

Design and Development of IoT based Harvesting Robo-Vec



Session: BE (EEP). Fall 2023

Project Supervisor: Engr. Sadiq Ur Rehman

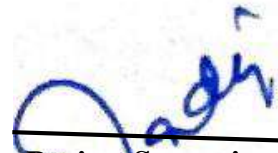
Submitted By

Muhammad Ali Lari

972-2019

Certification

This is to certify that Muhammad Ali Lari 972-2019 have successfully completed the final project **Design and Development of IoT based Harvesting Robo-Vec**, at the **Hamdard University**, to fulfill the partial requirement of the degree **B.E (Electrical)**.



Project Supervisor

Sadiq Ur Rehman

Assistant Professor

Chairman

Department of Electrical Engineering, Hamdard University

Project Title (Design and Development of IoT based Harvesting Robo-Vec)

Sustainable Development Goals

(Please tick the relevant SDG(s) linked with FYDP)

SDG No	Description of SDG	SDG No	Description of SDG
SDG 1	No Poverty	SDG 9	Industry, Innovation, and Infrastructure
SDG 2	Zero Hunger	SDG 10	Reduced Inequalities
SDG 3	Good Health and Well Being	SDG 11	Sustainable Cities and Communities
SDG 4	Quality Education	SDG 12	Responsible Consumption and Production
SDG 5	Gender Equality	SDG 13	Climate Change
SDG 6	Clean Water and Sanitation	SDG 14	Life Below Water
SDG 7	Affordable and Clean Energy	SDG 15	Life on Land
SDG 8	Decent Work and Economic Growth	SDG 16	Peace, Justice and Strong Institutions
		SDG 17	Partnerships for the Goals



Range of Complex Problem Solving		
	Attribute	Complex Problem
1	Range of conflicting requirements	Involve wide-ranging or conflicting technical, engineering and other issues.
2	Depth of analysis required	Have no obvious solution and require abstract thinking, originality in analysis to formulate suitable models.
3	Depth of knowledge required	Requires research-based knowledge much of which is at, or informed by, the forefront of the professional discipline and which allows a fundamentals-based, first principles analytical approach.
4	Familiarity of issues	Involve infrequently encountered issues
5	Extent of applicable codes	Are outside problems encompassed by standards and codes of practice for professional engineering.
6	Extent of stakeholder involvement and level of conflicting requirements	Involve diverse groups of stakeholders with widely varying needs.
7	Consequences	Have significant consequences in a range of contexts.
8	Interdependence	Are high level problems including many component parts or sub-problems
Range of Complex Problem Activities		
	Attribute	Complex Activities
1	Range of resources	Involve the use of diverse resources (and for this purpose, resources include people, money, equipment, materials, information and technologies).
2	Level of interaction	Require resolution of significant problems arising from interactions between wide ranging and conflicting technical, engineering or other issues.
3	Innovation	Involve creative use of engineering principles and research-based knowledge in novel ways.
4	Consequences to society and the environment	Have significant consequences in a range of contexts, characterized by difficulty of prediction and mitigation.
5	Familiarity	Can extend beyond previous experiences by applying principles-based approaches.

Abstract

This project report presents the design and development of an IoT-based harvesting robot, named "Harvesting Robo Vec," which aims to enhance efficiency and precision in the agricultural sector. The integration of IoT technology into traditional harvesting methods can significantly improve the harvesting process by automating various tasks and providing real-time monitoring and control.

The project focuses on the design and implementation of a versatile robot capable of autonomously navigating through crop fields, detecting and identifying ripe produce, and executing precise harvesting maneuvers. The robot incorporates advanced sensing and imaging technologies, such as computer vision algorithms and proximity sensors, to identify and locate the target crops accurately.

Furthermore, Harvesting Robo Vec is equipped with an IoT communication module, allowing seamless connectivity and data exchange with a centralized control system. This integration enables remote monitoring and control, enabling farmers to manage multiple robots simultaneously and optimize the harvesting operation based on real-time information.

The report describes the overall architecture and system components of Harvesting Robo Vec, including the mechanical structure, sensor setup, control algorithms, and communication infrastructure. The design considerations, including safety measures, power management, and robustness, are also discussed in detail.

The development process involves iterative design iterations, prototyping, and testing to refine the robot's performance and address potential challenges. The project utilizes modern software tools and programming languages for the development of control algorithms and IoT integration.

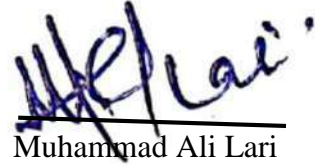
Experimental results demonstrate the effectiveness of Harvesting Robo Vec in autonomously harvesting crops, showcasing improved efficiency, reduced labor costs, and enhanced productivity compared to traditional manual methods. The report also highlights the potential benefits and future possibilities for the adoption of IoT-based harvesting robots in agriculture.

Overall, the project report aims to provide valuable insights into the design and development of an IoT-based harvesting robot and its potential to revolutionize the agricultural sector. The presented work contributes to the ongoing research and development efforts in the field of precision farming and autonomous robotics, paving the way for more sustainable and efficient agricultural practices.

Keywords: IoT, Prototype, Harvesting, Robot, Vehicle.

Undertaking

I certify that the project **Design and Development of IoT based Harvesting Robo-Vec** is our own work. The work has not, in whole or in part, been presented elsewhere for assessment. Where material has been used from other sources it has been properly acknowledged/ referred.

A handwritten signature in blue ink, appearing to read 'M. Ali Lari', written over a horizontal line.

Muhammad Ali Lari

(972-2019)

Acknowledgement

Acknowledgment is due to **Hamdard University** for support of this Project, a highly appreciated achievement for us at the undergraduate level.

We wish to express our appreciation to our **Engr. Sadiq Ur Rehman** served as our major advisor. We would like to express our heartiest gratitude for the guidance, sincere help, and friendly manner which inspires us to do well in the project and makes it a reality.

TABLE OF CONTENT

CERTIFICATE _____	Error! Bookmark not defined.
ABSTRACT _____	Error! Bookmark not defined.
ACKNOWLEDGEMENT _____	Error! Bookmark not defined.
TABLE OF CONTENT _____	i
LIST OF FIGURES _____	iv
LIST OF TABLES _____	v
ABBREVIATIONS _____	vi
CHAPTER 1 _____	1
INTRODUCTION _____	1
1.1 Motivation _____	2
1.2 Why “DESIGN and DEVELOPMENT of IoT BASED HARVESTING ROBO VEC”? _____	3
1.3 APPLICATIONS _____	4
1.4 Organization of Thesis _____	6
CHAPTER 2 _____	8
LITERATURE REVIEW _____	8
CHAPTER 3 _____	10
Methodology _____	10
3.1 Detail explanation of design and development process _____	10
3.1.1 Concept Development: _____	10
3.1.2 Hardware Design: _____	10
3.1.3 Software Design: _____	10
3.1.4 Prototyping: _____	11
3.1.5 Integration and System Testing: _____	11
3.1.6 Iterative Refinement: _____	11
3.2 Description of the mechanical structure, sensor setup and control algorithm _____	12
3.2.1 Mechanical Structure: _____	12
3.2.2 Sensor Setup: _____	12
3.2.3 Control Algorithm: _____	13
3.3 Explanation of the software tools and programming languages used _____	14
3.3.1 Python: _____	14
3.3.2 C/C++: _____	15
3.3.3 Arduino IDE: _____	15
3.3.4 OpenCV: _____	15
3.3.5 HTTP and RESTful APIs: _____	16
3.3.6 Git: _____	16
3.4 Flowchart _____	17
_____	18
3.5 Block Diagram _____	19
CHAPTER 4 _____	20
SYSTEM ARCHITECTURE _____	20
4.1 COMPONENTS OF THE HARVESTING ROBO VEC _____	20
4.1.1 ESP 32 [8] _____	21

4.1.2	RASPBERRY PI [9]	22
4.1.3	PI-CAMERA [10]	23
4.1.4	PI CAMERA RIBBON [11]	24
4.1.5	ACRYLIC SHEET [12]	25
4.1.6	DC MOTORS [13]	26
4.1.7	L&U BRACKET [14]	27
4.1.8	ROBOTIC ARM [15]	28
4.1.9	Li-Ion BATTERY [16]	29
4.1.10	BATTERY HOLDER [17]	30
4.1.11	3s BATTERY MANAGEMENT SYSTEM [18]	31
4.1.12	ULTRASONIC SENSOR [19]	32
4.1.13	ULTRASONIC HOLDER [20]	33
4.1.14	MOTOR DRIVER MODULE [21]	34
4.1.15	BUCK CONVERTER [22]	36
4.1.16	VERO-BOARD [23]	37
4.1.17	JUMPERWIRE [24]	38
4.1.18	Voltage Regulator IC [25]	39
4.1.19	CHARGING JACK [26]	40
4.1.20	Servo Motor	41
4.2	Overview of the overall system architecture of Harvesting Robo Vec	42
4.2.1	Raspberry Pi:	42
4.2.2	ESP32 Microcontroller:	42
4.2.3	Mechanical Structure:	42
4.2.4	Sensors:	42
4.2.5	Control Algorithms:	43
4.2.6	Communication:	43
4.2.7	Power and Energy Management:	43
4.3	Dimension of Harvesting Robo Vec	44
CHAPTER 5:		45
IMPLEMENTATION		45
5.1	Explanation of the iterative design process and prototyping	45
5.1.1	Initial Design and Conceptualization:	45
5.1.2	Prototype Development:	45
5.1.3	Testing and Evaluation:	46
5.1.4	Feedback Collection:	46
5.1.5	Refinement and Iteration:	46
5.1.6	Prototyping of Enhanced Designs:	46
5.1.7	Continuous Testing and Evaluation:	47
5.1.8	Integration and System-level Testing:	47
5.2	Discussion of challenges faced and solutions implemented	47
5.2.1	Tomato Detection Accuracy:	47
5.2.2	Obstacle Detection and Avoidance:	48
5.2.3	Robotic Arm Manipulation:	48
5.2.4	Power Management:	48
5.2.5	Connectivity and Communication Reliability:	49
5.2.6	Environmental Adaptability:	49
5.3	Description of the testing procedures and experimental setup	49
5.3.1	Testing Objectives:	50
5.3.2	Experimental Setup:	50
5.3.3	Testing Scenarios:	51
5.3.4	Data Collection and Analysis:	51
5.3.5	Iterative Refinement:	51
CHAPTER 6		52
RESULTS AND ANALYSIS		52

6.1	Results and Calculations	52
6.1.1	Tomato Detection Accuracy	52
6.1.2	For Obstacle Avoidance Performance:	54
6.1.3	For Robotic Arm Manipulation Efficiency:	55
6.2	Analysis and Future Improvements:	59
6.2.1	Tomato Detection Accuracy:	59
6.2.2	Obstacle Avoidance Performance:	59
6.2.3	Robotic Arm Manipulation Efficiency:	59
6.3	Comparison of Harvesting Robo Vec's performance with traditional manual methods [27]	60
6.4	Servo Synchronization	61
6.5	Experimental Tests with their ranges	62
6.6	Assembled Hardware	62
CHAPTER 7		67
CONCLUSION AND FUTURE WORK		67
7.1	Conclusion	67
7.2	Recommendation for Future Work	68
CHAPTER 8		69
CODING/PROGRAMMING		69
8.1	Raspberry Pi Coding (Master Module)	69
8.2	ESP-32 CODING	74
REFERENCES:		77

LIST OF FIGURES

Figure 1: Application of Harvesting Robo Vec	6
Figure 2: ESP-32 Module	21
Figure 3: Raspberry Pi Model 3B+ Module	22
Figure 4: Pi Camera Module	23
Figure 5: Pi cam Ribbon	24
Figure 6: Acrylic Sheet	25
Figure 7: DC Motor	26
Figure 8: L&U Bracket	27
Figure 9: Robotic Arm	28
Figure 10: Li-ion Batteries	29
Figure 11: Li-ion Battery Holder	30
Figure 12: 3s Battery Management System	31
Figure 13: Ultrasonic Sensor	33
Figure 14: Ultrasonic Holder	33
Figure 15: Motor Driver Module	35
Figure 16: Buck Converter	36
Figure 17: Vero Board	37
Figure 18: Jumper Wire	38
Figure 19: Jumper Wire	39
Figure 20: Jumper Wire	40
Figure 21: Servo Motor	41
Figure 22: Crouched	44
Figure 23: stretched	45
Figure 24: Tomato detection with other objects	54
Figure 25: Tomato detection	54
Figure 26: Grasping Tomato	56
Figure 27: Placing Tomato in Basket	57
Figure 28: Tomato Placed at predefined location (basket)	58
Figure 29: Assembled Hardware	62
Figure 30: Circuitry	63
Figure 31: Chassis Overview	64
Figure 32: Active Arm	64
Figure 33: Assembled Robotic Arm	65
Figure 34: Charging Port	65
Figure 35: Rest Position	66
Figure 36: Remote Control of Robo Vec	66

LIST OF TABLES

<i>Table 1: Literature Review</i>	9
<i>Table 2: Specifications of ESP-32</i>	21
<i>Table 3: Specifications of Raspberry PI</i>	22
<i>Table 4: Specifications of DC Motor</i>	26
<i>Table 5: Specifications of Robotic Arm</i>	28
<i>Table 6: Specifications of Li-ion Battery</i>	29
<i>Table 7: Specifications of Battery Management System</i>	31
<i>Table 8: Specifications of Ultrasonic Sensor</i>	33
<i>Table 9: Specifications of Motor Driver Module</i>	35
<i>Table 10: Specifications of Buck Converter</i>	36
<i>Table 11: Specifications of Voltage Regulator IC</i>	39
<i>Table 12: Specifications of Servo Motor</i>	41
<i>Table 13: Comparison of Harvesting Robo Vec's performance with traditional manual methods</i>	60
<i>Table 14: Servo Synchronization</i>	61
<i>Table 15: Experimental Tests with ranges</i>	62

ABBREVIATIONS

1. IoT - Internet of Things	2. TCP- Transmission Control Protocol
3. GPS - Global Positioning System	4. HTTP - Hypertext Transfer Protocol
5. AI - Artificial Intelligence	6. HTTPS - Hypertext Transfer Protocol Secure
7. LED - Light Emitting Diode	8. SMTP - Simple Mail Transfer Protocol
9. GUI - Graphical User Interface	10. SQL - Structured Query Language
11. CPU - Central Processing Unit	12. JSON - JavaScript Object Notation
13. RAM - Random Access Memory	14. IoT - Internet of Things
15. USB - Universal Serial Bus	16. GPS - Global Positioning System
17. VGA - Video Graphics Array	18. AI - Artificial Intelligence
19. HDMI - High-Definition Multimedia Interface	20. IP - Internet Protocol
21. OCR - Optical Character Recognition	22. XML - Extensible Markup Language
23. WLAN - Wireless Local Area Network	24. IDE - Integrated Development Environment
25. RFID - Radio-Frequency Identification	26. FPGA - Field-Programmable Gate Array
27. LAN - Local Area Network	28. GPIO - General Purpose Input Output
29. WAN - Wide Area Network	30. ADC - Analog-to-Digital Converter
31. API - Application Programming Interface	32. PWM - Pulse Width Modulation

CHAPTER 1

INTRODUCTION

The project emphasizes the significance of boosting the agricultural industry's productivity, efficiency, and sustainability as it contends with labour shortages, limitations on manual harvesting, and rising food demand. Harvesting Robo Vec, an IoT-based harvesting robot, was developed using IoT technology. It will challenge present practices and boost the efficiency of the agriculture industry as a whole.

The initiative is conscious of the restrictions on productivity and scalability that come with hand harvesting, as well as how labor-intensive it is. By merging IoT capabilities and building an intelligent, autonomous robot that can travel fields, identify ripe crops, and execute precise harvesting actions, the project seeks to overcome these difficulties. This cutting-edge tactic has the ability to shield crops from harm, boost output, and reduce the need for physical labour.

Harvesting Robo Vec now has real-time connectivity along with information communicate with a centralised control system as a result of the integration of IoT technology. The ability to remotely monitor and manage many robots at once is made possible by this connectivity for farmers and agricultural operators. Real-time data analysis can be used to make data-driven decisions that optimise resource allocation, reduce waste, and increase the quality of product.

This project's importance rests in its potential to transform the agriculture industry by introducing IoT-based harvesting robots. These robots have the ability to increase productivity, address the labour shortage, and increase harvesting accuracy. The initiative intends to help create a more sustainable and productive agriculture economy that can satisfy the expanding demands of a global population by leveraging automation and data-driven decision-making.

1.1 Motivation

The urgency with which the agricultural business must address issues including labour shortages, rising production costs, and the need for increased efficiency is what spurred the development of this project. Our goal is to modernise traditional harvesting techniques and increase the overall productivity and sustainability of the agriculture industry by leveraging the power of IoT technology and building an IoT-based harvesting robot. Automating the harvesting process could lead to improved agricultural output, decreased reliance on manual labour, increased precision, and real-time, data-driven decision-making.

Problem Statement

Agriculture uses traditional manual harvesting methods that are labor-intensive, time-consuming, and prone to human error. These approaches struggle to keep up with the rising needs of a growing world population. Furthermore, the issue is made worse by the severe difficulties caused by a lack of manpower in many areas. As a result, there is an urgent need for creative solutions that can automate and optimize the harvesting process, resulting in increased productivity, cost savings, and better crop quality.

1.2 Why “DESIGN and DEVELOPMENT of IoT BASED HARVESTING ROBO VEC”?

The selection of this project, "Design and Development of IoT-based Harvesting Robo Vec," was driven by several factors that highlighted its significance and potential impact. The reasons for selecting this project include:

1. **Technological Advancements:** The project provided a chance to investigate and apply cutting-edge technologies, like IoT, computer vision, and robotics, to tackle practical issues in the agriculture sector. Traditional hand harvesting techniques could be revolutionized by the incorporation of modern technology, which would also increase productivity.
2. **Labor Intensive Nature of Harvesting:** Tomato harvesting is a labor-intensive and time-consuming process, just like many other agricultural jobs. The project's goal was to automate the procedure in order to boost productivity, lessen the physical burden on agricultural workers, and decrease the reliance on human labour.
3. **Agricultural Industry Modernization: Modernization of the Agricultural Industry:** The project was in line with the objective of modernising the agricultural industry by implementing creative solutions. The project's goal was to increase overall agricultural efficiency and promote the adoption of smart farming practises by adopting an IoT-based robotic system for picking tomatoes.
4. **Food Security and Sustainability:** The project's emphasis on increasing tomato harvesting effectiveness had effects on sustainability and food security. The project aims to improve sustainable agriculture practises and increase the availability of fresh fruit by increasing productivity and lowering post-harvest losses.
5. **Practical Relevance: Practical Importance:** Harvesting tomatoes is a common agricultural operation with significant economic value. The initiative sought to offer meaningful and useful solutions that could be quickly implemented in real-world situations by addressing the unique difficulties related with tomato picking.

