MINISTRY OF EDUCATION AND TRAINING

HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION

SUBJECT: PROJECT ON ROBOTICS

HCMUTE

TOPIC: APPLICATION OF IMAGE PROCESSING AND 3-DOF SCARA ROBOTIC ARM IN OBJECT CLASSIFICATION BASED ON COLOR

Instructor: Assoc. Prof. Dr. Tran Duc Thien

Students:

1. Le Hoang Khang 21151022

2. Duong Hoang Khoi 21151027

Ho Chi Minh City, January 7th 2025

Table of Contents

List of Figure

List of Tables

CHAPTER 1: INTRODUCTION

1.1 PROBLEM STATEMENT

In the context of modern industry, the automation of manufacturing processes is becoming an inevitable trend to improve efficiency and reduce production costs. In particular, the process of sorting products by color plays a crucial role in many industries such as food, pharmaceuticals, and product packaging. However, traditional manual sorting methods are revealing many limitations such as slow processing speed, inconsistent accuracy, high labor costs, and difficulty meeting continuous production requirements.

The development of computer vision technology and industrial robots has opened up opportunities to address these challenges. Among them, SCARA robots, with advantages in simple structure, high accuracy, and fast processing capability, have become a suitable choice for industrial pick-and-place tasks. The combination of SCARA robots with vision systems enables complete automation of the sorting process, helping to increase productivity, ensure consistent quality, and minimize human errors.

Color-based object classification is one of the fundamental and effective applications of computer vision systems. Compared to other classification methods such as shape or size recognition, color classification has advantages in fast processing speed, high reliability, and ease of implementation. Particularly, when combined with a 3-DOF SCARA robot, the system can flexibly perform pick-and-place operations in a planar workspace, suitable for many industrial applications.

For these reasons, researching and developing an automated color-based object sorting system using a 3-DOF SCARA robot and image processing technology not only has academic significance but also brings high practical value in improving industrial production efficiency. This project aims to create an integrated solution, combining the advantages of vision technology and industrial robots to meet the growing demand for automation in manufacturing.

1.2 RELATED RESEARCH

In recent years, the integration of vision systems with industrial robots has received widespread research attention from both academia and industry. Numerous

studies have focused on developing efficient image processing algorithms for color-based object recognition and classification. Common methods include processing in RGB, HSV, or LAB color spaces, combined with image segmentation techniques and noise filtering to enhance recognition accuracy.

In terms of industrial robotics, SCARA robots have proven their effectiveness in pick-and-place applications due to their simple mechanical structure and precise control capabilities. Previous research has proposed various methods to optimize trajectory planning and kinematic control of SCARA robots, ranging from classical control algorithms like PID to intelligent control methods such as neural networks and fuzzy logic. Notably, the integration of cameras with SCARA robots has opened up new research directions in visual servoing control, allowing robots to adapt to dynamic work environments.

In the field of system integration, many studies have focused on developing efficient software architectures to synchronize vision systems and robot control. Proposed solutions typically concentrate on optimizing processing time, ensuring reliable communication between components, and developing user-friendly interfaces. However, most current research still has some limitations such as high implementation costs, complexity in system setup and calibration, as well as adaptability to different environmental conditions.

Based on previous research, this project aims to develop a complete integrated solution, focusing on optimizing the color sorting process using a 3-DOF SCARA robot. The project will inherit the advantages of previous studies while proposing improvements in image processing algorithms, robot control strategies, and system integration architecture to create an efficient and practical solution for automated object sorting in industry.

1.3 OBJECTIVES

The project "**Application of Image Processing and 3-DOF SCARA Robotic Arm in Object Classification Based on Color**" aims to design and develop an automated system capable of recognizing and classifying three types of bottle caps with distinct colors: red, yellow, and blue. The system is required to achieve an accuracy of at least 95% and efficiently process bottle caps with metallic components that are

uniform in shape and size. The project focuses on integrating and programming a 3-DOF SCARA robotic arm to perform pick-and-place operations based on color information obtained from the image processing system. A smartphone camera, connected to MATLAB via the DroidCam application, will be used for real-time image acquisition and processing. The camera will be fixed and adjusted to accurately capture images of the checkerboard grid, where each square measures 38 x 38 mm. Additionally, the project will conduct at least 50 trials with bottle caps of different colors to evaluate the system's accuracy and operational performance. The trial results will be analyzed to propose potential improvements for the system. Lastly, the project aims to develop a cost-effective solution utilizing readily available components such as a smartphone camera, stepper motors, and an electromagnetic gripper, ensuring practicality and scalability for future applications.

1.4 SCOPE AND LIMITATIONS

The scope of this project is to design and develop an automated system that integrates a 3-DOF SCARA robotic arm with an image processing system for object classification based on color. The system is designed for industrial applications, focusing on accuracy, speed, and cost-effectiveness. However, the project has several specific limitations as follows:

1.4.1 Object Limitations

The system classifies three types of bottle caps with distinct colors: red, yellow, and blue. These caps are designed with metallic components to facilitate grasping using an electromagnetic gripper. Objects must be uniform in shape and size, and non-reflective surfaces are preferred. Non-compliant objects or those with complex color patterns may reduce the system's classification accuracy.

1.4.2 Operating Environment Limitations

The camera used in the system is a smartphone camera, connected to MATLAB software through the DroidCam application. The camera is fixed on a holder and positioned to capture images within the workspace defined by a checkerboard grid. The checkerboard serves as a placement area for bottle caps, with each square having dimensions of 38 x 38 mm. The operating environment requires consistent lighting to minimize shadows or vibrations that may interfere with image processing. The robotic

arm’s workspace is constrained by a maximum height of 280 mm and a working radius of 479 mm.

1.4.3 Technical Limitations

The system uses stepper motors for actuation and an electromagnetic gripper for object handling. These components may limit speed and precision compared to advanced systems employing servo motors or modern control technologies. While the smartphone camera provides real-time images and the system can recognize the color and position of bottle caps almost instantly upon pressing the "Detect" button on the MATLAB GUI, the picking and placing process may still be influenced by the mechanical speed of the robotic arm.

1.4.4 Functional Limitations

The system is currently designed only to classify bottle caps based on color (red, yellow, blue) and does not support classification by other attributes such as size or shape. Although the system can detect color and position quickly, the overall processing speed is limited by the mechanical performance of the robotic arm. Additionally, the system’s ability to integrate with other industrial processes is constrained due to its current design, which lacks modularity and scalability.

1.5 IMPLEMENTATION CONTENTS

No.	Content	Responsibility
1	Theoretical calculations: • Forward and inverse kinematics	Both members
2	Hardware development: • 3D modeling in SolidWorks • Hardware manufacturing • Robot assembly • Electrical system design	Both members
3	Software development: • MATLAB programming for kinematics • MATLAB programming for image processing • MATLAB programming for GUI • MATLAB programming for workspace	Both members

	• Arduino IDE programming for step motor control	
4	System integration: • Hardware-software integration • System testing and calibration	Both members
5	Documentation: • Project report	Both members

CHAPTER 2: SYSTEM OVERVIEW

2.1 SYSTEM INTRODUCTION

The 3-degree-of-freedom robotic arm system utilizes image processing for object classification based on color, after being fully configured:

Figure 2.1: Complete system

The system is designed to classify bottle caps based on color by integrating image processing technology and a 3-degree-of-freedom SCARA robotic arm (3-DOF). The system includes the following main components:

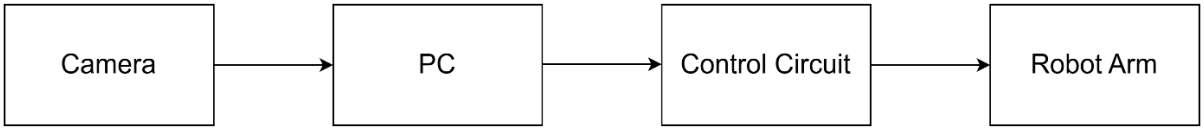


Figure 2.2: System Block Diagram

Detailed Description:

1. Camera: Captures images of the workspace, providing real-time input data for the processing system. A smartphone camera is used in this setup and is connected to MATLAB software via the **DroidCam** application. The camera is mounted on a fixed

holder and positioned to cover the designated workspace accurately. The checkerboard grid under the workspace ensures precise alignment and scale calibration for the captured images.

2. Image Processing System: This component processes the captured images to identify the colors and positions of bottle caps. The system utilizes MATLAB software to implement advanced recognition and classification algorithms. Images are converted into the HSV color space to improve detection accuracy under varying lighting conditions. The color classification is based on pre-defined HSV thresholds for red, blue, and yellow. Noise reduction filters and segmentation techniques are applied to enhance recognition reliability. This system also integrates a **Graphical User Interface (GUI)** designed in MATLAB, enabling users to monitor the real-time processing, adjust parameters, and interact intuitively with the system.

3. Arduino Controller: Serves as the central communication hub between the image processing system and the robotic arm. It receives control signals from MATLAB, processes them, and transmits appropriate commands to the robot's motors via the **CNC Shield V3** expansion board. The Arduino controller ensures synchronization of all hardware components and handles motor control with precise signal modulation through **A4988 motor drivers**.

4. SCARA Robotic Arm: Executes pick-and-place operations for sorting bottle caps into designated storage areas based on their classified colors. The robotic arm is a 3-degree-of-freedom (3-DOF) SCARA design, optimized for high accuracy and repeatability. It employs stepper motors for precise control of movements along the horizontal and vertical axes. An electromagnetic gripper attached to the arm securely handles the bottle caps during sorting operations.

The system operates in a stepwise manner as follows:

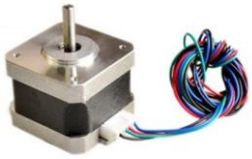
- **Image Acquisition**: The camera captures images of the workspace, including the randomly placed bottle caps.
- **Image Analysis**: The image processing system analyzes the data to detect the colors (red, blue, yellow) and positions of the bottle caps within the workspace.

- **User Interaction**: The GUI displays the analysis results, including real-time color classification and positions. The user can activate the robotic arm for sorting tasks through intuitive GUI controls.
- **Robotic Operation**: The SCARA robotic arm receives commands from the Arduino controller to pick and sort the bottle caps into their respective storage bins. The robot's actions are continuously monitored and updated on the GUI.

2.2 EQUIPMENT SELECTION

Below is a list of equipment used in this project:

Table 1: *List of equipment*

LIST OF EQUIPMENT			
No.	**Name**	**Images**	**Characteristic**
1	Arduino UNO R3 DIP		Operating Voltage: 5V DC Clock Frequency: 16 MHz Number of Digital I/O Pins: 14 (6 PWM pins) Flash Memory: 32 KB Output current at 3.3V pin: 50mA
2	Step motor NEMA 17 (42x48mm)		Length: 48mm Shaft diameter: 5mm Rated current: 1.5A Holding torque: 0.55 Nm Rotation angle per step: 1.8 °
3	Step motor NEMA 17 (42x34mm)		Length: 34mm Shaft diameter: 5mm Rated amperage: 1A Holding torque: 0.26 Nm Rotation angle per step: 1.8 °

4	Board CNC Shield V3 A4988 driver		Arduino UNO expansion board There are 4 slots for A4988 driver Controls 3 axes X, Y, Z and 1 optional axis Power supply: 12V - 36V
5	Driver A4988 stepper motor		Used to control Step Motor Input voltage: 8V – 35V Logic voltage 1: 3V – 5.5V Resolution: full, 1/2, 1/4, 1/8 and 1/16
7	12V5A Adapter Power Supply		Input voltage: AC100-240V 50/60HZ Output voltage: 12VDC Output current: Max 5A DC socket: 5.5 * 2.1mm
8	3Kg Electromagnet (24V)		Supply voltage: 24VDC Suction force: 3kg Power: 2 - 2.9W
10	D-Type Roller		To make Link 1 slide on aluminum more smoothly
11	608zz Bearing		To have the drive mechanism of the Links fixed and move smoothly
12	605z Bearing		To have the drive mechanism of the Links fixed and move smoothly
13	GT2 belt (meter)		To pull the Links for the prismatic joint

14	GT2 200mm Belt Loop		To connect with gears to form the drive mechanism for the Link
15	M5 40mm Hex Screw M3 10mm Hex Screw		To fix the motor and the Links
16	M5 Hex Nut M3 Hex Nut		To fix the motor and the Links
17	Module 1 Relay 5V		Max Load Voltage: AC 250V-10A/ DC 30V-10A Control Voltage: 5 VDC Relay Trigger Current: 5mA Trigger State: High Level
18	8mm diameter, 100mm length Rod		Used for attaching to an electromagnet
19	Domino TB1505 15A 600V		Number of Connections: 5 connections Rated Voltage: 600V Rated Current: 15A
20	Perforated Circuit Board		Mount electronic components in the project
21	Male and Female Bus Wires		Connect wires to each other or to other devices

22	DC-DC power supply circuit with multiple outputs 3.3V 5V 12V		Input voltage: 6~12V Output voltage 3V, 5V, 12V With power indicator light (red) DC Jack 5.5x2.1mm

2.3 EQUIPMENT COSTS

Table 2: *Equipment costs*

HARDWARE DEVICES					
STT	**Device Name**	**Unit price**	**Quantity**	**Costs**	**Link**
1	Step motor NEMA 17 (42x48mm)	189,000	2	378,000	https://icdayroi.com/step-motor-nema-17-size-42-x-48mm-dong-co-buoc
2	Step motor NEMA 17 (42x34mm)	139,000	1	139,000	https://icdayroi.com/step-motor-nema-17-size-42-x-34mm-dong-co-buoc
3	Arduino UNO R3 DIP	125,000	2	250,000	https://icdayroi.com/arduino-uno-r3-dip
4	Board CNC Shield V3 A4988 driver	32,000	1	32,000	https://icdayroi.com/board-cnc-shield-v3-a4988-driver
5	Driver A4988 stepper motor	20,000	2	40,000	https://icdayroi.com/driver-a4988-stepper-motor
6	12V5A Adapter Power Supply	95,000	1	95,000	https://icdayroi.com/nguon-adapter-12v5a
7	3Kg Electromagnet (24V)	94,000	1	94,000	https://www.thegioiic.com/ele-p20-15-24v-nam-cham-dien-3kg
8	D-Type Roller	13,000	8	104,000	https://www.thegioiic.com/banh-xe-con-lan-d-type-duong-kinh-24mm-truc-5mm

9	608zz Bearing	4,000	15	60,000	https://icdayroi.com/vong-bi-608zz
10	605z Bearing	5,000	1	5,000	https://dientutuyetnga.com/products/bac-dan-vong-bi-605z
11	GT2 belt (meter)	20,000	2	40,000	https://icdayroi.com/day-dai-gt2-buoc-rang-2mm-rong-6mm-met
12	GT2 200mm Belt Loop	14,000	5	70,000	https://icdayroi.com/vong-day-dai-gt2-200mm
13	M5 40mm Hex Screw	1,600	17	27,000	https://icdayroi.com/oc-luc-giac-m5-40mm
14	M5 Hex Nut	500	30	15,000	
15	M3 10mm Hex Screw	600	20	12,000	https://icdayroi.com/oc-luc-giac-m3-10mm
16	M3 Hex Nut	190	30	6,000	
17	Module 1 relay 5V	12,000	1	12,000	https://icdayroi.com/module-1-relay-5v-kich-muc-cao
18	8mm diameter, 100mm length Rod	18,000	1	18,000	https://dientutuyetnga.com/products/truc-tron-8mm-dai-100mm
19	Domino TB1505 15A 600V	5,000	1	5,000	https://icdayroi.com/domino-tb1505-15a-600v
20	Perforated Circuit Board	5,000	1	5,000	
21	Male and Female Bus Wires	450	30	14,000	https://icdayroi.com/day-bus-day-be-duc-cai-30cm-soi
22	DC-DC power supply circuit	25,000	1	25,000	https://icdayroi.com/mach-cap-nguon-dc-dc-nhieu-dau-ra-3-3v-5v-12v
23	3D Printing	1,578,000	1	1,547,000	
Total				3,024,000	

CHAPTER 3: THEORETICAL BASIC

3.1 FORWARD KINEMATICS USING DENAVIT-HARTENBERG APPROACH

Forward Kinematics is the calculation of the position and orientation of an end effector using the variables of the joints and linkages connecting to the end effector. Given the current positions, angles, and orientation of the joints and linkages, forward kinematics can be used to calculate the position and orientation of the end effector.

Denavit – Hartenberg (D-H) approach:

1. Attach the frames to the manipulator.
2. Determine the Denavit – Hartenberg parameters.
3. Compute the individual transformations for each link.
4. Find the transformation between the frame {N} and frame {0}

Summary of link-frame attachment procedure:

1. Identify the joint axes and imagine infinite lines along them.
2. Identify the common perpendicular between them.
3. Assign the Z_i axis pointing along ith joint axis.
4. Assign the X_i axis pointing along the common perpendicular.
5. Assign the Y_i axis to complete a right-hand coordinate system.
6. Assign {0} to match {1} when the first joint variable is zero.
7. For {N}, choose an origin location and X_N direction freely.

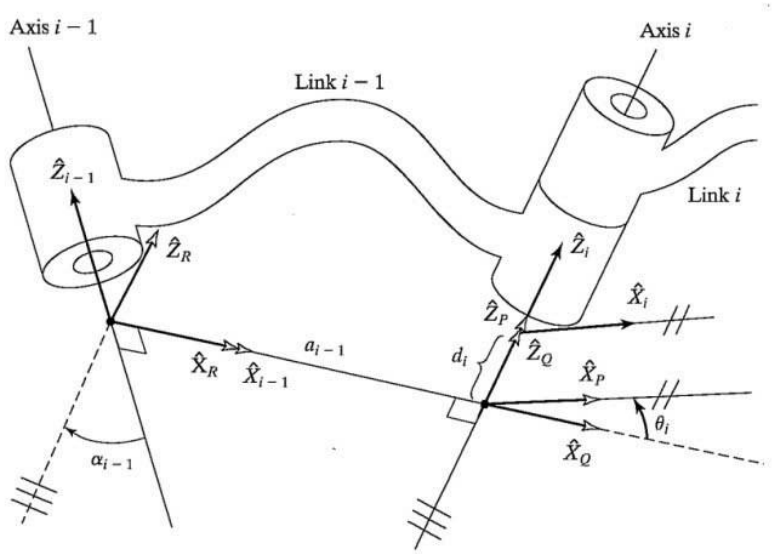


Figure 3.1: Location of intermediate frames $\{P\}$, $\{Q\}$, and $\{R\}$

Where:

α_i = the angle from Z_i to Z_{i+1} measured around X_i

a_i = the distance from Z_i to Z_{i+1} measured along X_i

d_i = the distance from X_{i-1} to X_i measured along Z_i

θ_i = the angle from X_{i-1} to X_i measured about Z_i

The Denavit – Hartenberg parameters:

i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
...				

The general form of the individual transformation:

$$ {}^{i-1}_{i}T = \begin{bmatrix} c_i & -s_i & 0 & a_{i-1} \\ s_i c\alpha_{i-1} & c_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s_i s\alpha_{i-1} & c_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

3.2 INVERSE KINEMATICS USING ALGEBRAIC METHOD

Inverse Kinematics is the calculation of the variables of the set of joints and linkages connected to an end effector. Given the position and orientation of the end effector, inverse kinematics can be used to calculate the variables regarding those joints and linkages including position, angle, and orientation.

