. Moreover, the ability to dynamically adjust to various chocolate types and colours via real-time image processing illustrates the robustness and flexibility of the system.

Future enhancements could focus on optimizing the system's efficiency and speed, possibly by incorporating machine learning algorithms to predict and adapt to the incoming chocolate types more swiftly. Additionally, scaling the technology to handle larger volumes and varying product types could open new avenues in personalized food manufacturing. This project serves as a compelling example of how robotics can be intricately woven into the fabric of creative industries, providing both functional solutions and sparking innovation.

References

[1] Shuja, H. (2019) Raspberry Pi 4 Model B, Free CAD Designs, Files & 3D Models | The GrabCAD Community Library. Available at: https://grabcad.com/library/raspberry-pi-4-model-b-1 (Accessed: 24 April 2024).

[2] Lavirotte, S. (2016) Raspberry Pi Camera, Free CAD Designs, Files & 3D Models | The GrabCAD Community Library. Available at: https://grabcad.com/library/raspberry-pi-4-model-b-1 (Accessed: 24 April 2024).

[3] PC/abs resin: Polycarbonate ABS blend - by polymer resources (2023) PC/ABS RESIN. Available at: https://prlresins.com/products/pc-abs-resin/#:~:text=PC%2FABS%20(polycarbonate%20%2F%20acrylonitrile,provides%20non%2Dhalo%20flame%20retardancy (Accessed: 24 April 2024).

[4] 3D-printers: Kits: Parts: Filament (no date) 123-3d. Available at: https://www.123-3d.co.uk/Polymaker-black-PC-ABS-filament-1-75mm-1kg-70256-PMPM-1006-001-i7511-t104854.html (Accessed: 24 April 2024).

[5] Easyfix bright zinc-plated pan machine screws M3 x 20mm 25 pack - screwfix (no date) Screwfix.com. Available at: https://www.screwfix.com/p/easyfix-bright-zinc-plated-pan-machine-screws-m3-x-20mm-25-pack/5875j (Accessed: 24 April 2024).

[6] Easyfix bright zinc-plated pan machine screws M4 x 30mm 25 pack - screwfix (no date) Screwfix.com. Available at: https://www.screwfix.com/p/easyfix-bright-zinc-plated-pan-machine-screws-m4-x-30mm-25-pack/3908j (Accessed: 24 April 2024).

[7] SKF miniature ball bearing open (no date) shop.eriks.co.uk. Available at: https://shop.eriks.co.uk/en/bearings-ball-bearings-deep-groove-ball-bearings/miniature-ball-bearing-steel-open-618-4-618-4-skf/ (Accessed: 24 April 2024).

[8] Easyfix steel large flat washers M4 X 1mm 100 pack - screwfix (no date) Screwfix.com. Available at: https://www.screwfix.com/p/easyfix-steel-large-flat-washers-m4-x-1mm-100-pack/194ft (Accessed: 24 April 2024).

[9] Easyfix steel large flat washers M3 x 0.8mm 100 pack - screwfix (no date) Screwfix.com. Available at: https://www.screwfix.com/p/easyfix-steel-large-flat-washers-m3-x-0-8mm-100-pack/656ft (Accessed: 24 April 2024).

[10] Cast steel price calculator - price estimation, price evaluation for Carbon Steel, alloy steel (no date) Iron-foundry.com. Available at: https://www.iron-foundry.com/cast-steel-price-calculator.html (Accessed: 24 April 2024).

[11] PLA 1.75mm - 3D printing filament (no date) 3Dprintz.co.uk. Available at: https://3dprintz.co.uk/collections/pla?gad_source=1&gclid=EAIaIQobChMI__jIr5HbhQMVCZRQBh2ACwDzEAAYASAAEgKSZ_D_BwE (Accessed: 24 April 2024).

[12] Easyfix A2 Stainless Steel Hex Nuts M3 100 pack - screwfix (no date) Screwfix.com. Available at: https://www.screwfix.com/p/easyfix-a2-stainless-steel-hex-nuts-m3-100-pack/1184t (Accessed: 24 April 2024).

[13] Easyfix A2 stainless steel nylon lock nuts M4 100 Pack - Screwfix (no date) Screwfix.com. Available at: https://www.screwfix.com/p/easyfix-a2-stainless-steel-hex-nuts-m3-100-pack/1184thttps:/www.screwfix.com/p/easyfix-a2-stainless-steel-nylon-lock-nuts-m4-100-pack/2340t (Accessed: 24 April 2024).

