DEVELOPMENT OF EMBEDDED CONTROL SYSTEM FOR SCARA ROBOT WITH AUGMENTED INTELLIGENCE GRAPHICAL TELEOPERATION STATION

By

Muhammad Faiez Ali (02-3-1-040-2020)

Muhammad Maaz Khan (02-3-1-045-2020)

Munad E Ali (02-3-1-059-2020)

Thesis submitted to the faculty of Engineering at PIEAS in partial fulfillment of requirements for the Degree of BS Electrical Engineering.

Department of Electrical Engineering, Pakistan Institute of Engineering and Applied Sciences,

Nilore, Islamabad 45650, Pakistan

June, 2024

Department of Electrical Engineering

Pakistan Institute of Engineering and Applied Sciences (PIEAS)

Nilore, Islamabad 45650, Pakistan

Declaration of Originality

We hereby declare that the work contained in this thesis and the intellectual content therein are the product of our own research. This thesis has not been previously published in any form nor does it contain any verbatim of the published resources which could be treated as infringement of the international copyright law. We also declare that we do understand the terms 'copyright' and 'plagiarism,' and that in case of any copyright violation or plagiarism found in this work, we will be held fully responsible of the consequences of any such violation.

Signature: _____

Name: Muhammad Faiez Ali

Signature: _____

Name: Muhammad Maaz Khan

Signature: _____

Name: Munad E Ali

Date: June 28, 2024

Place: PIEAS

Certificate of Approval

This is to certify that the work contained in this thesis entitled

"development of embedded control system for scara robot with augmented intelligence graphical teleoperation station"

carried out by

Muhammad Faiez Ali,

Muhammad Maaz Khan , and

Munad E Ali

It is fully adequate, in scope and quality, for the degree of BS Electrical Engineering from Pakistan Institute of Engineering and Applied Sciences (PIEAS).

Approved By:

Signature: _____

Supervisor: Dr. Nasir Rahman

Signature: _____

Co-Supervisor: Dr. Muhammad Aqil

Verified By:

Signature: _____

Head, Department of Electrical Engineering

Stamp:

To our loving parents, brothers, and sisters

Acknowledgements

All thanks to Almighty Allah for He has blessed us with countless bounties. We would like to express our deepest gratitude to our honorable supervisor Dr. Nasir Rahman and Co-supervisor Dr. Muhammad Aqil for guiding and supporting us throughout the project work, always showing patience, and sharing their knowledge. They have been very regular in checking our progress on the project. Furthermore, we would like to thank the Robotics and Automation Lab technicians especially Mr. Waqar Ahmad and Mr. Kaleem Dil Khan for helping us complete our project.

We would like to acknowledge other faculty members as well. They have remained very kind in teaching and guiding us. All of them are very nice, supportive.

Finally, we would like to thank our parents for their motivation and never ending support. It is because of their sacrifices, and prayers that we have reached the point, where we are.

Contents

List of Figures

Abbreviations and Acronyms

SCARA	Selective Compliance Articulated Robotic Arm
BDC	Brushed DC Motor
PWM	Pulse Width Modulation
DoF	Degree of Freedom
RTU	Remote Terminal Unit
IEEE	Institute of Electrical and Electronics Engineers
ASCII	American Standard Code for Information Interchange
UART	Universal Asynchronous Receiver-Transmitter
RS-485	Recommended Standard 485
ARR	Auto Reload Register
CCR	Capture Compare Register
CRC	Cyclic Redundancy Check
PID	Proportional Integral Derivative
IBT-2	Intelligent Power Module Bridge-two
PRR	Prismatic-Revolute-Revolute
PPR	Pulse Per Revolution
URDF	Unified Robot Description Format.
IDE	Integrated Development Environment
DH	Denavit-Hartenberg
CAD	Computer Aided Design
LRC	Longitudinal Redundnancy Check

Abstract

This thesis delves into the design and development of a teleoperation system for a modified SCARA robot utilizing an intelligence-augmented virtual environment for effective control. Phase I of the project involved designing and fabricating the mechanical hardware, which featured an offset between its first and second joints, distinguishing it from traditional SCARA robots. Phase II, covered in this thesis, focused on deploying a self-determining intelligent control system to maneuver the SCARA robot on a planar surface for picking and placing ICs.

The CAD model was designed in SolidWorks and environment simulation was performed in Coppeliasim where trajectory planning/programming was performed in order to test the mathematical model for the control system design parameters. Further, with the help of image processing in Python, the feedback loop sensors were utilized for manipulation of objects on a planar surface. The Master-Slave control system architecture has been developed, deployed and tested with the fabricated structure of the SCARA robot where Modbus communication protocol is employed between master and slaves in order to control respective joints synchronous to the pre-defined trajectory. The planar coordinates determined by image processing to pick and place the object in the CoppeliaSim virtual environment were used to analyze the path and calculate the joint angles which were then sent to the master controller via serial port for slave operations, effectively imitating the trajectory simulated in the virtual environment.

Chapter 1

Introduction

The term robot evokes images of futuristic machines, but the reality is far more diverse and impactful. The International Organization of Standardization (ISO) defines a robot as; an automatically controlled, re-programmable, multipurpose manipulator performing industrial tasks [1]. In simpler terms, robots are programmable machines that can automate tasks, traditionally performed by humans, often with greater precision, speed and consistency. The robotic applications are as diverse as their designs, from the factory floors to the healthcare applications assisting delicate procedures. The robots are now infiltrating nearly every sector. Beyond the industries, the robots are now also being used in domestic areas from cleaning homes to delivering packages, while the educational robots are igniting young minds and creating curiosity in STEM fields.

The control system of the robot defines and limits its functionality, determining how it responds to inputs and accomplishes the required tasks. Using mathematical models and algorithms, the control system enables the robot to perceive its surroundings, make decisions, and execute actions autonomously. Moreover, by designing an application-specific control system employed with intelligence and adaptability, increasingly make robots capable of operating in complex environments.

1.1 Historical Background

The history of teleoperation embedded control systems for SCARA robots is intertwined with a broader narrative of robot development. While SCARA robots have a unique architecture, their control systems have benefited from advancements in the robotics spectrum. The academic pioneers such as Jospeh Engelberger "The Father of Robotics", championed the potential of robots beyond industrial settings. Engelberger's article in the Scientific American Journal 1980, highlighted the evolution of self-replicating robots and their control systems, emphasizing the shift towards dedicated processors and embedded control solutions [2]. Later, Engel collaborative work with George D. Devol, led to the development of the first programmable industry robot, Unimate.

The earlier SCARA robots were designed by the industrial giants such as the General Motors and Westinghouse, they began experimenting with robots for automated tasks, often relying on the centralized mainframe computers for control. In 1969, Devol and Harmon's autonomous mobile robots represented early examples of embedded control technology in action [3]. The advent of powerful and affordable microcontrollers in the late 20th century revolutionized embedded control systems for SCARA robots. Koren's Robotics for Engineers serves as a valuable resource for understanding this era, highlighting the impact of microcontrollers on robot design and control [4].

Furthermore, Behm's Industrial Automation, delves deeper into the practical application of embedded control systems in industrial robots, with specific focus on the hardware and software aspects of microcontroller-based control [3]. Today, embedded control systems for robots continue to evolve, incorporating advanced algorithms, sensor fusion, and artificial intelligence. Research in areas like adaptive control, machine learning, and collaborative robotics promises even greater precision, efficiency, and autonomy for these versatile robots.

1.2 Motivation

The earlier motivation of this project lies in providing a robotic automation solution for the industries where price and repetitive movements are a crucial part of the manufacturing process. This project aims to leverage the capabilities of SCARA robot to enhance the efficiency and precision of electronic device manufacturing, particularly in the placement of integrated circuits on printed circuit boards. The successful implementation of this project will give us a chance to patent our control system and contribute to the research areas in commercial applications of robots. In addition, by providing a real-time teleoperation control of the robot using a virtual environment opens an opportunity to minimize the use of hardware sensory components and provide a better user interactive control. This fuels our motivation to expand entrepreneurial opportunities in the automation industry and enhances this project's market competitiveness among evolving robotic control systems.

1.3 Objectives

The objectives of our project are:

1. Development of an embedded control system for the SCARA robot to achieve accurate pick-and-place of ICs on a planar surface.

2. Designing a virtual environment for the SCARA robot to program and simulate the robot trajectory.

3. Integrating the virtual environment with the embedded control system architecture for the deployment of teleoperated control.

1.4 Scope

The conventional operating systems for industrial SCARA robots use operating systems with pre-defined parameters for their motion. With the use of a virtually

controlled robot control system, the SCARA robot will be able to intelligently decide its path for picking and placing the objects within its workspace. By leveraging the use of Master-Slave architecture the control system is programmed to be controlled centrally and the industrial-proven Modbus communication will allow the SCARA to be operated from far-off places.
In a broader context, this project aims to develop a prototype of SCARA robot control architecture which can be further expanded by installing multiple SCARA robots in the same work environment. This will help the industry to reduce its labor cost, nullify the human error and decrease the safety concerns regarding human-machine interaction.

1.5 Thesis Outline

In the introductory chapter of the thesis, we explore the evolution and impact of robotics in various industries, highlighting their significant role in modern society. The second chapter defines the use of embedded control system and its extensive interfaces in the context for the thesis's focus on SCARA robot and the diverse applications of automation, ranging from industrial to domestic settings.

Chapter 3 delves into the foundational elements of embedded control systems (ECS), shedding light on crucial components like sensors, actuators, and controllers. This chapter lays the groundwork for understanding the technologies and methodologies that underpin the project, paving the way for more technical discussions in subsequent chapters.

The fourth chapter is dedicated to the mathematical modeling of the SCARA robot. It intricately details the kinematic equations and transformation matrices that are vital to comprehending the robot's movement capabilities. The technical modeling is fundamental to grasp the precise functioning of the SCARA robot.

In Chapter 5, the focus shifts to software simulations, utilizing different tools. This chapter is pivotal in validating the kinematic models developed earlier, with

a thorough comparison of simulated results against theoretical predictions, emphasizing the model's accuracy and reliability.

Chapter 6 provides an in-depth look at the master-slave communication setup, centering on the implementation of the RS-485 ModBus protocol. It describes the roles of the BluePill microcontroller within this framework, and the technical aspects of motor control, including data packet queries for control functionalities.

Finally, Chapter 7 is the last chapter of this thesis which discusses the outcomes of the project and gives some future recommendations.

Chapter 2

Literature Review

2.1 Robot manipulators

Robot manipulators, commonly referred to as the **'Robotic arms'** are electronically controlled composition made up of multiple sub-assemblies which are capable of performing tasks by means of direct engagement with their surroundings [5]. The robot manipulators are majorly used in the industries focused in manufacturing and packaging. These manipulators are majorly composed of an assembly of **'links'** and **'joints'**. The links are the rigid sections of the robot while joints are the connections between the links. The tool attached to the manipulator's end, which interacts with the environment is called as the **'end effector'**.

2.1.1 Classification of Robot Manipulators

Manipulators can be classified accordingly in two separate criteria, depending upon their link and joint characteristics [3].

1. **By Motion Characteristics**

   (a) **Planar manipulator:** All the movable links move in planes parallel to the surface.

   (b) **Spherical manipulator:** All or some of the links perform spherical motions about a common stationary point.

(c) **Spatial manipulator:** At least one of the links of the mechanism possess a spatial motion.

2. **By Kinematic Structure**

(a) **Open-loop manipulator (or serial robot):** A manipulator is called an open-loop manipulator if its links form an open-loop chain.

(b) **Parallel manipulator:** A manipulator is called a parallel manipulator if it is made up of a closed-loop chain.

(c) **Hybrid manipulator:** A manipulator is called a hybrid manipulator if it consists of open loop and closed loop chains.

2.2 Selective Compliant Articulated Robot Arm (SCARA)

SCARA, short for Selective Compliant Articulated Robot Arm, is a planar manipulator with four degrees of freedom. The distinctive four jointed arms assembly, grants it flexibility and precision in handling tasks on planar surfaces. In comparison to the other robot manipulators, the industrial SCARA robots excel in the pick-and-place operation, material handling and package application which makes them a cornerstone of automated production lines [3]. The defining characteristics of SCARA Robot lies in its unique kinematic structure. The first joint which is typically a prismatic joint, allows the vertical movement of the body. The second and third joints, are commonly rotatory joints enabling horizontal reach and rotation. The final joint, which controls the end effector, is also a rotatory joint which acts for the motion of the tool tip. The Figure 2.1 mentions a common assembly of SCARA robot designed by EPSON Robotics [6], which is being widely used in packaging industries.

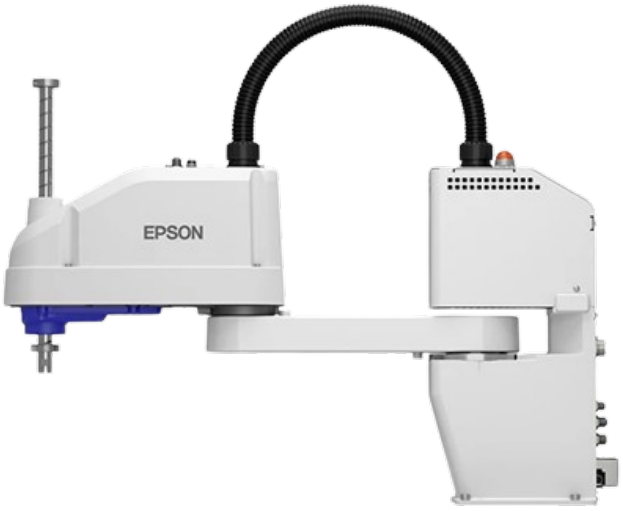


Figure 2.1: SCARA robot design by EPSON

The simplified design of the SCARA robot translates to several advantages, mainly because:

1. SCARA robot is suitable for tight spaces as it occupies less space as compared to other robot manipulators.

2. The SCARA robot has a lightweight design and it possess a focused movement, which enables it to perform rapid pick-and-place maneuvers with greater accuracy.

3. As it includes fewer components and simpler mechanism, SCARA robot is found to be more budget friendly in terms of design and manufacturing.

Its design flexibility makes it suitable for many diverse applications. The combination of precision, efficiency and affordability not only makes it suitable for research prospect, but also makes SCARA a valuable tool for business insights which are looking to streamline operations and improve the productivity.

2.3 Embedded Control Systems

In order to meet the rising requirements of automation, embedded control systems play a vital role of the operator, coordinating the complex movements of countless devices ranging from mobile devices to industrial robots. As defined by the Institute of Electrical and Electronics Engineers (IEEE), an embedded control system is a 'dedicated computer system with embedded software that is designed to control or regulate the behavior of other devices or systems' [7]. These silent operators hidden within the machines, are responsible for translating the software algorithms into precise actions for ensuring smooth operation and efficient performance. The applications of the embedded control systems are as diverse as their forms. From regulating temperatures in the vehicles and navigating spacecraft in deep space, they are responsible for interaction of hardware and software in a multitude of applications. The intricate coordination with sensors, actuators, and processors allow the systems to gather data, process information, and control devices in real time making the embedded control systems the backbone of modern automation and intelligent control services.

2.4 Master-Slave Architecture

In embedded systems, where microprocessors control the movements of complex machines, the master-slave micro-controller architecture emerges as a centralized system for communication and control. According to the International Organization for Standardization (ISO), this architecture describes 'a system in which a master device controls one or more slave devices' [8]. The master-slave architecture thrives on its simplicity and efficiency. The master device acts as the central unit, responsible for issuing commands, managing data flow, and coordinating the actions of the slave devices. These slaves, in turn, focus solely on executing their assigned tasks with precision and responsiveness, relying on the master for guidance. This division of labor allows for efficient control mechanism, particularly in

systems with repetitive or predictable tasks. An example of this architecture is demonstrated in Figure 2.2 where grid operations are being handled by conveying command signals from the master controller to the slaves. After the slaves perform the required task they send the performance results back to the master controller.

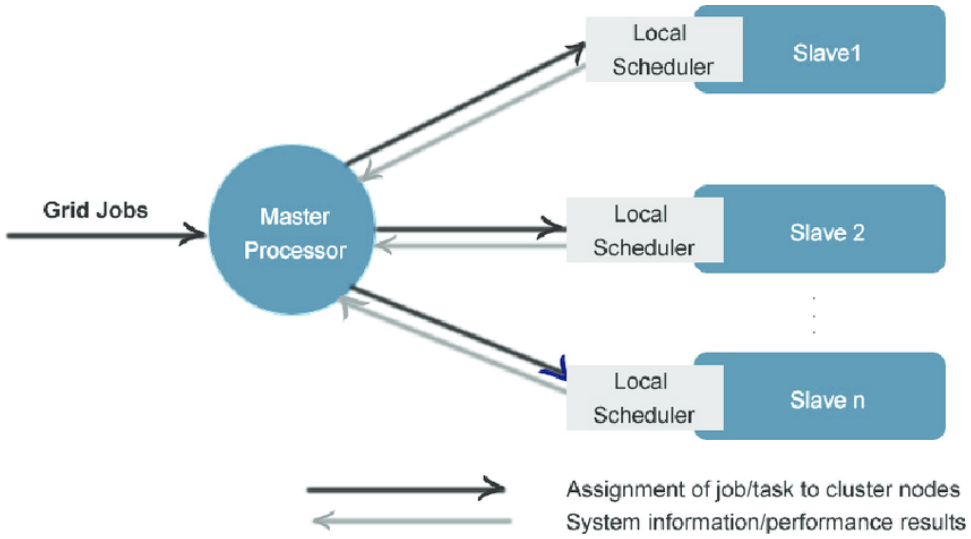


Figure 2.2: Master-Slave communication in a grid operation

By forming a clear hierarchy this architecture simplifies development and debugging, making it ideal for resource-constrained environments. Additionally, its predictable communication pattern ensures reliable data exchange and minimizes the risk of errors [9]. The scalability of the master-slave architecture makes it suitable for both small and complex systems, depending upon the adopted communication protocol compatibility, it allows the addition of devices with simple interfacing.

