



THE UNIVERSITY OF  
**JORDAN**



**School of Engineering**



**Department of Mechatronics Engineering**

**Bachelor of Science in Mechatronics Engineering**  
**Senior Design Graduation Project Report**

**AI – Enhanced Robotic System Design for Electronic Chips Sorting**

Report by

Rashed Al Ashqar

Mufid Abu-Sara

Dania Al Salhi

Moath Khanfar

Supervisor

Dr. Mohammad Al-Mashagbeh

Date

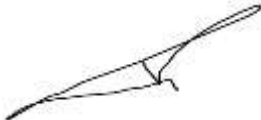

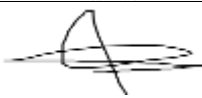

11/06/2024

# DECLARATION STATEMENT

We, the undersigned students, confirm that the work submitted in this project report is entirely our own and has not been copied from any other source. Any material that has been used from other sources has been properly cited and acknowledged in the report.

We are fully aware that any copying or improper citation of references/sources used in this report will be considered plagiarism, which is a clear violation of the Code of Ethics of University of Jordan.

In addition, we have read and understood the legal consequences of committing any violation of the University's Code of Ethics.

	Student Name	Student ID	Signature	Date
1	Dania Al Salhi	0194393		22 / May / 2024
2	Mufid Abu-Sara	0199345		22 / May / 2024
3	Moath Khunfur	0190058		22 / May / 2024
4	Rashed Al Ashqar	0195746		22 / May / 2024

# ABSTRACT

In Jordan specifically and the world in general, dealing with PCB electrical waste poses significant challenges due to the presence of toxic chemicals like lead and mercury, which threaten both the environment and human health. Informal recycling practices worsen the situation, leading to pollution and health risks from mishandling PCBs. The country's limited infrastructure for managing electronic waste exacerbates these issues, amplifying health risks for workers and local communities. Moreover, valuable materials in PCBs often go to waste as they're frequently discarded instead of being recycled. To address this, our project proposes an integrated system merging a robotic arm with computer vision technology to automate the sorting of electronic chips. The robotic arm handles chips based on their characteristics, guided by real-time imaging and classification from the vision system. Deep learning models help classify chips, while an efficient sorting algorithm place them into predefined bins. This approach promises precise sorting, increased efficiency, and cost-effectiveness in semiconductor manufacturing processes.

After making and deploying the AI-enhanced pick-and-place system for PCBs as outlined, we attained exceptional accuracy and performance. The system effectively distinguishes PCBs from other objects with a high percentage of success. Furthermore, it accurately identifies the position of detected PCBs and reliably picks and places them in their designated locations.

# ACKNOWLEDGEMENTS

First and foremost, we'd like to express our gratitude to Allah for guiding us to this point in our lives and enabling us to successfully complete our graduation project.

As a group, we would like to extend our sincere gratitude to all the individuals who have been instrumental in the successful completion of this project.

Our heartfelt appreciation goes to our project advisor, Dr. Mohammad Al-Mashagbeh, for his invaluable guidance, support, and profound insights that have been the cornerstone of our research journey. Their mentorship has been truly enlightening. Also we would like to thank Dr. Musa Al-yaman and Dr. Adham Al-sharkawi for their help too

We would like to acknowledge University of Jordan for providing us with the necessary resources and facilities that were vital for the execution of this project. Your commitment to fostering an environment of research and innovation has been highly commendable.

We also extend our appreciation to our friends and families for their unwavering encouragement and understanding throughout this demanding project. Your support has been a constant source of motivation.

Lastly, we would like to express our thanks to all the participants and contributors who have actively engaged in this project. Your involvement and dedication have been pivotal in achieving our goals.

This project has been a collective journey of growth and discovery, and it would not have been possible without the support and contributions of everyone mentioned above. Thank you all for being an integral part of this remarkable experience.

## TABLE OF CONTENTS

DECLARATION STATEMENT .....	i
ABSTRACT .....	ii
ACKNOWLEDGEMENTS .....	iii
LIST OF FIGURES.....	1
Chapter 1 Introduction.....	4
1.1 Background.....	4
1.2 Problem Definition .....	5
1.3 Literature Review .....	5
1.4 Aims and Objectives .....	7
1.5 Project Plan.....	8
1.6 Report Organization.....	9
Chapter 2 Design Options.....	10
2.1 Robotic Arm .....	10
2.1.1 Quanser Arm .....	10
2.1.2 Overview of Quanser Arm Components: -.....	11
2.2 Software tools .....	14
2.3 Programming Languages Options .....	15
2.4 YOLO Versions.....	17
2.4.1 YOLO overview.....	17
2.4.2 YOLO versions selection .....	17
2.4.3 YOLOv5 .....	18
2.4.4 YOLOv5 types comparison.....	19
2.5 Design Constraints and Standards.....	19
2.5.1 Design Constraints .....	19
2.5.2 Design Standards .....	20
Chapter 3 System Overview .....	21
3.1 Pick and place using computer vision .....	21

3.2 Data Management and PCB Detection .....	21
3.2.1 Data selection .....	22
3.2.2 Class labelling and annotation .....	22
3.2.3 Data split and adjustments .....	24
3.2.4 Dataset augmentation.....	24
3.2.5 Generating dataset version.....	26
3.2.6 Dataset training.....	26
3.2.7 Real-time object detection .....	27
3.3 Motion Control .....	28
3.3.1 QUARC Software.....	28
3.3.2 Sequence of pick and place mechanism .....	28
3.4 Final System .....	40
Chapter 4 Design Testing And Results .....	42
4.1 Data management and PCB detection results .....	42
4.2 Pick and place system results .....	48
4.3 System Limitations and Compliance with Design Constraints .....	53
4.4 Solution Impact .....	54
Chapter 5 Conclusion & Future Work .....	56
5.1 Conclusion .....	56
5.2 Recommendations for Future Work.....	56
REFERENCES .....	57
Appendices .....	61

# LIST OF FIGURES

Figure 2-3: Quanser Arm .....	10
Figure 2-4: QArm components.....	11
Figure 2-5: Intel RealSense D415 camera.....	12
Figure 2-6: End-Effector IO Board .....	13
Figure 2-7: gripper's multiple points of contact .....	13
Figure 3-1:Data management and PCB detection sequence .....	21
Figure 3-2: Printed Circuit Boards Labelling Process .....	23
Figure 3-3:False Positive dataset .....	23
Figure 3-4: Division of a dataset into three sets: Train set, Valid set, and Test set.....	24
Figure 3-5: Flip augmentation .....	25
Figure 3-6: Hue augmentation .....	25
Figure 3-7: Mosaic augmentation .....	26
Figure 3-8:camera usage at real time .....	27
Figure 3-9: Frame diagram for the Quanser Arm manipulator.....	30
Figure 3-10: Workspace of the QArm .....	31
Figure 3-11:Forward kinematics snapshot.....	32
Figure 3-12:Inverse kinematics snapshot .....	35
Figure 3-13:Trajectory Spline of initial and final positions .....	36
Figure 3-14:Cubic Spline snapshot.....	36
Figure 3-15: Waypoint Navigator snapshot.....	37
Figure 3-16:Waypoint navigator state machine .....	38
Figure 3-17:Gripper actuator snapshot .....	38
Figure 3-18:Gripper current protection circuit .....	39
Figure 3-19:Real-time Pick-Place Flowchart.....	40
Figure 3-20: Full system blocks in Simulink.....	41
Figure 4-1: Training F1 curve.....	42
Figure 4-2: Training Precision-Recall curve.....	43

Figure 4-3: Training Confusion Matrix.....	43
Figure 4-4: Training metrics graphs .....	44
Figure 4-5: Metrics for precision and recall and mean average precision .....	45
Figure 4-6: Training Process and the accuracy and performance of the YOLO model.....	45
Figure 4-7: Model performance on test images .....	46
Figure 4-8: Real-time Visualization on real PCBs with confidence results .....	47
Figure 4-9: Robotic arm motion in Pick position case 1 .....	48
Figure 4-10: Robotic arm motion in Pick position case 2.....	49
Figure 4-11: Robotic arm motion in Pick position case 3.....	50
Figure 4-12: Forward Kinematics plot when the QArm is in home position (Fixed).....	51
Figure 4-13: Forward Kinematics plot when the QArm is in moving condition .....	51
Figure 4-14: Current signal before tuning the gain ( $K_p=12$ ) .....	52
Figure 4-15: Current signal after tuning the gain ( $K_p=6$ ) .....	52

## LIST OF TABLES

Table 1-1 Project Plan.....	8
Table 2-3: Measurements of Quanser Arm.....	10
Table 2-4: QArm Components .....	11
Table 2-5: Intel RealSense D415 camera specifications .....	12
Table 2-6: Intel RealSense D415 field of view .....	12
Table 2-7: YOLOv5 types comparison .....	19
Table 3-1: DH- Table.....	30
Table 3-2: Linear mapping for the mathematical operations .....	32
Table 3-3: what symbols resemble .....	34
Table 3-4: Mapping variables to calculate $\theta_2$ .....	34



# GLOSSARY

ABBREVIATION	DESCRIPTION
AI	Artificial Intelligence
RGB	Red, Green Blue
DAQ	Data Acquisition
ROS	Robot Operating System
CPU	Central Process Unit
NN	Neural Network
CNN	Convolutional Neural Network
YOLO	You Only Look Once
ISO	International organization for standardization
PCB	Printed Circuit Board
ADC	Analog to Digital Converter
IR	Infrared
3D	3-Dimensional
E-waste	Electrical waste
DOF	Degrees Of Freedom
QArm	Quanser Arm
IO	Input Output
GIL	Global Interpreter Lock
GPU	Graphics processing unit
EXIF	Exchangeable Image File Format
ML	Machine Learning
PC	Personal Computer
VS Code	Visual Studio Code
DH	Deviant-Hartenberg
PI	Proportional Integral
IoU	Intersection over Union
ISMS	Information Security Management Systems

# Chapter 1 INTRODUCTION

## 1.1 Background

Electronic waste, especially things like tiny electronic chips, has become a big problem as we use more and more electronic gadgets. In the past, we didn't have good rules for getting rid of electronic waste, so it was often handled in ways that weren't safe for the environment [1]. With new and cool electronic devices coming out all the time, old ones get thrown away a lot. This creates a lot of waste, and dealing with it is tricky because electronic chips are tiny but have important stuff in them.

As people started realizing that tossing electronics is bad for the environment, there are now more rules and ways to handle electronic waste better. Sorting is considered the most effective way of classification and dispose of waste. [1]

Sorting stands as a foundational computational process essential for efficient database operations. By systematically organizing data into a specific order, sorting algorithms enable streamlined and expedited searching within a database [2]. Before the 20th century, people manually sorted items such as library shelves and card sorting, both methods were not appropriate for large-scale data processing [3].

In the early days of computing (1900s - 1960s), sorting algorithms were simple but inefficient. There were two important techniques bubble sort and selection sort [3], The foundation for more effective algorithms was built by these methods.

The breakthrough came in the 1960s with the invention of Quicksort by Tony Hoare. Quicksort introduced the concept of “divide and conquer,” significantly improving sorting efficiency. In 1980s, Włodzimierz Dobosiewicz introduced Comb Sort, a relatively simple sorting algorithm based on Bubble Sort. Comb Sort aims to improve upon Bubble Sort's inefficiency by eliminating small turtles (inversions) in the data [3].

In the contemporary landscape of sorting algorithms, there is a notable shift towards enhanced flexibility. Modern libraries employ hybrid and adaptive sorting methods that dynamically adjust their strategies based on the inherent properties of the data they encounter. Two prominent examples of such algorithms are Timsort and Bitonic sort [3]. Their advantages are that they adapt to real-world data patterns, making them efficient for both small and large datasets and leveraging the power of multi-core processors and distributed systems.

## **1.2 Problem Definition**

With the increased amount of electronics in the modern era, industrial waste is starting to become a real challenge for humanity to maintain the health of our planet [1]. This surge, driven by economic growth, urbanization, and increased consumer demand, poses environmental and economic challenges.

A robotic arm sorting system is an automated solution designed to classify objects and arrange them according to specific criteria-pick and place operation. This technology plays a crucial role in various industries, such as sorting objects [4].

Since the ultimate purpose of Machine/Robot is reducing the error and increasing the efficiency of Humans Robotic arms were chosen for sorting processes over other machines due to their unparalleled versatility, precision, adaptability, and efficiency. Their flexibility allows for easy programming and reprogramming to handle a diverse range of sorting tasks based on various criteria, making them adaptable to different industries and applications [5].

## **1.3 Literature Review**

Sorting algorithms are a fundamental procedure in computer science because they enable efficient data processing and analysis. The literature review covers the meaning of sorting algorithms and their development. The origins of sorting algorithms go back to the 1940s, when the first electronic computers were invented and required methods for organizing large amounts of information. Since then, many sorting algorithms have been developed, each with different advantages and disadvantages.

Hadi Salehi found that Artificial intelligence (AI) consists of two subfields that have grown significantly in recent years: Machine Learning, which recognises patterns, learns from data, and improves without being programmed, and Deep Learning, which is a type of Machine Learning that uses artificial neural networks to analyse data and solve complex problems. These technologies process huge amounts of data very quickly and use it to make decisions without human intervention. Machine Learning and Deep Learning can further improve the sorting process for operations that are already using sensor-based sorting, but also create new opportunities by processing very low-grade items that would have otherwise been thrown away [6]. A further advantage of AI is the vast amount of data it generates and processes. This data is useful for predictive maintenance as well as for getting useful information about the sorter's operations.

Maria Koskinopoulou and Fredy Raptopoulos have conducted research into the deployment of an autonomous robotic system for sorting recyclables based on their material types. Their focus lies in developing a cost-effective computer vision module utilizing deep learning technologies for object recognition and classification. This module, after undergoing extensive training, is integrated with the robotic system responsible for physically sorting recyclables. In conclusion they found out that

integration of these technologies in waste processing facilities has the potential to significantly improve the efficiency of recyclables processing [7].

Vishnu R. Kale (R. Kale n.d.) introduced an intelligent strategy for real-time inspection and sorting of objects in a continuous flow. This involves employing a robotic arm integrated with a machine learning algorithm. The sorting process is structured into two phases: firstly, a self-learning phase where the system learns to recognize objects, and secondly, an operational selection phase where objects are identified, classified using a decision-making algorithm, and sorted in real-time. However, certain challenges such as varying lighting conditions or component size may be encountered by the robotic arm, necessitating consideration of these factors. Consequently, knowledge of the materials is crucial for programming the arm accordingly. By ensuring accurate separation, favourable results can be achieved in the sorting process [8].

Dharmannagari Vinay (Kumar Reddy n.d.) stresses the importance of precise coordination in controlling the robotic arm's movements to effectively capture objects in motion along a conveyor belt, with the aim of achieving a fully automated material handling system. This can be accomplished by integrating a pair of IR sensors with the AT89S52 Micro Controller Unit to synchronize the robotic arm's movement for retrieving objects from the conveyor belt. The objective is to accurately classify colored items as they move along the conveyor by efficiently picking them up and placing them in their designated locations. This automated configuration also incorporates colour sensors responsible for identifying the object colours and transmitting pertinent signals to the microcontroller [9].

Cong Duy Vo, Duy Anh Dang, and Phuong (Duy Anh Dang n.d.) Hoai Le have devised a multi-robotic arm system equipped with stereo vision technology to proficiently sort objects according to their size and shape characteristics. This system comprises a master robot and three slave robots, each dedicated to a conveyor belt. The stereo vision system is engineered to compute 3D object coordinates by identifying centroids in two images, thereby sidestepping computationally demanding methods such as reconstructing entire disparity maps. Experimental results validate the system's efficacy, underscoring its potential to enhance production line efficiency and diminish processing time across diverse industrial settings [10].

Dr Andrius Dzedickis said that robots, with their precision and efficiency, have improved the speed and accuracy of sorting tasks in industries such as waste management. This has resulted in cost savings, increased productivity, and a safer work environment. The impact of these advancements is profound, paving the way for a more sustainable and efficient industrial future. the implementation of computer vision to localise and manipulate randomly placed mechanical parts on a conveyor fostered the robotisation of sorting processes in all industry fields [11].

Sun, Z.; Huang, employed a robotic system integrating computer vision and artificial intelligence (AI) to sort coal and gangue. Given that raw coal often contains a significant amount of gangue, which can degrade coal quality and harm the environment, separating the two materials is crucial. The utilization of computer vision seems to enhance sorting quality, particularly when dealing with extensive datasets [12].

Harsha A K and A Sampreeth developed an entirely automated sorting system capable of accurately categorizing objects by their shape and color. The system incorporates a robotic arm controlled via an ARM 7 microcontroller, with image input facilitated by a single webcam. Utilizing MATLAB for image processing, the system identifies both the color and shape of objects. Their research demonstrates the effective utilization of external sensors and processing to enable robotic arm manipulation for color-based object sorting. However, a notable limitation of their system is its reliance solely on color differentiation for sorting purposes [13].

## ***1.4 Aims and Objectives***

In this project our main concern is to reduce the negative effects of E-waste by sorting them out of other materials and making it easy to recycle them and for that we intend to deploy an innovative integrated system, combining a robotic arm with cutting-edge computer vision technology, to streamline the sorting process of electronic chips. By leveraging this system, the aim is to automate and enhance efficiency in handling electronic waste, particularly PCBs. The integration of a robotic arm ensures precise and swift manipulation of the chips, while computer vision technology facilitates real-time identification and classification of various chip types. Moreover, the main aim goes beyond just sorting well; it encompasses the application of this system in future recycling processes to mitigate the adverse environmental and health impacts stemming from the mishandling of PCBs. By automating the sorting process and subsequently optimizing recycling efforts, this initiative seeks to significantly reduce pollution levels and health risks associated with electronic waste management.

The main objectives of this project are: -

- Implement computer vision and machine learning algorithms for chip recognition and classification.
- Optimize the system's efficiency in sorting accuracy, speed, and resource utilization.
- Utilize MATLAB, Python, and other software to detect electronic chips and employ mathematical algorithms for sorting optimization.

Upon finishing this project, where we developed and implemented the AI-enhanced pick-and-place system for PCBs as described, we achieved remarkable accuracy and performance. The system adeptly recognizes PCBs from other objects with a high success rate. Additionally, it precisely detects the position of identified PCBs and consistently executes the task of picking and placing them in their designated locations.

## 1.5 Project Plan

Table 1-1 Project Plan

Graduation Project (1) Tasks Description																					
Task No.	Task Description	Semester Weeks																Students Involved			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	1	2	3	4
1	Topic selection																				
2	Papers Research																				
3	Literature Review																				
4	Data Collection																				
5	Obtaining Algorithm																				
6	Design and Simulation																				
7	Documentation submission																				
Graduation Project (2) Tasks Description																					
Task No.	Task Description	Semester Weeks																Students Involved			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	1	2	3	4
1	Build the model																				
2	Implement and Execute Algorithms																				
3	Test the model																				
4	Troubleshooting errors																				
5	Final documentation																				

## ***1.6 Report Organization***

Following the introduction, we have divided our project into four major chapters:

- Chapter 2 Presents selection and sizing options of hardware components and software selection, Design constraints.
- Chapter 3 Details the system design
- Chapter 4 Tests the Results
- Chapter 5 talks about conclusion and future work

# Chapter 2 DESIGN OPTIONS

## 2.1 Robotic Arm

The groundwork of our project is the robotic arm, so we chose the following robot which we will show you the specifications of it as following in this section:

### 2.1.1 Quanser Arm

The robotic arm that utilized in our project is the Quanser Arm, as it shown in Figure 2-3. This particular model is distinguished by its notable precision and accuracy. Additionally, the Quanser arm typically includes integrated sensors and software packages, streamlining operational procedures. These features make it easier for users to set up and use, which makes it great for trying new things and learning. The Quanser arm is specifically designed for teaching, so it's easy to use and works well for lots of different things.



Figure 2-1: Quanser Arm [14]

In the table below 2-3, we will show you some specifications to the QArm that we have it in the project:

Table 2-1: Measurements of Quanser Arm [14]

Base	Shoulder	Elbow	Wrist	Maximum joints speed	Manipulator weight	Payload	Reach
$\pm 170^\circ$	$\pm 85^\circ$	$-95^\circ/+75^\circ$	$\pm 160^\circ$	$\pm 90^\circ/\text{s.}$	8.25 kg	350-750 g	750 mm

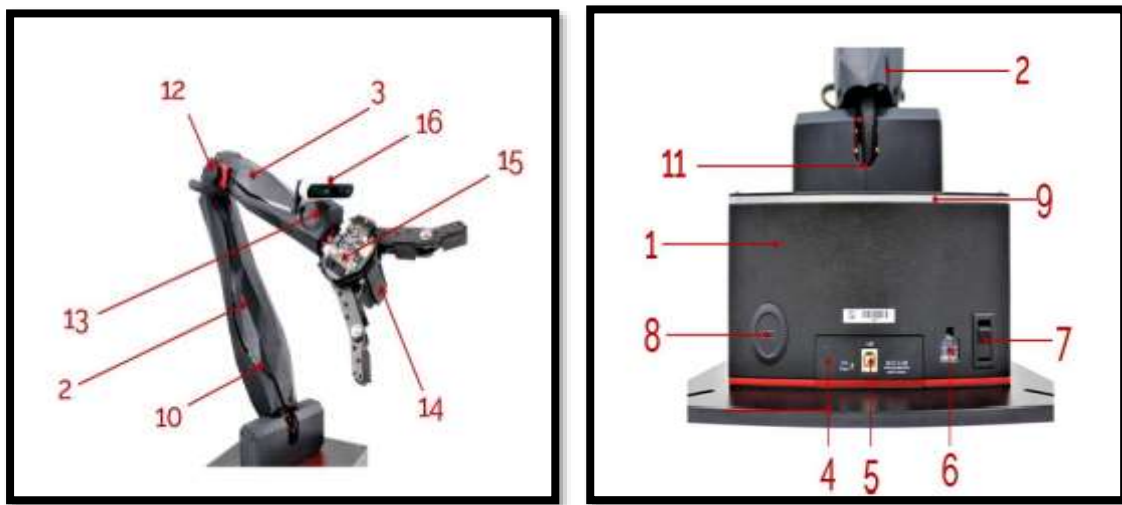


### 2.1.2 Overview of Quanser Arm Components: -

- **Manipulator**

The QARM by Quanser is a 4-degree-of-freedom (DOF) robotic arm which consists of four joints arranged in a revolute-prismatic-prismatic-revolute configuration [14], allowing for a large reachable workspace.

All the joints use Dynamixel servo motors that are complete with 12-bit programmable magnetic encoder (4096 counts/rev), velocity measurement, current sensors, and temperature sensors. The base, shoulder, and elbow use the XM540-W270-R Dynamixel servomotor [15].



