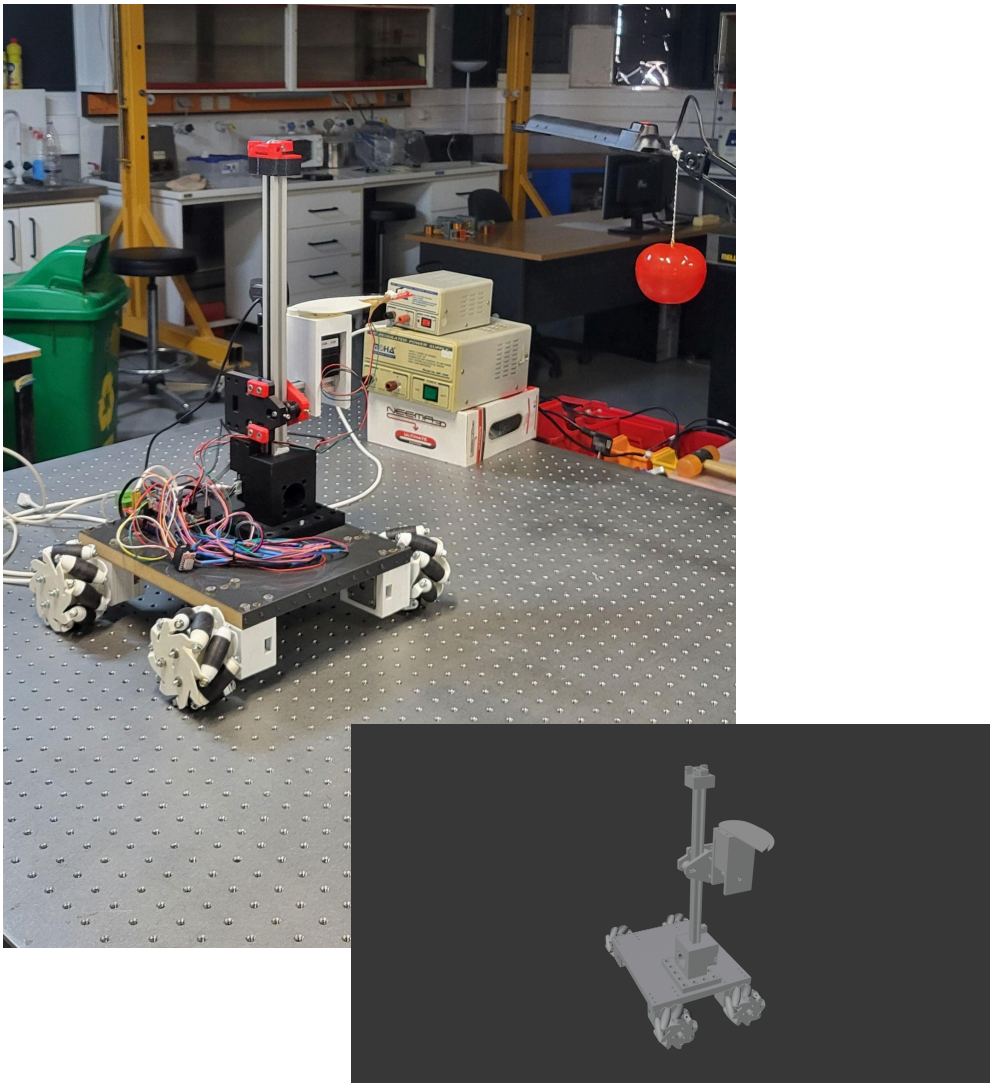
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κατασκευή Ρομποτικού Βραχίονα για Συγκομιδή Φυτών σε Θερμοκηπιακές Καλλιέργειες

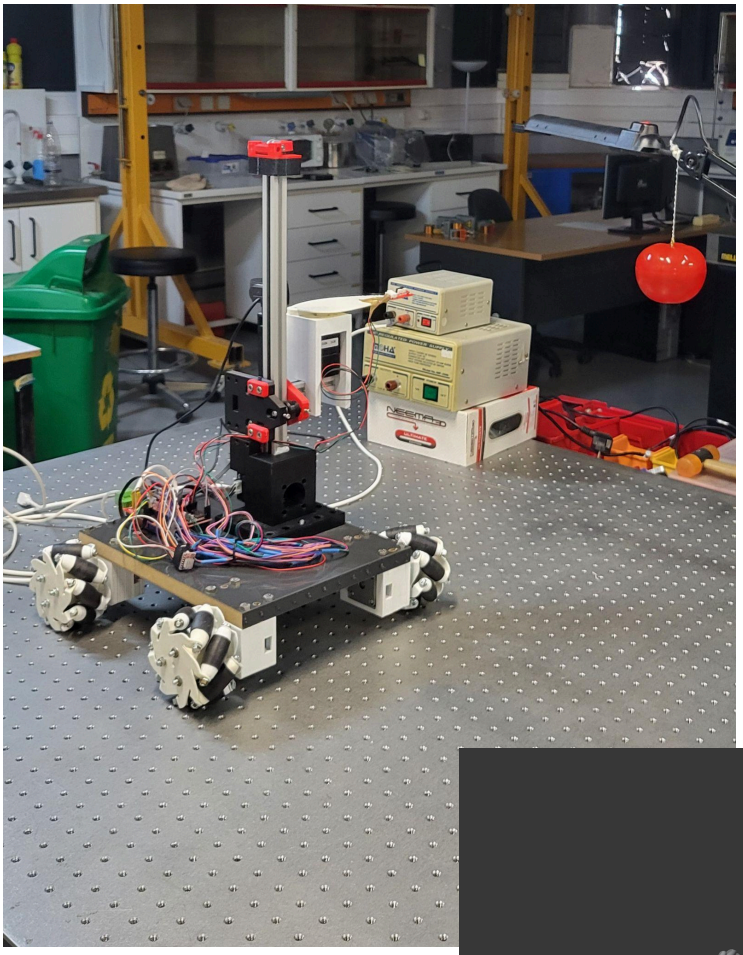


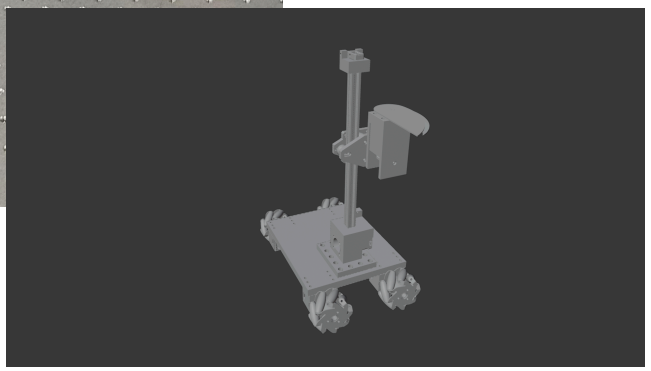
Επιμέλεια: Φλωράκης Θεόδωρος
Επιβλέπων καθηγητής: Κουτρούλης Ευτύχιος
Μέλη επιτροπής: Παπαευθυμίου Σπυρίδων, Γυφτάκης Κωνσταντίνος
Τόπος έκδοσης: Χανιά
Έτος ολοκλήρωσης: 2025

TECHNICAL UNIVERSITY OF CRETE

SCHOOL OF ELECTRICAL & COMPUTER ENGINEERING
DIPLOMA THESIS

Robotic Arm Construction for Plant Harvesting in Greenhouses





Supervised by: Theodoros Florakis
Supervisor: Eftychios Koutroulis
Committee Members: Spyridon Papaefthymiou, Konstantinos Gyftakis
Place of Publication: Chania

SCHOOL OF ELECTRICAL & COMPUTER ENGINEERING
DIPLOMA THESIS

Year of Completion: 2025

Περίληψη

Η αυτοματοποίηση της γεωργικής παραγωγής αποτελεί μια σημαντική πρόκληση στο σύγχρονο κόσμο. Οι νέες τεχνολογίες και εφαρμογές γεωργίας μπορούν να οδηγήσουν σε αύξηση της παραγωγής και της απόδοσης, καθώς και σε βελτίωση της ποιότητας των προϊόντων. Η αυτοματοποίηση έχει συνεισφέρει σε διάφορες εφαρμογές στη γεωργία, όπως η φύτευση, η συγκομιδή, η αναγνώριση ασθενειών, η εκτίμηση παραγωγής, ο ποιοτικός έλεγχος, η διαχείριση των υδάτων, η παρακολούθηση καλλιεργειών, ο έλεγχος εντομοκτόνων και η ποιότητα εδάφους και παρασιτοκτόνων. Μεταξύ αυτών των εφαρμογών, η συγκομιδή είναι η διαδικασία που έχει δεχτεί τη μικρότερη τεχνολογική ανάπτυξη για ικανοποιητική αυτοματοποίηση. Μέχρι και σήμερα, η πλειονότητα της συγκομιδής φρούτων και λαχανικών βασίζεται κατά κύριο λόγο σε χειροκίνητες τεχνικές. Τα ρομποτικά συστήματα, ικανά για ευφυή, αυτοματοποιημένη και επιλεκτική συγκομιδή μπορούν να συνεισφέρουν σημαντικά στον πρωτογενή τομέα. Στην παρούσα διπλωματική εργασία περιγράφονται η κατασκευή και προγραμματισμός ενός ρομποτικού βραχίονα για χρήση σε καλλιέργειες ντομάτας και πιπεριάς εντός θερμοκηπίων. Το σύστημα που υλοποιείται χρησιμοποιεί υπολογιστική όραση για να αναγνωρίζει ώριμους καρπούς με βάση το χρώμα, το μέγεθος και άλλες οπτικές ενδείξεις, επιτρέποντας ακριβείς αποφάσεις συγκομιδής. Για την επίτευξη του συγκεκριμένου σκοπού γίνεται χρήση αλγορίθμων μηχανικής μάθησης, οι οποίοι αναλύουν εκτεταμένες ποσότητες δεδομένων για να βελτιώσουν την απόδοση της πρόβλεψης, να βελτιστοποιήσουν τα χρονοδιαγράμματα συγκομιδής και να μειώσουν τη σπατάλη. Τα επιμέρους εξαρτήματα του ρομποτικού βραχίονα υλοποιούνται μέσω τρισδιάστατης εκτύπωσης και η τελική τους συναρμολόγηση συνθέτει το συνολικό αποτέλεσμα της κατασκευής. Με αυτόν τον τρόπο, ο ρομποτικός βραχίονας που υλοποιείται βελτιστοποιεί τη συγκομιδή ντομάτας και πιπεριάς, επιτρέποντας παράλληλα την εξυπνότερη λήψη αποφάσεων σε όλο τον κύκλο της καλλιέργειας.

Abstract

The automation of agricultural production constitutes a significant challenge in the modern world. Emerging technologies and agricultural applications can lead to increased productivity and efficiency, as well as improved product quality. Automation has contributed to various agricultural applications, including planting, harvesting, disease recognition, yield estimation, quality control, water management, crop monitoring, pesticide control, and soil and pest quality assessment. Among these applications, harvesting is the process that has seen the least technological advancement toward satisfactory automation. To this day, the majority of fruit and vegetable harvesting relies primarily on manual techniques.

Robotic systems capable of intelligent, automated, and selective harvesting can significantly contribute to the primary agricultural sector. This thesis presents the development and programming of a robotic arm designed for use in tomato and pepper cultivation within greenhouses. The implemented system utilizes computer vision to identify ripe fruits based on color, size, and other visual indicators, enabling precise harvesting decisions.

To achieve this goal, machine learning algorithms are employed to analyze vast amounts of data, enhancing predictive accuracy, optimizing harvesting schedules, and reducing waste. The individual components of the robotic arm are fabricated using 3D printing, and their final assembly results in the complete construction of the system. Through this approach, the developed robotic arm optimizes the harvesting process for tomatoes and peppers while simultaneously facilitating smarter decision-making throughout the entire cultivation cycle.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, Καθηγητή Κουτρούλη Ευτύχιο της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Πολυτεχνείου Κρήτης για την ανάθεση της διπλωματικής μου εργασίας όπως επίσης και για τη χρήσιμη καθοδήγηση του.

Επιπλέον, θα ήθελα να ευχαριστήσω πολύ τον Καθηγητή Παπαευθυμίου Σπυρίδωνα της Σχολής Μηχανικών Παραγωγής και Διοίκησης, μέλος της τριμελούς επιτροπής, για τη σημαντική βοήθειά του, την επιστημονική του υποστήριξη αλλά και για το χρόνο που αφιέρωσε για την επίλυση αποριών προκειμένου να ολοκληρωθεί η εν λόγω διπλωματική εργασία.

Ευχαριστώ πολύ τον Καθηγητή Γυφτάκη Κωνσταντίνο, ο οποίος δέχτηκε να είναι μέλος της τριμελούς επιτροπής αξιολόγησης της διπλωματικής μου εργασίας.

Τέλος, θα ήθελα να αποδώσω ιδιαίτερες ευχαριστίες στον κύριο Λουκάκη Κωνσταντίνο, υποψήφιο διδάκτορα της Σχολής Μηχανικών Παραγωγής και Διοίκησης του Πολυτεχνείου Κρήτης, για την πολύωρη ενασχόλησή του και την πολύτιμη προσφορά του στην υλοποίηση της συγκεκριμένης διπλωματικής εργασίας.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Professor Eftychios Koutroulis of the School of Electrical and Computer Engineering at the Technical University of Crete, for assigning me this thesis and for his valuable guidance throughout its development.

Furthermore, I would like to extend my heartfelt thanks to Professor Spyridon Papaefthymiou of the School of Production Engineering and Management, member of the thesis evaluation committee, for his significant assistance, scientific support, and the time he dedicated to addressing my inquiries, which were crucial for the completion of this thesis.

I am also deeply grateful to Professor Konstantinos Gyftakis for agreeing to serve as a member of the evaluation committee of my thesis.

Finally, I would like to express my special thanks to Mr. Konstantinos Loukakis, PhD candidate at the School of Production Engineering and Management at the Technical University of Crete, for his extensive involvement and invaluable contributions to the successful implementation of this thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction and Research Objectives

1.1 Introduction

This thesis focuses on the development of a robotic arm system designed for automated harvesting of tomatoes and peppers in controlled greenhouse environments. The robotic arm must be capable of identifying ripe tomatoes and peppers using computer vision systems and machine learning algorithms to distinguish between ripe, unripe, or damaged fruits, while simultaneously minimizing any potential crop damage. This is achieved through the development of appropriate mechanisms to regulate the force applied by the robotic arm during harvesting.

By implementing this system, automated and efficient fruit harvesting is achieved, significantly reducing the time required for collection compared to manual harvesting methods.

1.2 Objective of the Study

The objective of this thesis is to develop a robotic arm for use in tomato and pepper cultivation within greenhouses. These crops represent significant agricultural products, and their cultivation in greenhouses ensures production stability and product quality control.

This study aims to address challenges related to automation in greenhouse harvesting by designing an autonomous robotic system. The adoption of robotic arms in agriculture can lead to reduced production costs, increased harvesting efficiency and improved quality control of the final product. Through the integration of robotics, artificial intelligence, and machine learning, this research seeks to contribute to the modernization of agricultural harvesting techniques.

1.3 State of the Art

The use of robotic arms for plant harvesting represents one of the most significant research fields in agricultural automation, aiming to enhance efficiency and reduce costs. In recent years, extensive efforts have been made to develop such systems, which can replace traditional manual harvesting methods. The following sections outline the most important advancements in this domain, along with the challenges that remain

Historical Development and Research Approaches

The first generation of robotic harvesting arms primarily focused on the mechanical design and control mechanisms required to perform fundamental movements, such as lifting and grasping fruits. In the early 21st century, most systems relied on predefined paths and touch sensors to detect the presence of fruit, without advanced environmental perception. Early research primarily targeted crops such as tomatoes and strawberries, due to their relatively simple plant morphology.

With advancements in computer vision technologies, robotic systems began to exhibit increased flexibility. Studies such as those by Blok et al. [1], introduced vision-based fruit recognition systems that utilized color and shape detection, allowing robotic arms to identify fruits autonomously and adjust their movements based on fruit size and position. These techniques were successfully applied to more complex crops, such as apples and pears, which posed greater challenges due to the dense structure of trees and the irregular distribution of fruits.

Figure 1: Agriculture robotic arm for apple harvesting [2].

Advanced Computer Vision and Machine Learning Systems

In recent years, the use of machine learning and deep learning techniques has significantly transformed robotic harvesting. Research by Koirala et al. [3] demonstrated the effectiveness of neural networks in fruit detection, ripeness estimation, and quality prediction. These systems can distinguish fruits from the background with high accuracy, even in challenging environments where lighting and visibility conditions vary. Key features such as texture, brightness, and shape are analyzed to identify ripe fruits, minimizing the likelihood of harvesting unripe ones. A significant breakthrough in the field has been the development of three-dimensional (3D) vision systems. The study by Tang et al. [4] utilized 3D cameras and deep learning algorithms to guide robotic arms in three-dimensional fruit recognition and precise positioning for harvesting. This technology allowed robots to interact more efficiently with plants, reducing damage to both fruits and crops.

Applications and Technical Challenges

Despite advancements in robotic harvesting, full automation has not yet been widely adopted across all crops. One of the major challenges is the uncertainty in fruit positioning, especially in plants with dense foliage or unpredictable fruit distribution, such as olives and grapes. The development of robotic arms capable of harvesting fruits without causing surface damage remains an active research challenge.

Harvesting crops with delicate or fragile surfaces requires the design of robotic arms equipped with haptic feedback and precise force control. Studies such as those by Lehnert et al. [5] have explored the integration of flexible materials and soft robotics technologies, enabling robotic arms to adapt to the shape and fragility of fruits.

Conclusions and Future Directions

Research on robotic arms for plant harvesting has made significant progress; however, numerous challenges remain at the forefront of scientific inquiry. Future work is expected to focus on developing more adaptable systems by leveraging artificial intelligence and advanced vision algorithms. Additionally, future robotic arms should be designed to dynamically adjust to environmental conditions.

The vision of fully autonomous harvesting systems, capable of operating independently and efficiently, remains an active area of research and development, promising to revolutionize modern agriculture.

1.4 Structure of the Thesis

Chapter 2 introduces the role of computer vision and deep learning in robotic harvesting, with a particular focus on Convolutional Neural Networks (CNNs) and their application in fruit detection. It presents the YOLOv5 object detection algorithm, outlines the use of HSV color space for ripeness detection, and discusses the optimization algorithms—Stochastic Gradient Descent (SGD) and Adam—that were used during model training. It also describes the dataset creation process, the training and evaluation of YOLOv5 models.

Chapter 3 details the design process, components, and hardware used to construct the final version of the robotic arm.

Chapter 4 provides a comprehensive description of the hardware connections required for the robotic arm's assembly and operation.