2.5 Modbus Communication Protocol

Among many options for system communication, the Modbus protocol is a reliable messenger, efficiently relaying data between diverse devices. As defined by the Modbus Organization, it's a 'serial communication protocol, widely used in industrial automation for connecting a master device to multiple slave devices, communicating over twisted-pair wiring' [10]. Imagine a postman navigating a

network of factories, delivering messages from a central hub to various machines, ModBus communication protocol operates in a similar way.

Applications of Modbus are across diverse industries, as this protocol is being widely used in various Programmable Logic Controllers (PLCs) responsible for operating heavy to light machines. From monitoring temperatures in power plants to controlling valves in petrochemical refineries and managing lighting systems in buildings, this versatile protocol connects the automation networks, ensuring smooth operation and efficient data exchange [11].

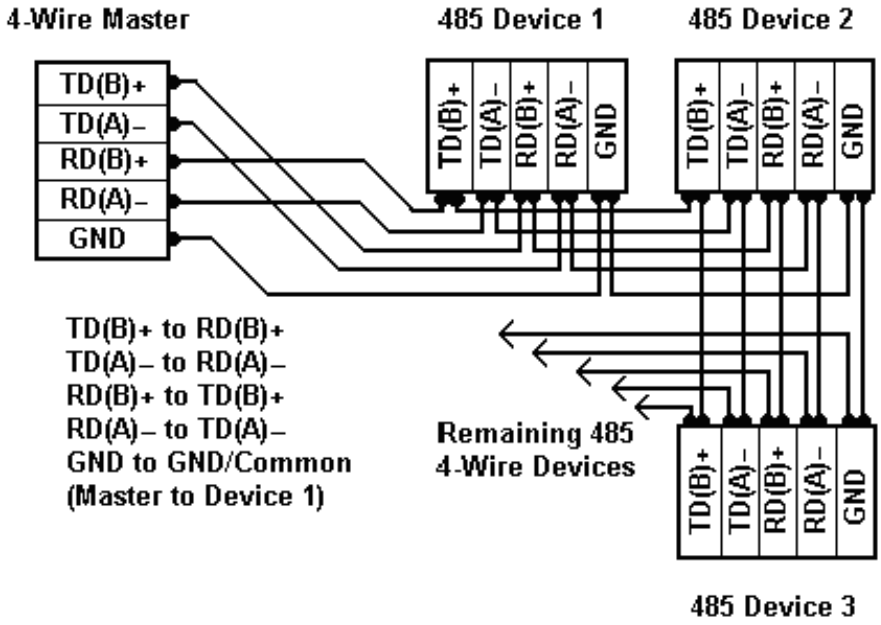


Figure 2.3: Dual channel industrial Modbus RS485 communication

The standard Modbus RS485 modular communication protocol is shown in Figure 2.3, here the master device is connected with the slaves using the 4 wire configuration, where the transmitter pins (TD(A),TD(B)) of master are connected to the receiver pins of the slaves (RD(A),RD(B)) and vice versa. This configuration enables a dual channel communication between the master and slave controllers.

2.6 Teleoperation System

As defined by the IEEE Transactions on Robotics and Automation, teleoperation refers to "the use of a human operator to control a remotely located device" [12]. The most advanced application of Teleoperation is an astronaut piloting a robotic rover on Mars, their commands traversing vast distances to guide the machine's every movement. The solidity of this protocol lies in its ability to extend human capabilities beyond physical limitations. It allows operators to interact with hazardous environments, explore remote locations, or perform delicate tasks with unmatched precision, all from the comfort of a safe and controlled setting. The example of a teleoperation and visulaization system is visible in Figure 2.4 This opens doors to new possibilities in diverse fields, from search and rescue operations in disaster zones to minimally invasive surgery and deep-sea exploration. At the heart of teleoperation lies a complex architecture of technologies. Real-time communication protocols ensure seamless transmission of sensory data (video, audio, force feedback) from the robot to the operator, while control signals flow in the opposite direction.This continuous exchange of information creates a sense of presence and allows the operator to react and adapt their actions with minimal latency. However, challenges remain like bandwidth limitations can introduce delays, hampering the feeling of presence and responsiveness. Additionally, issues of safety and security must be addressed to ensure reliable communication and prevent unauthorized access.

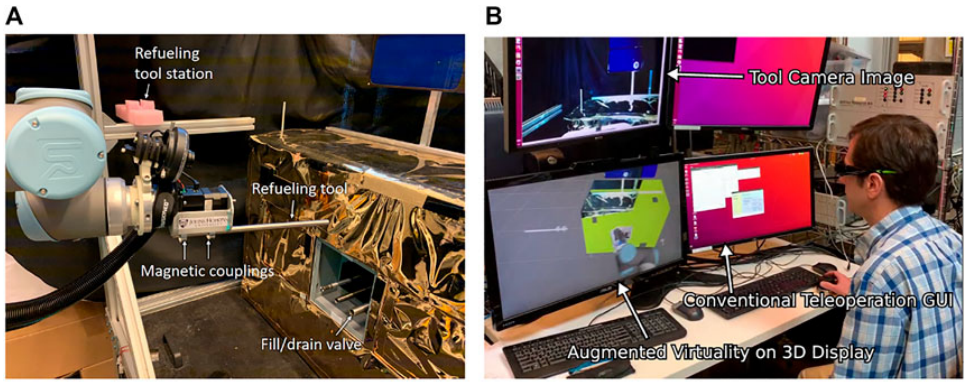


Figure 2.4: Teleoperation and visualization interface for controlling a robot

2.7 Control System for a Robot

The foundation of robot control lies in the embedded control system (ECS), coordinating the entire movement. Using sensors, actuators, and a dedicated controllers, the ECS constantly gathers data about the robot's internal state and surrounding environment [3]. This sensory information (joint angles, motor currents, external forces), serves as the data from which the commands are to be generated for constantly informing the next move. The micro-controllers serve as the backbone of the ECS, which analyze the sensory data and interprets it through advanced algorithms [13]. These algorithms (often referred to as control laws), translate the desired robot pose or path into specific commands for the actuators. Whether it's a PID controller ensuring smooth tracking of a desired path or a more complex adaptive algorithm compensating for environmental disturbances, these control laws act as the intricate guide for the robot's movements.

Finally, the actuators, which can be called as the muscles of the robot, transform the electrical signals from the controller into physical actions. The actuators can be servo motors that precisely adjust joint positions, hydraulic powered powerful movements, or pneumatic systems providing swift and agile responses [14]. These diverse actuators enable the instrumentation of the robot, translating the controller's commands into the robot's physical performance. However, robot control is not simply a one-way street. Feedback loops play a crucial role in ensuring accuracy and robustness. Sensors constantly monitor the robot's actual actions, comparing them to the desired trajectory. Any discrepancy generates corrective signals that are fed back to the controller, allowing for real-time adjustments and ensuring that the robot stays on track.

Beyond basic motion control, robots often require additional capabilities for complex tasks. Sensor fusion techniques, for example, combine data from multiple sensors to create a more comprehensive understanding of the environment [15]. This allows robots to perceive obstacles, track objects, and even interact with their surroundings in a more intelligent way. Moreover, advanced algorithms like

machine learning and computer vision can empower robots to adapt to changing environments, learn from experience, and make autonomous decisions. These tools prove to be helpful when creating a teleoperated control for the robot as they help providing better user interaction and improve learning capabilities of the robot.

2.8 Chapter Summary

In this chapter, we provide a comprehensive review of the existing literature pertinent to various aspects of robotic systems, with a particular focus on robot manipulators, SCARA robots, embedded control systems, master-slave architecture, ModBus communication protocol, teleoperation systems, and the control systems of robots. This literature review serves to contextualize the current state of research and identify gaps that this thesis aims to address.

Chapter 3

Preliminaries

This chapter provides an overview of the hardware and software components utilized in the project. Offering readers a clear understanding of the technological framework which was essential for the project's completion.

3.1 Hardware Components

Following is the list of all the hardware components that are used in our prototype, the role of each component is also explained in this section.

3.1.1 Hardware manipulator

As this project is in extension of an already fabricated design of the SCARA robot, which was the Phase-I titled as **"Design and Development of Scaled Version of SCARA Class of Robot for Handling Jobs Placed on Planer Surface with Arbitrary Orientations"**. Majority of the architecture of the SCARA robot was designed and fabricated in that phase. In this Phase-II, the electronic components; motors, encoders, limit switches and suction pump, were included in the hardware architecture. In addition, the tool roll and suction mechanism was also designed and fabricated in this phase.

Figure 3.1: Architecture of SCARA robot

The joint configuration in this modified structure of the SCARA robot is based on the PRR(Prismatic-Revolute-Revolute) manipulator robot configuration [5]. The assembly of SCARA is modified in such a way that the second joint being revolute is present at an offset to the first prismatic joint. In the fabricated architecture there is some dead area due to some design constraints and to reduce it further, the rotation of the robot along its axis is supported by the use of Geneva wheel. In addition, timing belts are used to link the pulleys to transfer the motion from the motors to the links. The design of the end effector depends upon the shape, size, and geometry of objects so a universal gripper should be used to pick random objects and place them at different orientations and positions. The most common gripper is multi-fingered but due to its complexities in programming and manufacturing, a vacuum suction technique was used which uses vacuum bellows. When it comes in contact with an object the air pump sucks the air and attaches the object to the gripper, which is then placed at the desired position by releasing the air pressure.

3.1.2 Incremental optical encoders

Incremental optical encoders are a class of rotary encoders that measure and communicate changes in angular position rather than absolute position [16]. They generate a series of pulses as a rotating shaft moves, with each pulse representing a discrete increment of angular displacement. It provides feedback to a control system, enabling it to monitor and control the movement of a mechanical system, which in our case is a motor.

Figure 3.2: OMRON E6H-CWZ6C incremental encoder

The OMRON E6H-CWZ6C, shown in Figure 3.2 which is a 1000 PPR (Pulses Per Revolution), 40 mm-dia, hollow shaft incremental encoder [17] was used in this project. At its core, a disc marked with a unique pattern of light and dark segments, known as the code disc. As this disc rotates, it passes beneath a stationary optical sensor, casting a precisely timed sequence of light and dark pulses onto its photosensitive elements. These pulses are then decoded by the encoder's internal circuitry, enabling a controller to determine the precise angular position of the shaft, as well as the speed and direction of rotation .

3.1.3 Brushed DC motors

The brushed DC motor is a widely utilized solution for converting electrical energy into mechanical motion. At its core, the functional mechanism of a brushed DC motor revolves around the interaction between magnetic fields and electric current. Within the motor housing, a rotor featuring windings or permanent magnets undergoes rotational movement when subjected to a magnetic field generated by the stator. The project utilized Maxon 12V brushed DC motors, shown in Figure 3.3, these motors have a range of 16-32 base RPM in accordance to the supplied input voltages which can be controlled. The 86:1 gear head allows these motors to offer a great deal of torque even at low voltages. These motors are attached with the respective joints using a mount and pulley mechanism with timer belts for controlled actuation.

Figure 3.3: Maxon 12V brushed DC motor

3.1.4 Arduino IBT-2 motor driver

Motor drivers function as interface circuits that take a low-power control signal and convert it into a suitable high-power signal to drive a motor. In our project we have utilized Arduino IBT-2 motor driver, shown in Figure 3.4 for interfacing the brushed DC motors with the embedded control system. In its internal circuitry lies a dual H-bridge circuit, enabling its ability to control two DC motors independently.

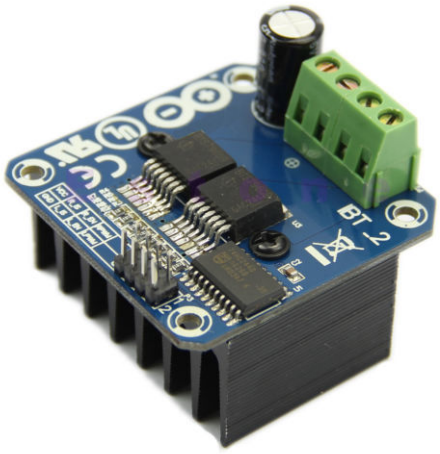


Figure 3.4: Arduino IBT-2 motor driver

Each H-bridge is composed of four transistors that are responsible to control the flow of current through a motor, enabling precise control over both its direction and speed [18]. It readily accepts direction and movement commands through digital pins, while interpreting PWM signals to fine-tune motor speeds. This adaptability extends to voltage, thanks to its built-in regulator, handling motors beyond Arduino's 5V limit. Its compact size, user-friendly screw terminals, and overcurrent protection further solidify its popularity among makers, making it a reliable component in our project.

3.1.5 SMT32f103C8T6 microcontroller (Blue Pill)

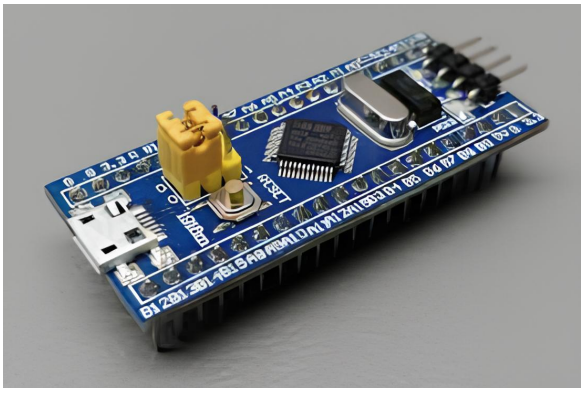


Figure 3.5: Blue Pill microcontroller

In the embedded control system design of our project, the master-slave architecture involves multiple micro-controllers acting as either a master or slave.

STM32f103C8T6 (Blue Pill) was used for the choice of both master and slave micro-controllers, shown in the Figure 3.5 as it aligns with the intricacies of our control system. The 32-bit Cortex-M3 ARM microprocessor of the Blue Pill provides a 72MHz stable operating clock frequency, with high-performance computing and floating-point computation capability, making it suitable for real-time applications [19]. The diverse arrays of peripherals and 37 general purpose input output (GPIO) pins with 10 analog pins, including timers, communication interfaces, and analog-to-digital converters, provide the necessary environment to interact with sensors, actuators, and several other external components. This high cost effectiveness and ease of access in the market made Blue Pill the suitable choice as a development board for our project.

3.1.6 MAX485 TTL to RS485 - MAXC85CSA converter

In our project, communication between the master and slave microcontrollers is facilitated by the MAX485 TTL (Transistor-to-Transistor Logic) to RS485 MAXC85CSA converter boards, visible in Figure 3.6. These compact yet powerful communication protocol boards, providing Modbus communication serve as a crucial link, ensuring reliable data exchange between the master and slave microcontrollers [11].

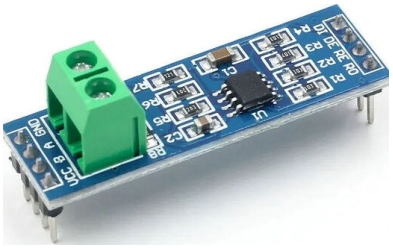


Figure 3.6: MAX485 TTL to RS485 converter

The pin configuration of the MAXC85CSA board aligns with the USART pins of the microcontrollers, simplifying the wiring process and enhancing compatibility within our master-slave framework. Utilizing the RS485 communication protocol, a single master board can efficiently broadcast data to multiple MAX485 converter

boards connected to the slave micro-controllers. This enables a streamlined and organized communication channel, allowing the master to transmit information effectively, while each individual MAXC85CSA converter ensures that the intended data is received and processed by the respective slave. This configuration optimizes data distribution in our project, offering a scalable and efficient solution for communication in our control system.

3.1.7 Linear magnetic hall effect sensor limit switches

Linear magnetic hall effect sensor boards were used as limit switches for the revolute joints. These sensor boards, shown in Figure 3.7 leverage the Hall effect principle to detect the presence of a magnetic field and provide an accurate measurement of its strength and polarity. They provide an analog voltage output proportional to the magnetic field strength, allowing for continuous monitoring and they can be fine-tuned by adjusting the potentiometer resistance [20]. This analog output facilitates easy integration with various micro-controllers and control systems.