**Figure 2-2: QArm components [14]**

The provided table 2-4, represent the components of the QArm: -

**Table 2-2: QArm Components [14]**

<b>1</b>	Base	<b>9</b>	Base LED
<b>2</b>	Upper Arm	<b>10</b>	Arm LEDs
<b>3</b>	Lower Arm	<b>11</b>	Shoulder joint
<b>4</b>	QFLEX USB Pane	<b>12</b>	Elbow joint
<b>5</b>	QArm USB-B connector	<b>13</b>	Wrist joint
<b>6</b>	Power Connector	<b>14</b>	Gripper
<b>7</b>	Power Switch	<b>15</b>	End-effector Data Acquisition (DAQ)
<b>8</b>	Camera USB-C Connector	<b>16</b>	Camera

- **Intel® RealSense™ D415 RGBD camera**

The QArm platform comes equipped with an Intel RealSense D415 RGB-D camera which is shown in figure 2-5. It includes an IR projector and two IR receivers, making this unit a stereo tracking solution. The camera can provide RGB, Infrared (left and right) and depth streams of data.

The D415 camera is a perfect choice because of the accuracy it provides and the high-quality depth per degree, it uses rolling shutter sensors which give precise measurements for small objects [16].



**Figure 2-3: Intel RealSense D415 camera [16]**

The following tables 2-5 and 2-6 are describe the main specifications to Intel RealSense Camera D415: -

**Table 2-3: Intel RealSense D415 camera specifications [16]**

Depth distance measurable	Ideal Range	Depth output resolution	Depth frame rate	RGB sensor Technology	RGB sensor resolution
$\geq 0.16\text{m}$	5 m to 3 m	$\leq 1280 \times 720$	$\leq 90 \text{ fps}$	Rolling shutter	2 MP

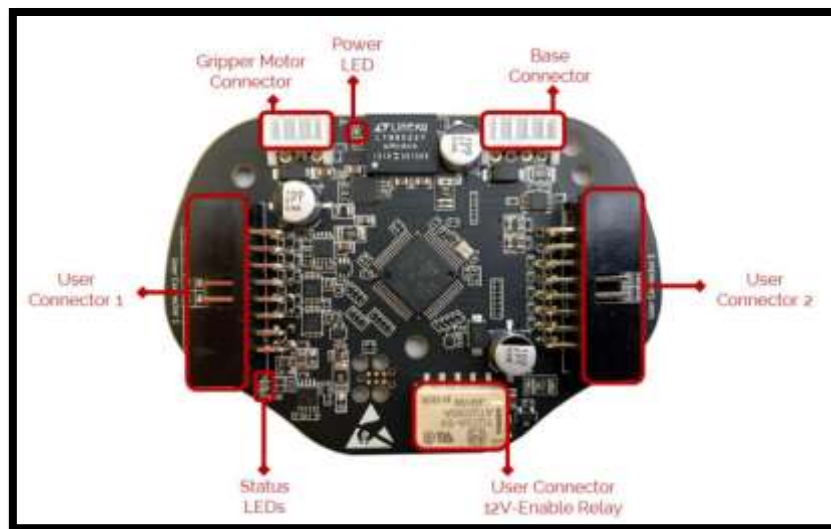
**Table 2-4: Intel RealSense D415 field of view [16]**

Camera	Horizontal	Vertical	Diagonal
RGB	$69.4^\circ \pm 3^\circ$	$42.5^\circ \pm 3^\circ$	$77^\circ \pm 3^\circ$
Depth	$65^\circ \pm 2^\circ$	$40^\circ \pm 1^\circ$	$72^\circ \pm 2^\circ$

- **End effector DAQ (data acquisition device):**

Located next to the gripper, The QArm's end-effector comes equipped with a Data Acquisition (DAQ) PCB with two user connectors, two ADCs and a relay for enabling 12V. Note that while the end-effector board provides connections for external user devices by having 16-pin connector, users are responsible for certifying any modifications or additions they make to the default configuration [17].

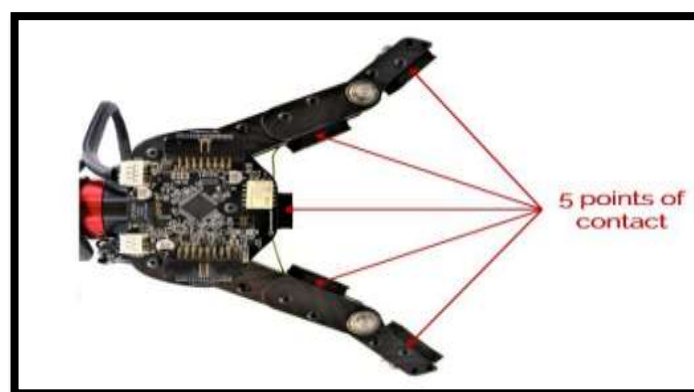
The end-effector DAQ that is shown in figure 2-6, has five analogue inputs analogue signal range from 0 V to 5 V with 12-bit accuracy.



**Figure 2-4: End-Effector IO Board [17]**

- **Gripper**

The QArm uses a tendon-based 2-finger gripper as it shown in figure 2-7. This under actuated system uses a single XC430-W240-T Dynamixel to actuate two articulated fingers [15], each with two links. Foam pads on the fingers improves grip with at least 2 and up to 5 points of contact as it grips around the shape of an object [14].



**Figure 2-5: gripper's multiple points of contact [14]**

## **2.2 Software tools**

There are many options that we must have to control the movement of robotic arm, so we came down to 2 options which they are: MATLAB / Simulink, and ROS (Robot Operation System).

MATLAB and Simulink are robust software platforms widely utilized in various domains for numerical computations, data analysis, and dynamic system simulations.

MATLAB functions as an advanced programming language, streamlining algorithm development, while Simulink presents a graphical interface for the creation, simulation, and analysis of complex dynamical systems across multiple domains [18].

ROS is an open-source middleware framework extensively utilized in the realm of robotics to create and manage robotic systems. Praised for its adaptability and modular design, ROS offers a robust toolkit and libraries that simplify functions like hardware abstraction, communication between diverse components [19].

### **Advantages of Simulink / MATLAB:**

1. **Graphical Modelling:** Simulink provides a visual environment for modelling and simulating dynamic systems, including robotic arms. You can create block diagrams to represent the system components and their interactions.
2. **Integration with MATLAB:** Simulink seamlessly integrates with MATLAB, allowing you to use MATLAB functions, scripts, and toolboxes within your Simulink models.
3. **Packages:** Simulink offers hardware support packages for interfacing with various sensors, actuators, and microcontrollers.
4. **Real-Time Simulation:** Simulink supports real-time simulation, which is useful for testing control algorithms before deploying them on actual hardware [20].

### **Disadvantages of Simulink / MATLAB:**

1. **Complexity:** Simulink models can become complex, especially for large-scale robotic systems.
2. **Resource Intensive:** Running simulations in Simulink may require significant computational resources.
3. **Proprietary:** MATLAB and Simulink are commercial software, which may not be ideal for open-source or collaborative projects [20].

### **Advantages of ROS:**

1. **Modularity:** ROS promotes a modular approach to robotics development. You can create separate nodes for different functionalities (e.g., perception, control, planning) and communicate between them using ROS messages.
2. **Community and Packages:** ROS has a large community and an extensive package ecosystem. You can find pre-built packages for various robotic tasks (e.g., kinematics, perception, navigation) [18].
3. **Platform Independence:** ROS is platform-independent and supports various programming languages (Python, C++, etc.).

### **Disadvantages of ROS:**

1. **Learning Curve:** ROS has a learning curve, especially for beginners. Understanding topics, nodes, messages, and launch files can take time [21].
2. **Resource Usage:** Running ROS nodes can consume memory and CPU resources.
3. **Real-Time Constraints:** ROS is not inherently real-time, although real-time extensions like ROS 2 can address this limitation.

So, we chose the MATLAB / Simulink because it's more powerful for modelling and simulation, especially because it has a real-time control software developed by Quanser called QUARC, which is built in Simulink that provides an efficient way to design, develop, deploy, and validate real-time applications.

## ***2.3 Programming Languages Options***

As you see before that there are two options to control the robotic arm, there are also many languages to implement AI algorithms in the robotic arms and we came down to 2 options which they are: Python, and C++.

Python stands out as a high-level, interpreted programming language renowned for its readability and simplicity. Engineered with a focus on code clarity [22], Python's interpreted nature and dynamic typing make it a preferred choice for rapid prototyping and development.

C++, on the other hand, is a lower-level, compiled programming language prized for high performance and fine-grained memory manipulation. Evolving as an extension of the C language, C++ supports both procedural and object-oriented programming paradigms [23].

**Advantages of Python: -**

1. Ease of Use: Python is known for its simplicity and readability. It's an excellent choice for rapid prototyping and experimentation.
2. Large Community and Libraries: Python has a vast community and numerous libraries (such as NumPy, SciPy, and Tensor Flow) that facilitate AI development.
3. Interoperability: Python can easily interface with other languages and tools [24].
4. Dynamic Typing: Python's dynamic typing allows flexibility during development.

**Disadvantages of Python: -**

1. Performance: Python is an interpreted language, which can be slower than compiled languages like C++. Real-time performance may suffer.
2. Global Interpreter Lock (GIL): The GIL restricts true parallelism in Python, affecting multi-threaded applications [24].
3. Memory Overhead: Python consumes more memory compared to C++.
4. Not Ideal for Low-Level Control: For low-level control of hardware, Python may not be the best choice.

**Advantages of C++: -**

1. Performance: C++ is highly efficient and offers low-level memory manipulation.
2. It's well-suited for real-time applications.
3. Control Over Hardware: C++ provides precise control over system resources and memory allocation [25].
4. Industry Standard: Many robotics companies use C++ extensively for performance-critical components.
5. Steeper Learning Curve: C++ requires more effort to learn but offers better performance.

**Disadvantages of C++: -**

1. Complexity: C++ syntax can be more challenging, especially for beginners.
2. Development Time: Writing C++ code often takes longer than Python due to its lower-level nature.
3. Debugging: Debugging C++ programs can be more involved.

So, we chose the python for ease of development and experimentation, also because it is commonly used for AI algorithms, sensor data processing, and simulation.

## **2.4 YOLO Versions**

### **2.4.1 YOLO overview**

The deep learning algorithm chosen for chip detection is the Neural Network (NN), with CNNs (Convolutional Neural Networks) selected specifically for their capacity in processing and analysing visual data like images and videos. Within the realm of CNNs, the YOLO (You Only Look Once) algorithm stands out as it offers enhanced speed and accuracy compared to many other CNN-based object detection algorithms. However, the diversity within the YOLO algorithm family can sometimes lead to confusion regarding the most suitable choice [26].

Therefore, it's crucial to meticulously evaluate the strengths and weaknesses of each variant to facilitate informed decision-making. Deep learning algorithms can be broadly categorized into single-stage classifiers and two-stage classifiers. Single-stage classifiers, like YOLO, execute object localization and classification in a single step, rendering them faster and suitable for real-time applications. Conversely, two-stage classifiers entail an additional region proposal step, which can impact speed but may offer other advantages in certain contexts.

### **2.4.2 YOLO versions selection**

To elucidate further, the forthcoming discussion will expound upon the distinctive merits and drawbacks inherent in the different types of YOLO algorithms. This comprehensive analysis will ultimately elucidate the rationale behind the selection of a specific YOLO variant, thereby ensuring a nuanced understanding of the considerations influencing this choice.

#### **Advantages of YOLOv2: -**

1. **Speed:** YOLOv2 is known for its speed, as it can process images in real-time. This is because it only requires a single pass through the neural network to detect objects [27].
2. **Efficiency:** It is efficient in terms of memory usage and computational resources compared to other object detection algorithms.
3. **Unified Framework:** YOLOv2 provides a unified framework for object detection and classification. It directly predicts bounding boxes and class probabilities for multiple objects in a single inference, simplifying the overall process.

#### **Disadvantages of YOLOv2: -**

1. **Model Size:** Although YOLOv2 is more efficient compared to some other object detection algorithms, the model size can still be relatively large, especially when compared to simpler models. This can be a limitation for deployment on devices with strict memory constraints.
2. **Difficulty with Crowded Scenes:** YOLOv2 may struggle with accurately detecting objects in crowded scenes. Resolving such cases often requires post-processing techniques [28].

**Advantages of YOLOv5: -**

1. Ease of Use: YOLOv5 is easier to use and has a simpler architecture.
2. Speed: YOLOv5 is faster than its predecessors, especially in inference.
3. Good Accuracy: It provides a good balance between speed and accuracy.

**Disadvantages of YOLOv5: -**

1. Not the Most Accurate: While it's accurate, YOLOv5 may not be the best choice for applications requiring the highest accuracy.

**Advantages of YOLOv8: -**

1. Best Performance: YOLOv8 offers the best performance in terms of accuracy and speed.
2. Improvement Over v5: It surpasses YOLOv5 in both accuracy and runtime.
3. Depth Camera Integration: YOLOv8 is suitable for combining with depth cameras for advanced applications like digital twins.

**Disadvantages of YOLOv8: -**

1. Not Ideal for High-Resolution Inference.
2. Computational Resources: YOLOv8 requires substantial computational resources, including powerful GPUs, for training and inference. Simulink simulations may already be computationally intensive, and adding YOLOv8 into the mix could strain your system resources and slow down simulation times [29].
3. When the application needs a real-time performance YOLOv8 lacks the ability to do it. [30]

So, we chose the YOLOv5 because it's great for speed, ease of use, compatibility with MATLAB/Simulink since it's the main software of the system, and more suitable for our project.

### **2.4.3 YOLOv5**

YOLOv5, the fifth iteration of the You Only Look Once (YOLO) object detection architecture, is a convolutional neural network (CNN) crafted for real-time object detection. Renowned for its powerful and efficient design, YOLOv5 has attained state-of-the-art performance across various benchmarks. Widely embraced in diverse applications such as object detection and face recognition, its versatility underscores its significance in computer vision tasks.



### 2.4.4 YOLOv5 types comparison

There were many types for the YOLOv5 model. So we will show you a simple comparison between these types in the following table 2-7: -

**Table 2-5: YOLOv5 types comparison**

Model	Size	mAP50	Speed (CPU) (ms)
YOLOv5n	640	45.7	45
YOLOv5s	640	56.8	98
YOLOv5m	640	64.1	224

So we found that the YOLOv5s model is the most suitable for our application because it prioritizes real-time performance and resource efficiency.

YOLOv5s offers faster inference speeds owing to its reduced model size and fewer parameters. This renders it ideal for applications demanding swift object detection, such as video surveillance or robotics. While YOLOv5s may compromise slightly on accuracy compared to larger variants, its balanced performance and agility make it well-suited for rapid prototyping, testing, and scenarios where speed is paramount [31].

## 2.5 Design Constraints and Standards

In this project, we encounter several significant constraints, that serves as a crucial foundation, drawing the guidelines that shape the development and deployment of our AI-enhanced robotic arm. Also, we were committed to several standards that satisfies our project aims.

### 2.5.1 Design Constraints

1. **Accessibility:** QLAB is a software provided by the Quanser company which is not reachable for the public which made us use its alternative the Simulink.
2. **Cost:** The QARM is expensive tool for anyone interested in using it, and if it wasn't available in our school, we wouldn't have used it.
3. **Ergonomics:** It can't be used in any environment that doesn't fulfil its requirements.
4. **Maintainability:** We are restricted from making any repairs for the manipulator in case of any malfunction and we need to depend on the school for the maintenance.
5. **Reach/Workspace:** We got a limited reach for the manipulator which consists of 750mm full stretch that prevents us from picking up far objects.

## **2.5.2 Design Standards**

### **ISO/TS 15066:**

Collaborative Robot Safety. Provides important information about how to implement a collaborative robot system in a manner that maintains safety for the human operator. In the U.S., this ISO TS has been Nationally Adopted as TR 606 [32].

### **TC 299:**

Develops high quality standards for the safety of industrial robots and service robots to enable innovative robotic product to be brought onto the market. In addition, develops standards in fields like terminology, performance measurement and modularity [33].

### **ISO 27001:**

is an international standard for information security management systems (ISMS). It provides a systematic approach to managing sensitive company information, ensuring its confidentiality, integrity, and availability [34].

### **ISO 16001:2017:**

specifies general requirements and describes methods for evaluating and testing the performance of object detection systems (ODSs) and visibility aids (VAs) used on earth-moving machines [35].

### **IPC-2221**

a generic standard for printed circuit board (PCB) design that provides guidelines and requirements for the design of PCBs, covers the design guidelines and requirements for single-sided, double-sided, and multilayer PCBs and provides information on materials used in PCBs, including their properties and characteristics. This includes considerations for the dielectric material, conductor material, and surface finishes [36].

## Chapter 3 SYSTEM OVERVIEW

### 3.1 Pick and place using computer vision

The main idea behind our project is to sort different kinds of electronic chips in a smart way using a Pick and place mechanism using Quanser arm by taking advantage of its compatibility with MATLAB/Simulink-QUARC and implementing a depth camera supported by computer vision to detect the PCB's.

The pick and place system for sorting electronic chips using object detection through computer vision integrates several key components to streamline the manufacturing process. At its core, the system employs advanced algorithms to identify and classify electronic chips swiftly, leveraging techniques like YOLO for precise detection. Upon identification, a sorting algorithm determines the chips' destination based on predefined criteria such as type and size, guided by the sorting decisions made by the algorithm. Once retrieved, the Quanser arm strategically places the chips into their designated locations, ensuring accurate sorting and efficient workflow. Ultimately, this integrated solution offers significant improvements in speed, accuracy, and productivity, revolutionizing electronic chip sorting in manufacturing facilities.

### 3.2 Data Management and PCB Detection

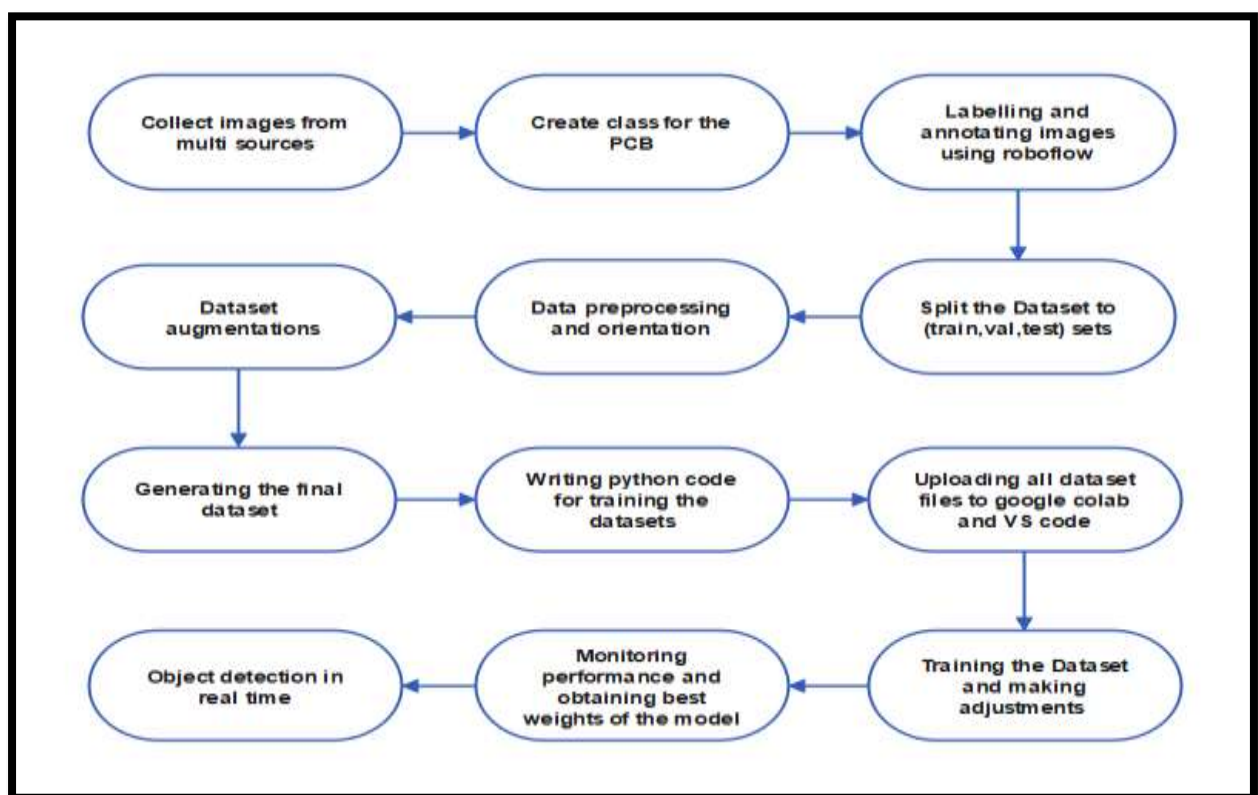


Figure 3-1:Data management and PCB detection sequence

### **3.2.1 Data selection**

At first, we aimed to develop and generate our dataset autonomously, without depending on pre-existing public datasets. However, we quickly grasped that achieving desirable performance in computer vision requires a significant volume of data, usually exceeding 1500 samples. Due to our restricted access to PCBs, obtaining such a dataset would demand substantial time and resources. Through research, we came across Roboflow platform, which offers a wide array of PCB datasets covering different shapes, colours, and electrical components [37].

Roboflow offers a robust platform tailored to simplify and optimize the management and annotation of visual data, specifically tailored for machine learning endeavours, with a primary emphasis on computer vision assignments.

Fundamentally, Roboflow serves as a centralized platform where individuals can effortlessly upload raw image data and efficiently convert it into annotated datasets primed for training machine learning algorithms. Through its user-friendly interface, it accommodates diverse annotation formats such as bounding boxes and key points, catering to a broad spectrum of computer vision tasks. Moreover, Roboflow provides robust data augmentation features, enriching dataset variety and bolstering model adaptability [38].

Roboflow seamlessly integrates with leading machine learning frameworks and platforms, such as TensorFlow, PyTorch, and Google Colab, among others. This interoperability guarantees alignment with current workflows and empowers users to utilize their preferred tools and environments effectively [39]. As a result, we imported these datasets into our workspace to kickstart our project.

### **3.2.2 Class labelling and annotation**

Labelling and annotating datasets for object detection applications holds paramount importance for several reasons. Primarily, these labelled datasets serve as the foundational training data for machine learning models. Object detection models, especially those employing deep learning architectures such as convolutional neural networks (CNNs), acquire the ability to recognize objects by studying examples presented in the training data. Labelled and annotated datasets, wherein the desired objects are tagged, furnish essential ground truth information about the objects depicted in images or videos. This information forms the bedrock for training these models [40].