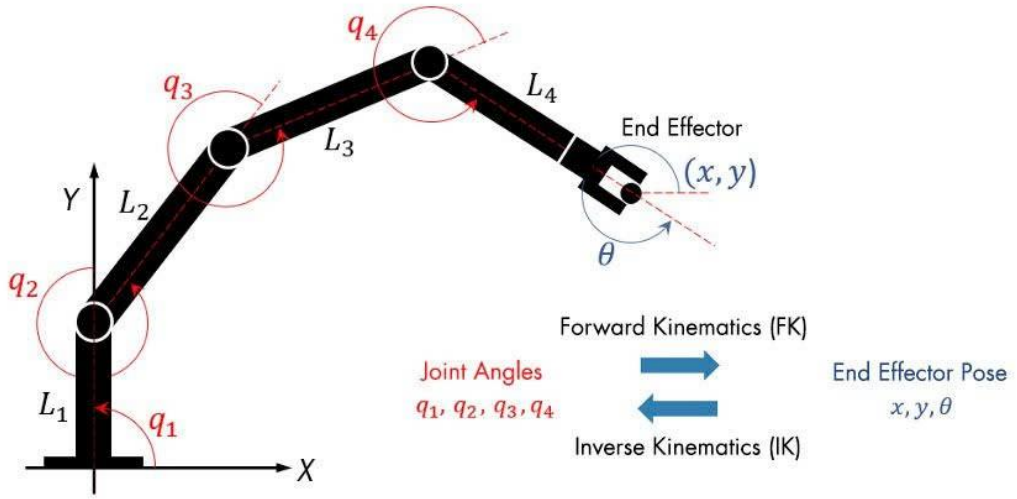


Figure 3.2: Configuring the joint positions of a robot using kinematics equation

3.2.1 Homogeneous Transformation Matrix

The homogeneous transformation matrix is an essential mathematical tool in robotics, used to represent the position and orientation of an object in 3D space. The special Euclidean group SE(3) is the set of all 4×4 real matrices T of the form:

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_1 \\ r_{21} & r_{22} & r_{23} & p_2 \\ r_{31} & r_{32} & r_{33} & p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R \in SO(3), p \in \mathbb{R}^3$$

Where:

R_{3x3}: 3x3 rotation matrix, describing orientation

p_{3x1}: 3x1 position vector, describing translation

[0 0 0 1]: augmented vector

Properties of transformation matrices:

Inverse: $T^{-1} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R^T & -R^T p \\ 0 & 1 \end{bmatrix} \in SE(3)$

Closure: $T_1 T_2 \in SE(3)$

Associative: $(T_1 T_2) T_3 = T_1 (T_2 T_3)$

Not commutative: $T_1 T_2 \neq T_2 T_1$

3.2.2 Trigonometric Relations

Basic trigonometric function:

$$cos(A + B) = cosAcosB - sinAsinB$$

$$sin(A + B) = sinAcosB + cosAsinB$$

$$sin^2(A) + cos^2(A) = 1$$

$$sin(-A) = -\sin(A)$$

$$cos(-A) = \cos(A)$$

$$sin(A) = cos(B)\ if\ A + B = \frac{\pi}{2}$$

$$General\ form: \begin{cases} sin(A) = x \\ cos(A) = y \end{cases} \rightarrow A = atan2(x; y)$$

CHAPTER 4: SYSTEM DESIGN

4.1 FORWARD AND INVERSE KINEMATICS CALCULATION

Functions/ equations/ formulas used for mathematical analysis:

Basic_01:

$$\text{General form: } a\sin(x)+b\cos(x)=d \qquad \left(\sqrt{a^2+b^2}\neq 0\right)$$

$$\Rightarrow \frac{a}{\sqrt{a^2+b^2}}\sin(x)+\frac{b}{\sqrt{a^2+b^2}}\cos(x)=\frac{d}{\sqrt{a^2+b^2}}$$

$$x=\operatorname{atan2}\left(\frac{d}{\sqrt{a^2+b^2}};\pm\sqrt{1-\frac{d^2}{a^2+b^2}}\right)-\alpha$$

$$\text{Where: } \alpha=\operatorname{atan2}\left(\frac{b}{\sqrt{a^2+b^2}};\frac{a}{\sqrt{a^2+b^2}}\right)$$

4.1.1 Modified DH table

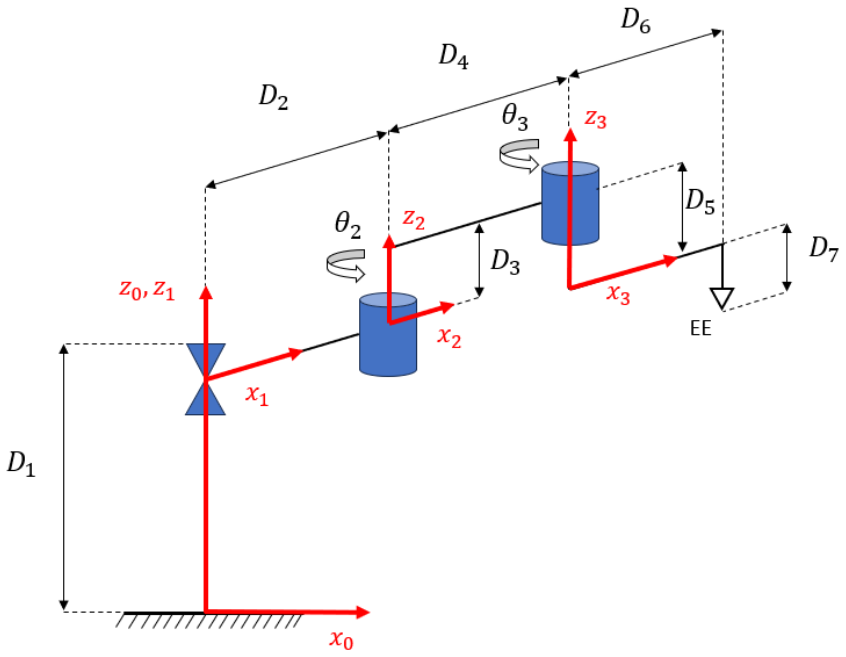


Figure 4.1: 3-DOF SCARA ROBOT MODEL

Dimension of robot is 652 x 220 x 688 mm (L x W x H)

Limit	
Prismatic Joint (D_1)	0 – 320 mm
Rotating Joint 1 (θ_2)	–90° ÷ 90°

Rotating Joint 2 (θ_3)	–90° ÷ 90°

Joint Length	
D_1	320 mm
D_2	146 mm
D_3	56 mm
D_4	160 mm
D_5	59 mm
D_6	173 mm
D_7	85 mm

Table 3: *DH table*

	i	a_{i-1}	α_{i-1}	d_i	θ_i	
$z_0 z_1 x_0$	1	0	0	D_1	0	$x_0 x_1 z_1$
$z_1 z_2 x_1$	2	D_2	0	0	θ_2	$x_1 x_2 z_2$
$z_2 z_3 x_2$	3	D_4	0	0	θ_3	$x_2 x_3 z_3$

4.1.2 Forward Kinematic Calculation

$$ {}^{3}P_{EE} = \begin{bmatrix} D_6 \\ 0 \\ -D_7 \end{bmatrix} \tag{3.1} $$

$$ {}^{0}_{1}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & D_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, {}^{0}_{1}R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, {}^{0}P_1 = \begin{bmatrix} 0 \\ 0 \\ D_1 \end{bmatrix} \tag{3.2} $$

$$ {}^{1}_{2}T = \begin{bmatrix} c_2 & -s_2 & 0 & D_2 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, {}^{1}_{2}R = \begin{bmatrix} c_2 & -s_2 & 0 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, {}^{1}P_2 = \begin{bmatrix} D_2 \\ 0 \\ 0 \end{bmatrix} \tag{3.3} $$

$$ {}^{2}_{3}T = \begin{bmatrix} c_3 & -s_3 & 0 & D_4 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, {}^{2}_{3}R = \begin{bmatrix} c_3 & -s_3 & 0 \\ s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}, {}^{2}P_3 = \begin{bmatrix} D_4 \\ 0 \\ 0 \end{bmatrix} \tag{3.4} $$

$$ \begin{bmatrix} {}^{2}P_{EE} \\ 1 \end{bmatrix} = {}^{2}_{3}T \begin{bmatrix} {}^{3}P_{EE} \\ 1 \end{bmatrix} = \begin{bmatrix} c_3 & -s_3 & 0 & D_4 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} D_6 \\ 0 \\ -D_7 \\ 1 \end{bmatrix} = \begin{bmatrix} c_3 D_6 + D_4 \\ s_3 D_6 \\ -D_7 \\ 1 \end{bmatrix} \tag{3.5} $$

$$ \begin{bmatrix} {}^{1}P_{EE} \\ 1 \end{bmatrix} = {}^{1}_{2}T \begin{bmatrix} {}^{2}P_{EE} \\ 1 \end{bmatrix} = \begin{bmatrix} c_2 & -s_2 & 0 & D_2 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_3 D_6 + D_4 \\ s_3 D_6 \\ -D_7 \\ 1 \end{bmatrix} = \begin{bmatrix} c_2 c_3 D_6 + c_2 D_4 - s_2 s_3 D_6 + D_2 \\ s_2 c_3 D_6 + s_2 D_4 + c_2 s_3 D_6 \\ -D_7 \\ 1 \end{bmatrix} \tag{3.6} $$

$$ \begin{bmatrix} {}^{0}P_{EE} \\ 1 \end{bmatrix} = {}^{0}_{1}T \begin{bmatrix} {}^{1}P_{EE} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & D_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_2 c_3 D_6 + c_2 D_4 - s_2 s_3 D_6 + D_2 \\ s_2 c_3 D_6 + s_2 D_4 + c_2 s_3 D_6 \\ -D_7 \\ 1 \end{bmatrix} $$

$$ = \begin{bmatrix} c_2 c_3 D_6 + c_2 D_4 - s_2 s_3 D_6 + D_2 \\ s_2 c_3 D_6 + s_2 D_4 + c_2 s_3 D_6 \\ -D_7 + D_1 \\ 1 \end{bmatrix} \tag{3.7} $$

$$ => {}^{0}P_{EE} = \begin{bmatrix} c_2 c_3 D_6 + c_2 D_4 - s_2 s_3 D_6 + D_2 \\ s_2 c_3 D_6 + s_2 D_4 + c_2 s_3 D_6 \\ -D_7 + D_1 \end{bmatrix} = \begin{bmatrix} c_{23} D_6 + c_2 D_4 + D_2 \\ s_{23} D_6 + s_2 D_4 \\ -D_7 + D_1 \end{bmatrix} \tag{3.8} $$

4.1.3 Inverse Kinematic Calculation

Set

$$ \begin{cases} c_2 c_3 D_6 + c_2 D_4 - s_2 s_3 D_6 + D_2 = P_x \\ s_2 c_3 D_6 + s_2 D_4 + c_2 s_3 D_6 = P_y \\ -D_7 + D_1 = P_z \end{cases} \tag{3.9} $$

$$ c_{23} D_6 + c_2 D_4 + D_2 = P_x \tag{3.10} $$

$$s_{23}D_6 + s_2D_4 = P_y \tag{3.11}$$
$$-D_7 + D_1 = P_z \tag{3.12}$$

From equation (3.10) and (3.11), we have

$$\begin{cases} c_{23}D_6 + c_2D_4 + D_2 = P_x \\ s_{23}D_6 + s_2D_4 = P_y \end{cases} \tag{3.13}$$

$$\Leftrightarrow \begin{cases} \left(c_{23}D_6\right)^2 = \left(P_x - c_2D_4 - D_2\right)^2 \\ \left(s_{23}D_6\right)^2 = \left(P_y - s_2D_4\right)^2 \end{cases}$$

$$\left(c_{23}D_6\right)^2 = \left(P_x - c_2D_4 - D_2\right)^2 \tag{3.14}$$
$$\left(s_{23}D_6\right)^2 = \left(P_y - s_2D_4\right)^2 \tag{3.15}$$

Adding (3.14) and (3.15), we have:

$$\left(c_{23}D_6\right)^2 + \left(s_{23}D_6\right)^2 = \left(P_x - c_2D_4 - D_2\right)^2 + \left(P_y - s_2D_4\right)^2$$
$$\Leftrightarrow D_6^2 = \left(P_x - D_2\right)^2 + P_y^2 - 2\left(P_x - D_2\right)c_2D_4 - 2P_ys_2D_4 + D_4^2c_2^2 + D_4^2s_2^2 \tag{3.16}$$
$$\Leftrightarrow 2D_4P_ys_2 + 2D_4\left(P_x - D_2\right)c_2 = \left(P_x - D_2\right)^2 + P_y^2 + D_4^2 - D_6^2 \tag{3.17}$$

Set

$$\begin{cases} a = 2D_4P_y \\ b = 2D_4\left(P_x - D_2\right) \\ d = \left(P_x - D_2\right)^2 + P_y^2 + D_4^2 - D_6^2 \end{cases} \tag{3.18}$$
$$\Rightarrow \sqrt{a^2 + b^2} = \sqrt{\left(2D_4P_y\right)^2 + \left(2D_4\left(P_x - D_2\right)\right)^2} \tag{3.19}$$

From Basic_01:

$$\Rightarrow \alpha = atan2\left(\frac{b}{\sqrt{a^2 + b^2}}; \frac{a}{\sqrt{a^2 + b^2}}\right) \tag{3.20}$$
$$= atan2\left(\frac{2D_4\left(P_x - D_2\right)}{\sqrt{\left(2D_4P_y\right)^2 + \left(2D_4\left(P_x - D_2\right)\right)^2}}; \frac{2D_4P_y}{\sqrt{\left(2D_4P_y\right)^2 + \left(2D_4\left(P_x - D_2\right)\right)^2}}\right) \tag{3.21}$$
$$\Rightarrow \theta_2 = atan2\left(\frac{d}{\sqrt{a^2 + b^2}}; \pm\sqrt{1 - \frac{d^2}{a^2 + b^2}}\right) - \alpha \tag{3.22}$$
$$= atan2\left(\frac{\left(P_x - D_2\right)^2 + P_y^2 + D_4^2 - D_6^2}{\sqrt{\left(2D_4P_y\right)^2 + \left(2D_4\left(P_x - D_2\right)\right)^2}}; \pm\sqrt{1 - \frac{\left(\left(P_x - D_2\right)^2 + P_y^2 + D_4^2 - D_6^2\right)^2}{\left(2D_4P_y\right)^2 + \left(2D_4\left(P_x - D_2\right)\right)^2}}\right) - \alpha \tag{3.23}$$

Substituting θ_2 into (3.10) and (3.11)

$$=> \begin{cases} c_{23} = \dfrac{P_x - c_2 D_4 - D_2}{D_6} \\ s_{23} = \dfrac{P_y - s_2 D_4}{D_6} \end{cases} \tag{3.24}$$

$$=> \theta_{23} = atan2\left(\frac{P_y - s_2 D_4}{D_6}; \frac{P_x - c_2 D_4 - D_2}{D_6} \right) \tag{3.25}$$

We have:

$$\theta_{23} = \theta_2 + \theta_3$$

$$=> \theta_3 = \theta_{23} - \theta_2 \tag{3.26}$$

From equation (3.12), we have:

$$D_1 = P_z + D_7 \tag{3.27}$$

In conclusion, there are 2 sets of solution in total.

$$\begin{cases} \theta_2 = atan2\left(\dfrac{\left(P_x - D_2\right)^2 + P_y^2 + D_4^2 - D_6^2}{\sqrt{\left(2D_4 P_y\right)^2 + \left(2D_4\left(P_x - D_2\right)\right)^2}}; \sqrt{1 - \dfrac{\left(\left(P_x - D_2\right)^2 + P_y^2 + D_4^2 - D_6^2\right)^2}{\left(2D_4 P_y\right)^2 + \left(2D_4\left(P_x - D_2\right)\right)^2}} \right) - \alpha \\ \theta_3 = \theta_{23} - \theta_2 \\ D_1 = P_z + D_7 \end{cases}$$

$$\begin{cases} \theta_2 = atan2\left(\dfrac{\left(P_x - D_2\right)^2 + P_y^2 + D_4^2 - D_6^2}{\sqrt{\left(2D_4 P_y\right)^2 + \left(2D_4\left(P_x - D_2\right)\right)^2}}; -\sqrt{1 - \dfrac{\left(\left(P_x - D_2\right)^2 + P_y^2 + D_4^2 - D_6^2\right)^2}{\left(2D_4 P_y\right)^2 + \left(2D_4\left(P_x - D_2\right)\right)^2}} \right) - \alpha \\ \theta_3 = \theta_{23} - \theta_2 \\ D_1 = P_z + D_7 \end{cases}$$

4.2 HARDWARE DESIGN

4.2.1 Designing Robot Model using Solidworks software

4.2.1.1 Drawing Robot Model in Solidworks software

After launching SolidWorks, we need to select the appropriate measurement unit for our design requirements. To draw the robot, the required unit is millimeters, so we will convert the units to millimeters. To change the unit, adjust the settings in the bottom right corner of the SolidWorks screen as illustrated in **Figure 4.2**.

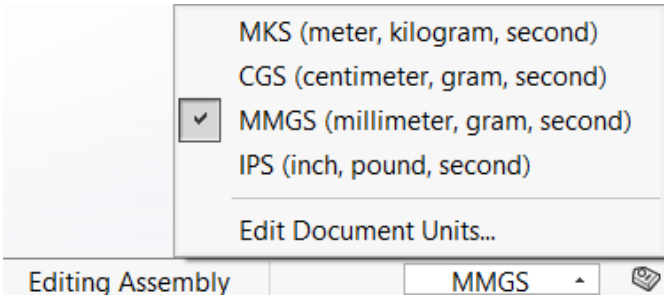


Figure 4.2: Unit conversion in SolidWorks

This is the 3D model of the Robot in SolidWorks Assembly environment as shown in **Figure 4.3**:

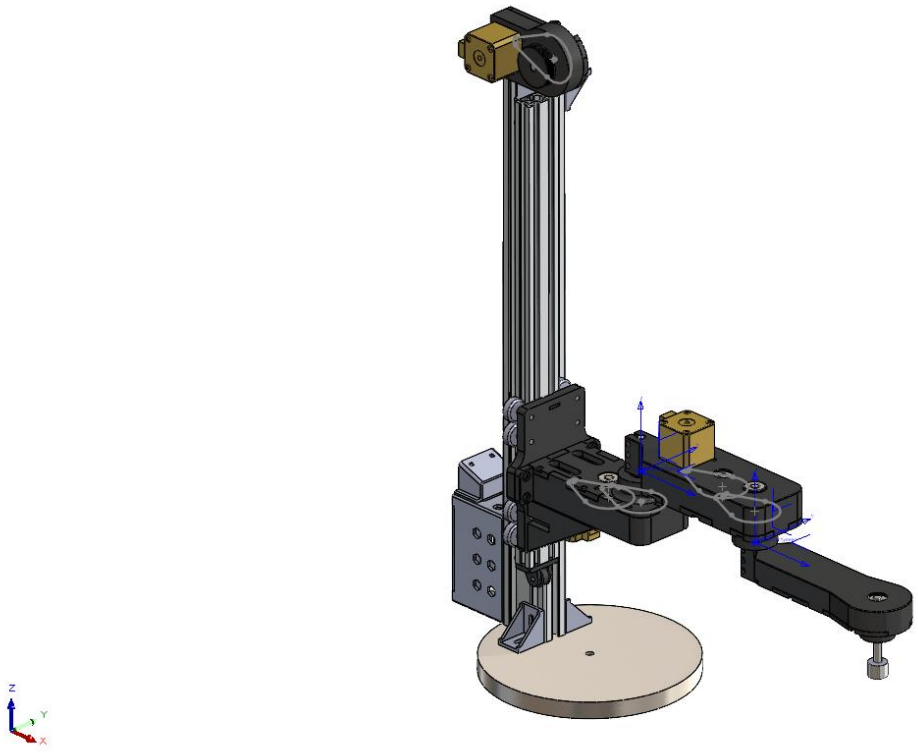


Figure 4.3: 3D Robot model in SolidWorks

We will look at each link in detail as follows:

- **Base Link:** During construction, our team selected aluminum and iron materials for aesthetics, sufficient weight (heavy to increase the robot's center of gravity, preventing tipping) and cost efficiency. We also designed an additional box to house the robot's control circuit (**Figure 4.5**). The iron base has holes to secure the robot to a table or any flat surface. The base link drawing is shown in **Figure 4.4**.