Chapter 5 presents a detailed explanation of the code implementation used to develop the system.

Chapter 6 presents the final experimental results obtained from deploying the robotic system in a controlled environment.

Chapter 7 provides conclusions and recommendations for future research in the field of robotic automation in agricultural harvesting.

Chapter 2: Computer Vision and Deep Learning in Robotic Harvesting

2.1 Introduction

Computer vision, a key branch of Artificial Intelligence (AI), enables machines to interpret visual information. In agricultural automation, computer vision forms the basis of robotic perception, allowing systems to detect crops, assess ripeness, and operate autonomously within greenhouse environments.

Advancements in deep learning, particularly through Convolutional Neural Networks (CNNs), have significantly improved the capabilities of visual systems, making autonomous harvesting increasingly practical in real-world applications.

This chapter presents the visual perception system developed for this thesis, which includes CNN-based object detection using the YOLOv5 algorithm and HSV color filtering for ripeness evaluation. It also outlines dataset collection, model training, and performance analysis across various model configurations, offering a comprehensive overview of the visual intelligence supporting the robotic harvesting platform.

2.2 CNN-Based Vision Systems

Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision, especially in object recognition and localization tasks. They form the backbone of modern visual perception systems, offering a powerful framework for analyzing and interpreting image data [6]. CNNs consist of multiple layers that progressively extract increasingly abstract representations of visual input, such as edges, textures, and object shapes [7]. In the context of robotic harvesting, CNNs enable the system to identify target crops, distinguish them from background elements like leaves or stems, and evaluate ripeness by analyzing color or surface patterns [8].

CNNs operate by transforming raw image data through a series of operations that capture spatial hierarchies and relevant features. These learned features are then used by the network to identify the type and location of the target object within the frame. Importantly, CNNs are highly effective at handling variability in lighting, perspective, and occlusion—conditions that are frequently encountered in greenhouse or outdoor agricultural settings [9].

Training CNNs involves using large annotated datasets and optimizing model parameters through backpropagation, typically using algorithms like Stochastic Gradient Descent (SGD) or Adam—both of which are discussed in the following sections [10], [11]. Over multiple training epochs, the network refines its ability to recognize task-relevant features—such as the shape and color of ripe produce—leading to robust and reliable visual recognition under real-world conditions [12].

2.3 The YOLOv5 Algorithm

YOLOv5 (You Only Look Once, version 5) represents a state-of-the-art object detection algorithm widely recognized for its efficiency and accuracy in real-time applications [13]. Its capacity to process an entire image in a single forward pass renders it particularly suitable for embedded systems and robotics, where low latency and high responsiveness are essential. Given these characteristics, YOLOv5 was selected as the core detection framework for the vision module in this study's robotic harvesting platform.

The system was specifically trained to detect two agriculturally significant crops—tomatoes and peppers—selected for the unique visual characteristics they exhibit, such as color variation and shape diversity. Tomatoes are typically characterized by their bright red hue and rounded geometry, whereas peppers exhibit a broader range of shapes and often appear in

green variants. The model was trained to distinguish tomatoes and peppers from unripe produce.

YOLOv5 is available in multiple model configurations, including YOLOv5n (nano), YOLOv5s (small), YOLOv5m (medium), YOLOv5l (large), and YOLOv5x (extra-large) [14]. These configurations offer scalability in terms of computational complexity and detection performance. Lightweight variants such as YOLOv5n are optimized for speed and are suitable for deployment on edge devices with limited processing capacity. In contrast, larger models like YOLOv5l and YOLOv5x provide superior detection accuracy and feature representation, albeit at higher computational cost. The choice of model in this thesis was guided by the operational constraints and hardware capabilities of the robotic system, with an emphasis on achieving a reliable balance between real-time processing and detection precision.

Through its integration into the robotic platform, YOLOv5 enabled accurate, real-time identification of ripe tomatoes and peppers, forming the basis for subsequent manipulation and harvesting actions. Its deployment in this context underscores its suitability for precision agriculture applications, where reliable object detection is critical to system effectiveness [11].

2.4 Optimization Algorithms: Stochastic Gradient Descent and Adam

Stochastic Gradient Descent (SGD)

In this thesis, the Stochastic Gradient Descent (SGD) algorithm was used to train several YOLOv5 models. SGD was selected for its reliability and consistency during training, particularly in larger models where stability and precise gradient updates are essential for minimizing detection errors [10].

Adam (Adaptive Moment Estimation) Optimizer

The Adam (Adaptive Moment Estimation) optimizer was also employed in this study to train and compare against SGD-based models. Adam is known for its faster convergence and was tested on lightweight YOLOv5 variants to assess its impact on detection volume and learning efficiency under different model sizes [11].

2.5 HSV Color Space for Ripeness Detection

In addition to spatial features, color is a critical parameter in determining ripeness. The HSV (Hue, Saturation, Value) color model is widely used in computer vision for this purpose, as it separates chromatic content (hue) from image brightness (value), making it more robust to varying lighting conditions compared to the traditional RGB color space [10].

The HSV model comprises three components: Hue, which defines the color type (e.g., red, green), Saturation, which measures the intensity or purity of the color, and Value, which corresponds to brightness or lightness. By adjusting HSV thresholds, the system can isolate color ranges associated with ripeness. For instance, ripe tomatoes exhibit strong red hues, while unripe ones tend to appear green. HSV filtering enables the model to reliably distinguish between these states, improving both the accuracy and precision of the detection process [15].

An illustrative representation of the HSV model used in this work is shown in Figure 2 [16], which demonstrates the circular hue distribution along with the corresponding gradients of value and saturation.

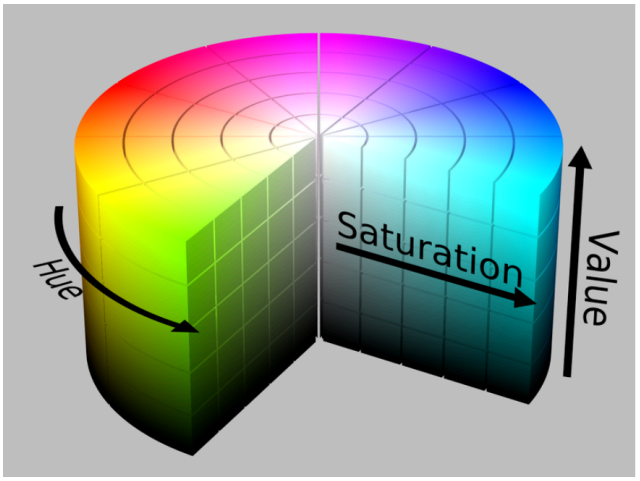


Figure 2: HSV color model.

2.6 Code Extension and Ripeness Filter

In the implementation of the YOLOv5-based detection system, additional logic was incorporated to support object identification and selection for harvesting tasks. This includes assessing the ripeness of each detected item and avoiding duplicates before adding them to the harvesting list. When a vegetable is detected, the algorithm checks if the total number of detections remains within a preset limit. It then calculates the bounding box center, crops the image accordingly, and evaluates its ripeness using HSV color analysis. Tomatoes are considered ripe if red pixels exceed a defined threshold, while peppers are judged based on green pixel ratios. If the item is both ripe and not already listed (based on distance from previous detections), it is added to the harvest queue. These extensions ensure that the system only collects valid and unique targets, optimizing both accuracy and harvesting efficiency. Examples of this ripeness evaluation logic are illustrated in Figures 3 through 5 [17], showing ripe and unripe detections based on color thresholds.

Figure 3: Example of ripe detection with a red color percentage of 67.45%.

Figure 4: Example of ripe detection with a green color percentage of 62.90%.

Figure 5: Example of unripe detection with a percentage of 62.90%.

2.7 Dataset Creation and Processing

To train the object detection model used in the harvesting system, a dedicated image dataset was created using a Trust Trino HD Webcam. Images were collected under controlled conditions, with the camera positioned at varying distances—specifically 20 cm, 40 cm, and 60 cm from the target crops—to simulate different working ranges of the robot. Additionally, the dataset incorporated multiple viewing angles to account for variability in crop orientation and partial occlusions that may occur in real greenhouse environments.

The dataset was composed of 190 labeled images, including 94 samples of tomatoes and 96 of peppers. This balance allowed the model to develop a robust understanding of both categories. Each image was manually annotated to define the location of each fruit, enabling the model to learn object localization alongside classification. To prepare the dataset for training, the images were divided into three subsets: training, validation, and testing. This structure allowed for effective model development and evaluation by ensuring that the model was not only trained on a portion of the data but also validated and tested on unseen examples. The resulting dataset configuration supported the development of a detection system capable of operating reliably under varying visual conditions. An example of the captured training images is shown in Figure 6 [17].

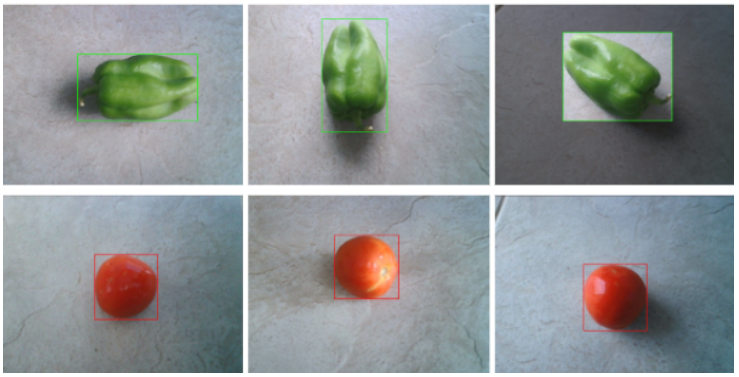


Figure 6: Example of images captured for training.

2.8 Model Training and Performance Evaluation

In this thesis, a comprehensive evaluation was conducted using five variants of the YOLOv5 object detection architecture—YOLOv5n (nano), YOLOv5s (small), YOLOv5m (medium), YOLOv5l (large), and YOLOv5x (extra-large)—trained with two widely adopted optimization algorithms: Stochastic Gradient Descent (SGD) and Adam. All training sessions were carried out on a high-performance computing setup, equipped with a 12th Gen Intel Core i9 processor, 16 GB of RAM, and an NVIDIA GeForce RTX 3060 GPU with 6 GB VRAM, ensuring consistent training performance across architectures.

To assess the performance of the detection system, the following metrics were defined and analyzed:

- **Valid tomato detections** – Correctly localized and labeled detections of tomatoes.
- **Valid pepper detections** – Correctly localized and labeled detections of peppers.
- **False detections** – Instances where the model incorrectly detected an object or misclassified the target.
- **Invalid detections** – Poorly localized detections, such as overlapping boxes or imprecise boundaries.
- **Total detections** – The sum of all valid, false, and invalid outputs.

The model used in this thesis was **YOLOv5l**, trained with the **Stochastic Gradient Descent (SGD)** optimizer. This configuration was selected based on its balance between accuracy and computational efficiency, making it well-suited for real-time robotic harvesting in greenhouse environments.

The detection system demonstrated robust performance across various testing conditions, consistently identifying ripe tomatoes and peppers with a high rate of valid detections and a low occurrence of false or invalid results. The use of SGD contributed to stable training dynamics and reduced overfitting, resulting in precise and reliable detection outputs that supported downstream harvesting operations.

Figures 18–21 illustrate the distribution of valid, false, and invalid detections for both tomato and pepper categories. These results validate the effectiveness of the chosen model and optimizer and meet the system's requirements for real-time object detection in controlled agricultural environments [17].

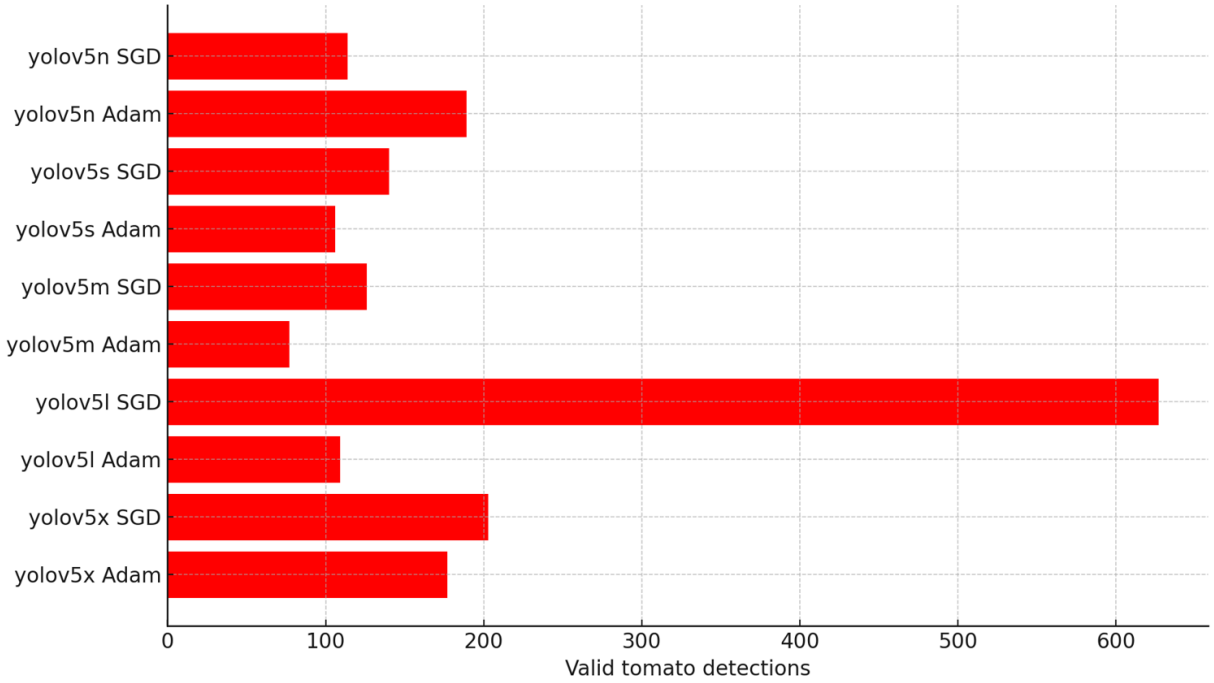


Figure 7: Valid tomato detections by model and optimizer.

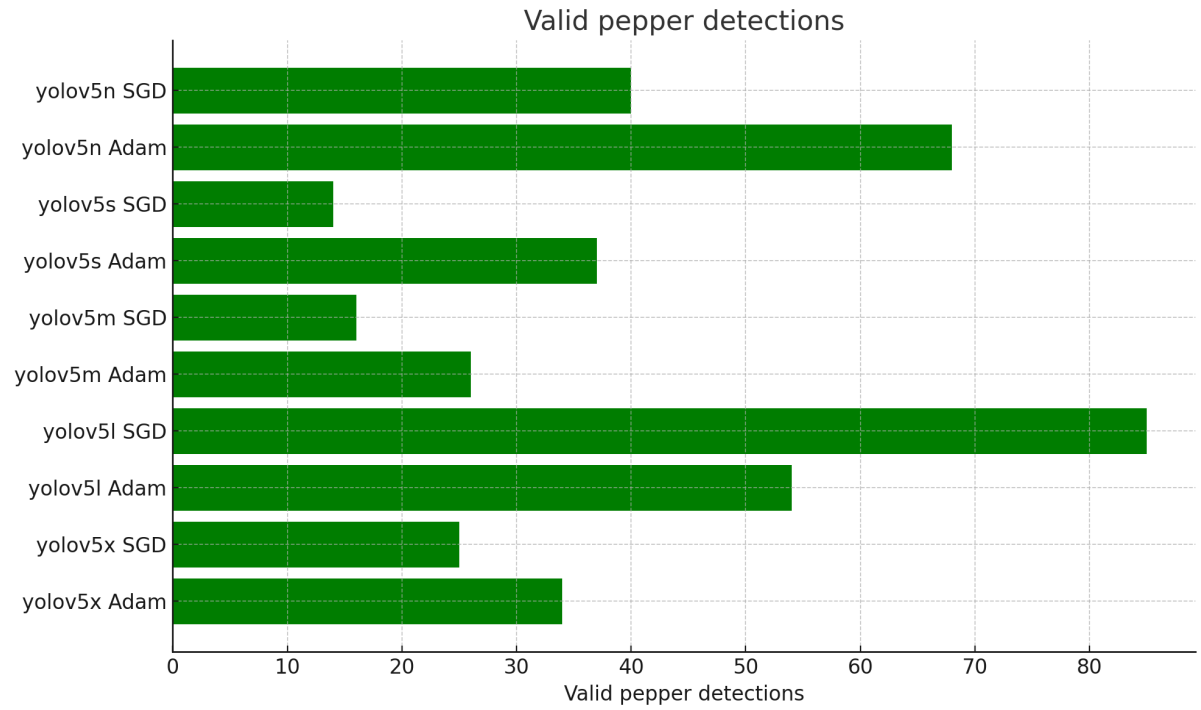


Figure 8: Valid pepper detections by model and optimizer.

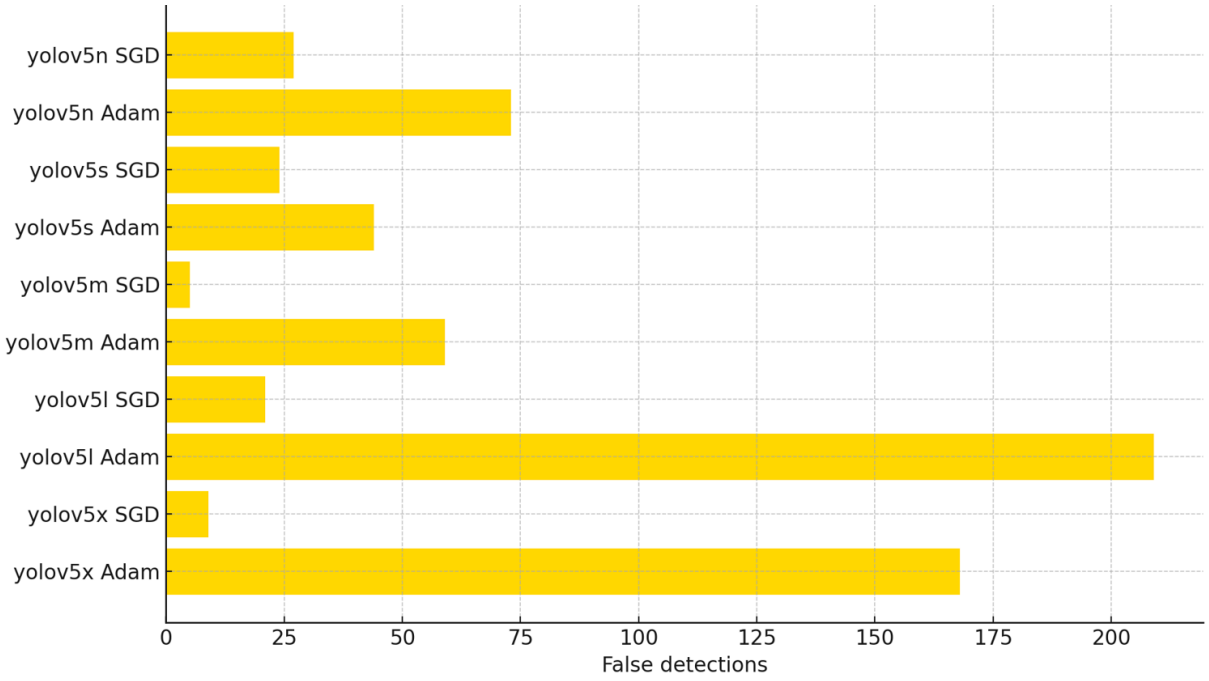


Figure 9: False detections by model and optimizer.