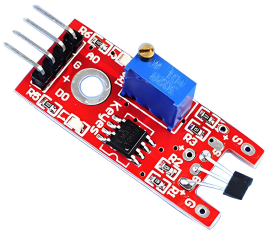


Figure 3.7: Linear magnetic hall effect sensor board

The Linear Magnetic Hall Effect Sensor boards are employed to detect the end-of-travel positions of the rotatory joint links. A magnet is attached at a suitable position on the link and when the sensor detects the magnetic field generated by the strategically placed magnet it sends an analog signal to the control system. This signal is then interpreted to stop or reverse the movement, ensuring precise control over the component's position and preventing mechanical overrun or damage.

3.1.8 Roller level limit switches

The roller level limit switches were used to detect the limit of the prismatic joint of the SCARA robot. These switches visible in Figure 3.8 are mechanical devices featuring a roller actuator that is triggered when the prismatic structure comes in contact with it. When the prismatic joint reaches its designated end-of-travel position, the roller engages with a cam or stopper, causing the switch to actuate and send a signal to the control system. This action ensures precise detection of the joint's limits, preventing over extension and potential mechanical damage [21].

Figure 3.8: Roller level limit switch

3.1.9 Control station

This project incorporates a control station designed to provide comprehensive control and monitoring capabilities for the embedded control system. The control station is hosted on a personal computer, featuring an intuitive user interface software that establishes communication with the master microcontroller through COM (communication) port serial monitoring. This interface allows users to exert precise control over the SCARA robot's movements, including trajectory adjustments, speed modulation, system halts, and initialization commands and warnings for system errors. The station serves as a centralized monitor for overseeing the robot's operations, receiving real-time feedback from limit switches. Additionally,

the control station interfaces with encoders, capturing and analyzing the robot's motion data for further assessment and optimization [11].

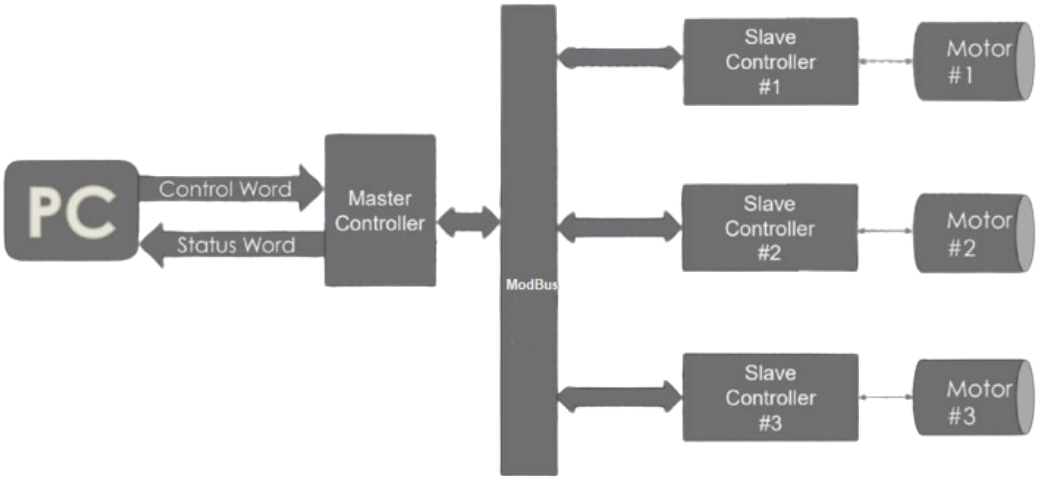


Figure 3.9: Master-Slave architecture with control station

The user sends the control word to the master controller which initializes the robot movement by sending commands to the slaves through the Modbus protocol, the feedback of the actuation is sent from the encoders to their respective slave controllers which is then sent to the master and the master data is read by the user as the status word for each joint movement.

3.2 Software components

The project is underpinned by a framework of software components, each playing a focal role in ensuring the success and functionality of the overall system. The use of these software in our project is briefly explained in this section.

3.2.1 MATLAB

MATLAB, a versatile and powerful computational tool, played a critical role in performing the calculations related to the motion analysis of the SCARA robot. As we executed calculations using the MATLAB's extensive library of mathematical functions to determine the forward and inverse kinematic equations of our robot, by utilizing the pre-defined data about the joint lengths and range of motion

of individual joints. MATLAB, tool also enabled us to determine the precise trajectory equations which were utilized in the programming of real-time operating system for the robot's control.

3.2.2 CoppeliaSim

For the simulation and analysis of SCARA robot's workspace, CoppeliaSim robotics tool was used, offering an immersive environment for dynamic simulations for robots. Utilizing the Universal Robot Description Format (URDF) file extracted from SolidWorks, the prismatic and revolute joints of the robot were accurately placed within the model. Coppeliasim allowed for the establishment of a systematic hierarchy, defining the robot's structure from its base to the end effector. The tool's compatibility with both threaded and non-threaded Lua and Python languages facilitated the dynamic simulation of the robot, enabling us to rigorously test and analyze its motion within the defined workspace envelope.

3.2.3 SolidWorks

SolidWorks was used as the core tool in the designing of SCARA robot, facilitating the design of individual parts and assembly. The software's professional interface and extensive features allowed the creation of a comprehensive 3D model of our SCARA robot., Utilizing SolidWorks, we navigated the design process with ease, ensuring precision in every aspect of our robot's structure. To extend the utility of our model, SolidWorks provided an efficient Universal Robot Description Format (URDF export tool, enabling the conversion of our detailed design into a URDF file). This export feature allowed us to integrate our SolidWorks model into other simulation software, fostering a cohesive workflow and ensuring the accurate representation of our robot's design across various platforms.

3.2.4 STM32CubeMx

STM32CubeMX played a necessary role in configuring the slave Blue Pill (STM32f103C8T6) microcontrollers, serving as a fundamental tool in our project's development. This software streamlined the setup process by allowing us to define crucial parameters such as GPIO configurations, USART settings, clock frequencies, and timer interrupts for the Blue Pill microcontrollers. Its significance lies in its ability to generate the necessary initialization code based on the specified configurations, significantly expediting the configuration process. CubeMX's intuitive graphical interface provided a user-friendly environment for defining peripheral settings, offering a visual representation of the microcontroller's configuration.

3.2.5 Keil µVision IDE

The Keil µVision IDE was used for programming and emulating the STM microcontrollers. With an array of extensive tools, Keil µVision streamlined the coding process by facilitating the inclusion of essential libraries for configuring the Blue Pill microcontroller. Its significance lies in its efficiency and reliable interface, enabling seamless coding, debugging, and emulation of the slave microcontroller. Keil's comprehensive set of features allowed us to incorporate algorithms, ensuring precise control and coordination in our embedded control system.

3.3 Chapter Summary

In this chapter all the hardware and software components introduced were organized to perform their individual tasks resulting in successful completion of the project.

Chapter 4

Kinematic Modeling

This chapter includes the complete set of mathematical calculations for the kinematic modeling including the D-H parameters, forward and inverse kinematics followed by path planning of the SCARA robot. Kinematic mathematical modeling is the step in determining the set of parameters and their relationships that are required to find joint angles with respect to the end effector coordinates. By using these calculations, we will be able to generate the real-time operating system algorithms for controlling the SCARA robot for its pick and place operations.

4.1 Forward kinematics

The forward kinematics provide the tool tip positions with respect to the base joint, followed by motion (acceleration and velocity) given its current configuration (joint positions and velocities) and the applied joint torques. This allows engineers to simulate and analyze robot behavior under various control inputs. For forward kinematics, we need to determine the link coordinate diagram of the SCARA robot, with which the D-H parameters, link coordinate transformation matrices, the arm matrix, and arm equation are determined [3].

4.1.1 Link coordinate diagram

A link coordinate diagram serves as the first step, in the structure design of a robot. As defined by Craig [14], it's a "graphical representation of a robot's kinematic structure, showing the connections and relative positions of its individual links." The link coordinate diagram is used to assign reference and design transformation matrices for each link. These diagrams employ a systematic approach to assign reference frames and define transformation matrices for each link. By visualizing and analyzing link coordinate diagrams, researchers and engineers gain invaluable insights into the geometric constraints and kinematic equations of the robot.

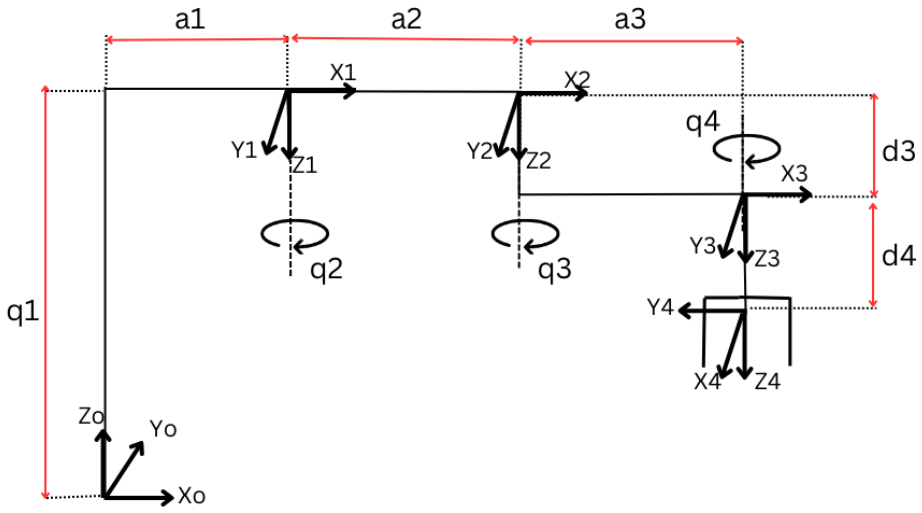


Figure 4.1: The link coordinate diagram of SCARA

The link coordinate diagram of our SCARA robot is shown in Figure 4.1. Here each joint is represented by an individual axis with reference to the base axis of the robot acting as the origin. The base is labeled as the 0th axis where the prismatic joint is placed, the 1st and 2nd axis show the respective revolute joints, while the 3rd and 4th axis signifies the third revolute joint along with the tooltip's roll movements. The origin to the end effector link lengths are labelled as "ak" where "a1", "a2" and "a3" represent the link lengths from the prismatic joint to the shoulder, shoulder to the elbow, and then from elbow to tool tip in the same order. The lengths "d1" and "d2" are the tooltip joint lengths from the previous axes. Additionally, "qk" shows the joint movements about their respective axes.

4.1.2 D-H parameters

Denavit-Hartenberg (D-H) parameters act as a universal language for assigning four specific parameters to each joint in a robot, providing a consistent way to describe its configuration [14]. In D-H parameters, several variables are used to describe different design characteristics including;

- θ **(qk)**: The joint angle, which is the rotational movement of one link relative to the next.
- **d (dk)** The link offset, defined as the distance between consecutive joint axes along a common normal.
- **a (ak)**: The link length, defined as the distance between consecutive joint axes along the previous link's axis.
- α **(**α**k)**: The twist angle is the angle between the common normals of adjacent joint axes.

here k = axis number. These parameters as shown in Table 4.1 and Table 4.2 represent the systematic coordinate frames at each joint, creating a consistent reference system for representing position and orientation throughout the robot's structure. Crucially, D-H parameters enable the calculation of homogeneous transformation matrices that describe the relative position and orientation between any two links of the robot. These matrices form the building blocks for essential kinematic computations.

Table 4.1: Evaluated D-H parameters of SCARA robot

Axis	$\boldsymbol{\theta}$	**dk**	**ak**	$\boldsymbol{\alpha}k$	Home
1	0	q1	a1	π	h1
2	q2	0	a2	0	0
3	q3	d3	a3	0	0
4	q4	d4	0	0	$\pi/2$

Table 4.2: Designated D-H parameters of SCARA robot

Axis	$\boldsymbol{\theta}$	**dk**	**ak**	$\boldsymbol{\alpha}k$	Home
1	0	q1	145.4	π	59
2	q2	0	250	0	0
3	q3	86	152.7	0	0
4	q4	114	0	0	$\pi/2$

4.1.3 Link coordinate transformation matrix

The link coordinate transformation matrix is a mathematical representation used to describe the relationship between the coordinates of adjacent robotic links in a manipulator. It enables the translation and rotation of one coordinate system, typically associated with one link, to another coordinate system, often associated with an adjacent link. The transformation matrix for a specific link is denoted by $\mathrm{T}^{\mathrm{k}}_{\mathrm{k}-1}$ where k represents the link number. The general link coordinate transformation matrix for the SCARA robot [22] can be expressed in the form 4.1:

$$\mathrm{T}^{\mathrm{k}}_{\mathrm{k}-1} = \begin{pmatrix} \mathrm{C}\theta_k & -\mathrm{C}\alpha_k\ \mathrm{S}\theta_k & \mathrm{S}\alpha_k\ \mathrm{S}\theta_k & a_k\mathrm{C}\theta_k \\ \mathrm{S}\theta_k & \mathrm{C}\alpha_k\mathrm{C}\theta_k & -\mathrm{S}\alpha_k\mathrm{C}\theta_k & a_k\ \mathrm{S}\theta_k \\ 0 & \mathrm{S}\alpha_k & \mathrm{C}\alpha_k & d_k \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.1}$$

here,

- "C" and "S" represent sine and cosine respectively
- "θ_k" is the joint angle
- "α_k" is the link twist angle
- "a_k" is the link length
- "d_k" is the joint offset

This link transformation matrix facilitates the description of the orientation of a given link with respect to its adjacent link which is significant in performing the kinematic modeling of the robot. By using the general link coordinate transformation matrix, as shown in the matrix 4.1, we determine the subsequent transformation matrices from the joint "k-1" to "k" by following the values assigned by the D-H parameters.

4.1.3.1 Transformation matrix from joint 0 to 1

$$\mathrm{T}_0^1 = \begin{pmatrix} 1 & 0 & 0 & a_1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & h_1 + q_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

4.1.3.2 Transformation matrix from joint 1 to 2

$$\mathrm{T}_1^2 = \begin{pmatrix} c_2 & -s_2 & 0 & a_2 c_2 \\ s_2 & c_2 & 0 & a_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.3}$$

4.1.3.3 Transformation matrix from joint 2 to 3

$$\mathrm{T}_2^3 = \begin{pmatrix} c_3 & -s_3 & 0 & a_3 c_3 \\ s_3 & c_3 & 0 & a_3 s_3 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.4}$$

4.1.3.4 Transformation matrix from joint 3 to 4

$$\mathrm{T}_3^4 = \begin{pmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.5}$$

4.1.4 The arm matrix

The arm matrix is the coordinate transformation matrix from the base joint to the tooltip. In particular, if T_{base}^{tool} represents, the transformation from tool-tip coordinates (link n) to base coordinates (link 0), then:

$$T_{base}^{tool} = T_0^4(q) = T_0^1(q1)T_1^2(q2)T_2^3(q3)T_3^4(q4)$$

when a sequence of coordinate transformations is performed, the coordinate transformation algebra leads to the formation of the arm matrix 4.6 with a linear set of equations.

$$ArmMatrix = T_0^4(q)$$

$$\mathrm{T}_0^4 = \begin{pmatrix} \sigma_3 - \sigma_2 & \sigma_1 & 0 & a_1 + a_2c_2 + a_3c_2c_3 - a_3s_2s_3 \\ \sigma_1 & \sigma_2 - \sigma_3 & 0 & -a_2s_2 - a_3c_2s_3 - a_3c_3s_2 \\ 0 & 0 & -1 & h_1 - d_4 - d_3 + q_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.6}$$

here

$$\sigma_1 = -c_4\left(c_2s_3 + c_3s_2\right) - s_4\left(c_2c_3 - s_2s_3\right)$$

$$\sigma_2 = s_4\left(c_2s_3 + c_3s_2\right)$$

$$\sigma_3 = c_4\left(c_2c_3 - s_2s_3\right)$$

$$c_k = \text{cosine of angle } (q_k)$$

$$s_k = \text{sine of angle } (q_k)$$

The arm equation when simplified using basic trigonometric identities is given in matrix equation 4.7.

$$Arm\ Equation = \begin{pmatrix} c_{234} & -s_{234} & 0 & a_1 + a_2c_2 + a_3c_{23} \\ -s_{234} & -c_{234} & 0 & -a_2s_2 - a_3s_{23} \\ 0 & 0 & -1 & q_1 - d_3 - d_4 + h_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.7}$$

Here, by including the values of (q_n) in the forward kinematic arm equation we can determine the translation or revolute motion of each joint with reference to the design parameters of the SCARA robot.

4.2 Inverse kinematic equations

Inverse kinematics is a major concept in robotics that addresses the challenge of determining the joint configurations required for a robot's tool tip to achieve a desired position in its working envelope [23]. Unlike forward kinematics, which involves calculating the tool tip's position based on given joint angles, inverse kinematics involves solving the mathematical relationships that allow us to find the joint angles needed to position the tool tip as desired.