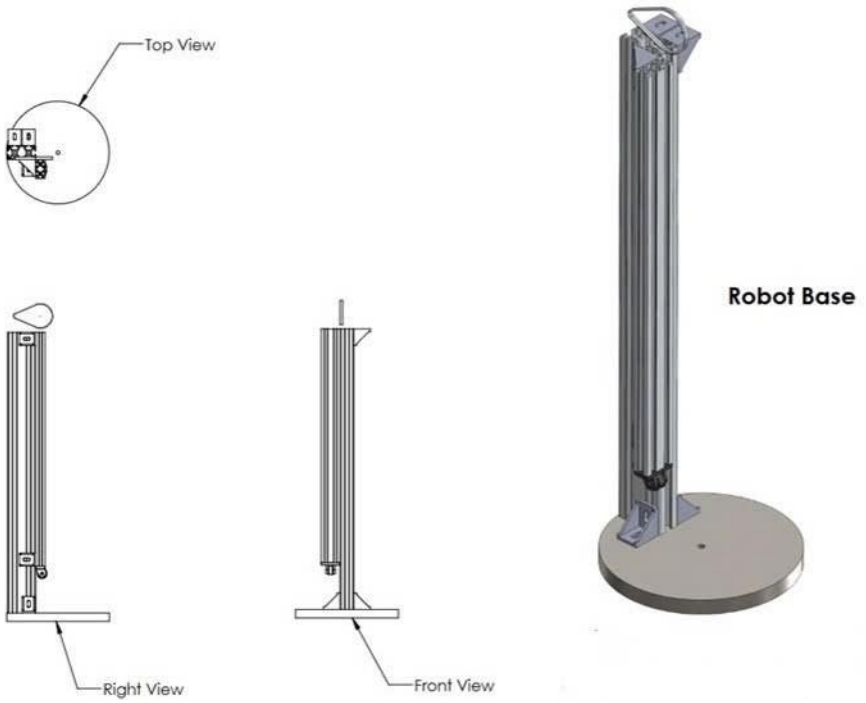


Figure 4.4: 3D model of Base Link

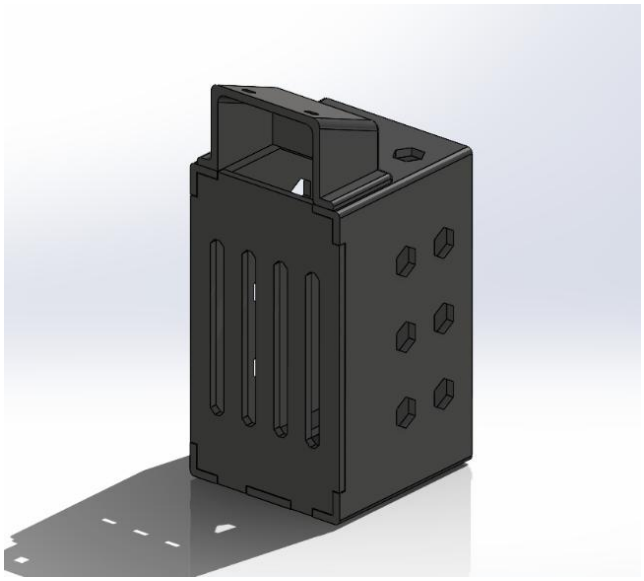


Figure 4.5: Circuit Box Housing

- **Z-axis Gearbox:** The team designed a gearbox using a NEMA 17 size 42x48 motor to drive the translational movement, with a gear ratio of 1:5. The gearbox and drive mechanism drawings are shown in **Figure 4.6** and **Figure 4.7**.

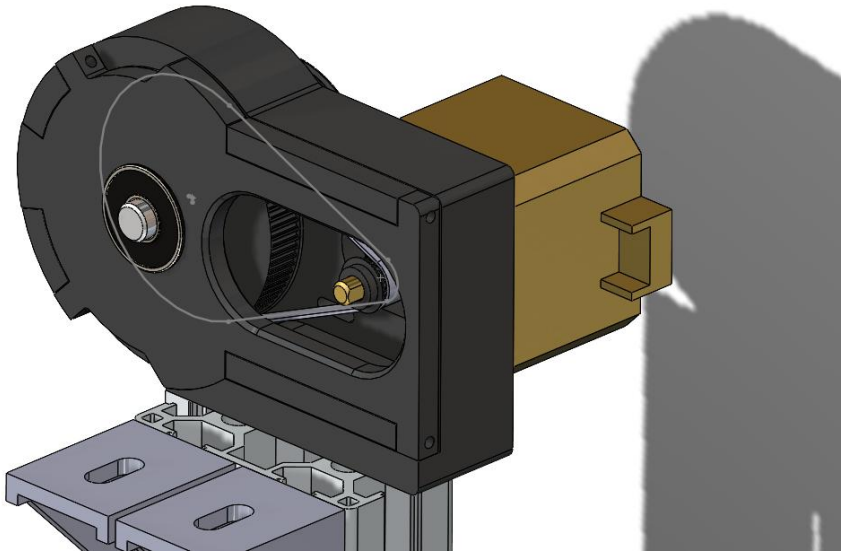


Figure 4.6: 3D drawing of gearbox

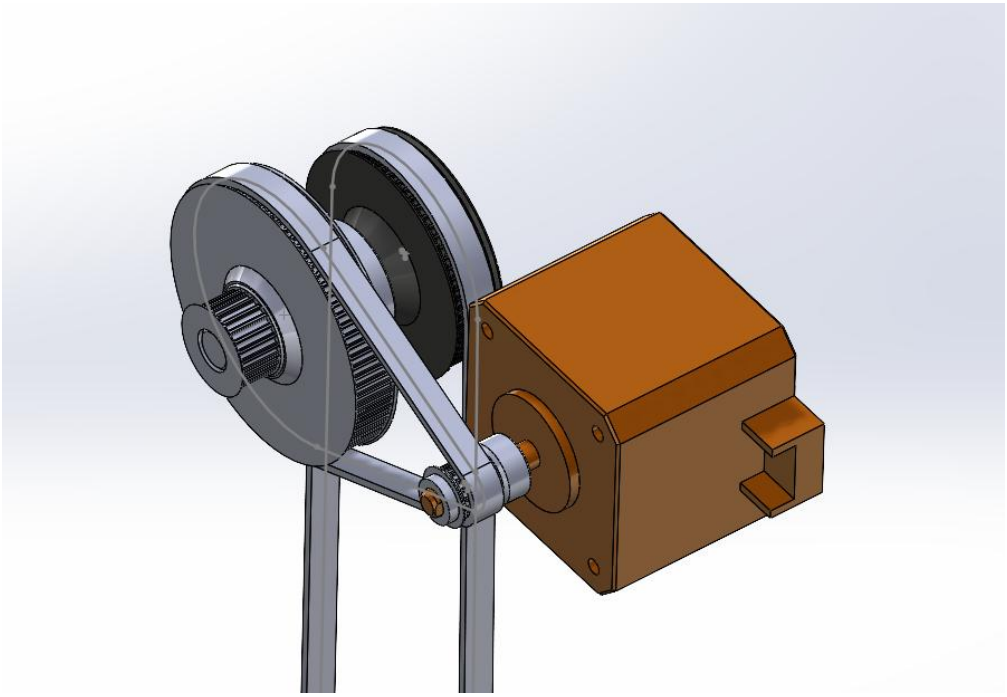


Figure 4.7: Gearbox drive mechanism

- **Link 1:** The team chose 3D printing for the links to optimize cost and link mass, and the gears were also 3D printed to match our desired dimensions. Link 1's drawing and drive mechanism are shown in **Figure 4.8** and **Figure 4.9**.

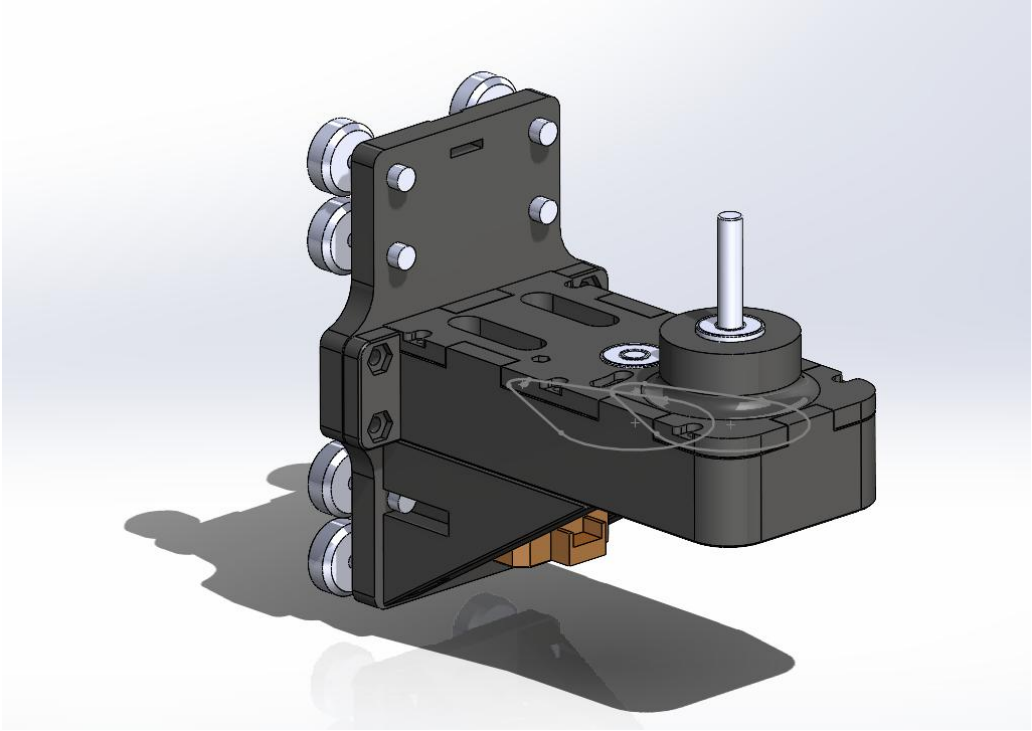


Figure 4.8: 3D drawing of Link 1

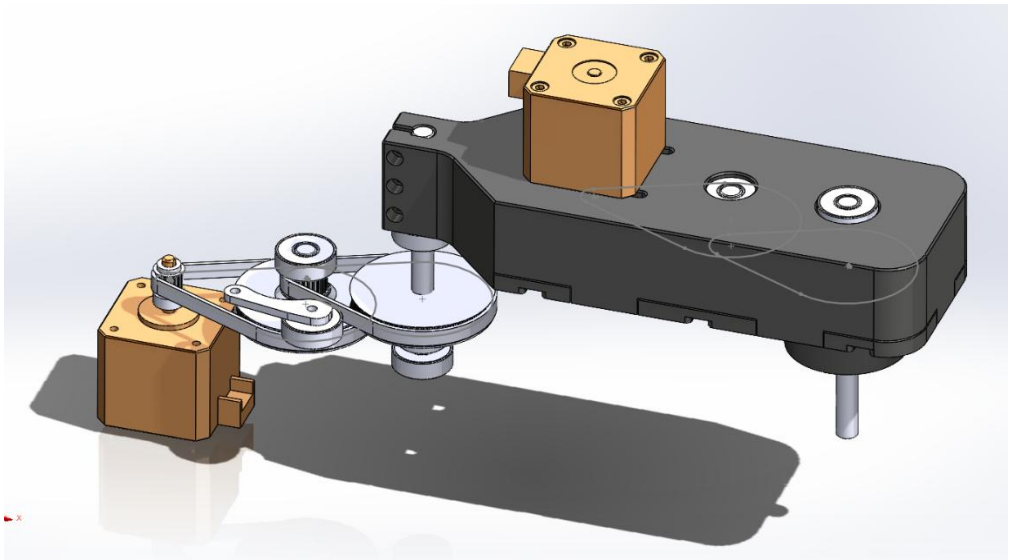


Figure 4.9: Drive mechanism of Link 1

Next are the drawings for Link 2 and Link 3 as shown in **Figures 4.10**, Figure **4.11** and Figure **4.12** below:

- **Design Requirements:** Similar to Link 1.

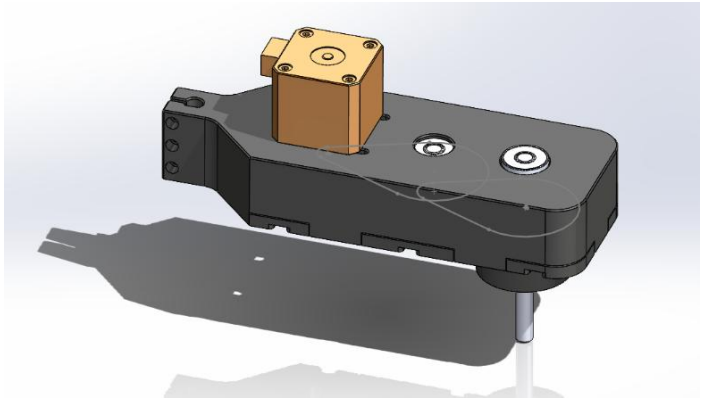


Figure 4.10: 3D Drawing of Link 2

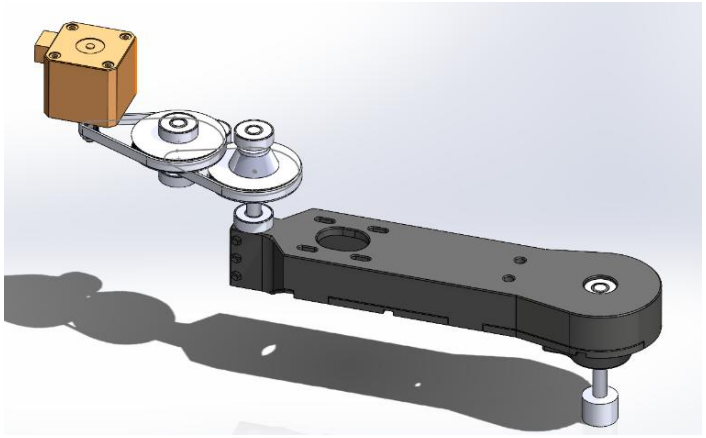


Figure 4.11: Drive mechanism of Link 2

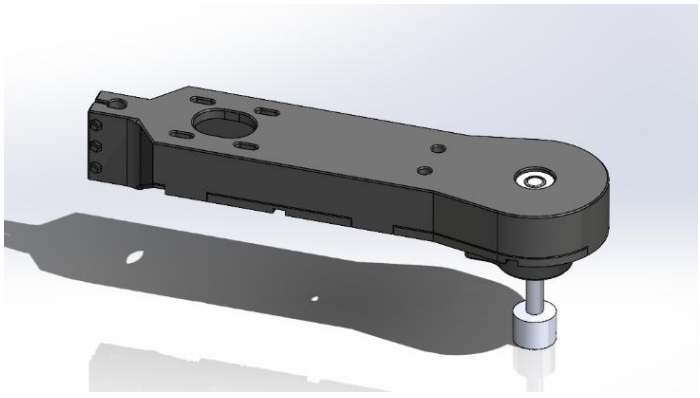


Figure 4.12: 3D Drawing of Link 3

4.2.1.2 Setting up robot axes in Solidworks

To set up the robot axis, the following steps are performed:

- Step 1: Draw the robot configuration and determine the common orthogonal lines of the robot.
- Step 2: Draw the coordinate axis {0} coinciding with the origin of the coordinate system.
- Step 3: Determine the Z_i axis coinciding with the motor axis (at the i-th rotary joint).
- Step 4: Determine the X_i axis along the common orthogonal lines between the two adjacent Z-axes.
- Step 5: Determine the Y_i axis according to the right-hand rule.
- Step 6: Determine so that the axis of the coordinate system {0} coincides with the coordinate system {1}, and the coordinate system {N} will choose the X_N randomly (usually along the position of the end point).

We set up the axis for a 3-DOF robot in SolidWorks as shown in Figure 3.12 as follows:

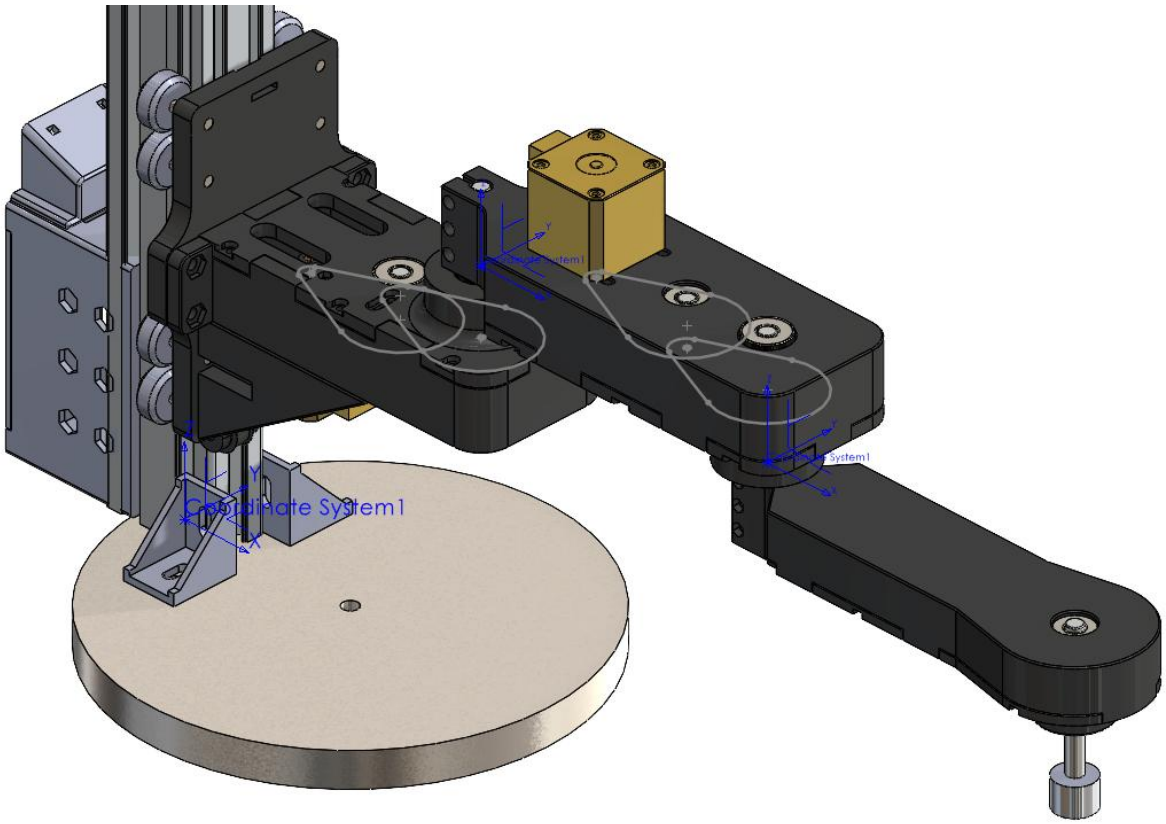


Figure 4.13: Setting up Axis in SolidWorks

4.2.2 Connection diagram and Wiring diagram

The connection diagram of the system:

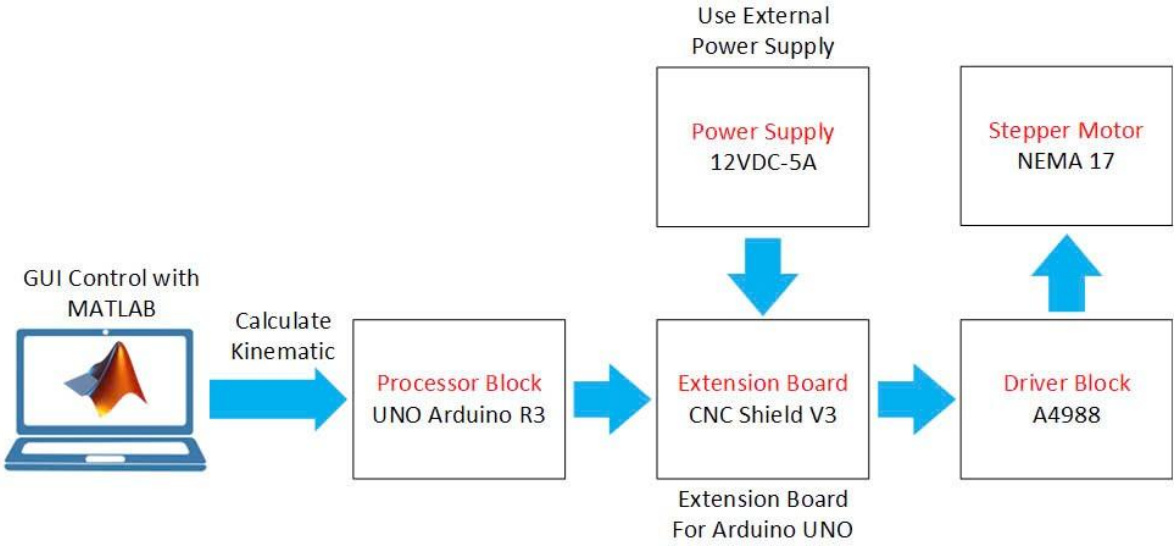


Figure 4.14: Connection diagram

Detailed description:

1. GUI Control with MATLAB: The MATLAB user interface calculates kinematics and sends control commands to the Arduino to control the motors of the robotic arm.