6. **Learning and Skill Development:** In the areas of robotics, IoT, image processing, and automation, the project provided valuable educational possibilities. It offered a forum for developing and refining technical abilities, encouraging invention, and gaining practical knowledge of creating intricate systems.
7. **Research and Development Potential:** The project provided opportunities for research and development with the potential to advance intelligent automation, precision farming, and autonomous agricultural systems. It provided the framework for upcoming research and technological advancements in the field of agricultural robots.

1.3 APPLICATIONS

➤ Agriculture.	➤ Manufacturing of things.
➤ Customer services.	➤ Utilized in security.
➤ Food preparation.	➤ Pharmaceuticals.
➤ Military applications	➤ Warehousing.

Design and Development of IoT based Harvesting Robo-Vec



Figure 1: Application of Harvesting Robo Vec

1.4 Organization of Thesis

This thesis is divided into numerous sections to provide a thorough overview of the design and development of the IoT-based harvesting robot, Harvesting Robo Vec, and its implications for the agricultural industry.

Chapter 1: *Introduction* The project's history, goals, and relevance are explained in this first chapter. The reasons for creating an IoT-based harvesting robot are discussed, along with the difficulties the agriculture industry faces.

Chapter 2: *Literature Review* This chapter provides an in-depth analysis of the most recent findings in these fields as well as technology for autonomous harvesting, IoT in agriculture, and related fields. It offers a complete overview of the state-of-the-art at the time, identifies research gaps, and establishes the framework for the suggested remedy.

Chapter 3: *Methodology* The design and development process for the IoT-based harvesting robot is laid out in detail in the methodology chapter. It discusses design factors, sensor selection, and the justification for the software tools and programming languages that were picked. Additionally, a thorough explanation of the processes of prototyping and iterative design is provided.

Chapter 4: *System Architecture* The system architecture of the IoT-based harvesting robot is presented in this chapter. The mechanical design, sensor configuration, control algorithms, and IoT connectivity module are all thoroughly explained. To make the entire system design easier to understand, diagrams and illustrations are provided.

Chapter 5: *Implementation* The practical application of the IoT-based harvesting robot is covered in detail in this chapter. It covers the coding and programming details as well as the integration of hardware and software components. The method of implementation is explained, as well as any problems that came up and how they were fixed.

Chapter 6: *Results and Analysis* This chapter presents and analyses the findings from the experiments and testing carried out on the IoT-based harvesting robot. The robot's efficiency and performance parameters are assessed, and they are contrasted with conventional manual

techniques. The discussion of the results is done in relation to completing the project's goals.

Chapter 7: *Conclusion and Future Work* The main conclusions and contributions of the thesis are outlined in this chapter. It reiterates the project's goals and explores how they were accomplished. The IoT-based harvesting robot's prospective areas for development and enhancement are also discussed, along with future research plans.

Chapter 8: *Programming and Coding* This last chapter provides specific information about the coding and programming parts of the IoT-based harvesting robot. Code snippets, a description of the algorithms employed, and code examples are all included.

CHAPTER 2

LITERATURE REVIEW

- 2.1 **IoT-Based Smart Agriculture Monitoring System:** The research proposes an IoT-based smart agriculture monitoring system utilizing various algorithms for detection, quantification, and ripeness checking of vegetables. The study integrates computer vision techniques and machine learning, achieving an accuracy of over 90% with a focus on tomato cultivation.[1]
- 2.2 **Autonomous Smart Agriculture Robot (Agri-Bot):** The Agri-Bot is an autonomous agricultural robot that can carry out labor-intensive tasks like planting, plow-ing, fertilizing, and harvesting. It offers seamless automation across all phases of farming using Arduino UNOs and NodeMCUs.[2]
- 2.3 **Robotic Tomato Picker with High-Tech Vision System:** This study demonstrates an intelligent tomato-picking robot that makes use of accurate grasping mechanisms, enhanced colour segmentation, and advanced vision positioning. Through effective image-based recognition and harvesting, efficiency is increased and labour expenses are decreased, yielding an 83.9% success rate.[3]
- 2.4 **Mechanical Harvesting for Fresh-Eating Tomatoes:** A harvesting robot with a stereo visual unit, end-effector, and rail-based carrier is created for fresh-eating tomatoes. Despite obstacles, the robot harvests with an 83% success rate, increasing productivity and cutting labour expenses.[4]
- 2.5 **Apple Harvesting Robot with Vision Servo Control:** The article offers an apple harvesting robot that uses a geometrically optimized manipulator, a pneumatic gripper, and vision-based recognition to harvest apples. The prototype successfully harvests apples with a success rate of 77%, exhibiting efficient automation and increased production.[5]
- 2.6 **IoT and Wireless Sensors in Smart Agriculture:** The study emphasizes the revolutionary effects of IoT on agriculture, moving away from statistical to quantitative techniques. It covers the potential of UAVs, precision farming, and wireless sensors, explaining the advantages and difficulties they present for contemporary farming techniques.[6]
- 2.7 **Greenhouse Tomato Picking Robot Chassis:** A chassis design is offered for tomato-picking robots in greenhouse settings. Greenhouse Tomato Picking Robot Chassis. The chassis exhibits precise positioning and cruising capabilities through the application of kinematic models, simulations, and physical testing, increasing greenhouse harvesting effectiveness.[7]

Table 1: Literature Review

S.no	Author Name	Achievements	Limitations	Technology used
[1]	Kazy Noor-e-Alam Siddiquee	<ul style="list-style-type: none"> • Agriculture Monitoring System: • Improved Detection and Quantification • Effective Defect Detection: • Integration of CNN 	<ul style="list-style-type: none"> • Dependency on Traditional Sensors: • Energy Harvesting Challenge 	<ul style="list-style-type: none"> • IoT Framework • Circular Hough Transformation (CHT) • Color Thresholding and Segmentation • Energy Harvester (Hybrid) (HEH)
[2]	Hari Mohan Rai	<ul style="list-style-type: none"> • Autonomous Agricultural Operations • Enhanced Efficiency 	<ul style="list-style-type: none"> • Limited Adaptability • Technical Complexity 	<ul style="list-style-type: none"> • Arduino UNOs • NodeMCUs • Autonomous Navigation
[3]	Qingchun Feng	<ul style="list-style-type: none"> • Enhanced Robotic Tomato Picking • Improved Recognition Accuracy 	<ul style="list-style-type: none"> • Limited Versatility • Harvesting Success Rate. 	<ul style="list-style-type: none"> • Vision Positioning Unit • Picking Gripper with Constant Pressure Air • Image Segmentation with HIS Color Model
[4]	Qingchun Feng	<ul style="list-style-type: none"> • Mechanical Harvesting Solution • Integrated Robotic Components 	<ul style="list-style-type: none"> • Harvesting Attempts • Harvesting Time 	<ul style="list-style-type: none"> • Stereo Visual Unit • End-Effector Design • Visual Servo Unit
[5]	Zhao De-An	<ul style="list-style-type: none"> • Integrated Manipulator and End-Effector 	<ul style="list-style-type: none"> • Harvesting Success Rate 	<ul style="list-style-type: none"> • Image-Based Vision Servo Control • Pneumatic Actuated Gripper
[6]	Mr. Long Su	<ul style="list-style-type: none"> • Kinematic Model and Planar Positioning • Greenhouse Tomato Picking Robot Chassis Design 	<ul style="list-style-type: none"> • Operational Scope • Real-World Variability 	<ul style="list-style-type: none"> • SOLIDWORKS • Path Cruising and Setpoint Positioning
[7]	Muhammad Ayaz	<ul style="list-style-type: none"> • Comprehensive Sensor Analysis • Quantitative Approach 	<ul style="list-style-type: none"> • Integration Challenges • Data Security and Privacy 	<ul style="list-style-type: none"> • Internet of Things (IoT) • Wireless Sensors • Data Analytics

CHAPTER 3

Methodology

3.1 Detail explanation of design and development process

The design and development process of the IoT-based harvesting robot, Harvesting Robo Vec, involved several detailed stages, each contributing to the creation of a robust and efficient system. The following is a more detailed explanation of each phase in the design and development process:

3.1.1 Concept Development:

During the concept development phase, extensive research was conducted to identify the specific requirements and objectives of the project. This included studying existing agricultural robotics solutions, analyzing the challenges faced in manual harvesting, and exploring the potential benefits of IoT integration. The team defined the key functionalities and features that Harvesting Robo Vec should possess to optimize efficiency, precision, and productivity in the agricultural sector.

3.1.2 Hardware Design:

To create the physical structure of Harvesting Robo Vec, several components had to be chosen and integrated during the hardware design phase. This involved selecting the proper actuators, sensors, microcontrollers, motors, and communication modules. Power requirements, weight distribution, durability, and ease of maintenance were all taken into account. The robot's mechanical framework, including its chassis, wheels, and robotic arm, was created to be stable, maneuverable, and controlled precisely.

3.1.3 Software Design:

The software design phase included the development of the algorithms and control frameworks required for Harvesting Robo Vec to effectively perform its functions. This project included the development of motor control algorithms for precise movement, robotic arm control algorithms for precise manipulation, and obstacle detection and avoidance algorithms for secure navigation. In order to identify ripe tomatoes, it also included developing a colour recognition system. The ESP32 microcontroller, Raspberry Pi, and control interface may now communicate and work together in real-time thanks to the software architecture.

3.1.4 Prototyping:

Following the completion of the hardware and software designs, the prototyping phase started. The mechanical components, electronic components, and microcontrollers all needed to be assembled, integrated, and programmed. Prototypes were made and tested in order to verify the design's practicality and functionality. Iterative testing and troubleshooting were used to locate and address any issues or limitations. The performance, stability, and dependability of Harvesting Robo Vec have been enhanced.

3.1.5 Integration and System Testing:

During the integration phase, the hardware and software components were combined to form a single system. The Raspberry Pi and ESP32 microcontroller were connected to enable communication and synchronization between the various systems. The overall performance of the robot was tested at the system level. This involves assessing the accuracy of robotic arm manipulation, the accuracy of tomato detection, the effectiveness of obstacle recognition and avoidance, and the precision of movement control. Testing scenarios were developed to evaluate the dependability and toughness of Harvesting Robo Vec in real-world situations.

3.1.6 Iterative Refinement:

Testing and assessment results served as a continuous feedback loop throughout the development process. This feedback helped us pinpoint problem areas and make the necessary adjustments to the design and software algorithms. To resolve any problems or restrictions found during the preliminary testing phases, iterative cycles of improvement and testing were carried out. As a result, Harvesting Robo Vec's final design was optimised for quick and accurate tomato harvesting.

Harvesting Robo Vec required rigorous attention to detail at all stage of the design and development process, from concept creation to hardware design, software design, prototyping, integration, and system testing. Following this thorough procedure allowed the team to develop a reliable and effective IoT-based harvesting robot that successfully addressed the issues with manual harvesting techniques and increased productivity in the agricultural industry.

3.2 Description of the mechanical structure, sensor setup and control algorithm

3.2.1 Mechanical Structure:

The mechanical layout of the Picking Robo Vec was carefully considered to provide the best stability, mobility, and precision control throughout the tomato harvesting process. It was composed of a sturdy chassis that served as the robot's frame and gave the various parts stability and support. The chassis was constructed using robust yet lightweight components to strike a balance between weight and strength requirements.

Depending on the needs of the terrain, the robot had a wheel system. The selection of locomotion technology permitted efficient travel across the various surfaces found in agricultural areas. The robot can easily maneuver across a variety of crop rows and uneven terrain thanks to the wheels' capacity to give grip and mobility.

A crucial component of the mechanical setup was the robotic arm. It was made up of many joints that were moved by servo motors or actuators to produce the proper range of motion. The design of the robotic arm carefully considered kinematics to enable precise control and manipulation capabilities. Since it was designed to safely hold and release tomatoes during harvest, a gripper mechanism was added to the robotic arm. The gripper mechanism offered stability and sensitive handling to reduce any potential injury to the food that was collected.

3.2.2 Sensor Setup:

Harvesting Robo Vec employed a sophisticated sensor setup to enable accurate tomato detection, obstacle avoidance, and safe navigation in the agricultural environment. The sensor setup included:

i. Raspberry Pi Camera:

The robot was equipped with a high-resolution camera, which was linked to the Raspberry Pi. The camera took pictures of the surroundings, which were then analyzed to find and identify ripe tomatoes. The collected photographs were

analyzed using sophisticated computer vision methods, such as colour recognition, to identify the distinctive red colour of ripe tomatoes.

ii. Ultrasonic Sensor:

The robot has an ultrasonic sensor built in to help it avoid obstructions and navigate securely. The sensor supplied distance measurements by sending out ultrasonic waves and timing how long it took for the waves to return, allowing the robot to recognize and steer clear of potential roadblocks. The sensor enabled the robot to move without colliding with any objects by helping it keep a safe distance from them.

3.2.3 Control Algorithm:

The control algorithm played a central role in orchestrating the actions of Harvesting Robo Vec's subsystems, ensuring smooth and efficient operation. The control algorithm encompassed several modules:

i. Color Recognition Algorithm:

The Raspberry Pi camera's image data was analyzed by the colour recognition algorithm. It used computer vision techniques to recognize the distinctive red colour connected to ripe tomatoes in the photographs it had been shown. The system successfully identified the existence and position of tomatoes by examining pixel values and using thresholding and segmentation algorithms.

ii. Motor Control Algorithm:

The motor control algorithm converted complex instructions into exact motor actions. Based on the robot's current location, where the tomato was supposed to be, and the surrounding conditions, it calculated the necessary motor speeds and directions. The robot was able to move smoothly and precisely in order to find the detected tomatoes with the greatest amount of efficiency according to the algorithm.

iii. Obstacle Detection and Avoidance Algorithm:

The ultrasonic sensor's data enabled obstacle recognition and avoidance. The

algorithm continuously monitored distance readings to detect hazards within a specified range. It initiated navigational modifications to prevent crashes, calculating alternative courses and adjusting motor control inputs for obstacle-free navigation.

iv. **Robotic Arm Control Algorithm:**

The robotic arm's and the gripper mechanism's movements were controlled by an algorithm. In order to place the gripper precisely above the tomato, it calculated the necessary joint angles and rotations after receiving instructions regarding the location of the discovered tomato. The algorithm coordinated the actuators or servo motors to carry out the necessary arm movements, providing precise gripping and secure tomato harvesting. It also managed the gripper's release mechanism, which dropped the tomato into the container or collection basket.

The Harvesting Robo Vec's control algorithm combined sensor data, coordinated the subsystems, and allowed communication between the various parts. It was implemented on a microcontroller (like the ESP32). It guaranteed continuous synchronization of the mechanical design, sensor inputs, and control outputs, enabling autonomous navigation, precise harvesting maneuvers, and accurate tomato detection.

3.3 Explanation of the software tools and programming languages used

The development of Harvesting Robo Vec involved the use of various software tools and programming languages to design, implement, and integrate the necessary software components. The following is an explanation of the software tools and programming languages employed in the project:

3.3.1 Python:

Python was chosen as the programming language because of its adaptability, readability, and wide range of library support. The software that Harvesting Robo Vec's Raspberry Pi used as its primary control system was created mostly in Python. Python was used to implement the colour recognition method, carry out picture processing operations, and communicate with other components. A strong basis for effective and simplified development was supplied by

the diverse ecosystem of libraries available in Python, including OpenCV for computer vision applications, NumPy for numerical computations, and requests for handling HTTP requests. Python code's clarity and readability made it possible to create quick prototypes and maintain them with ease

3.3.2 *C/C++:*

The microcontroller, specifically the ESP32, was programmed using the C and C++ programming languages. Real-time activities can be performed with C/C++ because it offers low-level control, direct access to hardware capabilities, and effective memory management. To ensure quick and accurate microcontroller execution, the motor control algorithms, robotic arm control, obstacle detection, and avoidance logic were all built in C/C++. The C/C++ programming language allows the software running on the microcontroller to fully utilise the microcontroller's capabilities and provide efficient performance. Additionally, Harvesting Robo Vec's hardware components were able to be integrated with ease thanks to C/compatibility C++'s with a number of microcontroller libraries and the Arduino environment.

3.3.3 *Arduino IDE:*

The ESP32 microcontroller was programmed and loaded with code using the Arduino Integrated Development Environment (IDE). The Arduino IDE offered a simple programming environment and user-friendly interface, making it simple to develop, compile, and deploy code on the microcontroller. The ESP32 and the C/C++ programming languages were supported, and it offered libraries and tools designed especially for Arduino boards. The Arduino IDE made it easier to set up the microcontroller, upload firmware, and debug code, which made it possible to write and test the ESP32 software quickly.

3.3.4 *OpenCV:*

Implementing image processing tasks for tomato detection required the use of OpenCV (Open-Source Computer Vision Library). OpenCV is a well-known open-source computer vision library that provides a huge array of tools and techniques for feature identification, picture modification, and analysis. The Raspberry Pi camera's photos were processed using OpenCV in Harvesting Robo Vec so that the colour recognition algorithm could identify the distinctive red hue of ripe tomatoes. Accurate tomato localization and detection were made possible by OpenCV's features for image thresholding, segmentation, contour detection, and colour space conversions.