4.2.1 Tool-configuration vector

The tool configuration vector specifies the rotational and translational aspect of the tool tip. The vector typically includes information about the rotational angles or parameters that define the orientation and position of the tool in three-dimensional space. By using the previously calculated arm equation 4.7 of the SCARA robot, we get the following expression 4.8 for the tool-configuration vector.

$$w(q) = \begin{pmatrix} a_1 + a_2c_2 + a_3c_{23} \\ -a_2s_2 - a_3s_{23} \\ q_1 - d_3 - d_4 + h_1 \\ 0 \\ 0 \\ -e^{q_4/\pi} \end{pmatrix} \tag{4.8}$$

here,

$$c_{xy} = cosine(x + y) \; and \; s_{xy} = sine(x + y)$$

4.2.2 Joint angle calculations

In order to extract the joint angles q_n from the nonlinear tool-configuration vector w(q), we again apply row operations to the components of $w(q)$ and exploit trigonometric identities in order to isolate the individual joint variables.

As,

$$w_1 = a_1 + a_2c_2 + a_3c_{23} \tag{4.9}$$

$$w_2 = -a_2s_2 - a_3s_{23} \tag{4.10}$$

$$w_3 = q_1 - d_3 - d_4 + h_1 \tag{4.11}$$

$$w_4 = 0 \tag{4.12}$$

$$w_5 = 0 \tag{4.13}$$

$$w_6 = -e^{q_4/\pi} \tag{4.14}$$

Now,

$$w_1' = w_1 - a_1$$

$$a_2c_2 + a_3c_{23} = w_1 - a_1$$

Taking sqaure on both sides of the above equation,

$$(a_2c_2 + a_3c_{23})^2 = (w_1 - a_1)^2$$

$$a_2^2c_2^2 + a_3^2c_{23}^2 + 2a_2c_2a_3c_{23} = (w_1 - a_1)^2$$

Also,

$$w_2^2 = (-a_2s_2 - a_3s_{23})^2$$

$$w_2^2 = a_2^2 + a_3^2s_{23}^2 + 2a_2a_3s_2s_{23}$$

$$(w_1 - a_1)^2 + w_2^2 = a_2^2 + a_3^2 + 2a_2a_3(c_2c_{23} + s_2s_{23})$$

further, by using the trigonometric identities

$$(w_1 - a_1)^2 + w_2^2 = a_2^2 + a_3^2 + 2a_2a_3c_3$$

$$q_3 = \pm cos^{-1}(\frac{(w_1 - a_1)^2 + w_2^2 - a_2^2 - a_3^2)}{2a_2a_3}) \tag{4.15}$$

From eq 4.11

$$q_1 = w_3 + d_3 + d_4 - h_1 \tag{4.16}$$

$$w_1 - a_1 = a_2c_2 + a_3c_{23} = a_2c_2 + a_3(c_2c_3 - s_2s_3)$$

$$w_1 - a_1 = a_2c_2 + a_3c_2c_3 - a_3s_2s_3$$

$$w_2 = -a_2s_2 - a_3s_{23} = -a_2s_2 - a_3(c_2s_3 + c_3s_2)$$

$$w_2 = -a_2s_2 - a_3c_2s_3 - a_3c_3s_2$$

Hence, we have

$$w_1 - a_1 = (a_2 + a_3c_3)c_2 - (a_3s_3)s_2 \tag{4.17}$$

$$w_2 = -(a_3s_3)c_2 - (a_2 + a_3c_3)s_2 \tag{4.18}$$

By solving eq 4.17 and eq 4.18 simultaneously using MATLAB, we get the joint angle equation for q_2. which results in:

```
syms a1 a2 a3 c3 s3 c2 s2 w1 w2;
eq1=(a2+a3*c3)*c2-a3*s3*s2==w1-a1;
eq2=-a3*s3*c2-(a2+a3*c3)*s2==w2;
solution = solve([eq1, eq2], [c2, s2]);
disp(['c2: ', char(solution.c2)]);
disp(['s2: ', char(solution.s2)]);
tanq2=solution.s2/solution.c2
```

Figure 4.2: Solving for q_2

$$q_2 = \pm tan^{-1}(\frac{a_2w_2 - a_1a_3s_3 + a_3c_3w_2 + a_3s_3w_1}{a_1a_2 - a_2w_1 + a_1a_3c_3 - a_3c_3w_1 + a_3s_3w_2}) \tag{4.19}$$

From eq 4.14

$$q_4 = \pi ln|w_6| \tag{4.20}$$

Hence, the calculated joint angle equations are;

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} w_3 + d_3 + d_4 - h_1 \\ \pm \tan^{-1}\left(\frac{a_2w_2 - a_1a_3s_3 + a_3c_3w_2 + a_3s_3w_1}{a_1a_2 - a_2w_1 + a_1a_3c_3 - a_3c_3w_1 + a_3s_3w_2}\right) \\ \pm \cos^{-1}\left(\frac{(w_1 - a_1)^2 + w_2^2 - a_2^2 - a_3^2}{2a_2a_3}\right) \\ \pi \ln |w_6| \end{bmatrix} \tag{4.21}$$

The solution to the inverse kinematics equations, outlined above is based on the assumption that the tool-configuration vector w is supplied as input data. As an alternative, the pair {p, R} can be also be specified.

4.3 Path planning

The workspace envelope of a SCARA robot represents the spatial region within which the end-effector can operate [14]. Path planning involves determining a smooth and feasible path for the robot to traverse from its initial position to the desired end position while adhering to kinematic and dynamic constraints. Efficient trajectory planning for SCARA robots must take into account factors such as joint limitations, velocity constraints, and obstacle avoidance to ensure safe and optimal motion.

For the purpose of illustrations, let us assume that the SCARA is initially at its home position, it is required to pick up a cube placed at a certain position H(0)(x0,y0,z0) and place it at a target position H(1)(x1,y1,z1). The SCARA robot will move along the pre-defined via points named as Lift-off, Pick-up, Drop-off, and Set Down to complete its trajectory. However, the joint angles computed by the joint angle equations 4.21 do not always provide the same set of angles to achieve the pick up point and drop off point coordinates. But as the desired pick and place motion is successfully achieved then the dynamic variation of the joint angles becomes insignificant.

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 + k \end{bmatrix} \quad \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \quad \begin{bmatrix} x_0 \\ y_0 \\ z_0 + k \end{bmatrix} \quad \begin{bmatrix} x_1 \\ y_1 \\ z_1 + k \end{bmatrix} \quad \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$$

(a) Lift off (b) Pick up (c) Lift off (d) Drop off (e) Set down

In this trajectory, SCARA's tool tip will move to the lift off point from its home position, then it will pick up the object present at the pick up point and come back to the lift off position. SCARA will then move towards the drop off point and will place the object down to the set down point.

4.4 Chapter Summary

The chapter covered the comprehensive kinematic modeling of the SCARA robot, involving a range of mathematical and trigonometrical concepts essential for understanding and controlling the robot's movements. The detailed analysis provided in this chapter forms the foundation for subsequent discussions on control and teleoperation systems. Key topics covered include forward kinematic equations, link coordinate diagrams, Denavit-Hartenberg (D-H) parameters, link coordinate transformation matrices, the arm matrix, inverse kinematic solutions, joint angle calculations, and path planning of the SCARA robot.

Chapter 5

Software Simulations

This chapter covers simulations and modeling for the SCARA robot in software environments. The simulations were performed using the software components mentioned in Chapter 2. These simulations helped to test and analyze the design and perform calculations for the SCARA robot's motion control. The complete path for the software simulations for the design analysis can be explained by the following flow chart 5.1.

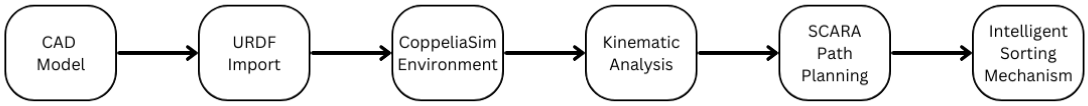


Figure 5.1: Procedural simulations flowchart

5.1 CAD model

The computer aided design (CAD) model of the SCARA robot was modeled in SolidWorks. The CAD model visible in Figure 5.2 encapsulates integration of various components such as parts, joints, links, and actuators. This assembly is a digital counterpart, allowing for the visualization and examination of the SCARA robot, which was designed in phase I of this project. Key features of the SCARA robot, including its selective compliance and articulated arm structure compo-

nents mentioned in Figure 5.3 , are graphically represented in the SolidWorks assembly. The model also facilitates the derivation of joint parameters and exploration of potential design improvements, as we can iterate and simulate different configurations before finalizing the physical prototype.

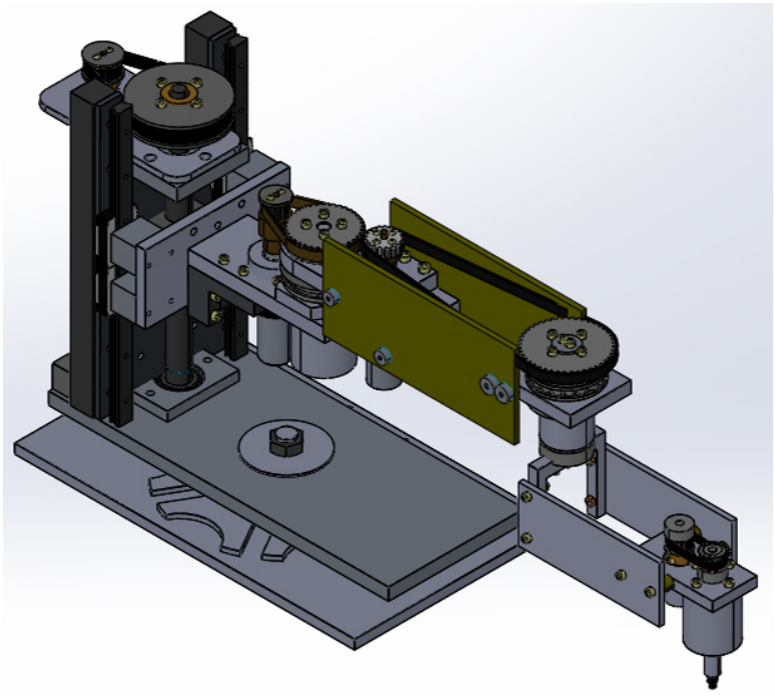
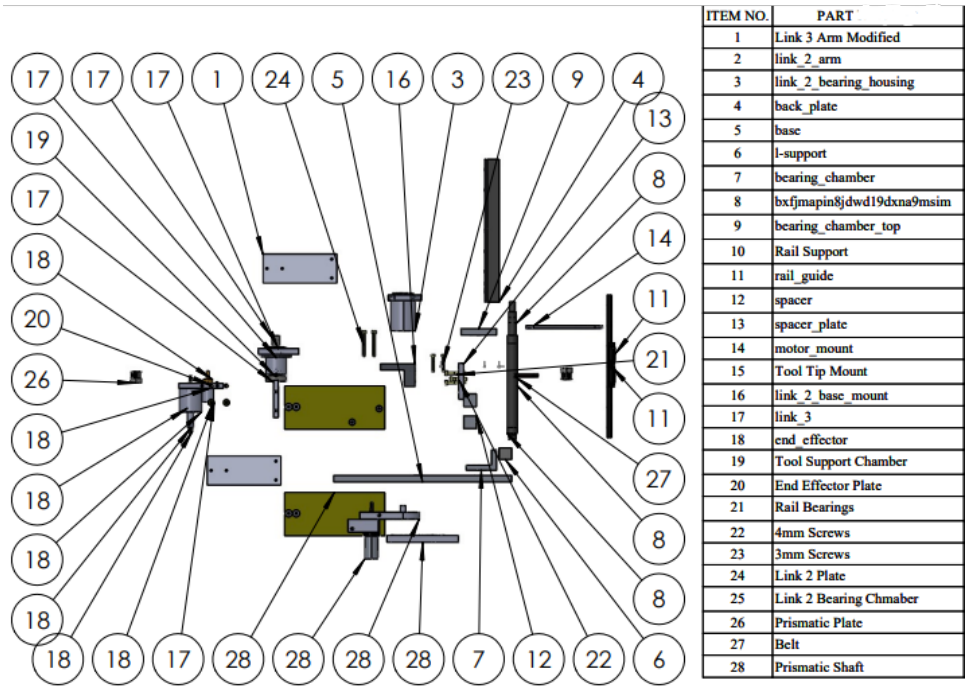


Figure 5.2: SCARA CAD model (Isometric view)



ITEM NO.	PART
1	Link 3 Arm Modified
2	link_2_arm
3	link_2_bearing_housing
4	back_plate
5	base
6	l-support
7	bearing_chamber
8	bxfjmapin8jdwd19dxna9msim
9	bearing_chamber_top
10	Rail Support
11	rail_guide
12	spacer
13	spacer_plate
14	motor_mount
15	Tool Tip Mount
16	link_2_base_mount
17	link_3
18	end_effector
19	Tool Support Chamber
20	End Effector Plate
21	Rail Bearings
22	4mm Screws
23	3mm Screws
24	Link 2 Plate
25	Link 2 Bearing Chmaber
26	Prismatic Plate
27	Belt
28	Prismatic Shaft

Figure 5.3: CAD model components

5.2 URDF to CoppeliaSim

The integration of SolidWorks and CoppeliaSim, facilitated by the URDF file, streamlines the design-to-simulation workflow, providing a powerful platform for in-depth analysis and optimization of the robot's behavior before physical implementation. This approach enhances the efficiency of the development process, enabling refinement and validation of the robot's design in a virtual environment.

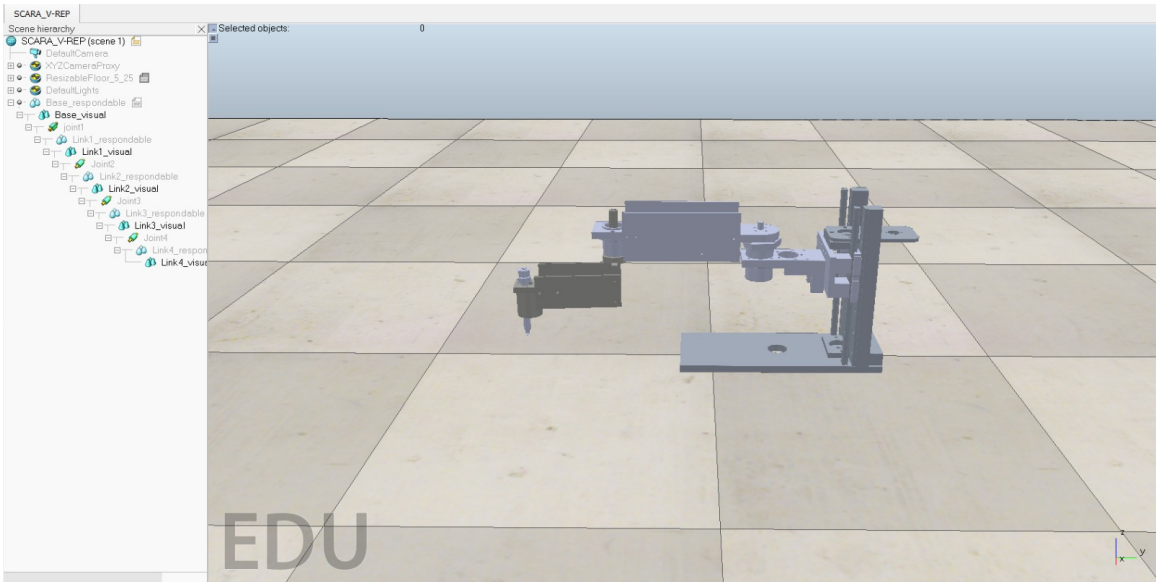


Figure 5.4: Coppleiasim work environemnt

The joint hierarchy mentioned in Figure 5.5 was also defined for the SCARA robot from its Base to the Tool tip. The joint hierarchy reflects the physical connections and dependencies between the robot's links, ensuring that the motion of each joint influences the overall orientation of the robot.

5.3 Workspace envelope

Determining the workspace envelope involves analyzing the robot's kinematics and understanding the limitations imposed by its joint ranges. We used MATLAB to determine the workspace envelope of SCARA robot, by employing forward kinematics to compute the end-effector positions for various joint configurations. By systematically varying the joint angles within their allowable ranges, a set of

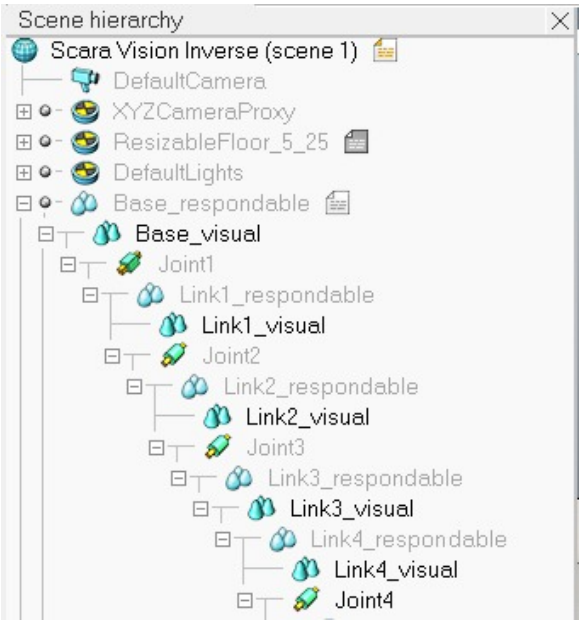


Figure 5.5: Joint hierarchy of SCARA robot

Cartesian coordinates were generated representing the reachable positions of the tool tip. The enveloping shape of these points then defines the workspace.