2. Processor Block - Arduino UNO R3: The Arduino UNO R3 receives commands from MATLAB and controls output signals to operate the motors. This serves as the main processor of the system.

3. Extension Board - CNC Shield V3: This is an extension board mounted on the Arduino, capable of controlling stepper motors through drivers. The CNC Shield V3 facilitates connection of control signals from the Arduino to the motor drivers.

4. Driver Block - A4988: The A4988 driver controls the NEMA 17 stepper motor. Each A4988 driver operates one motor, providing control signals like pulse and direction.

5. Stepper Motor - NEMA 17: The NEMA 17 stepper motor generates motion for the robotic arm in horizontal direction. It receives control signals from the A4988 driver to rotate in steps.

6. Power Supply (12VDC-5A): Provides external power to the stepper motors, ensuring they have sufficient energy to operate.

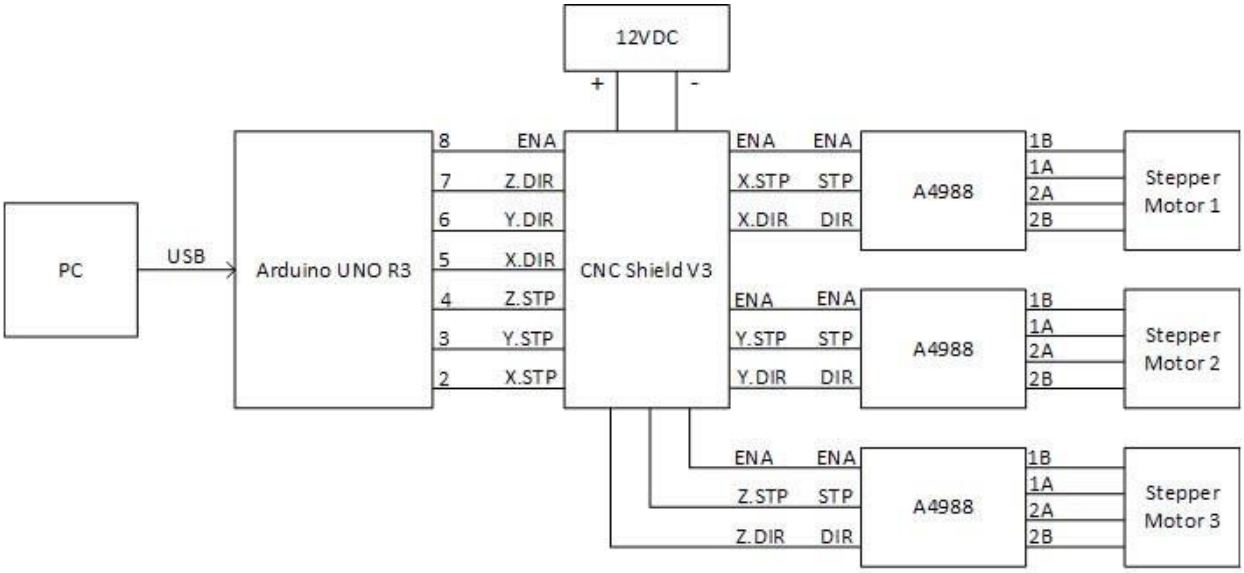


Figure 4.15: Wiring diagram

4.2.3 Camera

A smartphone camera is used in this setup and is connected to MATLAB software via the **DroidCam** application. The camera is mounted on a fixed holder and positioned to cover the designated workspace accurately. The checkerboard grid under the workspace ensures precise alignment and scale calibration for the captured images.

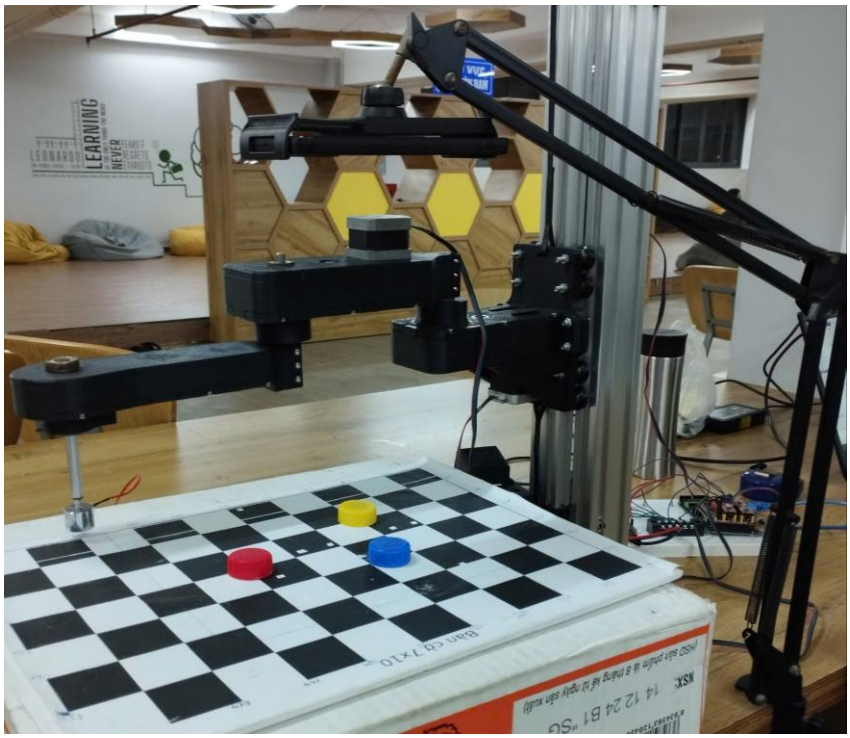


Figure 4.16: Setting up Camera

4.3 SOFTWARE DESIGN

4.3.1 MATLAB Support Package for Arduino Hardware

In this project, with the goal of verifying the kinematics for Robot, we will use MATLAB support packages for Arduino Hardware to communicate between MATLAB software and Arduino.

We can install MATLAB Support Package for Arduino Hardware and based on the links below:

https://www.mathworks.com/matlabcentral/fileexchange/47522-matlab-support-package-for-arduino-hardware

To check the installation, go to Manage Add-Ons, now we will see support package installed as follows:

Figure 4.17: Check the installation

4.3.2 Graphic User Interface Design

MATLAB App Designer is an integrated tool in MATLAB used to create applications (apps) with a graphical user interface (GUI). App Designer provides an intuitive design environment, allowing users to easily build and deploy applications for data analysis, simulations, or specific computational tasks.

Some key features of MATLAB App Designer:

1. Drag-and-Drop Interface
   - Enables users to design the user interface (UI) by dragging and dropping components like buttons, text boxes, tables, graphs, etc.
   - A visual interface helps adjust the position, size, and properties of components easily.
2. Rich Library of UI Components:
   - Includes buttons, checkboxes, menus, sliders, tables, charts, etc.
   - Supports data visualization in various ways.

3. Integrated Code Editor:

- Supports writing MATLAB code to program the behavior of UI components.
- Allows handling user interactions such as button clicks or data input.

4. Tight Integration with MATLAB:

- Easily access and use MATLAB functions, data, and tools.
- Apps can be run within the MATLAB environment or packaged as standalone files (.exe).

5. Responsive Design Support: Enables the interface to adjust automatically for different screen sizes.

6. Easy Sharing and Deployment: Users can share apps by exporting them as MATLAB files (.mlapp) or packaging them into standalone applications.

How to use MATLAB App Designer:

1. Open App Designer:

In MATLAB, go to Home > New > App or type “appdesigner” in the Command Window.

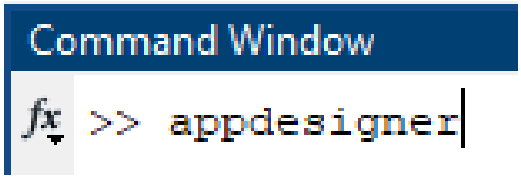


Figure 4.18: Open App Designer

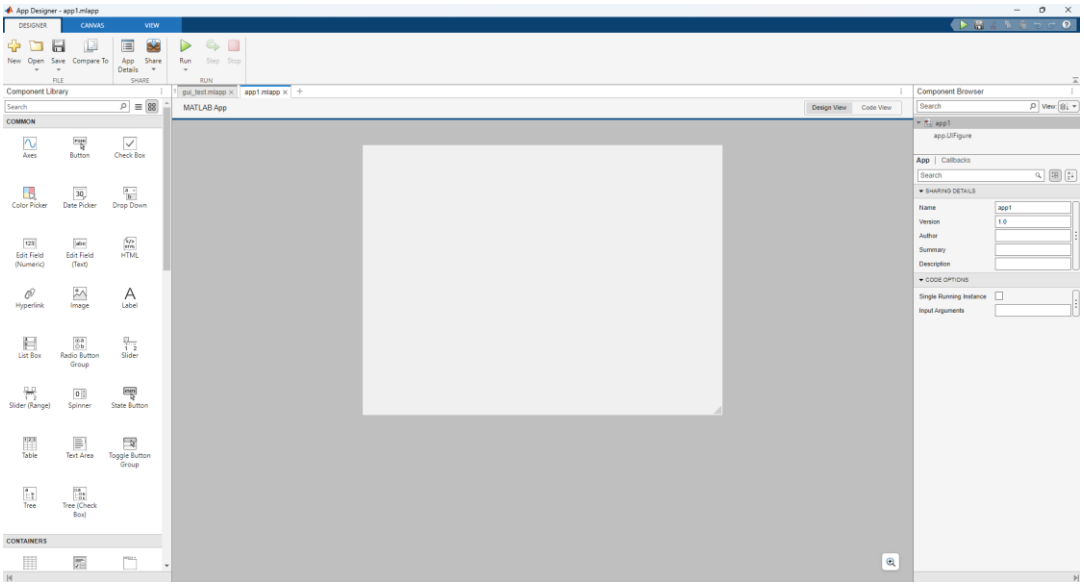


Figure 4.19: App Designer interface

2. Design the Interface:

Drag and drop components from the Component Library into the design area.

Customize the properties of components in the Component Browser.

3. Write Control Logic: Use the Code View to add logic and handle events.

4. Test and Refine:

- Run the app directly in App Designer to test it.
- Refine the interface and code until complete.

5. Save and Share the Application:

- Save the app as a ".mlapp" file.
- Package or share the app with others.

Below is the simple graphic user interface (GUI) of this project:

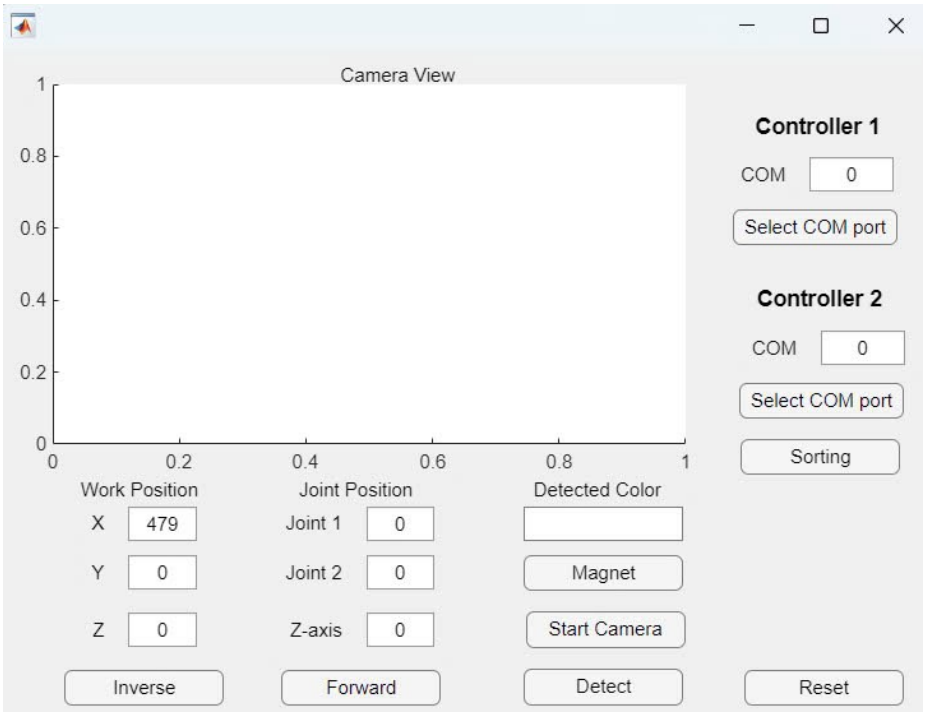


Figure 4.20: Graphic user interface of the robot

The GUI can control the robot by two ways:

- First is controlling the joint directly by typing the joint angles. Then a MATLAB program will calculate the forward kinematics and show the position on the screen.
- Second is controlling by typing the work position. A MATLAB program will calculate the Inverse kinematics and show the position on the screen, then the

program will use the calculated joint angles and control the robot to the work position.

4.4 ALGORITHM

4.4.1 Robot control algorithm

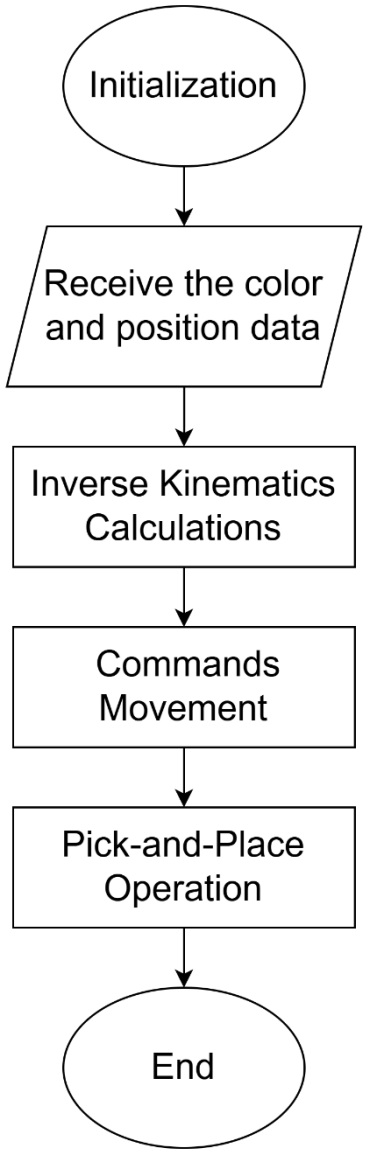


Figure 4.21: Robot control algorithm

Algorithm Description:

1. Initialization

- Establish communication between the MATLAB software and the Arduino controller.

- Initialize the SCARA robotic arm, including setting the home position for all joints.
- Verify that the stepper motors and electromagnetic gripper are functional.

2. Input Data: Receive the color and position data of the detected bottle caps from the image processing system.

3. Inverse Kinematic Calculations: Calculate the required joint angles using inverse kinematics to move the robotic arm from its current position to the target position.

4. Movement Commands

- Send joint angle commands to the Arduino controller through MATLAB.
- Arduino interprets the commands and generates pulse signals to control the stepper motors.

5. Pick-and-Place Operation

- The robotic arm moves to the target position above the detected bottle cap.
- Activate the electromagnetic gripper to pick up the bottle cap.
- Move the robotic arm to the corresponding storage bin based on the bottle cap's color.
- Release the gripper to place the bottle cap into the bin.
- Repeat the process for all detected bottle caps.

6. System Termination: Once all bottle caps are sorted, return the robotic arm to its home position.

4.4.2 Image processing algorithm

The image processing algorithm is designed to classify the colors of objects (red, yellow, blue) based on the HSV color space and determine the position of objects in the image and convert their pixel coordinates to real-world coordinates.

The input of this algorithm is the RGB image captured by the camera connected through DroidCam application, the reference point or initial point in the image and the real word dimensions which the camera capture (width and height in millimeters).

The result or output of this algorithm is the color of detected object color, real-world object position relative to the reference point (in millimeters).