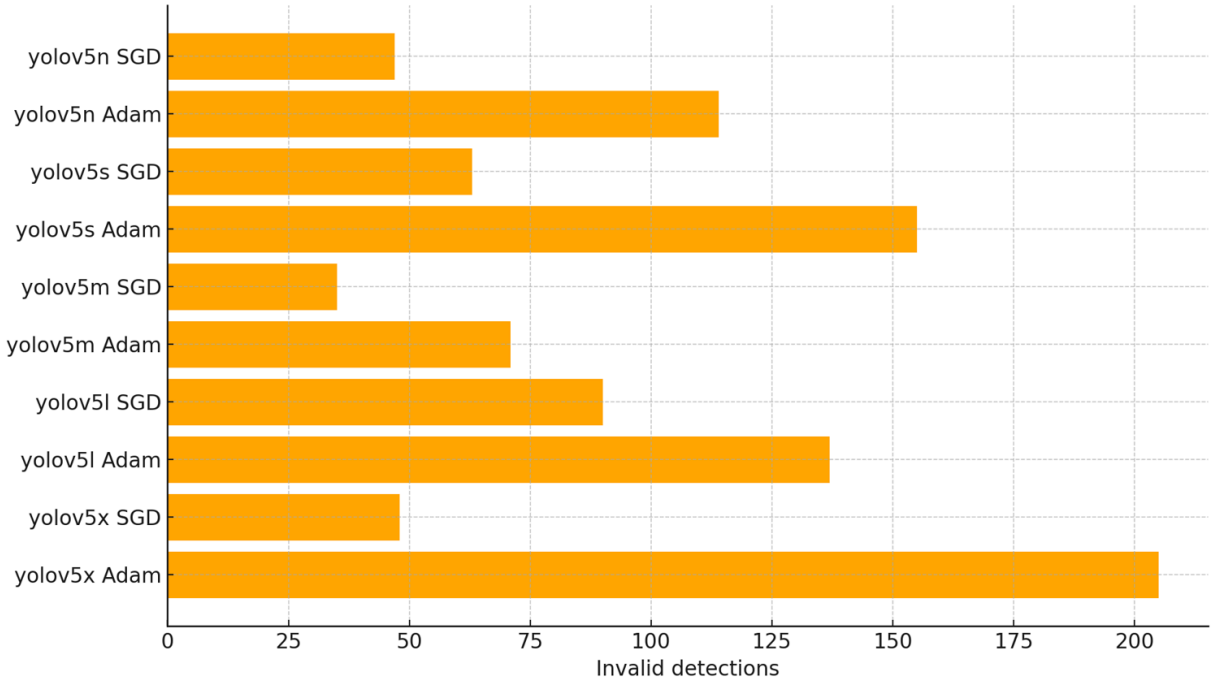


Figure 10: Invalid detections by model and optimizer.

Chapter 3: Mechanical Design and Assembly

3.1 Introduction

The robotic harvesting system is built on a carefully engineered mechanical framework that integrates a mobile platform, an actuator system for precise vertical and horizontal movement, and a cutting tool for efficient harvesting. The mechanical structure is designed to enable the robot to navigate agricultural environments with ease while performing accurate cutting and harvesting operations.

This chapter provides a detailed overview of the robot's mechanical structure, focusing on the individual components and the final assembly. The key components discussed are:

- Mobile platform – Provides mobility and stability.
- Vertical axis and motor mount – Ensures controlled movement and structural integrity.
- Cutting tool – Executes the harvesting action.
- Final assembly – Integration of all components into a functional unit (Figure 11).

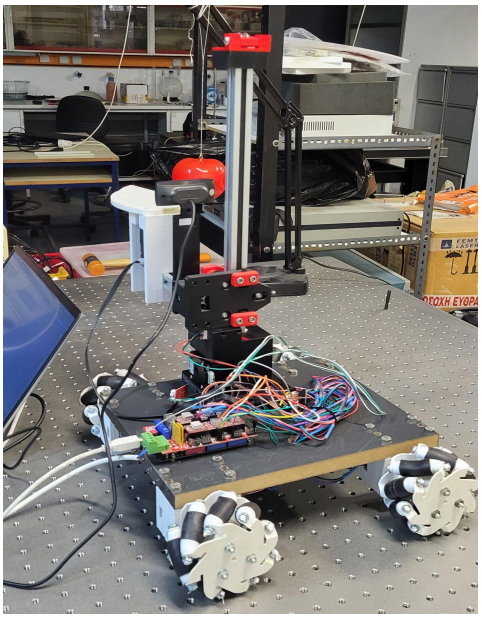


Figure 11: Final assembly of the robotic harvester.

3.2 Mechanical Structure

The mechanical design is based on lightweight yet durable materials to ensure stability and long-term operational reliability. The system's modular design allows for easy maintenance and replacement of parts.

Mobile Platform

The base of the robot consists of a custom-designed platform supported by four Mecanum wheels, providing omnidirectional movement. The Mecanum wheel configuration allows the robot to maneuver laterally and rotate in place, which is essential for precise alignment with target vegetables.

Design Features

The robotic platform is equipped with four Mecanum wheels, each mounted at one corner of the chassis. This configuration provides omnidirectional mobility, enabling the robot to perform complex movement patterns such as lateral (sideways), diagonal, and rotational motion without changing its orientation. The unique roller arrangement of Mecanum wheels allows the robot to maneuver with high precision, which is especially beneficial in confined environments like greenhouses, where space is limited and accurate positioning is crucial for tasks such as harvesting. This mobility feature is shown in Figure 12 [18].

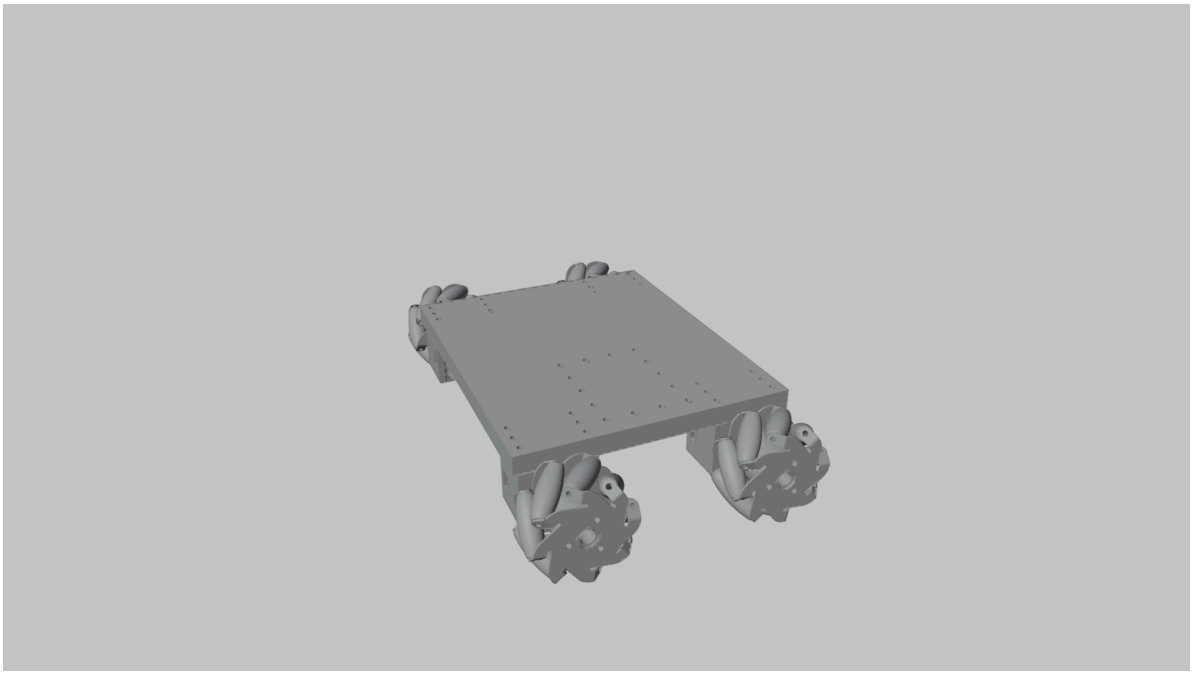


Figure 12: Completed platform with mounted Mecanum wheels.

The chassis of the robotic platform includes strategically placed pre-drilled holes that serve as mounting points for attaching essential components. These include the vertical axis rail system, responsible for guiding the cutting mechanism, as well as the control electronics such as the Arduino Mega 1280 and RAMPS 1.4 shield. Each of the four NEMA17 stepper motors driving the Mecanum wheels is mounted onto a dedicated 3D-printed motor mount securely fastened to the chassis.

This modular design not only simplifies assembly and maintenance but also enhances the overall structural stability of the system. It ensures that all subsystems are securely fixed to the main body, reducing vibrations during operation and allowing seamless integration of mechanical and electrical parts. The platform serves as the foundation for the entire robotic system, providing structural integrity and balanced weight distribution. An example motor mount used on the mobile platform is depicted in Figure 13 [18].

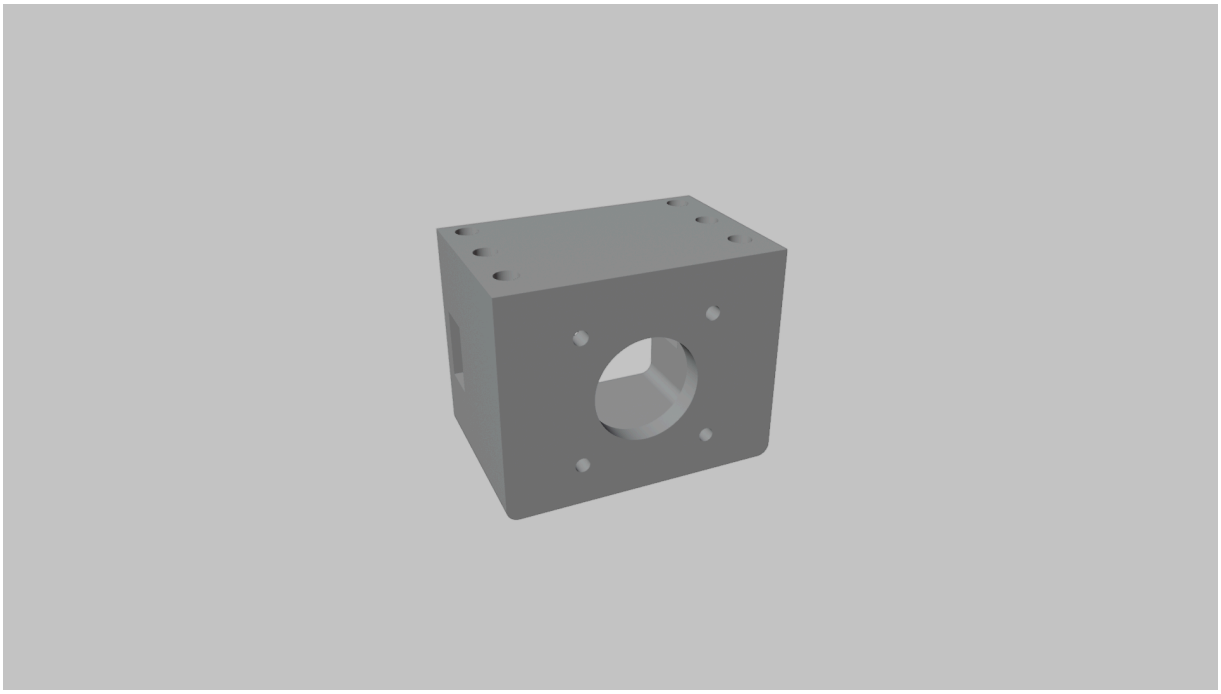


Figure 13: Motor mount for Mecanum wheel on the mobile platform.

Vertical Axis

The vertical axis constitutes a vital mechanical subsystem within the robotic harvesting system. Its primary role is to provide vertical mobility to the cutting tool, enabling it to adjust its position based on the height of the detected vegetable. Accurate vertical motion is essential to ensure that the cutting mechanism engages with the vegetable stem at the correct location, minimizing the risk of crop damage and maximizing harvesting precision. This subsystem directly contributes to the robot's adaptability in greenhouse environments, where the height of crops can vary significantly across the same field.

Design Features

The vertical axis is built around a high-precision linear rail that guides the cutting tool in the vertical direction. This rail offers stable and smooth motion under varying loads, ensuring the system remains aligned and vibration-free during movement. Attached to the rail is a dedicated motor mount, which houses a stepper motor responsible for driving the vertical movement. To establish a reference position for vertical motion, a limit switch is installed at the base of the axis. This switch serves as a homing point during system initialization, allowing the stepper motor to determine its starting position. Although the system does not employ a fully closed-loop feedback mechanism, the homing procedure ensures repeatable and consistent positioning for vertical movements throughout each harvesting cycle. Together, the linear rail, motor mount, and homing switch enable the robotic system to achieve adequate positional accuracy, operational stability, and adaptability. This ensures that the cutting process remains consistent across a variety of vegetable heights, making the vertical axis an essential component for reliable and precise harvesting in controlled agricultural environments. The vertical motion subsystem is illustrated in Figure 14 [18].

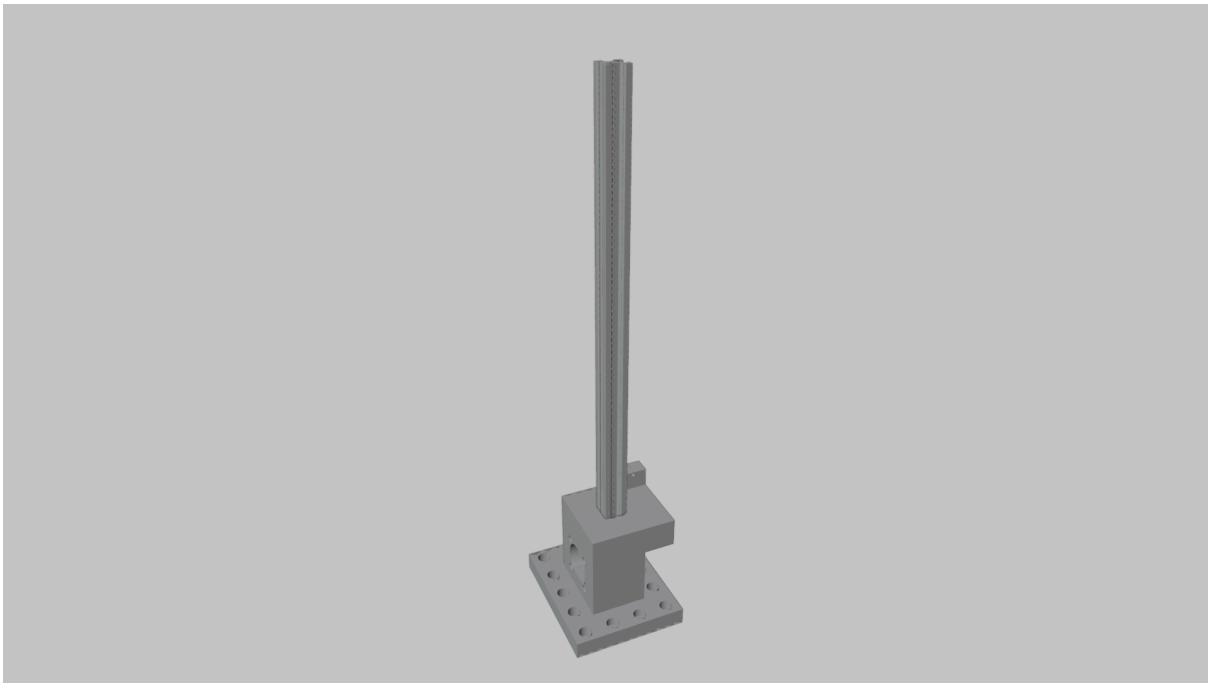


Figure 14: Vertical axis and motor mount.

Cutting Tool

The cutting tool is designed to provide a clean and precise cut without damaging the vegetable or the plant. The tool is mounted on the vertical axis and is actuated using a stepper motor for controlled cutting motion. The assembly is shown in Figure 15 [18].

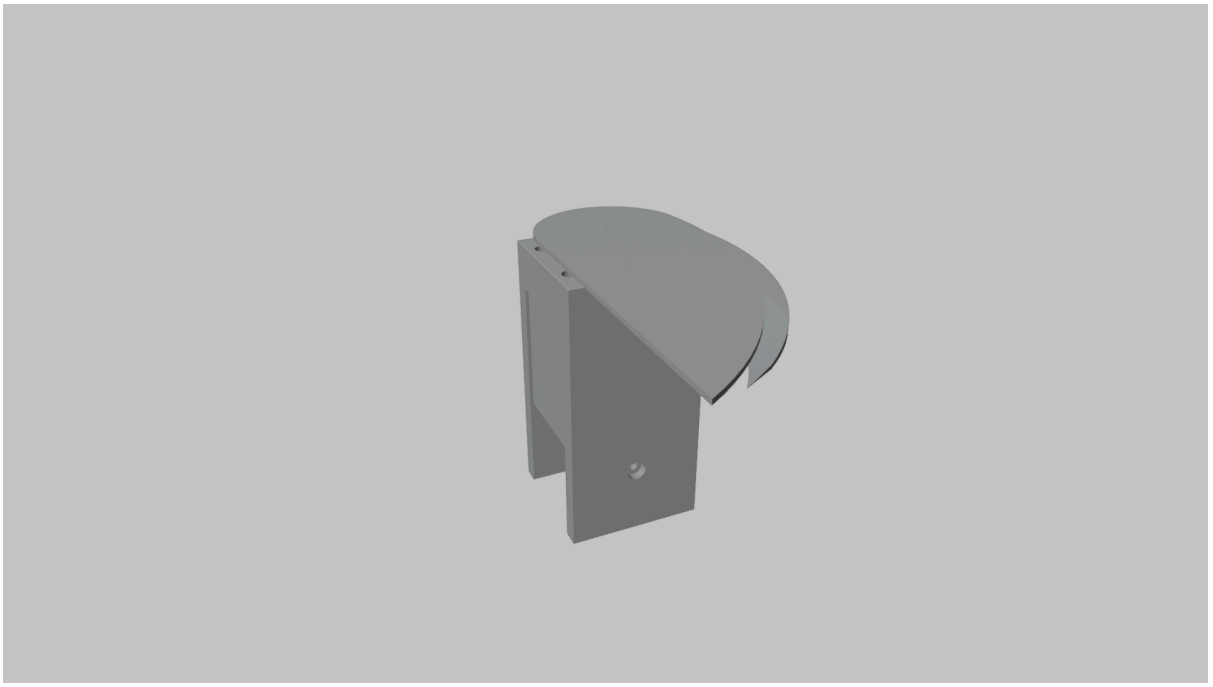


Figure 15: Cutting tool assembly.