[14] TowerPro servo motor - MG996R (Metal Gear) (no date) Thepihut.com. Available at: https://thepihut.com/products/servo-motor-mg996r-high-torque-metal-gear (Accessed: 24 April 2024).

[15] Raspberry pi 4 model B (no date) Thepihut.com. Available at: https://thepihut.com/products/raspberry-pi-4-model-b?variant=20064052740158&src=raspberrypi (Accessed: 24 April 2024).

[16] Raspberry pi camera module 3 (no date) Thepihut.com. Available at: https://thepihut.com/products/raspberry-pi-camera-module-

3?variant=42305752072387&currency=GBP&utm_medium=product_sync&utm_source=google&utm_content=sag_organic&utm_campaign=sag_organic&gad_source=1&gclid=EAIaIQobChMItNbY_YnbhQMVD5VQBh30ZQLjEAQYASABEgJlwvD_BwE (Accessed: 24 April 2024).

[17] RS pro slot pan A2 304 stainless steel machine screws DIN 85, M2X5MM (no date) RS-online.com. Available at: https://uk.rs-online.com/web/p/machine-screws/9141563 (Accessed: 24 April 2024).

[18] “PARTcommunity 2D/3D Models,” autodesk-fusion.partcommunity.com. https://autodesk-fusion.partcommunity.com/3d-cad-models/sso/?cwid=1689 (accessed Apr. 24, 2024).

[19] “623-2Z SKF Shielded Deep Groove Ball Bearing 3x10x4mm SKF Deep Groove Ball Bearings - Bearing King,” www.bearing-king.co.uk. https://www.bearing-king.co.uk/bearing/623-2z-skf-shielded-deep-groove-ball-bearing-3x10x4mm/3561?gad_source=1&gclid=CjwKCAjw26KxBhBDEiwAu6KXtxTU1y5vJaUfzbO4HoWynte_EIQjEgqavu8IehE23lX_lGYha9QmgBoCqa8QAvD_BwE (accessed Apr. 24, 2024).

[20] “DTGN M3-0.5 x 30mm Flat Head Socket Cap Screws, 50 Pack 304 Stainless Steel Hex Socket Flat Head Screws, Full Thread, Countersunk Bolts : Amazon.co.uk: DIY & Tools,” www.amazon.co.uk. https://amzn.eu/d/arU2XLN (accessed Apr. 24, 2024).

[21] “(3mm) M3 x 15 Black 12.9 Grade High Tensile Hex Socket Cap Screws Self Colour Allen Bolts DIN 912-20 Pack : Amazon.co.uk: DIY & Tools,” www.amazon.co.uk. https://amzn.eu/d/a0Jv6vY (accessed Apr. 24, 2024).

[22] “CY1SG10-100Z | SMC Rodless Actuator 100mm Stroke, 10mm Bore | RS,” uk.rs-online.com. https://uk.rs-online.com/web/p/pneumatic-rodless-cylinders/8852234?gb=s (accessed Apr. 24, 2024).

[23] “SKF 619/6-2Z Single Row Deep Groove Ball Bearing- Both Sides Shielded 6mm I.D, 15mm O.D | RS,” uk.rs-online.com. https://uk.rs-online.com/web/p/ball-bearings/2077029?cm_mmc=UK-PLA-DS3A-_-google-_-CSS_UK_EN_PMAX_Catch+All-_--_-2077029&matchtype=&&gad_source=1&gclid=CjwKCAjw26KxBhBDEiwAu6KXtwtAfd4294xzVlq8gCXmcRU1qra3IbMv93eimeeY-5Cz6KjNWVCBAhoCR9sQAvD_BwE&gclsrc=aw.ds (accessed Apr. 24, 2024).

[24] “SS-U-N-S-00001 | I.E.E. Low Profile Strain Gauge, Compression, Tension Measure, >1MΩ | RS,” uk.rs-online.com. https://uk.rs-online.com/web/p/strain-gauges/1895556?gb=s (accessed Apr. 24, 2024).

[25] “JIANTA 130 Pack Small Magnets, 6x2mm, Refrigerator Mini Magnets, Round Fridge Magnets, Strong Small Neodymium Magnets for Crafts, Whiteboard, Kitchen Cabinet : Amazon.co.uk: Business, Industry & Science,” www.amazon.co.uk. https://amzn.eu/d/ekJCtfL (accessed Apr. 24, 2024).

[26] “101020073 | Seeed Studio Grove-Electromagnet | RS,” uk.rs-online.com. https://uk.rs-online.com/web/p/power-motor-robotics-development-tools/1743254?gb=s (accessed Apr. 24, 2024).