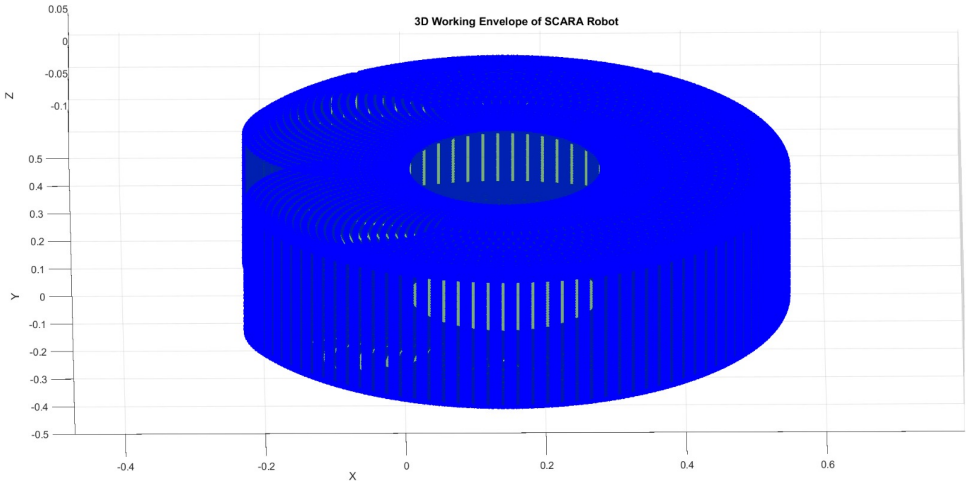


Figure 5.6: SCARA workspace envelope

The shape of the SCARA robot's workspace envelope as shown in Figure5.6 , is cylindrical with some spherical regions. The cylindrical shape arises due to the rotational nature of the joints, and the spherical regions are present near singularities or in areas where the robot's motion is more constrained.

5.4 Forward kinematic analysis

The forward kinematic analysis of the SCARA robot involved a two-step process. Initially, forward kinematic equations from equation matrix 4.7 were used in MATLAB to compute the end-effector coordinates based on desired joint angles. Subsequently, these joint angles were implemented in a CoppeliaSim simulation using Lua programming. The simulation allowed us to visualize and validate the robot's movements. To ensure accuracy, the calculated coordinates from MATLAB were cross-checked against the coordinates obtained directly from CoppeliaSim, specifically by inspecting the tooltip's position.

5.4.1 MATLAB - Forward kinematic analysis

We initialized the joint translation length and rotational angles in the forward kinematic equations programmed in MATLAB in accordance to the designed model of the SCARA robot. Here,

Translation of Prismatic Joint 1 = q1 = 0.18m

Rotational Angle of Joint 2 = q2 = 30 degrees

Rotational Angle of Joint 3 = q3 = 45 degrees

Rotational Angle of Joint 4 = q4 = 60 degrees

The MATLAB program performed the forward kinematic calculations and produced the desired Arm Equation parameter values, shown in Figure 5.7. The fourth column of the Arm Equation gave the 3 axis (x, y, z) coordinates of the tool tip.

5.4.2 CoppeliaSim - Forward kinematic analysis

The SCARA robot scene hierarcy was also programmed in Lua programming language using Coppeliasim, for the same set of forward kinematic designated joint displacement and angular values as used in MATLAB.

```
T = 4×4
   -0.7071   -0.7071         0    0.4016
   -0.7071    0.7071         0   -0.2725
         0         0   -1.0000    0.0390
         0         0         0    1.0000
```

Figure 5.7: Arm equation calculated by MATLAB

The resulting end effector orientation was monitored in CoppeliaSim environment shown in Figure5.8 and compared with the MATLAB results from Figure5.7, to verify the forward kinematic equations.

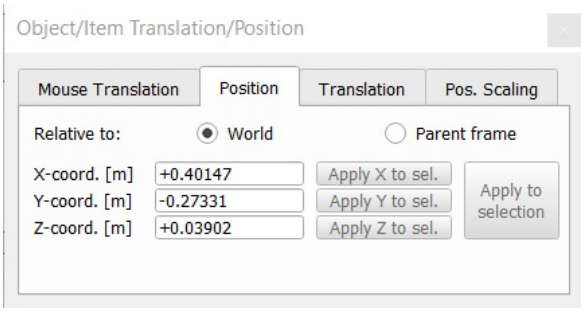


Figure 5.8: Tooltip orientation in CoppeliaSim verifying arm equation

This iterative approach of comparing MATLAB calculations with CoppeliaSim simulations provided a verification mechanism, ensuring that the robot's kinematics were accurately modeled and validated through a comprehensive comparison of calculated and simulated coordinates.

5.5 Inverse kinematic analysis

The inverse kinematic analysis of the SCARA robot also involved the same two-step process, as in forward kineamtic analysis.

5.5.1 MATLAB- Inverse Kinematic Analysis

We initialized the MATLAB code for inverse kinematics with the desired tooltip coordinates, extracted from the fourth column of the Arm Equation5.7. MATLAB

program performed the inverse kinematic calculations and produced the desired angles mentioned in Figure5.9 for individual joint rotation.

```
q1 = 0.0850
q2 = -77.5120
q3 = -90.8114
```

Figure 5.9: MATLAB calculated angles of individual joints

5.5.2 CoppeliaSim - Inverse kinematic analysis

Similarly, the desired orientation of tooltip was given as input to the SCARA robot scene hierarchy by Lua programming in CoppeliaSim. From where the tool tip orientation shown in Figure5.10 was monitored and compared with the MATLAB input parameters, to verify the inverse kinematic equations. The results of the

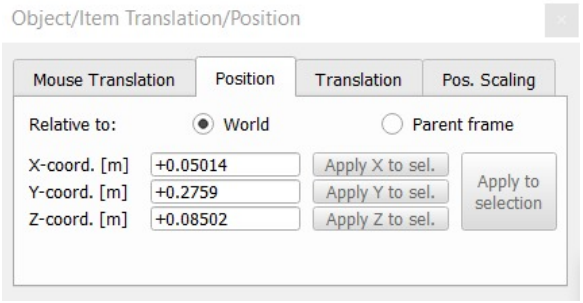


Figure 5.10: Achieved tool position from Coppelia scene

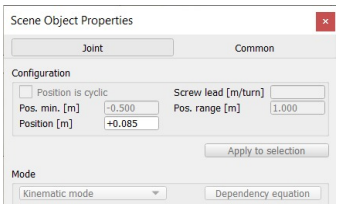


Figure 5.11: Joint 1

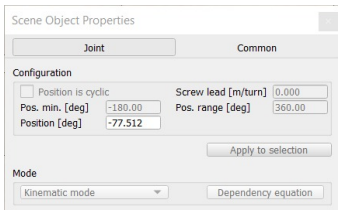


Figure 5.12: Joint 2

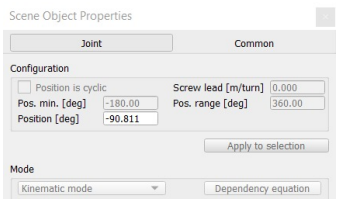


Figure 5.13: Joint 3

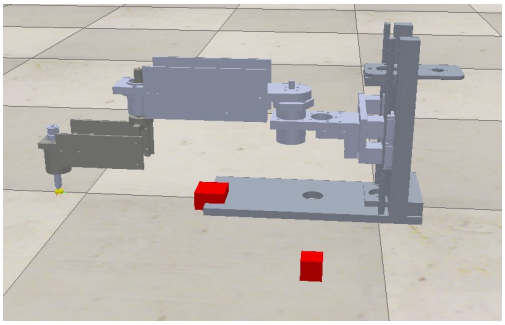
MATLAB and CoppeliSim program were cross-verified which validated the inverse kinemtaic equations for the SCARA robot.

5.6 SCARA path planning

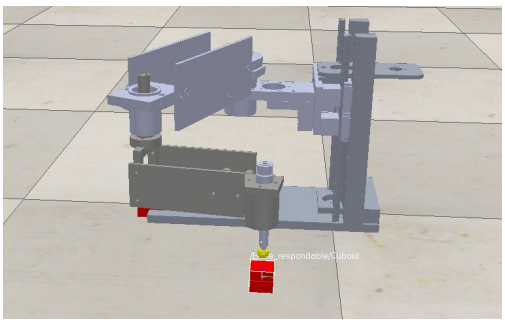
The comprehensive trajectory analysis of the SCARA robot was conducted in CoppeliaSim through the implementation of a non-threaded Lua code. The Lua script was designed to control the entire trajectory of the simulated SCARA robot. To define the trajectory, a cube object was strategically placed within the robot's work envelope, and its coordinates were provided as input to the Lua code. Additionally, a target position was determined by placing another cube in the work envelope, and its coordinates were specified as the goal for the robot. The Lua code utilized kinematic equations to calculate and execute a complete trajectory for the SCARA robot, ensuring it accurately followed the defined path from the initial position to the target. This simulation allowed for a thorough analysis of the robot's movement and provided valuable insights into its trajectory planning capabilities within the specified workspace.

The complete trajectory was divided into the Lift off, Pick up, Drop off and Set down points, displaying the pick-and-place motion of the SCARA robot with predefined parameters on a linear surface mentioned in Figures 5.14a to 5.14f.

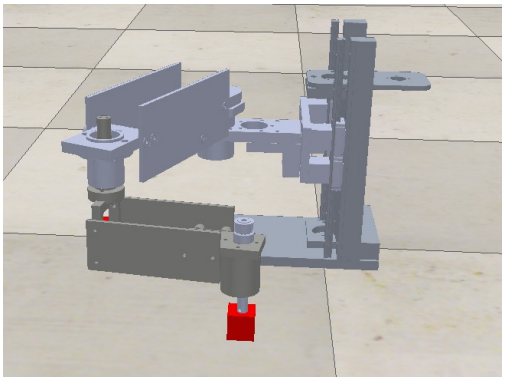
The simulation environment is configured to move the SCARA robot in order to pick up the cube from its initial position. The SCARA will move towards the cube and will orient its assembly in order to pick up the cube from its location. SCARA's tool tip will pick up the cube, in order to manipulate it to the desired location. The cube will be lifted off from its location in order to freely move in its workspace envelope. The cube is moved towards its desired drop off position, by manipulating each arm according to the required path. In the end, the cube is set down and placed at the desired location where the simulation will terminate.



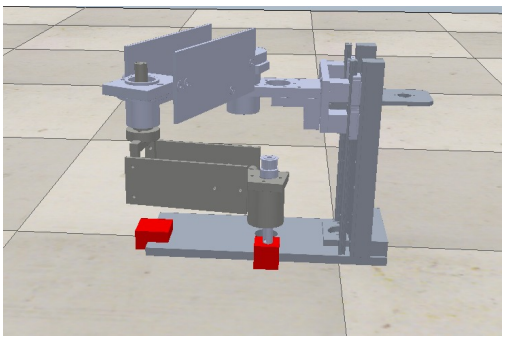
(a) Initial position of SCARA and cube



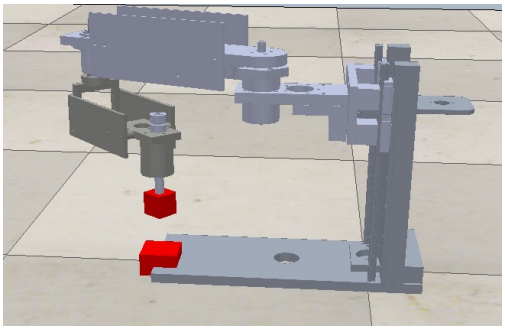
(b) Lift Off 1 position



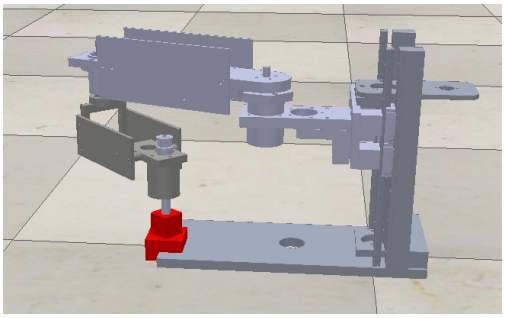
(c) Pick Up position



(d) Lifts off the cube



(e) Drop Off position



(f) Set down position

Figure 5.14: Pick and place of cube on a planar surface

5.7 Intelligent Sorting Mechanism

In order to develop an intelligent mechanism to sort different shapes of different sizes in their respective inverses, while correcting the orientation as well, CoppeliaSim simulation environment was integrated with an external Python Script using remote API server. OpenCV Python Library was used for required image processing task. The image taken from the vision sensor was processed in the Python script to detect objects, their sizes, coordinates and orientations [24]. The detected values were sent back to CoppeliaSim for corresponding actuation.

The work flow diagrams for both the CoppeliaSim and Pyhton environment are represented in Figure ?? and Figure ??.

5.7.1 Integrating CoppeliaSim with Python

In order to merge the CoppeliaSim work environment with Python script, a local server was created using Python script interpreted on Visual Studio Code . The remote API of this locally created server was accessed by CoppeliaSim by enabling the same remote server address in Lua script.

In order to capture the image of the workspace environment of the virtual environment a vision sensor was placed within this environment. This vision sensor was responsible for capturing the image sending it to python script using the remote API [25].

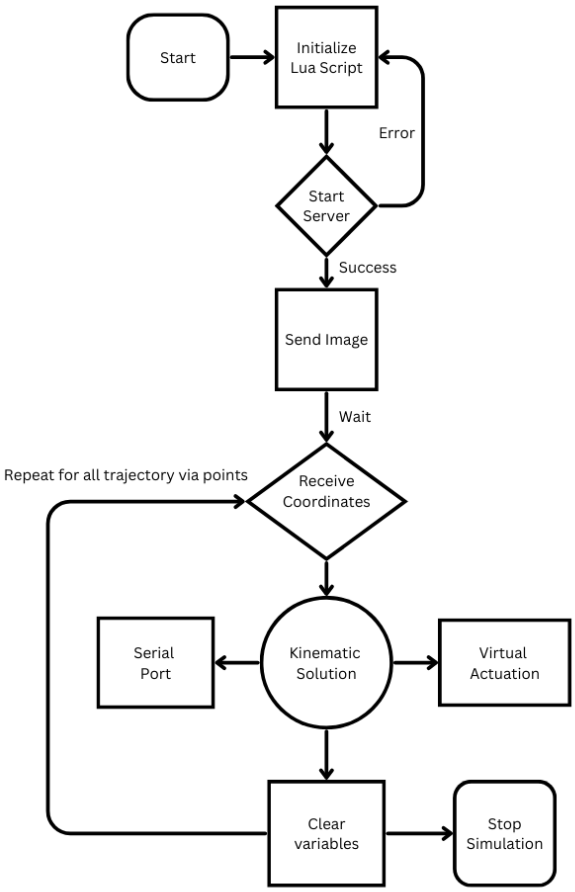


Figure 5.15: CoppeliaSim remote connection

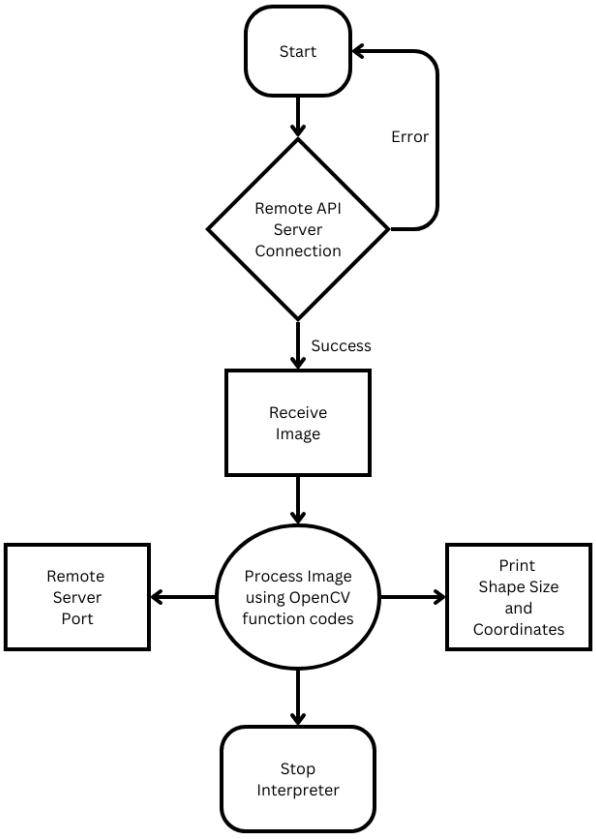


Figure 5.16: Python Remote Connection

5.8 Virtual Environment Intelligent Control Simulation

The Figure5.17 displays the CoppeliaSim environment of the SCAR robot with objects of different shapes of and sizes along with their inverses, behaving like structures of lock and key. The task is to pick these shapes and place them inside their counter-inverses with precision.