Design Features

The blade is supported by a 3D-printed plastic structure. This choice of material ensured a lightweight yet functional design, easily replaceable or modifiable during the prototyping phase. To improve its cutting performance, the blade was manually sharpened, providing an effective yet low-cost solution without compromising functionality. A dedicated motor mount, located at the base of the cutting tool, securely holds the stepper motor responsible for actuating the blade, ensuring proper alignment and mechanical stability throughout the cutting operation. To detect contact with the target crop, a limit switch is positioned adjacent to the blade assembly. When the blade reaches the surface of the crop, the switch is triggered, signaling that the appropriate cutting depth has been achieved. This straightforward mechanism eliminates the need for complex sensing systems, offering a reliable and mechanically robust method for detecting crop contact. Overall, this design enables consistent and safe harvesting, while offering the adaptability necessary for handling both tomatoes and peppers within controlled greenhouse environments.

3.3 Assembly and Integration

The assembly of the robotic system involves the integration of its modular subsystems—namely the mobile platform, vertical axis, and cutting tool—into a cohesive structure capable of autonomous harvesting. The modular design facilitates independent construction of each subsystem, allowing for simplified maintenance, part replacement, and future upgrades.

Assembly Process

1) **Platform Assembly:**

The mobile platform was constructed from a custom-designed 3D-printed chassis. Pre-drilled mounting points were included to accommodate all key components. Four Mecanum wheels were mounted at each corner, enabling omnidirectional motion and precise maneuvering within confined agricultural environments.

2) **Vertical Axis Installation:**

A high-precision linear rail was vertically mounted onto the platform to guide the cutting tool. A dedicated stepper motor and its 3D-printed motor mount were installed to drive vertical motion. At the base of the rail, a limit switch was installed to define the home position for vertical alignment, providing repeatable reference points at system startup.

3) **Cutting Tool Installation:**

The cutting tool was fixed to the carriage of the vertical rail. A second stepper motor with a dedicated mount drives the blade mechanism. An additional limit switch located near the blade detects contact with the crop, serving as a trigger for harvesting action without the need for complex sensors. Manual sharpening of the plastic blade provided the necessary cutting effectiveness for delicate crops such as tomatoes and peppers.

4) **Wiring and Calibration:**

All stepper motors and limit switches were wired to the control electronics mounted on the platform. Initial calibration procedures established homing positions and motion limits for each axis, ensuring consistent operation during repeated harvesting cycles.

The final integration of the system components is displayed in Figure 16 [18].

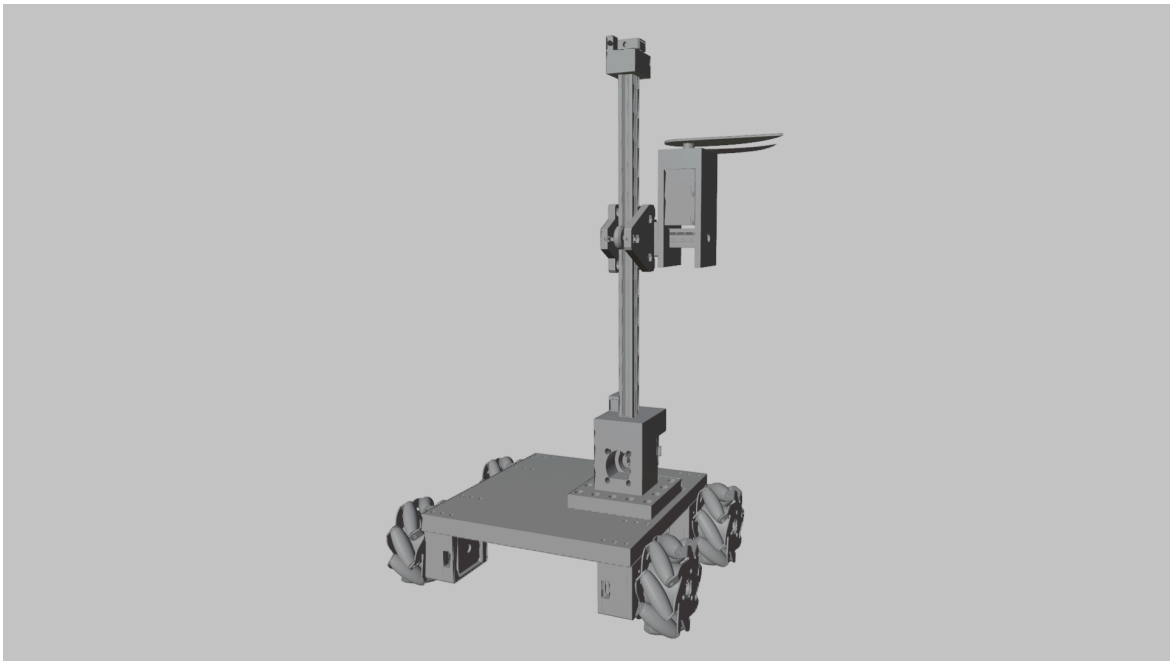


Figure 16: Final assembly of the robot.

3.4 Structural and Operational Considerations

The design of the robotic harvesting system emphasizes stability, mobility, and precision to ensure effective operation within greenhouse environments. Particular attention was devoted to the spatial arrangement of mechanical components and the integration of motion control systems, with the objective of optimizing performance under dynamic and unpredictable agricultural conditions.

Weight distribution was a key design priority. The vertical rail and attached cutting mechanism were centrally positioned on the platform to ensure even weight distribution across the four Mecanum wheels. Each wheel is mounted on a 3D-printed motor mount, reducing overall weight and enabling cost-effective customization. This configuration minimizes the likelihood of instability during locomotion and harvesting operations.

For navigation, Mecanum wheels were selected due to their ability to perform omnidirectional movements, including lateral, diagonal, and in-place rotations. This capability is essential in greenhouse environments where the robot must maneuver between densely planted rows. Directional commands are interpreted by the Arduino controller, which adjusts wheel speed and direction accordingly, enabling smooth traversal over surfaces.

Vertical positioning is controlled by a stepper motor mounted on the vertical axis. A limit switch at the base of the rail acts as a homing sensor, establishing a consistent starting point for each harvesting session. Although the system does not employ a fully closed-loop feedback system, the use of this switch enables accurate vertical alignment without additional sensors. Similarly, the cutting depth is regulated by a second limit switch positioned near the blade, which is activated upon contact with the crop. This mechanical feedback method ensures reliable crop engagement, avoiding the need for more complex and costly sensing technologies.

Collectively, these structural and operational design choices contribute to a robust and efficient robotic platform capable of performing precise and delicate harvesting operations with consistency and reliability.

3.5 Safety and Error Handling

To ensure safe and reliable operation, the robotic harvesting system incorporates several safety and error-handling mechanisms. These features are critical for protecting both the robot and its surrounding environment, particularly in greenhouse conditions where space is limited and crops are delicate.

A limit switch mounted at the base of the vertical axis serves as a reference point for homing, enabling the establishment of a known starting position. During system initialization, the vertical stepper motor moves downward until it contacts the switch, which defines the zero position. This procedure ensures that each harvesting cycle begins from a known, repeatable location, supporting consistent motion control without requiring complex sensor feedback.

A second limit switch is installed near the blade within the cutting tool assembly, serving as a crop-contact sensor. Upon reaching the crop surface, the switch is activated, confirming that the appropriate cutting depth has been achieved. This simple mechanical feedback mechanism provides reliable contact detection without the need for complex distance sensors, ensuring accurate positioning prior to blade actuation.

In terms of manual intervention, a hardware emergency stop button is included to allow human operators to instantly halt the entire system. When pressed, it overrides all software commands and disables all motors, preventing further movement. This safety layer is particularly important during testing and calibration.

At this stage, the system does not include active proximity sensors or dynamic collision detection algorithms. Instead, it relies on pre-mapped trajectories and controlled movements within known environments. The modular structure and predictable stepper motor motion reduce the risk of collisions under typical operating conditions.

Collectively, these safety measures—including homing switches, mechanical feedback and emergency stop functionality—contribute to the operational stability of the robot, enabling it to perform delicate agricultural tasks with reduced risk of mechanical failure or environmental disruption.

Chapter 4: Electronics and Motor Control System

4.1 Introduction

This chapter presents the core electronic and electromechanical subsystems powering the robotic harvesting arm. The architecture integrates NEMA17 stepper motors, A4988 motor drivers, a RAMPS 1.4 shield, and an Arduino Mega 1280 board. The system was carefully designed to achieve precise, repeatable, and robust movements essential for delicate agricultural tasks, while minimizing wiring complexity and improving reliability via modular hardware integration.

The use of the RAMPS 1.4 shield significantly simplified the physical wiring, centralized control routing, and enabled faster maintenance during development. Furthermore, special attention was given to current regulation, microstepping configuration, and motion synchronization using dedicated libraries to ensure smooth, accurate, and safe actuation across the entire robotic arm structure.

The control system is centered around the Arduino Mega 1280, which interfaces with the host PC via USB for both power (logic-level VDD) and serial communication. A second USB port on the host PC powers the HD webcam used for real-time image acquisition and object detection. The stepper motors are powered independently by a 12V external Power Supply Unit (PSU), connected via the RAMPS 1.4 shield to ensure clean and isolated power delivery to the A4988 motor drivers. An overview of the complete electronic architecture of the robotic system is illustrated in Figure 17, highlighting the key hardware components and their interconnections.

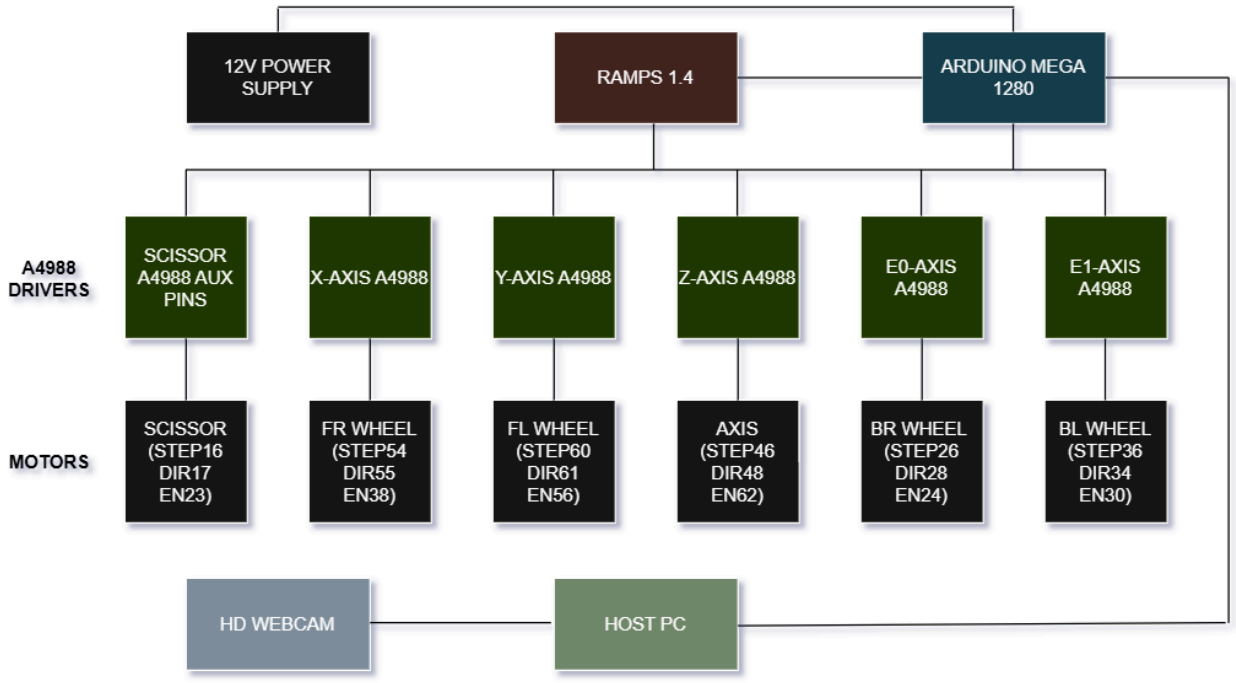


Figure 17: Hardware architecture of the robotic harvesting system, illustrating power distribution, motion control, and communication components.

4.2 Stepper Motors and Motion Control

Stepper Motors Overview

Stepper motors are electromechanical devices that translate discrete electrical pulses into defined incremental mechanical movements. Unlike conventional DC motors, which produce continuous rotation, stepper motors operate based on open-loop control, moving in discrete angular steps without requiring feedback mechanisms such as encoders. This inherently predictable stepping behavior eliminates the complexity of closed-loop systems, significantly simplifying the control architecture while maintaining high positional accuracy [19]. An external view of a typical NEMA17 bipolar stepper motor used in this system is shown in Figure 18 [20].

The design of stepper motors makes them particularly suitable for applications requiring precise positioning, where any missed or inaccurate step could result in operational failure. They also excel in systems requiring highly repeatable movements, a critical attribute in fields such as CNC machining, 3D printing, and agricultural robotic harvesting, where operational consistency over repeated cycles is mandatory. Stepper motors are commonly classified as either bipolar or unipolar, with bipolar motors offering higher torque density at lower speeds. This feature is especially valuable for systems requiring firm holding force and precise actuation under variable loading conditions, as encountered in agricultural environments [21]. The internal configuration of such motors, including the rotor and stator arrangement, is illustrated in Figure 19 [22], while the working principle based on sequential coil activation is demonstrated in Figure 20 [22].

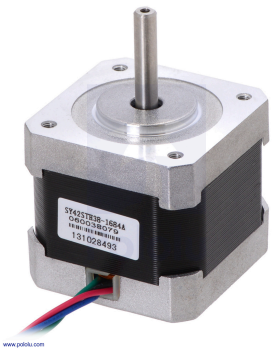


Figure 18: External view of a NEMA17 bipolar stepper motor.

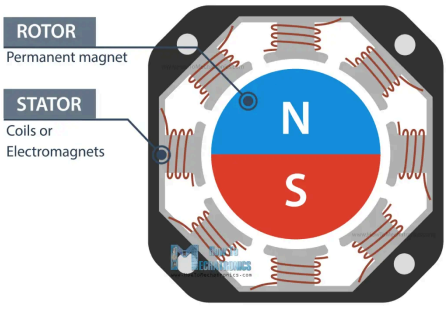


Figure 19: Internal structure of a stepper motor showing rotor and stator components.

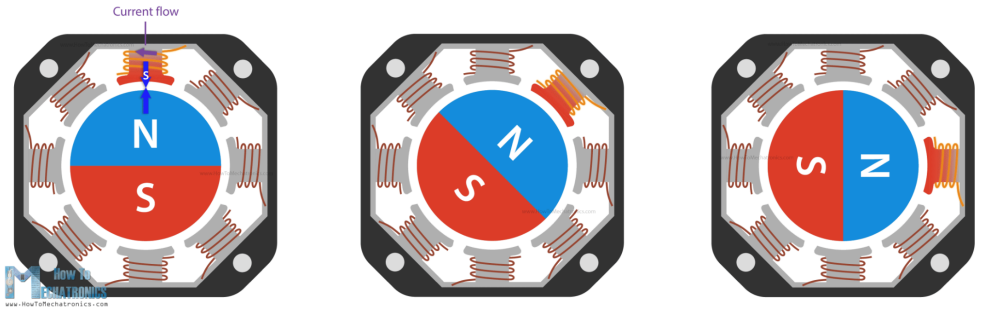


Figure 20: stepper motor working principle illustrated by sequential coil activation.

Principle of Operation

The mechanical design of a stepper motor revolves around two essential components: the stator and the rotor. The stator is the stationary section of the motor, housing multiple electromagnetic coils arranged into several phases. The rotor, on the other hand, can either be a permanent magnet (in permanent magnet stepper motors) or a toothed iron core (in hybrid stepper motors) designed to interact with the magnetic fields generated by the stator.

The motor operates by sequentially energizing the stator coils. As each set of coils is activated, a rotating magnetic field is established, causing the rotor to align with the field's orientation. Each change in the energization sequence leads to a discrete mechanical step of the rotor. Consequently, the motor achieves predictable, incremental movement, where each electrical pulse sent to the motor corresponds directly to a mechanical advancement.

In the context of this thesis, NEMA17 bipolar stepper motors were employed due to their optimal balance between torque output, compact size, and control simplicity. These motors offer a native mechanical resolution of 200 full steps per revolution. Structurally, the rotor contains 50 magnetic teeth, while the stator provides four distinct magnetic field orientations per electrical cycle. As a result, multiplying the 50 rotor teeth by the four stator step positions yields 200 discrete steps for a full 360-degree rotation. This configuration results in a step angle of 1.8 degrees per step, as calculated by the equation:

$$Step\ Angle = \frac{360°}{200\ Steps} = 1.8°\ per\ step \quad (1)$$

This intrinsic precision makes the NEMA17 particularly well-suited for applications requiring accurate, repeatable motion control, such as aligning harvesting tools with delicate produce stems.

The motor's two-phase winding configuration can be easily observed by examining the wiring: a typical bipolar stepper motor has four wires, corresponding to two coils.

By controlling current direction through each phase, four unique magnetic field states are achieved, enabling the precise stepping action. Some stepper motors offer additional wiring (5, 6, or even 8 wires), providing flexibility for alternative drive configurations like unipolar mode, which can simplify control circuits but typically at the expense of available torque [22]. An illustration of the internal rotor-stator configuration and the two-phase winding system (Phase 1 and Phase 2) is provided in Figure 21 [22]. Energizing these windings in sequence creates a rotating magnetic field that drives the rotor in discrete steps.