[27] “Vector and matrix norms - MATLAB norm - MathWorks United Kingdom,” uk.mathworks.com. https://uk.mathworks.com/help/matlab/ref/norm.html

Appendix

A.1 MATLAB Scripts

A.1.1 'ForwardKinematics.m' MATLAB Script

InverseKinematics.m | Main.m | ForwardKinematics.m

```
% Forward Kinematics Function for 5 Bar Planar Parallel Robot

%Returns positions of each of the 5 joints
function [J1, J2, J3, J4, J5] = ForwardKinematics(Servo ,L)

    %Servo Angles (S.A) of Joints J1 and J2
    Ang_J1 = Servo(1,1);
    Ang_J2 = Servo(1,2);

    % Servo positions (J1 and J2) [X,Y]
    J1 = [-L(1)/2, 0];
    J2 = [L(1)/2, 0];

    % Compute passive joints J3 and J4
    J3 = [J1(1,1) + L(2) * cosd(Ang_J1),    J1(1,2) + L(2) * sind(Ang_J1)];
    J4 = [J2(1,1) + L(2) * cosd(Ang_J2),    J2(1,2) + L(2) * sind(Ang_J2)];

    % Compute distance between passive joints (J3 and J4) = H
    % Angle of H (equation 1 in PDF)
    H = norm(J3 - J4);
    Theta_H = atand(norm(J4(1,2) - J3(1,2))/ norm(J4(1,1) - J3(1,1)));

    % Using Cosine rule to find Angle from H to EE (equation 3 in PDF)
    Theta_C = acosd(H / (2 * L(3)));

    % Angle of Joint J3 relative to x-axis:
    Ang_J3 = Theta_H + Theta_C;

    % Compute EE position:
    J5 = [J3(1,1) + L(3) * cosd(Ang_J3),    J3(1,2) + L(3) * sind(Ang_J3)];

    % Compare link lengths from J3 to EE and J4 to EE
    dist1 = norm(J3(1,:) - J5(1,:));
    dist2 = norm(J4(1,:) - J5(1,:));
    tolerance = 1e-6; % Define a tolerance level

    %Checking Link J3 to EE and J4 to EE are same length
    if abs(dist1-dist2) > tolerance
        disp('J3 to EE and J4 to EE are not the same.')
        % Flip angles to run ForwardKinematicsRight for corrected EE
        Left = 180 - Ang_J2; % New S.A left = 180 - S.A right
        Right = 180 - Ang_J1; % New S.A right = 180 - S.A left

        J5 = ForwardKinematicsRight([Left,Right],L);

    else
        disp('J3 to EE and J4 to EE are the same.')
    end
```

A.1.2 'ForwardKinematicsRight.m' MATLAB Script

InverseKinematics.m | Main.m | ForwardKinematics.m | ForwardKinematicsRight.m

```
% Forward Kinematics Function for 5 Bar Planar Parallel Robot

%Returns positions of ONLY EE
function [J5] = ForwardKinematicsRight(Servo ,L)

    %Servo Angles of Joints J1 and J2
    Ang_J1 = Servo(1,1);
    Ang_J2 = Servo(1,2);

    % Servo positions (J1 and J2) X,Y
    J1 = [-L(1)/2, 0];
    J2 = [L(1)/2, 0];

    % Compute passive joints J3 and J4
    J3 = [J1(1,1) + L(2) * cosd(Ang_J1),    J1(1,2) + L(2) * sind(Ang_J1)];
    J4 = [J2(1,1) + L(2) * cosd(Ang_J2),    J2(1,2) + L(2) * sind(Ang_J2)];

    % Compute distance between passive joints (J3 and J4) = H
    % Angle of H (equation 1 in PDF)
    H = norm(J3 - J4);
    Theta_H = atand(norm(J4(1,2) - J3(1,2))/ norm(J4(1,1) - J3(1,1)));

    % Using Cosine rule to find Angle from H to EE (equation 3 in PDF)
    Theta_C = acosd(H / (2 * L(3)));

    % Angle of Joint J3:
    Ang_J3 = Theta_H + Theta_C;

    % Compute EE position: (Negative of the X component)
    J5 = [-(J3(1,1) + L(3) * cosd(Ang_J3)), J3(1,2) + L(3) * sind(Ang_J3)];
```