Given our project's objective of detecting and classifying PCBs from industrial waste, it was only natural to establish a single class, which we labelled "PCB," and proceeded to annotate the PCBs while excluding other undesired objects from the dataset.

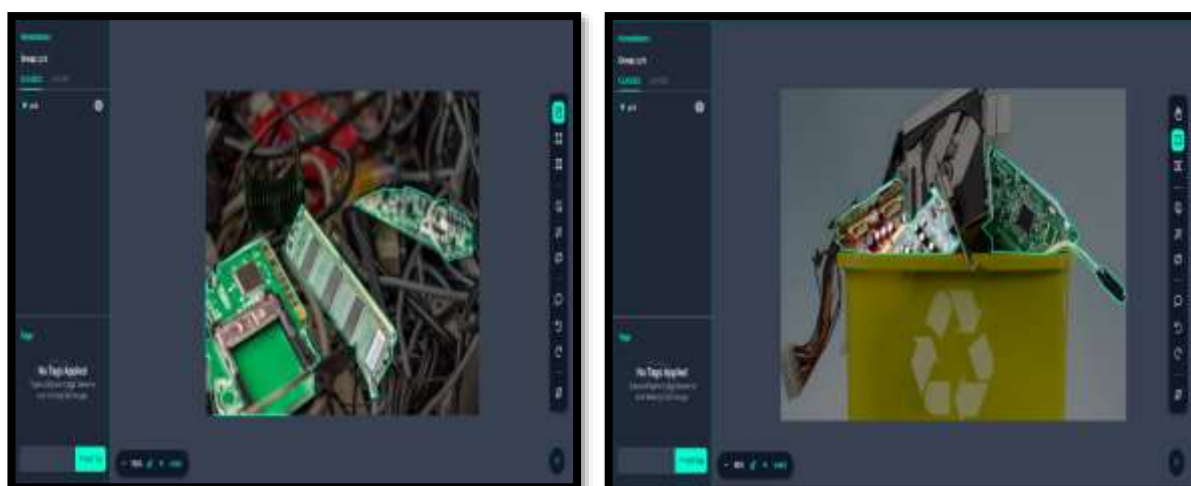


Figure 3-2: Printed Circuit Boards Labelling Process [38]

- Negative Examples:** Including images without the object of interest helps the model distinguish between the object and its absence. This provides context for the model to understand what it shouldn't be detecting and to avoid False Positives which means that Without negative examples, the model might erroneously detect the object in situations where it shouldn't be present. Including images without the object helps reduce false positives [41].

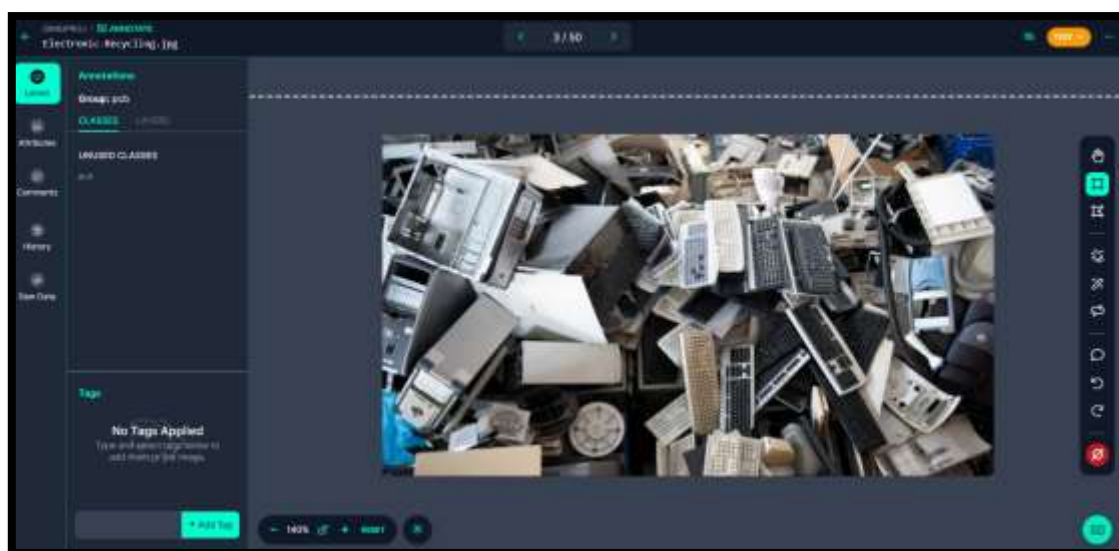


Figure 3-3:False Positive dataset [41]

### 3.2.3 Data split and adjustments

In this stage, we determine the allocation of our dataset into three distinct categories: Train, Validation, and Test. This split is crucial as it enables the evaluation of how effectively a machine learning model can generalize to unseen data. Furthermore, it serves as a safeguard against overfitting, wherein a model excels with the training data but struggles with new instances [42].



**Figure 3-4: Division of a dataset into three sets: Train set, Valid set, and Test set [42]**

The size of the training data is usually larger than that of the testing data. This is intentional, as we aim to provide the model with ample data to discern and understand significant patterns effectively. As the machine learning algorithm processes the data from our datasets, it assimilates these patterns to inform its decision-making process. The validation set serves as an initial assessment before the final evaluation on the test set [43].

Data adjustments in machine learning plays a pivotal role in improving data quality, facilitating the extraction of valuable insights. It involves cleaning and organizing raw data to render it apt for constructing and training machine learning models. This process aids in reducing training time and enhancing performance through the application of image transformations across the entire dataset [44].

To ensure optimal accuracy and precision for our YOLOv5 Model, we implement two crucial pre-processing steps. Firstly, auto-orientation is applied to strip the images of their EXIF data, ensuring uniform display regardless of how they're stored on disk. Secondly, we resize the images to a compatible size of 640 x 640, aligning them with the requirements of the YOLO5 model for effective training [45].

### 3.2.4 Dataset augmentation

Data augmentation involves expanding the training set by generating modified versions of existing data. This technique encompasses making minor alterations to the dataset or employing deep learning methods to produce new data points. Essentially, it entails artificially creating additional data from existing samples, primarily to facilitate the training of new machine learning (ML) models [46].

In our project we applied several augmentation steps which tripled the amount of the dataset we have, and they were:

1. **Flipping:** Flipping an image (and its annotations) is a deceptively simple technique that can improve model performance in substantial ways [47].



**Figure 3-5: Flip augmentation [47]**

2. **Hue:** Hue augmentation randomly alters the color channels of an input image, causing a model to consider alternative color schemes for objects and scenes in input images. This technique is useful to ensure a model is not memorizing a given object or scene's colours. While output image colours can appear odd, even bizarre, to human interpretation, hue augmentation helps a model consider the edges and shape of objects rather than only the colours [48].



**Figure 3-6: Hue augmentation [48].**

3. **Mosaic:** It works by taking four source images and combining them together into one. This does a few things, It simulates four random crops (while maintaining the relative scale of your objects compared to the image) which can help your model perform better in cases of occlusion and translation and It combines classes that may not be seen together in your training set (for example, if you have pictures of apples and pictures of oranges, but no pictures of apples with oranges in the same photo, mosaic will simulate that) [49].





**Figure 3-7: Mosaic augmentation [49].**

### **3.2.5 Generating dataset version**

A version represents a snapshot of the dataset at a specific moment in time. Versions meticulously document the images included, pre-processing steps applied, and augmentation techniques utilized in every iteration of your model, ensuring the reproducibility of results.

The final dataset was uploaded to the Google Colab website, in addition to a YAML file (YAML is a data serialization language used to create configuration files with any programming language) [50]. The YAML file containing model configuration and class values. YOLOv5 algorithm was used to train the model since it provides objectives the fit the project requirements.

### **3.2.6 Dataset training**

In this section a typical training process is discussed. Google Colab was used to train the model. Google Colab is a free Jupyter notebook that allows the user to write and execute Python code, and it supports many machine learning libraries. 12 GB of GPU Tesla 4 type is allocated to each user. After uploading the required files on the site and training the model, a PyTorch file containing the model weights has been created and then was moved to the CPU of the robotic arm PC to be used by the YOLOv5 algorithm [51].

First, we imported the dataset that we gathered and labelled using Roboflow API key and inserted it in google Colab and the dataset was exported in PyTorch YOLOv5s format and PyTorch stands as a robust open-source machine learning framework for Python, crafted by Facebook's AI Research lab (FAIR). It boasts a dynamic computational graph, simplifying the creation and troubleshooting of deep learning models Plus, PyTorch has a big community of users who share tips and solutions, making it easier to troubleshoot any problems you run into. Another big benefit is that PyTorch works well with other popular tools and libraries.



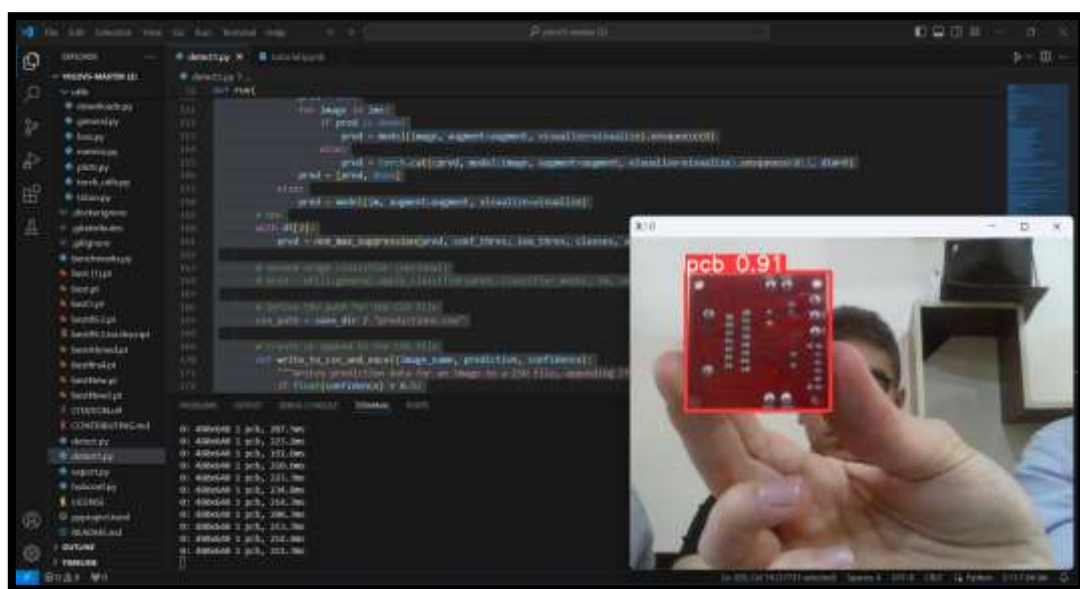
Then we started the actual training process by running the training code passing several arguments such as:

- Input image size: 640x640 which is the compatible size for YOLOv5s.
- Batch size: we chose 64 batch for our dataset to have a combination of fast and accurate dataset training.
- Number of epochs: we found out that 85 epochs was the number of epochs that gave us the best accuracy and without causing an overfitting.

We trained our dataset with 85 epochs and it lasted approximately 1.6 hours using the T4 Microsoft google Colab GPU and we had 157 layers. It gave an Overall Precision of 0.96 out of 1 and Overall Recall of 0.97 out of 1, Accuracy of 98.5% and Performance of 88.1% and by completing these steps we can acquire the actual model as the best weights we got from one of the epochs that gave the best accuracy and precision.

### 3.2.7 Real-time object detection

After obtaining the YOLOv5 model best weights we implemented it in a python code we wrote using Visual Studio code (VScode) platform to detect PCBs in real time that involves both Image Classification and Object Localization. Image Classification is where an algorithm is applied to an image to predict the class of one object, and object localization predicts the class of objects if it figures out the location of the detected object by drawing a bounding box around it. [52].



**Figure 3-8:camera usage at real time**

The process initiates with the camera system detecting Printed Circuit Boards (PCBs) within its field of view, employing image processing techniques for accurate identification. Once a PCB is detected, pertinent information is appended to a text file. This file is configured to overwrite itself with each new

detection, ensuring it always contains the latest data without redundancy. The text file serves as input for MATLAB/Simulink, then the confidence of the detection process will be compared with a specific value (can be adjusted by the user). If the text file result from the comparison process gives 1, then it will give a command to the robotic arm to start Pick and Place mechanism, else null won't make the Pick and Place movement. This entire process typically operates in real-time or near-real-time, enabling swift and efficient interaction with detected PCBs.

### **3.3 Motion Control**

#### **3.3.1 QUARC Software**

Quarc, a product of Quanser, is a real-time control software tailored for designing, simulating, and deploying control algorithms within the MATLAB/Simulink environment. Its seamless integration allows users to swiftly prototype and implement control strategies on real-time hardware like Quanser's platforms. With Quarc, users can model dynamic systems, design controllers, and interface with hardware effortlessly. It's a popular choice in academia and industry for teaching, research, and development across fields like robotics, aerospace, and automotive engineering [53].

QUARC software plays a critical role in pick-and-place application of our project involving the Quanser Arm. QUARC excels in seamlessly integrating MATLAB/Simulink with real-time hardware platforms, including Quanser's systems, thereby simplifying the workflow for users to design, simulate, and deploy control algorithms within their familiar software environments. Moreover, QUARC's support for real-time control implementation ensures the effective execution of control algorithms on hardware platforms with stringent timing requirements, crucial for applications like robotics and mechatronics. Additionally, QUARC provides comprehensive tools for hardware interfacing, enabling effortless deployment of control algorithms onto Quanser hardware for real-world testing and experimentation purposes [54].

#### **3.3.2 Sequence of pick and place mechanism**

Our project revolves around developing a pick and place mechanism utilizing the Quanser Arm. To execute this, a thorough understanding of the QArm and its functioning is imperative. This encompasses familiarity with the QArm's workspace, as well as proficiency in determining both end effector position and joint angles, facilitated by employing DH parameters for inverse and forward kinematics computation. Additionally, as our QArm is equipped with a gripper for retrieving PCBs from a designated pick point, it is essential to grasp the methodology involved in handling it, a topic we will go deep into further in this section.

- **Workspace:** -

Kinematics deals with understanding the geometric and time-related aspects of object motion, excluding the forces and moments causing that motion. When analysing intricate manipulators, coordinate frames are affixed to different parts, such as the fixed base frame and the end-effector frame attached to the robotic arm's end. Manipulator kinematics examines how these frames' positions and orientations change across various configurations, determined by the robot arms' joint angles. The collection of reachable positions and orientations delineates the manipulator's workspace.

Now to determine the workspace of the Quanser Arm, we would typically engage in a process involving both theoretical analysis and practical experimentation. The Quanser Arm, like many robotic manipulators, possesses joint limitations and physical constraints that define its reachable space. Firstly, we would delve into its kinematic structure, understanding the number and types of joints it comprises, along with their range of motion. This analysis often involves studying the robot's forward and inverse kinematics equations, which describe how joint movements translate into end-effector positions in three-dimensional space.

With this theoretical groundwork, we can then proceed to experimentally explore the arm's capabilities. This may involve systematically moving each joint through its entire range of motion while recording the resulting end-effector positions. By aggregating these data points, you can construct a representation of the arm's workspace, delineating the regions of space it can access. Advanced techniques such as numerical simulations or optimization algorithms may also be employed to refine and visualize the workspace boundaries more precisely.

Robotic manipulators are generally constructed from joints and links. Joints can be revolute, meaning they rotate, or prismatic, meaning they translate. Although joints can have multiple degrees of freedom (DOF), which this QArm is consist of 4-DOF with all revolute joints.

The standard Deviant-Hartenberg (DH) convention will be used to assign joint frames  $\{0\}$  to  $\{4\}$  and find the homogenous transformations between the frames. Each homogeneous transformation  $T_i^{i-1}$  between frame  $\{i-1\}$  and frame  $\{i\}$  is represented as a product of four "basic" translation/rotation transformations.

$$\begin{aligned}
 {}^{i-1}T_i &= Rot_{z,\theta} Trans_{z,d} Trans_{x,a} Rot_{x,\alpha} \\
 &= \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & 0 \\ \sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha_i & -\sin \alpha_i & 0 \\ 0 & \sin \alpha_i & \cos \alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} \cos \theta_i & -\sin \theta_i \cos \alpha_i & \sin \theta_i \sin \alpha_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}
 \end{aligned}$$

The QArm that we use it in this project it dimensions are as the following:

$$L_1 = 0.14m, L_2 = 0.35m, L_3 = 0.05m, L_4 = 0.25m, \text{ and } L_5 = 0.15m$$

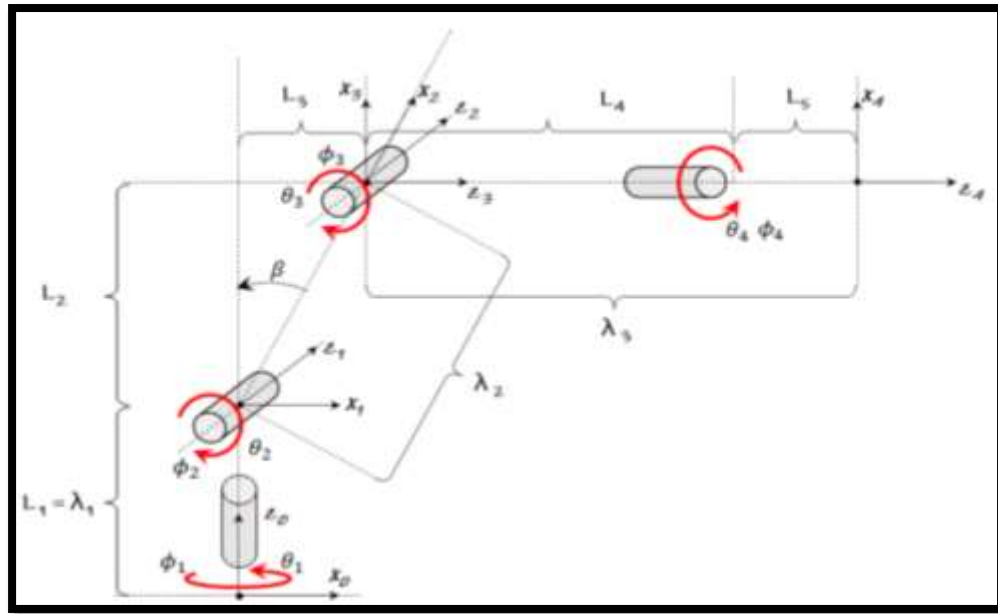
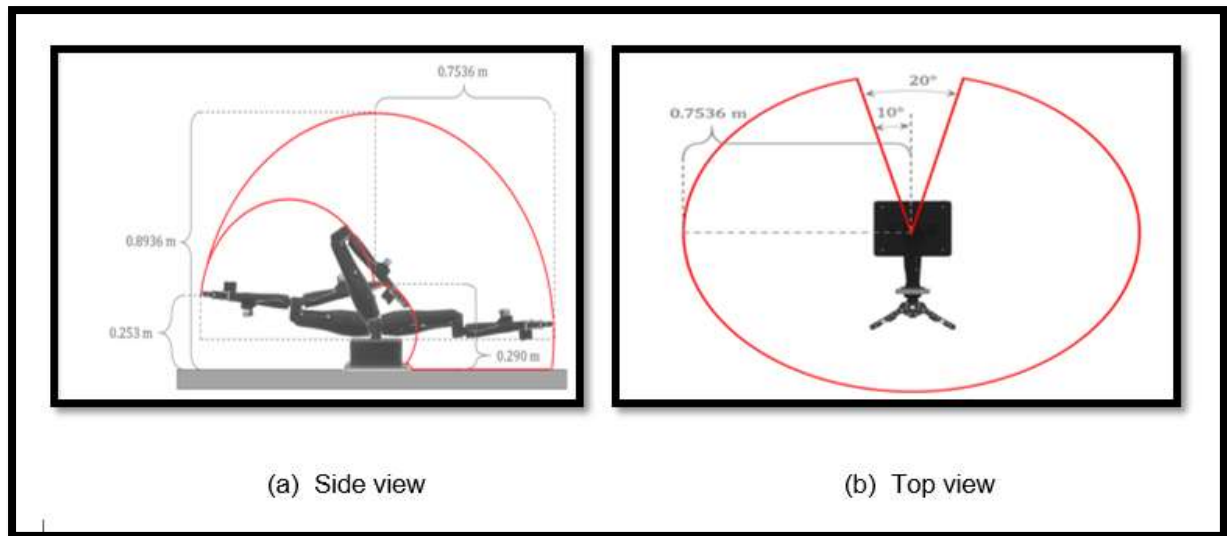


Figure 3-9: Frame diagram for the Quanser Arm manipulator

Which we will use them to make the following DH-Table:

Table 3-1: DH- Table

$i$	$a_i$	$\alpha_i$	$d_i$	$\theta_i$
1	0	$-90^0$	0.14	$\theta_1$
2	0.35	0	0	$\theta_2$
3	0	$-90^0$	0	$\theta_3$
4	0	0	0.05	$\theta_4$



**Figure 3-10: Workspace of the QArm**

- **Forward kinematics:**

Forward kinematics is a fundamental concept in robotics and computer graphics that describes the relationship between the individual parts of a mechanism and its overall configuration or pose. In robotics, it refers to determining the position and orientation of the end-effector (such as a robot arm or gripper) based on the joint angles or lengths of its segments. This process involves a series of transformations starting from a known reference frame, typically the base of the robot, and moving through each joint, applying rotation and translation operations according to the joint parameters. Each transformation is represented by a homogeneous transformation matrix, combining rotation and translation components, which allows for the seamless sequence of transformations along the robot's kinematic chain. By multiplying these transformation matrices together, the forward kinematics equations result in a transformation matrix that represents the end-effector's position relative to the base frame.

Overall, forward kinematics provides a means to compute the end-effector's position and orientation in space.