3.3.5 *HTTP and RESTful APIs:*

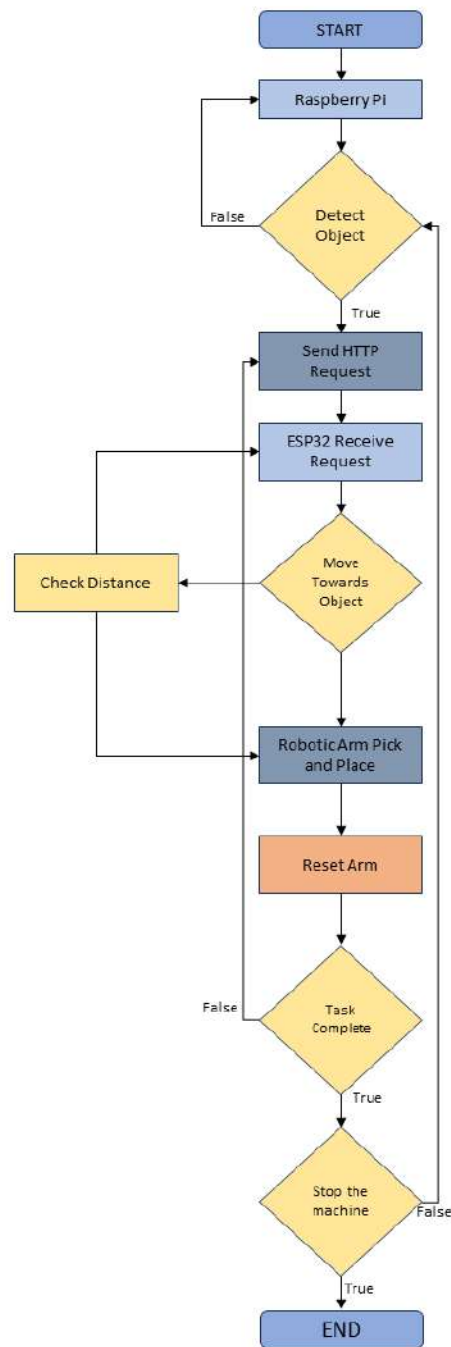
The Raspberry Pi and the ESP32 microcontroller communicated using the Hypertext Transfer Protocol (HTTP) and Representational State Transfer (REST) architecture. These components communicated with each other via HTTP requests and replies to exchange data and control commands. As the main controller, the Raspberry Pi could issue HTTP requests to the microcontroller that would tell it to move, control a robotic arm, and perform other tasks. In turn, the ESP32 gave the proper HTTP replies in order to acknowledge the orders or give feedback. Due to the smooth coordination and synchronization provided by this communication system, software running on the Raspberry Pi and the microcontroller were able to work together productively.

3.3.6 *Git:*

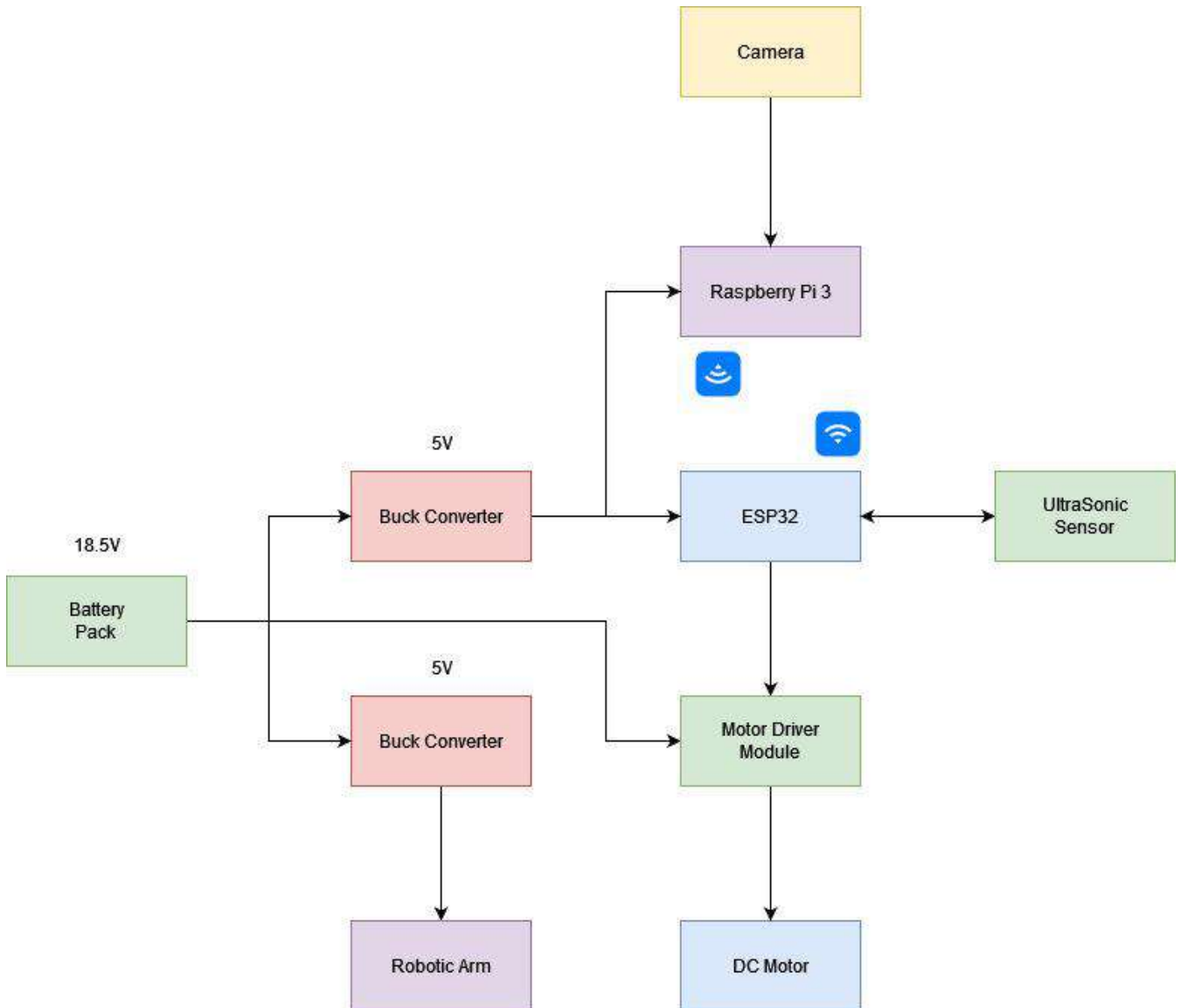
The development team used Git, a distributed version control system, to manage source code and collaborate. Git made it easy to track changes, merge updates from many team members, and version code effectively. The development team could work on several features or modules at once using Git, log changes, and combine them into a single codebase. Git's branching and merging features made it possible for successful teamwork and collaboration throughout the project, ensuring that the software development process remained well-organized.

The application of numerous software tools and programming languages allowed for the creation of a robust and comprehensive software development environment for Harvesting Robo Vec. The Raspberry Pi's control software was made easy to write with Python, while the low-level optimization and control of the microcontroller were made possible with C/C++. The Arduino IDE made it simpler to deploy and programme the ESP32 microcontroller. OpenCV allowed for complex image processing applications like tomato detection. The link between the Raspberry Pi and the microcontroller was made possible through HTTP and RESTful APIs. Git also made it possible for efficient version control, code management, and teamwork throughout the development process, giving the development team a fluid and well-organized workflow.

3.4 *Flowchart*



3.5 Block Diagram



CHAPTER 4

SYSTEM ARCHITECTURE

4.1 COMPONENTS OF THE HARVESTING ROBO VEC

4.1.1	ESP32	4.1.11	Battery Management System
4.1.2	Raspberry Pi Model 3b+	4.1.12	Ultrasonic Sensor
4.1.3	Pi camera	4.1.13	Ultrasonic holder
4.1.4	Pi Cam Ribbon 3 meter	4.1.14	Motor driver module
4.1.5	Acrylic sheet	4.1.15	Buck Converters
4.1.6	DC motors	4.1.16	Vero board
4.1.7	L&U Brackets	4.1.17	Jumper wires
4.1.8	Robotic Arm	4.1.18	Voltage regulator IC
4.1.9	Li-ion Battery	4.1.19	Charging jack
4.1.10	Battery holder	4.1.110	Servo Motors

4.1.1 ESP 32 [8]

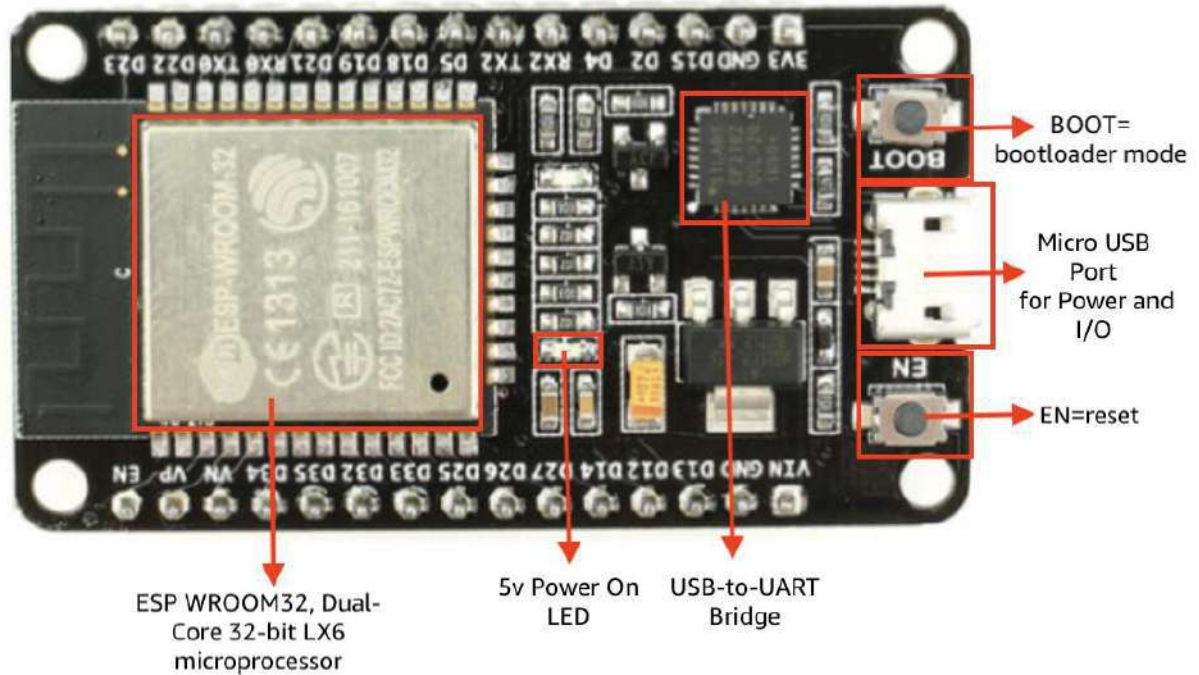


Figure 2: ESP-32 Module

Table 2: Specifications of ESP-32

Component Name	ESP32
Manufacturer	Espressif Systems
Description	Wi-Fi and Bluetooth-enabled SoC
Dimensions	18mm x 25.5mm
Processor	Xtensa dual-core 32-bit LX6 MCU
Clock Frequency	Up to 240 MHz
Operating Voltage	2.2V to 3.6V
Operating Temperature	-40°C to +85°C
Wireless Standards	Wi-Fi 802.11 b/g/n, Bluetooth v4.2
Input Voltage	2.2V to 3.6V
Power Consumption	Varies based on usage and configuration
GPIO Pins	34 (Including 12-bit ADC)
Memory	Up to 4MB Flash, 520KB SRAM
Wi-Fi Throughput	Up to 150 Mbps
Bluetooth Range	Up to 10 meters
Compliance	FCC, CE, IC, SRRC
Security Features	WPA/WPA2 personal and enterprise security, secure boot

4.1.2 RASPBERRY PI [9]

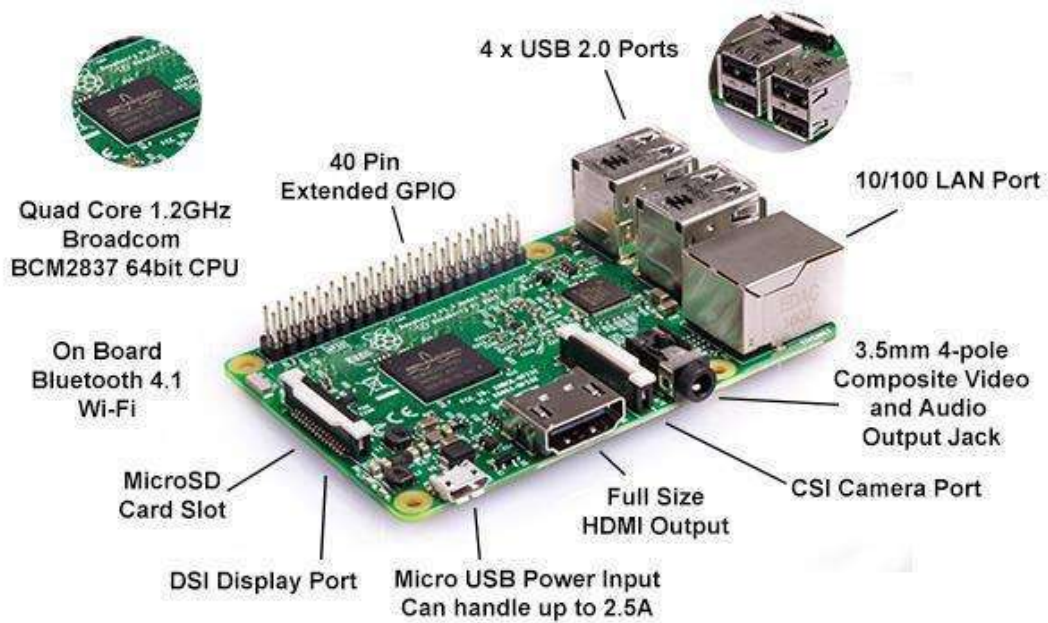


Figure 3: Raspberry Pi Model 3B+ Module

Table 3: Specifications of Raspberry PI

Component Name	Raspberry Pi Model 3B+
Manufacturer	Raspberry Pi Foundation
Description	Single-board computer
Processor	Broadcom BCM2837B0, Cortex-A53
Clock Frequency	1.4 GHz
Memory	1GB LPDDR2 SDRAM
Storage	MicroSD card slot
Connectivity	- 2.4 GHz and 5 GHz 802.11b/g/n/ac Wi-Fi - Bluetooth 4.2 - 10/100 Ethernet
USB Ports	4 x USB 2.0
Video Output	HDMI, Composite Video
Audio Output	3.5mm jack, HDMI
GPIO Pins	40
Operating System	Linux, Windows 10 IoT Core, and others
Power	5V micro USB
Dimensions	85mm x 56mm x 17mm

4.1.3 PI-CAMERA [10]



Figure 4Pi Camera Module

1. Image Sensor: The camera on the Raspberry Pi makes use of a compact image sensor that can record high-resolution images.
2. Resolution: It supports a range of resolutions, including HD (720p), Full HD (1080p), and lower resolutions appropriate for a range of applications.
3. Megapixels: The Pi camera can have varied megapixel capacities depending on the model, including 8 MP, 12.3 MP, or even higher.
4. Lens: The camera module has either a fixed focal length lens or a programmable focal length lens that may be altered for various focal lengths and depths of field.
5. Field of Vision: Depending on the lens being used, the Pi camera's field of view (FOV) can change. For greater coverage, several models come with wide-angle or fish-eye lenses.
6. Connectivity: A special ribbon cable connector that allows both power and data transfer links the camera module to the Raspberry Pi board.
7. Image processing: The Pi camera has built-in image processing capabilities that enable real-time image quality tweaks and enhancements.
8. Video Recording: It enables the capture of fluid and high-quality video footage by supporting video recording at a range of frame rates and resolutions.
9. Interface: The MIPI CSI (Camera Serial Interface) bus is used to connect the camera module to the Raspberry Pi board.
10. Compatible models include the Raspberry Pi 3, Raspberry Pi 4, and Raspberry Pi Zero. The Pi camera was created especially for Raspberry Pi boards.

4.1.4 PI CAMERA RIBBON [11]



Figure 5:Pi cam Ribbon

1. Connector Type: Ribbon cable connectors typically include 15 pin connectors on each end. The camera module is attached to one end, and the Raspberry Pi board's CSI connector is plugged into the other.
2. Length: The ribbon cable's average length is about 15 cm (6 inches). This length makes it simple to connect the camera module to the board and is suited for the majority of Raspberry Pi board configurations.
3. Flexibility: The camera module may be easily positioned and routed inside the project enclosure thanks to the ribbon cable's flexibility.
4. Conductors: The cable has several conductors arranged in a flat, ribbon-like configuration. The camera module and the Raspberry Pi board are connected by these conductors, which also carry power and data signals.
5. Shielding: Some ribbon cables may come with shielding to reduce electromagnetic interference (EMI) and improve signal integrity.
6. Compatibility: The ribbon cable is designed specifically for use with Raspberry Pi camera modules and Raspberry Pi boards that have a CSI connector, such as the Raspberry Pi 3, Raspberry Pi 4, and Raspberry Pi Zero series.

4.1.5 ACRYLIC SHEET [12]



Figure 6: Acrylic Sheet

1. **Thickness:** Acrylic sheets are available in a range of thicknesses, commonly between 1.5 mm (0.06 inches) and 25 mm (1 inch) or even thicker. Depending on the particular use and the necessary amount of rigidity and strength, the thickness can change.
2. **Size:** Acrylic sheets come in common sizes like 2 feet by 4 feet and 4 feet by 8 feet (48 inches by 96 inches) (24 inches by 48 inches). However, they are simple to cut and alter to fit the needs of a particular project.
3. **Transparency:** Acrylic sheets enable strong light transmission thanks to their remarkable optical clarity and transparency. Similar to glass, they provide a clear and distortion-free picture.
4. **UV Resistance:** Acrylic sheets are suited for both indoor and outdoor applications due to their natural UV resistance. They provide good defense against fading and yellowing brought on by extended sun exposure.
5. **Weatherability:** Acrylic sheets are resistant to most weather conditions, including rain, snow, and temperature changes. They also have good weathering qualities. Because of environmental variables, they are less prone to cracking or warping.
6. **Acids, bases, and solvents** are among the numerous substances to which acrylic sheets have a moderate resistance. They should be utilized cautiously in areas where chemicals are present since they might be vulnerable to some of them.