A.1.3 'PlotLinks.m' MATLAB Script

InverseKinematics.m | Main.m | ForwardKinematics.m | ForwardKinematicsRight.m | PlotLinks.m

```
function PlotLinks(Joints, workspace)

    figure;
    grid on;
    xlabel=('X');
    ylabel=('Y');
    axis equal;
    hold on;

    oY = 2;
    labelOffset = 2;

    % Plotting Workspace
    plot(workspace(:,1),workspace(:,2),'LineStyle','None','Marker','.','Color','g');


    % Plot Robot (links and joint coordinates)
    % Links
    plot([Joints(1,1) Joints(2,1)], [Joints(1,2) Joints(2,2)]-oY, 'b-o');
    plot([Joints(1,1) Joints(3,1)], [Joints(1,2) Joints(3,2)]-oY, 'b-o');
    plot([Joints(2,1) Joints(4,1)], [Joints(2,2) Joints(4,2)]-oY, 'b-o');
    plot([Joints(3,1) Joints(5,1)], [Joints(3,2) Joints(5,2)]-oY, 'b-o');
    plot([Joints(4,1) Joints(5,1)], [Joints(4,2) Joints(5,2)]-oY, 'b-o');

    % Joint Labels
    text(Joints(1,1), Joints(1,2) - labelOffset, 'J1', 'VerticalAlignment','bottom','HorizontalAlignment','center');
    text(Joints(2,1), Joints(2,2) - labelOffset, 'J2', 'VerticalAlignment','bottom','HorizontalAlignment','center');
    text(Joints(3,1) - labelOffset, Joints(3,2) + labelOffset, 'J3', 'VerticalAlignment','bottom','HorizontalAlignment','center');
    text(Joints(4,1) + labelOffset, Joints(4,2) + labelOffset, 'J4', 'VerticalAlignment','bottom','HorizontalAlignment','center');
    text(Joints(5,1) - labelOffset, Joints(5,2) + labelOffset, 'J5', 'VerticalAlignment','bottom','HorizontalAlignment','center');

    % Coordinates
    text(Joints(3,1), Joints(3,2), sprintf('(%.1f, %.1f)', Joints(3,1), Joints(3,2)), 'VerticalAlignment','bottom','HorizontalAlignment','right');
    text(Joints(4,1), Joints(4,2), sprintf('(%.1f, %.1f)', Joints(4,1), Joints(4,2)), 'VerticalAlignment','bottom','HorizontalAlignment','right');
    text(Joints(5,1), Joints(5,2), sprintf('(%.1f, %.1f)', Joints(5,1), Joints(5,2)), 'VerticalAlignment','bottom','HorizontalAlignment','right');

    % Plotting Environment
    rectangle('Position',[-15,-10,30,20],'EdgeColor','r','LineWidth',0.5,'LineStyle','--'); % Area for mounting robot
    rectangle('Position',[-4,12,8,8], 'EdgeColor','r','LineWidth',0.5,'LineStyle','-'); % Chocolate pickup region
    rectangle('Position',[-15,12,30,8], 'EdgeColor','b','LineWidth',0.25,'LineStyle','--'); % Conveyor
    rectangle('Position',[-10,22,20,20], 'EdgeColor','r','LineWidth',0.5,'LineStyle','-'); % Chocolate tray
```

A.1.4 ‘InverseKinematics.m’ MATLAB Script

InverseKinematics.m | Main.m | ForwardKinematics.m | ForwardKinematicsRight.m

```
% Inverse Kinematics Function for 5 Bar Planar Parallel Robot

% Returns positions of each of the 5 joints
function [Ang1, Ang2] = InverseKinematics(EE ,L)

    % Servo Positions: J1 and J2
    Servos = [-L(1,1)/2,    0;
               L(1,1)/2,    0];

    % EE positions
    XE = EE(1,1);
    YE = EE(1,2);

    % Hypotenuse from EE to Servos (R1 - LEFT servo, R2 - RIGHT servo)
    R1 = norm([XE,YE] - Servos(1,:));
    R2 = norm([XE,YE] - Servos(2,:));

    % Computing Servo 1 Angle (LEFT SERVO): Ang1
    % Alpha1 and Beta1 as per diagram
    Alpha1 = atand(YE/norm(XE - Servos(1,1)));
    Beta1 = acosd((R1^2 + L(1,2)^2 - L(1,3)^2)/(2 * R1 * L(1,2)));

    Ang1 = Alpha1 + Beta1;

    % Computing Servo 2 Angle (RIGHT SERVO): Ang2
    % Alpha2 and Beta2 as per diagram
    Alpha2 = atand(YE/norm(XE - Servos(2,1)));
    Beta2 = acosd((R2^2 + L(1,2)^2 - L(1,3)^2)/(2 * R2 * L(1,2)));

    Ang2 = 180 - Alpha2 - Beta2;

    Ang1 = round(Ang1,1);
    Ang2 = round(Ang2,1);
```