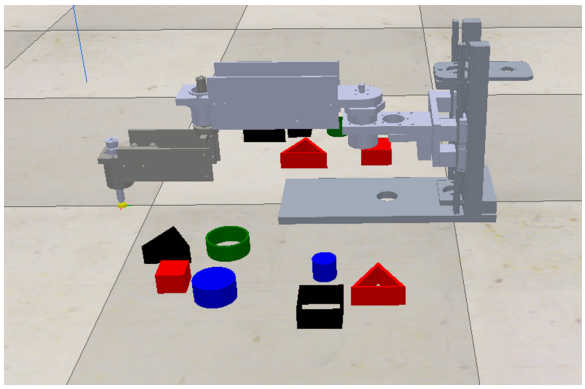


Figure 5.17: CoppeliaSim Environment

Once the simulation runs, the Remote API server is turned on for the given port number which is confirmed by the Python script once the connection is established, both the CoppeliaSim and Python Script softwares display an acknowledgment of succesful connection as mentioned in Figures 5.18 and 5.19.

```
[/Base_respondable@childScript:info] Remote API server started on port 19994
```

Figure 5.18: Coppeliasim turning the server on

```
PS C:\Users\ADMIN> & "C:/Program Files/Python312/python.exe" c:/Users/ADMIN/OneDrive/Desktop/ScaraKinematics/pycvscara/cvtest.py
Connection successful.
```

Figure 5.19: Connection established between the scripts

The Python script processes the image by detecting each shape, along with their sizes, coordinates, and orientations. The detected values are then normalized, printed, displayed, and sent back to CoppeliaSim for the corresponding actuation. Figures 5.20, 5.21 and 5.22 illustrate the detection and display of these values.

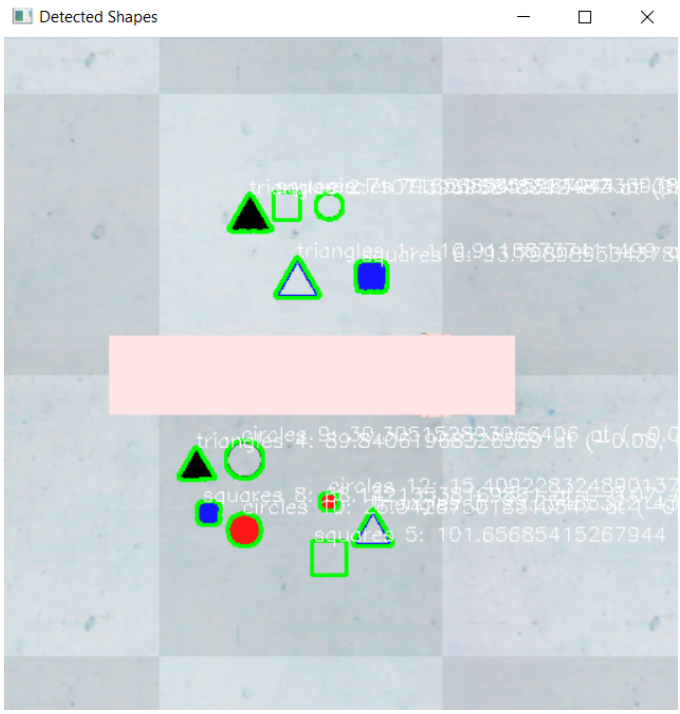


Figure 5.20: Detection of shapes, sizes, and coordinates

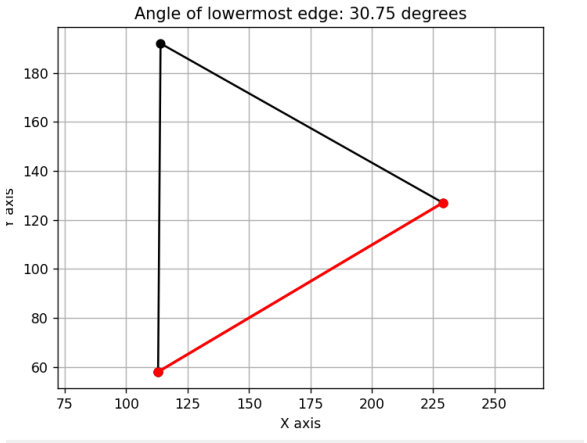


Figure 5.21: Detection of orientation

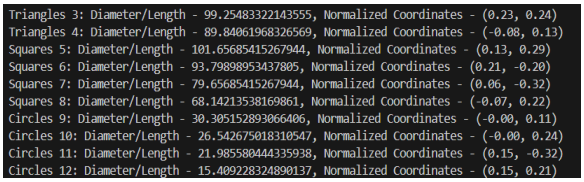


Figure 5.22: Printing the detected values

Once receiving the detected values, the Lua script in CoppeliaSim applies the inverse kinematics and trajectory function to sort the shapes in the inverses. Figures 5.23, 5.24 and 5.25 display how the shapes are placed in their inverses while also correcting the orientation where necessary.

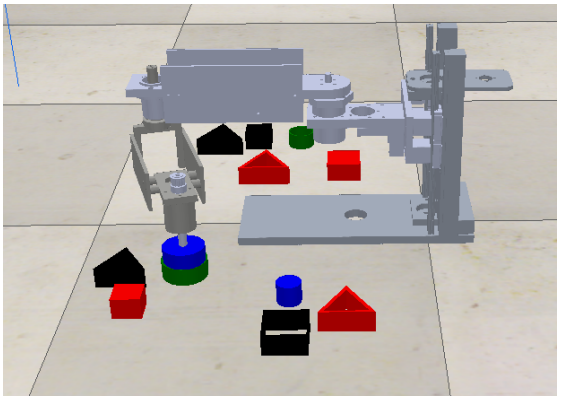


Figure 5.23: Placing larger circle in its respective inverse

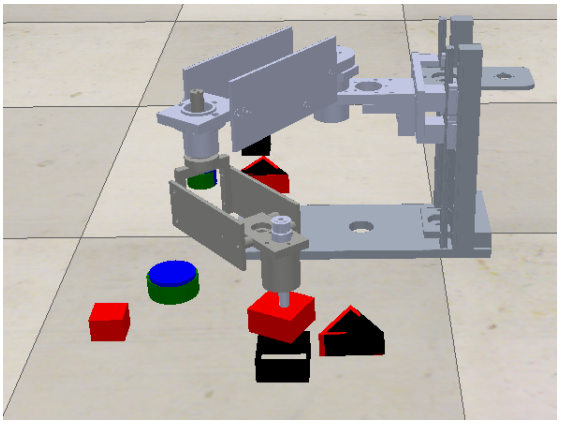


Figure 5.24: Lift off point of cuboid before correcting the orientation

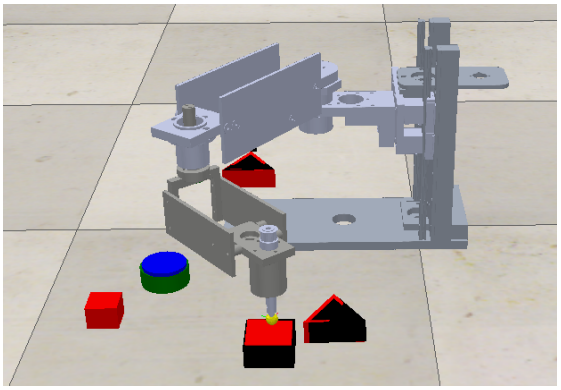


Figure 5.25: Placing the cuboid after correcting the orientation

Simulatnaeously in the CoppeliaSim environment, joint angles computed through kinematic modeling were transmitted to the serial port 'COM7' visible in Figure 5.26. This transmission allowed the data to be sent from CoppeliaSim to the master micro-controller.

```
q1: 5.000000, q2: -28.000000, q3: -68.000000, q4: 0.000000
Sent
Received:     q1: 5.000000, q2: -28.000000, q3: -68.000000, q4: 0.000000
Q1: 5.000000, Q2: 0.000000, Q3: -107.000000, Q4: 0.000000
Sent
Received: Q1: 5.000000, Q2: 0.000000, Q3: -107.000000, Q4: 0.000000
```

Figure 5.26: Data Sent and Received through CoppeliaSim

Upon receiving the data on 'COM7', it was further analyzed using a serial port monitor tool shown in Figure 5.27. This tool facilitated the inspection and evaluation of both the transmitted and received data. The comparison between

the sent joint angles and the received data was crucial for verifying the accuracy and integrity of the communication process.

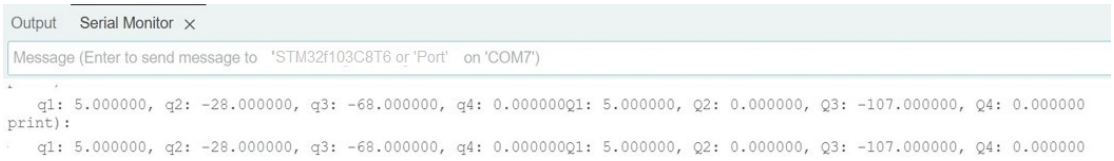


Figure 5.27: Data Sent and Received through Serial Port

5.9 Chapter Summary

This chapter explores the comprehensive virtual modeling and analysis of the SCARA robot using various software tools and environments.By exploring these software simulations and their applications, we get a comprehensive overview of how virtual modeling and analysis contribute to the development and optimization of control system of the SCARA robot.

Chapter 6

Control System Design

The chapter explains the functionality of the embedded control system in terms of communication and control mechanism. First, we develop the architectural design for the hardware connectivity and software program of our project. Followed by the design implementation concurrent to the Modbus communication protocol with the Master-Slave Network Architecture.

6.1 Hardware architecture

The complete hardware architecture of our project, can be shown in Figure 6.1. The user sends the operational coordinate frames which are either pre-defined or intelligently computed through virtual environment interfacing. The master controller is connected to the Tele-operating station via serial port, the master interprets the desired trajectory and broadcasts the individual joint motion parameters to the slaves using Modbus communication protocol. The slaves acknowledge the received signal and performs the required processing for generating a PWM signal as per the requirement of motion for each joint. The motor driver, operates the motors using the input PWM and from the generated torque the joint moves in a controlled operation. After the completion of task, slave raises a flag which is sent back to the master indicating successful trajectory completion.

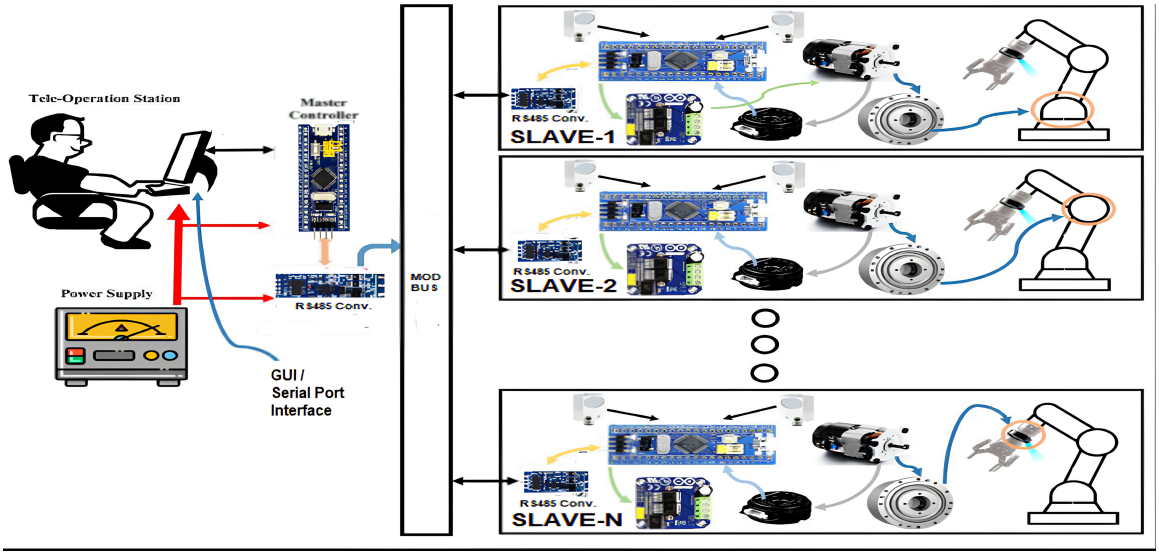


Figure 6.1: Hardware architecture

6.1.1 Pulse width modulation (PWM)

The pulse width modulation (PWM) technique is used to control the speed of the motor. It computes the percentage of motor on and off position without explicitly altering the supply voltages [5]. The respective Blue Pill micro-controller for each joint was configured for PWM generation, using the following TIMER parameters in CubeMX software.

1. Setting up the Prescalar.
2. Configuring the auto reload register (ARR) or counter period.
3. Enabling Capture Compare mode.
4. Frequency Computation for PWM.
5. Duty cycle computation.

The Prescaler helps in finding the desired frequency for the PWM generation. The timers in Blue Pill are connected with APB2 bus so prescalar will helps in step down the required PWM frequency as given by the following equation 6.1.

$$\text{Desired Frequency} = \frac{\text{TIMER CLOCK FREQ}}{Prescalar} \tag{6.1}$$

We set the counter period value so that ensures maximum number of pulses that micro controller will produce in one duty cycle using equation 6.2. Simultaneously the capture compare register (CCR) will ensure the switch states in one counter period.

$$\text{Duty cycle} = \frac{CCR}{ARR \times 100} \tag{6.2}$$

6.1.2 Hardware interfacing of generated PWM signals

In our design, we had used the two timers for each Blue Pill micro-controller in order to perfrom clock wise and anti-clock wise rotation respectively. Depending upon the command signal one will activate the PWM signals. These signals are verified through a Logic analyzer which ensures that the micro-controller generates desired pulses and then it these pulses are directed towards the motor through IBT-2 motor driver circuit. The timer pins are directly coupled with the motor drivers and are isolated from the 12V power supply.

The PWM signals generated from the micro-controller with varying Duty Cycles were analyzed using the Logic analyzer, the response of which is given in the Figures 6.2, 6.3, 6.4.

- **20% duty cycle PWM generation**

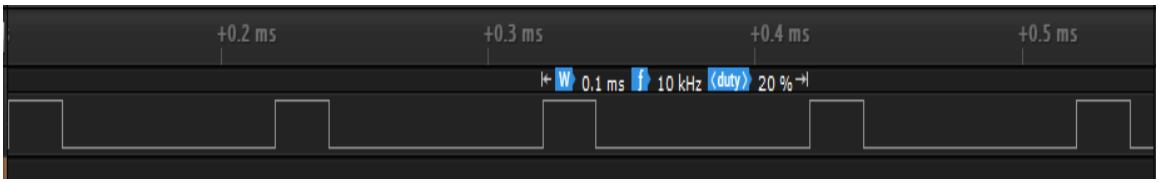


Figure 6.2: 10kHz input pulse with 20% duty cycle

- **50% duty cycle PWM generation**

Figure 6.3: 10kHz input pulse with 50% duty cycle

- **70% Duty Cycle PWM generation**

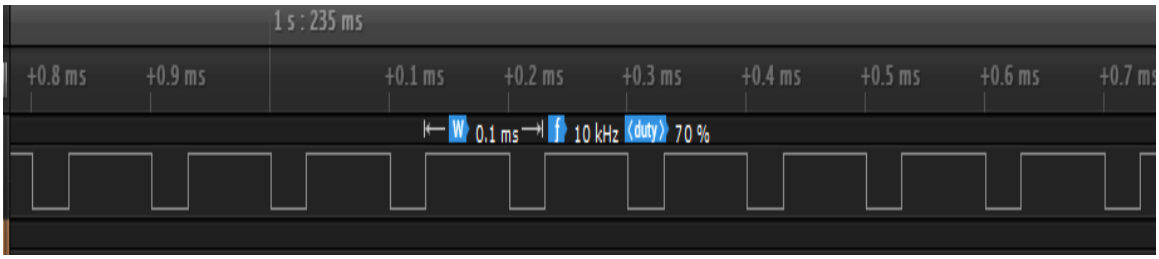


Figure 6.4: 10kHz input pulse with 70% duty cycle

6.1.3 Shaft optical encoders

In our project design we had used the incremental optical encoders with open collector output having 1000 lines/ rpm. This high precision with flexible geometrical configuration is suited best to provide the feedback response for the speed of brushed DC motors. With advance coupling these encoders won't damage the motors, even if they are moving with 100 percent duty cycle.

In Blue pill the Timer 1 is configured with the encoder mode with the counter period of 65535 and in the code premises it is initialized with the middle value so that our counter 8 byte variable acknowledge the pulses allowing it to increment and decrement its value in real time.

6.2 Modbus RS485 communication network

Many intelligent devices like programmable logic controllers, remote terminal units and loop controllers have some sort of memory management system that stores programs, data, and backups. Now if a device is Modbus compliant it must have a portion of its memory dedicated to Modbus. This area is called a Modbus memory area. Modbus compliant memory is divided into four separate areas or sub areas. These are coils, inputs, inputs registers, and then holding registers..

6.2.1 Modbus unit ID

Modbus unit ID is the unique assigned identifier. In the Modbus network when device (Master) sends the message on the network every single device receives the message (Slave) more like broadcasting. If we want to address to some specific slave then we define the Modbus unit ID such that only specific Id acknowledge the query response. This concept is further elaborated in following Figure 6.5.