Now we display the mathematical equations that resulted from Workspace to obtain the end's effector position using the forward kinematics technique as following:

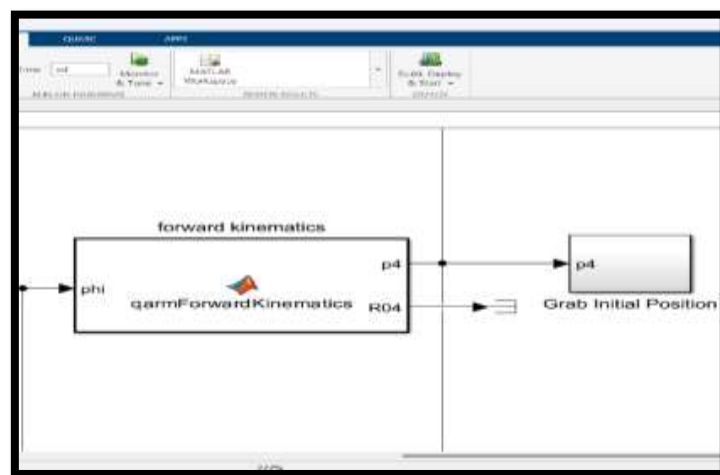
**Table 3-2:Linear mapping for the mathematical operations**

New parameter	Original Parameter
$\lambda_1$	$L_1$
$\lambda_2$	$\sqrt{L_2^2 + L_3^2}$
$\lambda_3$	$L_4 + L_5$
$\beta$	$\tan^{-1}(L_3/L_2)$
$\phi_1$	$\theta_1$
$\phi_2$	$\theta_2 + \frac{\pi}{2} - \beta$
$\phi_3$	$\theta_3 + \beta$
$\phi_4$	$\theta_4$

$$\begin{aligned} \vec{p} &= f_{FK}(\vec{\theta}) \\ p_x &= \lambda_2 c_1 c_2 - \lambda_3 c_1 s_{23} \\ p_y &= \lambda_2 s_1 c_2 - \lambda_3 s_1 s_{23} \\ p_z &= \lambda_1 - \lambda_2 s_2 - \lambda_3 c_{23} \end{aligned} \quad (2)$$

where s and c represent the trigonometric functions sine and cosine, respectively,  $\lambda_1$  through  $\lambda_3$  are arm-specific constants,  $\vec{\theta}$  is the vector representing the joint states,

Figure 3-14 below shows the Simulink Block for the Forward kinematics used in the project.


**Figure 3-11:Forward kinematics snapshot**

- **Inverse kinematics**

Inverse kinematics is a crucial concept in robotics, which offers a solution to determine the joint configurations necessary to achieve a desired end-effector position and orientation. Unlike forward kinematics, which computes the position of the end-effector given the joint parameters, inverse kinematics works backward, deriving the joint angles required to attain a specified end-effector pose. This process involves solving a set of nonlinear equations or optimization problems, aiming to minimize the error between the desired and actual end-effector positions and orientations. In robotics, inverse kinematics finds applications in tasks such as trajectory planning, motion control, and obstacle avoidance, allowing robots to navigate complex environments and manipulate objects with precision. Inverse algorithms have facilitated their implementation in various real-world applications, enhancing the flexibility and versatility of robotic systems.

The purpose of the Inverse kinematics in our project is to calculate the joint angles for the links so each one of them could perform its own movement resulting in the end effector reaching its desired destination, now we show the mathematical formulations to calculate the joint angles in full details:

$$\vec{\theta} = f_{IPK}(\vec{p}) \quad (3)$$

Begin with the equations describing the end-effector's  $p_x$  and  $p_y$ .

$$\begin{aligned} p_x &= \lambda_2 c_1 c_2 - \lambda_3 c_1 s_{23} = c_1 (\lambda_2 c_2 - \lambda_3 s_{23}) \\ p_y &= \lambda_2 s_1 c_2 - \lambda_3 s_1 s_{23} = s_1 (\lambda_2 c_2 - \lambda_3 s_{23}) \end{aligned} \quad (4)$$

Next, we square and then add the equations describing  $p_x$  and  $p_y$ , giving,

$$\begin{aligned} p_x^2 + p_y^2 &= c_1^2 (\lambda_2 c_2 - \lambda_3 s_{23})^2 + s_1^2 (\lambda_2 c_2 - \lambda_3 s_{23})^2 = (\lambda_2 c_2 - \lambda_3 s_{23})^2 \\ \pm \sqrt{p_x^2 + p_y^2} &= \lambda_2 c_2 - \lambda_3 s_{23} \end{aligned} \quad (5)$$

By substituting equation 2 and 5 we get the following set of equations:

$$\begin{aligned} \pm \sqrt{p_x^2 + p_y^2} &= \lambda_2 c_2 - \lambda_3 s_{23} \\ \lambda_1 - p_z &= \lambda_2 s_2 + \lambda_3 c_{23} \end{aligned} \quad (6)$$

This resembles the following trigonometric equations

$$\begin{aligned} A \cos \theta_2 + B \cos(\theta_2 + \theta_3) + C \sin(\theta_2 + \theta_3) &= D \\ A \sin \theta_2 + B \sin(\theta_2 + \theta_3) - C \cos(\theta_2 + \theta_3) &= H \end{aligned} \quad (7)$$

**Table 3-3:what symbols resemble**

Symbol	Resemblance
$A$	$\lambda_2$
$B$	$0$
$C$	$-\lambda_3$
$D$	$\pm\sqrt{p_x^2 + p_y^2}$
$H$	$\lambda_1 - p_z$
$F$	$\frac{D^2+H^2-A^2-C^2}{2A}$

Because there are two possible values for  $D$ . This results in two possible solutions for  $\theta_3$  which are:

$$\begin{aligned}\theta_{3,1} &= 2 \tan^{-1} \left( \frac{C + \sqrt{C^2 + F^2}}{F} \right) \\ \theta_{3,2} &= 2 \tan^{-1} \left( \frac{C - \sqrt{C^2 + F^2}}{F} \right)\end{aligned}\tag{8}$$

Before solving for  $\theta_2$ , mapping variables are required for calculations:

**Table 3-4:Mapping variables to calculate  $\theta_2$**

Symbol	Resemblance
$M$	$A + C \sin \theta_3$
$N$	$-C \cos \theta_3$
$\cos \theta_2$	$\frac{DM+HN}{M^2+N^2}$
$\sin \theta_2$	$\frac{H-N \cos \theta_2}{M}$

This results in the following equation for  $\theta_2$

$$\theta_2 = \tan^{-1} \left( \frac{\sin \theta_2}{\cos \theta_2} \right)\tag{9}$$



Now we can obtain  $\theta_1$  by using this equation:

$$\tan \theta_1 = \frac{p_y/\lambda_2 c_2 - \lambda_3 s_{23}}{p_x/\lambda_2 c_2 - \lambda_3 s_{23}} \gg \theta_1 = \tan^{-1} \left( \frac{p_y/\lambda_2 c_2 - \lambda_3 s_{23}}{p_x/\lambda_2 c_2 - \lambda_3 s_{23}} \right) \quad (10)$$

$\theta_4$  which is the wrist angle and its basically the gripper orientation can be given as:

$$\theta_4 = \gamma \quad (11)$$

Figure 3-15 below shows the inverse kinematics costum block used on simulink:

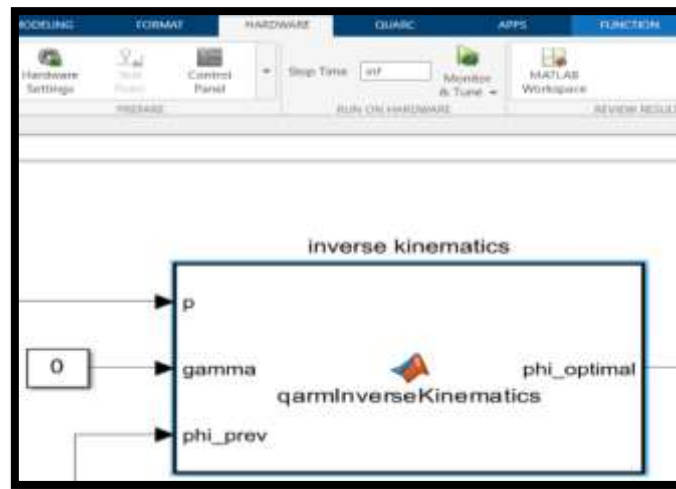


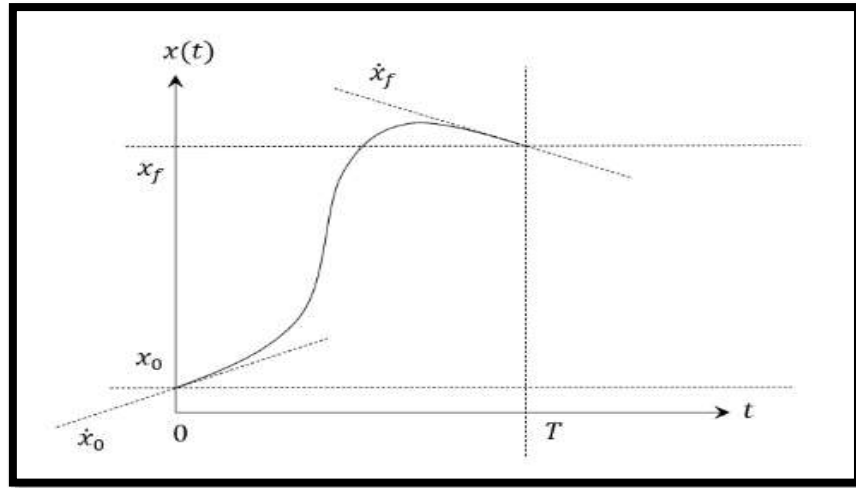
Figure 3-12:Inverse kinematics snapshot

- **Cubic Spline**

Cubic splines are necessary in robotics, serving various crucial functions within the field. Primarily, they excel in generating smooth trajectories essential for precise and efficient robotic motion. Whether for robotic arms, mobile robots, or other systems.

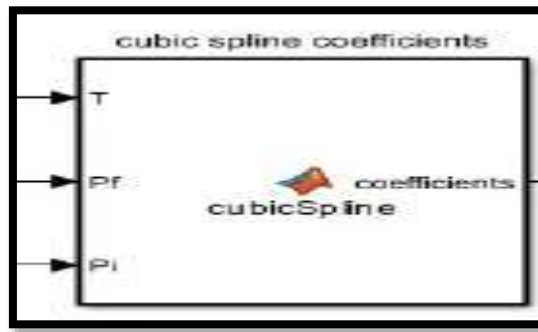
In Simulink, the Cubic Spline block is used for interpolation purposes. It computes interpolated values using cubic spline interpolation methods that is used to construct a smooth curve that passes through a given set of points. It works by fitting a cubic polynomial between each pair of adjacent points, ensuring that the resulting curve is smooth and continuous.

In summary the cubic spline is useful for generating continuous curves (smooth motion) from discrete data points. Figure 3-16, shows the cubic spline initial and final position graph:



**Figure 3-13: Trajectory Spline of initial and final positions**

The following figure 3-17, shows the block of the cubic spline:



**Figure 3-14: Cubic Spline snapshot**

As shown above the block has three inputs which are: Periodic time, Initial Position, and final position and there will be one output which is a matrix that contains the coefficients of the third polynomial equation which will be shown below:

$$\theta(t) = a_0 + a_1t + a_2t^2 + a_3t^3 \quad (12)$$

After we derived the above equation(position equation) we got the following equation(velocity equation):

$$\dot{\theta}(t) = a_1 + 2a_2t + 3a_3t^2 \quad (13)$$

The Summation of the velocity and position equations above results in the following matrix:

$$\begin{bmatrix} \theta_i \\ \dot{\theta}_i \\ \theta_f \\ \dot{\theta}_f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad (14)$$

Given the known initial location and orientation of the robot, the final joint angles corresponding to the desired position and orientation can be determined using inverse kinematic equations. A common approach involves employing a polynomial to plan the joints movements, ensuring that the initial and final boundary conditions become equals to their specified values. Specifically, this entails specifying known values for  $\theta_i$  and  $\theta_f$ , as well as the velocities at the onset and conclusion of the motion segment, which may either be known or set to zero. Consequently, applying this third-order polynomial to each joint motion results in a motion profile that drives each joint to result in the end effector desired position.

Note: we can change the speed of the arm by adjusting the value of time(t).

- **Waypoint Navigator**

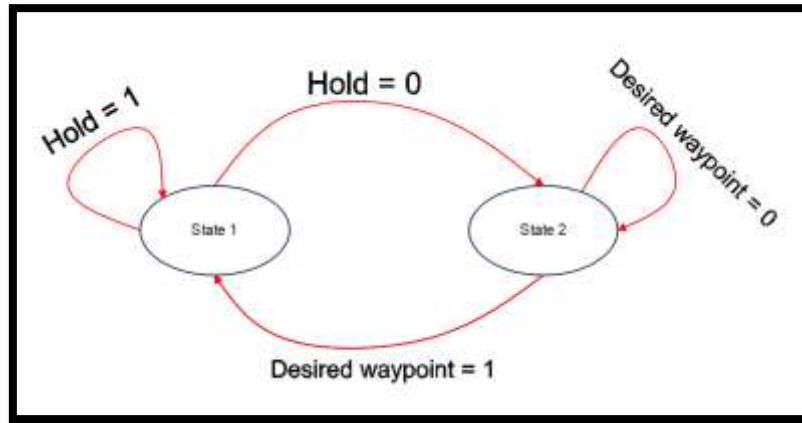


**Figure 3-15: Waypoint Navigator snapshot**

At the beginning of the path, the current position and speed align with the initial position and speed set by the trajectory generator. As the trajectory progresses, the desired set point and speed represent the final position and speed. Upon reaching the final set point, these points become the new initial position and speed for subsequent tasks, while the final position and speed set points shift to the next waypoint.

To manage this transition effectively and ensure the manipulator reaches desired set points, a waypoint generator is essential. One straightforward approach to handle this is by employing a Finite State Machine (FSM). FSMs are simplified versions of Turing Machines, where the system can transition between a finite number of unique states. Based on available measurements, the FSM determines the next action and state transition at each iteration.

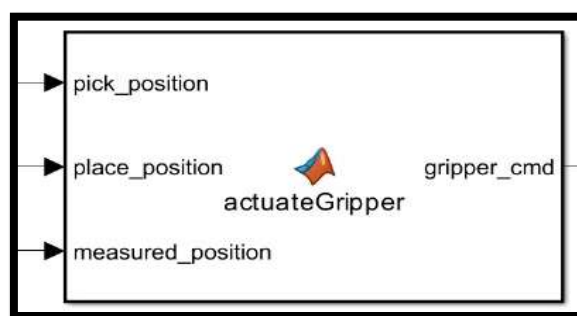
The following state flow describe that when the user holds the last position, then the desired set point of the arm will be in the same Current position and the next state will be state 1. But if the user doesn't hold the last position that mean the desired set point will be the next state which will be state 2. This will be also happened in vice versa image if the arm is in the state 2 and it want to go to the next position.



**Figure 3-16:Waypoint navigator state machine**

This way, you will be able to provide the set of waypoints for any task like pick/place, and the waypoint navigator will cycle through them. After the last waypoint, the navigator will simply pick the first waypoint again of course this is considered as a fundamental step to understand the application of our application since our project is more complex than just going for a known waypoint, but it will be discussed later the adjustments we made to it.

- **Gripper Controller**

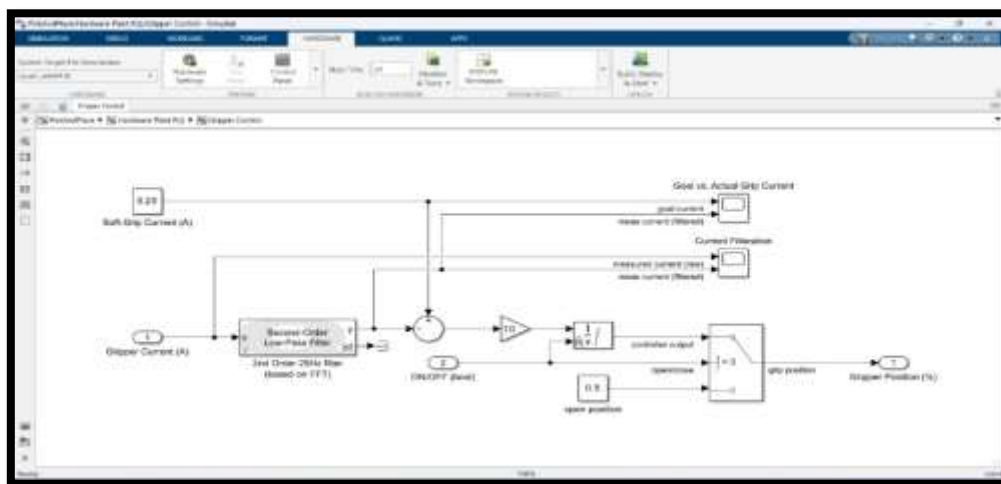


**Figure 3-17:Gripper actuator snapshot**

The MATLAB function actuate Gripper is designed to autonomously control the operation of a gripper mechanism based on the robot's measured position in relation to predetermined pick and place locations. Initially, the internal variable gripper\_state is initialized as a fixed variable, retaining its value across function calls, and is set to 0, indicating that the gripper is in an open

state. The distance threshold for gripper actuation is defined by the variable threshold, specifying the proximity within which the gripper should respond to the pick and place positions. The function's logic dictates that if the measured position of the robot is within the defined threshold distance of the pick position and the gripper is currently in an open state, the gripper is actuated to close. On the other hand, if the measured position of the robot is within the threshold distance of the place position and the gripper is currently closed, the gripper is actuated to open. The output of the function, gripper\_cmd, reflects the current state of the gripper (either 0 for open or 1 for closed), based on the internal gripper\_state. In essence, this function continuously monitors the robot's position and triggers gripper actions as required, ensuring efficient and precise handling of objects during pick-and-place operations.

For the Gripping control we implemented a PI controller with comparison switch in the Hardware plant to control the opening and speed of the gripper and to not exceed the current limit of the servo motor which will respectively increase the amount of torque applied to the gripper which will cause the gripper motor to breakdown. the following is the block diagram of the protection circuit of the gripper servo motor: -



**Figure 3-18: Gripper current protection circuit**

The protection circuit contains several components, and we will talk about them as follows:

1. Soft gripper current: Improves the smoothness of motion of the gripper, the value 0.25A is the peak current that gives the best performance.
2. Second-Order Low-pass filter: this filter doesn't allow frequencies above 25Hz to go through to the gripper so the gripper speed that felt smooth was at the cut-off frequency equals to 157rad/sec ( $25 \times 2 \times \pi$ ).
3. PI controller: by adjusting the gain of the integral controller we can change the opening of gripper as our project needs.
4. Open position constant: by adjusting this value we can control the maximum opening of the gripper.

### 3.4 Final System

The figure 3-21 below shows the flowchart for the entire project system. It is presented as a series of blocks that explain the algorithm of the Pick-Place of PCBs in Real-time.

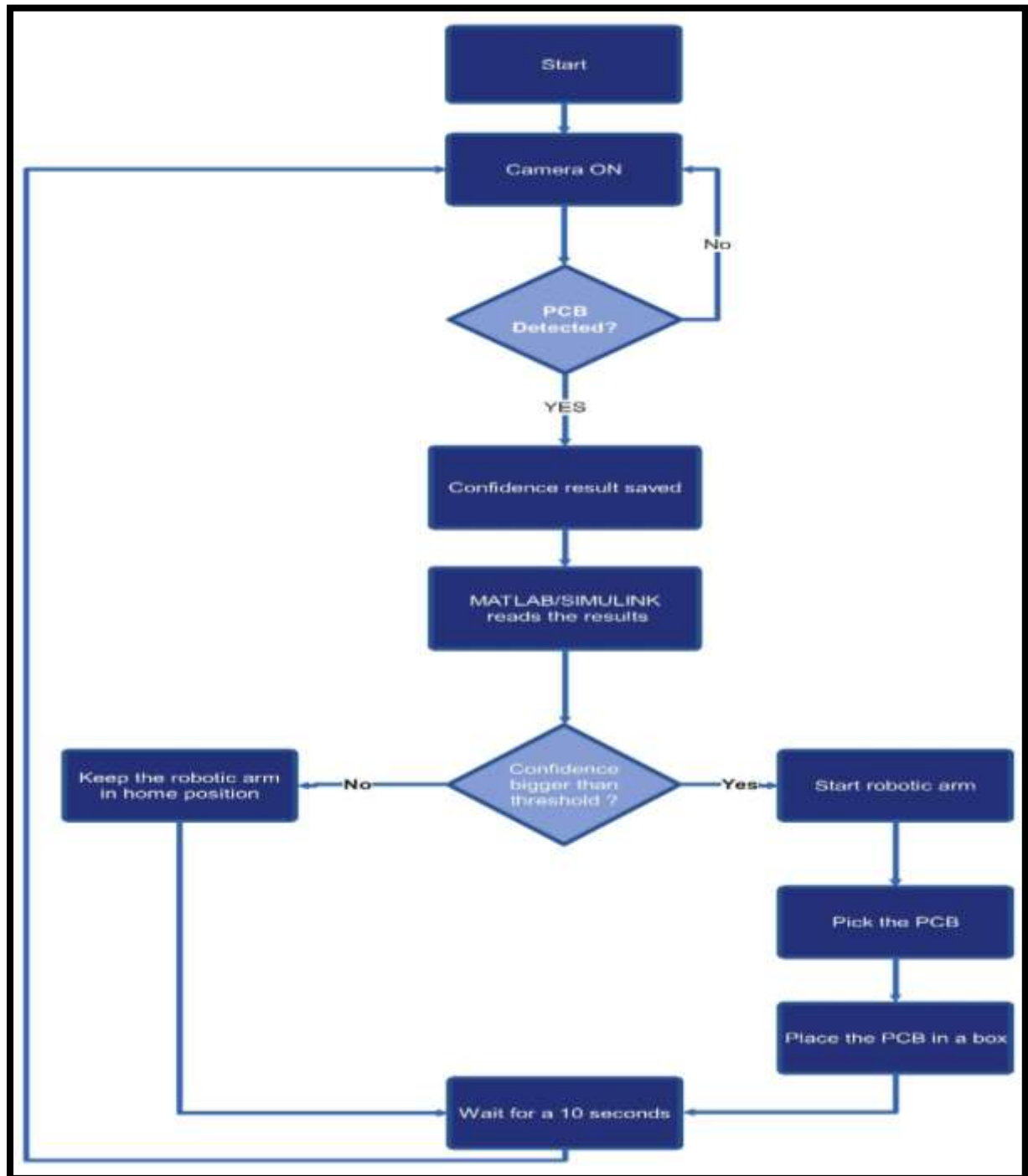


Figure 3-19:Real-time Pick-Place Flowchart

In Figure 3-22 below, we will demonstrate the full screen of the block that we used to facilitate the process of reading the object detection results. Subsequently, we compare these results with a specific value, enabling the robotic arm to receive the command to execute the pick-and-place mechanism for the PCB detected by the camera.