A.1.5 ‘CircleXY.m’ MATLAB Script

CircleXY.m | InverseKinematics.m | Main.m | ForwardKinematics.m

```
% Circle EE Positions

function [X,Y] = CircleXY(Center, radius, n)

Circle_Angles = linspace(1,360,n);

% Computing all end-effector positions for circle points
X = radius * cosd(Circle_Angles) + Center(1,1);
Y = radius * sind(Circle_Angles) + Center(1,2);
```

A.1.6 'SquareXY.m' MATLAB Script

SquareXY.m | CircleXY.m | InverseKinematics.m | Main.m | ForwardKinematics.m

```
% Square EE Positions

function [EE] = SquareXY(Center, radius, n)

% Calculate number of points per side, ensuring corners are included once
ppe = n / 4; % Points per edge

% Define corners
TL = [Center(1,1) - radius, Center(1,2) + radius]; % top left
TR = [Center(1,1) + radius, Center(1,2) + radius]; % top right
BL = [Center(1,1) - radius, Center(1,2) - radius]; % bottom left
BR = [Center(1,1) + radius, Center(1,2) - radius]; % bottom right

% Points on each edge
TE = [linspace(TL(1), TR(1), ppe + 1)', repmat(TR(2), ppe + 1, 1)];%top
RE = [repmat(BR(1), ppe + 1, 1), linspace(TR(2), BR(2), ppe + 1)'];%right
BE = [linspace(BR(1), BL(1), ppe + 1)', repmat(BL(2), ppe + 1, 1)];%bottom
LE = [repmat(BL(1), ppe + 1, 1), linspace(BL(2), TL(2), ppe + 1)'];%left

% Combined edges removing last point of each edge to avoid duplicates
EE = [TE; RE(1:end-1, :); BE(1:end-1, :); LE(1:end-1, :)];
```

A.1.7 'AngleMapping.m' MATLAB Script

AngleMapping.m | SquareXY.m | Circle

```
function x = AngleMapping(input)
x=input*1/180;
end
```

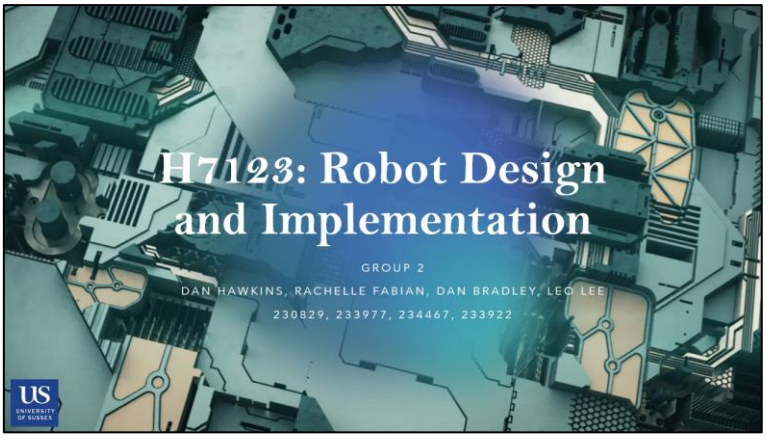
A.1.8.1 'Main.m' MATLAB Script – part 1

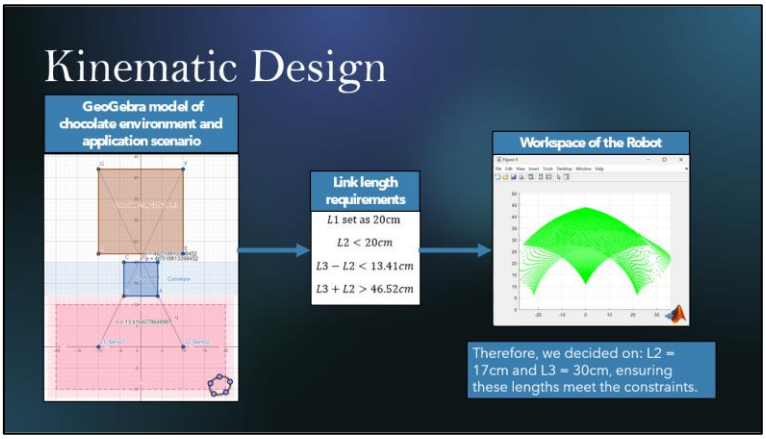
```
% Define Input Joint Angles
Servo = [90, 90];

% Define lengths of the links: L1, L2, L3 as per figure 1 in PDF.
%L = [20, 17, 30]; % Lengths for chocolate solution
L = [6, 5, 10]; % Lengths for circle and square live demo

[J1, J2, J3, J4, J5] = ForwardKinematics(Servo, L);
Joints =[J1;J2;J3;J4;J5];

% Workspace
workspace=zeros(100,2);
workspace_Angles_J2 = linspace(0,180,100);
workspace_Angles_J1 = linspace(180,0,100);
index = 1;

for i=1:length(workspace_Angles_J2)
    for j=1:length(workspace_Angles_J1)
        A2 = workspace_Angles_J2(i);
        A1 = workspace_Angles_J1(j);
        A = [A1,A2];
        [J1_w, J2_w, J3_w, J4_w, J5_w] = ForwardKinematics(A, L);
        workspace(index, :) = J5_w;
        index = index + 1;
    end
end

PlotLinks(Joints, workspace);
```