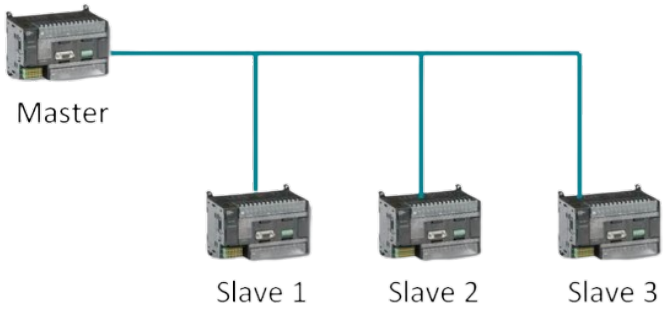


Figure 6.5: Assigning IDs to each device

In the following message, master nodes broadcast the message and subsequently, each slave nodes acknowledge that query message. Notably, only that slave node whose slave ID is identical to the master address generates the specific master-slave communication.

6.2.2 Significance of Modbus protocol

Modbus consists of differential pairs of wire meaning that the data is being transmitted using Differential Signaling method. It is based upon the principle of 'Data transmission for co-axial cable' which leads us to have the good signal to nose ratio (SNR). Leading to noise immunity and best retrieval of data for the long distance transmission.

6.2.3 Utilizing USART interfacing

As we are using the Blue Pill micro controller which explicitly doesn't have the inbuilt library for Modbus, for which we coupled the Modbus RS485 RTU module

mentioned in section 3.2.6 with the Universal Synchronous Asynchronous Receiver Transmitter (USART) protocol enabled pins of the Blue Pill micro-controller. The RS485 RTU module is configured in asynchronous mode with a baud rate of 115200. The asynchronous mode has been selected for our communication protocol as it is easily initialized using global interrupts.

6.2.4 Modbus messaging method

A. **Message structure**

This communication network is based on Master-Slave nodes architecture. The master prototype generates the master query and the slave upon acknowledgment generates the slave response as depicted in the following Figure 6.6.

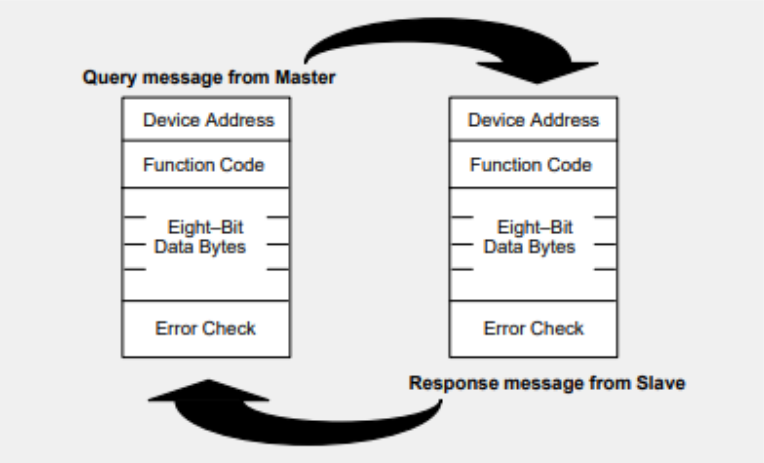


Figure 6.6: Message structuring

The attributes from the query message are described as:

(a) **Device address**

It is the first byte of the master query which consists of a slave identifier address such that the master can address the specific slave in the communication protocol.

(b) **Function code**

It is the second byte sent by the master which basically deals with which type of action needs to be performed by the slave nodes.

(c) **Eight byte data registers**

In this process, we specify the action to be taken by indicating the registers from which data needs to be accessed. This involves defining the starting and ending points for the registers. The data is then sequentially read or written across these registers. However, a limitation arises in the form of a selective filtering scheme. This means that we cannot skip registers during the ordering process, as all registers are addressed sequentially.

(d) **Cyclic Redundancy Check (CRC)**

Two bytes are initiated at the end of every Modbus message for error detection. Every byte in the message is used to calculate the CRC. The receiving device (slave) also calculates the CRC and compares it to the CRC from the sending device (master). If even one bit in the message is received incorrectly, an error indication will be sent back to the sending device.

6.2.5 Feedback response from the Slave

The master is responsible for generating a query with specific slave Unit ID, which is broadcasted over the Modbus communication network. The corresponding slave acknowledges the master data and performs the addressed action after which the slave status is sent back to the master. It consists of a slave identifier address as it refers to acknowledgment that confirms the master device referring to the slave is indeed the one it is addressing. It ensures a clear link between the master and the specific slave it is communicating with.

6.2.6 Master query formulation

The master micro-controller generates queries for each slave and the slaves perfrom the action demanded in each query after acknowledging the message from the master sent over by the communication protocol. The queries for each slave are listed in the Tables 6.1 to 6.3.

Table 6.1: Data transmission table for slave 1

Index	**Description**	**Data**
0	Slave ID	0
1	Home Position	Q1_home
2	Lift-Off Position	Q1_lift_off
3	Pick-Up Position	Q1_pick_up
4	Lift-Off Position	Q1_lift_off
5	Set-Down Position	Q1_set_down
6	Place Position	Q1_place
7	Set-Down Position	Q1_set_down
8	Home Position	Q1_home

Table 6.2: Data transmission table for slave 2

Index	**Description**	**Data**
0	Slave ID	1
1	Home Position	Q2_home
2	Lift-Off Position	Q2_lift_off
3	Pick-Up Position	Q2_pick_up
4	Lift-Off Position	Q2_lift_off
5	Set-Down Position	Q2_set_down
6	Place Position	Q2_place
7	Set-Down Position	Q2_set_down
8	Home Position	Q2_home

Table 6.3: Buffer for slave 3 (joint 3)

Index	**Description**	**Data**
0	Slave ID	2
1	Home Position	Q3_home
2	Lift-Off Position	Q3_lift_off
3	Pick-Up Position	Q3_pick_up
4	Lift-Off Position	Q3_lift_off
5	Set-Down Position	Q3_set_down
6	Place Position	Q3_place
7	Set-Down Position	Q3_set_down
8	Home Position	Q3_home

Table 6.4: Buffer for tool-tip motors (slave 4)

Index	**Description**	**Data**
0	Slave ID	3
1	Sucker Motor On	Sucker_on
2	Sucker Motor Off	Sucker_off
3	Orientation Motor Start	Orientation_start
4	Orientation Motor Stop	Orientation_stop

6.2.7 Operational stages

The Operational stages of the complete communication between the master and slave queries can easily be interpreted from the following Table 6.5.

Table 6.5: Operational stag description

Stage	Buffer Data	Operations
Initialization	Home Position	Initialize buffers with home position counts.
Buffer Formation	Buffers Creation	Initialize and store counts in buffers.
Transmission	Buffers to Slaves	Use threading to send data.
Slave 1	buff1 (Joint 1)	Transmission: [0, Q1_home, Q1_lift-off, Q1_pick-up, Q1_set-down, Q1_place, Q1_home]
Slave 2	buff2 (Joint 2)	Transmission: [1, Q2_home, Q2_lift-off, Q2_pick-up, Q2_set-down, Q2_place, Q2_home]
Slave 3	buff3 (Joint 3)	Transmission: [2, Q3_home, Q3_lift-off, Q3_pick-up, Q3_set-down, Q3_place, Q3_home]
Tool-Tip	buff4 (Tool Tip Motors)	Transmission: [3, Sucker_on, Sucker_off, Orientation_start, Orientation_stop]

6.2.8 Motor control in feedback loop

The desired position is compared with the actual position feedback from the encoder. This gives the desired speed, which is compared with the actual speed obtained as a feedback. This gives the desired PWM signal which is adjusted by the inner loop of PID controller. A control signal is generated which along with supply voltage from supply system is given to the motor by the motor driver as input.

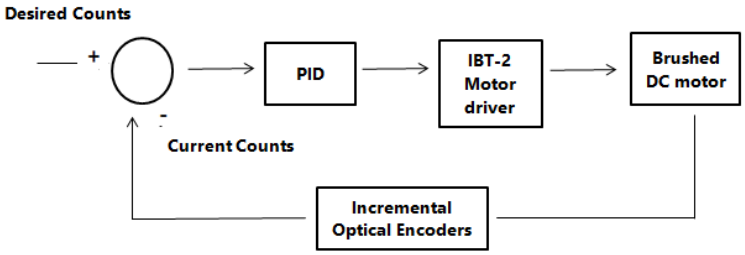


Figure 6.7: Feedback control mechanism

After obtaining the required angle, the data was pre-processed by the respective controller which was quantized for each joint actuation occurring synchronously, and for each actuation the PID controller is being called frequently. Every time the current counter value is being compared with the actual count rate and an error is being computed which drives the control signal respectively the duty cycle varied against control signal such that the capture compare register of PWM decreases synchronously providing us the smooth trajectory motion for each joint by controlling the respective motor.

6.2.9 PID controller

A proportional– integral– derivative controller (PID controller) is widely used in industrial control systems. It is a generic feedback control loop mechanism and used as feedback controller. PID loop defines how much energy is to be fed to the motor at any instant during a move. This is based on where the motor is and where it was expected to be. There are four parts in the equation and determines this load in which three main components are referred to as the P, I, D and the minor friction component is referred to as K.

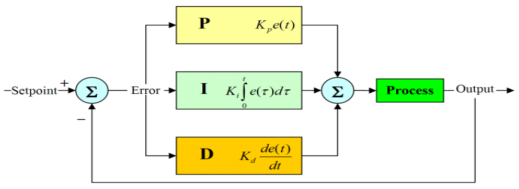


Figure 6.8: Error reduction using PID

In the Figure6.8, **P** determines the reaction to current error, **I** determine reaction to the sum of recently appeared errors, and **D** determines reaction according to the rate off error changing. The sum of all three parts contribute the control mechanism such as speed control of a motor in which **P** value depends upon current error, **I** on the accumulation of previous error and **D** predict future error based on the current rate of change.
The tuning of PID was performed manually by emulating the parameters for nullifying the jerk in the motion of joints and reducing the speed of the motor when the desired position is about to be reached, fed back by the encoders.

6.3 Master-Slave programming

The comprehensive program for the master and slave controllers was generated using the CubeMx and keil IDE. These programs were debugged and flashed on the individual controllers using ST-LINK/V2 debugger.

6.3.1 Master control program

The master code is responsible for receiving user input for the pick-up and place positions, calculating the corresponding lift-off and set-down points, converting these Cartesian coordinates into joint angles, and then transmitting these angles as counts to the slave controllers.

6.3.1.1 Trajectory points

- Home Position: (X0, Y0, Z0)
- Lift-off Point: (X1, Y1, Z1+δ)
- Pick-up Point: (X1, Y1, Z1)
- Set-down Point: (X2, Y2, Z2+δ)
- Place Point: (X2, Y2, Z2)

6.3.1.2 Conversion process

The Cartesian coordinates (X, Y, Z) are converted into joint angles (Q1, Q2, Q3) and joint angles are converted into counts, accounting for the resolution of the optical encoders.

6.3.1.3 Buffer formation

Three buffers (buff1, buff2, buff3) store the counts for each joint angle at different trajectory points. These buffers include the Modbus slave ID as the first byte to address specific slaves. The data is transmitted to slaves using threading for simultaneous communication.

6.3.2 Slave control program

Each slave operates in an interrupt mode, continuously checking for incoming data. Upon receiving data, the slave verifies the Modbus ID and processes the counts to control the motors' positions.

6.3.2.1 Control process

- **Interrupt handling:** The slave acknowledges data by comparing its ID with the incoming data's first byte.
- **Error calculation:** Desired counts (trajectory points) are compared with current counts (obtained from encoders) to calculate the error.
- **PID control:** The error is used in a PID controller to generate a control signal.
- **PWM signal:** The control signal adjusts the PWM duty cycle to drive the motor until the desired position is reached.

6.3.2.2 Switch cases

The slave uses switch statements to handle different trajectory points, ensuring synchronous execution of movements.

6.3.2.3 Tool tip configuration

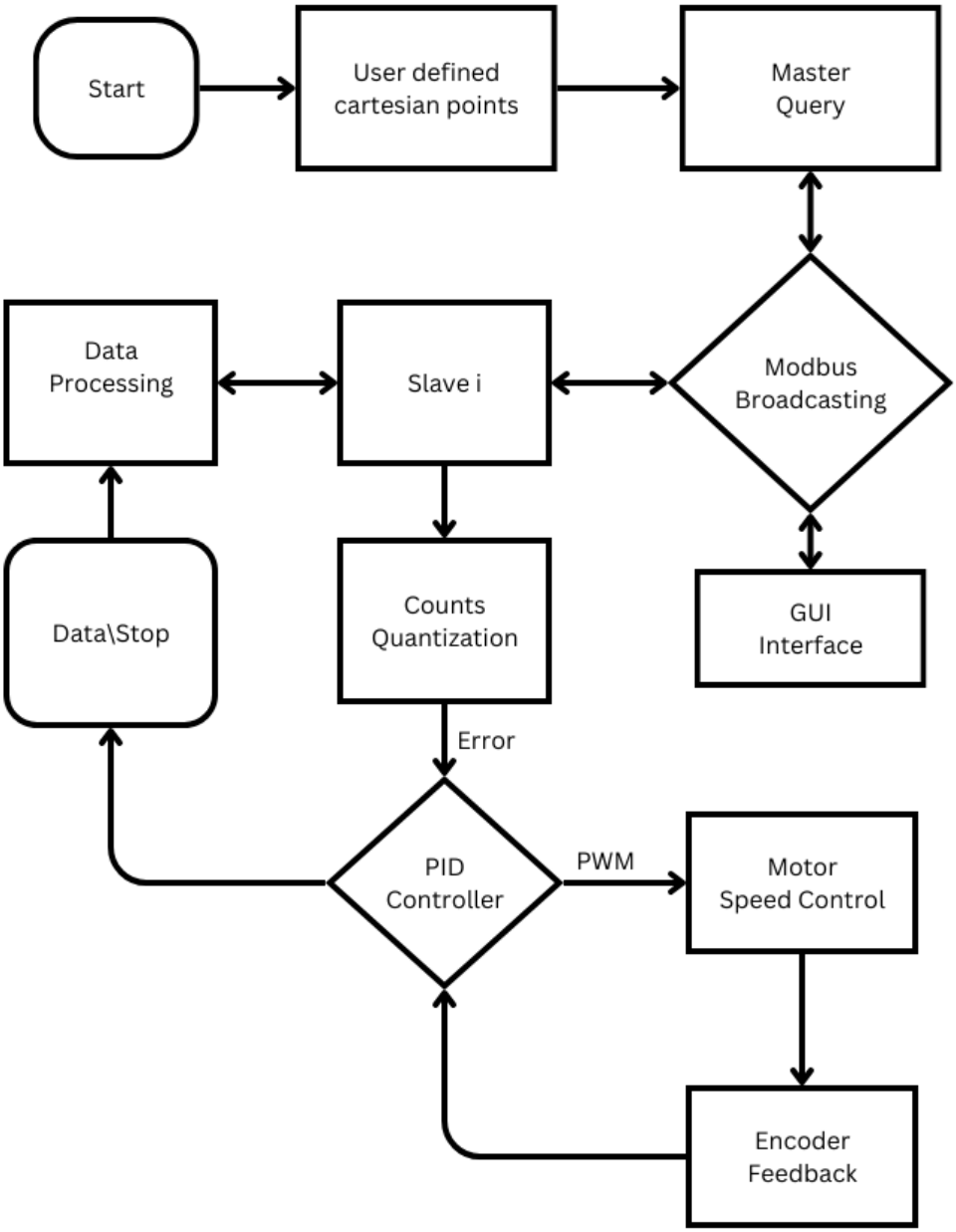
The tool-tip includes a 12V sucker motor for IC handling and an orientation motor. The sucker motor is controlled via a relay module, while the orientation motor operates with a fixed duty cycle.

6.3.2.4 Tool-Tip operation

- **Pick-Up:** The sucker motor is activated to grip the IC.
- **Place:** The orientation motor adjusts the IC's orientation, and the sucker motor releases the IC.

6.3.3 Program execution

The process is repeated for each slave. After system initialization, the master query is sent to the slaves through Modbus broadcasting, the master signal is processed by the slave controller. Counts are qunatized for moving the joint, which are given as input to the PID controller, responsible for deciding the PWM signal duty cycle from the timer pins. The PWM signals are then supplied to the motor driver for operating the motor. The encoders placed over the motors provide feedback to the pid controller, from where the error margin is calculated in order to obtain the desired speed. After, the completion of task, a completion flag is generated which is transmitted back to the master. This process is repeated by every slave controller until successful completion of complete robot trajectory.