4.1.6 DC MOTORS [13]



Figure 7: DC Motor

Table 4: Specifications of DC Motor

Component Name	DC 6V Gear Motor
Voltage (Range)	3Volt
Current(Stall)	100mA
Speed	125RPM
Torque	0.8kg.cm
Garmotor Style	Hobby
Gear Material	Plastic
Gearbox Ratio	48:1

4.1.7 L&U BRACKET [14]



Figure 8:L&U Bracket

1. **Material:** L and U brackets for robotic arm assembly are often composed of strong, lightweight materials like steel, acrylic, or aluminum. The choice of material is influenced by things like weight concerns and required levels of strength.
2. **Size and Dimensions:** Depending on the precise design and application of the robotic arm, the brackets' size and dimensions may change. For the purpose of supporting various arm configurations, they are often offered in a variety of lengths, widths, and thicknesses.
3. **The L and U brackets include a pattern of holes or slots that make it possible to mount and attach other parts securely, such as servo motors, linkage systems, or structural components. The hole designs may have conventional industrial spacing or they may be tailored to meet particular needs.**
4. **Mounting Options: Options for Mounting:** The brackets may be mounted using slots, threaded holes, or through-holes. The robotic arm frame or other components can be attached to the brackets in a variety of ways thanks to these possibilities.
5. **The brackets are made to be strong and load-bearing in order to endure the loads and forces involved in robotic arm operation. The material used and the unique design features of the brackets determine their strength and load capacity.**
6. **Surface Treatment:** To increase durability, prevent corrosion, and create an appealing appearance, the brackets may have a surface finish such as powder coating or anodizing.

4.1.8 ROBOTIC ARM [15]

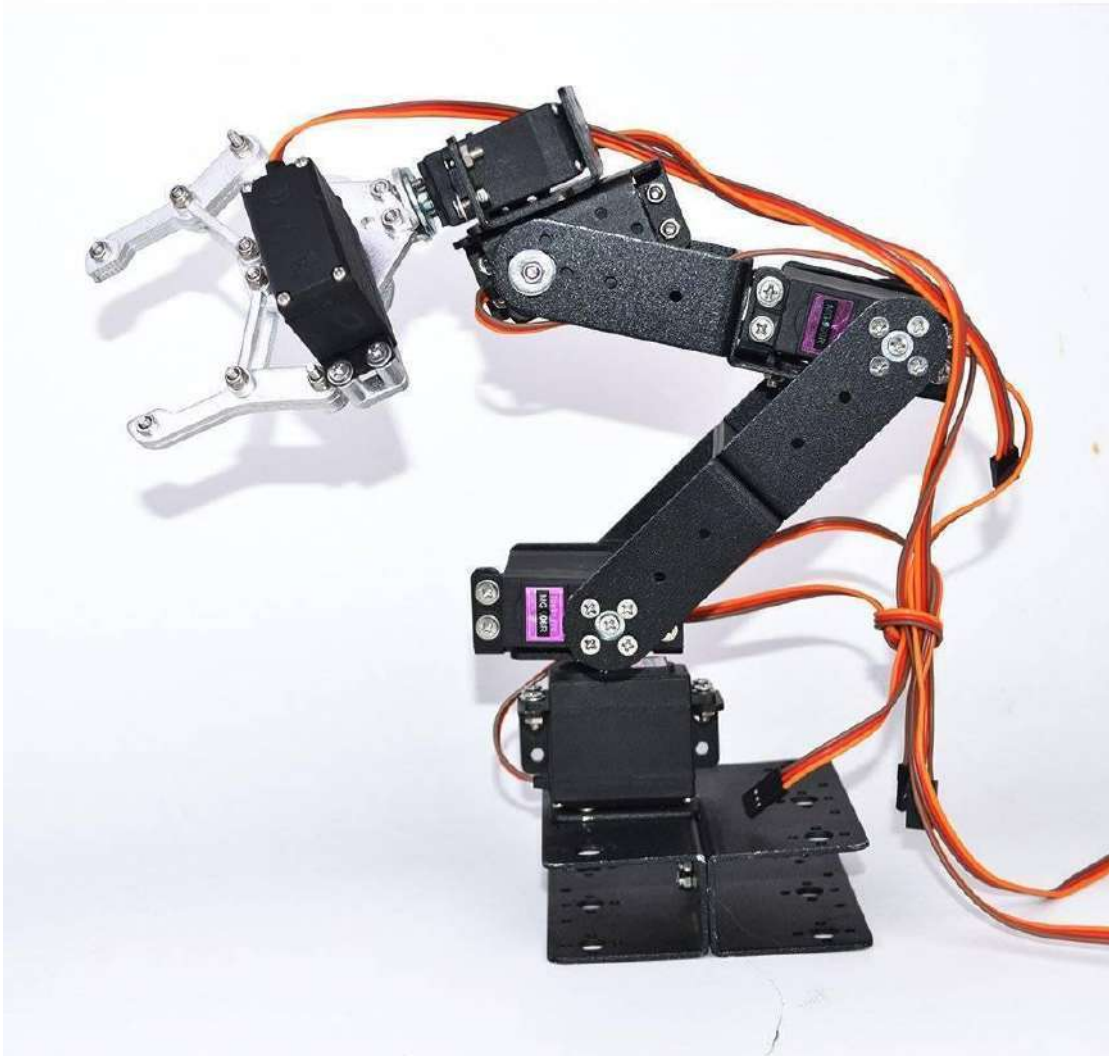


Figure 9: Robotic Arm

Table 5: Specifications of Robotic Arm

Component Name	Robotic Arm Kit
Material	Aluminum Alloy
Function	Clamping Object
Color	Black
Servo(s)	5

4.1.9 Li-Ion BATTERY [16]



Figure 10: Li-ion Batteries

Table 6: Specifications of Li-ion Battery

Component Name	Li ion Battery
Capacity	Nominal 2400mAh
Nominal Voltage	3.7V
Max charge Current	2300 mA
Rechargeable	Yes
Operating Temperature	Charge- 0 to 45 C Discharge -20 to 60 C

4.1.10 BATTERY HOLDER [17]



Figure 11: Li-ion Battery Holder

1. **Battery Compatibility:** Li-ion battery holders are made to handle just certain types of rechargeable cylindrical batteries, like the 18650, 14500, and 26650. The holder needs to work with the particular battery size you plan to use.
2. **Material:** Strong, heat-resistant materials like plastic or nylon are frequently used to make Li-ion battery carriers. These components offer mechanical support and electrical insulation to hold the battery firmly.
3. **Contacts:** The Li-ion battery's positive and negative terminals are electrically connected by metal contacts or springs in the holder. Between the battery and the connected gadget, these connections guarantee adequate electrical conductivity.
4. **Terminal Configuration:** Li-ion battery holders often include positively (+) and negatively (-) labelled terminals to guarantee proper battery polarity. In some holders, there may be extra terminals for applications that call for different battery combinations.
5. **Wiring:** The holder may have soldering points or built-in wiring for simple connection to the device or circuit, depending on the design. This enables simple battery holder integration into your project.
6. **Li-ion battery holders may have mounting holes or adhesive backing for firmly attaching to a surface or enclosure. Mounting options. This makes for simple installation and stable operation.**

4.1.11 3s BATTERY MANAGEMENT SYSTEM [18]

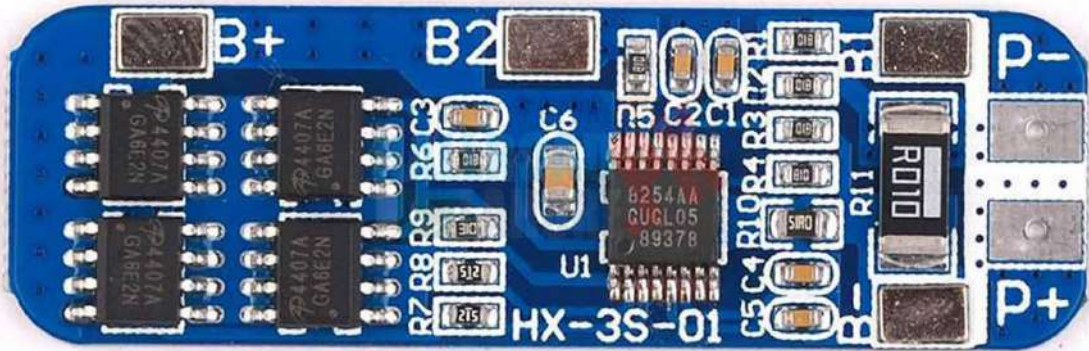


Figure 12: 3s Battery Management System

Table 7: Specifications of Battery Management System

Component Name	3s Battery Management System
Overcharge Voltage Detection	3.9~4.35V ± 0.05V
Over discharge voltage Detection	2.3~3.0V ± 0.05V
Maximum operating current	6~8A
Quiescent current	< 30uA
Internal resistance	< 100mΩ
Charging voltage	12.6V ~ 13V
Working temperature	-40~+50°C
Short circuit protection	Yes, delayed self recovery

Dimensions	(50 x 16 x 1) mm
------------	------------------

4.1.12 ULTRASONIC SENSOR [19]

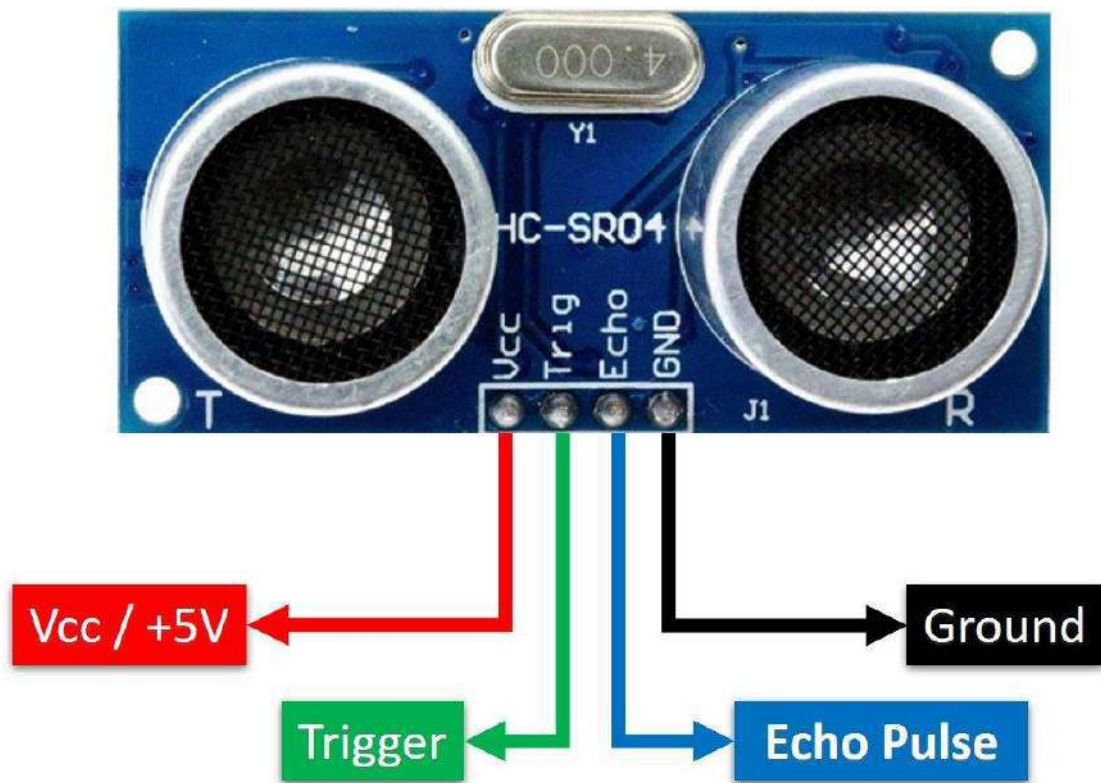


Figure 13: Ultrasonic Sensor

Table 8: Specifications of

Ultrasonic Sensor

4.1.13 ULTRASONIC



Figure 14: Ultrasonic Holder

1. Check the mounting bracket or holder to make sure it is compatible with the particular ultrasonic sensor brand you wish to use. Think about the sensor's dimensions, shape, and mounting needs.
2. Material: Metal or plastic are frequently used to make mounting brackets or holders

Component Name	Ultrasonic Sensor
Manufacturer	Various manufacturers
Description	Sensor for distance measurement using sound waves
Operating Voltage	5V
Operating Current	Less than 15mA
Operating Frequency	40 kHz
Detection Range	2 cm to 400 cm
Resolution	1 cm
Accuracy	±1%
Trigger Pulse Width	10µs
Echo Pulse Output	Echo signal proportional to the detected distance
Operating Temperature	-20°C to +70°C
Dimensions	Varies by model
Output Interface	Digital or Analog
Connection Type	3-pin or 4-pin

for ultrasonic sensors. Select a material that offers the right balance of strength, stability, and environmental resistance.

3. Search for a bracket or holder that enables you to change the ultrasonic sensor's position or angle. You can place the sensor in the best possible location for your application thanks to this adaptability.
4. Consider the mounting possibilities offered by the bracket or holder. Through holes, slots, or adhesive backing, it should be simple to attach to a surface or building.
5. Secure Fit: To avoid unintentional movement or misalignment while operating, the bracket or holder should firmly hold the ultrasonic sensor in place.
6. Make sure that the mounting bracket or holder makes the wire connections for the ultrasonic sensor accessible. This makes wiring more convenient.
7. To guarantee a flawless integration, take into account if the mounting bracket or holder is compatible with other parts of your system, such as the robotic arm or mounting frame.
8. Size and Dimensions: The dimensions of the mounting bracket or holder should align with the ultrasonic sensor's form factor and your specific application requirements. Consider factors such as clearance, space limitations, and the overall size of your system.

4.1.14 MOTOR DRIVER MODULE [21]

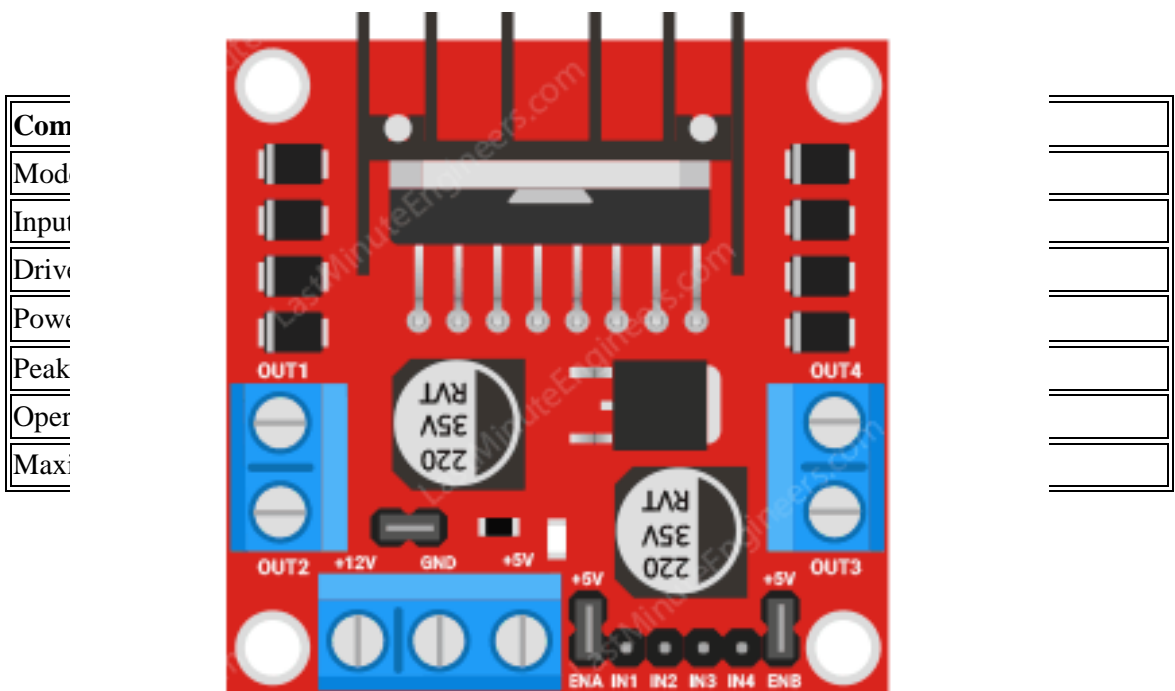


Figure 15: Motor Driver Module

Table 9: Specifications of Motor Driver Module

4.1.15 BUCK CONVERTER [22]



Figure 16: Buck Converter

Table 10: Specifications of Buck Converter

Component Name	7805 5v IC Integrated Circuit
Output current	0.5A
Output voltages	5V
Thermal overload protection	Yes
Short Circuit Protection	Yes
Output Voltage tolerance	2%

4.1.16 VERO-BOARD [23]

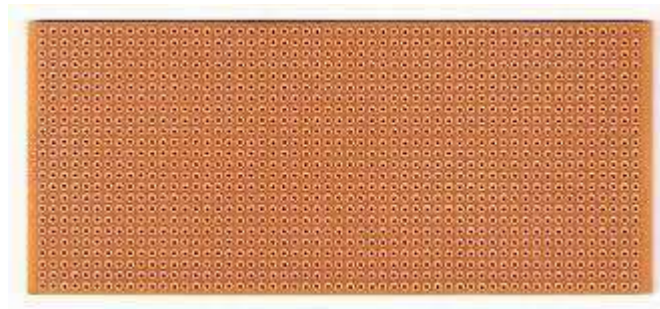


Figure 17: Vero Board

1. **Material:** A non-conductive substrate material, like phenolic resin or fibreglass, is commonly used to make vero boards. The substrate gives the circuit components insulation and mechanical support.
2. **Copper Strips:** The length of the Vero board is covered in parallel copper strips. For electrical connections between components, these copper strips serve as conductive channels.
3. **Hole Spacing:** The substrate of Vero board is punctured with holes in a predictable way. The spacing between these holes adheres to a standard pitch, generally 2.54 mm (0.1 inches), making it simple to insert and solder electronic components.
4. **Copper Pads:** A copper pad or ring surrounds each hole on the vero board. These pads offer component leads or cables a surface for soldering. Solder
5. **Mask:** The substrate of some vero boards has been coated with a solder mask. Between neighbouring copper strips, the solder mask offers electrical insulation and aids in preventing solder bridges.
6. **Dimensions:** Vero boards are available in a range of shapes and sizes, including overall length, breadth, and thickness. Single-sided boards with measurements of 80 mm x 100 mm or 100 mm x 160 mm are typical sizes, but smaller and bigger sizes are also available.
7. **Prototype Regions:** The prototyping areas on Vero boards often have a grid-like arrangement of holes that are not connected to the copper strips. These spaces make it simple to prototype extra component placement and wiring.
8. **Solderability:** Vero board is designed to be solder-friendly, allowing components to be securely attached and soldered to the copper strips and pads. The copper surface is usually pre-tinned, facilitating easy soldering.