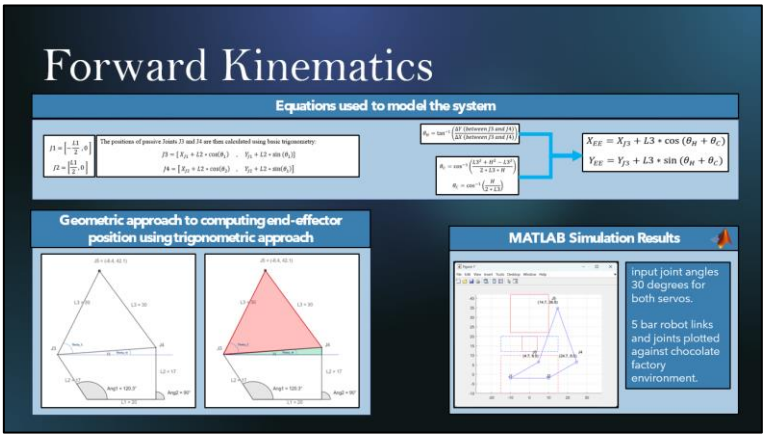
A.1.8.2 'Main.m' MATLAB Script – part 2 (for circle and square tracing live demo)

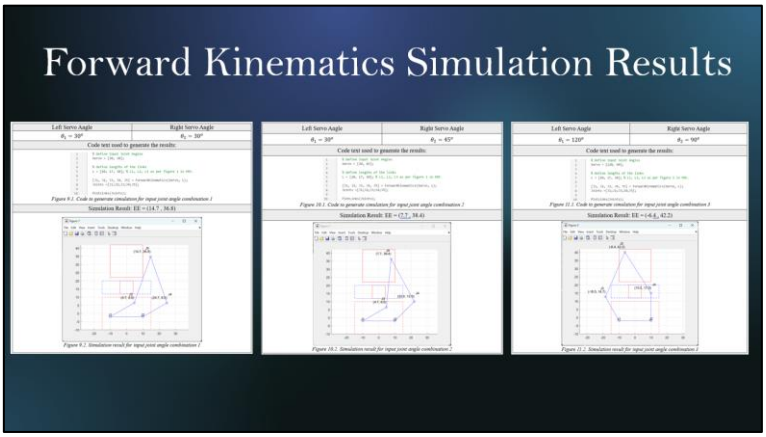
```
29
30      Shape = 2; % 1 for circle, 2 for square
31
32      Center = [0,12];
33      radius = 2;
34      n = 100; % number of points
35
36      if Shape == 1
37          [X,Y] = CircleXY(Center, radius, n);
38          plot(X, Y, '.','Color','b');
39
40          Servo1_Angles = [1,n];
41          Servo2_Angles = [1,n];
42
43          % InverseKinematics to store computed Servo 1 and 2 angles resp.
44          for i = 1:n
45              EndEffector = [X(1,i),Y(1,i)];
46              [A1, A2] = InverseKinematics(EndEffector,L);
47              Servo1_Angles(1,i)=A1;
48              Servo2_Angles(1,i)=A2;
49          end
50      elseif Shape == 2
51          SquareEE = SquareXY(Center, radius, n);
52
53          Servo1_Angles = [1,(n+1)];
54          Servo2_Angles = [1,(n+1)];
55
56          %Plotting Square on Workspace
57          for i = 1 : n+1
58              plot(SquareEE(i,1),SquareEE(i,2),'.','Color','b');
59          end
60          % InverseKinematics to store computed Servo 1 and 2 angles resp.
61          for i = 1 : n+1
62              EndEffector = [SquareEE(i,1),SquareEE(i,2)];
63              [A1, A2] = InverseKinematics(EndEffector,L);
64              Servo1_Angles(1,i)=A1;
65              Servo2_Angles(1,i)=A2;
66          end
67      end
68
69      % Passing Servo Angles to Servo
70      a = arduino('COM5','Mega2560');
71      s1 = servo(a,'D3','MaxPulseDuration',2.5e-3,'MinPulseDuration',0.5e-3);
72      s2 = servo(a,'D5','MaxPulseDuration',2.5e-3,'MinPulseDuration',0.5e-3);
73
74      %Converting Between Angle and PWM duty cycle
75      MappedAngles1=[];
76      MappedAngles1=arrayfun(@AngleMapping,Servo1_Angles);
77      MappedAngles2=[];
78      MappedAngles2=arrayfun(@AngleMapping,Servo2_Angles);
79
80      % Controlling the Servo Motors
81      for i=1:size(MappedAngles1,2)
82          writePosition(s1,MappedAngles1(1,i));
83          writePosition(s2,MappedAngles2(1,i));
84          pause(0.05);
85      end
```