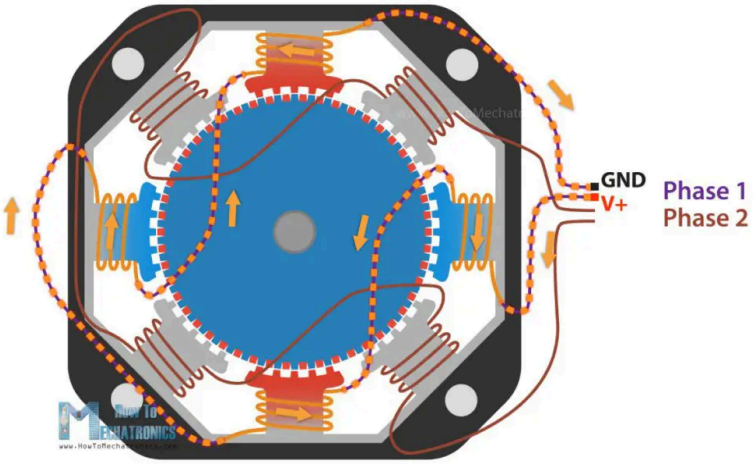


Figure 21: Internal structure and phase configuration of a bipolar stepper motor.

Microstepping and Precision Enhancement

To achieve even finer control, microstepping was implemented using the A4988 driver modules. Microstepping involves partially energizing two adjacent motor phases simultaneously with controlled current ratios, allowing the rotor to achieve intermediate positions between full steps. This technique provides several benefits, including smoother motion profiles, reduced resonance and vibration, and higher positional resolution.

For this project, 1/16 microstepping was configured, meaning that each full step of 1.8 degrees was subdivided into sixteen smaller microsteps. Thus, the effective movement per microstep is reduced to approximately 0.1125 degrees. This configuration dramatically enhances the system's movement precision, yielding 3200 microsteps per full rotation. Such high granularity is critical for tasks requiring delicate, precise movements, like the positioning of a harvesting blade around fragile fruits without causing damage. Additionally, microstepping improves the mechanical behavior of the motors, reducing wear on components and minimizing audible noise during operation. These attributes directly contribute to the long-term operational reliability and mechanical robustness of the robotic

harvesting system, especially important in agricultural field conditions where environmental factors can vary unpredictably.

4.3 Arduino Mega 1280 and RAMPS 1.4 Integration

Arduino Mega 1280 Overview

Arduino is an open-source electronics platform widely recognized for enabling rapid

development and prototyping of embedded systems. Designed for ease of integration and modular expansion in mind, Arduino boards are commonly utilized in research, industrial, and educational environments for automating physical systems. Among the various available models, the Arduino Mega 1280 was selected for this study due to its high input/output (I/O) capacity, robust memory resources, and broad compatibility with third-party expansion shields.

The Arduino Mega 1280 features a total of 54 digital I/O pins and 16 analog inputs, enabling simultaneous interfacing with multiple sensors, actuators, and peripheral modules. Furthermore, it is equipped with 128 kilobytes of flash memory for code storage and 8 kilobytes of SRAM, providing ample space for complex firmware operations that manage motion control, sensor fusion, and communication tasks [23]. These hardware capabilities are critical for autonomous robotic systems that must execute real-time operations without compromising responsiveness or reliability. A visual overview of the board's pin configuration and functional layout is shown in Figure 22 [25].

In the robotic harvesting platform developed in this project, the Arduino Mega 1280 functions as the core processing unit. It interprets high-level motion commands, manages precise timing for stepper motor control, handles feedback from limit switches and other sensors, and coordinates decision-making processes. By executing a custom firmware stack based on libraries such as AccelStepper, the Arduino facilitates synchronized multi-axis control essential for delicate harvesting operations. Its proven reliability, ease of programming, and extensive documentation made it an ideal choice for this agricultural automation application.

RAMPS 1.4 Shield

To interface the Arduino Mega 1280 with the electromechanical components of the robotic system in a structured and scalable way, the RAMPS 1.4 (RepRap Arduino Mega Pololu Shield) was employed. RAMPS 1.4 is a hardware expansion board that layers directly onto the Arduino Mega, providing an organized and modular platform for connecting stepper motor drivers, endstop switches, power supplies, and other peripherals critical to robotic control.

The RAMPS 1.4 shield streamlines the otherwise complex process of wiring multiple high-current and logic-level signals. It includes pre-soldered sockets for four A4988 stepper motor driver modules, corresponding to the X, Y, Z, and E axes, along with dedicated headers for limit switches used in homing and safety mechanisms. In addition, RAMPS separates the power supply architecture into distinct rails: a higher voltage bus (typically 12V to 36V) for motor actuation and a lower-voltage logic bus (5V) powered directly from the Arduino Mega's onboard regulator [24].

The adoption of RAMPS 1.4 significantly reduced the number of manual point-to-point wire connections required, leading to improved system robustness and minimized the risk of miswiring during assembly. Moreover, by consolidating all power and signal distribution onto a single printed circuit board (PCB), electromagnetic interference (EMI) effects were mitigated, and overall system reliability was enhanced. The modular design of the RAMPS shield further facilitated maintenance and future upgrades, such as the addition of extra sensors, actuator channels, or even secondary processing units. A diagram of the RAMPS 1.4 pin mapping and integration with the Arduino Mega is provided in Figure 23 [26].

In practice, the integration of RAMPS 1.4 not only accelerated the prototyping phase but also contributed to better mechanical organization within the enclosure, simplifying debugging and troubleshooting processes. This methodical system design approach, leveraging the Arduino Mega 1280 and RAMPS 1.4 combination, ensured that the robotic harvesting platform achieved the necessary precision, scalability, and operational stability required for field deployment in dynamic agricultural environments.

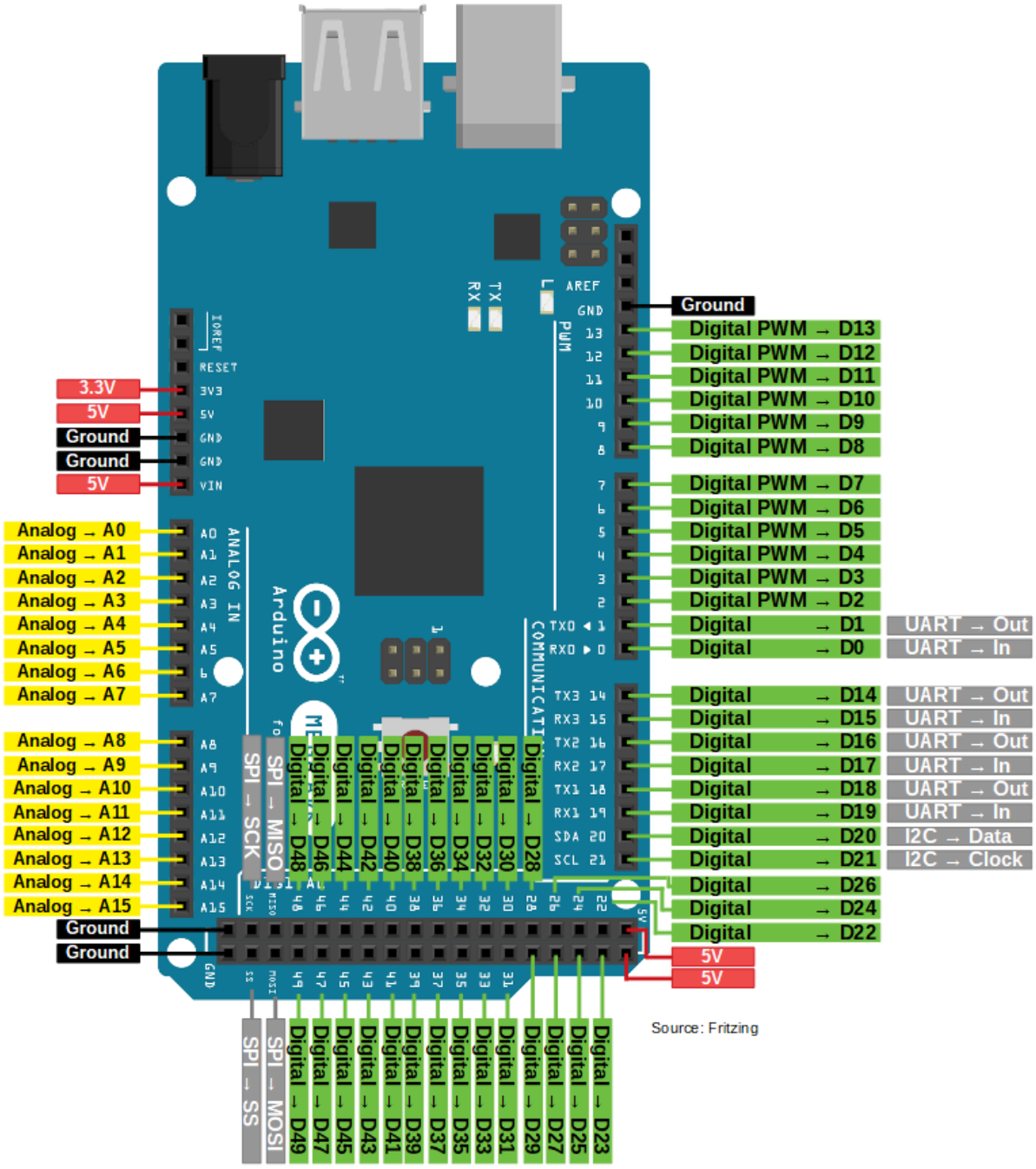


Figure 22: Arduino Mega 1280: pinout and functional overview.

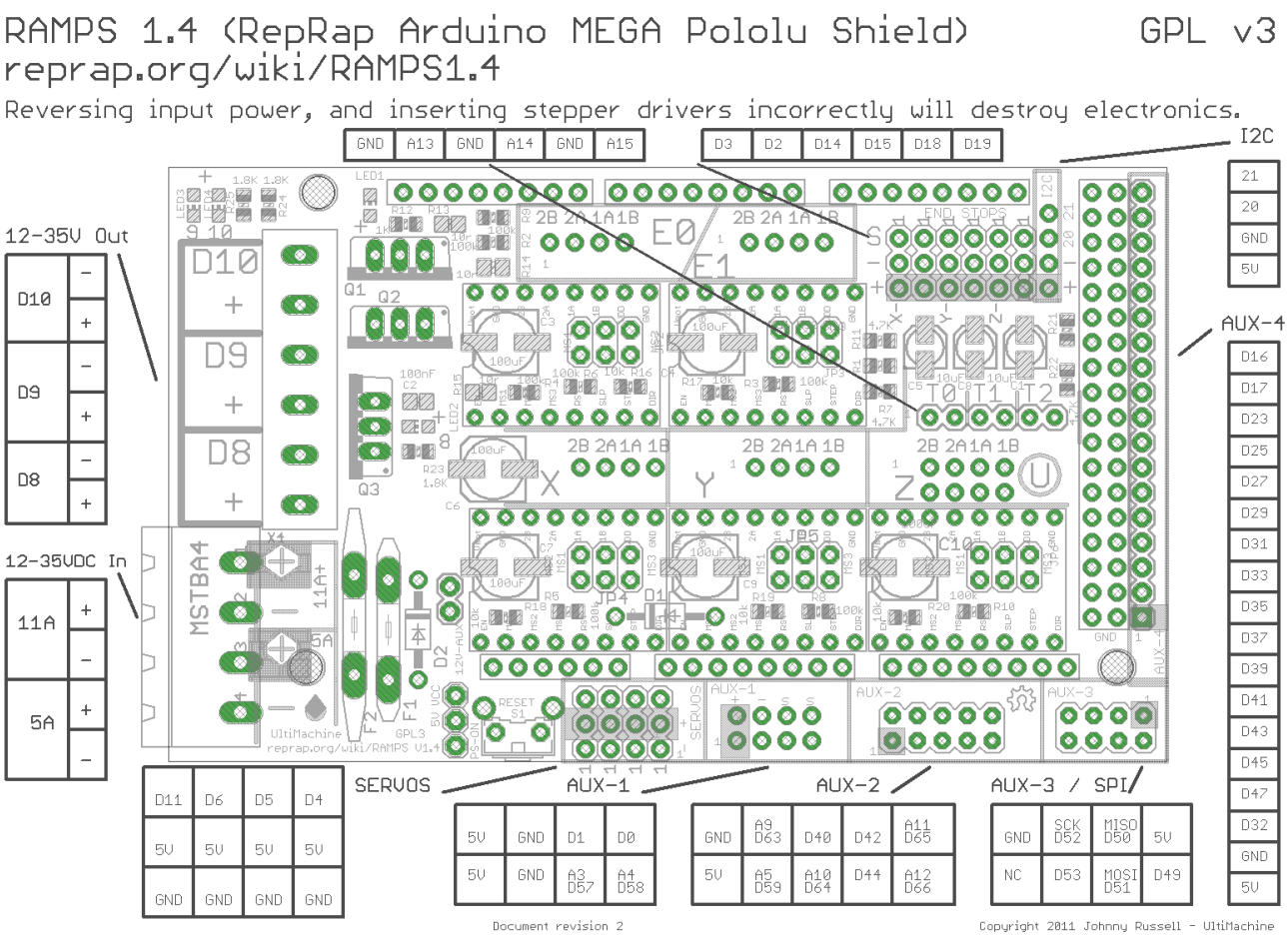


Figure 23: RAMPS 1.4 shield pinout diagram for Arduino Mega integration.

4.4 A4988 Stepper Motor Driver Configuration

Overview

The A4988 stepper motor driver plays a central role in the electromechanical control architecture of the robotic harvesting system, providing precise control of bipolar stepper motors through microstepping and adjustable current regulation [27]. Its compact form factor and built-in safety mechanisms make it an ideal solution for embedded automation systems requiring accurate, smooth, and efficient motion. The physical layout and pin functionality of the A4988 driver are depicted in Figure 24 [29].

This driver supports multiple stepping resolutions—ranging from full-step to sixteenth-step modes—making it adaptable to tasks with varying precision and torque demands [28]. The ability to microstep allows the rotor to move in fractions of its native step angle by modulating the current between motor windings. This results in significantly smoother motion and improved control accuracy, which is particularly crucial in applications such as fruit harvesting where mechanical vibrations could damage delicate produce.

To ensure safe and efficient operation, the A4988 includes integrated current limiting, thermal shutdown, and short-circuit protection. The current limit can be manually configured using an onboard potentiometer, allowing the driver to match the rated current of the connected NEMA17 stepper motors, thereby preventing overheating and extending motor lifespan [27].

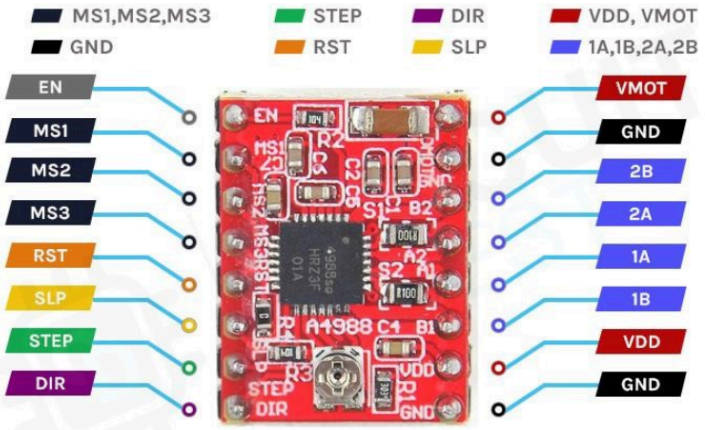


Figure 24: Pinout diagram of the A4988 stepper motor driver module.

Driver Configuration

Each A4988 module is interfaced with the Arduino Mega 1280 via the RAMPS 1.4 shield, which routes all essential control signals—namely STEP, DIR, and EN—from the microcontroller to the driver and ultimately to the motor [24]. This setup allows consistent hardware abstraction and simplifies firmware configuration. A schematic representation of the A4988 stepper motor driver is shown in Figure 25 [32].

The STEP pin acts as the pulse generator. Each rising edge signal causes the motor to advance by one increment, which could be a full step or a microstep depending on the selected resolution [30]. The DIR pin sets the rotational direction—clockwise or counterclockwise—depending on the logic level applied. The EN pin enables or disables the motor output stage, where a LOW logic level activates the driver and a HIGH disables it, allowing for controlled power savings or safe shutdowns.

To achieve high-resolution motion, the system was configured for 1/16 microstepping by setting the MS1, MS2, and MS3 pins to HIGH. This setting divides the motor's native 1.8° step into 16 smaller increments of 0.1125°, achieving finer motion and smoother acceleration profiles. As a result, a full 360° rotation of a NEMA17 stepper motor, which would normally require 200 full steps, now involves 3200 microsteps:

$$Step\ Angle\ =\ \frac{360°}{200\ Steps}\ =\ 0.0125°\ per\ microstep \tag{2}$$

This finer granularity in movement allows the robotic arm to align more precisely with the fruit stem, reducing the likelihood of harvesting errors or crop damage.

The A4988's microstepping configuration is summarized in the table below:

MS1	MS2	MS3	Stepping Mode
Low	Low	Low	Full step (1.8° per step)
High	Low	Low	Half step (0.9° per step)
Low	High	Low	Quarter step (0.45° per step)
High	High	Low	Eighth step (0.225° per step)
High	High	High	Sixteenth step (0.1125° per step)

Table 1: Microstepping modes of the A4988 driver.

In this implementation, all microstepping pins were tied HIGH to utilize the maximum resolution offered by the driver. This not only improved the motion fidelity but also reduced mechanical resonance and audible noise from the motors [31].

The RAMPS 1.4 board standardized the pin mapping for each axis as follows:

Motor Axis	STEP Pin	DIR Pin	EN Pin
X Axis	D54	D55	D38
Y Axis	D60	D61	D56
Z Axis	D46	D48	D62

Table 2: Pin mapping between RAMPS 1.4 and A4988 motor drivers.