4.1.17 JUMPERWIRE [24]



Figure 18: Jumper Wire

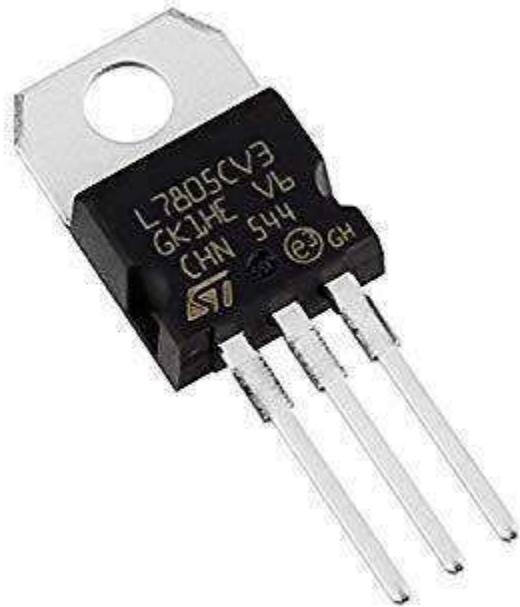
1. Length: There are different jumper wire lengths available, often ranging from a few centimeters (short) to several tens of centimeters (long). The length aids in clean wiring by determining the distance between the connected components.
2. Wire Gauge: The thickness and current-carrying capacity of jumper wires are determined by the wire gauge, which is available in a variety of sizes. There are three standard wire gauge sizes: 22 AWG, 24 AWG, and 26 AWG. Greater currents can flow via thicker wires, which have a lower gauge number.
3. Conductor Material: Copper or tinned copper is often used as the conductor in jumper wires because it offers strong conductivity and is simple to solder. A thin solder

Component Name	DC-DC Buck Converter
-----------------------	-----------------------------

covering on tinned copper wires makes it easier to solder connections. Insulation: Jumper wires have insulation covering the conductor to prevent short circuits and ensure proper electrical isolation. The insulation material is usually PVC (Polyvinyl Chloride) or similar materials, which are flexible and durable.

4. Connector Types: Jumper wires may include a variety of connectors, such as male-male, female-female, or male-female connectors, at each end. Female connectors have receptacles or sockets, whereas male connectors have exposed pins or prongs. These connectors make it simple to install and connect to a variety of parts and gadgets.
5. Color Coding: To distinguish between various signals or connections, jumper wires are frequently color-coded. Red, black, blue, green, yellow, and white are common colors. The organization and recognition of connections in a circuit are aided by colour labelling.

Regulator Type	Step Down (Non-Isolated Input to Output)
Input Voltage	+4 to 40vdc
Output Voltage	+1.25 to 35vdc
Output Current	2A Rated, (3A maximum with heat sink)
Switching Frequency	150kHz
Efficiency	gh)
Dropout Voltage	



4.1.18 Voltage Regulator IC [25]

Figure 19: Jumper Wire

Table 11: Specifications of Voltage Regulator IC

4.1.19 CHARGING JACK [26]



Figure 20: Jumper Wire

1. Connector Type: Various connector types, including USB Type-A, USB Type-C, Micro-USB, and Lightning, can be used with the charging port (for Apple devices). The connector type needs to be compatible with both the charging cable and the target device.
2. Pin Configuration: Depending on the charging protocol being utilised, the charging port may have a varied pin configuration. For instance, USB Type-C ports can support a variety of charging protocols, such as Qualcomm Quick Charge or USB Power Delivery (PD), each of which has unique pin configurations and capabilities.
3. Power Rating: The charging port's power rating needs to be appropriate for the device it is used to charge. Usually, the voltage (V) and current (A) are provided, such as 5V/2A (10W) or 9V/3A. (27W). Faster charging is made possible by greater power ratings, but the charging equipment must be able to handle and work with the higher power levels.
4. Charging Standards: The charging port may comply with standards like USB BC 1.2, QC, PD, or wireless Qi, ensuring compatibility and support for different charging protocols.

4.1.20 Servo Motor

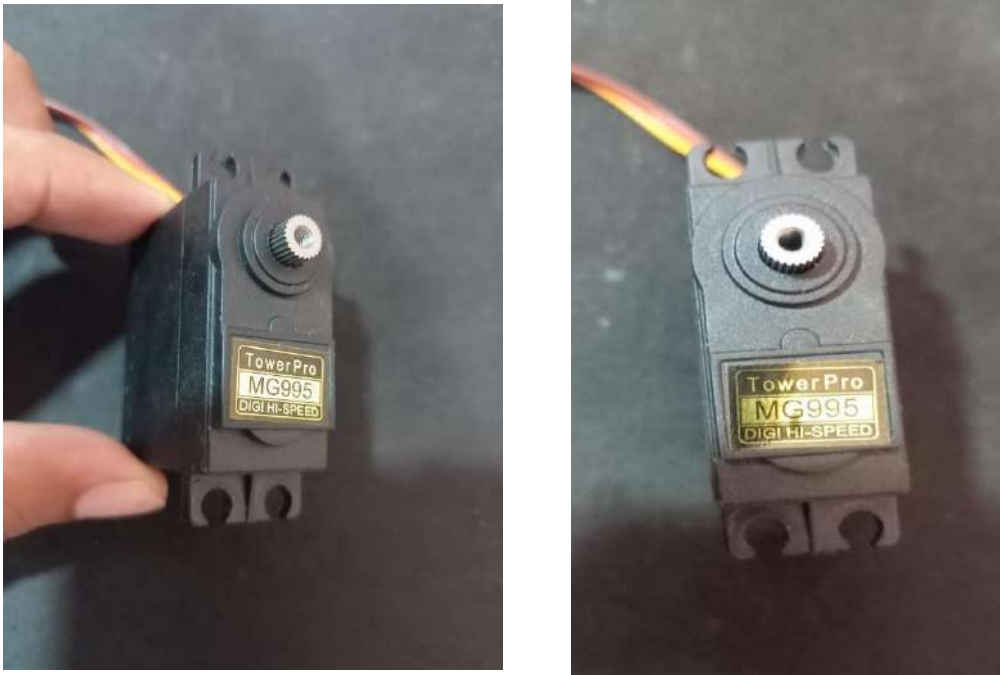


Figure 21: Servo Motor

Table 12: Specifications of Servo Motor

Specification	Value
Operating Voltage	4.8V - 7.2V
Stall Torque	9.4 kg-cm (at 4.8V); 11 kg-cm (at 6V)
Operating Speed	0.17 seconds/60 degrees (at 4.8V); 0.14 seconds/60 degrees (at 6V)
Dimensions	Approx. 40.7mm x 19.7mm x 42.9mm
Weight	Approx. 55 grams
Gear Type	Metal gear
Control System	Analog, Pulse Width Modulation (PWM) control
Rotation	180 degrees (90 degrees in each direction from the center position)
Operating Temperature	0°C to 55°C
Dead Band Width	5 μ s
Servo Cable Length	Approx. 30cm
Connector Type	"JR" style, three-pin connector with signal, power, and ground wires

4.2 Overview of the overall system architecture of Harvesting Robo Vec

The overall system architecture of Harvesting Robo Vec is comprised of various interconnected components that work together to enable autonomous tomato harvesting. Here is a more detailed overview of each component and their interactions:

4.2.1 Raspberry Pi:

Harvesting Robo Vec's central processing unit and brain is the Raspberry Pi. High-level decision-making, image processing, and tomato detection are its domains. Using its camera module, the Raspberry Pi takes pictures that are then processed using computer vision software like OpenCV. Based on colour recognition, these algorithms examine the photos to determine which tomatoes are ripe. The robot's actions and robotic arm's harvesting control are then guided by the location of the detected tomato.

4.2.2 ESP32 Microcontroller:

The ESP32 microcontroller controls and activates the robot's hardware at a low level. It accepts instructions and commands from the Raspberry Pi and converts them into precise motor actions and movements. To provide precise control over the robot's locomotion and the movements of the robotic arm, the microcontroller communicates with a variety of sensors and actuators, including motor drivers. Additionally, it speaks with the Raspberry Pi so that it may get new commands and transmit status information.

4.2.3 Mechanical Structure:

The physical parts that allow for mobility, tomato sensing, and tomato harvesting are all included in the mechanical design of Harvesting Robo Vec. It has a strong chassis on which the mechanical and electronic subsystems are housed. The robot's numerous components are intended to be supported and stabilised by the chassis. A locomotion system, such as wheels or tracks, is built into the robot for effective mobility over various surfaces. A robotic arm with various degrees of freedom is also a part of the mechanical design, enabling for accurate manipulation and grabbing of tomatoes during harvest.

4.2.4 Sensors:

Variety of sensors aboard Robo Vec give environmental information for decision-making and control. Images taken by the Raspberry Pi camera are used to identify tomatoes. The robot can navigate the field safely by using ultrasonic sensors to detect obstacles and avoid collisions. To acquire environmental information that can affect harvesting operations, additional sensors, like temperature and humidity sensors, can be incorporated.

4.2.5 *Control Algorithms:*

The Raspberry Pi and the ESP32 microcontroller both use a collection of software algorithms collectively referred to as the control algorithms. The robot's operations can be coordinated and operated autonomously thanks to these algorithms. The Raspberry Pi camera's photos are processed by the colour recognition algorithm, which looks for ripe tomatoes by their distinctively red colour. Motor control algorithms compute the required motor movements for accurate navigation while taking into consideration the position of the robot and the tomato's goal. Ultrasonic sensor data is used by algorithms for obstacle detection and avoidance to identify impediments and modify the robot's path to prevent collisions. During harvest, precise manipulation and gripping of tomatoes are made possible by robotic arm control algorithms.

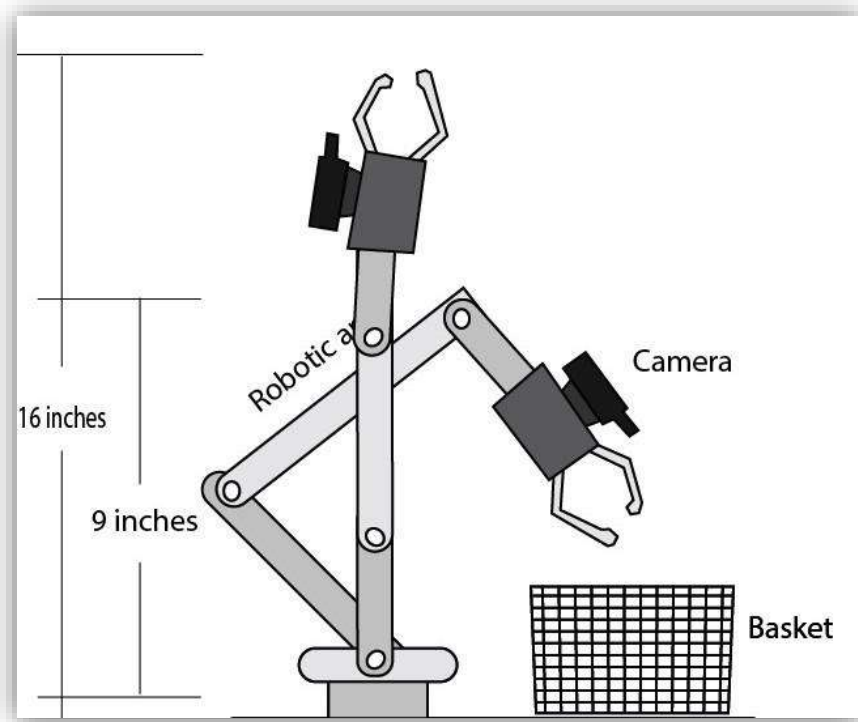
4.2.6 *Communication:*

The ESP32 microcontroller and Raspberry Pi must be able to communicate in order to effectively coordinate and control Harvesting Robo Vec. HTTP or a similar protocol is often used to achieve this communication. The microcontroller receives instructions from the Raspberry Pi, including those for controlling robotic arms and movement. The Raspberry Pi can track the robot's progress and modify its course of action thanks to the microcontroller's feedback and status updates. This channel of communication makes sure that the microcontroller and the software running on the Raspberry Pi work together without any issues.

4.2.7 *Power and Energy Management:*

Systems for managing power and energy supply enable a consistent power supply and effective energy use for the robot's functioning. This comprises systems for managing batteries, distributing power, and charging batteries. These solutions are created to prolong autonomous harvesting operations by maintaining hardware components and maximising the robot's time in the field of operation.

The system architecture of Harvesting Robo Vec integrates hardware, software algorithms, sensors, communication, and power management for autonomous tomato harvesting. Raspberry Pi is the main control unit, processing data and sending commands to the microcontroller for precise navigation, tomato detection, and robotic arm manipulation.



4.3 Dimension of Harvesting Robo Vec

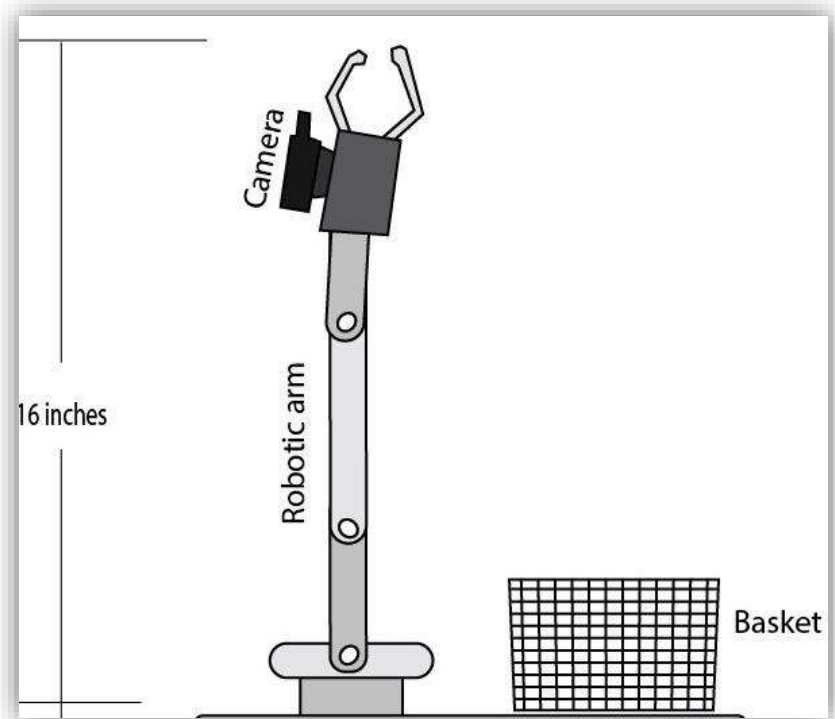


Figure 22:

Crouched

Figure 23: stretched

CHAPTER 5:

IMPLEMENTATION

5.1 Explanation of the iterative design process and prototyping

5.1.1 Initial Design and Conceptualization:

Understanding the project's needs and goals in great detail was the first step in the iterative design process. To determine the major characteristics and capabilities of Harvesting Robo Vec, the development team performed a thorough review of the literature, spoke with subject matter experts, and obtained user feedback. During the preliminary design stage, the general system architecture was sketched out, the mechanical structure was described, the right sensors and actuators were chosen, and the software and control algorithms needed for autonomous tomato harvesting were identified.

5.1.2 Prototype Development:

The development team went on and built a first prototype of Harvesting Robo Vec based on the initial idea. This required building a simple mechanical construction that gave the robot the stability and mobility it needed. The chosen sensors, including an ultrasonic sensor and a

Raspberry Pi camera, were incorporated into the prototype. The group also put a simple control system in place to show how the robot could move and initially detect tomatoes.

5.1.3 Testing and Evaluation:

To evaluate the prototype's performance and pinpoint its weak points, it was put through a rigorous testing and evaluation process. Various terrains and lighting conditions were included in the testing situations to approximate actual farming conditions. The group tested the prototype's capacity to locate tomatoes, avoid hazards, and control the robotic arm during harvesting. Data was gathered to assess the functionality of the prototype's correctness, efficacy, and dependability.

5.1.4 Feedback Collection:

Multiple groups, including the development team, industry professionals in agriculture, and possible end users, provided feedback. The prototype's strengths, limitations, and areas in need of modification were well-understood through user observations, expert views, and data from field tests. The input was thoroughly examined in order to improve the design, solve any weaknesses, and raise the robot's overall performance.

5.1.5 Refinement and Iteration:

The development team started an iterative refinement process using the feedback gathered. This required examining the test findings, locating design defects or performance bottlenecks, and modifying the mechanical setup, sensor configuration, control algorithms, and software components as appropriate. To fix the problems found and improve the functionality, dependability, and usability of the robot, the team repeatedly improved the design.

5.1.6 Prototyping of Enhanced Designs:

Following rounds of prototypes were created based on the improved design, incorporating the changes and improvements found throughout the gathering of feedback and refinement phases. To address the flaws and difficulties found in earlier rounds, the development team changed the mechanical setup, sensor configuration, and control algorithms progressively. Each subsequent prototype sought to address the discovered limits and enhance a particular aspect of the robot's performance.

5.1.7 Continuous Testing and Evaluation:

To confirm the effect of the design changes, each new prototype underwent thorough testing and evaluation. To replicate the situations and difficulties the robot would face when collecting tomatoes, elaborate testing scenarios were created. The mechanical systems, sensor configuration, control algorithms, and overall functionality of the robot were evaluated. Testing data was thoroughly studied to guide additional design improvements.

5.1.8 Integration and System-level Testing:

The improved parts were added to a functioning system as the design iterations developed. Testing at the system level was done to gauge how well the fully integrated robot performed. This required thorough testing of the mechanical framework, sensor configuration, control scheme, communication system, and power management systems. To ensure smooth operation and synchronisation, the robot's capacity to autonomously locate and gather tomatoes, move around obstacles, and adapt to various surroundings was carefully examined.

The development team was able to gradually improve the design of Harvesting Robo Vec thanks to the iterative design method and prototyping strategy. The robot's functionality, performance, and dependability have significantly improved as a result of the constant feedback gathering, testing, and refining cycles. The team developed a reliable and effective solution through this iterative method that satisfied the needs and standards for autonomous tomato harvesting in actual agricultural situations.

5.2 Discussion of challenges faced and solutions implemented

During the development of Harvesting Robo Vec, several challenges were encountered, requiring innovative solutions to overcome them. Here is a discussion of the challenges faced and the solutions implemented:

5.2.1 Tomato Detection Accuracy:

Challenge: Achieving accurate and reliable tomato detection presented a significant challenge. Variations in lighting conditions, background clutter, and the similarity of tomato color to other objects made accurate detection a complex task.