Here are steps to get the color and position of the detected object:

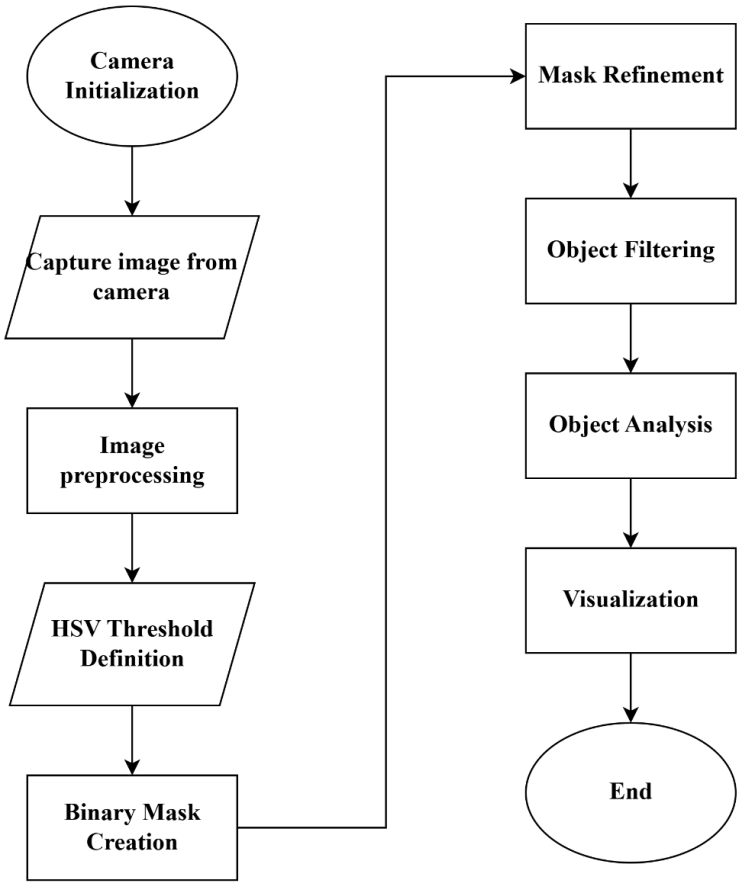


Figure 4.22: Classify and determine object's position program flowchart

Algorithm Description:

- **Step 1:** Camera Initialization
    - At the beginning, we need to initialize the camera object and set its resolution. Next, we define the reference point or initial point in the image. Then calculate the real-world dimensions of the camera's field of view and determine pixel-to-millimeter scaling factors.
- **Step 2:** Capture image from camera
    - In order to increase the accuracy for the image processing, the code only capture and process 1 image or 1 frame at a time.
- **Step 3:** Image preprocessing
    - After capturing a frame from the camera, the frame will be converted from RGB color space to HSV
- **Step 4:** HSV Threshold Definition

    - Each color will have a specific range of HSV (hue, saturation, value) value, to know the range of target color, we can refer online.
- **Step 5:** Binary Mask Creation
    - Generate binary masks (for example: blue_mask, red_mask, yellow_mask) for each color based on the HSV thresholds.. After that, combine the individual masks into a composite mask.
- **Step 6:** Mask Refinement
    - In order to reduce noise of the detected object, we will apply morphological operations (opening and closing) to remove noise and fill gaps in the masks
- **Step 7:** Object Filtering
    - In this step, we will label connected regions in the composite mask and filter objects based on their area (adding min area and max area) to eliminate noise and irrelevant regions.
- **Step 8:** Object Analysis
    - After detecting object, we will need to return color and position of detected object. So, for each detected object, we will compute the bounding box and centroid and calculate the object's relative position in pixels with respect to the reference point. Then, we will convert that relative position in pixels to millimeters, extract HSV values at the centroid and classify the object’s color.
- **Step 9:** Visualization
    - Display the result of image processing like the bounding boxes around detected objects, HSV values and the classified color near the centroid and the relative positions in millimeters

CHAPTER 5: EXPERIMENT RESULT

5.1 DETECTING COLOR EXPERIMENT RESULT

Image processing is used in this system in order to detect and sort objects based on color. So, in this part, image processing program will be tested with 3 different colors: red, yellow and blue.

We will use 3 bottle caps with 3 different colors and test for 10 times each bottle cap to know the accuracy of the image processing program.

- **Case 1: Testing red color**

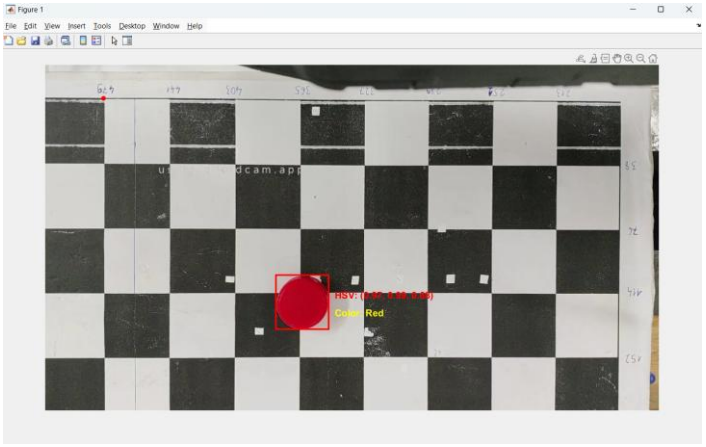


Figure 5.1: Detecting red color using HSV

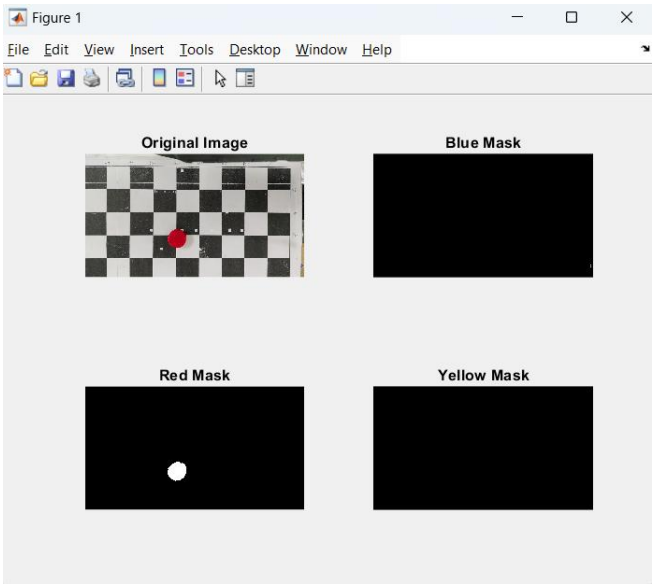


Figure 5.2: HSV mask

The red bottle cap is placed randomly on the checker board like in **Figure 5.1**. And after 10 times testing with 10 different places, the result is shown in **Table 4.**

Table 4: *Testing red color result*

No.	Result
1	Red
2	Red
3	Red
4	Red
5	Red
6	Red
7	Red
8	Red
9	Red
10	Red

From the experiment, the accuracy of detecting red color is high in ideal condition when the lighting is good, camera resolution is high (1920x1080 pixels).

- **Case 2: Testing yellow color**

Figure 5.3: Detecting yellow color using HSV

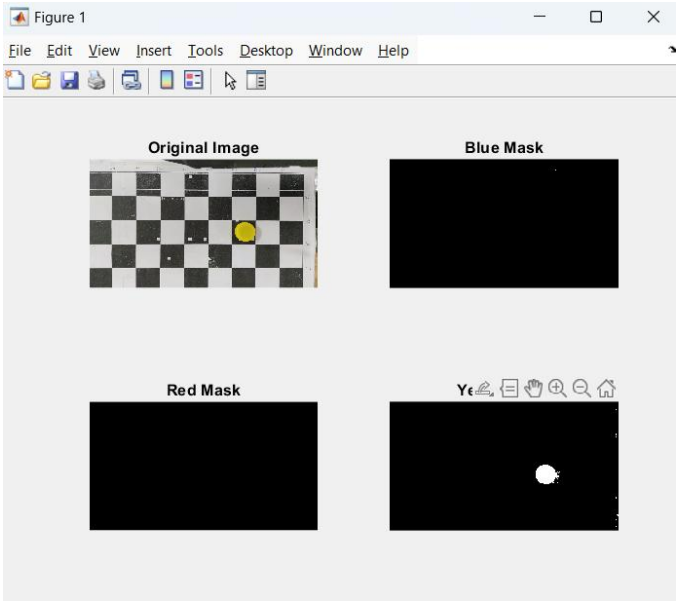


Figure 5.4: HSV mask

The yellow bottle cap is placed randomly on the checker board like in **Figure 5.3**. And after 10 times testing with 10 different places, the result is shown in **Table 5**.

Table 5: *Testing yellow color result*

No.	Result
1	Yellow
2	Yellow
3	Yellow
4	Yellow
5	Yellow
6	Yellow
7	Yellow
8	Yellow
9	Yellow
10	Yellow

Based on the experiment, we can conclude that accuracy of detecting yellow color is high in ideal condition when the lighting is good, camera resolution is high (1920x1080 pixels).

- **Case 3: Testing blue color**

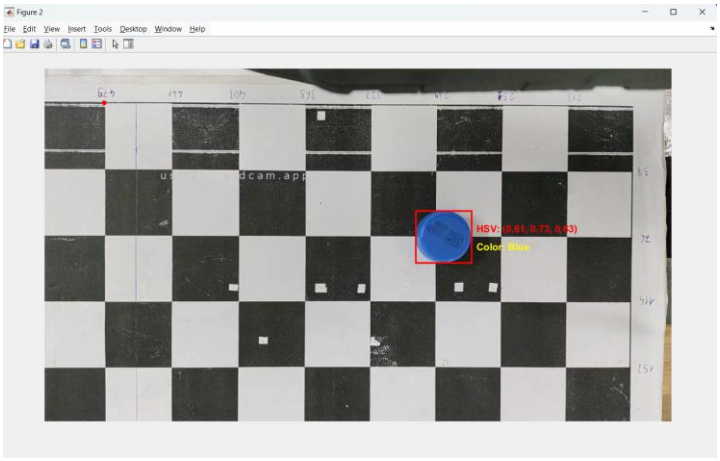


Figure 5.5: Detecting blue color using HSV

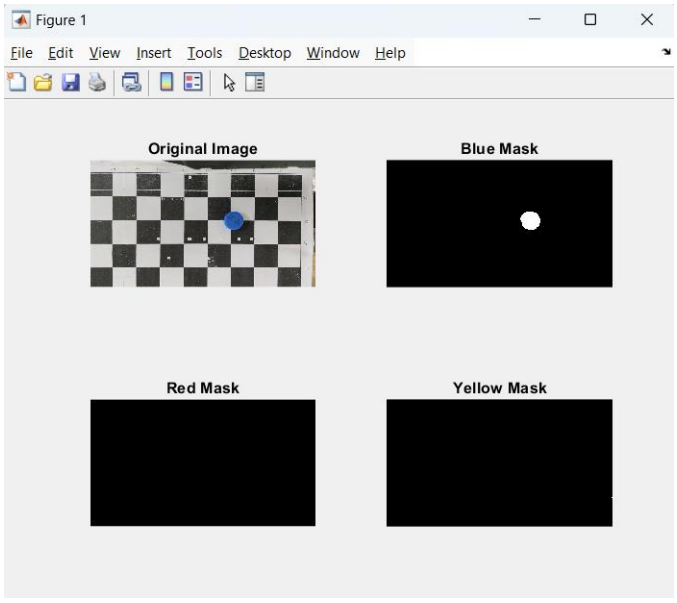


Figure 5.6: HSV mask

The blue bottle cap is placed randomly on the checker board like in **Figure 5.5**. And after 10 times testing with 10 different places, the result is shown in **Table 6**

Table 6: *Testing blue color result*

No.	Result
1	Blue
2	Blue
3	Blue
4	Blue
5	Blue
6	Blue
7	Blue
8	Blue
9	Blue
10	Blue

Based on the experiment, we can conclude that accuracy of detecting yellow color is high in ideal condition when the lighting is good, camera resolution is high (1920x1080 pixels). In other condition like the light is too bright, the result will be unknown.

Table 7: *Color detection testing*

Predicted / Actual	Red bottle cap	Yellow bottle cap	Blue bottle cap
Red	10	0	0
Yellow	0	10	0
Blue	0	0	10

From all 3 cases, the color detecting program using HSV is stable and high accuracy. For 3 colors, the accuracy are 100% which is the ideal condition.

5.2 FORWARD AND INVERSE KINEMATIC EXPERIMENT RESULT

In this part, we will input position for the robot then check the robot pose and position based on the checkerboard as shown in **Figure 5.7**.

Figure 5.7: Using checkerboard to check the robot position

The model will be tested 5 position by using GUI in MATLAB.

- **Case 1: x=403, y=114, z=0**

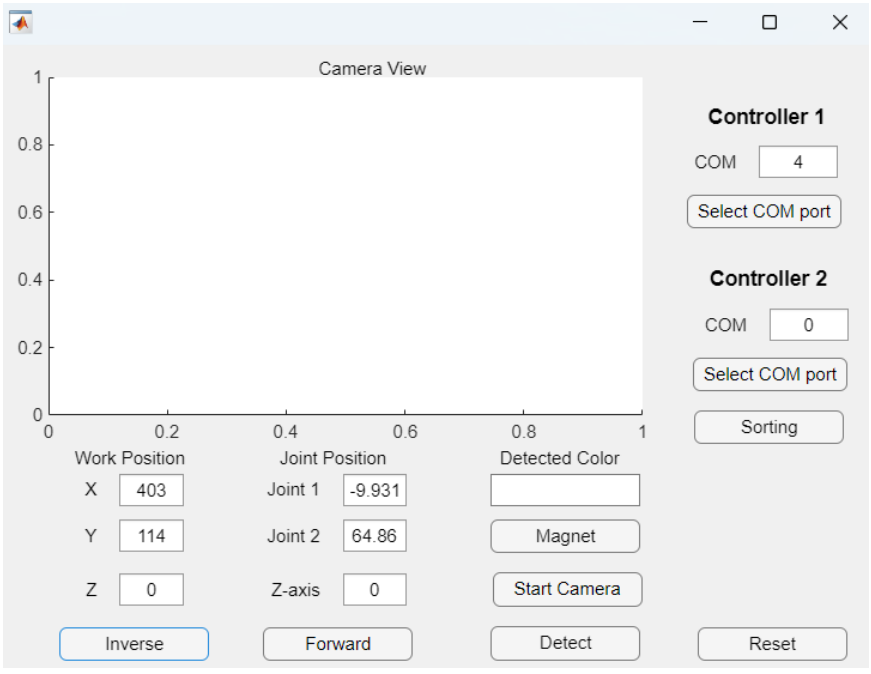


Figure 5.8: Input work position on GUI (case 1)

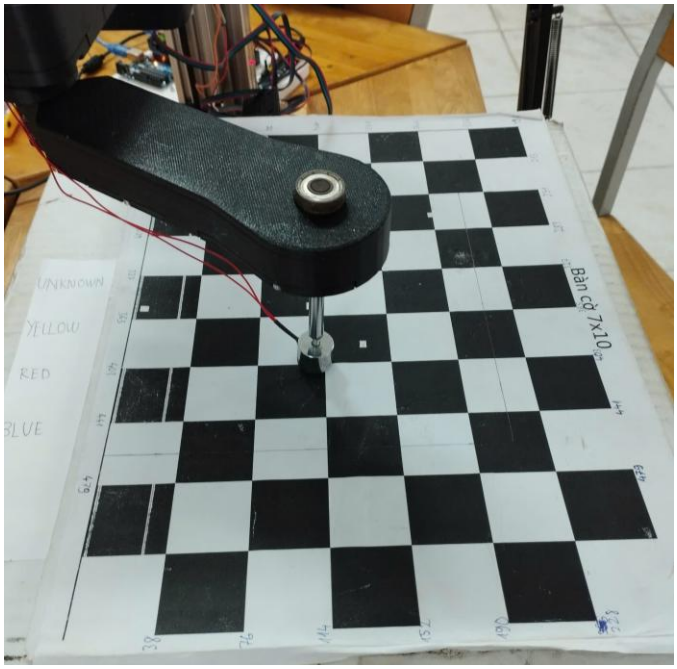


Figure 5.9: Robot position after input work position (case 1)

From **Figure 5.9**, the robot position is almost correct, the error on x-axis is 0 mm and on y-axis is about 2 mm.

- **Case 2: x=251, y=76, z=0**

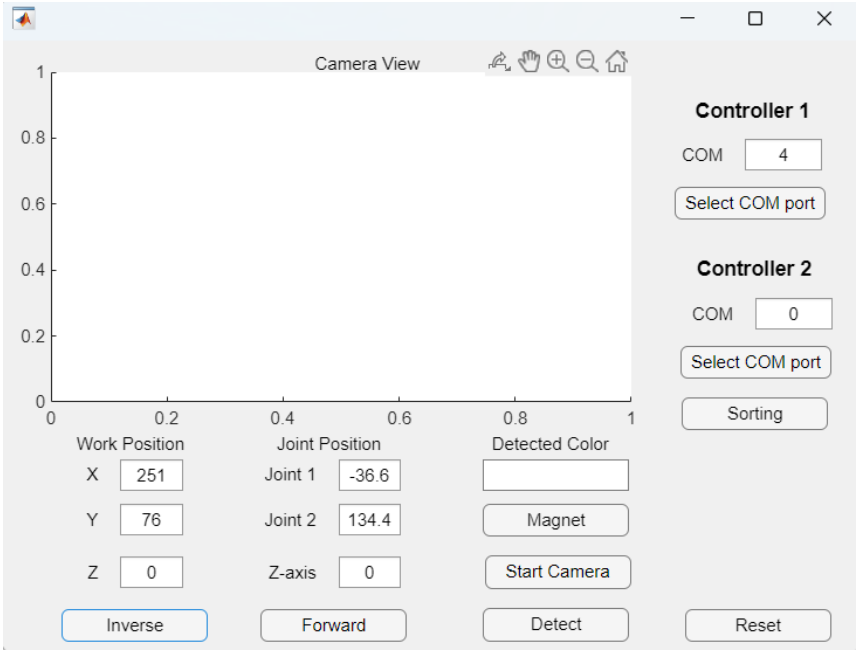


Figure 5.10: Input work position on GUI (case 2)

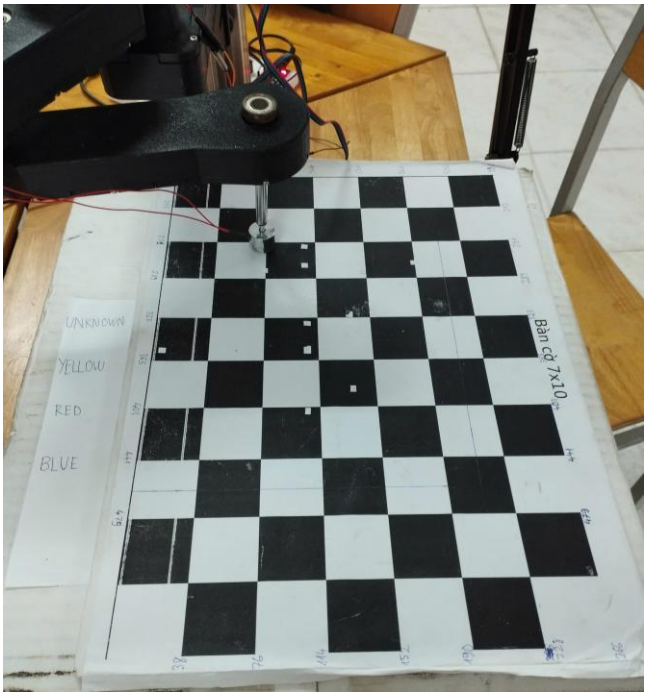


Figure 5.11: Robot position after input work position (case 2)

From **Figure 5.11**, the robot position is correct, the error on x-axis and y-axis is almost 0 mm.

- **Case 3: x=365, y=152, z=0**

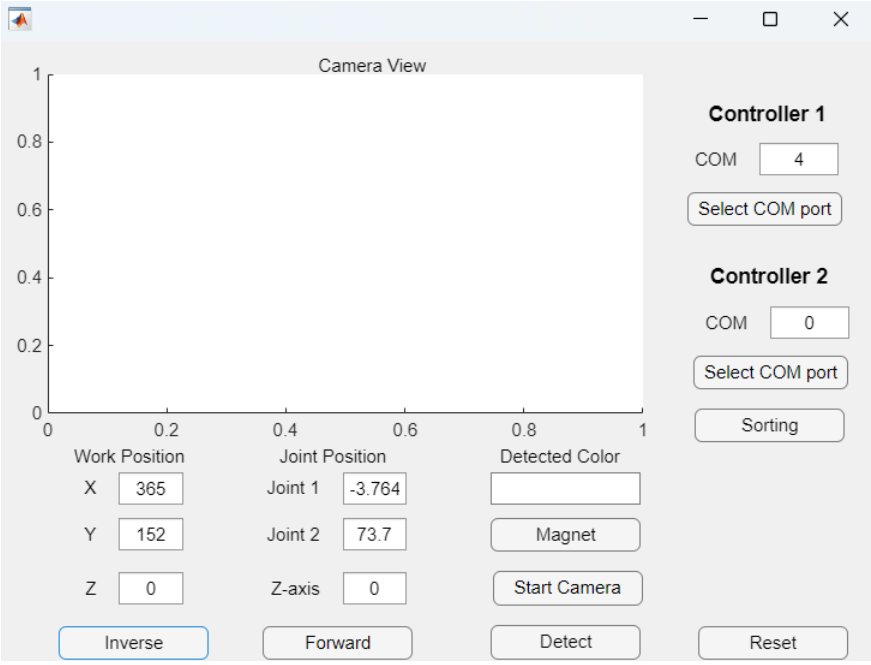


Figure 5.12: Input work position on GUI (case 3)

Figure 5.13: Robot position after input work position (case 3)

From **Figure 5.13**, the robot position is correct, the error on x-axis and y-axis is almost 0 mm.