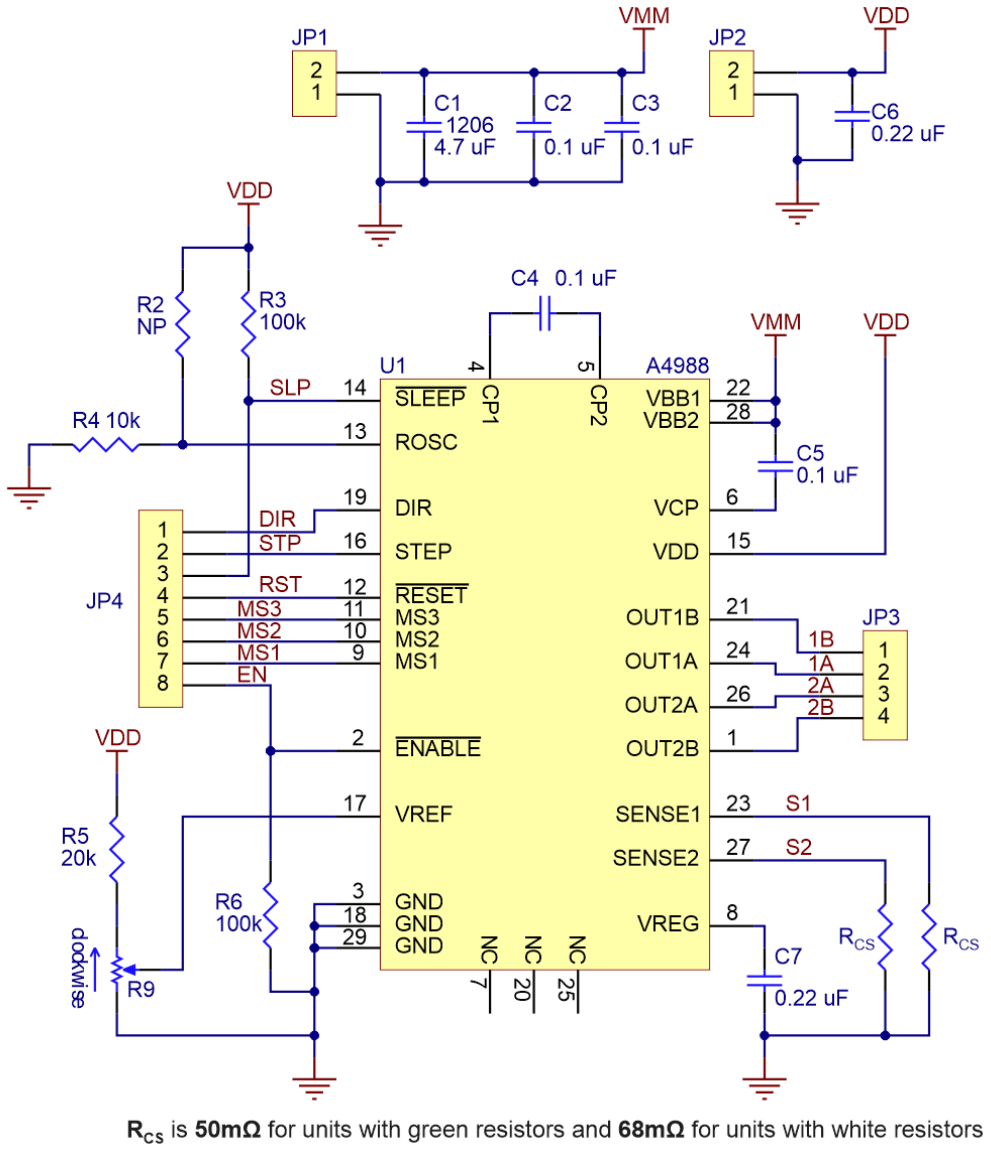


Figure 25: Schematic diagram of the A4988 stepper motor driver.

4.5 Power Architecture and Current Limiting

A robust and carefully designed power architecture is critical to ensure reliable and safe operation of stepper motor-driven robotic systems, especially in applications such as autonomous agricultural harvesting where continuous load cycles and variable mechanical stresses are common.

In the implemented system, two distinct voltage rails were established. The first rail, designated VMOT, supplied the motor voltage at 12V DC through an external regulated Power Supply Unit (PSU). This line was dedicated exclusively to powering the high-current phases of the NEMA17 stepper motors via the A4988 drivers. The second rail, designated VDD, provided a stable 5V logic voltage, sourced directly from the Arduino Mega 1280 board. The separation of motor power and logic power was critical to prevent electrical noise and transient disturbances generated by motor actuation from interfering with the sensitive microcontroller logic circuits [33]. This dual-rail configuration is illustrated in Figure 26, which depicts the overall power and communication layout of the system.

Each A4988 driver incorporated an onboard current-limiting potentiometer (often referred to as a trimpot) to prevent excessive current draw through the motor windings. Proper current limiting was essential not only to protect the stepper motors from thermal degradation but also to prevent driver overheating, which could otherwise lead to thermal shutdowns or irreversible damage.

The current limit for each motor phase was carefully calibrated based on the reference voltage (Vref) measured at the driver's test point. The relationship between Vref, the sense resistor value (Rcs), and the maximum phase current (Imax) is given by the following equation:

$$I_{max} = \frac{V_{ref}}{8 \times R_{cs}} \tag{3}$$

where:

- V_{ref} is the reference voltage set via the potentiometer,
- R_{cs} is the value of the current sense resistor on the driver PCB, typically 0.1Ω for standard A4988 modules.

By setting the V_{ref} to 0.8V, and assuming R_{cs} = 0.1 Ω, the resulting maximum current per phase was calculated as:

$$I_{max} = \frac{0.8\,V}{8 \times 0.1} = 1\text{A} \tag{4}$$

The implemented power architecture effectively separated motor and logic voltages, enabling stable operation under varying load conditions. The combination of a dedicated PSU for motor power, USB-based logic supply, and carefully configured current limits ensured safe, reliable performance of the robotic harvesting platform throughout extended harvesting sessions.

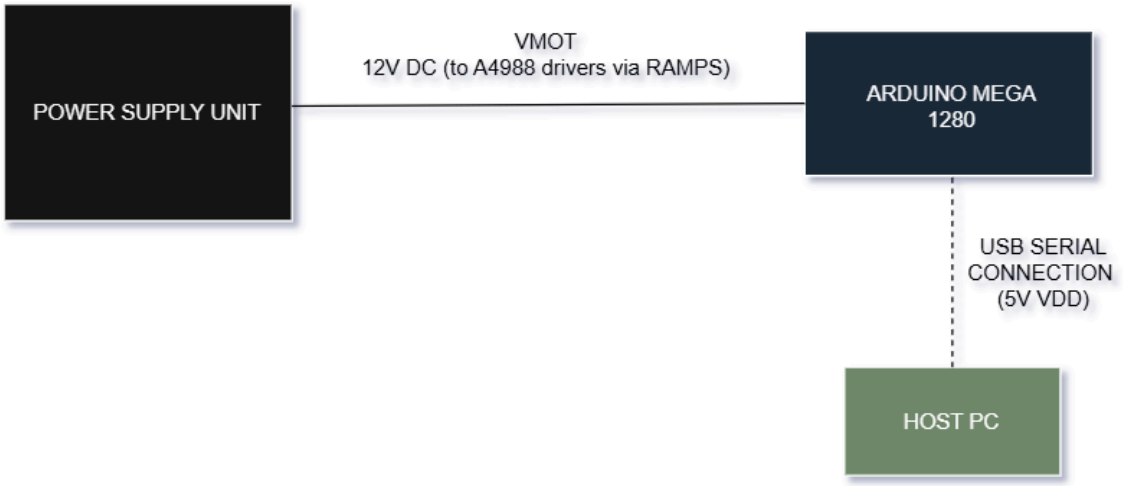


Figure 26: Power and communication architecture of the robotic system.

4.6 System Wiring and RAMPS Advantages

The integration of the RAMPS 1.4 shield into the robotic harvesting platform provided a significant simplification of the overall system wiring and offered substantial operational advantages. All limit switches, including the endstop switches for the X, Y, and Z axes, were connected directly to the clearly labeled headers available on the RAMPS board. This standardized connection approach eliminated the need for custom wiring adaptations, thus reducing the likelihood of wiring errors and facilitating straightforward system diagnostics and maintenance.

The necessity for breadboards or point-to-point hand-soldered interconnections was entirely removed. Instead, all motor driver modules, stepper motors, and peripheral sensors interfaced through dedicated sockets and terminal blocks on the RAMPS shield. This not only streamlined the physical assembly process but also provided a more robust and vibration-resistant electrical architecture, an important consideration for mobile or field-deployed agricultural robots.

Electromagnetic Interference (EMI), often exacerbated by long unshielded wire runs and floating connections, was minimized. The compact and organized wiring layout achieved by utilizing RAMPS' dedicated paths ensured shorter wire lengths and better shielding practices. As a result, signal integrity was improved, and the system demonstrated higher resilience against noise-induced malfunctions during real-world operations inside a greenhouse environment.

An additional benefit of using the RAMPS shield was the dramatic reduction in assembly time. It is estimated that the total wiring and system integration time was reduced by approximately 50% compared to conventional point-to-point wiring methods. This efficiency gain not only expedited the initial prototyping phase but also simplified future maintenance tasks, allowing faulty motor drivers or limit switches to be replaced modularly without invasive rewiring or soldering.

Finally, the modularity provided by the RAMPS 1.4 design offers an excellent foundation for system scalability. Future expansions, such as the addition of extra robotic arms, gripper modules, or advanced sensor suites, can be accommodated with minimal changes to the existing hardware framework. This level of flexibility is crucial for ensuring that the robotic harvesting platform can evolve alongside emerging agricultural automation needs.

Chapter 5: Software Implementation: Integration of Machine Vision and Robotic Control

5.1 Introduction

The implementation of robotic automation in agricultural harvesting relies on the seamless integration of software and hardware components. This chapter presents the development and implementation of the software system used in this thesis, detailing both the machine vision algorithms responsible for fruit detection and classification and the hardware control system that governs the robotic arm's movements. The software was developed using Python for object detection and Arduino for low-level motor control and communication with the robotic system.

The software architecture consists of:

- Python-based detection and decision-making system using YOLOv5, responsible for image processing and object identification.
- Arduino-based motion control system, managing the stepper motors and executing movement commands based on the detected fruit locations.
- Serial communication between the two systems, allowing real-time interaction between image recognition and robotic actuation.

An overview of this architecture and its communication flow is illustrated in Figure 27, which shows how data moves between the software layers and the robotic hardware. The following sections present a detailed breakdown of the Python implementation for vision processing and the Arduino implementation for robotic control.

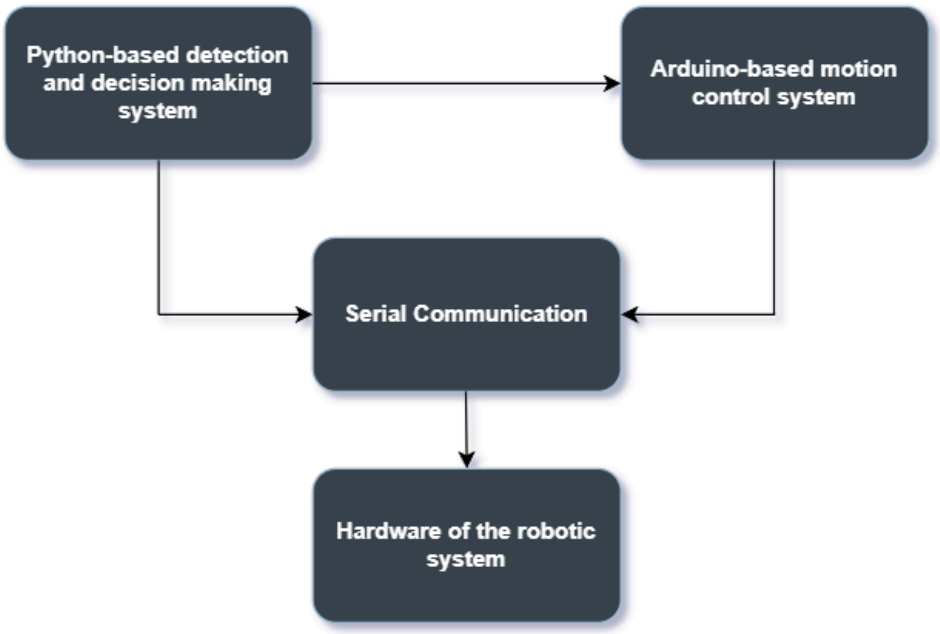


Figure 27: Block diagram of the interaction between the machine vision module, motion control system, and robotic hardware via serial communication.

5.2 Arduino Implementation for Robotic Control

This chapter presents the Arduino software responsible for controlling the robotic system's navigation, vertical movement, and scissor mechanism. The system is designed to operate through serial commands while incorporating safety mechanisms, such as limit switches, to prevent excessive movement and ensure safe operation. The Arduino microcontroller serves as the central control unit, interpreting movement commands, translating them into motor control signals, and executing precise movements required for the automation of the harvesting process.

The Arduino code facilitates the following core functionalities:

- Control of stepper motors for the robotic arm's movement.
- Execution of movement commands received from the Python script.
- Ensuring safety through limit switches and emergency stop mechanisms.
- Enabling smooth, real-time interaction between the robotic hardware and software system.

The Arduino Mega 1280 microcontroller is selected due to its large number of I/O pins, which are required for stepper motor control via step, direction, and enable signals.

The structure of the code follows a logical and modular approach to ensure smooth and efficient operation of the robotic system. Through real-time command execution, the system optimizes precision in harvesting actions while minimizing potential errors. This section provides a comprehensive overview of the Arduino code flow, detailing how it governs the robotic system's motion, safety mechanisms, and overall performance.

System Overview

The robotic system consists of:

- Four-wheel stepper motors for movement.
- Axis motor for vertical motion.
- Scissor mechanism motor for expansion and contraction.
- Limit switches for homing and movement restriction.
- Serial communication for command-based operation.

The hardware setup includes:

- Arduino Mega 1280 as the central microcontroller.
- A4988 motor drivers to control NEMA17 stepper motors.
- Serial communication via USB to receive commands from the Python script.

Each stepper motor is driven by an A4988 motor driver, which requires three control signals:

- Step (STEP pin) – Triggers the motor to move one step.
- Direction (DIR pin) – Determines the direction of rotation.
- Enable (EN pin) – Activates or deactivates the motor.

Additionally, limit switches are employed to ensure safety, preventing the robotic arm from exceeding its designated movement range.

Motion Control Logic

To enable precise and smooth multi-axis control, the AccelStepper library was employed. Unlike traditional blocking motor control methods, AccelStepper supports non-blocking, asynchronous operation, allowing motors to move independently while the microcontroller handles serial communication and sensor input. This approach is essential in robotic harvesting applications, where the robot must react to external triggers in real time and maintain smooth motion to avoid damaging crops or destabilizing the platform. By combining modular hardware control through the RAMPS 1.4 shield and software abstraction via AccelStepper, the robotic platform achieves reliable, smooth, and coordinated motion [34].

Arduino Code Overview

The Arduino firmware is programmed to continuously listen for incoming commands sent by the Python script and to execute corresponding movement actions based on those instructions. Its key functionalities include the reception of movement instructions via serial communication, ensuring that external commands are reliably processed in real-time. Upon receiving these instructions, the firmware controls the A4988 stepper motor drivers to coordinate the movements of the robotic arm with high precision, translating abstract commands into exact mechanical actions. Additionally, the Arduino reads sensor data from limit switches installed throughout the system to monitor positional boundaries and guarantee operational safety, ensuring that all movements are executed within defined mechanical limits to prevent damage or unintended behavior. A flowchart summarizing the structure of the Arduino code is presented in Figure 28, illustrating the major stages from initialization to motion control and safety monitoring. Table 3 summarizes the recognized commands and their corresponding motion instructions used throughout the control system.

Code flow

1) Initialization

Upon startup, the Arduino initializes serial communication and configures all motor and limit switch pins. All motors are set to an inactive state to prevent unintended movement.

2) Serial Communication Handling

The Arduino continuously monitors the serial interface for incoming commands. Commands are parsed and processed, determining the type of movement requested. The system expects commands in the format: COMMAND:DISTANCE (table 5.1)

3) Motor Movement Execution

Once a command is received and parsed, the appropriate motor control signals are activated:

- Forward, backward, left, and right movement is executed by setting the direction pins accordingly and triggering motor steps.
- Upward and downward movement of the vertical axis is performed while continuously checking the limit switch state to prevent exceeding travel limits.
- Scissor opening and closing is executed based on command parameters, ensuring the mechanism does not overextend or fully collapse.

4) Limit Switch Monitoring

Throughout execution, the Arduino continuously monitors the state of the limit switches.

- If a limit switch is triggered, the corresponding motion is immediately halted to prevent hardware damage.
- If the vertical axis reaches its lowest or highest point, any further movement in that direction is disabled.
- If the scissor mechanism reaches its limits, further actuation is prevented to maintain system integrity.

5) Emergency Stop Mechanism

If a STOP (S) command is received, all motor activity is immediately halted, and the robot remains idle until a new command is processed.

Command	Function	Example
`F:mm`	Move forward by `mm` millimeters	`F:100` (Move forward 100 mm)
`B:mm`	Move backward by `mm` millimeters	`B:50` (Move backward 50 mm)
`L:mm`	Shift left by `mm` millimeters	`L:30` (Shift left 30 mm)
`R:mm`	Shift right by `mm` millimeters	`R:30` (Shift right 30 mm)
`TR:mm`	Rotate right by `mm` millimeters	`TR:30` (Turn right 30 mm)
`TL:mm`	Rotate left by `mm` millimeters	`TL:30` (Turn left 30 mm)
`UP:mm`	Move vertical axis up by `mm` millimeters	`UP:200` (Move up 200 mm)
`DN:mm`	Move vertical axis down by `mm` millimeters	`DN:150` (Move down 150 mm)
`OP:mm`	Expand scissor mechanism by `mm` millimeters	`OP:50` (Expand by 50 mm)
`CL:mm`	Contract scissor mechanism by `mm` millimeters	`CL:50` (Contract by 50 mm)
`H`	Home the vertical axis to its default position	`H` (Homing procedure)
`S`	Stop all movements immediately	`S` (Emergency stop)

Table 3: Command summary table.