A.2 Presentation Slides:

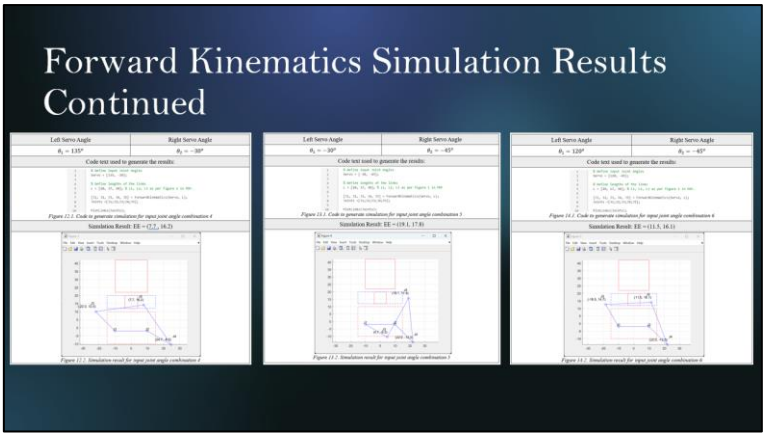




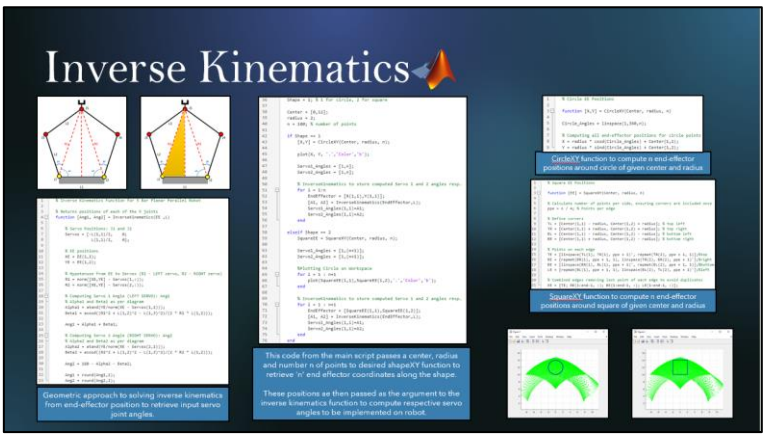




Forward Kinematics Simulation Results Continued



Inverse Kinematics



Geometric approach to solving inverse kinematics from end-effector position to retrieve input servo joint angles.

This code from the main script passes a center, radius and number n of points to desired shapeXY function to retrieve 'n' end effector coordinates along the shape.

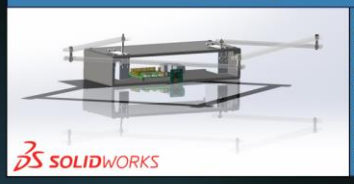
These positions se then passed as the argument to the inverse kinematics function to compute respective servo angles to be implemented on robot.

CircleXY function to compute n end-effector positions around circle of given center and radius

SquareXY function to compute n end-effector positions around square of given center and radius

Initial Design

The robot assembly (excluding the end effector)



We selected a Raspberry Pi 4 to operate the robot, using a Raspberry Pi camera 3 for machine vision.

The servo motors used are MG996R's (similarly to the demonstration model).