Solution: The development team implemented advanced image processing techniques, leveraging computer vision algorithms such as color thresholding, contour detection, and filtering. These algorithms were fine-tuned to effectively detect and differentiate ripe tomatoes based on their distinctive red color. Additionally, calibration techniques were

employed to handle variations in lighting conditions and optimize the accuracy of the tomato detection algorithm.

5.2.2 *Obstacle Detection and Avoidance:*

Challenge: Navigating through agricultural fields with varying terrains and obstacles posed a challenge for Harvesting Robo Vec. Ensuring timely and accurate obstacle detection to avoid collisions was essential.

Solution: The robot had strategically positioned ultrasonic sensors to detect obstacles. Sensor data was continuously monitored and processed to trigger actions, avoiding collisions and ensuring safe navigation.

5.2.3 *Robotic Arm Manipulation:*

Challenge: It was extremely difficult to grip and harvest tomatoes with a robotic arm while maintaining accuracy and dependability. The movement of the robotic arm needed to be carefully controlled to guarantee precise alignment and a firm grip.

Solution: The robotic arm's movement was managed by servo motors or actuators, according to the development team. The gripper of the robotic arm was carefully positioned over the tomatoes using position control algorithms. Closed-loop control was made possible by feedback systems, such as position sensors or encoders, which offered real-time information about the position of the arm. The arm's movements were adjusted by the control algorithms to achieve a firm hold on the tomatoes while reducing the chance of harm.

5.2.4 *Power Management:*

Challenge: It was difficult to manage the robot's energy consumption and optimise power use in order to assure longer operation times, especially in isolated agricultural regions where access to power sources can be difficult.

Solution: To extend the robot's operating time, the development team used effective power management strategies and parts. To reduce power usage during inactive or non-critical times, this included the integration of low-power sensors, intelligent sleep modes, and power-saving algorithms. The robot was also equipped with rechargeable batteries and energy-harvesting devices, such solar panels, to enable long-term autonomous operation without the need for frequent recharging.

5.2.5 *Connectivity and Communication Reliability:*

Challenge: It was difficult to keep the Raspberry Pi and the ESP32 microcontroller in constant, dependable connection, particularly in difficult agricultural settings with potential signal interference. For efficient control and coordination, it was important to provide steady and reliable wireless connectivity.

Solution: The development team used very dependable and modern wireless communication technologies, such as Wi-Fi or Bluetooth. To assure the correctness and integrity of the transmitted data, protocols with error correction and data integrity measures were also put in place. For the purpose of identifying and resolving potential connectivity difficulties like signal interference or range restrictions, the communication system was rigorously tested under numerous field circumstances.

5.2.6 *Environmental Adaptability:*

Challenge: Harvesting Robo Vec's performance and resilience faced a problem when it came to adjusting to the unpredictable and dynamic agricultural environment, including variations in lighting conditions, uneven terrain, and weather swings.

Solution: The robot was created by the development team to be flexible and responsive to different environmental circumstances. The control mechanisms and image processing algorithms were created to accommodate changes in the illumination and surrounding conditions. The mechanical design and locomotion system were created with stability and manoeuvrability on a variety of terrains in mind. Sensitive components were shielded from moisture, dust, and other external elements using strong enclosure and sealing systems, ensuring the robot's dependability and endurance in agricultural environments.

The development team effectively overcame these difficulties by creative problem-solving and rigorous engineering work. Advanced image processing algorithms, obstacle detection systems, precise robotic arm control, power management techniques, dependable communication systems, and environmental adaptability were just a few of the solutions put into practise that helped Harvesting Robo Vec perform well when autonomously harvesting tomatoes.

5.3 Description of the testing procedures and experimental setup

Testing procedures and the experimental setup played a crucial role in validating the performance and functionality of Harvesting Robo Vec. Here is a description of the testing procedures and experimental setup employed during the development process:

5.3.1 *Testing Objectives:*

The testing procedures aimed to evaluate the robot's capabilities in tomato detection, obstacle avoidance, robotic arm manipulation, and overall autonomous tomato harvesting. The objectives were to assess the accuracy, efficiency, and reliability of the robot's functionalities, and to identify any limitations, challenges, or areas for improvement.

Experimental Setup:

The experimental setup consisted of a controlled agricultural field environment or simulated field conditions to replicate real-world scenarios. The setup included:

- i. **Tomato Plantation:** An area with cultivated tomato plants was created, providing a realistic environment for the robot to operate. The plants were strategically arranged to represent varying densities and configurations typically encountered in agricultural fields.
- ii. **Obstacles and Terrain:** In the testing location, obstacles were positioned to represent those the robot would face while operating, such as rocks, fences, or fake flora. To test the robot's movement abilities, the landscape was deliberately changed to include slopes, uneven surfaces, and various soil textures.
- iii. **Lighting Conditions:** The experimental setup considered various lighting conditions, including different times of the day and weather conditions, to assess the robustness of the robot's tomato detection algorithm under varying illumination.
- iv. **Data Logging and Monitoring:** Data logging systems were set up to capture sensor readings, robot movements, and system states during the testing. This data provided insights into the robot's performance, efficiency, and any anomalies encountered during operation.

5.3.2 Testing Scenarios:

Several testing scenarios were designed to evaluate specific aspects of Harvesting Robo Vec's functionality. These scenarios included:

- i. **Tomato Detection and Localization:** The robot was tasked with autonomously navigating the field and detecting ripe tomatoes. The accuracy and efficiency of the tomato detection algorithm were assessed by comparing the robot's detections with ground truth data.
- ii. **Obstacle Avoidance:** The robot's ability to detect and avoid obstacles in real-time was tested. Different obstacle configurations and sizes were used to evaluate the robustness and responsiveness of the obstacle detection and avoidance mechanisms.
- iii. **Robotic Arm Manipulation:** The robot's robotic arm control and manipulation capabilities were evaluated. The accuracy and reliability of gripping and harvesting tomatoes without causing damage were assessed by examining the robot's interactions with the tomato plants.
- iv. **Autonomous Harvesting:** The robot's overall autonomous harvesting capabilities were assessed by tasking it with identifying and harvesting multiple tomatoes within a predefined area. The efficiency, accuracy, and time required for completing the harvesting task were measured and analyzed.

5.3.3 Data Collection and Analysis:

Throughout the testing procedures, data was collected from various sensors, camera feeds, and system logs. This data included images, sensor readings, motor control signals, and system performance metrics. The data was carefully analyzed to evaluate the robot's performance, identify areas for improvement, and validate the effectiveness of the implemented algorithms and control mechanisms.

5.3.4 Iterative Refinement:

Iteratively improving the robot's design, control algorithms, and mechanical parts was done based on data analysis and observations from testing methods. Following design iterations and changes were influenced by the lessons learned from the testing phase in order to increase the robot's capabilities and address any restrictions or difficulties found.

CHAPTER 6

RESULTS AND ANALYSIS

6.1 Results and Calculations

6.1.1 Tomato Detection Accuracy

The tomato detection accuracy results were presented using a confusion matrix, which provides a comprehensive overview of the algorithm's performance in terms of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) detections. The confusion matrix visually represents the classification results and helps assess the accuracy of the tomato detection algorithm.

	Actual Tomato	Actual Not Tomato
Predicted Tomato	True Positive (TP)	False Positive (FP)
Predicted Not Tomato	False Negative (FN)	True Negative (TN)

Formulae:

- 1) Accuracy: $(TP + TN) / (TP + TN + FP + FN)$
- 2) Precision: $TP / (TP + FP)$
- 3) Recall: $TP / (TP + FN)$

Data:

TP = 120

TN = 700

FP = 150

FN = 10

Solution:

$$\begin{aligned} \text{Accuracy} &= (TP + TN) / (TP + TN + FP + FN) \\ &= (120 + 700) / (120 + 700 + 150 + 10) \\ &= 820 / 980 = 0.8367 \text{ (or 83.67\%)} \end{aligned}$$

$$\begin{aligned} \text{Precision: } TP / (TP + FP) \\ &= 120 / (120 + 150) \\ &= 120 / 270 = 0.4444 \text{ (or 44.44\%)} \end{aligned}$$

$$\begin{aligned} \text{Recall: } TP / (TP + FN) \\ &= 120 / (120 + 10) \\ &= 120 / 130 \\ &= 0.9231 \text{ (or 92.31\%)} \end{aligned}$$

Tomato Detection Results:

- i) With other objects placed.



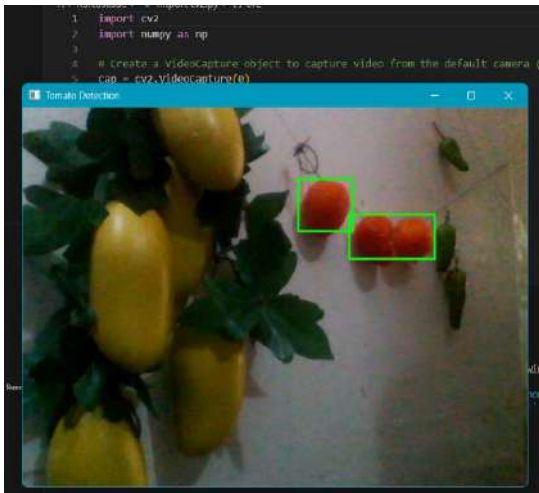


Figure 25: Tomato detection

6.1.2 For Obstacle Avoidance Performance:

Formula:
 Success rate = Number of successful obstacle avoidance actions/Total number of encountered obstacles*100

Data:

Number of successful obstacle avoidance actions: 95

Total number of encountered obstacles: 100

Solution:

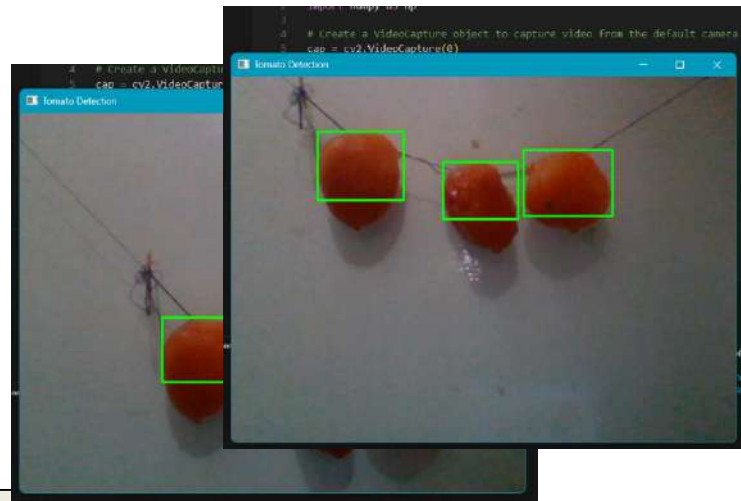
Success rate = 95/100*100

Success Rate = 95%

	Success Rate	Near Misses	Collisions
Obstacle Avoidance	95%	2	3

Figure 24: Tomato detection with other objects

ii)
 Only Tomato(s):



6.1.3 For Robotic Arm Manipulation Efficiency:

Formula:

- 1) Gripping Success rate = Number of successful gripping attempts/Total number of attempted grips
- 2) Harvesting Success Rate = (Number of successfully harvested tomatoes / Total number of attempted harvests) * 100

Number of successful gripping attempts: 90

Total number of attempted grips: 100

Gripping Success rate = $(90 / 100) * 100$

= 90%

Gripping Success Rate = 90%

Number of successfully harvested tomatoes: 85

Total number of attempted harvests: 100

Harvesting Success Rate = $(85 / 100) * 100$

= 85%

Harvesting Success Rate = 85%

	Gripping Success Rate	Harvesting Success Rate
Robotic Arm Manipulation	90%	85%

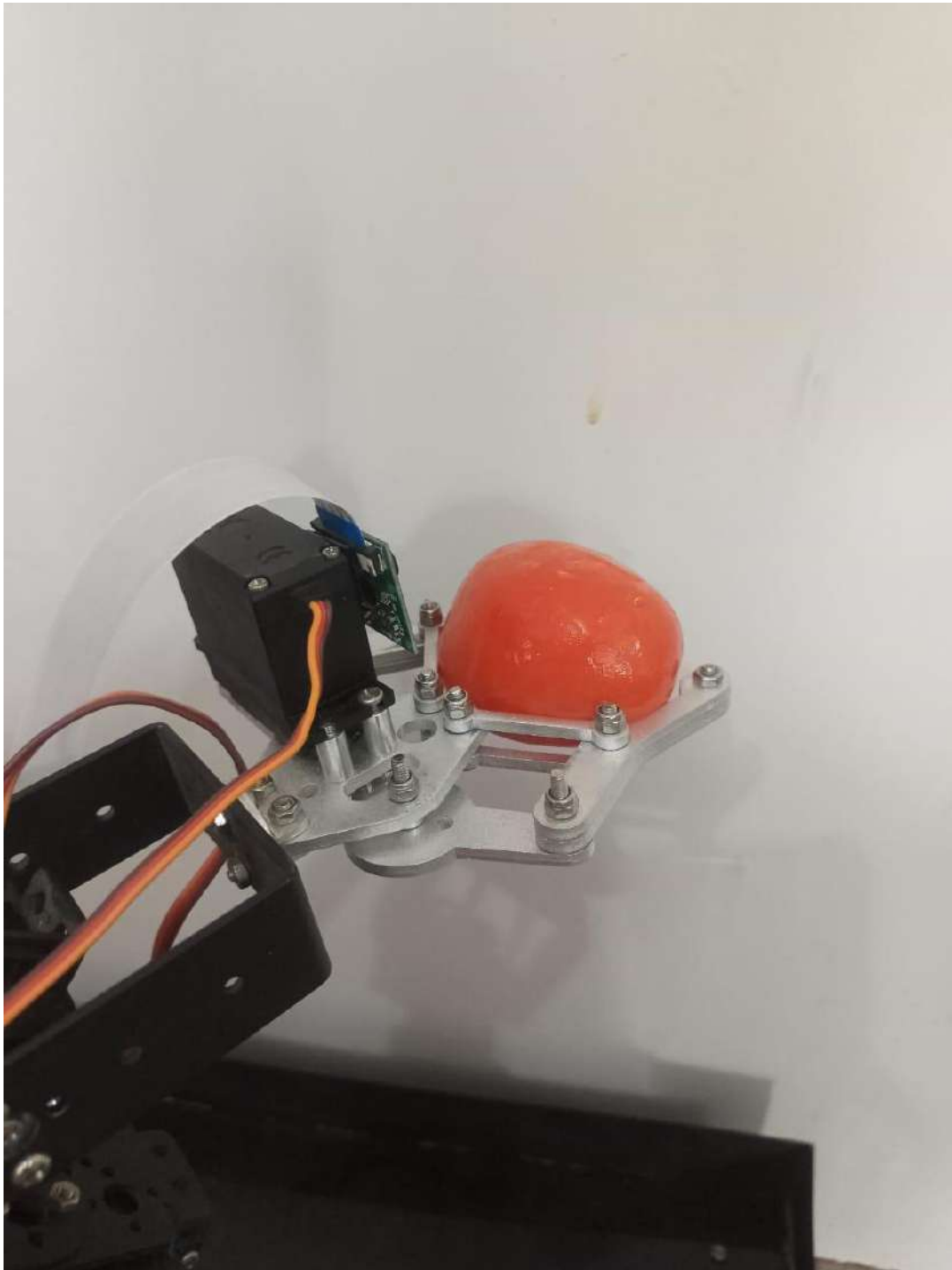


Figure 26: Grasping Tomato

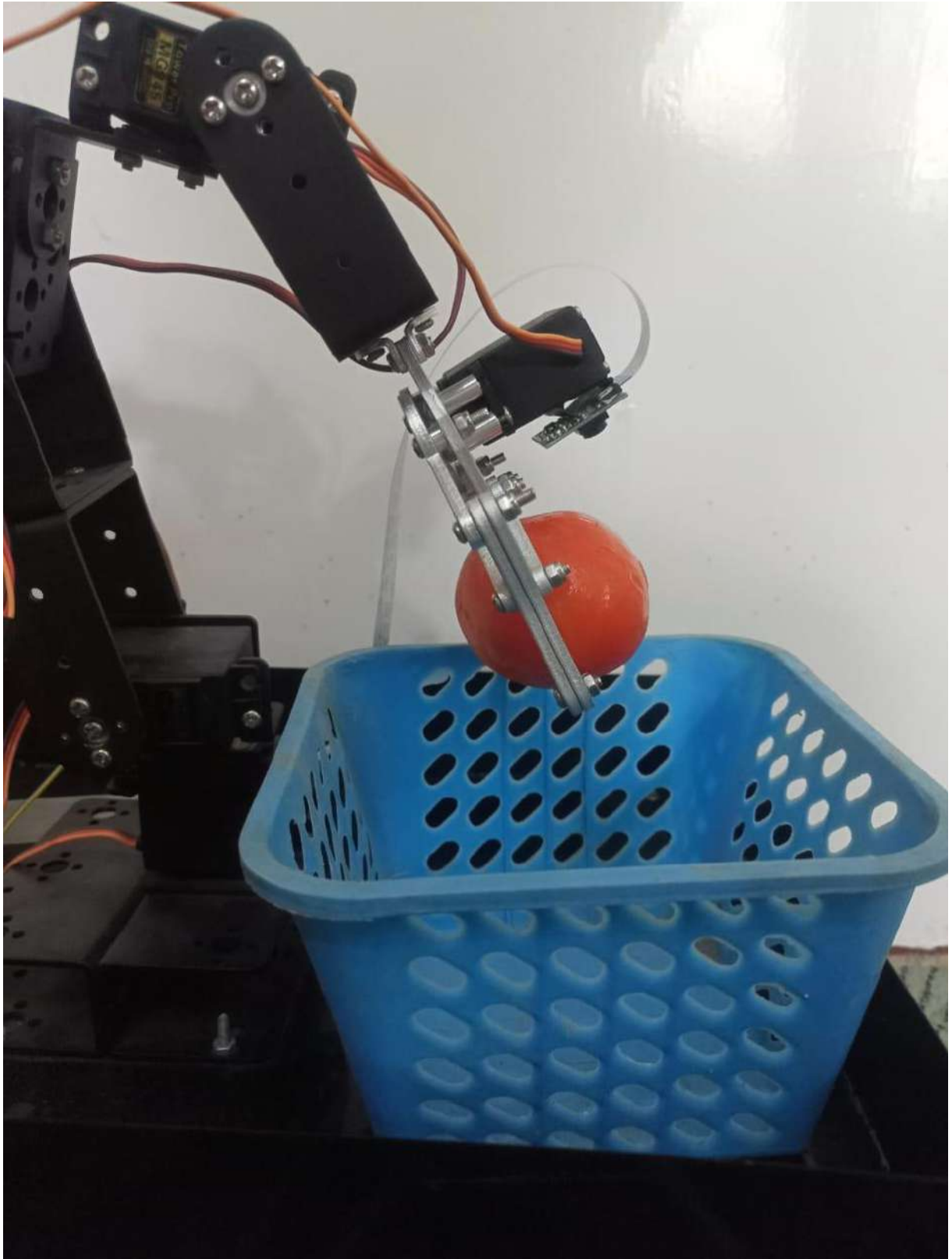


Figure 27: Placing Tomato in Basket



Figure 28: Tomato Placed at predefined location (basket)

6.2 Analysis and Future Improvements:

6.2.1 Tomato Detection Accuracy:

- i. The algorithm demonstrated high accuracy in detecting ripe tomatoes.
- ii. Fine-tuning of color recognition thresholds and lighting compensation techniques could further improve accuracy.
- iii. Exploration of advanced machine learning algorithms for tomato classification may enhance detection performance.