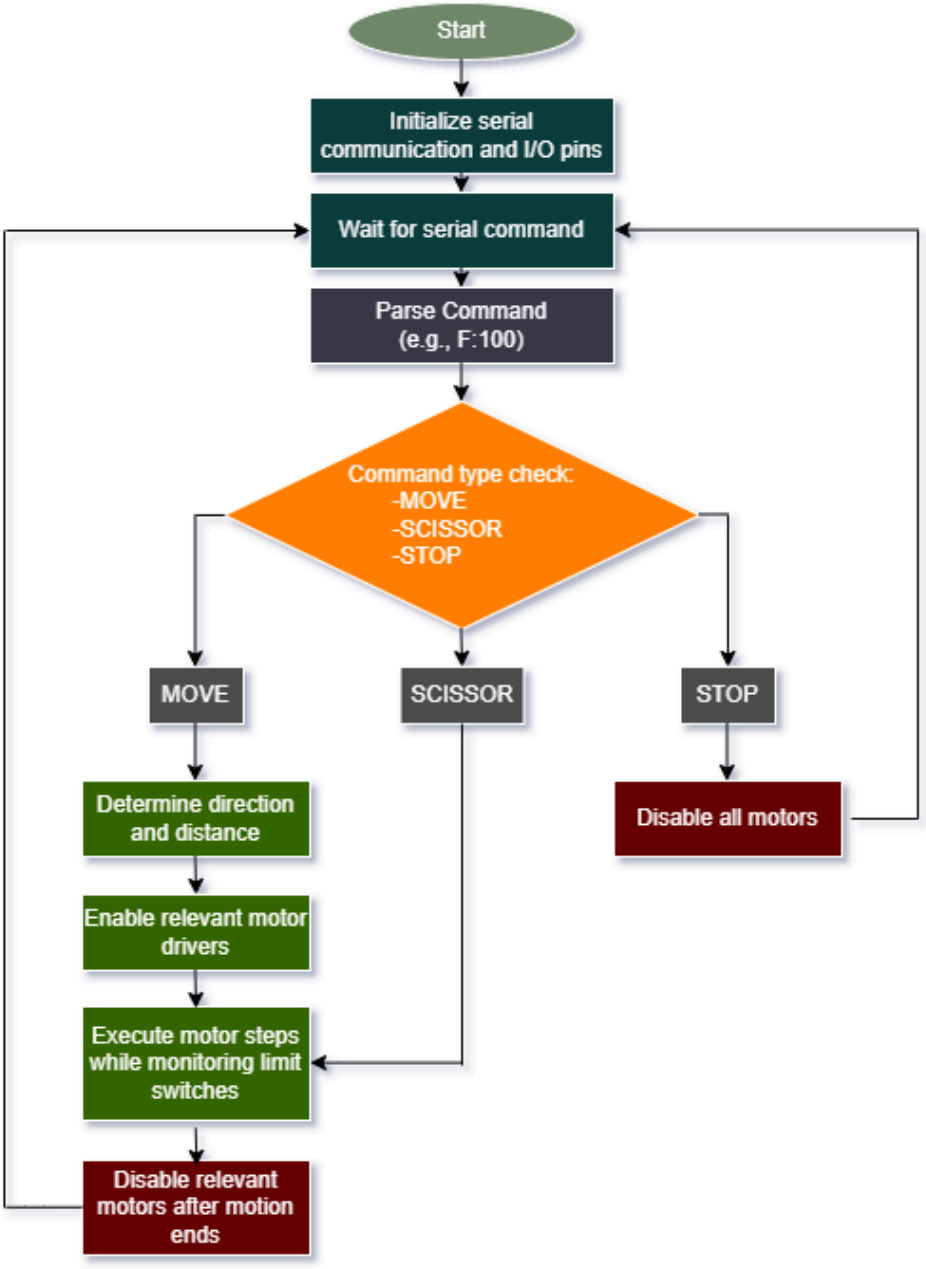


Figure 28: Arduino code flowchart for command-based robotic control.

5.3 Python Implementation for Machine Vision and Decision-Making

The Python code is responsible for fruit detection, classification, and decision-making regarding the robot's movement. The implementation utilizes YOLOv5, a deep learning-based object detection algorithm, to identify ripe tomatoes and peppers based on color, size, and spatial positioning.

The Python code is structured into two main modules:

1. **detect.py** – The primary script for object detection and decision-making.
2. **utilities.py** – A utility module containing helper functions for image processing, fruit classification, and robotic movement coordination.

Figure 29 illustrates the functional relationship between detect.py and utilities.py, showing how the main script relies on utility functions for tasks such as ripeness evaluation, duplicate filtering, and position calculation.

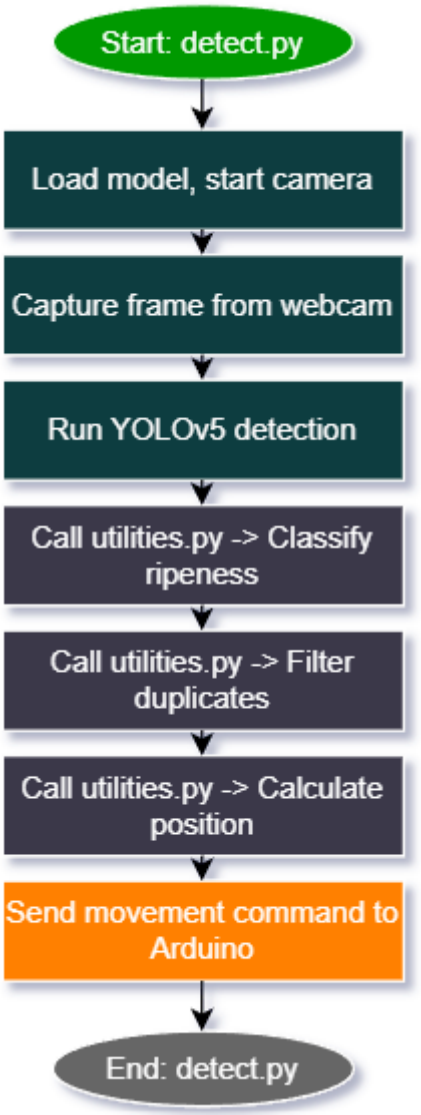


Figure 29: Functional interaction between detect.py and utilities.py in the vision system.

5.3.1 Object Detection and Classification

The detect.py script is responsible for real-time object detection and classification, guiding the autonomous harvesting robot in recognizing and processing tomatoes and peppers. This module is a fundamental component of the system, as it enables precise identification of harvestable vegetables, ensuring that only ripe produce is collected. The script utilizes YOLOv5, a state-of-the-art deep learning model for object detection, to perform inference on image or video streams.

The detection module performs multiple functions:

- Capturing image or video data from an external camera.
- Processing and analyzing frames using a trained neural network.
- Determining the location and ripeness of the detected vegetables.
- Sending real-time commands to the robotic system for movement and harvesting.
- Implementing safety mechanisms to prevent mechanical errors.

This chapter details the implementation of the object detection pipeline, its integration with the robotic system, and the decision-making process for harvesting.

System Architecture and Workflow

The object detection module follows a structured pipeline consisting of several key stages. Each stage plays a crucial role in ensuring accurate and efficient vegetable recognition and robotic control.

1) Input Data Acquisition

The system supports multiple data sources, including:

- Live Camera Feed: Real-time image frames from a connected camera.
- Pre-recorded Video Files: Processing of stored video footage.
- Image Datasets: Analysis of static images for testing and validation.

The input data is continuously streamed and preprocessed before being fed into the detection model.

2) Loading the YOLOv5 Model

The detection system employs the YOLOv5 deep learning model, which is pre-trained to recognize tomatoes and peppers. This model is selected due to its high inference speed and accuracy in object detection tasks. The neural network is optimized to detect multiple vegetables within a single frame while filtering out unnecessary background information.

3) Image Preprocessing and Feature Extraction

Before running the inference, the input images undergo several preprocessing steps:

- Resizing: Adjusting the image dimensions to match the model's expected input size.
- Normalization: Scaling pixel values for improved neural network performance.
- Color Space Conversion: Converting images to the appropriate format for feature extraction.

These preprocessing techniques ensure that the model receives clean and standardized input data for accurate predictions.

4) Object Detection and Non-Maximum Suppression (NMS)

Once the image is processed, the YOLOv5 model performs inference to detect objects of interest. The model generates bounding boxes, confidence scores, and class labels for each detected object.

To refine the detection results, Non-Maximum Suppression (NMS) is applied. This algorithm eliminates redundant overlapping detections and retains only the most relevant bounding boxes, ensuring that each vegetable is counted only once.

5) Decision-Making Process

The harvesting robot must decide whether to harvest a detected vegetable based on several key factors. After detecting a vegetable, the system evaluates its ripeness using an image classification technique known as ripeness classification. This classification is based primarily on two features:

- Color analysis: Identifying shades of red for tomatoes or yellow/orange for peppers, which are indicative of maturity.
- Texture recognition: Differentiating between mature and unripe produce based on surface texture patterns visible in the captured image.

If the detected vegetable meets the ripeness threshold according to these criteria, it is added to the harvesting list. Otherwise, it is ignored, ensuring selective harvesting and preventing premature or unripe vegetables from being picked.

6) Robot Navigation and Position Adjustment

For efficient harvesting, the robotic system must move into the correct position relative to the detected vegetable. The system follows a structured movement approach:

- Determine the vegetable's coordinates within the camera frame.
- Calculate the robot's movement trajectory to align with the target.
- Adjust the robot's position by moving forward, left, or right based on detection data.
- Verify position accuracy before proceeding with harvesting.

The robot continuously refines its position until it is correctly aligned with the target vegetable.

7) Harvesting Execution

Once the robot reaches the optimal harvesting position, the cutting mechanism is activated. The robotic arm extends towards the detected vegetable and performs a precise cutting motion using a scissor-like mechanism. The system updates the status of harvested vegetables and removes them from the detection list, preventing duplicate processing.

8) Safety Mechanisms and Error Handling

To ensure safe and reliable operation, the detection system incorporates multiple safety checks.

A limit switch is installed on the robotic arm to detect obstructions. If the switch is triggered, the system:

- Immediately halts all robotic movement.
- Prevents further cutting actions.
- Logs an error message and waits for manual intervention or re-adjustment.

If an emergency stop command is issued, the detection system:

- Stops all motors and robotic actions.
- Saves the current operational state.
- Awaits user confirmation before resuming operations.

These measures ensure that the system does not cause unintended damage to plants or mechanical components.

The flowchart illustrated in Figure 30 outlines the step-by-step operation of the object detection and harvesting system implemented in the detect.py script. The process begins with the system initialization, where image frame processing is prepared. The YOLOv5 model is then loaded into memory to enable neural network inference. Following this, the system continuously captures input frames from the camera. Each frame undergoes object detection using the YOLOv5 model to identify target crops such as tomatoes and peppers. To refine these detections, Non-Maximum Suppression (NMS) is applied to eliminate redundant or overlapping bounding boxes.
Detected vegetables are subsequently processed to evaluate their ripeness. If a vegetable is determined to be ripe, the robot calculates and adjusts its position accordingly to align itself with the target. Once correctly aligned, the harvesting mechanism is activated to perform the cutting operation. The system then logs the harvested item and updates the internal dataset. Finally, the cycle concludes with the system returning to a standby state, ready to process the next detection frame. This structured workflow ensures that the robot performs reliable, accurate, and safe harvesting actions during each operational loop.

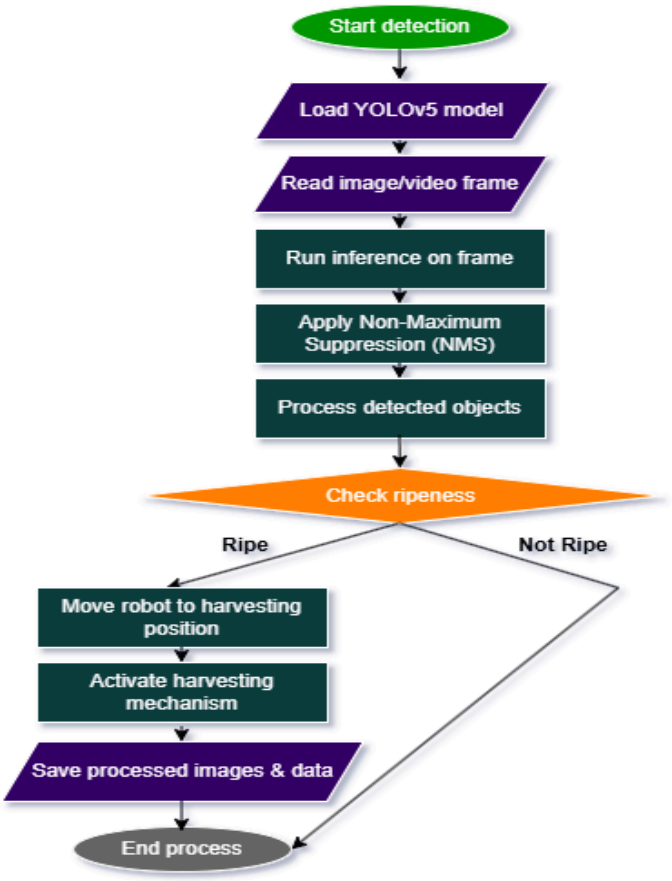


Figure 30: Flowchart of the object detection process.

State-Based Decision-Making for Harvesting

This section explains the decision-making mechanism in the object detection script, where the robot determines its next action based on the detected vegetables. The state machine approach is used to transition between different operational phases, including detection, adjustment, and harvesting. This method ensures efficient and structured control over the harvesting process. The overall logic is summarized in the state transition diagram shown in Figure 30, which illustrates how the system moves between key states based on detection results and sensor feedback.

Overview of the State-Based System

The decision-making logic operates through a state machine, where the system transitions between different modes based on the detection results and real-time sensor feedback. The system follows these primary states:

- **DETECT:** Identifies vegetables and evaluates ripeness.
- **ADJUST_APPROACH:** Aligns the robot for optimal harvesting.
- **HARVESTING:** Activates the cutting mechanism to harvest the detected vegetable.
- **IDLE:** A fallback state when no detections or actions are required.

The robot starts in the DETECT state and transitions based on detection results and limit switch triggers.

1) Detecting and Evaluating Vegetables

The DETECT state is the initial phase where the system processes image frames, detects objects, and evaluates their ripeness.

A. Vegetable Detection

- The YOLOv5 model identifies objects in the frame and provides bounding box coordinates.
- The center position of each detected vegetable is calculated for further processing.

B. Ripeness Evaluation

- A classification function determines whether the detected vegetable is ripe.
- Only ripe vegetables are considered for harvesting.

C. Duplicate Check

- The system verifies if the detected vegetable was already processed.
- If the vegetable is new, its coordinates are added to the harvesting list.

D. State Transition

- If a new, ripe vegetable is detected, the system transitions to ADJUST_APPROACH.
- If no valid detections are found, the robot moves forward and continues scanning.

2) Adjusting the Robot's Position

The ADJUST_APPROACH state ensures the robot is correctly aligned with the vegetable before harvesting.

A. Position Correction

- The robot calculates the necessary adjustments based on the vegetable's coordinates.
- It moves left or right to precisely position the cutting mechanism.

B. State Transition

- Once the robot is aligned, it transitions to the HARVESTING state.

3) Executing the Harvesting Process

The HARVESTING state activates the robotic arm to perform the cutting operation.

A. Cutting Mechanism Activation

- The cutting mechanism is moved to the vegetable's height.
- The system initiates the cutting process to detach the vegetable from the plant.

B. Completion and Reset

After harvesting, the system removes the vegetable from the harvesting list. The system resets to the DETECT state for the next cycle.

4) Handling the Idle State

The IDLE state acts as a default or fallback condition.

- If no vegetables are detected or all harvesting tasks are completed, the system enters to the IDLE mode.
- The robot remains stationary until new detections prompt further action.

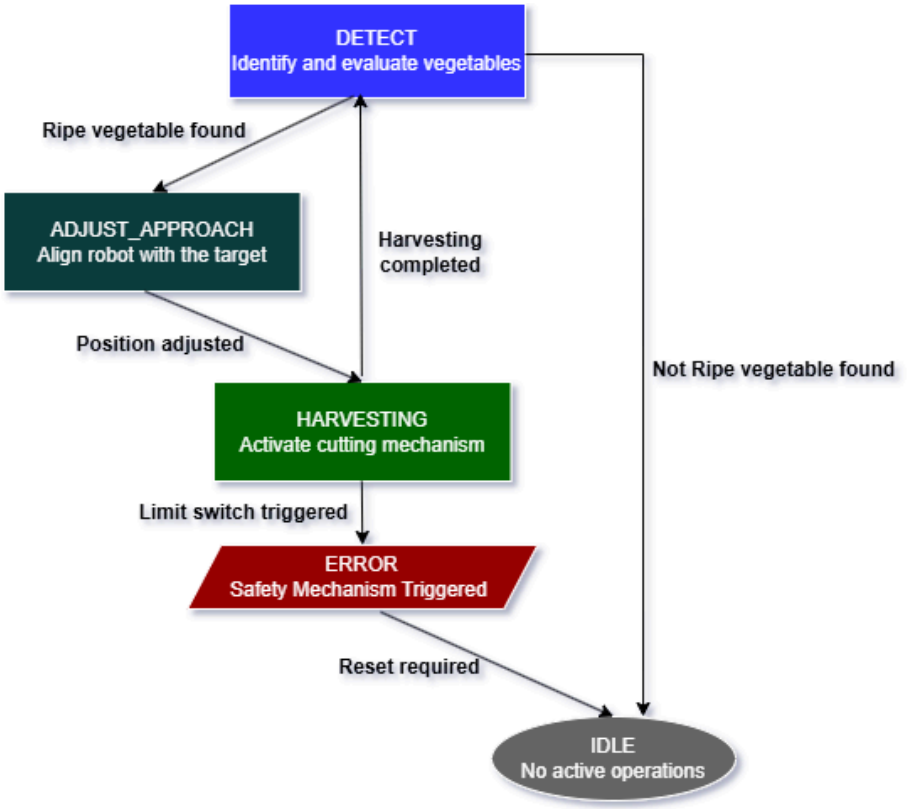


Figure 31: State transition diagram for the harvesting process.

5.3.2 Implementation of Image Processing and Robotic Control Functions

The second Python module, contains a suite of specialized utility functions that underpin the core operational logic of the robotic harvesting system. This script plays a pivotal role in the post-detection processing pipeline by managing key tasks such as ripeness evaluation, duplicate coordinate filtering, positional correction, and movement orchestration of the robotic arm. The functionality encapsulated in this module is divided into the following categories:

1) Ripeness Detection and Classification

One of the key functionalities of the harvesting robot is the ability to classify vegetables based on their ripeness. This is achieved using color-based segmentation applied to images of detected vegetables.

Ripeness Evaluation Methodology

For each detected vegetable, the system:

1. Extracts the vegetable image from the captured frame.
2. Converts the image to the HSV color space for better color segmentation.
3. Applies a color filter to identify specific ripeness conditions.
4. Calculates the percentage of relevant color pixels in the image.
5. Determines whether the vegetable is ripe based on the percentage threshold.

Ripeness Criteria

The system applies different color thresholds depending on the vegetable type:

- Tomatoes: The system checks for red pigmentation, ensuring that the detected tomato has reached the required maturity level.
- Peppers: The system evaluates the presence of green pixels to distinguish between ripe and unripe peppers.

After calculating the percentage of relevant color pixels, the system applies the following rules:

- Tomatoes: If red color coverage is greater than 25%, the tomato is considered ripe.
- Peppers: If green color coverage is greater than 50%, the pepper is considered ripe.

If the vegetable meets these conditions, it is added to the harvesting list. Otherwise, it is ignored.

2) Duplicate Coordinate Detection

To prevent redundant harvesting, the system checks whether the detected vegetable is a new candidate or a duplicate detection.

Distance-Based Filtering

- Each detected vegetable's center coordinates are stored in a harvesting list.
- When a new vegetable is detected, the system calculates the Euclidean distance between the new detection and previously stored vegetables.
- If the distance is greater than a predefined threshold, it is considered a new vegetable and added to the list.
- If the vegetable is too close to an existing detection, it is discarded as a duplicate.

This process ensures that the robot does not attempt to harvest the same vegetable multiple times.

3) Robotic Position Adjustment

Once a ripe vegetable has been identified, the robot must align itself precisely before harvesting. The adjustment process consists of two key movements:

- Horizontal (X-axis) Adjustment: The robot moves left or right to center itself.
- Vertical (Y-axis) Adjustment: The robotic arm moves up or down to align the cutting mechanism.