- **Case 4: x=327, y=190, z=0**

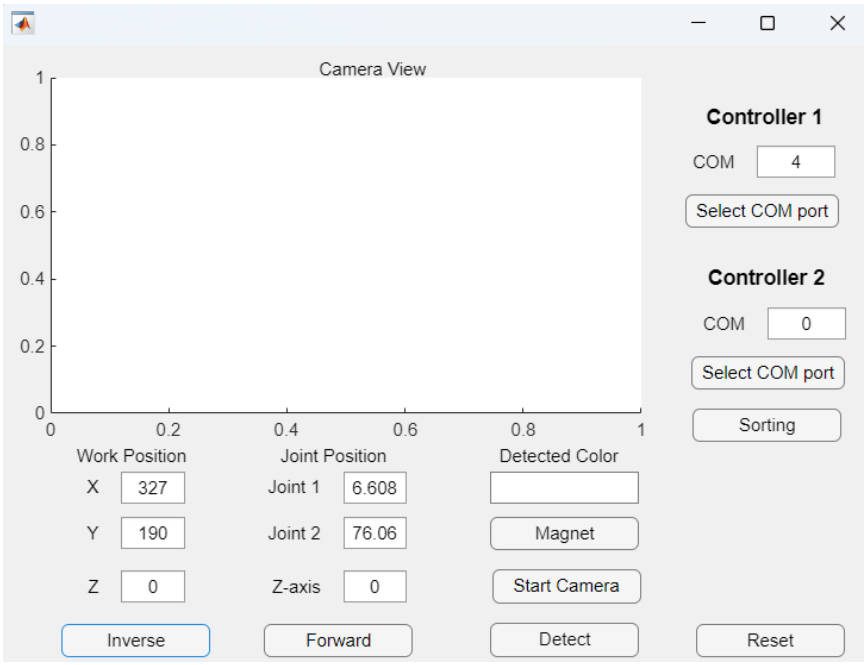


Figure 5.14: Input work position on GUI (case 4)

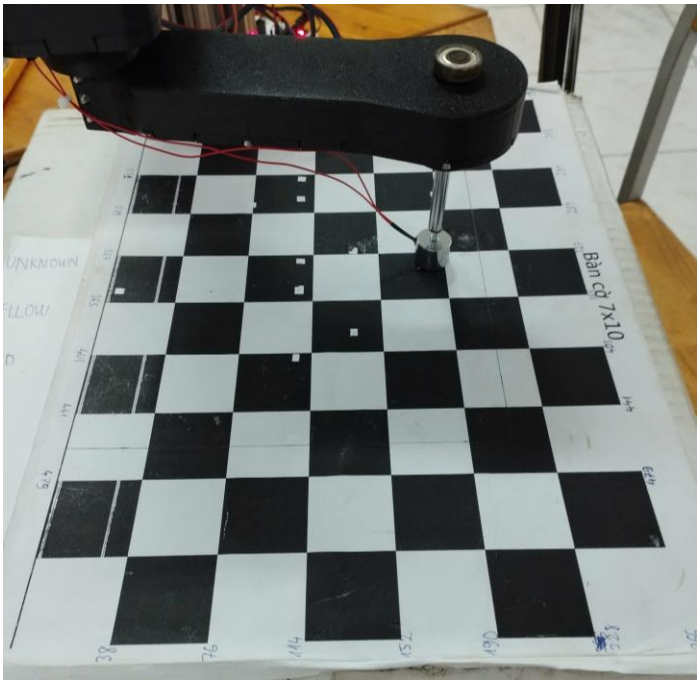


Figure 5.15: Robot position after input work position (case 4)

From **Figure 5.15**, the robot position is almost correct, the error on x-axis is about 2-3 mm and on y-axis is 4-5 mm.

- **Case 5: x=251, y=152, z=0**

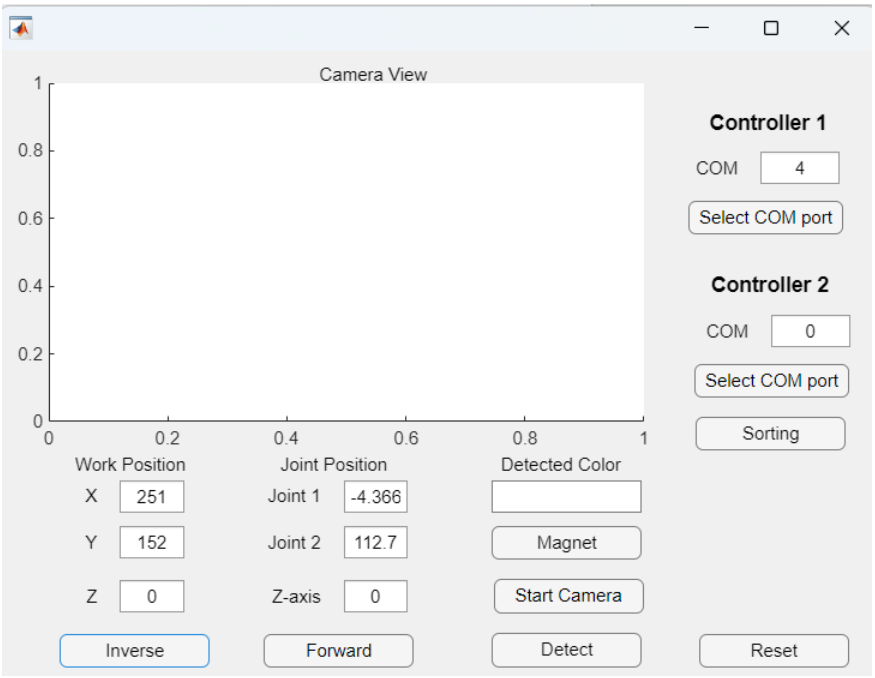


Figure 5.16: Input work position on GUI (case 5)

Figure 5.17: Robot position after input work position (case 5)

From **Figure 5.17**, the robot position is almost correct, the error on x-axis is 0 mm and on y-axis is 2-3 mm.

After this experiment, we saw that the position is not absolutely correct. When the End-Effector come close to the base, the error will increase. The reason for this could be the rounded result when calculating.

5.3 IDENTIFYING POSITION USING IMAGE PROCESSING EXPERIMENT RESULT

To test whether the position return to the laptop is correct or not, we will use 4 bottle caps as shown in **Figure 5.18.**

Figure 5.18: Experiment on deteminning position using image processing

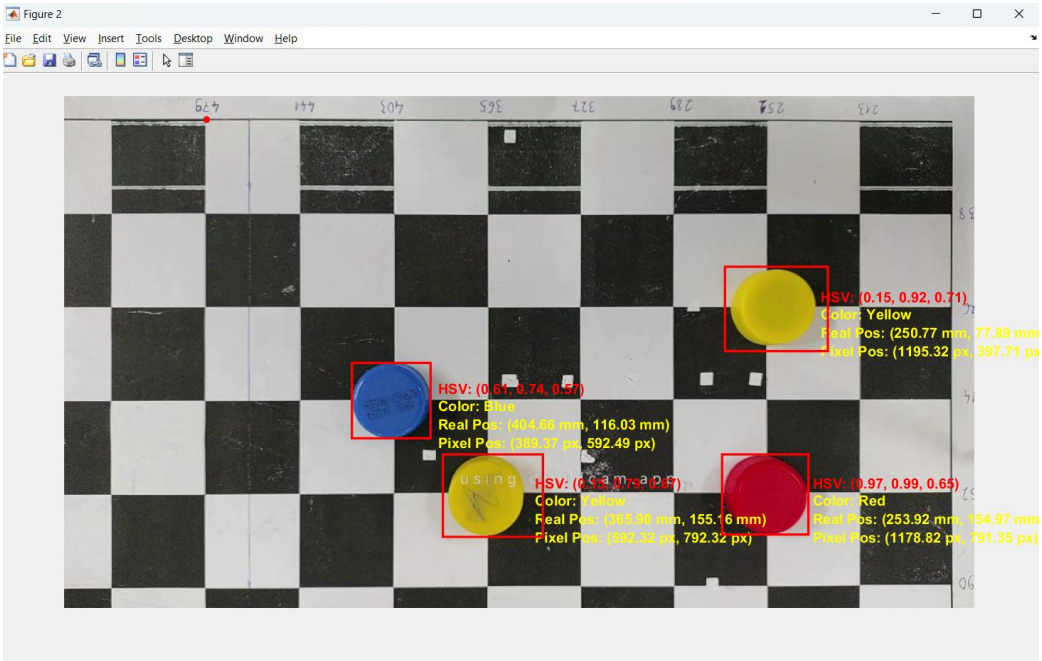


Figure 5.19: Position and color of detected objects

From **Figure 5.19**, the information about color and position of detected object is almost correct. The error on x-axis and y-axis is about 0-2 mm, this error will not affect too much on the robot position.

5.4 Working experiment result

The working of robot is tested by placing bottle caps on the checkerboard as in **Figure 5.19** above. When press the “Sorting” button, the robot will take the position and color of each bottle cap as the input. Then it will go to the position of those bottle caps, turn on the magnet then go to the Blue area and turn off magnet as shown in **Figure 5.20**.

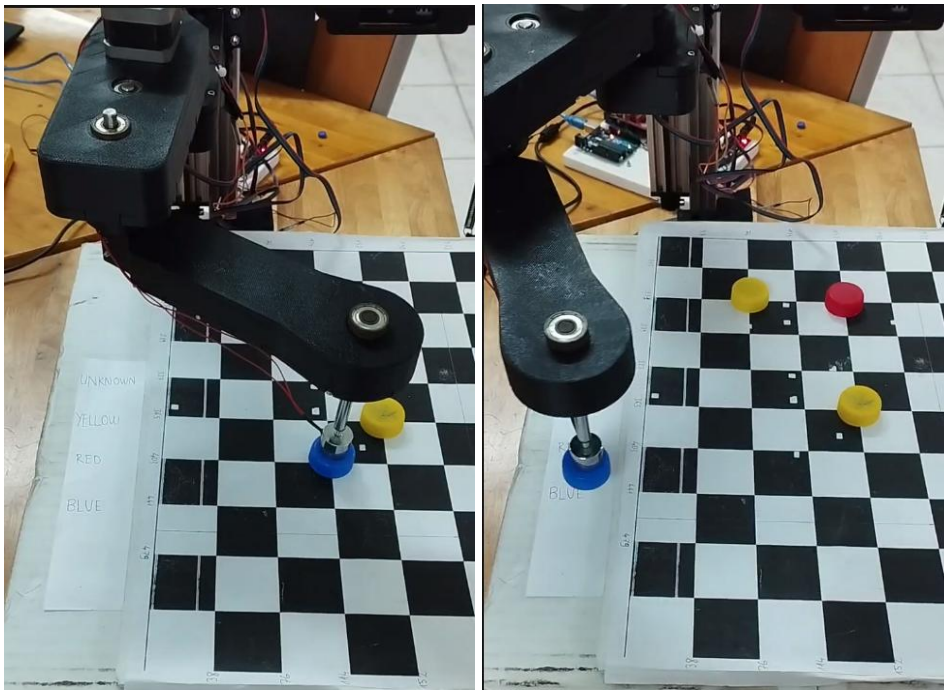


Figure 5.20: Picking blue bottle cap and drop it at Blue as program

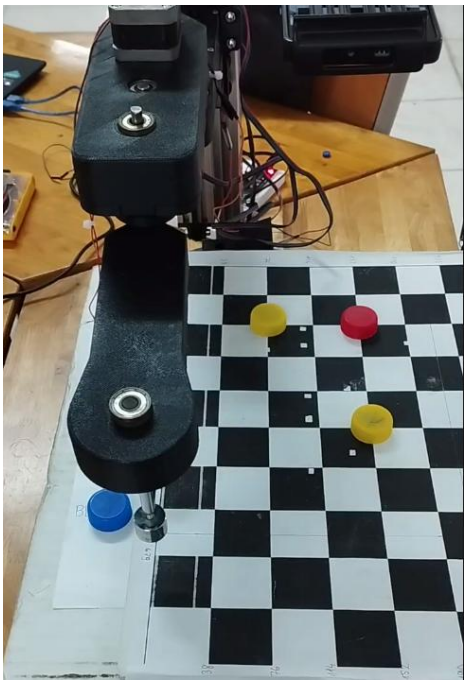


Figure 5.21: Finish sorting

From **Figure 5.21** we can see the robot did not sort all of the bottle cap, this is due to the movement of z-axis is not correct. The robot arm go to the correct position receive from the image processing program, but the robot can not reach the bottle cap as in **Figure 5.22**. The reason could be the surface, it is not completely flat and the slope is greater than 0.

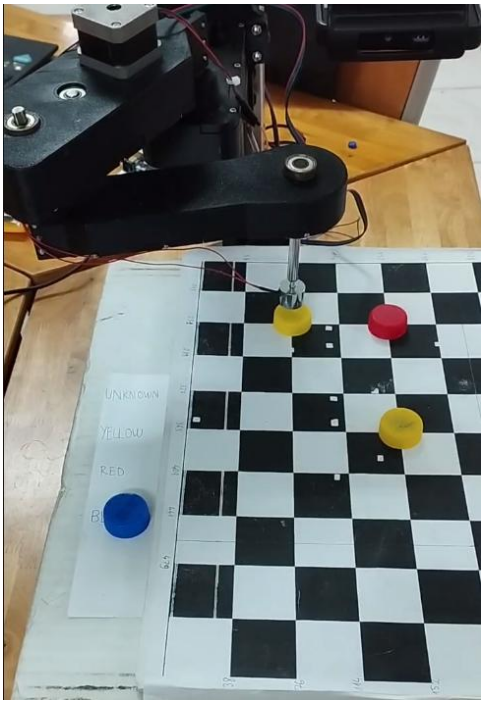


Figure 5.22: Failed due to z-axis

After fixing the problem due to slope of the surface, the robot has sorted all of the bottle caps, here are the results:

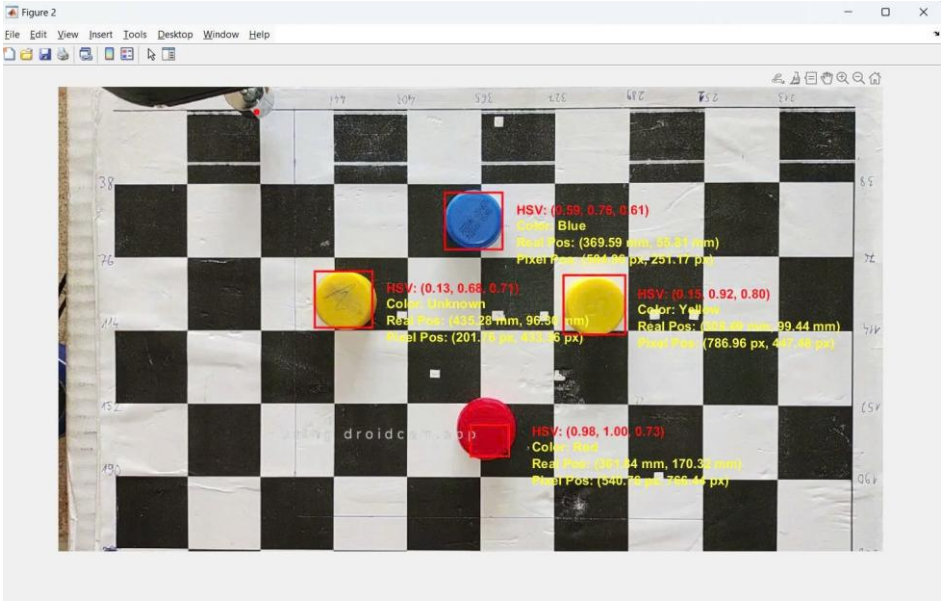


Figure 5.23: Position and color of detected objects

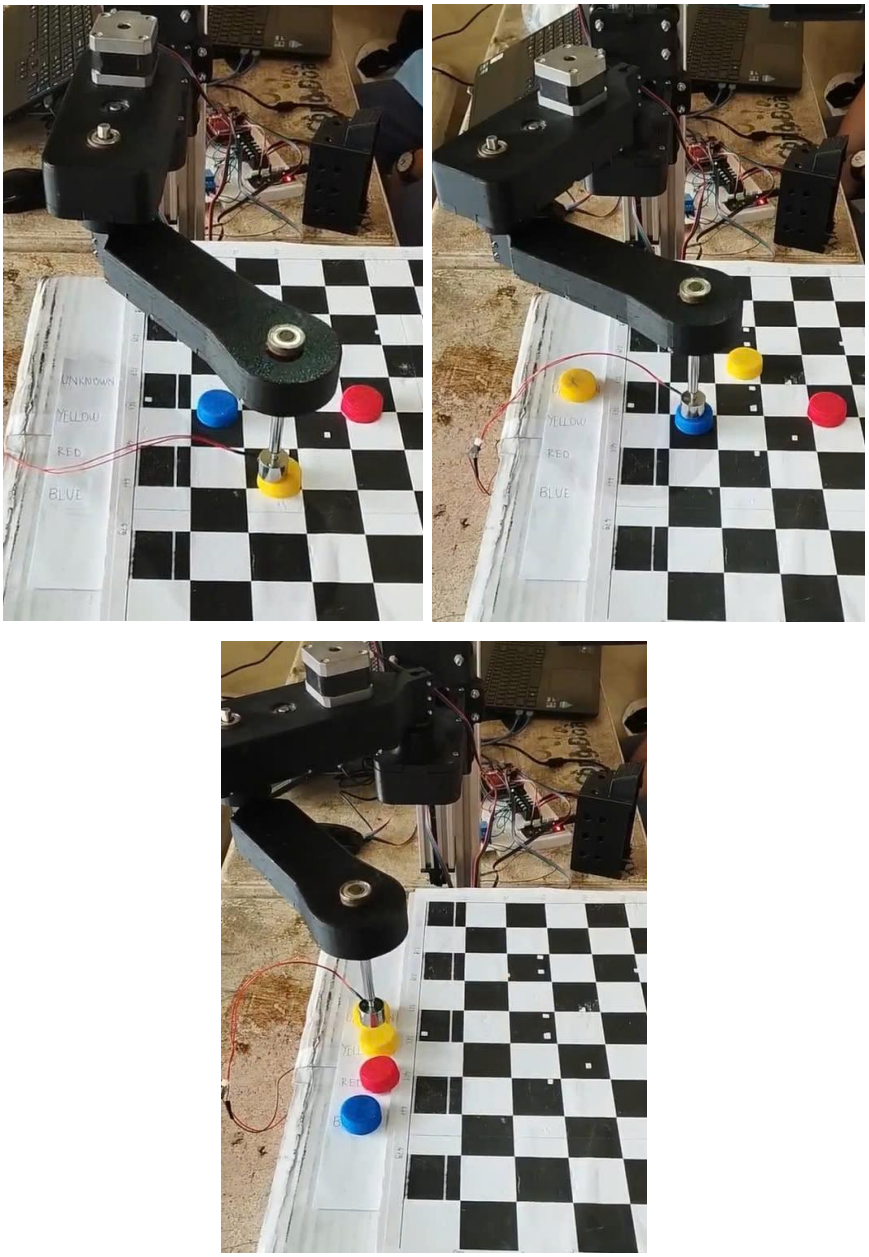


Figure 5.24: Sorting based on color

Based on the result in this chapter, we can conclude that the working of robot is not as good as expected due to the robot arm did not go to the correct position as the input from image processing. But not all functions are failed, the detecting color has the high accuracy in ideal condition, able to detect 3 different colors such as red, yellow and blue. The position identification using image processing program return correct position of detected object with the error fluctuated from 0-2 mm.