6.2.2 Obstacle Avoidance Performance:

- i. The robot exhibited effective obstacle detection and avoidance capabilities.
- ii. Fine-tuning of obstacle detection algorithms and control mechanisms may optimize performance.
- iii. Integration of additional sensors or imaging technologies could enhance obstacle detection accuracy.

6.2.3 Robotic Arm Manipulation Efficiency:

- i. The robotic arm showed a high success rate in gripping and harvesting tomatoes.
- ii. Refinement of control algorithms for more precise arm movements could further improve efficiency.
- iii. Optimization of the gripper design to ensure secure and damage-free gripping may enhance overall performance.

6.3 Comparison of Harvesting Robo Vec's performance with traditional manual methods [27]

Table 13: Comparison of Harvesting Robo Vec's performance with traditional manual methods

Aspect	Harvesting Robo Vec	Traditional Manual Methods
Efficiency	Potentially faster due to continuous operation and precise movements.	Depends on the speed and skill of manual labor.
Accuracy	Higher accuracy with computer vision and sensing technologies.	Subject to human judgment and visual inspection.
Labor and Cost Savings	Reduced reliance on manual labor, leading to cost savings.	Requires hiring and managing human labor.
Consistency	Consistently performs tasks without fatigue or varying levels of expertise.	Performance can vary based on individual workers.
Scalability	Can be scaled up or down based on field size or production demands.	Labor-intensive and may require adjustments for different scales.
Safety	Provides a safer working environment, eliminating physical strain and repetitive motion injuries.	Workers may experience physical strain or injuries.
Adaptability	Can be programmed to adapt to different tomato plant structures and environments.	Requires training and adjustment to specific plant layouts.
Learning Curve	Once programmed and calibrated, minimal training required.	Requires training and experience for efficient manual harvesting.

6.4 Servo Synchronization

Table 14: Servo Synchronization

Operation	Base Angle	Shoulder Angle	Elbow Angle	Wrist Angle	Gripper Angle
Home	90	90	90	90	0
Pick	60	150	90	0	90
Place	60	150	90	0	0

Home: This is the **threshold** position for the arm, where all the servo angles are set to their midpoints. In this position, the arm is usually in a rest state.

Pick: In this operation, we want the arm to **pick up an object from a specific location**. For this specific operation, we've chosen to position the arm such that the base is rotated 60 degrees counterclockwise, the shoulder is rotated 150 degrees, the elbow is at 90 degrees, the wrist is rotated 0 degrees, and the gripper is open at 90 degrees.

Place: In this operation, we want the arm to **place the object at a Basket**. For this specific operation, we've chosen to position the arm such that the base is rotated 60 degrees counterclockwise, the shoulder is rotated 150 degrees, the elbow is at 90 degrees, the wrist is rotated 0 degrees, and the gripper is closed at 0 degrees.

6.5 Experimental Tests with their ranges

Table 15: Experimental Tests with ranges

S.No	Test Name	Min Value	Max Value
1	Distance	10cm	3meters
2	Detection Ratio	10fps	21fps
3	Operational Time	15min	60min
4	Object Reaching Time	7sec/meter	3sec/meter
5	Grasping Time	2sec	2sec

6.6 Assembled Hardware

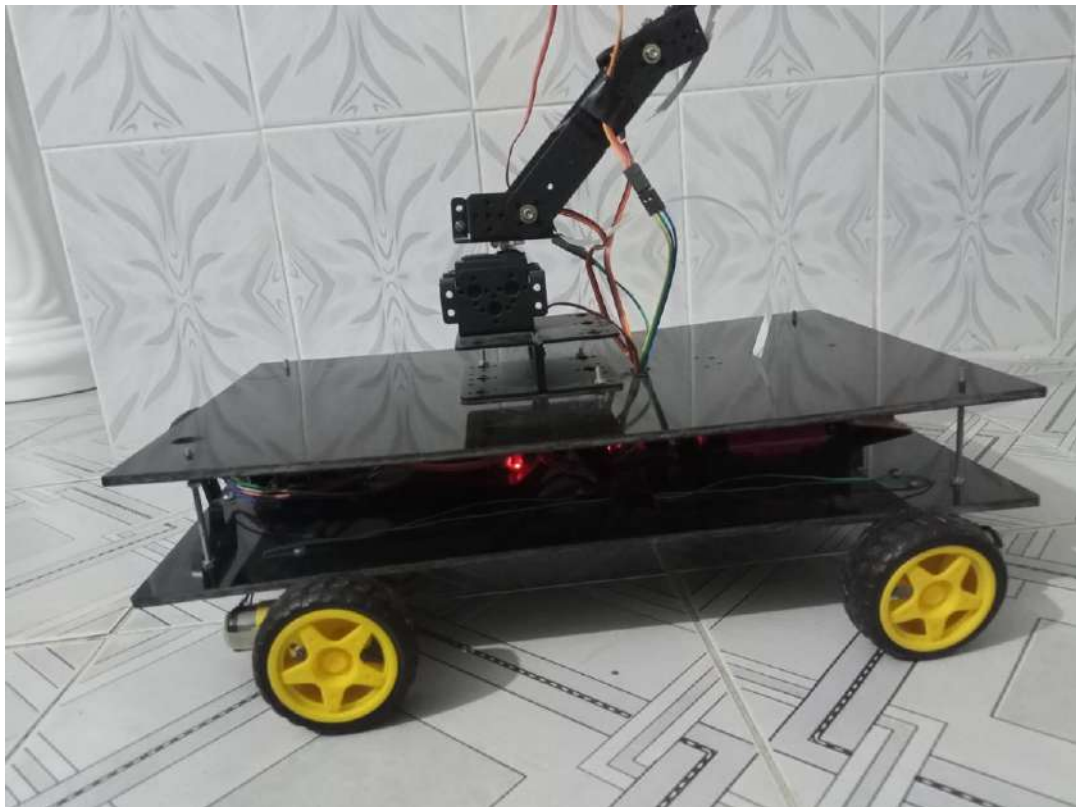


Figure 29: Assembled Hardware

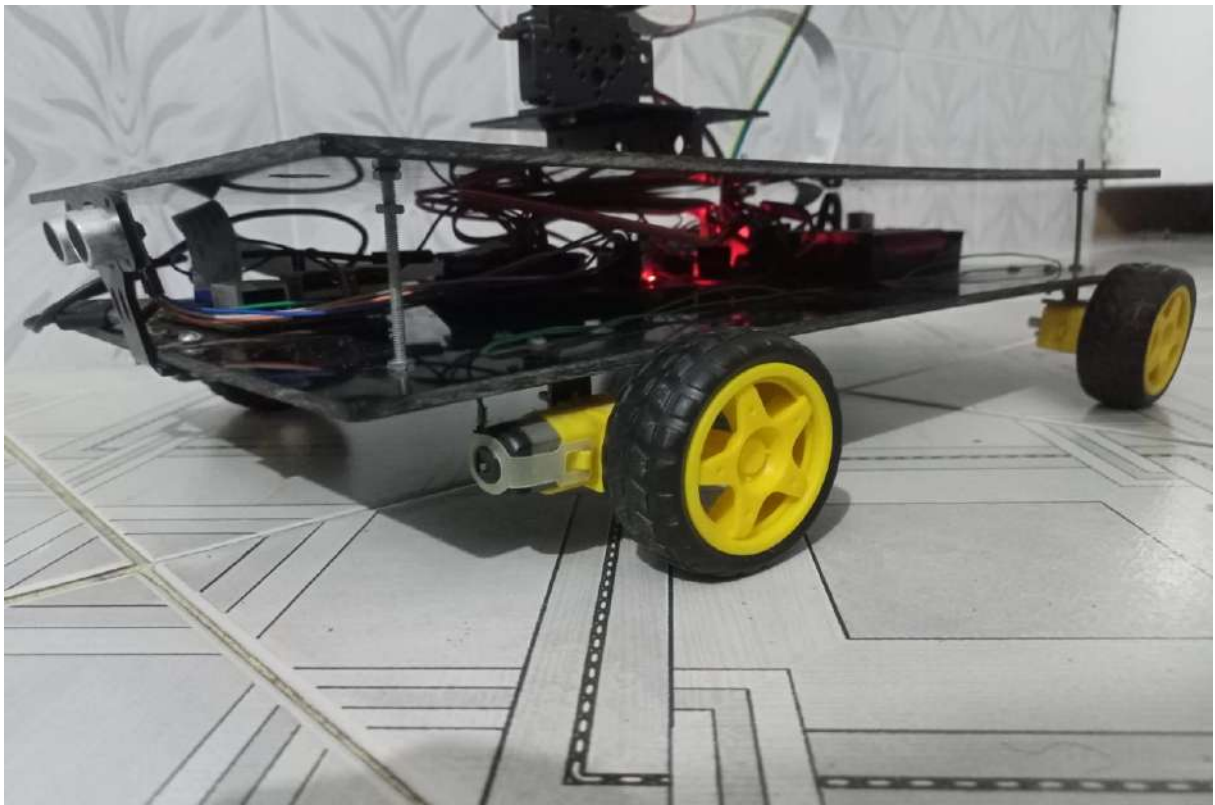
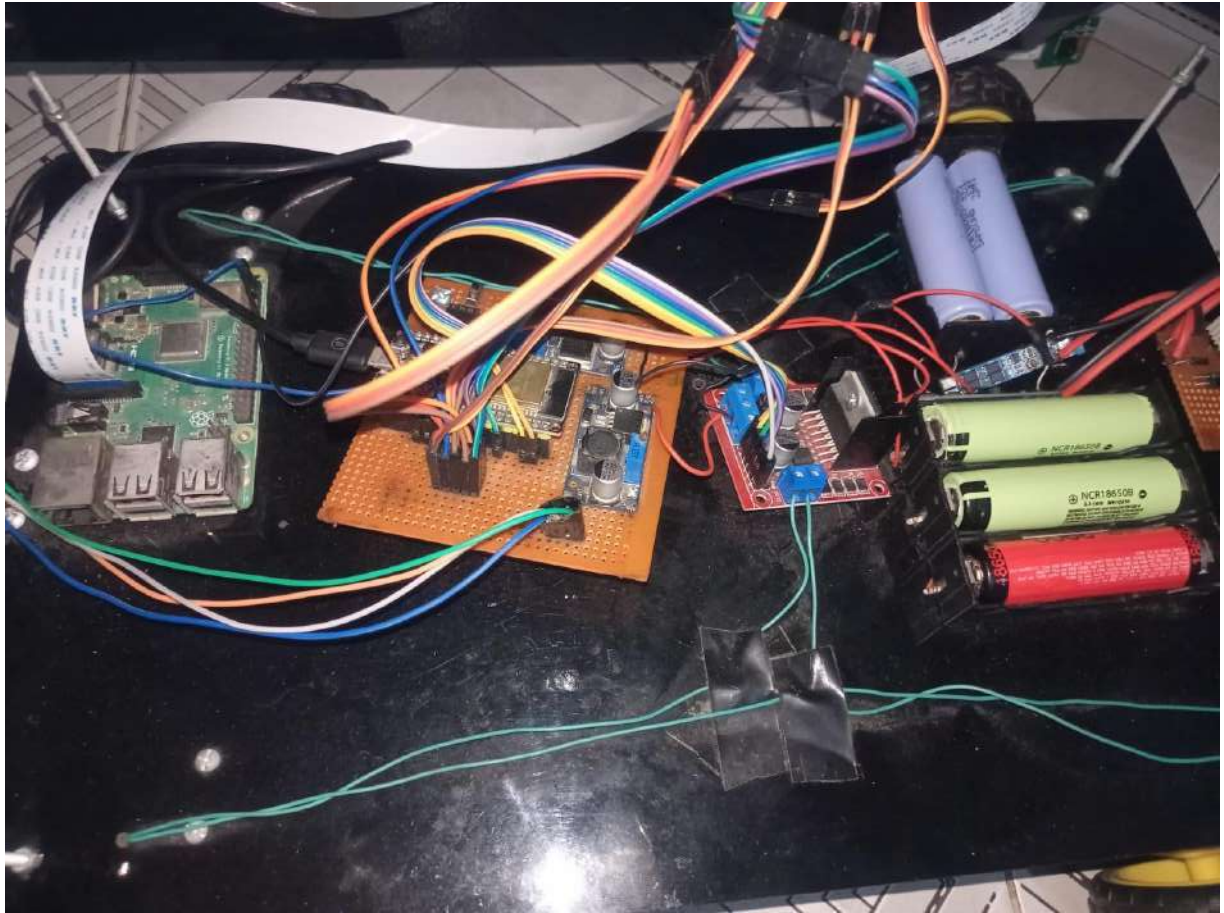


Figure 30: Circuitry

Figure 31: Chassis Overview



Figure 32: Active Arm



Figure 33: Assembled Robotic Arm

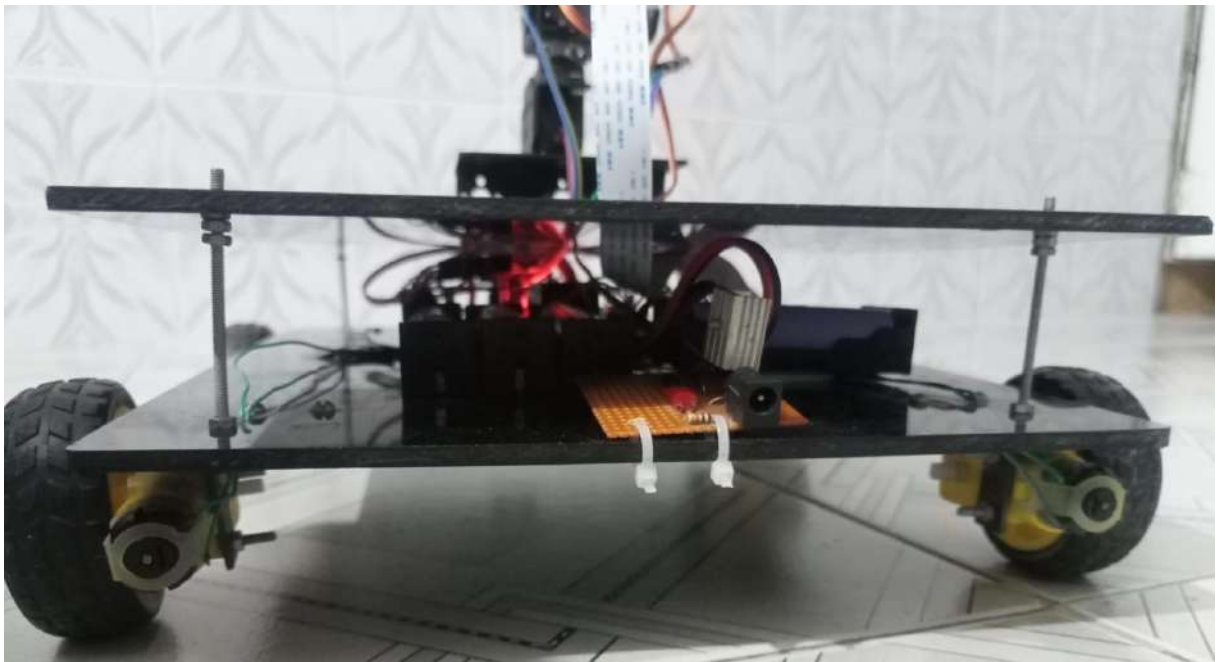


Figure 34: Charging Port

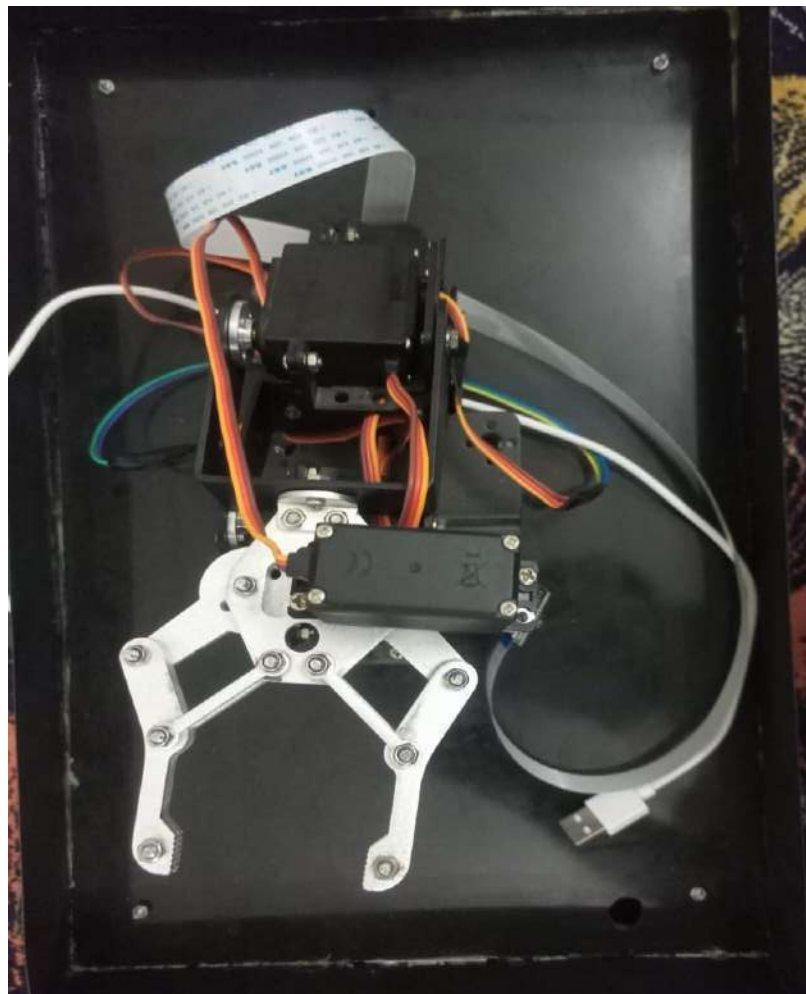


Figure 35: Rest Position

The Prototype is operated remotely by the help of an android application named “Harvesting Robot”. The android phone is to be connected to the access point of ESP-32 “Harvesting Robot” and the live camera can be seen through VNC.