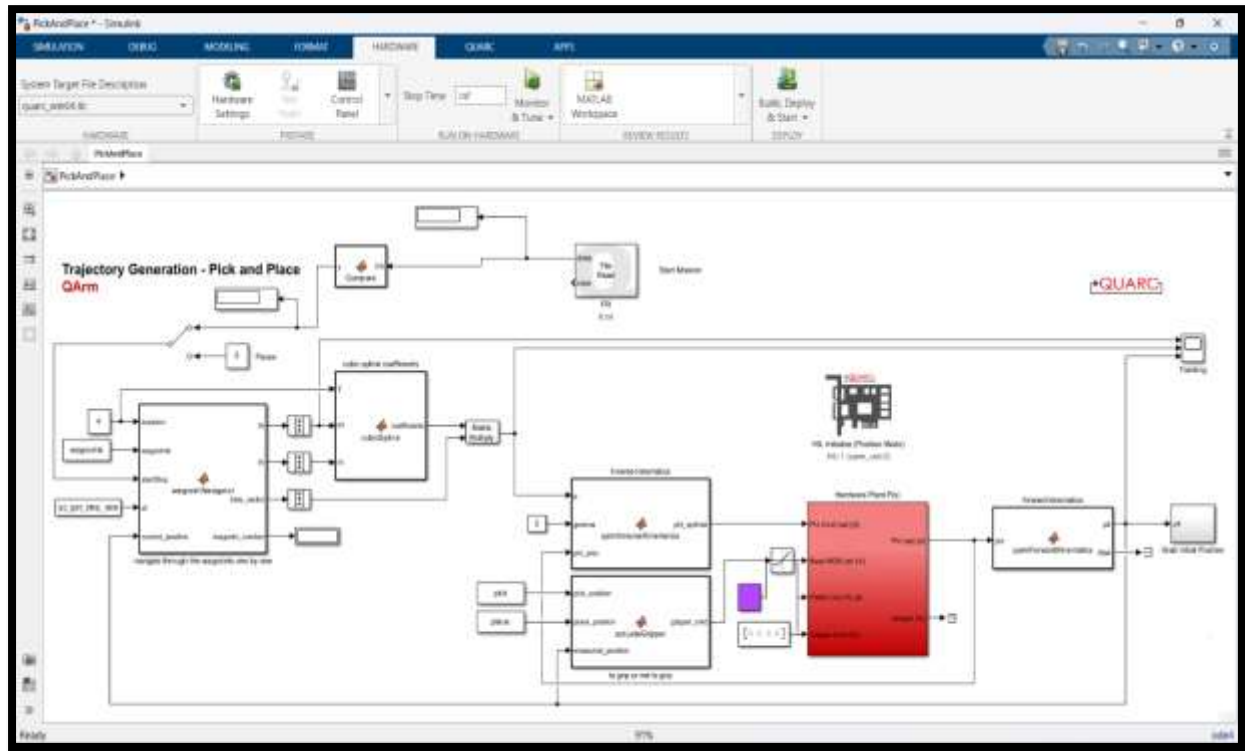


Figure 3-20: Full system blocks in Simulink

# Chapter 4 DESIGN TESTING AND RESULTS

## 4.1 Data management and PCB detection results

Training losses and performance metrics were done using Tensor board and the following results were given:

### 1. F1 curve

The F1 Confidence Curve illustrates the harmonic mean of precision and recall, showcasing how the F1 score varies across different confidence thresholds. By merging precision and recall measures, the F1 score identifies the optimal confidence threshold where both precision and recall yield the highest F1 value. This score represents the balanced average of precision and recall, with each metric contributing equally to its computation. A perfect F1 score is 1, indicating all predictions were accurate, while the worst score is 0. Thus, a model achieving an F1 score of 1 demonstrates flawless performance, with all predictions being correct [55].

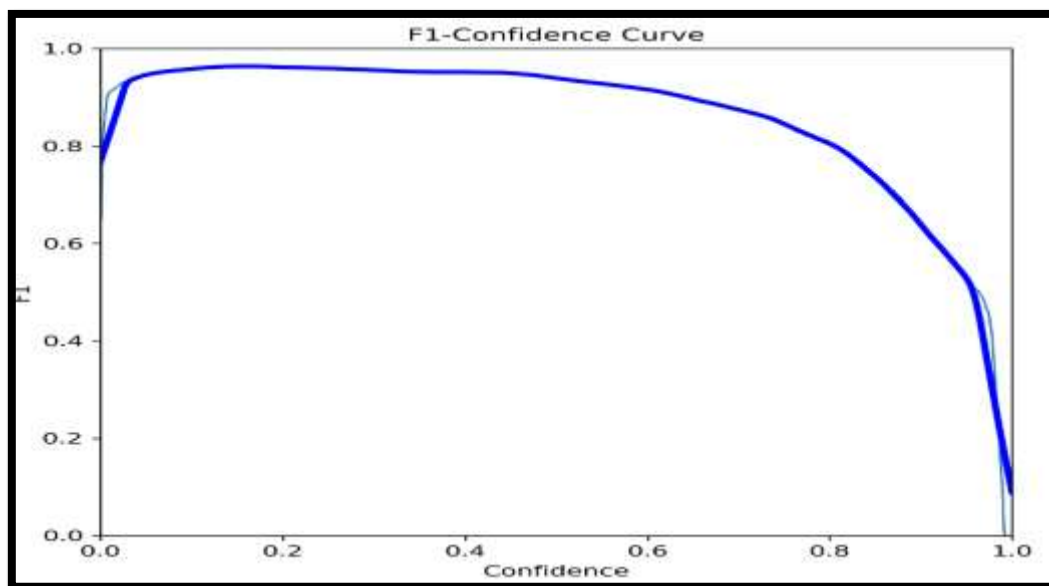


Figure 4-1: Training F1 curve

### 2. The precision-recall curve

The precision-recall curve, typically depicted graphically, illustrates the reciprocal relationship between recall and precision within a computer vision model. A larger area under this curve indicates the model's robust performance in both recall and precision, while a smaller area suggests weaker performance in one or both metrics.

Models incorporating confidence levels can strategically balance precision and recall by adjusting the required confidence threshold for making predictions. The precision-recall curve is generated by plotting the model's precision and recall against its confidence threshold [56]. This curve



demonstrates a downward trend because decreasing confidence leads to more predictions (boosting recall) but also less precise predictions (affecting precision negatively).

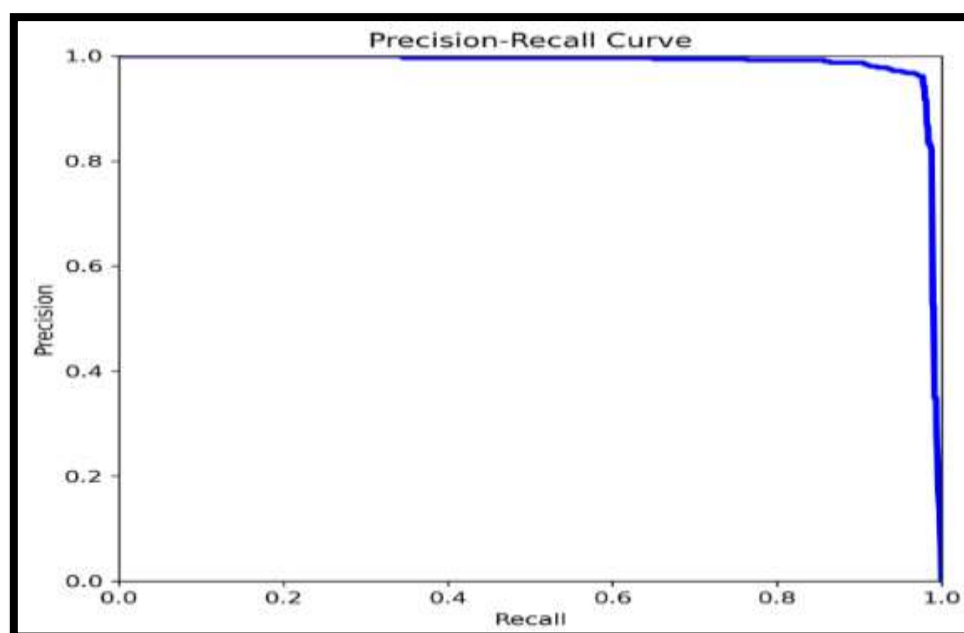


Figure 4-2: Training Precision-Recall curve

### 3. Confusion Matrix

The confusion matrix is a concise table employed to assess how well a classification model performs. It condenses the model's correct and incorrect predictions into categories such as true positives, true negatives, false positives, and false negatives. By organizing these outcomes, the matrix provides key metrics like accuracy, precision, recall, and F1 score, aiding in the evaluation and enhancement of the model's effectiveness.

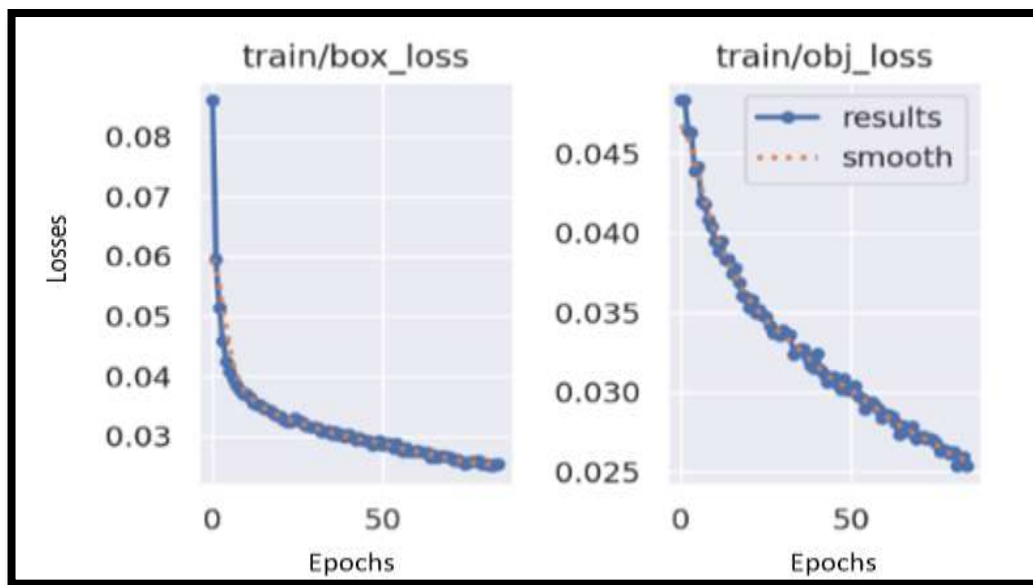


Figure 4-3: Training Confusion Matrix

Other metrics were found and gave high accuracy and good performance with very low losses, but are not as important as the previous metrics so we summed it up as follows:

- **Losses Graphs:**

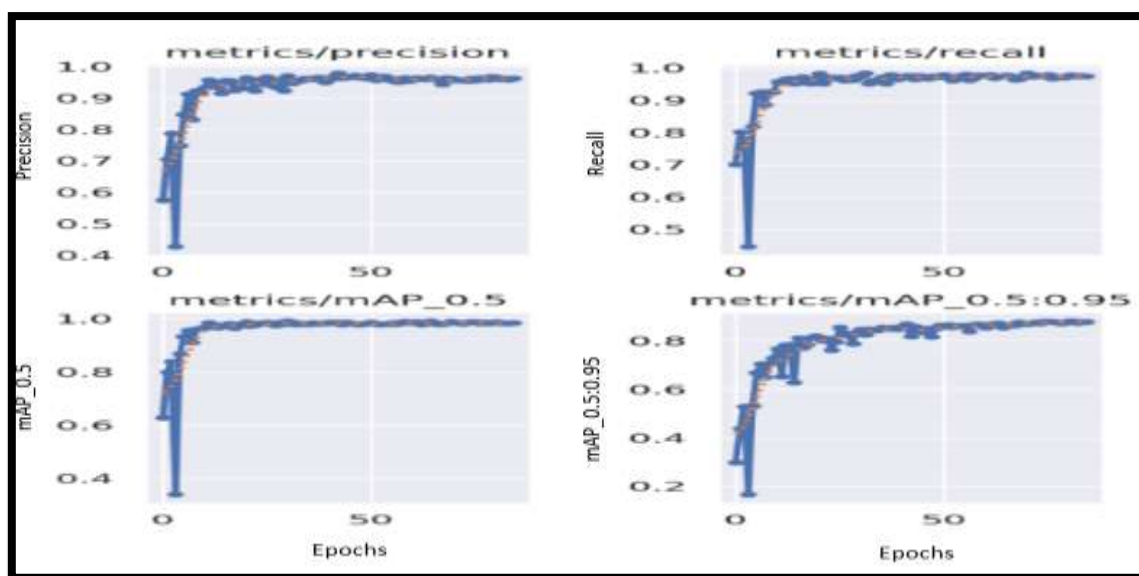
During YOLOv5 model training, the loss graphs illustrate the development of essential loss elements, including box loss, objectness loss (obj\_loss), and class loss (cls\_loss). Box loss compares predicted bounding box coordinates with ground truth values, while objectness loss evaluates the accuracy of predicting object presence within bounding boxes, penalizing incorrect predictions. Similarly, class loss measures the difference between predicted class probabilities and actual class labels [57]. As seen in the following figure 4-4:



**Figure 4-4: Training metrics graphs [57]**

- **Precision/Recall and mAP50 graphs**

The precision/recall and mAP50 (mean Average Precision at 50% IoU threshold) graphs in object detection models like YOLOv5, monitor the model's performance concerning precision, recall, and average precision across different confidence thresholds. Precision gauges the accuracy of correctly identified objects among all predicted ones, while recall measures the completeness of correctly identified objects among all true objects. The mAP50 graph indicates the average precision across all classes at a specific Intersection over Union (IoU) threshold (typically 0.5), offering an overall evaluation of the model's object detection prowess [58]. As seen in figure 4-5 below:



**Figure 4-5: Metrics for precision and recall and mean average precision [58]**

The figure 4-6 below shows the training results of the Colab code for YOLOv5 model and the results of the PCB detection and classification accuracies and performance were shown as follows:

Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
82/84	11.3G	0.01408	0.006913	0	141	640: 100% 57/57 [01:05<00:00, 1.15s/it]
Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 4/4 [00:07<00:00, 1.81s/it]
all	438	435	0.954	0.966	0.976	0.845
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
83/84	11.3G	0.01363	0.006862	0	126	640: 100% 57/57 [01:02<00:00, 1.10s/it]
Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 4/4 [00:13<00:00, 3.40s/it]
all	438	435	0.953	0.968	0.974	0.847
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
84/84	11.3G	0.01424	0.007086	0	117	640: 100% 57/57 [01:05<00:00, 1.16s/it]
Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 4/4 [00:13<00:00, 3.39s/it]
all	438	435	0.958	0.966	0.976	0.848

85 epochs completed in 1.782 hours.  
 Optimizer stripped from runs/train/exp/weights/last.pt, 14.4MB  
 Optimizer stripped from runs/train/exp/weights/best.pt, 14.4MB

Validating runs/train/exp/weights/best.pt...  
 Fusing layers...  
 Model summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs

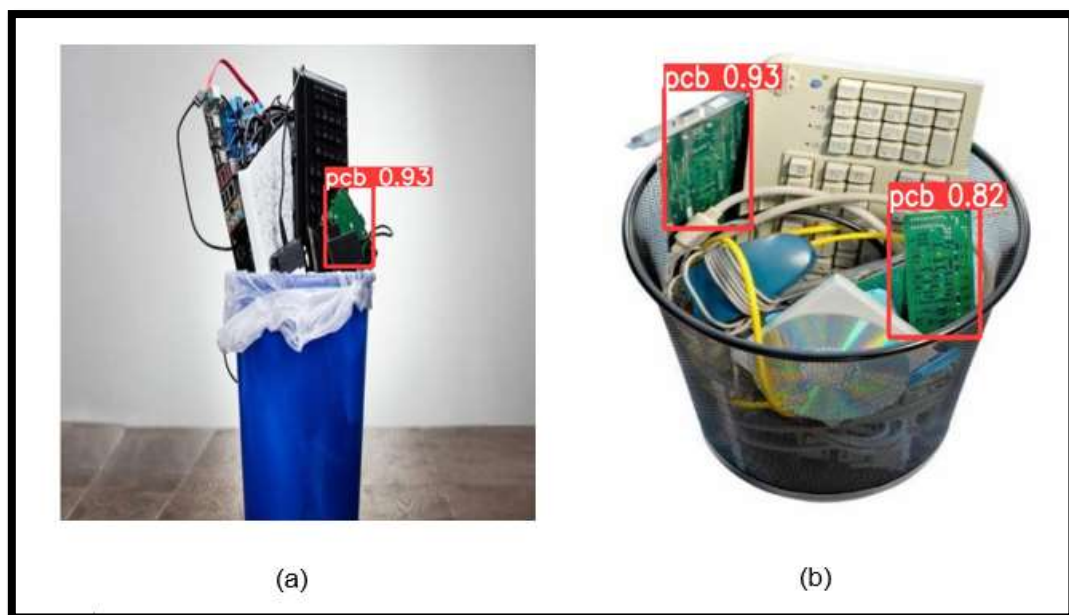
Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 4/4 [00:20<00:00, 5.04s/it]
all	438	435	0.956	0.972	0.98	0.849

**Figure 4-6: Training Process and the accuracy and performance of the YOLO model**

We conducted training on our dataset for 85 epochs, which took approximately 1.6 hours utilizing the T4 GPU provided by Google Colab. Our model consisted of 157 layers. We achieved an Overall Precision of 0.96 and an Overall Recall of 0.97, with an Accuracy of 98.5% and a Performance of 88.1%. By following these steps, we obtained the finalized model by selecting the best weights from one of the epochs that yielded the highest accuracy and precision

- **Training Visualization**

In this step we can visualize the model's performance on actual test images that it did not train on to see if the model has good generalization which is the capability of the model to adapt and perform accurately on new, unseen data after being trained on a subset [59], data. We used an external code that can obtain the address link of any picture on the internet web such as the images down below in figure 4-6:

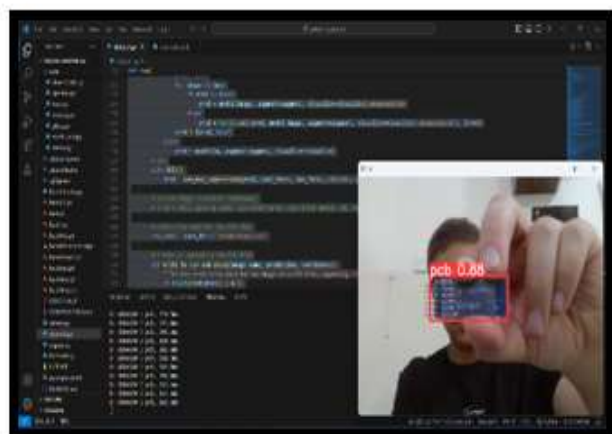


**Figure 4-7:Model performance on test images**

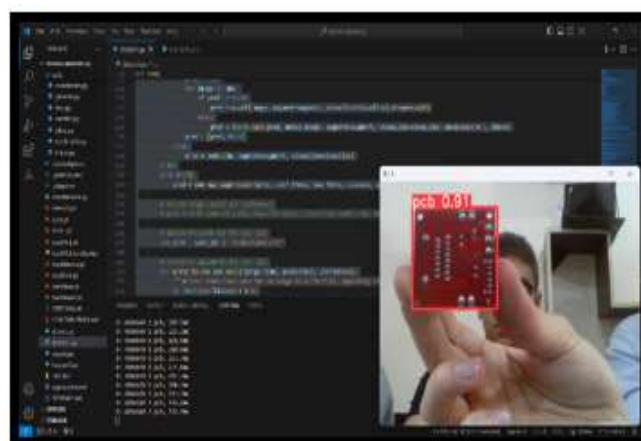
As it shows we got a Decent detection percentage on the PCBs inserted in environment and we can take this as evidence that the object detection model is ready to be downloaded and used in the actual robotic arm microcontroller and camera.