CHAPTER 6: CONCLUSION

6.1 ACHIEVED RESULTS

In this project, the team successfully completed the design, integration, and testing of a 3-degree-of-freedom (3-DOF SCARA) robotic system combined with image processing. The system was able to recognize and classify three bottle cap colors: red, blue, and yellow with high accuracy. Updates to the MATLAB GUI interface ensured smooth and effective user interaction, while completing the integration between the robot and the image processing system, enabling the robot to operate seamlessly from recognition to classification.

6.2 LIMITATIONS

Despite achieving encouraging results, the system still has some limitations, such as recognition accuracy being affected by lighting and environmental conditions. The bottle cap sorting speed is not fast enough due to the limitations of stepper motors. The system currently only processes bottle caps based on color and has not yet integrated classification by shape or size. Additionally, advanced features such as robot control in dynamic environments have not been implemented.

6.3 FUTURE DIRECTIONS

In the future, the team plans to improve and expand the system in directions such as enhancing color recognition accuracy by using higher-resolution cameras and integrating advanced image processing algorithms. The team also intends to upgrade the stepper motor system to servo motors to improve speed and accuracy during operation, as well as integrate additional features for recognition by shape, size, and material. Finally, the team proposes developing a more intuitive user interface and integrating connectivity capabilities into automated production lines to increase practical application potential.

REFERENCES

1. J. J. Craig, *Introduction to Robotics Mechanics and Control*. 2005.
2. GLADSTONE J, "Introduction to computer programming," *Brit Chem Eng*, vol. 16, no. 2–3, pp. 206–210, 1971, doi: 10.2307/2003721.
3. S. E. Jewett, *Physics 9th*. doi: 10.5860/choice.34-3910.
4. S. G. Tzafestas, Introduction to Mobile Robot Control. 2013. doi: 10.1016/C2013-0-01365-5.

APPENDIX

1. Arduino code

Control 3 Stepper motor:

```
// Define pin connections
const int dirPin1 = 5;
const int dirPin2 = 6;
const int dirPin3 = 7;
const int stepPin1 = 2;
const int stepPin2 = 3;
const int stepPin3 = 4;
const int Enable = 8;

int Theta1 = 0;
int Theta2 = 0;
int Theta3 = 0;

int steps1 = 0;
int steps2 = 0;
int steps3 = 0;

int ratio1 = 1280/9;
int ratio2 = 1280/9;
int ratio3 = 124.513;

int temp;
int temp2 = 0;
int temp3 = 0;
String content = "";

int data[10];

void setup() {
  // put your setup code here, to run once:
    Serial.begin(9600);
  pinMode(stepPin1, OUTPUT);
  pinMode(dirPin1, OUTPUT);
  pinMode(stepPin2, OUTPUT);
  pinMode(dirPin2, OUTPUT);
  pinMode(stepPin3, OUTPUT);
  pinMode(dirPin3, OUTPUT);
  pinMode(Enable,OUTPUT);
  digitalWrite(Enable,HIGH);
}

void Step_motor_1() {
  if (data[1] > Theta1){
```

```
  // Set motor direction clockwise
  digitalWrite(dirPin1, HIGH);
  digitalWrite(Enable,LOW);

  steps1 = (data[1] - Theta1)*ratio1;
  // Serial.print("Steps1: ");
  // Serial.println(steps1);
  // Spin motor clockwise
  for(int x = 0; x < steps1; x++)
  {
    digitalWrite(stepPin1, HIGH);
    delayMicroseconds(200);
    digitalWrite(stepPin1, LOW);
    delayMicroseconds(200);
  }
  }
  else if (Theta1 > data[1]) {
  // Set motor direction counterclockwise
  digitalWrite(dirPin1, LOW);
  digitalWrite(Enable,LOW);

  steps1 = (Theta1 - data[1])*ratio1;
  // Serial.print("Steps1: ");
  // Serial.println(steps1);
  // Spin motor counterclockwise
  for(int x = 0; x < steps1; x++)
  {
    digitalWrite(stepPin1, HIGH);
    delayMicroseconds(200);
    digitalWrite(stepPin1, LOW);
    delayMicroseconds(200);
  }
  }
  Theta1 = data[1];
  steps1 = 0;
}

void Step_motor_2() {
  if (data[2] > Theta2){
  // Set motor direction clockwise
  digitalWrite(dirPin2, HIGH);
  digitalWrite(Enable,LOW);

  steps2 = (data[2] - Theta2)*ratio2;
  // Serial.print("Steps2: ");
  // Serial.println(steps2);
  // Spin motor clockwise
  for(int x = 0; x < steps2; x++)
```

```
  {
    digitalWrite(stepPin2, HIGH);
    delayMicroseconds(200);
    digitalWrite(stepPin2, LOW);
    delayMicroseconds(200);
  }
  }
  else if (Theta2 > data[2]) {
  // Set motor direction counterclockwise
  digitalWrite(dirPin2, LOW);
  digitalWrite(Enable,LOW);

  steps2 = (Theta2 - data[2])*ratio2;
  // Serial.print("Steps2: ");
  // Serial.println(steps2);
  // Spin motor counterclockwise
  for(int x = 0; x < steps2; x++)
  {
    digitalWrite(stepPin2, HIGH);
    delayMicroseconds(200);
    digitalWrite(stepPin2, LOW);
    delayMicroseconds(200);
  }
  }
  Theta2 = data[2];
  steps2 = 0;
}

void Step_motor_3() {
  if (data[3] > Theta3){
  // Set motor direction clockwise
  digitalWrite(dirPin3, HIGH);
  digitalWrite(Enable,LOW);

  steps3 = (data[3] - Theta3)*ratio3;
  // Serial.print("Steps3: ");
  // Serial.println(steps3);
  // Spin motor clockwise
  for(int x = 0; x < steps3; x++)
  {
    digitalWrite(stepPin3, HIGH);
    delayMicroseconds(200);
    digitalWrite(stepPin3, LOW);
    delayMicroseconds(200);
  }
  }
  else if (Theta3 > data[3]) {
  // Set motor direction counterclockwise
```

```
  digitalWrite(dirPin3, LOW);
  digitalWrite(Enable,LOW);

  steps3 = (Theta3 - data[3])*ratio3;
  // Serial.print("Steps3: ");
  // Serial.println(steps3);
  // Spin motor counterclockwise
  for(int x = 0; x < steps3; x++)
  {
    digitalWrite(stepPin3, HIGH);
    delayMicroseconds(200);
    digitalWrite(stepPin3, LOW);
    delayMicroseconds(200);
  }
  }
  Theta3 = data[3];
  steps3 = 0;
}

void loop() {
  //put your main code here, to run repeatedly:
  if (Serial.available()) {
    content = Serial.readString(); // Read the incomding data from Processing
    // Serial.println(content);
    // Extract the data from the string and put into separate integer
variables (data[] array)
    for (int i = 1; i < 4; i++) {
      int index = content.indexOf(";"); // locate the first ";"
      data[i] = atol(content.substring(0, index).c_str()); //Extract the
number from start to the ";"
      content = content.substring(index + 1); //Remove the number from the
string
    }
    temp=1;
  }

  if (((data[1]<=0)&&(data[1]>-91)&&(data[2]<146)
  &&(data[2]>-91))||((data[1]>=0)&&(data[1]<91)&&(data[2]>-146)
  &&(data[2]<91)))
  {
    if ((temp==1)&&((data[2]*temp3)<0))
    {
      temp2=data[2];
      data[2]=0;
      Step_motor_2();
      data[2]=temp2;
    }
    Step_motor_1();
```

```
    Step_motor_2();
    Step_motor_3();
    if (temp==1)
    {
      Serial.println("1");
      temp=0;
    }
  }
  temp3 = data[2];
  digitalWrite(Enable,HIGH);
}
```

2. MATLAB code

Basic 01:

```
function the = basic_01(a,b,d) %a*sin(the) + b*cos(the) = d;
    alp = atan2(b,a)  ; %sin(alp) = b/sqrt(a*a + b*b); cos(alp) = a/sqrt(a*a +
b*b)
    the1 = atan2(d, sqrt(a*a + b*b - d*d)) - alp;
    the2 = atan2(d, -sqrt(a*a + b*b - d*d)) - alp;
    the = [the1; the2];
```

Transformation matrix:

```
function T=TransMatrix(a,anpha,d,theta)
T=[cos(theta),-sin(theta),0,a;...
    sin(theta)*cos(anpha), cos(theta)*cos(anpha),-sin(anpha),-sin(anpha)*d;...
    sin(theta)*sin(anpha), cos(theta)*sin(anpha),cos(anpha),cos(anpha)*d;...
    0,0,0,1];
end
```

Forward kinematics:

```
function out = fk_3dof(the2, the3, D1)
%% Length
D2 = 146;
% D3 = 20;
D4 = 160;
% D5 = 20;
D6 = 173;
D7 = 0;

%% Forward Kinematic
P3_EE = [D6; 0 ; -D7];
% Input cua FKRobot la bang DH
T01 = TransMatrix(0,0,D1,0);
T12 = TransMatrix(D2,0,0,the2*pi/180);
T23 = TransMatrix(D4,0,0,the3*pi/180);

P = T01*T12*T23*[P3_EE;1];
```

```
Px = P(1,1);
Py = P(2,1);
Pz = P(3,1);
out = [Px Py Pz];
end
```

Inverse kinematics:

```
function out = ik_2dof(Px,Py,Pz)
    D2 = 146;
    D4 = 160;
    D6 = 173;
    D7 = 0;
    l1 = D4;
    l2 = D6;
    k=0;
    z = [Pz;Pz];
    x = Px-D2; y = Py;
    a1 = 2*y*l1; b1 = 2*x*l1; d1 = x*x + y*y + l1*l1 - l2*l2;

    the1 = basic_01(a1, b1, d1);
    the2 = atan2(y-l1*sin(the1), x - l1*cos(the1)) - the1;

    the = [the1 the2]*180/pi;
    the = [the z];

    if (((the(1,1)<=0)&&(the(1,1)>-91)&&(the(1,2)<146)...
            &&(the(1,2)>-91))||((the(1,1)>=0)&&(the(1,1)<91)&&(the(1,2)>-146)...
  &&(the(1,2)<91)))
        out = the(1,1:3);
    elseif (((the(2,1)<=0)&&(the(2,1)>-91)&&(the(2,2)<146)...
            &&(the(2,2)>-91))||((the(2,1)>=0)&&(the(2,1)<91)&&(the(2,2)>-146)...
  &&(the(2,2)<91)))
        out = the(2,1:3);
    else
        out = 0;
    end
    kk1 = the(1,1:3)
    kk2 = the(2,1:3)
end
```

Control the joint angles of robot by sending serial command to Arduino:

```
% a1=serialport("COM4",9600)
% a2=serialport("COM3",9600)

% Box location
zm=20;
xb=441; yb=-38; zb=0;
xr=403; yr=-38; zr=0;
xy=365; yy=-38; zy=0;
xu=327; yu=-38; zu=0;

xh=[];
```

```
for q=1:detected
    if (objects_position_mm(q,4) == 1)
        xh(q,1) = xb; xh(q,2) = yb;
    elseif (objects_position_mm(q,4) == 2)
        xh(q,1) = xr; xh(q,2) = yr;
    elseif (objects_position_mm(q,4) == 3)centroids
        xh(q,1) = xy; xh(q,2) = yy;
    elseif (objects_position_mm(q,4) == 4)
        xh(q,1) = xu; xh(q,2) = yu;
    end
end

if (detected == 1)
    x1=objects_position_mm(1,2); y1=objects_position_mm(1,3); z1=0;
    m=[479 0 zm 0;
        x1 y1 zm 1;
        x1 y1 z1 1;
        x1 y1 zm 1;
        xh(1,1) xh(1,2) zm 0;

        479 0 zm 0;
        479 0 0 0];
elseif (detected == 2)
    x1=objects_position_mm(1,2); y1=objects_position_mm(1,3); z1=0;
    x2=objects_position_mm(2,2); y2=objects_position_mm(2,3); z2=0;

    m=[479 0 zm 0;
        x1 y1 zm 1;
        x1 y1 z1 1;
        x1 y1 zm 1;
        xh(1,1) xh(1,2) zm 0;

        x2 y2 zm 1;
        x2 y2 z2 1;
        x2 y2 zm 1;
        xh(2,1) xh(2,2) zm 0

        479 0 zm 0;
        479 0 0 0];
elseif (detected == 3)
    x1=objects_position_mm(1,2); y1=objects_position_mm(1,3); z1=0;
    x2=objects_position_mm(2,2); y2=objects_position_mm(2,3); z2=0;
    x3=objects_position_mm(3,2); y3=objects_position_mm(3,3); z3=0;

    m=[479 0 zm 0;
        x1 y1 zm 1;
        x1 y1 z1 1;
        x1 y1 zm 1;
        xh(1,1) xh(1,2) zm 0;

        x2 y2 zm 1;
        x2 y2 z2 1;
        x2 y2 zm 1;
        xh(2,1) xh(2,2) zm 0

        x3 y3 zm 1;
        x3 y3 z3 1;
        x3 y3 zm 1;
        xh(3,1) xh(3,2) zm 0
```

```
        479 0 zm 0;
        479 0 0 0];
elseif (detected == 4)
    x1=objects_position_mm(1,2); y1=objects_position_mm(1,3); z1=0;
    x2=objects_position_mm(2,2); y2=objects_position_mm(2,3); z2=0;
    x3=objects_position_mm(3,2); y3=objects_position_mm(3,3); z3=0;
    x4=objects_position_mm(4,2); y4=objects_position_mm(4,3); z4=0;

    m=[479 0 zm 0;
        x1 y1 zm 1;
        x1 y1 z1 1;
        x1 y1 zm 1;
        xh(1,1) xh(1,2) zm 0;

        x2 y2 zm 1;
        x2 y2 z2 1;
        x2 y2 zm 1;
        xh(2,1) xh(2,2) zm 0

        x3 y3 zm 1;
        x3 y3 z3 1;
        x3 y3 zm 1;
        xh(3,1) xh(3,2) zm 0

        x4 y4 zm 1;
        x4 y4 z4 1;
        x4 y4 zm 1;
        xh(4,1) xh(4,2) zm 0

        479 0 zm 0;
        479 0 0 0];
elseif (detected == 5)
    x1=objects_position_mm(1,2); y1=objects_position_mm(1,3); z1=0;
    x2=objects_position_mm(2,2); y2=objects_position_mm(2,3); z2=0;
    x3=objects_position_mm(3,2); y3=objects_position_mm(3,3); z3=0;
    x4=objects_position_mm(4,2); y4=objects_position_mm(4,3); z4=0;
    x5=objects_position_mm(5,2); y5=objects_position_mm(5,3); z5=0;

    m=[479 0 zm 0;
        x1 y1 zm 1;
        x1 y1 z1 1;
        x1 y1 zm 1;
        xh(1,1) xh(1,2) zm 0;

        x2 y2 zm 1;
        x2 y2 z2 1;
        x2 y2 zm 1;
        xh(2,1) xh(2,2) zm 0

        x3 y3 zm 1;
        x3 y3 z3 1;
        x3 y3 zm 1;
        xh(3,1) xh(3,2) zm 0

        x4 y4 zm 1;
        x4 y4 z4 1;
        x4 y4 zm 1;
        xh(4,1) xh(4,2) zm 0
```

```
        x5 y5 zm 1;
        x5 y5 z5 1;
        x5 y5 zm 1;
        xh(5,1) xh(5,2) zm 0

        479 0 zm 0;
        479 0 0 0];
end

d=[];
n=height(m);
for i=1:n
    k=ik_2dof(m(i,1),m(i,2),m(i,3));
    k=fix(k);
    d=[d;k];
end

for i=1:n
    theta1=d(i,1);
    str1=int2str(theta1);
    theta2=d(i,2);
    str2=int2str(theta2);
    theta3=d(i,3);
    str3=int2str(theta3);
    nc = m(i,4);
    str4=int2str(nc);
    str=append(str1,';',str2,';',str3,';',str4);
    writeline(a1,str);
    b=0;
    while b==0
        c=readline(a1);
        if (nc == 0)
            writeline(a2,'0');
        elseif (nc == 1)
            writeline(a2,'1');
        end
        b=str2double(c);
    end
end
```