Connections between the links

M4 screw
Washer
Bearing

Washer
Designed Spacer
Washer
Bearing
Washer
Nut

Cost of assembly: £225.92

Initial End Effector Ideas

Electromagnetic gripper

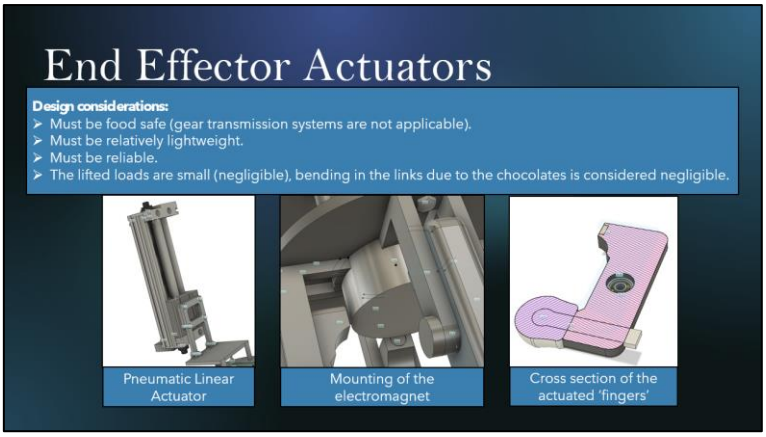
Vacuum gripper

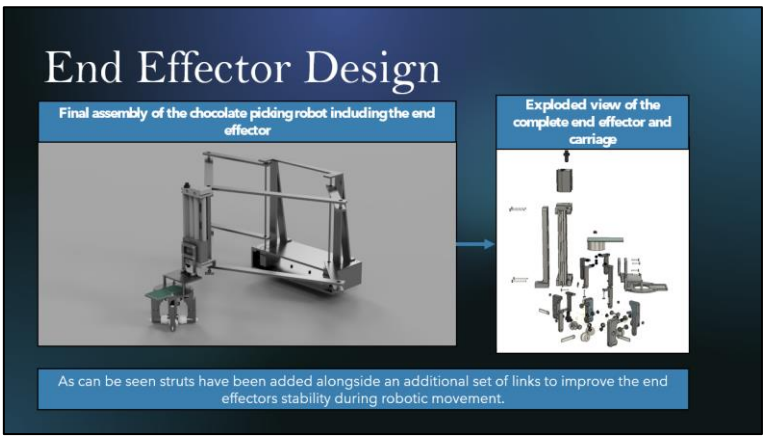
Having considered the pros and cons of each type of end effector:

The chosen solution was to use an electromagnetic actuator.

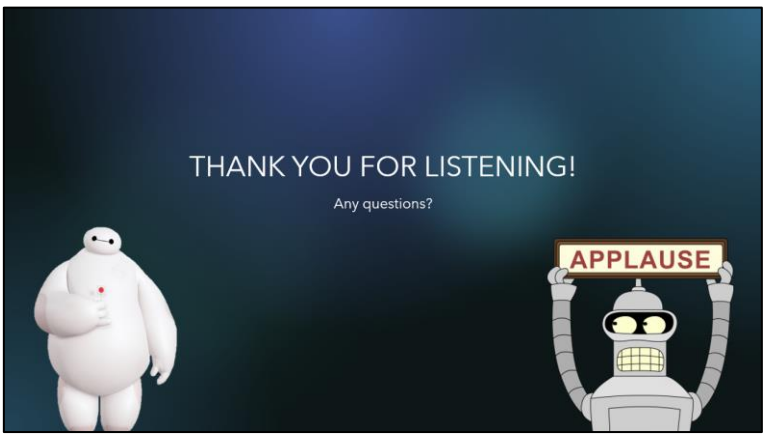
The use of a vacuum gripper requires the use of a compressor which is costly if it is not already installed.

The mounting of tubing connections for the vacuum gripper given the range and rate of movement was found to be problematic.


End Effector Actuators
Design considerations:
Must be food safe (gear transmission systems are not applicable).
Must be relatively lightweight.
Must be reliable.
The lifted loads are small (negligible), bending in the links due to the chocolates is considered negligible.
Pneumatic Linear Actuator
Mounting of the electromagnet
Cross section of the actuated 'fingers'


End Effector Design
Final assembly of the chocolate picking robot including the end effector
Exploded view of the complete end effector and carriage
As can be seen struts have been added alongside an additional set of links to improve the end effectors stability during robotic movement.


Complete Assembly Cost
ADD TO CART
Complete Assembly Cost : £899.50


THANK YOU FOR LISTENING!
Any questions?
APPLAUSE