- **Real time Visualization**

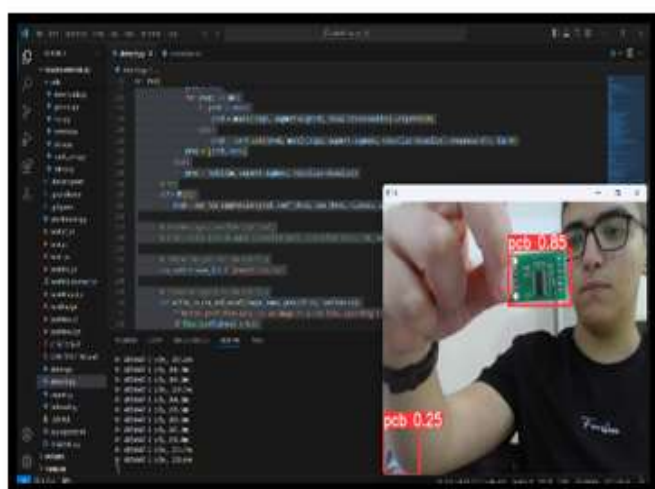
We successfully managed to connect a web camera with the VS code computer vision code and obtained a real-time data acquisition with the YOLO5 model connected to it and it gave bounding boxes to predict the PCBs used and classify it from other objects showed in the figure 4-7 below:



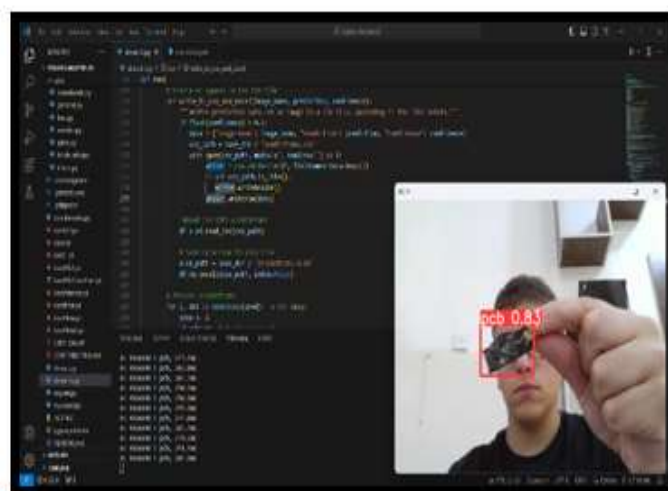
(a)



(b)



(c)

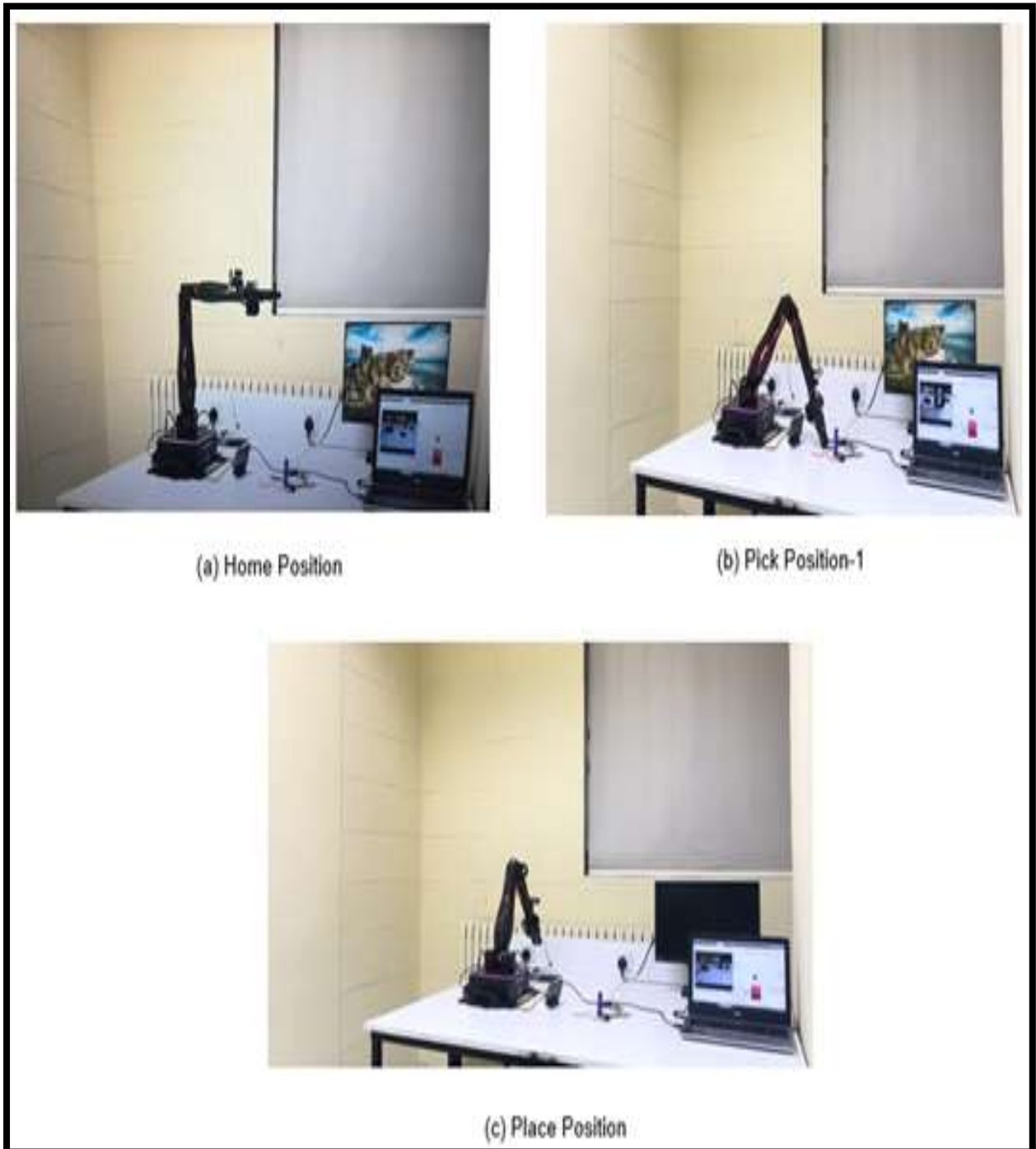


(d)

Figure 4-8:Real-time Visualization on real PCBs with confidence results

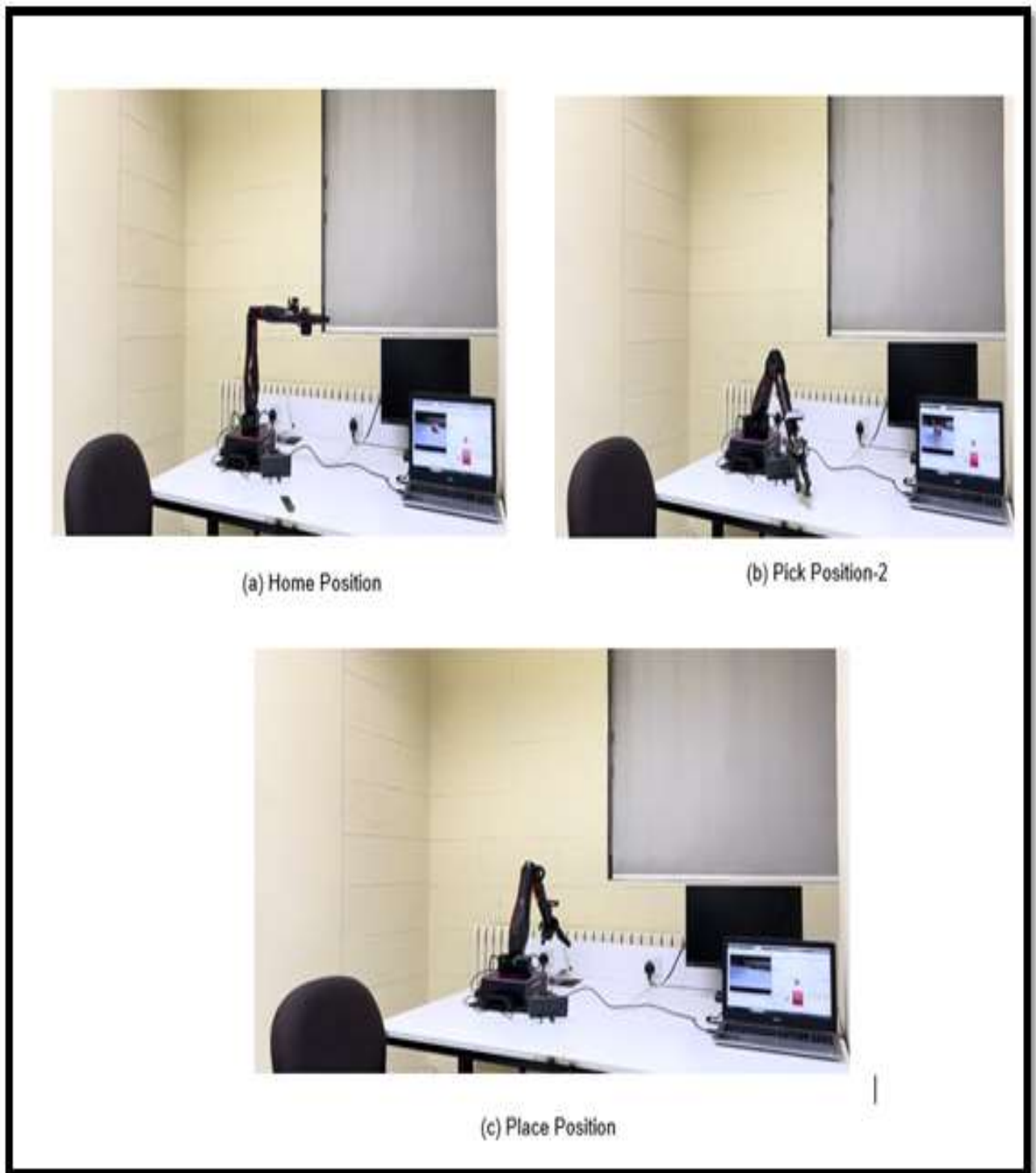
## ***4.2 Pick and place system results***

The Robotic arm motion process can be divided into 3 stages: starting at home position then forwarding to reach pick position of the PCB and then after picking it the robotic arm goes to the place position to drop the PCB in its slot, we were able to adjust the pick position of the gripper as we needed, we will show it in the next figures 4-9,10,11:

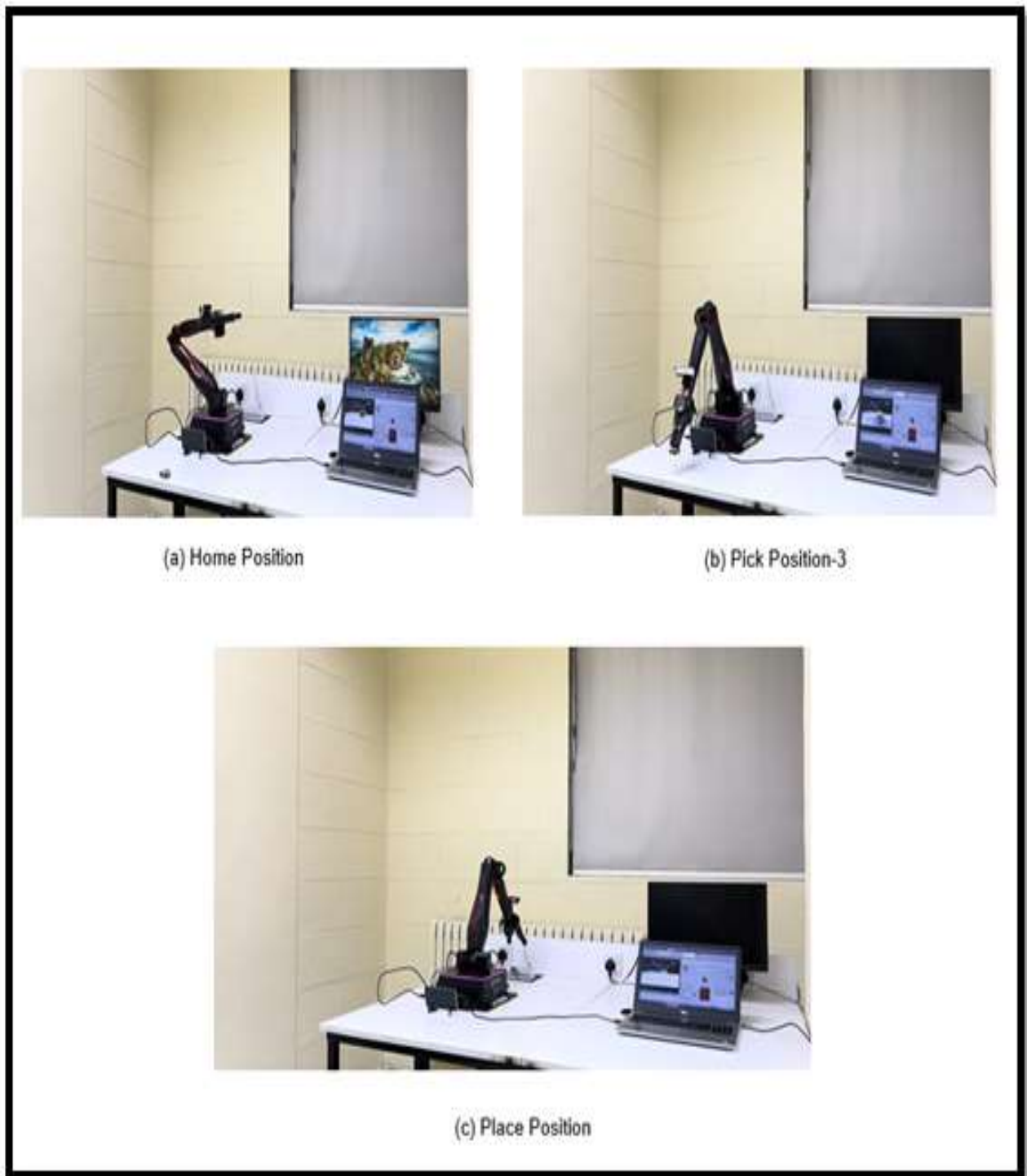


**Figure 4-9:Robotic arm motion in Pick position case 1**





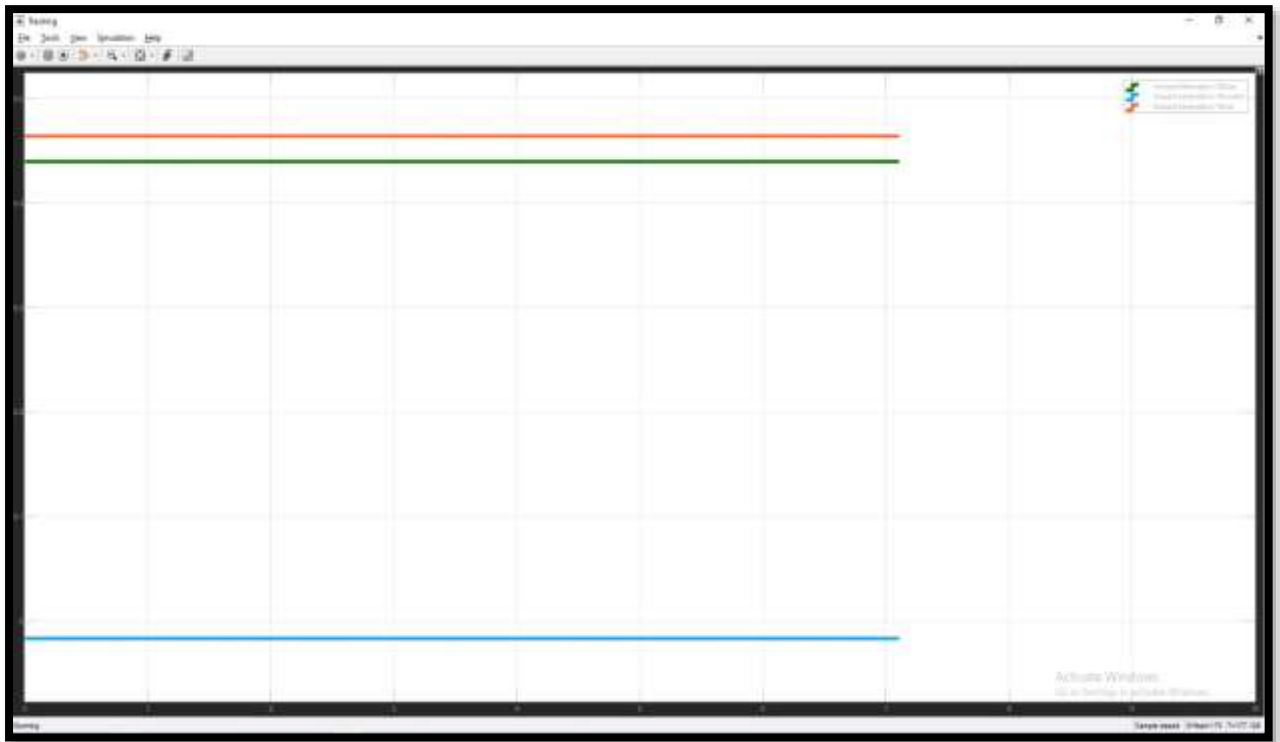
**Figure 4-10:Robotic arm motion in Pick position case 2**



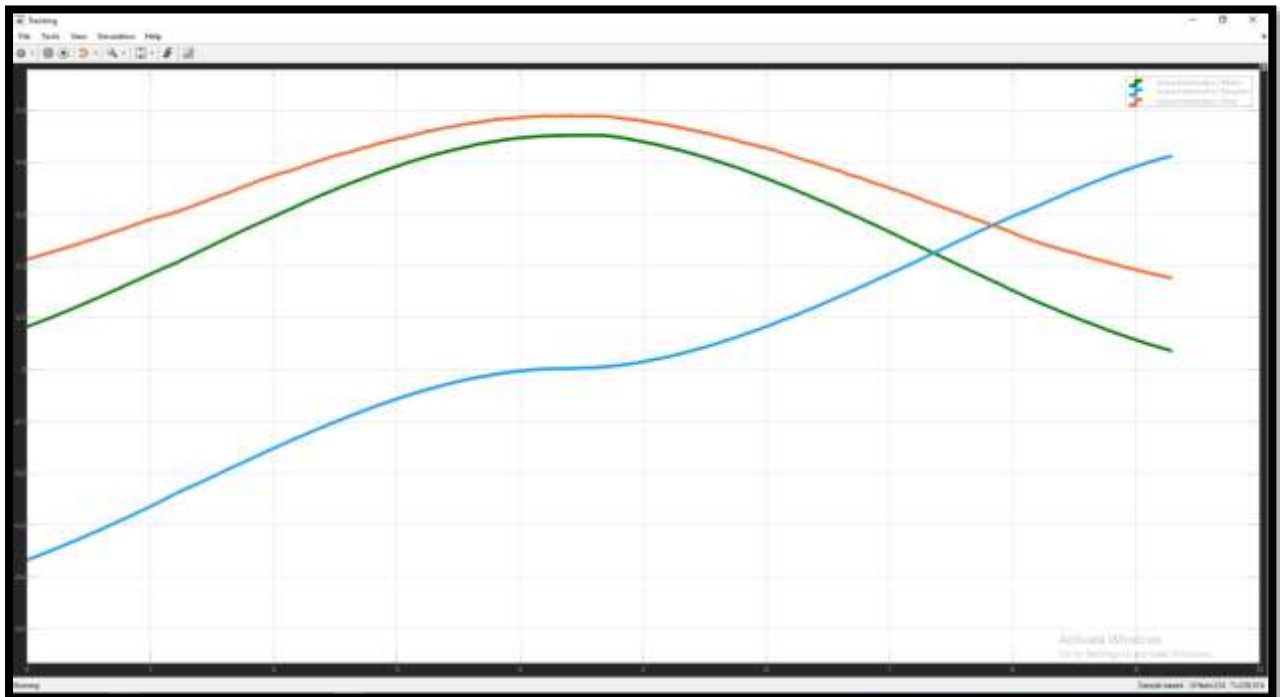
**Figure 4-11:Robotic arm motion in Pick position case 3**



After we saw the previous results of the motion in pick-place, we will show you the motion of the forward kinematics in three positions: initial position, current position, & final position as we will show in the following figures 4-12, 13: -



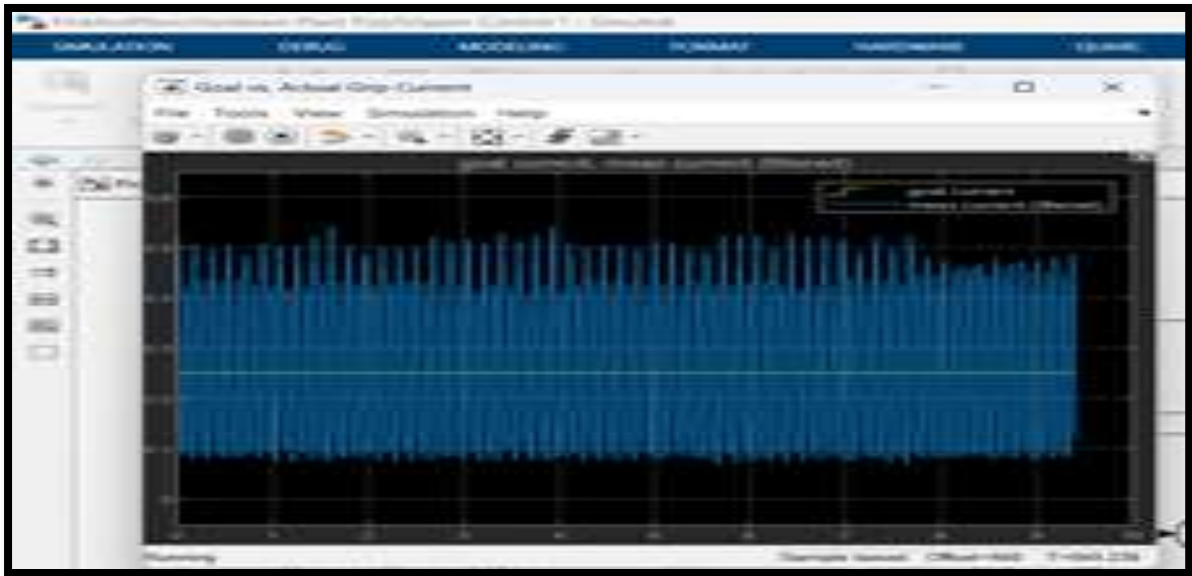
**Figure 4-12: Forward Kinematics plot when the QArm is in home position (Fixed)**



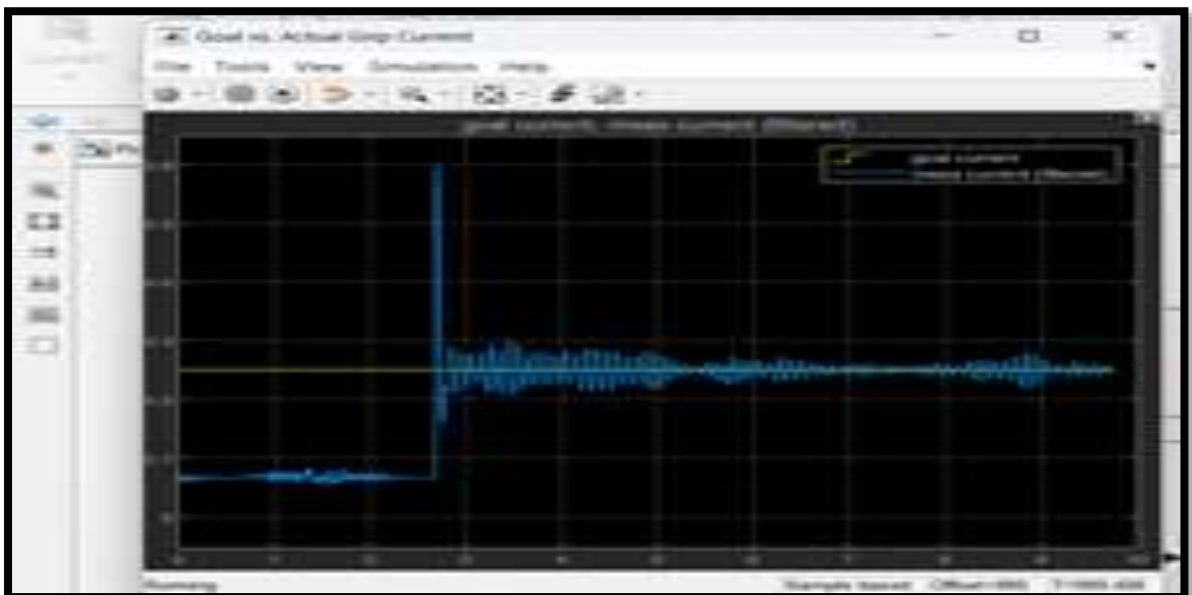
**Figure 4-13: Forward Kinematics plot when the QArm is in moving condition**

- **Gripper control performance**

As we mentioned previously, we controlled the gripper using a PI controller to control and adjust the amount of current flowing into the gripper which is directly proportional to the torque applied on the gripper. So, by adjusting the proportional gain we can control the grip strength of the end-effector and we plotted the signal of the current before and after adjusting the gain to obtain the optimal performance with minimum vibrations and instability, the figures 4-12 and 4-13 below shows the graphs of the current signals when holding the PCBs:



**Figure 4-14:Current signal before tuning the gain ( $K_p=12$ )**



**Figure 4-15:Current signal after tuning the gain ( $K_p=6$ )**

### ***4.3 System Limitations and Compliance with Design Constraints***

There were some limitations in our project as we will mention now in the two sections below: -

- **Software Limitations**

At the beginning of the project, we didn't have any access to the QUARC library on Simulink for technical problems in the robotic arm and it took a while for this problem to be solved, plus that the virtual machine for the Quanser company (Qlabs) required a special type of access which cannot be usually obtained by students.

the Quanser products are not open sourced which gave us hard time finding and collecting information about the QArm since it has a very small community using it.