Graphic user interface for buttons and display:

```
classdef gui_test < matlab.ui.componentcontainer.ComponentContainer

    % Properties that correspond to underlying components
    properties (Access = private, Transient, NonCopyable)
        Controller2Label       matlab.ui.control.Label
        Controller1Label       matlab.ui.control.Label
        COMEditField_2         matlab.ui.control.NumericEditField
        COMEditFieldLabel_2    matlab.ui.control.Label
        SelectCOMportButton_2  matlab.ui.control.Button
        MagnetButton           matlab.ui.control.StateButton
        ResetButton            matlab.ui.control.Button
        COMEditField           matlab.ui.control.NumericEditField
        COMEditFieldLabel      matlab.ui.control.Label
        SelectCOMportButton    matlab.ui.control.Button
```

```
        DetectedColorLabel      matlab.ui.control.Label
        EditField_2             matlab.ui.control.EditField
        DetectButton            matlab.ui.control.Button
        CameraViewLabel         matlab.ui.control.Label
        StartCameraButton       matlab.ui.control.Button
        ZEditField              matlab.ui.control.NumericEditField
        ZEditFieldLabel         matlab.ui.control.Label
        YEditField              matlab.ui.control.NumericEditField
        YEditFieldLabel         matlab.ui.control.Label
        XEditField              matlab.ui.control.NumericEditField
        XEditFieldLabel         matlab.ui.control.Label
        WorkPositionLabel       matlab.ui.control.Label
        InverseButton_2         matlab.ui.control.Button
        JointPositionLabel      matlab.ui.control.Label
        ZaxisEditField          matlab.ui.control.NumericEditField
        ZaxisEditFieldLabel     matlab.ui.control.Label
        Joint2EditField         matlab.ui.control.NumericEditField
        Joint2EditFieldLabel    matlab.ui.control.Label
        Joint1EditField         matlab.ui.control.NumericEditField
        Joint1EditFieldLabel    matlab.ui.control.Label
        ForwardButton           matlab.ui.control.Button
        UIAxes                  matlab.ui.control.UIAxes
    end

    properties (Access = private)
        cam
    end

    properties (Access = public)
        a
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function postSetupFcn(comp)

        end

        % Button pushed function: ForwardButton
        function ForwardButtonPushed2(comp, event)
            Theta1=comp.Joint1EditField.Value;
            Theta2=comp.Joint2EditField.Value;
            Theta3=comp.ZaxisEditField.Value;
            assignin('base','inTheta1',Theta1);
            assignin('base','inTheta2',Theta2);
            assignin('base','inTheta3',Theta3);
            out=fk_3dof(Theta1,Theta2,Theta3);
            comp.XEditField.Value=out(1,1);
            comp.YEditField.Value=out(1,2);
            comp.ZEditField.Value=out(1,3);
            evalin("base",'FK_control');
        end

        % Button pushed function: InverseButton_2
        function InverseButton_2Pushed(comp, event)
            X=comp.XEditField.Value;
            Y=comp.YEditField.Value;
```

```
        Z=comp.ZEditField.Value;
        out=ik_2dof(X,Y,Z);
        comp.Joint1EditField.Value=out(1,1);
        comp.Joint2EditField.Value=out(1,2);
        comp.ZaxisEditField.Value=out(1,3);
        assignin('base','inTheta1',out(1,1));
        assignin('base','inTheta2',out(1,2));
        assignin('base','inTheta3',out(1,3));
        evalin("base",'FK_control');
    end

    % Button pushed function: StartCameraButton
    function StartCameraButtonPushed(comp, event)
        comp.cam = webcam(1);
        frame = snapshot(comp.cam);
        im = image(comp.UIAxes,zeros(size(frame),'uint8'));
        axis(comp.UIAxes,'image');

        preview(comp.cam,im);
        evalin("base",'setup_cam');
    end

    % Button pushed function: DetectButton
    function DetectButtonPushed(comp, event)
        evalin("base",'detect_pos');
        comp.XEditField.Value=evalin("base",'position(1,1)');
        comp.YEditField.Value=evalin("base",'position(1,2)');
        comp.ZEditField.Value=0;
        comp.EditField_2.Value = evalin("base",'color');
    end

    % Button pushed function: SelectCOMportButton
    function SelectCOMportButtonPushed(comp, event)
        c=comp.COMEditField.Value
        c=int2str(c);
        str=append("COM",c);
        a1=serialport(str,9600);
        assignin('base','a1',a1);
    end

    % Button pushed function: ResetButton
    function ResetButtonPushed(comp, event)
        evalin('base','Reset_button');
    end

    % Value changed function: MagnetButton
    function MagnetButtonValueChanged(comp, event)
        value = comp.MagnetButton.Value;
        if value==0
            nc=0;
        end
        if value==1
            nc=1;
        end
        assignin('base','inNC',nc);
        evalin('base','Magnet');
    end

    % Button pushed function: SelectCOMportButton_2
```

```
        function SelectCOMportButton_2Pushed(comp, event)
            c2=comp.COMEditField_2.Value
            c2=int2str(c2);
            str=append("COM",c2);
            a2=serialport(str,9600);
            assignin('base','a2',a2);
        end

        % Value changed function: COMEditField
        function COMEditFieldValueChanged(comp, event)
            value = comp.COMEditField.Value;

        end

        % Value changed function: COMEditField_2
        function COMEditField_2ValueChanged(comp, event)
            value = comp.COMEditField_2.Value;

        end
    end

    methods (Access = protected)

        % Code that executes when the value of a public property is changed
        function update(comp)
            % Use this function to update the underlying components

        end

        % Create the underlying components
        function setup(comp)

            comp.Position = [1 1 585 417];
            comp.BackgroundColor = [0.94 0.94 0.94];

            % Create UIAxes
            comp.UIAxes = uiaxes(comp);
            comp.UIAxes.Position = [14 156 426 250];

            % Create ForwardButton
            comp.ForwardButton = uibutton(comp, 'push');
            comp.ForwardButton.ButtonPushedFcn =
matlab.apps.createCallbackFcn(comp, @ForwardButtonPushed2, true);
            comp.ForwardButton.Position = [178 10 100 22];
            comp.ForwardButton.Text = 'Forward';

            % Create Joint1EditFieldLabel
            comp.Joint1EditFieldLabel = uilabel(comp);
            comp.Joint1EditFieldLabel.HorizontalAlignment = 'right';
            comp.Joint1EditFieldLabel.Position = [176 113 40 22];
            comp.Joint1EditFieldLabel.Text = 'Joint 1';

            % Create Joint1EditField
            comp.Joint1EditField = uieditfield(comp, 'numeric');
            comp.Joint1EditField.Tag = 'Theta1';
            comp.Joint1EditField.HorizontalAlignment = 'center';
            comp.Joint1EditField.Position = [231 113 43 22];

            % Create Joint2EditFieldLabel
```

```
            comp.Joint2EditFieldLabel = uilabel(comp);
            comp.Joint2EditFieldLabel.HorizontalAlignment = 'right';
            comp.Joint2EditFieldLabel.Position = [176 82 40 22];
            comp.Joint2EditFieldLabel.Text = 'Joint 2';

            % Create Joint2EditField
            comp.Joint2EditField = uieditfield(comp, 'numeric');
            comp.Joint2EditField.Tag = 'Theta2';
            comp.Joint2EditField.HorizontalAlignment = 'center';
            comp.Joint2EditField.Position = [231 82 43 22];

            % Create ZaxisEditFieldLabel
            comp.ZaxisEditFieldLabel = uilabel(comp);
            comp.ZaxisEditFieldLabel.HorizontalAlignment = 'right';
            comp.ZaxisEditFieldLabel.Position = [178 46 38 22];
            comp.ZaxisEditFieldLabel.Text = 'Z-axis';

            % Create ZaxisEditField
            comp.ZaxisEditField = uieditfield(comp, 'numeric');
            comp.ZaxisEditField.Limits = [0 Inf];
            comp.ZaxisEditField.Tag = 'Theta3';
            comp.ZaxisEditField.HorizontalAlignment = 'center';
            comp.ZaxisEditField.Position = [231 46 43 22];

            % Create JointPositionLabel
            comp.JointPositionLabel = uilabel(comp);
            comp.JointPositionLabel.Position = [189 134 76 22];
            comp.JointPositionLabel.Text = 'Joint Position';

            % Create InverseButton_2
            comp.InverseButton_2 = uibutton(comp, 'push');
            comp.InverseButton_2.ButtonPushedFcn =
matlab.apps.createCallbackFcn(comp, @InverseButton_2Pushed, true);
            comp.InverseButton_2.Position = [42 10 100 22];
            comp.InverseButton_2.Text = 'Inverse';

            % Create WorkPositionLabel
            comp.WorkPositionLabel = uilabel(comp);
            comp.WorkPositionLabel.Position = [52 134 79 22];
            comp.WorkPositionLabel.Text = 'Work Position';

            % Create XEditFieldLabel
            comp.XEditFieldLabel = uilabel(comp);
            comp.XEditFieldLabel.HorizontalAlignment = 'right';
            comp.XEditFieldLabel.Position = [42 113 25 22];
            comp.XEditFieldLabel.Text = 'X';

            % Create XEditField
            comp.XEditField = uieditfield(comp, 'numeric');
            comp.XEditField.Tag = 'X';
            comp.XEditField.HorizontalAlignment = 'center';
            comp.XEditField.Position = [82 113 43 22];
            comp.XEditField.Value = 479;

            % Create YEditFieldLabel
            comp.YEditFieldLabel = uilabel(comp);
            comp.YEditFieldLabel.HorizontalAlignment = 'right';
            comp.YEditFieldLabel.Position = [42 82 25 22];
            comp.YEditFieldLabel.Text = 'Y';
```

```
            % Create YEditField
            comp.YEditField = uieditfield(comp, 'numeric');
            comp.YEditField.Tag = 'Y';
            comp.YEditField.HorizontalAlignment = 'center';
            comp.YEditField.Position = [82 82 43 22];

            % Create ZEditFieldLabel
            comp.ZEditFieldLabel = uilabel(comp);
            comp.ZEditFieldLabel.HorizontalAlignment = 'right';
            comp.ZEditFieldLabel.Position = [42 46 25 22];
            comp.ZEditFieldLabel.Text = 'Z';

            % Create ZEditField
            comp.ZEditField = uieditfield(comp, 'numeric');
            comp.ZEditField.Limits = [0 Inf];
            comp.ZEditField.Tag = 'Z';
            comp.ZEditField.HorizontalAlignment = 'center';
            comp.ZEditField.Position = [82 46 43 22];

            % Create StartCameraButton
            comp.StartCameraButton = uibutton(comp, 'push');
            comp.StartCameraButton.ButtonPushedFcn =
matlab.apps.createCallbackFcn(comp, @StartCameraButtonPushed, true);
            comp.StartCameraButton.Tag = 'Start_camera';
            comp.StartCameraButton.Position = [331 46 100 23];
            comp.StartCameraButton.Text = 'Start Camera';

            % Create CameraViewLabel
            comp.CameraViewLabel = uilabel(comp);
            comp.CameraViewLabel.Position = [215 394 77 22];
            comp.CameraViewLabel.Text = 'Camera View';

            % Create DetectButton
            comp.DetectButton = uibutton(comp, 'push');
            comp.DetectButton.ButtonPushedFcn =
matlab.apps.createCallbackFcn(comp, @DetectButtonPushed, true);
            comp.DetectButton.Tag = 'Snapshot';
            comp.DetectButton.Position = [330 10 100 23];
            comp.DetectButton.Text = 'Detect';

            % Create EditField_2
            comp.EditField_2 = uieditfield(comp, 'text');
            comp.EditField_2.HorizontalAlignment = 'center';
            comp.EditField_2.Position = [330 113 100 22];

            % Create DetectedColorLabel
            comp.DetectedColorLabel = uilabel(comp);
            comp.DetectedColorLabel.Position = [336 134 85 22];
            comp.DetectedColorLabel.Text = 'Detected Color';

            % Create SelectCOMportButton
            comp.SelectCOMportButton = uibutton(comp, 'push');
            comp.SelectCOMportButton.ButtonPushedFcn =
matlab.apps.createCallbackFcn(comp, @SelectCOMportButtonPushed, true);
            comp.SelectCOMportButton.Position = [461 299 103 22];
            comp.SelectCOMportButton.Text = 'Select COM port';

            % Create COMEditFieldLabel
```

```
            comp.COMEditFieldLabel = uilabel(comp);
            comp.COMEditFieldLabel.HorizontalAlignment = 'right';
            comp.COMEditFieldLabel.Position = [461 332 33 22];
            comp.COMEditFieldLabel.Text = 'COM';

            % Create COMEditField
            comp.COMEditField = uieditfield(comp, 'numeric');
            comp.COMEditField.ValueChangedFcn =
matlab.apps.createCallbackFcn(comp, @COMEditFieldValueChanged, true);
            comp.COMEditField.HorizontalAlignment = 'center';
            comp.COMEditField.Position = [509 332 53 22];

            % Create ResetButton
            comp.ResetButton = uibutton(comp, 'push');
            comp.ResetButton.ButtonPushedFcn = matlab.apps.createCallbackFcn(comp,
@ResetButtonPushed, true);
            comp.ResetButton.Position = [468 10 100 22];
            comp.ResetButton.Text = 'Reset';

            % Create MagnetButton
            comp.MagnetButton = uibutton(comp, 'state');
            comp.MagnetButton.ValueChangedFcn =
matlab.apps.createCallbackFcn(comp, @MagnetButtonValueChanged, true);
            comp.MagnetButton.Text = 'Magnet';
            comp.MagnetButton.Position = [330 82 100 22];

            % Create SelectCOMportButton_2
            comp.SelectCOMportButton_2 = uibutton(comp, 'push');
            comp.SelectCOMportButton_2.ButtonPushedFcn =
matlab.apps.createCallbackFcn(comp, @SelectCOMportButton_2Pushed, true);
            comp.SelectCOMportButton_2.Position = [464 92 103 22];
            comp.SelectCOMportButton_2.Text = 'Select COM port';

            % Create COMEditFieldLabel_2
            comp.COMEditFieldLabel_2 = uilabel(comp);
            comp.COMEditFieldLabel_2.HorizontalAlignment = 'right';
            comp.COMEditFieldLabel_2.Position = [467 125 33 22];
            comp.COMEditFieldLabel_2.Text = 'COM';

            % Create COMEditField_2
            comp.COMEditField_2 = uieditfield(comp, 'numeric');
            comp.COMEditField_2.ValueChangedFcn =
matlab.apps.createCallbackFcn(comp, @COMEditField_2ValueChanged, true);
            comp.COMEditField_2.HorizontalAlignment = 'center';
            comp.COMEditField_2.Position = [515 125 53 22];

            % Create Controller1Label
            comp.Controller1Label = uilabel(comp);
            comp.Controller1Label.FontSize = 14;
            comp.Controller1Label.FontWeight = 'bold';
            comp.Controller1Label.Position = [475 362 84 22];
            comp.Controller1Label.Text = 'Controller 1';

            % Create Controller2Label
            comp.Controller2Label = uilabel(comp);
            comp.Controller2Label.FontSize = 14;
            comp.Controller2Label.FontWeight = 'bold';
            comp.Controller2Label.Position = [475 156 84 22];
            comp.Controller2Label.Text = 'Controller 2';
```

```
            % Execute the startup function
            postSetupFcn(comp)
        end
    end
end
```

Matlab code for setting up camera

```
% Step 1: Create a webcam object
cam = webcam(1);  % Uses the default camera on your laptop
cam.Resolution = '1920x1080'; % Set camera resolution

% Step 2: Set up a figure to display the webcam feed
% figure;

% Step 3: Define the initial reference point (adjust when setting up the camera)
initial_point = [278, 47]; % Initial point on the image for 1920x1080 resolution

% Step 4: Define real-world dimensions of the camera's field of view
cam_height = 274;                                % Distance between camera and
checkerboard (mm)
zoom_in = 1.0;                                   % Camera zoom ratio
width_in_mm = cam_height*1.3*(1/zoom_in);   % Width of the real-world area covered
by the camera
height_in_mm = cam_height*0.75*(1/zoom_in); % Height of the real-world area
covered by the camera

% Calculate scaling factors for converting pixels to meters
scale_x = width_in_mm / 1920; % Meters per pixel (horizontal)
scale_y = height_in_mm / 1080; % Meters per pixel (vertical)

% Step 5: Capture a frame from the webcam
img = snapshot(cam);  % Capture one frame

detected = int16(0);
color_detected = int16(0);
```

Matlab code for detecting position and color

```
close all;
detected = 0;
color_detected = 0;
% Step 5: Capture a frame from the webcam
img = snapshot(cam);  % Capture one frame

% Step 6: Convert the image to the HSV color space
hsv_img = rgb2hsv(img);


% Step 7: Define HSV ranges for blue, red, and yellow
% Blue
blue_hue_min = 0.55;
blue_hue_max = 0.75;
blue_sat_min = 0.5;
blue_val_min = 0.3;
```

```
% Red (split across hue space: lower and upper range)
red_hue_min1 = 0.0;
red_hue_max1 = 11/360; % First range for red
red_hue_min2 = 351/360;
red_hue_max2 = 1.0; % Second range for red
red_sat_min = 0.7;
red_val_min = 0.1;

% Yellow
yellow_hue_min = 45/360;
yellow_hue_max = 64/360;
yellow_sat_min = 0.15;
yellow_val_min = 0.75;

% Step 8: Create binary masks for each color
blue_mask = (hsv_img(:,:,1) >= blue_hue_min & hsv_img(:,:,1) <= blue_hue_max) &
...
    (hsv_img(:,:,2) >= blue_sat_min) & (hsv_img(:,:,3) >= blue_val_min);

red_mask = ((hsv_img(:,:,1) >= red_hue_min1 & hsv_img(:,:,1) <= red_hue_max1) |
...
    (hsv_img(:,:,1) >= red_hue_min2 & hsv_img(:,:,1) <= red_hue_max2)) & ...
    (hsv_img(:,:,2) >= red_sat_min) & (hsv_img(:,:,3) >= red_val_min);

yellow_mask = (hsv_img(:,:,1) >= yellow_hue_min & hsv_img(:,:,1) <=
yellow_hue_max) & ...
    (hsv_img(:,:,2) >= yellow_sat_min) & (hsv_img(:,:,3) >= yellow_val_min);

% Combine the masks
combined_mask = blue_mask | red_mask | yellow_mask;

% Step 9: Clean the binary mask using morphological operations
combined_mask = imopen(combined_mask, strel('disk', 7)); % Remove small objects
combined_mask = imclose(combined_mask, strel('disk', 5)); % Close small gaps

% Show each color mask to debug
figure(1); % Create a separate figure for debugging
subplot(2, 2, 1); imshow(img); title('Original Image');
subplot(2, 2, 2); imshow(blue_mask); title('Blue Mask');
subplot(2, 2, 3); imshow(red_mask); title('Red Mask');
subplot(2, 2, 4); imshow(yellow_mask); title('Yellow Mask');

% Step 10: Filter objects by size
min_area = 1000; % Minimum area of the object (adjust as needed)
max_area = 500000; % Maximum area of the object (adjust as needed)
labeled_mask = bwlabel(combined_mask);
stats = regionprops(labeled_mask, 'BoundingBox', 'Centroid', 'Area');

% Filter out regions that don't meet the size criteria
valid_stats = stats([stats.Area] >= min_area & [stats.Area] <= max_area);

% Step 11: Display the original image
figure
imshow(img);
hold on;

% Step 12: Draw the initial point
plot(initial_point(1), initial_point(2), 'r.', 'MarkerSize', 20); % Red dot at the
initial point
```

```
% Step 13: Draw bounding boxes and calculate relative positions
objects_position_mm = [];
centroids = [];
for k = 1:length(valid_stats)
    % Get the bounding box and centroid coordinates
    bbox = valid_stats(k).BoundingBox;
    centroid = valid_stats(k).Centroid;

    % Calculate relative position in pixels
    relative_position_pixels = [centroid(1) - initial_point(1), centroid(2) -
initial_point(2)];

    % Convert relative position to real-world units
    relative_position_mm = [-relative_position_pixels(1) * scale_x + 479, ...
        relative_position_pixels(2) * scale_y];

    % Extract the HSV values at the centroid
    centroid_x = round(centroid(1));
    centroid_y = round(centroid(2));
    hsv_value_at_centroid = hsv_img(centroid_y, centroid_x, :); % Extract HSV
value at centroid
    hue = hsv_value_at_centroid(1); % Hue
    saturation = hsv_value_at_centroid(2); % Saturation
    value = hsv_value_at_centroid(3); % Brightness/Value

    % Display the HSV values near the centroid
    text(centroid_x + 100, centroid_y - 25, ...
        sprintf('HSV: (%.2f, %.2f, %.2f)', hue, saturation, value), ...
        'Color', 'red', 'FontSize', 12, 'FontWeight', 'bold');

    % Determine the color of the object (blue, red, yellow)
    if blue_mask(centroid_y, centroid_x)
        color = 'Blue';
        color_detected = 1;

    elseif red_mask(centroid_y, centroid_x)
        color = 'Red';
        color_detected = 2;
    elseif yellow_mask(centroid_y, centroid_x)
        color = 'Yellow';
        color_detected = 3;
    else
        color = 'Unknown';
        color_detected = 4;
    end

    % Display the color of the object
    % text(centroid_x, centroid_y - 10, ...
    %     sprintf('Color: %s', color), ...
    %     'Color', 'yellow', 'FontSize', 12, 'FontWeight', 'bold');

    % Display the centroid position and relative position in real-world units
    % text(centroid_x, centroid_y, ...
    %     sprintf('Rel Pos: (%.2f px, %.2f px)\n(%.2f mm, %.2f mm)', ...
    %     relative_position_pixels(1), relative_position_pixels(2), ...
    %     relative_position_mm(1), relative_position_mm(2)), ...
    %     'Color', 'yellow', 'FontSize', 12, 'FontWeight', 'bold');
```

```
    % Display the centroid position in real-world units
    text(centroid_x + 100, centroid_y + 50, ...
        sprintf('Color: %s\nReal Pos: (%.2f mm, %.2f mm)\nPixel Pos: (%.2f px,
%.2f px)', ...
        color,...
        relative_position_mm(1), relative_position_mm(2), ...
        relative_position_pixels(1), relative_position_pixels(2)),...
        'Color', 'yellow', 'FontSize', 12, 'FontWeight', 'bold');
    % Draw a rectangle around the object
    rectangle('Position', bbox, 'EdgeColor', 'r', 'LineWidth', 2);

    if centroid ~= 0
        detected = detected + 1;
        objects_position_mm(detected,1:4) = [detected, relative_position_mm,
color_detected]; % Collect objects_position_mm in a row-wise fashion
        centroids(detected,1:4) = [detected, centroid_x, centroid_y,
color_detected];
    end
end
position = round(relative_position_mm);
% Step 14: Pause for a moment to display the frame before capturing the next one
% pause(0.1);
hold off;
```