Master-Slave program architecture

6.3.4 Pseudo code

```
start procedure
// Master Code
void main() {
    // User inputs for pick-up and place positions
    (X1, Y1, Z1) = getUserInput();
    (X2, Y2, Z2) = getUserInput();
```

```
    // Compute lift-off and set-down points
    (X1, Y1, Z1+o) = computeLiftOff(X1, Y1, Z1);
    (X2, Y2, Z2+o) = computeSetDown(X2, Y2, Z2);
    // Convert to joint angles and counts
    Q1 = cartesianToJointAngles(X1, Y1, Z1);
    counts1 = angleToCounts(Q1);
    buff1 = createBuffer(counts1);
    // Transmit buffers to slaves
    transmitBuffers(buff1, slaveID1);
    transmitBuffers(buff2, slaveID2);
    transmitBuffers(buff3, slaveID3);
}
// Slave Code
void interruptHandler() {
    if (dataReceived()) {
        slaveID = readSlaveID();
        if (validateID(slaveID)) {
            processData();
            executeTrajectory();
        }
    }
}
void executeTrajectory() {
    switch (currentState) {
        case LIFT_OFF:
            desiredCounts = liftOffCounts;
            controlMotor(desiredCounts);
            break;
        case PICK_UP:
```

```
                desiredCounts = pickUpCounts;
                controlMotor(desiredCounts);
                activateSuckerMotor();
                break;
            // Other cases for different states
            case PLACE:
                desiredCounts = placeCounts;
                controlMotor(desiredCounts);
                deactivateSuckerMotor();
                break;
        }
    }
    void controlMotor(int desiredCounts) {
        int currentCounts = getCurrentCounts();
        int error = desiredCounts - currentCounts;
        int controlSignal = PIDController(error);
        setPWM(controlSignal);
    }
    end procedure
```

6.4 Prototype Deployment

After the control system architecture was successfully implemented and the electrical components were interfaced with the mechanical architecture of the SCARA robot. The system was tested to pick up and IC within the SCARA's workspace envelope and place it at another desired position which is clearly visible in the Figures 6.9 to 6.14, here first the SCARA is present at its home position and the IC is mentioned in the workspace envelope. The SCARA will move towards the IC and will pick it up, after which it will drop the IC at the desired location.

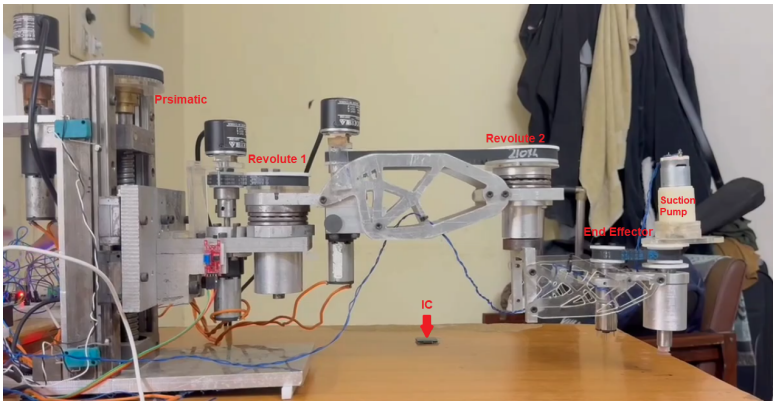


Figure 6.9: Home position of SCARA and IC

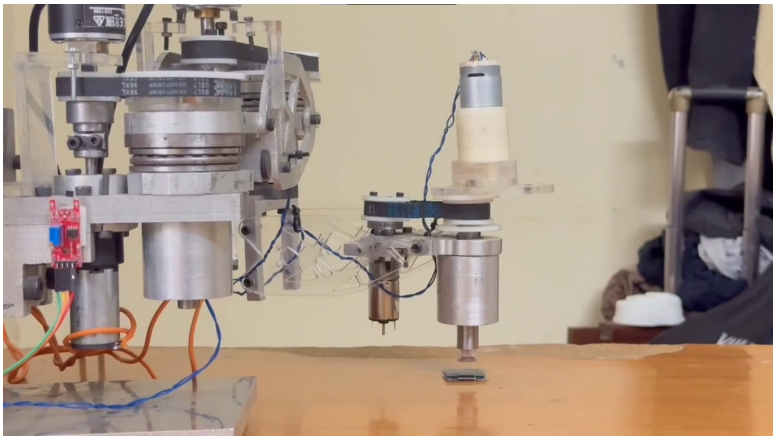


Figure 6.10: Lift off position of the SCARA

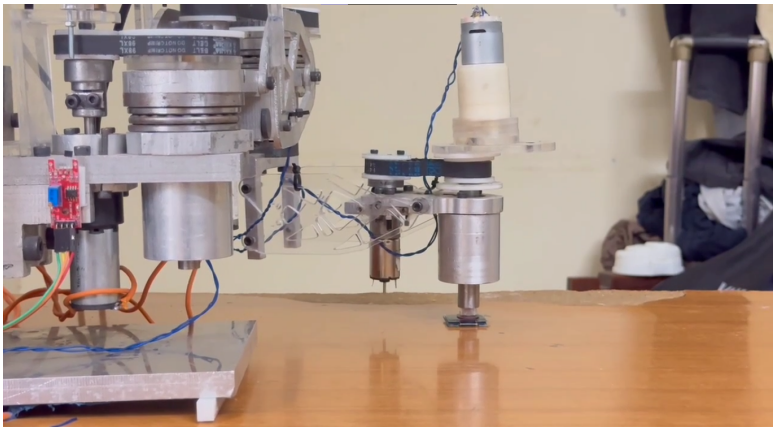


Figure 6.11: Set Down position of the SCARA

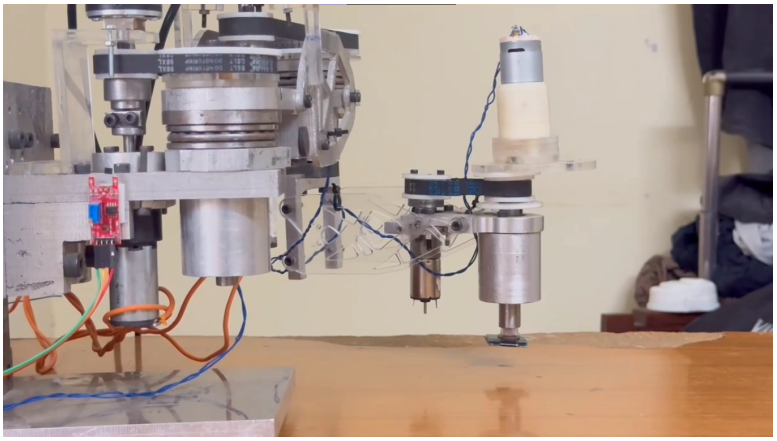


Figure 6.12: Picking up the IC

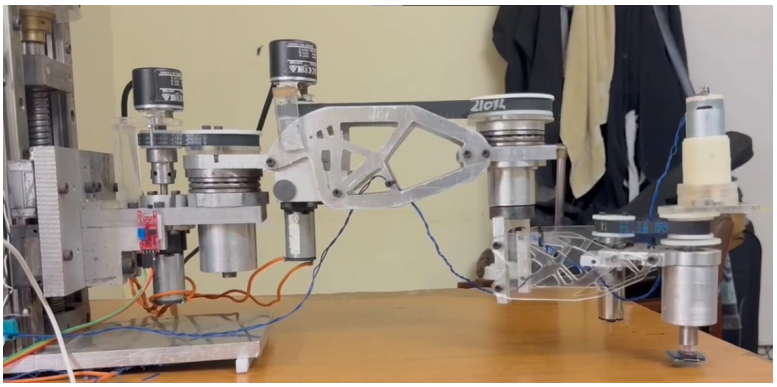


Figure 6.13: Moving the IC to drop off point

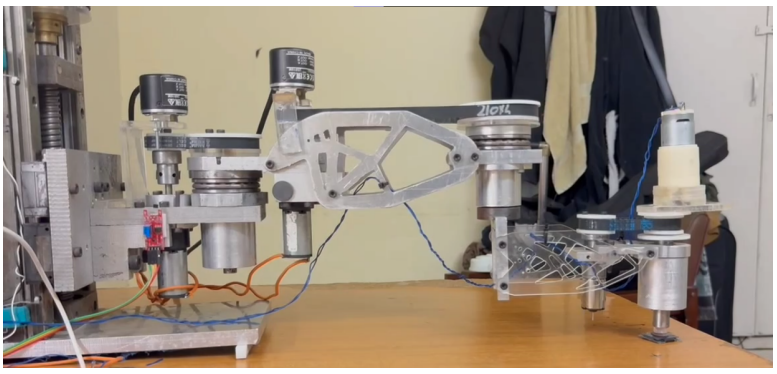


Figure 6.14: Dropping the IC to desired location

The control system was tested several times and the error margins were reduced by adjusting the PID controller and adjusting the duty cycles for the PWM generation for each motor. After successive trials the minimal error was achieved for the motion of the robot and to pick and place objects within its environment. The virtual environment was then integrated with the master micro-controller via

serial port, using which we applied a trajectory in the virtual environment and it was imitated by the real time control. The complete prototype can be seen in the Figure 6.15.

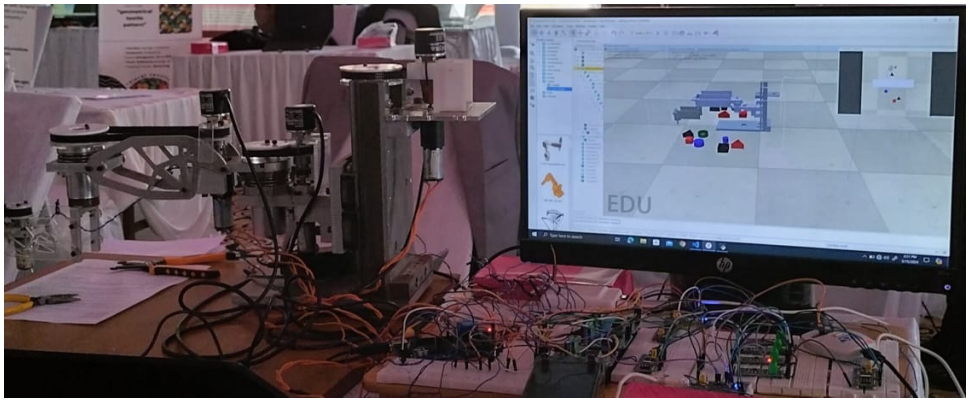


Figure 6.15: SCARA Teleoperated Control System Prototype

6.5 Chapter Summary

Following the hardware architecture, the chapter explores the complete control system of the SCARA robot. This includes the algorithms used for motion control, feedback mechanisms, and error correction to ensure precise movements. The communication protocol governing the robot's actions are explained, and the control loops designed for accuracy and stability are discussed.

The chapter also highlights the integration of the SCARA robot with a virtual environment intelligence-augmented control system. This integration enhances the robot's capabilities and allows for more advanced control strategies. Additionally, it is noted that the control system can be further improved through hardware layout testing and experimentation, which will help refine the system and enhance its performance.

Chapter 7

Conclusion and Future Recommendations

This thesis has successfully addressed the key objectives including the development of a control system was using a master-slave architecture with Modbus communication protocol, ensuring reliable and synchronized operation between the components. The PID controller was effectively implemented to maintain precise motor control, enhancing the robot's accuracy and efficiency. Furthermore, a virtual environment using CoppeliaSim was created to simulate the intelligent control of the SCARA robot, demonstrating its ability to perform tasks involving picking and placing objects with precision and allowing the users to effectively control the path of the robot using the intelligence-augmented virtual environment.

The results obtained from both the physical implementation and virtual simulations confirm that the developed system meets the intended design requirements and objectives. The system's ability to process and respond to real-time data ensures that the SCARA robot can perform the task of IC manipulation with high accuracy and reliability.

Building upon the success of this project, several enhancements and extensions can be pursued to further advance the capabilities of the SCARA robot and its control system.

7.1 Integration of camera modules

Incorporating camera modules into the SCARA robot's system can significantly enhance its capabilities. By capturing real-time images, the robot can utilize computer vision tools to identify and locate objects, monitor its surroundings, and adapt its actions accordingly. This integration can facilitate tasks such as object recognition, quality inspection, and more precise manipulation.

7.2 Teleoperation station with AI NLP and HMIs

To enhance the control system, it is recommended to develop an advanced teleoperation station integrating Human-Machine interface (HMIs) and natural language processing (NLP). This includes creating intuitive graphical interface, and incorporating augmented reality for immersive control. Voice recognition and conversational interfaces powered by AI NLP will allow for hands-free operation and efficient troubleshooting.

7.3 Scalability and modular design

Developing a modular design and exploring the usage of programmable logic controller (PLC) for the control system can facilitate scalability, allowing for the easy addition of new functionalities or the integration of additional robotic arms. This can make the system more adaptable to different tasks and environments. The SCARA can itself behave as a slave, centrally controlled in an industry where multiple robots are in operation.

By pursuing these recommendations, the SCARA robot's control system can be further refined and enhanced, paving the way for more advanced, intelligent, and versatile robotic applications in various industrial settings.

References

[1] I. O. for Standardization, *ISO 8373:2012: Manipulation robots and associated equipment – Vocabulary*. 2012.

[2] J. F. Engelberger, *Future capabilities*, pp. 117–138. Boston, MA: Springer US, 1980.

[3] S. Y. Nof, *Robot manipulators: Mechanics, modeling, and control*. Springer Science & Business Media, 2009.

[4] R. A. Brooks, "Robots that build other robots," *Scientific American*, vol. 260, no. 4, pp. 120–125, 1989.

[5] W. W. Melek, "Me 547: Robot manipulators: Kinematics, dynamics, and control," 2010.

[6] I. Mitsubishi Electric Automation, "Scara image." 500 Corporate Woods Pkwy - Vernon Hills, IL - 60061 - US - www.meau.com.

[7] I. of Electrical and E. Engineers, *IEEE Std 100-1990: Standard Definitions of Terms for Programmable Electronic Systems*. IEEE-USA, 1990.

[8] I. O. for Standardization, *ISO/IEC 11593:1995: Information technology – Programmable logic controllers – Part 1: General requirements and test methods*. 1995.

[9] X. Qing, "The master-slave control system design and implementation by serial communication," 01 2009.

[10] M. Organization, "About modbus," 2023.

[11] T.-S. Nguyen and T.-H. Huynh, "Design and implementation of modbus slave based on arm platform and freertos environment," *2015 International Conference on Advanced Technologies for Communications (ATC)*, pp. 462–467, 2015.

[12] T. B. Sheridan, *Telerobotics and telepresence in virtual worlds.* MIT Press, 1992.

[13] P. Corke, *Robotics and control: An introduction.* Springer International Publishing, 2017.

[14] J. J. Craig, *Introduction to robotics: Mechanics and control.* Pearson Education, 3rd ed., 2013.

[15] S. S., *Introduction to robot autonomy and perception.* Springer International Publishing, 2018.

[16] M. P. Groover, *Fundamentals of Modern Manufacturing: Materials, Processes, and Systems.* Hoboken, NJ: John Wiley & Sons, Inc., 5th ed., 2014.

[17] OMRON Industrial Automation, "E6h-cwz6c 1000p/r 0.5m," 2024. Accessed: 2024-05-30.

[18] R. Hessmer, "Ibt-2 h-bridge with arduino," 2013. Accessed: 2024-05-31.

[19] STMicroelectronics, *STM32 Blue Pill Microcontroller User Guide*, n.d.

[20] M. P. Systems, "Hall effect sensors: A comprehensive guide," 2020.

[21] S. PLC, "Limit switch explained," 2021.

[22] M. Soyaslan, M. Uk, F. S. A. Shah, and O. Eldogan, "Modelling, control and simulation of a scara prr-type robot manipulator," *Scientia Iranica*, vol. 27, pp. 330–340, 2020.

[23] J. J. Craig, *Introduction to robotics: Mechanics and control*. Pearson Education, 3rd ed., 2013.

[24] S. Mehtab and J. Sen, "Object detection and tracking using opencv in python," 03 2020.

[25] Postman, "How to build an api in python," 2023.

About the Authors

Muhammad Faiez Ali was born on October 21, 2002 in Faisalabad, Punjab Province, Pakistan. He did his matriculation from Divisional Public School, Faisalabad in 2018 and his F.Sc - Pre-Engineering from Punjab College, Faisalabad in 2020 securing high merits. The author joined the Pakistan Institute of Engineering & Applied Sciences, Islamabad as Bachelors student of Electrical Engineering. His areas of interest are Embedded Systems, Robotics & Automation and Internet of Things (IoT).

Muhammad Maaz Khan was born on October 30, 2002 in Karachi, Sindh Province, Pakistan. He did his matriculation and F.Sc - Pre-Engineering from Mari Petroleum Higher Secondary School securing high merits. The author joined the Pakistan Institute of Engineering & Applied Sciences, Islamabad as Bachelors student of Electrical Engineering. His areas of interest are Embedded Control Systems, Robotics & Automation and Communication Systems.

Munad E Ali was born on July 18, 2002 in Faisalabad, Punjab Province, Pakistan. He did his O-Levels and O-Levels from Divisional Model College, Faisalabad securing high merits. The author joined the Pakistan Institute of Engineering & Applied Sciences, Islamabad as Bachelors student of Electrical Engineering. His areas of interest are Embedded Control Systems, Robotics & Automation and Computational Intelligence in Engineering Applications.