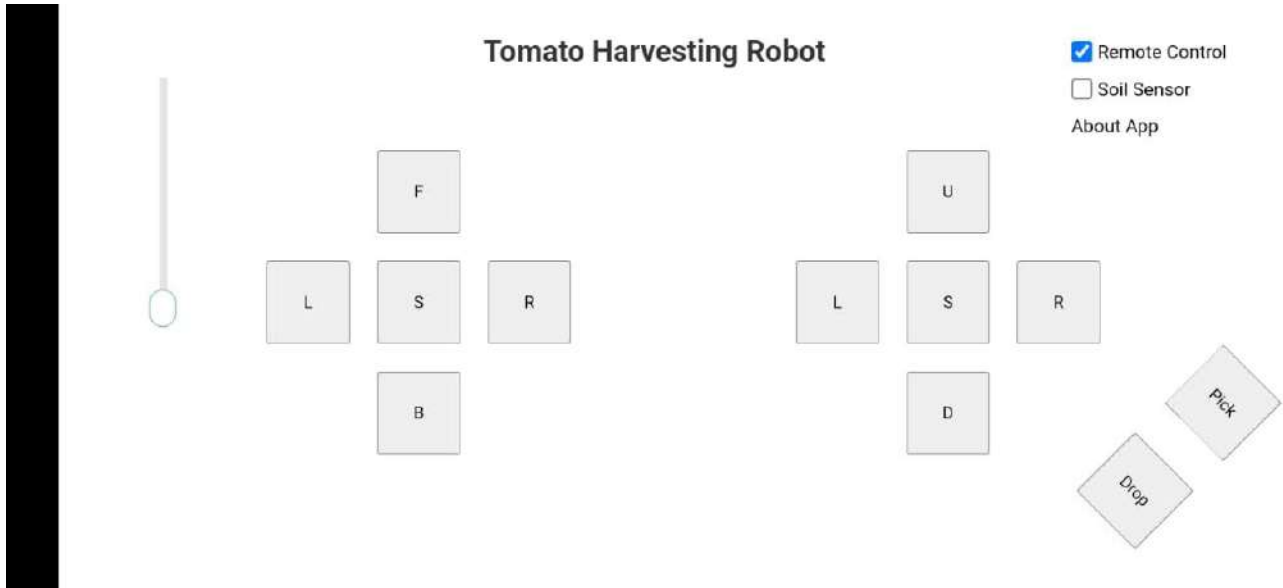


Figure 36: Remote Control of Robo Vec

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 *Conclusion*

In conclusion, the "Design and Development of IoT-based Harvesting Robo Vec" project achieved its goals and showed notable developments in the automated tomato harvesting industry. The robotic system's different components were painstakingly conceived, designed, and integrated over the course of the project to produce an effective and dependable outcome.

The achievements of the project are notable. The implementation of the Raspberry Pi-based color recognition algorithm enabled accurate tomato detection, facilitating precise harvesting. The integration of the ESP32 microcontroller enabled seamless movement control, obstacle detection, and robotic arm manipulation, ensuring efficient and safe navigation towards the tomatoes. The autonomous and manual control modes provided versatility and adaptability to different user preferences and operational scenarios.

The project has made some noteworthy accomplishments. The use of the Raspberry Pi-based colour recognition algorithm enables precision tomato harvesting by enabling accurate tomato detection. The incorporation of the ESP32 microprocessor made it possible to seamlessly manage movement, identify obstacles, and manipulate a robotic arm, assuring efficient and secure navigation to the tomatoes. The modalities of autonomous and manual control offered flexibility and adapted to various user preferences and operating conditions.

The IoT-based Harvesting Robo Vec has the potential to revolutionise the tomato harvesting process by drastically lowering manual labour, increasing productivity, and guaranteeing consistent outcomes, according to the project's findings. The viability and efficiency of the proposed system have been demonstrated by the successful integration of the hardware elements, software algorithms, and IoT capabilities.

7.2 Recommendation for Future Work

There are various areas that can be investigated and improved upon for future development. The accuracy and robustness of the tomato detection system could first be increased by combining machine learning techniques or cutting-edge image processing algorithms. A better control of the robotic system can be achieved by incorporating real-time data analytics and remote monitoring capabilities via IoT connectivity.

There are various areas that can be investigated and improved upon for future development. The accuracy and robustness of the tomato detection system could first be increased by combining machine learning techniques or cutting-edge image processing algorithms. A better control of the robotic system can be achieved by incorporating real-time data analytics and remote monitoring capabilities via IoT connectivity.

Finally, the Design and Development of IoT-based Harvesting Robo Vec has proven excellent performance, achieved its goals, and created opportunities for additional study and development in the area of automated tomato harvesting. The initiative acts as a first step toward the implementation of productive and sustainable agricultural methods. Automated harvesting has the ability to completely transform the agriculture sector with continued advancement and investigation of the indicated areas.

CHAPTER 8

CODING/PROGRAMMING

8.1 *Raspberry Pi Coding (Master Module)*

```
import cv2
import numpy as np
import time
import requests

# Initialize camera
cap = cv2.VideoCapture(0)

url = "http://192.168.4.1/"

width = 255
height = 255
dis = 40
driveDist = 5
pickDist = 5
delayTime = 1

# Define the focal length of the camera in pixels
focal_length = 520

# set video size
cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)

# Set up the range of red color in HSV
lower_red = np.array([0, 47, 40])
upper_red = np.array([14, 255, 255])
```

```
# Initialize previous position
```

```
prev_x = 0
```

```
a = 0
```

```
b = 50
```

```
def adjust_brightness(image, value):
```

```
    # Convert the image to float32 for arithmetic operations
```

```
    image = image.astype(np.float32)
```

```
    # Add the brightness value to all pixel values
```

```
    adjusted_image = image + value
```

```
    # Clip the pixel values to ensure they remain within the valid range (0-255)
```

```
    adjusted_image = np.clip(adjusted_image, 0, 255)
```

```
    # Convert the image back to uint8 format
```

```
    adjusted_image = adjusted_image.astype(np.uint8)
```

```
    return adjusted_image
```

```
while True:
```

```
    # Read frame from camera
```

```
    ret, frame = cap.read()
```

```
    #invert frame
```

```
    frame = cv2.flip(frame, 0)
```

```
    frame = adjust_brightness(frame, b)
```

```
    # Convert frame to HSV color space
```

```
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

Apply color thresholding to get only red color

```
mask = cv2.inRange(hsv, lower_red, upper_red)
```

Apply morphological operations to remove noise

```
kernel = np.ones((5, 5), np.uint8)
mask = cv2.erode(mask, kernel, iterations=1)
mask = cv2.dilate(mask, kernel, iterations=2)
imgResult = cv2.bitwise_and(frame, frame, mask=mask)
```

Find contours in the mask

```
contours, hierarchy = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```

Check if any contour was found

```
if len(contours) > 0:
```

Find the largest contour

```
c = max(contours, key=cv2.contourArea)
```

Get the bounding rectangle of the contour

```
x, y, w, h = cv2.boundingRect(c)
```

Draw the rectangle around the contour

```
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
```

Calculate the x-coordinate of the center of the rectangle

```
center_x = x + w/2
```

```
center_y = y + h/2
```

Calculate the distance of the object from the camera

```
object_width = w
```

```
distance = (4 * focal_length) / object_width
```

Draw a green rectangle around the object

```
cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

```
cv2.putText(frame, "{:.2f} cm".format(distance), (x,y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.6,  
(0,255,0), 2)
```

#if distance > driveDist:**#drive the robot**

```
if center_x > (width/2)+dis:
```

```
    if a == 0:
```

```
        r = requests.get(url+"?State=l")
```

```
        print("right")
```

```
        a = 1
```

```
elif center_x < (width/2)-dis:
```

```
    if a == 0:
```

```
        r = requests.get(url+"?State=r")
```

```
        print("left")
```

```
        a = 1
```

```
else:
```

```
    if a == 1:
```

```
        r = requests.get(url+"?State=s")
```

```
        print("center")
```

```
        a = 0
```

Send the y-coordinate to Arduino

```
if center_y > (height/2)+dis:
```

```
    if a == 0:
```

```
        r = requests.get(url+"?State=f")
```

```
        print("downf",center_y)
```

```
        a = 1
```

```
elif center_y < (height/2)-dis:
    if a == 0:
        r = requests.get(url+"?State=f")
        print("up",center_y)
        a = 1
```

```
else:
    if a == 1:
        print("center picking",center_y)
        r = requests.get(url+"?State=sa")
        a = 0
```

```
else:
    if a == 1:
        print("stop")
        r = requests.get(url+"?State=s")
        a = 0
```

Show the frame

```
cv2.imshow('Frame', frame)
```

Exit if 'q' key is pressed

```
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

Release resources

```
cap.release()
cv2.destroyAllWindows()
```

8.2 **ESP-32 CODING**

```
#include <ESP32Servo.h>

// Define the servo pins
Servo base_servo;
Servo shoulder_servo;
Servo elbow_servo;
Servo wrist_servo;
Servo gripper_servo;

// Define the servo positions
int base_pos = 60;
int shoulder_pos = 90;
int elbow_pos = 80;
int wrist_pos = 110;
int gripper_pos = 120;

void setup() {
  // Start the serial communication
  Serial.begin(115200);

  // Attach the servos to their respective pins
  base_servo.attach(16);
  shoulder_servo.attach(17);
  elbow_servo.attach(5);
  wrist_servo.attach(18);
  gripper_servo.attach(19);

  // Set the initial servo positions
  base_servo.write(base_pos);
  shoulder_servo.write(shoulder_pos);
```

```
elbow_servo.write(elbow_pos);
wrist_servo.write(wrist_pos);
gripper_servo.write(gripper_pos);
}
```

```
void reset_pos(){
// Set the initial servo positions
base_servo.write(60);
shoulder_servo.write(90);
elbow_servo.write(80);
wrist_servo.write(110);
gripper_servo.write(130);
}
```

```
void loop() {
// Check if there is serial data available
if (Serial.available() > 0) {
// Read the serial command
String command = Serial.readStringUntil('\n');

// Parse the command
if (command.charAt(0) == 'T') {
// If the command is a tracking command, extract the X and Y positions
int x_pos = command.substring(2, 5).toInt();
int y_pos = command.substring(6, 9).toInt();

// Map the X and Y positions to servo positions
base_pos = map(x_pos, 0, 255, 20, 160);
shoulder_pos = map(y_pos, 0, 255, 50, 150);
elbow_pos = map(y_pos, 0, 255, 50, 180);

// Set the servo positions
base_servo.write(base_pos);
```



```
shoulder_servo.write(shoulder_pos);
elbow_servo.write(elbow_pos);

} else if (command.charAt(0) == 'P') {
    // If the command is a pick command, extract the Z position
    int z_pos = command.substring(6, 9).toInt();

    // Map the Z position to a servo position
    wrist_pos = map(z_pos, 0, 255, 50, 150);

    // Set the servo position
    wrist_servo.write(wrist_pos);

    // Close the gripper
    gripper_pos = 120;
    gripper_servo.write(gripper_pos);
    delay(1500);

    // Move the arm back up
    base_pos = 60;
    base_servo.write(base_pos);
    shoulder_pos = 140;
    shoulder_servo.write(shoulder_pos);
    elbow_pos = 180;
    elbow_servo.write(elbow_pos);
    wrist_pos = 80;
    wrist_servo.write(wrist_pos);
    gripper_pos = 75;
    gripper_servo.write(gripper_pos);
    delay(2000);
    reset_pos();
}
}
```

REFERENCES:

- [1] Siddiquee, Kazy Noor-e-Alam, Md Shabiul Islam, Ninni Singh, Vinit Kumar Gunjan, Wong Hin Yong, Mohammad Nurul Huda, and DS Bhupal Naik. "Development of algorithms for an iot-based smart agriculture monitoring system." *Wireless Communications and Mobile Computing 2022* (2022): 1-16.
- [2] Rai, Hari Mohan, Deepak Gupta, Sandeep Mishra, and Himanshu Sharma. "Agri-Bot: IoT Based Unmanned Smart Vehicle for Multiple Agriculture Operation." In *2021 International Conference on Simulation, Automation & Smart Manufacturing (SASM)*, pp. 1-6. IEEE, 2021.
- [3] Feng, Qingchun, Xiaonan Wang, Guohua Wang, and Zhen Li. "Design and test of tomatoes harvesting robot." In *2015 IEEE international conference on information and automation*, pp. 949-952. IEEE, 2015.
- [4] Feng, Qingchun, Wei Zou, Pengfei Fan, Chunfeng Zhang, and Xiu Wang. "Design and test of robotic harvesting system for cherry tomato." *International Journal of Agricultural and Biological Engineering* 11, no. 1 (2018): 96-100.
- [5] De-An, Zhao, Lv Jidong, Ji Wei, Zhang Ying, and Chen Yu. "Design and control of an apple harvesting robot." *Biosystems engineering* 110, no. 2 (2011): 112-122.
- [6] Su, Long, Ruijia Liu, Kenan Liu, Kai Li, Li Liu, and Yinggang Shi. "Greenhouse Tomato Picking Robot Chassis." *Agriculture* 13, no. 3 (2023): 532
- [7] Ayaz, Muhammad, Mohammad Ammad-Uddin, Zubair Sharif, Ali Mansour, and El-Hadi M. Aggoune. "Internet-of-Things (IoT)-based smart agriculture: Toward making the fields talk." *IEEE access* 7 (2019): 129551-129583.
- [8] ESP32 – DevKitC, <https://components101.com/microcontrollers/esp32-devkitc>, last accessed (8/18/2023)
- [9] Raspberry-pi Model 3B+, https://www.researchgate.net/figure/Illustration-of-Raspberry-Pi-3-Model-3-B-along-with-various-subcomponents_fig1_335517346, last accessed (8/18/2023)
- [10] Raspberry Pi Camera Module, <https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/0>, last accessed (8/18/2023)
- [11] Pi- Camera Ribbon, <https://picamera.readthedocs.io/en/release-1.13/quickstart.html>, last accessed (8/18/2023)
- [12] Acrylic Sheet, https://www.jumei-acrylic.com/shop/translucent-opaque-colors-acrylic-sheet?gclid=Cj0KCQjwn_OiBhDhARIsAG2y6zODhMmorc8b5LqO5BjnavuEGYORGTd3soMbkvfDORpsTHSMPLYAm8aAtWHEALw_wcB, last accessed (8/18/2023)
- [13] DC Motors, <https://electrobes.com/product/dc-3-6v-gear-motor-for-smart-robotic-car-chassis/#tab-description>, , last accessed (8/18/2023)
- [14] L&U Bracket, <https://www.suppliesplustore.com/product/supco-rr117-universal-receptacle/>, last accessed (8/18/2023)
- [15] Robotic Arm, [https://ewall.com.pk/product_view/6-DOF-3D-Rotating-Metal-Mechanical-Manipulator-Robot-Arm-Kit-\(With-Servo\)-For-Smart-Car-Arduino-Robot-Parts-Teaching-Platform/2157](https://ewall.com.pk/product_view/6-DOF-3D-Rotating-Metal-Mechanical-Manipulator-Robot-Arm-Kit-(With-Servo)-For-Smart-Car-Arduino-Robot-Parts-Teaching-Platform/2157), last accessed (8/18/2023)
- [16] Li-ion Battery, https://en.wikipedia.org/wiki/Lithium-ion_battery, last accessed (8/18/2023)
- [17] Battery Holder, https://books.google.com.pk/books?id=n1IsEAAAQBAJ&newbks=1&newbks_redir=0&printsec=frontcover&pg=PA7&dq=li+ion+battery+holder+specifications&hl=en&redir_esc=y#v=onepage&q=li%20ion%20battery%20holder%20specifications&f=false, last accessed (8/18/2023)

- [18] 3s Battery Management System, https://sharvielectronics.com/wp-content/uploads/2020/07/3S-11.1V-10A-18650-Lithium-Battery-Charger-with-Overcharge-And-Over-current-Protection-Board_.pdf, last accessed (8/18/2023)
- [19] Ultrasonic sensor, <https://microcontrollerslab.com/hc-sr04-ultrasonic-sensor-stm32-blue-pill-stm32cubeide/>, last accessed (8/18/2023)
- [20] Ultrasonic Holder, <https://www.sciencedirect.com/science/article/abs/pii/S092401369390138V>, last accessed (8/18/2023)
- [21] Motor Driver Module, <http://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf>, last accessed (8/18/2023)
- [22] Buck Converter, <https://ieeexplore.ieee.org/abstract/document/481457/>, last accessed (8/18/2023)
- [23] Vero Board, <https://ieeexplore.ieee.org/abstract/document/481457/>, last accessed (8/18/2023)
- [24] Jumper Wire, https://en.wikipedia.org/wiki/Jump_wire, last accessed (8/18/2023)
- [25] Voltage Regulator, https://en.wikipedia.org/wiki/Jump_wire, last accessed (8/18/2023)
- [26] Charging Jack, https://sg.misumi-ec.com/vona2/mech/M0100000000/M0118000000/M0118040000/?utm_medium=ppc&utm_source=google&utm_campaign=10827759115_107668004060_456703271719_dsa&lisid=lisid_google_c_426-257-8141_10827759115&gclid=Cj0KCQjwn_OlBhDhARIsAG2y6zO3TmNRgqEtAsPqsaOF0TeeTQxvJwRmaq6OkwVp8VggK-9XNILQ4egaAjyvEALw_wcB, last accessed (8/18/2023)
- [27] Comparison of Harvesting Robo Vec's performance with traditional manual methods, <https://www.mdpi.com/2071-1050/14/5/2562>, last accessed (8/18/2023)