Another software constrain we faced was the inability to obtain real time data from the camera into the Simulink blocks to compare confidence and classify PCBs so that the robotic arm can decide to operate or not.

The robotic arm had another program to operate instead of using Simulink which is ROS, unfortunately to learn and start using it will take us approximately about a whole year but in case of doing that we could have made it easier for us and got a better result.

- **Hardware Limitations**

We encounter an issue when operating the robotic arm due to a limitation on the z-axis of the end-effector. This is because the wooden table upon which the robotic arm is placed is larger than the QArm's actual base, restricting our ability to reach positions beneath the table, despite the QArm's capability to do so.

Implementing a real-time application using Simulink with Python necessitates a high-performance PC with advanced specifications, including abundant RAM size and swift processing capabilities. Regrettably, the connected PC does not meet these necessary requirements.

After conducting experiments, we observed that the placement of the depth camera attached to the robotic arm is suboptimal for object detection, especially for our project. This is primarily due to the short wires connecting the robotic arm to the camera. Additionally, the presence of a DAQ device positioned on top of the gripper adversely impacts our confidence and performance in detecting PCBs.

## **4.4 Solution Impact**

Implementing our system to detect and sort (PCBs) from electrical waste using industrial robotic arm in general can have significant impacts across various domains:

- **Global Impact:**

- Supply Chain Efficiency: Efficient sorting of PCBs from electrical waste can contribute to the recovery of valuable resources such as gold, silver, and copper, reducing the dependency on mining for these materials. [60]

- Global Trade: Standardizing the process of PCB sorting can simplify the global trade in electronic waste (e-waste) by ensuring that valuable components are salvaged and recycled efficiently, potentially reducing e-waste export to developing countries where recycling methods may be less environmentally friendly. [60]

- Technology Transfer: Implementing such advanced sorting technologies can facilitate technology transfer between developed and developing countries.

- **Economic Impact:**

- Resource Recovery: Recovering valuable metals from PCBs can create new revenue streams for recycling companies and incentivize the recycling of e-waste, making it more economically viable.

- Job Creation: The deployment of robotic arms for sorting PCBs may lead to the creation of jobs in robotics maintenance, programming, and supervision, although it may reduce manual sorting jobs. [61]

- Cost Savings: By automating the sorting process, companies can potentially reduce labour costs and increase operational efficiency, leading to cost savings in the long run. [61]

- **Environmental Impact:**

- Resource Conservation: Recovering and recycling PCBs reduces the need for extracting raw materials, thus conserving natural resources and reducing the environmental impact of mining activities. [62]

- Waste Reduction: Proper sorting and recycling of PCBs minimize the amount of hazardous electronic waste that ends up in landfills or is incinerated, thereby reducing pollution and associated environmental risks.

- Energy Savings: Recycling metals from PCBs typically requires less energy compared to primary extraction processes, leading to lower greenhouse gas emissions and energy consumption. [62]

- **Societal Impact:**

- Health and Safety: By diverting PCBs from informal recycling processes, which often involve hazardous practices such as open burning and acid leaching, the health and safety of informal recyclers and nearby communities can be improved. [63]

- Education and Awareness: Implementing advanced sorting technologies can raise awareness about the environmental and social impacts of e-waste and promote education on sustainable waste management practices. [63]

Overall, the implementation of automated systems for detecting and sorting PCBs from electrical waste can lead to positive outcomes at the global, economic, environmental, and societal levels by promoting resource efficiency, reducing pollution, creating economic opportunities, and improving waste management practices.

# Chapter 5 CONCLUSION & FUTURE WORK

## **5.1 Conclusion**

In conclusion, our integration of robotic arm technology with computer vision has emerged as a transformative solution for PCB waste management challenges. Despite initial setbacks, our system has successfully automated sorting processes, improving efficiency and accuracy while reducing environmental and health risks linked to informal recycling. The exceptional performance of our solution highlights its potential to revolutionize semiconductor manufacturing, offering a cost-effective and sustainable approach to electronic waste management. With continued refinement and scalability, this approach holds promise for broader adoption in global waste management systems, making substantial contributions to environmental preservation and public health.

## **5.2 Recommendations for Future Work**

Given more time, we would have opted to switch the end-effector type to a pneumatic suction mechanism instead of a gripping end-effector. This decision stems from the varying geometries of different PCBs. Additionally, we believe that integrating a conveyor belt would be optimal for this application, especially since we sort PCBs at a fixed location. Another enhancement we would consider is implementing a feature to obtain the position of the PCB from the camera, allowing for multiple point locations from which the PCB can be picked.

Regarding the software aspect, if time permitted, we would have chosen to utilize ROS (Robot Operating System) instead of Simulink. ROS is more tailored for robotics and position acquisition and tends to perform better in real-time applications. It offers smoother integration of real-time data acquisition with the motion of the robotic arm compared to Simulink.

## REFERENCES

- [1] G. L. E. Corporation, "Top Solutions to E-waste Problems," *Great Lakes Electronics Blog*, 2019.
- [2] M. & K. Shabaz, "A Novel Sorting Technique for Large-Scale Data," *Comput. Networks Commun*, 2019.
- [3] Sifat, "The evolution of sorting algorithms over the years," *bubble sort to ai-driven sort*, 2023.
- [4] E.-E. N.-B. T. C. O.-A. Rosca, "A Sorting Solution Based on an Industrial Robotic Arm with Image Processing," *Lecture notes in networks and systems*, 2022.
- [5] E. K. M. M. a. E. H. I. Karabegović, "The application of service robots for logistics in manufacturing processes," *Advances in Production Engineering & Management*, 2015.
- [6] H. & B. R. Salehi, "Emerging artificial intelligence methods in structural engineering," *Engineering Structures*, 2018.
- [7] M. & R. F. & P. G. & M. N. & M. M. Koskinopoulou, "Robotic Waste Sorting Technology: Toward a Vision-Based Categorization System for the Industrial Robotic Separation of Recyclable Waste," *IEEE Robotics & Automation Magazine*, vol. 28, pp. 50-60, 2021.
- [8] V. n. R. Kale, "OBJECT SORTING SYSTEM USING ROBOTIC ARM," 2022.
- [9] D. V. n. Kumar Reddy, "SORTING OF OBJECTS BASED ON COLOUR BY PICK AND PLACE ROBOTIC ARM AND WITH CONVEYOR BELT ARRANGEMENT," 2022.
- [10] C. D. V. n. Duy Anh Dang, "Development of Multi-Robotic Arm System for Sorting System Using Computer Vision," 2023.
- [11] A. Dziedzickis and Subačiūtė-Žemaitienė, "Advanced Applications of Industrial Robotics: New Trends and Possibilities," *New Trends and Possibilities. Appl. Sci*, 2022.
- [12] Z. Sun, L. Huang and R. Jia, "Coal and Gangue Separating Robot System Based on Computer Vision," *Sensors*, 2021.
- [13] H. A. K. A. R. A. Sampreeth, "Object Sorting Robot Using Image Processing," *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCESC*, 2018.
- [14] Quanser, "MODERN MANIPULATOR ARM FOR ROBOTICS COURSES AND RESEARCH," 15 Nov 2021.
- [15] Dynamixel, "Dynamixel XM540-W270-R Servo," *Génération Robots*, 2024.

- [16] "Depth camera D415," *Intel® RealSense™ Depth and Tracking Cameras.*, 2023.
- [17] F. V. HEIDI WIGHT, "EXTENDING YOUR REACH: QARM END-EFFECTOR DATA ACQUISITION DEVICE," *ROBOTICS & HAPTICS*, 2021.
- [18] MathWorks, "ROS Simulink Interaction - MATLAB & Simulink," 2023.
- [19] A. Ademovic, "An Introduction to Robot Operating System: The Ultimate Robot Application Framework," *Toptal Engineering Blog*, 2022.
- [20] javatpoint, "Advantages and Disadvantages of MATLAB Programming Language," *Introduction to MATLAB and Simulink*, 2011.
- [21] S. P. M. a. Y. L. G. Shashank, "Bridging ROS with MATLAB and Simulink: From Algorithms to Deployment," *Components of Autonomous Systems*, 2024.
- [22] GeeksforGeeks, "Introduction To PYTHON," *GeeksforGeeks Blog*, 2015.
- [23] C. BasuMallick, "C vs. C++: 12 Key Differences and Similarities," *Spiceworks*, 2023.
- [24] T. Team, "Python Advantages and Disadvantages - Step in the right direction - TechVidvan," *TechVidvan*, 2019.
- [25] s. adnanirshad, "Advantages and Disadvantages of C++," *geeksforgeeks*, 2023.
- [26] T. S. R. Mupparaju Sohan, "A Review on YOLOv8 and Its Advancements," *Data Intelligence and Cognitive Informatics Conference (ICDICI)*, 2023.
- [27] B. C. F. L. A. W. Mohammad Javad Shafiee, "Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video," *Journal of Real-Time Image Processing*, Volume 20, Article number 24, 2023.
- [28] C. C. R. J. a. X. Fei Gao, "Improved YOLOX for pedestrian detection in crowded scenes," *Journal of Real-Time Image Processing*, Volume 20, 2023.
- [29] shreya, "Regarding GPU for YOLOv8," *Issue #8109*, 2024.
- [30] A. A.I, "YOLOv8 vs. YOLOv5: Choosing the Best Object Detection Model," [www.augmentedstartups.com/blog](http://www.augmentedstartups.com/blog), 2023.
- [31] N. F. F. N Kumar, "YOLO-based light-weight deep learning models for insect detection system with field adaption," *Agriculture blog*, 2023.
- [32] C. conversion, "Robot Safety Resources," *Conversion Technology Inc*, 2012.
- [33] F. H. a. J. H. V. Murashov, "Working safely with robot workers: Recommendations for the new workplace," *Journal of Occupational and Environmental Hygiene*, 2016.



- [34] NQA, "What Is the ISO 27001 Standard?," *ISO 27001: Information Security Management Systems*, 2017.
- [35] G. Boesch, "Object Detection in 2021: The Definitive Guide," *viso.ai*, 2021.
- [36] C. F. Coombs, "SUMMARY OF KEY COMPONENT, MATERIAL, PROCESS, AND DESIGN STANDARDS," *Jr. Printed Circuits Handbook*, 2008.
- [37] L. G. A. PhD, "The practical guide for Object Detection with YOLOv5 algorithm," *Medium*, 2022.
- [38] roboflow, "Evaluating Roboflow: Do You Need a Proof of Concept?," *Roboflow Blog*, 2023.
- [39] TensorFlow, "How Roboflow enables thousands of developers to use computer vision with TensorFlow.js," *computer-vision-with-TensorFlow*, 2022.
- [40] A. A. L. G. a. A. S. K. A. Gupta, "Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues," *Array*, vol. 10, no. 100057, p. 100057, 2021.
- [41] Zegorax, "How to decrease false positives ?," *Issue #666*, 2024.
- [42] S. Khanna, "A Comprehensive Guide to Train-Test-Validation Split in 2023," *Analytics Vidhya*, 2023.
- [43] obviously, "The Difference Between Training Data vs. Test Data in Machine Learning," *obviously.ai blog*, 2021.
- [44] "Data Preprocessing in Machine Learning," *kaggle.com*, 2023.
- [45] I. Kelk, "Search and Rescue: Augmentation and Preprocessing on Drone-Based Water Rescue Images With YOLOv5," *W&B*, 2022.
- [46] A. A. Awan, "A Complete Guide to Data Augmentation," *Radar.ai blog*, 2022.
- [47] J. N., "How Flip Augmentation Improves Model Performance," *Roboflow Blog*, 2020.
- [48] J. N., "ntroducing Grayscale and Hue/Saturation Augmentations," *Roboflow Blog*, 2020.
- [49] B. Dwyer, "Advanced Augmentations in Roboflow," *Roboflow Blog*, 2020.
- [50] M. J. Garbade, "What Is Google Colab," *Education*, 2021.
- [51] S. J. a. C. J. S. Schmalz, "Using YOLOv5 Object Detection and a Raspberry Pi to Improve the Safety," *OpenRiver Blog*, 2022.
- [52] J. Redmon, "You only look once: Unified, real-time object detection.," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

- [53] Quanser, “What Each Block Reference Page Contains :: QUARC Targets Library,” *Quanser / Quarc blog*, 2024.
- [54] S. T. Center, “Quanser Lab Workstation Support from Simulink,” *Mathworks Hardware Support*, 2024.
- [55] glenn-jocher, “F1 confidence, Precision-Recall curve, Precision-Confidence curve, Recall Confidence curve and Confusion matrix,” *Issue #7307*, 2023.
- [56] J. S., “What is Mean Average Precision (mAP) in Object Detection?,” *Roboflow Blog*, 2020.
- [57] S. Overflow, “What is loss\_cls and loss\_bbox and why are they always zero in training,” *Stack Overflow Blog*, 2024.
- [58] D. Shah, “Mean Average Precision (mAP) Explained: Everything You Need to Know,” *www.v7labs.com*, 2022.
- [59] magnimind, “What Is Generalization In Machine Learning?,” *Magnimind Academy Blogs*, 2021.
- [60] E. A. O. a. H. Potgieter, “Discarded e-waste/printed circuit boards,” *a review of their recent methods of disassembly, sorting and environmental implications*, 2022.
- [61] R. S. S. L. Masoud Ahmadinia, “A Circular Economy Approach to PCB Recycling,” *Transforming E-Waste into Value*, 2023.
- [62] B. A. B. Saurabh P. Tembhare, “a review on environmental concerns, remediation and technological developments with a focus on printed circuit boards,” *E-waste recycling practices*., 2023.
- [63] C. S. K. L. D. C. W. H. & G. M. Chao Ning, “Waste Printed Circuit Board (PCB) Recycling Techniques,” *Printed Circuit Board (PCB) Recycling blog*, 2024.

# Appendices

- **Appendix A**

**Python (Detect in real-time) Code: -**

```
# YOLOv5 🚀 by Ultralytics, AGPL-3.0 license
"""
Run YOLOv5 detection inference on images, videos, directories, globs,
YouTube, webcam, streams, etc.

Usage - sources:
    $ python detect.py --weights yolov5s.pt --source
0                                     # webcam
                                     img.jpg
                                     # image
                                     vid.mp4
                                     # video
                                     screen
                                     # screenshot
                                     path/
                                     # directory
                                     list.txt
                                     # list of images
                                     list.streams
                                     # list of streams
                                     'path/*.jpg'
                                     # glob
                                     'https://youtu.be/LNw
ODJXcvt4' # YouTube
                                     'rtsp://example.com/m
edia.mp4' # RTSP, RTMP, HTTP stream

Usage - formats:
    $ python detect.py --weights yolov5s.pt                # PyTorch
                                yolov5s.torchscript        # TorchScript
                                yolov5s.onnx                # ONNX Runtime
or OpenCV DNN with --dnn
                                yolov5s_opencv_model       # OpenVINO
                                yolov5s.engine              # TensorRT
                                yolov5s.mlmodel              # CoreML
(macOS-only)
                                yolov5s_saved_model         # TensorFlow
SavedModel
                                yolov5s.pb                  # TensorFlow
GraphDef
```

```

                                yolov5s.tflite                # TensorFlow
Lite
                                yolov5s_edgetpu.tflite        # TensorFlow
Edge TPU
                                yolov5s_paddle_model         # PaddlePaddle
"""

import argparse
import csv
import os
import platform
import sys
from pathlib import Path
import pandas as pd
import openpyxl

import torch

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0] # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT)) # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd())) # relative

from ultralytics.utils.plotting import Annotator, colors, save_one_box

from models.common import DetectMultiBackend
from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages,
LoadScreenshots, LoadStreams
from utils.general import (
    LOGGER,
    Profile,
    check_file,
    check_img_size,
    check_imshow,
    check_requirements,
    colorstr,
    cv2,
    increment_path,
    non_max_suppression,
    print_args,
    scale_boxes,
    strip_optimizer,
    xyxy2xywh,
)
from utils.torch_utils import select_device, smart_inference_mode

```

```

@smart_inference_mode()
def run(
    weights=ROOT / "yolov5s.pt", # model path or triton URL
    source=ROOT / "data/images", # file/dir/URL/glob/screen/0(webcam)
    data=ROOT / "data/coco128.yaml", # dataset.yaml path
    imgsz=(640, 640), # inference size (height, width)
    conf_thres=0.50, # confidence threshold 0.25
    iou_thres=0.45, # NMS IOU threshold 0.45
    max_det=1000, # maximum detections per image #1000
    device="", # cuda device, i.e. 0 or 0,1,2,3 or cpu
    view_img=False, # show results
    save_txt=False, # save results to *.txt
    save_csv=False, # save results in CSV format
    save_conf=False, # save confidences in --save-txt labels
    save_crop=False, # save cropped prediction boxes
    nosave=False, # do not save images/videos
    classes=None, # filter by class: --class 0, or --class 0 2 3
    agnostic_nms=False, # class-agnostic NMS
    augment=False, # augmented inference
    visualize=False, # visualize features
    update=False, # update all models
    project=ROOT / "runs / detect", # save results to project/name
    name="exp", # save results to project/name
    exist_ok=False, # existing project/name ok, do not increment
    line_thickness=3, # bounding box thickness (pixels)
    hide_labels=False, # hide labels
    hide_conf=False, # hide confidences
    half=False, # use FP16 half-precision inference
    dnn=False, # use OpenCV DNN for ONNX inference
    vid_stride=1, # video frame-rate stride
):
    source = str(source)
    save_img = not nosave and not source.endswith(".txt") # save
inference images
    is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
    is_url = source.lower().startswith(("rtsp://", "rtmp://", "http://",
"https://"))
    webcam = source.isnumeric() or source.endswith(".streams") or (is_url
and not is_file)
    screenshot = source.lower().startswith("screen")
    if is_url and is_file:
        source = check_file(source) # download

    # Directories
    save_dir = Path(project) / name # increment run
    (save_dir if save_txt else save_dir).mkdir(parents=True,
exist_ok=True) # make dir #### (save_dir / "labels" if save_txt else
save_dir).mkdir(parents=True, exist_ok=True)

```

```

# Load model
device = select_device(device)
model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data,
fp16=half)
stride, names, pt = model.stride, model.names, model.pt
imgsz = check_img_size(imgsz, s=stride) # check image size

# Dataloader
bs = 1 # batch_size
if webcam:
    view_img = check_imshow(warn=True)
    dataset = LoadStreams(source, img_size=imgsz, stride=stride,
auto=pt, vid_stride=vid_stride)
    bs = len(dataset)
elif screenshot:
    dataset = LoadScreenshots(source, img_size=imgsz, stride=stride,
auto=pt)
else:
    dataset = LoadImages(source, img_size=imgsz, stride=stride,
auto=pt, vid_stride=vid_stride)
    vid_path, vid_writer = [None] * bs, [None] * bs

# Run inference
model.warmup(imgsz=(1 if pt or model.triton else bs, 3, *imgsz)) #
warmup
seen, windows, dt = 0, [], (Profile(device=device),
Profile(device=device), Profile(device=device))
for path, im, im0s, vid_cap, s in dataset:
    with dt[0]:
        im = torch.from_numpy(im).to(model.device)
        im = im.half() if model.fp16 else im.float() # uint8 to
fp16/32
        im /= 255 # 0 - 255 to 0.0 - 1.0
        if len(im.shape) == 3:
            im = im[None] # expand for batch dim
        if model.xml and im.shape[0] > 1:
            ims = torch.chunk(im, im.shape[0], 0)

    # Inference
    with dt[1]:
        visualize = save_dir / Path(path).stem if visualize else
False ### increment_path(save_dir / Path(path).stem, mkdir=True) if
visualize else False
        if model.xml and im.shape[0] > 1:
            pred = None
            for image in ims:
                if pred is None:

```

```

        pred = model(image, augment=augment,
visualize=visualize).unsqueeze(0)
    else:
        pred = torch.cat((pred, model(image,
augment=augment, visualize=visualize).unsqueeze(0)), dim=0)
        pred = [pred, None]
    else:
        pred = model(im, augment=augment, visualize=visualize)
# NMS
with dt[2]:
    pred = non_max_suppression(pred, conf_thres, iou_thres,
classes, agnostic_nms, max_det=max_det)

# Second-stage classifier (optional)
# pred = utils.general.apply_classifier(pred, classifier_model,
im, im0s)

# Define the path for the CSV file
csv_path = save_dir / "predictions.csv"

# Create or append to the CSV file
def write_to_csv_and_excel(image_name, prediction, confidence):
    """Writes prediction data for an image to a CSV file,
appending if the file exists."""
    if float(confidence) > 0.5:
        data = {"Image Name": image_name, "Prediction": prediction,
"Confidence": confidence}
        csv_path = save_dir / "predictions.csv"
        with open(csv_path, mode="w", newline="") as f:
            writer = csv.DictWriter(f, fieldnames=data.keys())
            if not csv_path.is_file():
                writer.writeheader()
            writer.writerow(data)

#Read CSV into a DataFrame
df = pd.read_csv(csv_path)

# Save DataFrame to xlsx file
xlsx_path = save_dir / "predictions.xlsx"
df.to_excel(xlsx_path, index=False)

# Process predictions
for i, det in enumerate(pred): # per image
    seen += 1
    if webcam: # batch_size >= 1
        p, im0, frame = path[i], im0s[i].copy(), dataset.count
        s += f"{i}: "
    else:

```

```

        p, im0, frame = path, im0s.copy(), getattr(dataset,
"frame", 0)

        p = Path(p)  # to Path
        save_path = str(save_dir / p.name)  # im.jpg
        txt_path = str(save_dir / p.stem)  # im.txt ### txt_path =
str(save_dir / "labels" / p.stem) + (" " if dataset.mode == "image" else
f"_{frame}")
        s += "%gx%g " % im.shape[2:]  # print string
        gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization
gain whwh
        imc = im0.copy() if save_crop else im0  # for save_crop
        annotator = Annotator(im0, line_width=line_thickness,
example=str(names))
        if len(det):
            # Rescale boxes from img_size to im0 size
            det[:, :4] = scale_boxes(im.shape[2:], det[:, :4],
im0.shape).round()

            # Print results
            for c in det[:, 5].unique():
                n = (det[:, 5] == c).sum()  # detections per class
                s += f"{n} {names[int(c)]}'s' * (n > 1)}, "  # add to
string

            # Write results
            for *xyxy, conf, cls in reversed(det):
                c = int(cls)  # integer class
                label = names[c] if hide_conf else f"{names[c]}"
                confidence = float(conf)
                confidence_str = f"{confidence:.2f}"

                if save_csv:
                    write_to_csv_and_excel(p.name, label,
confidence_str)

                if save_txt:  # Write to file
                    xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) /
gn).view(-1).tolist()  # normalized xywh
                    line = (cls, *xywh, conf) if save_conf else (cls,
*xywh)  # label format
                    with open(f"{txt_path}.txt", "w") as
f:  ##### "a"
                        f.write((" %g " * len(line)).rstrip() % line +
"\n")

                if save_img or save_crop or view_img:  # Add bbox to
image

```



```

        c = int(cls) # integer class
        label = None if hide_labels else (names[c] if
hide_conf else f"{names[c]} {conf:.2f}")
        annotator.box_label(xyxy, label, color=colors(c,
True))

        if save_crop:
            save_one_box(xyxy, imc, file=save_dir / "crops" /
names[c] / f"{p.stem}.jpg", BGR=True)

    # Stream results
    im0 = annotator.result()
    if view_img:
        if platform.system() == "Linux" and p not in windows:
            windows.append(p)
            cv2.namedWindow(str(p), cv2.WINDOW_NORMAL |
cv2.WINDOW_KEEPRATIO) # allow window resize (Linux)
            cv2.resizeWindow(str(p), im0.shape[1], im0.shape[0])
            cv2.imshow(str(p), im0)
            cv2.waitKey(1) # 1 millisecond

    # Save results (image with detections)
    if save_img:
        if dataset.mode == "image":
            cv2.imwrite(save_path, im0)
        else: # 'video' or 'stream'
            if vid_path[i] != save_path: # new video
                vid_path[i] = save_path
                if isinstance(vid_writer[i], cv2.VideoWriter):
                    vid_writer[i].release() # release previous
video writer
            if vid_cap: # video
                fps = vid_cap.get(cv2.CAP_PROP_FPS)
                w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                h =
int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
            else: # stream
                fps, w, h = 30, im0.shape[1], im0.shape[0]
                save_path =
str(Path(save_path).with_suffix(".mp4")) # force *.mp4 suffix on results
videos
            vid_writer[i] = cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc(*"mp4v"), fps, (w, h))
            vid_writer[i].write(im0)

    # Print time (inference-only)
    LOGGER.info(f"{s}{' ' if len(det) else '(no detections),'}{dt[1].dt * 1E3:.1f}ms")

```

```

    # Print results
    t = tuple(x.t / seen * 1e3 for x in dt) # speeds per image
    LOGGER.info(f"Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS
per image at shape {(1, 3, *imgsz)}" % t)
    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to
{save_dir / 'labels'}" if save_txt else ""
        LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}{s}")
    if update:
        strip_optimizer(weights[0]) # update model (to fix
SourceChangeWarning)

def parse_opt():
    """Parses command-line arguments for YOLOv5 detection, setting
inference options and model configurations."""
    parser = argparse.ArgumentParser()
    parser.add_argument("--weights", nargs="+", type=str, default=ROOT /
"yolov5s.pt", help="model path or triton URL")
    parser.add_argument("--source", type=str, default=ROOT /
"data/images", help="file/dir/URL/glob/screen/0(webcam)")
    parser.add_argument("--data", type=str, default=ROOT /
"data/coco128.yaml", help="(optional) dataset.yaml path")
    parser.add_argument("--imgsz", "--img", "--img-size", nargs="+",
type=int, default=[640], help="inference size h,w")
    parser.add_argument("--conf-thres", type=float, default=0.25,
help="confidence threshold")
    parser.add_argument("--iou-thres", type=float, default=0.45, help="NMS
IoU threshold")
    parser.add_argument("--max-det", type=int, default=1000, help="maximum
detections per image")
    parser.add_argument("--device", default="", help="cuda device, i.e. 0
or 0,1,2,3 or cpu")
    parser.add_argument("--view-img", action="store_true", help="show
results")
    parser.add_argument("--save-txt", action="store_true", help="save
results to *.txt")
    parser.add_argument("--save-csv", action="store_true", help="save
results in CSV format")
    parser.add_argument("--save-conf", action="store_true", help="save
confidences in --save-txt labels")
    parser.add_argument("--save-crop", action="store_true", help="save
cropped prediction boxes")
    parser.add_argument("--nosave", action="store_true", help="do not save
images/videos")
    parser.add_argument("--classes", nargs="+", type=int, help="filter by
class: --classes 0, or --classes 0 2 3")

```

```

    parser.add_argument("--agnostic-nms", action="store_true",
help="class-agnostic NMS")
    parser.add_argument("--augment", action="store_true", help="augmented
inference")
    parser.add_argument("--visualize", action="store_true",
help="visualize features")
    parser.add_argument("--update", action="store_true", help="update all
models")
    parser.add_argument("--project", default=ROOT / "runs/detect",
help="save results to project/name")
    parser.add_argument("--name", default="exp", help="save results to
project/name")
    parser.add_argument("--exist-ok", action="store_true", help="existing
project/name ok, do not increment")
    parser.add_argument("--line-thickness", default=3, type=int,
help="bounding box thickness (pixels)")
    parser.add_argument("--hide-labels", default=False,
action="store_true", help="hide labels")
    parser.add_argument("--hide-conf", default=False, action="store_true",
help="hide confidences")
    parser.add_argument("--half", action="store_true", help="use FP16
half-precision inference")
    parser.add_argument("--dnn", action="store_true", help="use OpenCV DNN
for ONNX inference")
    parser.add_argument("--vid-stride", type=int, default=1, help="video
frame-rate stride")
    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1 # expand
    print_args(vars(opt))
    return opt

def main(opt):
    """Executes YOLOv5 model inference with given options, checking
requirements before running the model."""
    check_requirements(ROOT / "requirements.txt", exclude=("tensorboard",
"thop"))
    run(**vars(opt))
    run(weights=opt.weights, source=opt.source, data=opt.data,
imgsz=opt.imgsz, conf_thres=opt.conf_thres,
        iou_thres=opt.iou_thres, max_det=opt.max_det, device=opt.device,
view_img=opt.view_img, save_txt=opt.save_txt,
        save_csv=opt.save_csv, save_conf=opt.save_conf,
save_crop=opt.save_crop, nosave=opt.nosave, classes=opt.classes,
        agnostic_nms=opt.agnostic_nms, augment=opt.augment,
visualize=opt.visualize, update=opt.update,
        project=opt.project, name=opt.name, exist_ok=opt.exist_ok,
line_thickness=opt.line_thickness,

```

```
        hide_labels=opt.hide_labels, hide_conf=opt.hide_conf,
half=opt.half, dnn=opt.dnn, vid_stride=opt.vid_stride)

if __name__ == "__main__":
    opt = parse_opt()
    main(opt)
```

- **Appendix B**

- Python (Training) Code: -**

```
#clone YOLOv5 and
!git clone https://github.com/ultralytics/yolov5 # clone repo
%cd yolov5
%pip install -qr requirements.txt # install dependencies
%pip install -q roboflow
import torch
import os
from IPython.display import Image, clear_output # to display images
print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name} if
torch.cuda.is_available() else 'CPU')")
# set up environment
os.environ["DATASET_DIRECTORY"] = "/content/datasets"
!pip install roboflow
from roboflow import Roboflow
rf = Roboflow(api_key="NnJwpN7nRZabGRHmdmp8")
project = rf.workspace("pcbgraduationproject").project("confusion-matrix-fixx")
version = project.version(3)
dataset = version.download("yolov5")
!python train.py --img 640 --batch 8 --epochs 400 --data {dataset.location}/data.yaml --weights
yolov5s.pt --cache
# Start tensorboard
# Launch after you have started training
# logs save in the folder "runs"
%load_ext tensorboard
%tensorboard --logdir runs
!python detect.py --weights runs/train/exp/weights/best.pt --img 640 --conf 0.1 --source
{dataset.location}/test/images
#display inference on ALL test images
import glob
from IPython.display import Image, display
for imageName in glob.glob('/content/yolov5/runs/detect/exp/*.jpg'): #assuming JPG
    display(Image(filename=imageName))
    print("\n")

#export your model's weights for future use
```

from google.colab import files

files.download('./runs/train/exp/weights/best.pt')

- **Appendix C**

**Testing on a picture obtained by image address link from any source on the web:**

```
!pip install yolov5
# URL of the image
image_url = "https://commons.bcit.ca/news/files/2019/05/GettyImages-1058042428.jpg"

# Download the image
response = requests.get("https://commons.bcit.ca/news/files/2019/05/GettyImages-1058042428.jpg")
image = Image.open(BytesIO(response.content))
# Resize the image if necessary
image = image.resize((640, 640))
# Load your custom YOLOv5 model from the best.pt file
model = torch.hub.load('ultralytics/yolov5', 'custom', path='./content/best.pt')
# Set the model to evaluation mode
model.eval()
# Perform inference
results = model(image)
# Display the results
results.show()
```

- **Appendix D**

**Confusion matrix generation code:**

```
!pip install -q tensorflow-gpu==2.0.0-beta1
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import seaborn as sns
import pandas as pd
logdir='log'

(train_images, train_labels), (test_images, test_labels) = datasets.mnist.load_data()

train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))
train_images, test_images = train_images / 255.0, test_images / 255.0
classes=[0,1,2,3,4,5,6,7,8,9]
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
```

```
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1)
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x=train_images,
        y=train_labels,
        epochs=5,
        validation_data=(test_images, test_labels))

y_true=test_labels
y_pred=model.predict_classes(test_images)
classes=[0,1,2,3,4,5,6,7,8,9]
con_mat = tf.math.confusion_matrix(labels=y_true, predictions=y_pred).numpy()
con_mat_norm = np.around(con_mat.astype('float') / con_mat.sum(axis=1)[:, np.newaxis], decimals=2)
con_mat_df = pd.DataFrame(con_mat_norm,
                        index = classes,
                        columns = classes)

figure = plt.figure(figsize=(8, 8))
sns.heatmap(con_mat_df, annot=True,cmap=plt.cm.Blues)
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')

plt.show()
```

- **Appendix E**

- Inverse Kinematics (MATLAB / Simulink) Code: -**

```
function phi_optimal = qarmInverseKinematics(p, gamma, phi_prev)
```

```
%% QUANSER_ARM_IPK
% v 4.0 - 4th June 2020
```

```
% REFERENCE:
% Chapter 4. Inverse Kinematics
% Robot Dynamics and Control
% Spong, Vidyasagar
% 1989
```

```
% INPUTS:
% p      : end-effector position vector expressed in base frame {0}
% gamma  : wrist rotation angle gamma
```

```
% OUTPUTS:
% phi_optimal : Best solution depending on phi_prev
```

```
% phi      : All four Inverse Kinematics solutions as a 4x4 matrix.
%          Each col is a solution.

%% Manipulator parameters:
L1 = 0.1400;
L2 = 0.3500;
L3 = 0.0500;
L4 = 0.2500;
L5 = 0.1500;
beta = atan(L3/L2);

%% Alternate parameters
lambda1 = L1;
lambda2 = sqrt(L2^2 + L3^2);
lambda3 = L4 + L5;

%% Initialization
theta = zeros(4, 4);
phi = zeros(4, 4);

%% Joint angle 2 and 3:
% Equations:
% lambda2 cos(theta2) + (-lambda3) sin(theta2 + theta3) = sqrt(x^2 + y^2)
% A cos( 2 ) + C sin( 2 + 3 ) = D

% lambda2 sin(theta2) - (-lambda3) cos(theta2 + theta3) = lambda1 - z
% A sin( 2 ) - C cos( 2 + 3 ) = H

% Solution:
A = lambda2;
C = -lambda3;
H = lambda1 - p(3);
D1 = -sqrt(p(1)^2 + p(2)^2);
D2 = sqrt(p(1)^2 + p(2)^2);
F = (D1^2 + H^2 - A^2 - C^2)/(2*A);

theta(3,1) = 2*atan2( C + sqrt(C^2 - F^2) , F );
theta(3,2) = 2*atan2( C - sqrt(C^2 - F^2) , F );
theta(3,3) = 2*atan2( C + sqrt(C^2 - F^2) , F );
theta(3,4) = 2*atan2( C - sqrt(C^2 - F^2) , F );

M = A + C*sin(theta(3,1));
N = -C*cos(theta(3,1));
cos_term = (D1*M + H*N)/(M^2 + N^2);
sin_term = (H - N*cos_term)/(M);
theta(2,1) = atan2(sin_term, cos_term);

M = A + C*sin(theta(3,2));
N = -C*cos(theta(3,2));
cos_term = (D1*M + H*N)/(M^2 + N^2);
sin_term = (H - N*cos_term)/(M);
theta(2,2) = atan2(sin_term, cos_term);

M = A + C*sin(theta(3,3));
N = -C*cos(theta(3,3));
cos_term = (D2*M + H*N)/(M^2 + N^2);
sin_term = (H - N*cos_term)/(M);
theta(2,3) = atan2(sin_term, cos_term);

M = A + C*sin(theta(3,4));
N = -C*cos(theta(3,4));
cos_term = (D2*M + H*N)/(M^2 + N^2);
sin_term = (H - N*cos_term)/(M);
theta(2,4) = atan2(sin_term, cos_term);
```

%% Joint angle 1:

% Equation:

%  $y / x = \tan(\theta_1)$ ;

% Solution

$\theta_{1,1} = \text{atan2}(p(2)/(\lambda_2 \cos(\theta_{2,1}) - \lambda_3 \sin(\theta_{2,1} + \theta_{3,1})), p(1)/(\lambda_2 \cos(\theta_{2,1}) - \lambda_3 \sin(\theta_{2,1} + \theta_{3,1})))$ ;

$\theta_{1,2} = \text{atan2}(p(2)/(\lambda_2 \cos(\theta_{2,2}) - \lambda_3 \sin(\theta_{2,2} + \theta_{3,2})), p(1)/(\lambda_2 \cos(\theta_{2,2}) - \lambda_3 \sin(\theta_{2,2} + \theta_{3,2})))$ ;

$\theta_{1,3} = \text{atan2}(p(2)/(\lambda_2 \cos(\theta_{2,3}) - \lambda_3 \sin(\theta_{2,3} + \theta_{3,3})), p(1)/(\lambda_2 \cos(\theta_{2,3}) - \lambda_3 \sin(\theta_{2,3} + \theta_{3,3})))$ ;

$\theta_{1,4} = \text{atan2}(p(2)/(\lambda_2 \cos(\theta_{2,4}) - \lambda_3 \sin(\theta_{2,4} + \theta_{3,4})), p(1)/(\lambda_2 \cos(\theta_{2,4}) - \lambda_3 \sin(\theta_{2,4} + \theta_{3,4})))$ ;

%% Remap theta back to phi

%  $\phi(1) = \theta(1)$ ;

%  $\phi(2) = \theta(2) - \beta + \pi/2$ ;

%  $\phi(3) = \theta(3) + \beta$ ;

%  $\phi(4) = \theta(4)$ ;

$\phi(1,:) = \theta(1,:)$ ;

$\phi(2,:) = \theta(2,:) - \beta + \pi/2$ ;

$\phi(3,:) = \theta(3,:) + \beta$ ;

$\phi(4,:) = \gamma$ ;

$\phi = \text{mod}(\phi + \pi, 2\pi) - \pi$ ;

$\phi_{\text{optimal}} = \phi(:,1)$ ;

if (norm( $\phi_{\text{optimal}} - \phi_{\text{prev}}$ ) > norm( $\phi(:,2) - \phi_{\text{prev}}$ )) || (check\_joint\_limits( $\phi_{\text{optimal}}$ ))

$\phi_{\text{optimal}} = \phi(:,2)$ ;

end

if (norm( $\phi_{\text{optimal}} - \phi_{\text{prev}}$ ) > norm( $\phi(:,3) - \phi_{\text{prev}}$ )) || (check\_joint\_limits( $\phi_{\text{optimal}}$ ))

$\phi_{\text{optimal}} = \phi(:,3)$ ;

end

if (norm( $\phi_{\text{optimal}} - \phi_{\text{prev}}$ ) > norm( $\phi(:,4) - \phi_{\text{prev}}$ )) || (check\_joint\_limits( $\phi_{\text{optimal}}$ ))

$\phi_{\text{optimal}} = \phi(:,4)$ ;

end

if (check\_joint\_limits( $\phi_{\text{optimal}}$ ))

$\phi_{\text{optimal}} = [0;0;0;0]$ ;

end

end

function flag = check\_joint\_limits(phi)

flag = 0;

if  $\phi(1) > 170\pi/180$  ||  $\phi(1) < -170\pi/180$  ...

||  $\phi(2) > 80\pi/180$  ||  $\phi(2) < -80\pi/180$  ...

||  $\phi(3) > 75\pi/180$  ||  $\phi(3) < -95\pi/180$  ...

||  $\phi(4) > 160\pi/180$  ||  $\phi(4) < -160\pi/180$

flag = 1;

end

end



- **Appendix F**

**Forward Kinematics (MATLAB / Simulink) Code: -**

```
function [p4, R04] = qarmForwardKinematics(phi)

%% QUANSER_ARM_FPK
% v 4.0 - 4th June 2020

% REFERENCE:
% Chapter 3. Forward Kinematics
% Robot Dynamics and Control
% Spong, Vidyasagar
% 1989

% INPUTS:
% phi : Alternate joint angles vector 4 x 1

% OUTPUTS:
% p4 : End-effector frame {4} position vector expressed in base frame {0}
% R04 : rotation matrix from end-effector frame {4} to base frame {0}

%% Manipulator parameters:
L1 = 0.1400;
L2 = 0.3500;
L3 = 0.0500;
L4 = 0.2500;
L5 = 0.1500;

%% Alternate parameters
l1 = L1;
l2 = sqrt(L2^2 + L3^2);
l3 = L4 + L5;
beta = atan(L3/L2);

%% From phi space to theta space
theta = phi;
theta(1) = phi(1);
theta(2) = phi(2) + beta - pi/2;
theta(3) = phi(3) - beta;
theta(4) = phi(4);

%% Transformation matrices for all frames:

% T{i-1}{i} = quanser_arm_DH( a, alpha, d, theta );
T01 = quanser_arm_DH( 0, -pi/2, l1, theta(1) );
T12 = quanser_arm_DH( l2, 0, 0, theta(2) );
T23 = quanser_arm_DH( 0, -pi/2, 0, theta(3) );
T34 = quanser_arm_DH( 0, 0, l3, theta(4) );

T02 = T01*T12;
T03 = T02*T23;
T04 = T03*T34;

%% Position of end-effector Transformation

% Extract the Position vector
% p1 = T01(1:3,4);
% p2 = T02(1:3,4);
% p3 = T03(1:3,4);
p4 = T04(1:3,4);

% Extract the Rotation matrix
% R01 = T01(1:3,1:3);
% R02 = T02(1:3,1:3);
```

```
% R03 = T03(1:3,1:3);
R04 = T04(1:3,1:3);

end

function T = quanser_arm_DH(a,alpha,d,theta)

%% QUANSER_ARM_DH
% v 1.0 - 26th March 2019

% REFERENCE:
% Chapter 3. Forward and Inverse Kinematics
% Robot Modeling and Control
% Spong, Hutchinson, Vidyasagar
% 2006

% INPUTS:
% a : translation : along : x_{i} : from : z_{i-1} : to : z_{i}
% alpha : rotation : about : x_{i} : from : z_{i-1} : to : z_{i}
% d : translation : along : z_{i-1} : from : x_{i-1} : to : x_{i}
% theta : rotation : about : z_{i-1} : from : x_{i-1} : to : x_{i}
% NOTE: Standard DH Parameters

% OUTPUTS:
% T : transformation : from : {i} : to : {i-1}

%%
% Rotation Transformation about z axis by theta
T_R_z = [cos(theta) -sin(theta) 0 0;
         sin(theta) cos(theta) 0 0;
         0 0 1 0;
         0 0 0 1];

% Translation Transformation along z axis by d
T_T_z = [1 0 0 0;
         0 1 0 0;
         0 0 1 d;
         0 0 0 1];

% Translation Transformation along x axis by a
T_T_x = [1 0 0 a;
         0 1 0 0;
         0 0 1 0;
         0 0 0 1];

% Rotation Transformation about x axis by alpha
T_R_x = [1 0 0 0;
         0 cos(alpha) -sin(alpha) 0;
         0 sin(alpha) cos(alpha) 0;
         0 0 0 1];

% For a transformation FROM frame {i} TO frame {i-1}: A
T = T_R_z*T_T_z*T_T_x*T_R_x;

end
```

- **Appendix G**

**Cubic Spline (MATLAB / Simulink) Code: -**

```
function coefficients = cubicSpline(T, Pf, Pi)

% Cubic spline coefficients for X, Y and Z (coloumns)
a0 = [ Pi(1,1); Pi(2,1); Pi(3,1)];
a1 = [ 0; 0; 0];
a2 = [ 3*(Pf(1,1) - Pi(1,1))/(T^2); 3*(Pf(2,1) - Pi(2,1))/(T^2); 3*(Pf(3,1) - Pi(3,1))/(T^2)];
a3 = [-2*(Pf(1,1) - Pi(1,1))/(T^3); -2*(Pf(2,1) - Pi(2,1))/(T^3); -2*(Pf(3,1) - Pi(3,1))/(T^3)];

% Output coefficients as a 3 x 4 matrix (1 row each for x, y and z)
coefficients = [a0, a1, a2, a3];
```

- **Appendix H**

**Waypoint Navigator (MATLAB / Simulink) Code: -**

```
function [Pf, Pi, time_vector, waypoint_number] = waypointNavigator(duration, waypoints, startStop, dt,
current_position)

%% Initialization of internal variables
% Use persistent variables to track states across multiple iterations, and
% define them on first call
persistent initial final next t
if isempty(initial) || isempty(final)
    initial = current_position;
    final = current_position;
end
if isempty(next)
    next = 0;
end
if isempty(t)
    t = 0;
end

% Find the number of waypoints in total
[num_waypoints, ~] = size(waypoints);

%% Timing between the waypoints
% Ensure timer remains between 0 and duration
t_mod = mod(t, duration);

% Raise the flag if the timer resets to 0 (proceed to the next waypoint)
flag = t_mod < t;

%% Updating waypoints
if flag
    % proceed to the next waypoint
    next = next + 1;

    % cycle back to the first waypoint after the last one
    if next > num_waypoints
        next = 1;
    end

    % set the final and inital internal variables
    final = waypoints(next, :);
    initial = current_position;
end
```

```
%% Update the internal timer t
t = t_mod;
if startStop
    t = t + dt;
end

%% Set the outputs
waypoint_number = next;
time_vector = [1; t; t^2; t^3];
Pf = final;
Pi = init
```