Position Adjustment Process

To achieve accurate alignment, the system performs the following steps:

1. Calculates the vegetable's center coordinates.
2. Compares them with the camera's center position.
3. Determines the necessary movement distance in centimeters.
4. Sends movement commands to the robot to align it properly.
5. Verifies alignment and transition to the harvesting phase.

This structured adjustment sequence ensures that the robotic cutter is precisely positioned before initiating the harvesting operation, minimizing the risk of damaging the vegetable or surrounding plant structures.

4) Robotic Movement Control

The system includes low-level motor control functions that allow the robot to move in various directions. The robot operates with serial communication to receive movement commands.

Motion Control Commands

The following movement commands are sent to the Arduino-based motor control system:

- Move Forward: The robot moves towards the detected vegetable.
- Move Backward: The robot retreats after a harvesting attempt.
- Move Left/Right: The robot aligns itself laterally.
- Turn Left/Right: The robot rotates in place to adjust its orientation.
- Move Up/Down: The robotic arm adjusts its height to align with the vegetable.

Each movement is executed using predefined distances, ensuring precision in navigation and harvesting.

Movement Execution Process

- The system calculates the required movement distance.
- The appropriate movement function is triggered, sending a serial command to the Arduino.
- The robot executes the movement and confirms completion.
- If necessary, the movement is repeated until proper alignment is achieved.

5) Harvesting Execution

After the robot is aligned with the vegetable, the cutting mechanism is activated.

Cutting Mechanism Process

1. The robot moves slightly backward to position itself for cutting.
2. The robotic arm moves upwards to the vegetable's height.
3. The scissor mechanism opens to prepare for cutting.
4. The robot maneuvers into the cutting position
5. The scissor mechanism closes, detaching the vegetable.
6. The harvested vegetable is collected, and the system resets for the next detection.

Safety Measures in Cutting

- A limit switch sensor prevents excessive movement.
- The system verifies the vegetable's removal before proceeding to the next cycle.

This entire harvesting workflow is visually summarized in Figure 32, which presents a structured flowchart beginning with ripeness detection and ending with the final cutting action. The chart below illustrates the structured workflow of the harvesting process, starting from ripeness detection to the final cutting action. The system begins by detecting the ripeness of the vegetable using color-based segmentation. If the vegetable is classified as ripe, the system checks for duplicate detections to avoid redundant harvesting. Once confirmed as a new detection, the robot aligns itself with the vegetable by adjusting its horizontal and vertical position. The cutting mechanism is then activated, and the robot performs the cutting action. After verifying the success of the harvest, the system resets for the next detection.

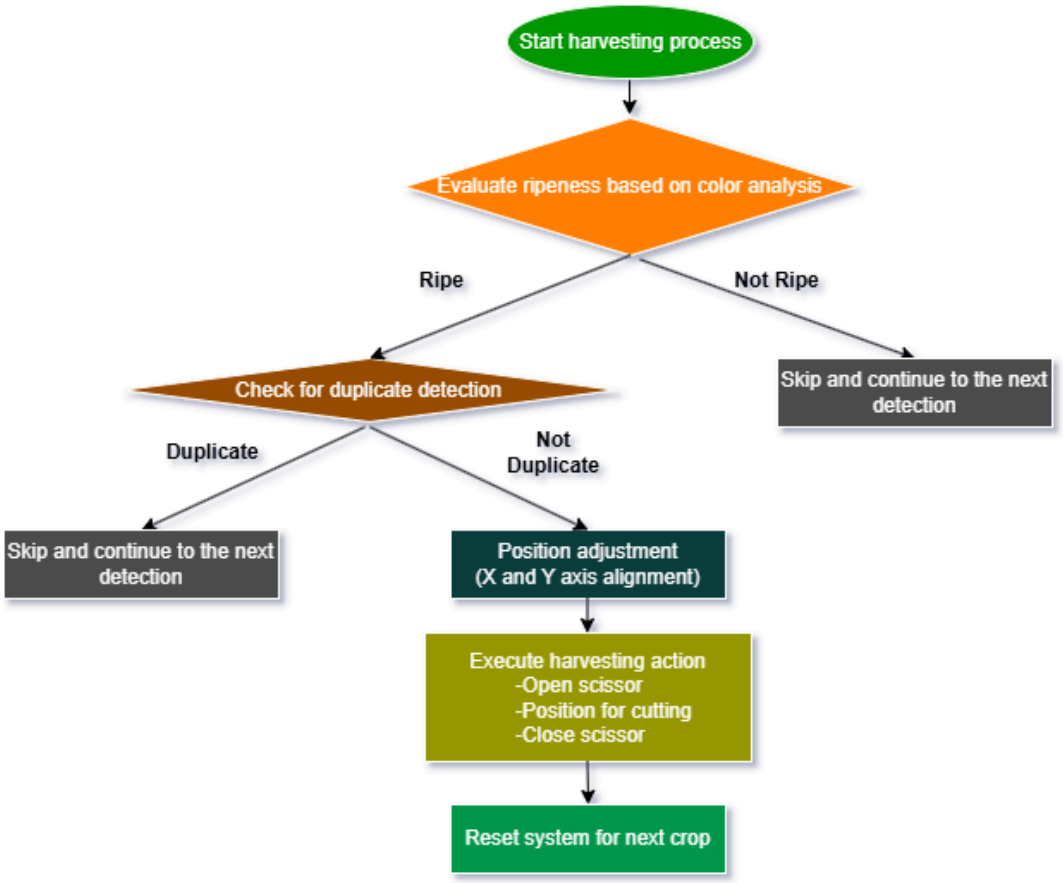


Figure 32: Flowchart of the harvesting process.

Chapter 6: Final Results

6.1 Introduction

This chapter presents the final results of the implemented robotic harvesting system, emphasizing its performance and validating its functionality through experimental testing. The primary objective is to demonstrate that the integrated hardware–software platform can autonomously detect and harvest ripe tomatoes and peppers using a combination of computer vision and precise motion control. These results confirm that the robotic prototype operates according to design specifications and achieves the intended functional requirements outlined during the initial design phase.

The final prototype exhibited high accuracy in identifying and harvesting ripe crops, significantly reducing the need for human intervention and improving operational efficiency. Experimental evaluations were carried out in both a laboratory table setup and a greenhouse environment, verifying that the robot could correctly detect target crops, adjust its position, and execute harvesting actions without damaging nearby structures or unripe produce. The use of a stepper motor–driven scissor mechanism enabled reliable and precise cutting, ensuring consistent performance during repeated trials.

6.2 System Overview and Integration Summary

The developed system successfully integrates mechanical components, electronic control units, and artificial intelligence algorithms into a cohesive prototype capable of real-time operation. The robot is powered by NEMA17 stepper motors controlled via an Arduino Mega 1280, interfaced with A4988 motor drivers, which enable microstepping and current regulation for smooth and efficient actuation.
The vision component is based on a YOLOv5 deep learning architecture trained to detect ripe tomatoes and peppers. Communication between the Python-based detection logic and the Arduino motor control system is established via a serial interface, allowing synchronized execution of detection, alignment, and cutting commands. A labeled image showing the final assembly of the robot and its main components is presented in Figure 33.

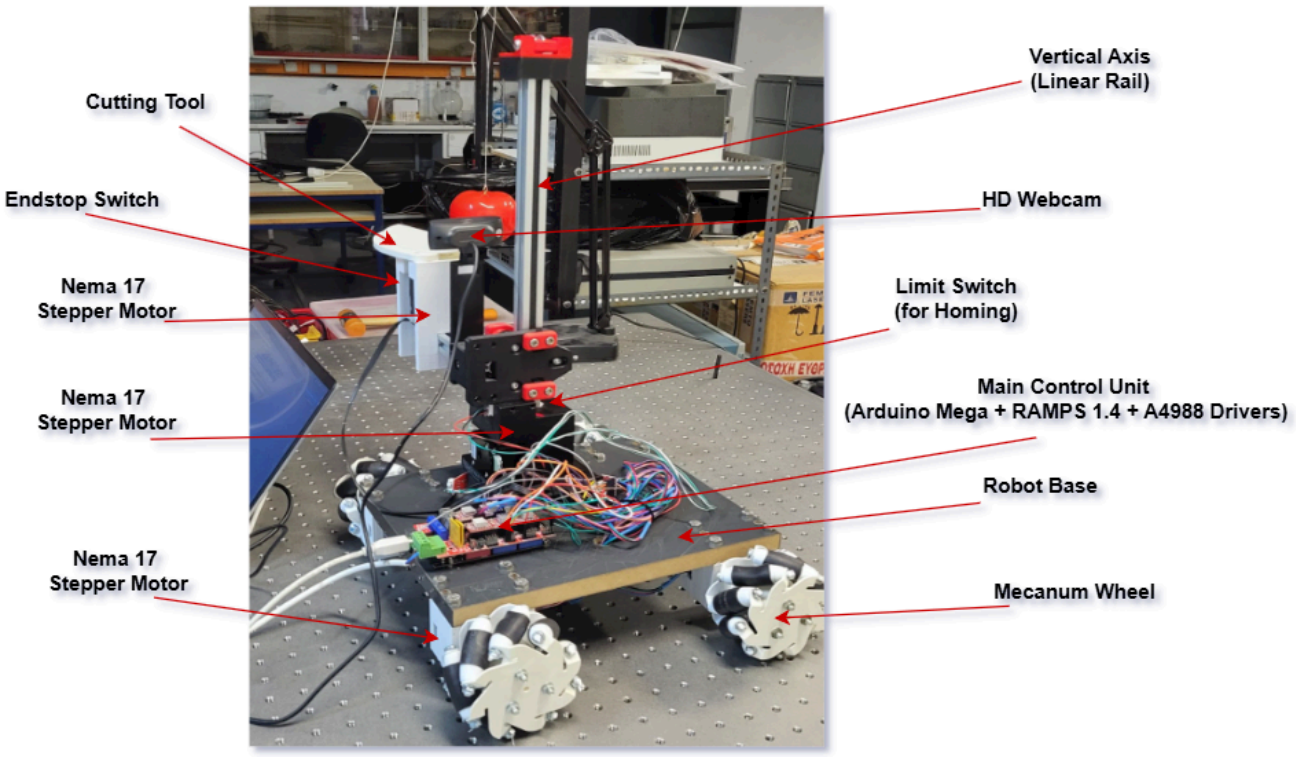


Figure 33: Final assembly of the robotic harvester with labeled components.

6.3 Visual Output from the Detection Pipeline

Before transitioning to lab-based integration, early testing of the detection algorithm was conducted outdoors in a greenhouse environment to evaluate its effectiveness under natural lighting and real agricultural conditions. These preliminary tests focused on confirming the system's ability to identify ripe vegetables, determine their position, and assess ripeness using HSV color filtering.

Figure 34 demonstrates the detection of a ripe tomato, where the YOLOv5 model has accurately drawn a bounding box and assigned a high confidence score, confirming successful classification. Similarly, Figure 35 highlights the detection of a ripe pepper. The center of the bounding box is visualized, showing the coordinate used for robot alignment during harvesting.

These visual outputs provide direct insight into the internal processing pipeline, from object recognition to spatial localization, and affirm that the vision system performs reliably across different crops and environments.

Figure 34: Tomato detection with bounding box.

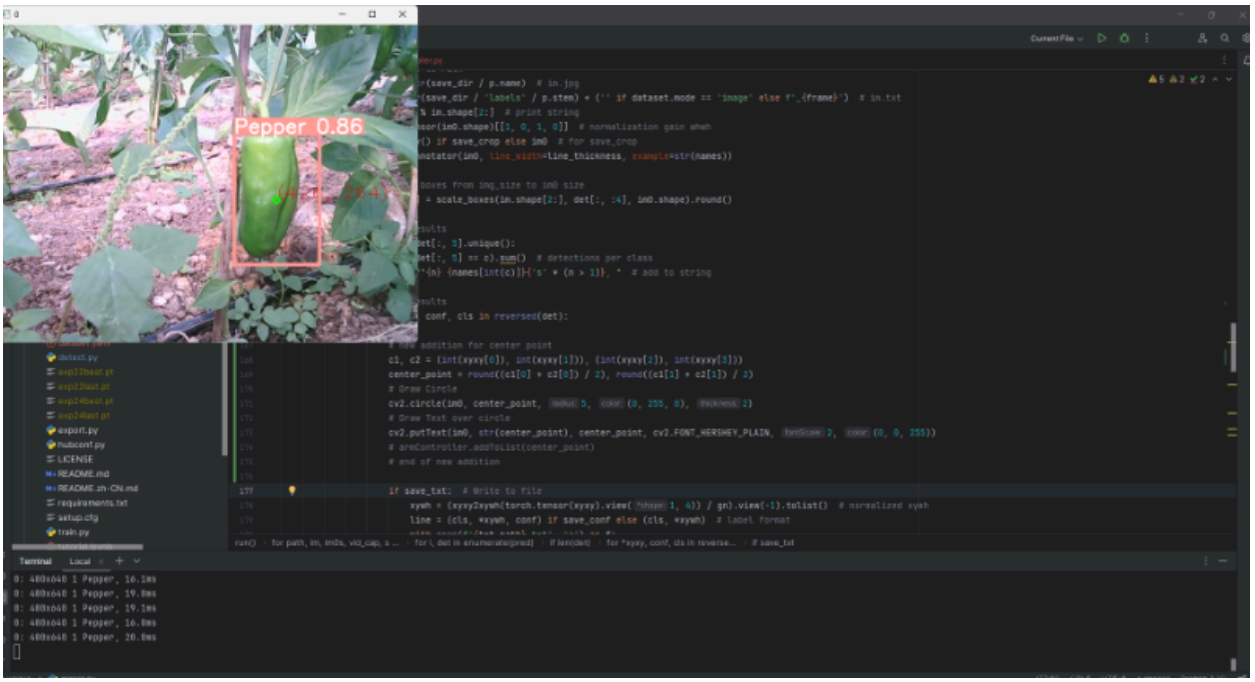


Figure 35: Detected pepper and center point localization.

6.4 Real-Time Functionality of the Harvesting System

To validate the real-time functionality of the robotic harvesting system, both software outputs and physical responses were monitored during controlled test sessions conducted on a laboratory table setup. These outputs demonstrate that the robot was successfully executing its intended operations—detecting target vegetables, classifying ripeness, issuing motion commands, and actuating the cutting mechanism accordingly.

The screenshot presented in Figure 34 captures real-time log output from the Python terminal during a functional test using a tomato placed on the laboratory table. The log displays key events such as successful object detection, ripeness classification, and the issuance of corresponding serial commands (e.g., movement and cutting instructions) to the Arduino controller. This output demonstrates the system's real-time decision-making capability and confirms the effective integration between the machine vision module and the robotic control hardware.

```
140     # Dataloader
141     bs = 1  # batch_size
142     if webcam:
143         view_img = check_imshow(warn=True)
144         dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt, vid_stride=vid_stride)

OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

Initializing robotic harvesting system...
Camera stream started.
Status: DETECT
No vegetables detected.
Moving forward...
Status: DETECT
Image captured.
Vegetable detected: tomato
Bounding box: Top-Left (240, 180), Bottom-Right (310, 250)
Center calculated at (275, 215)
Red color percentage: 36.7%
Tomato is classified as ripe.
Checking for duplicates in harvest list...
No duplicates found.
Vegetable added to the list.
Switching status to: ADJUST_APPROACH
Status: ADJUST_APPROACH
Adjusting alignment with detected tomato...
Moving LEFT by 4.2 cm to align with X-axis.
Distance too close. Re-adjusting...
Moving RIGHT by 1.5 cm for safe offset.
Switching status to: HARVESTING
Status: HARVESTING
Starting cutting process...
Raising scissor to 7.2 cm based on top boundary of detection.
Checking contact limit switch...
Limit switch triggered
Opening scissor...
Cutting in progress...
Cutting process finished successfully.
```

Figure 36: Real-time console output demonstrating system integration.

Chapter 7: Conclusions, and Recommendations for Future Research

7.1 Conclusions

The research and development of this robotic harvesting system demonstrated the feasibility and effectiveness of robotic automation in agricultural harvesting. One of the main outcomes of this thesis was the successful integration of computer vision and robotic motion. The combination of a deep learning-based detection model with real-time robotic control enabled efficient and selective harvesting. The YOLOv5 algorithm provided accurate fruit detection, while the robotic arm responded to visual inputs with precise movements. This confirms that computer vision technologies can be reliably applied to agricultural automation.

Another significant achievement was the optimized motion control system, which utilized NEMA17 stepper motors and A4988 drivers. With proper tuning of current limits and microstepping configurations, the robotic arm achieved smooth, high-precision movements with minimal vibration—critical for interacting with delicate produce. Additionally, the implementation of serial communication between the Arduino and the Python-based detection system ensured seamless, real-time coordination between software decisions and hardware actuation, underlining the importance of robust communication protocols in autonomous robotic platforms.

The project also confirmed the benefits of data preprocessing and augmentation in improving detection accuracy. Techniques such as rotation and noise addition contributed to the robustness of the YOLOv5 model when exposed to different visual scenarios, reinforcing the value of a diverse and well-prepared dataset.

However, several challenges were encountered when evaluating the system in more complex environments. While the robot performed reliably during indoor tests and showed promising results in outdoor greenhouse conditions—accurately detecting and localizing both tomatoes and peppers—its performance could still be improved. Factors such as dense foliage, partial occlusions, and varying lighting introduced occasional detection inconsistencies. These observations suggest that while the current system provides a solid foundation, future enhancements such as the integration of 3D vision systems or additional sensors could further increase robustness and reliability in unstructured agricultural settings.

7.2 Recommendations for Future Research

While the developed system represents a significant step towards autonomous robotic harvesting, further improvements and research directions can enhance its capabilities and practical implementation. The following areas are recommended for future exploration:

1. Advanced 3D Vision and Multi-Sensor Fusion

- The integration of LiDAR or stereo vision cameras could provide depth perception, allowing the robot to navigate more effectively in complex environments.
- Multi-sensor fusion, combining RGB, infrared, and depth cameras, could improve fruit detection under low-light conditions and dense foliage.

2. Enhanced Deep Learning Models for Detection

- The current YOLOv5 model could be upgraded to YOLOv8 or other state-of-the-art architectures with improved accuracy and efficiency.
- Implementing self-learning AI models that adapt to different agricultural environments could increase robustness in varied conditions.

3. Precision Control Mechanisms for Robotic Manipulation

- The incorporation of force sensors and haptic feedback systems could refine the cutting and grasping mechanisms, reducing potential fruit damage.
- Implementing adaptive control algorithms could enable the robot to adjust its grip and cutting force based on fruit size and stem thickness.

4. Autonomous Navigation and Path Planning

- Currently, the system follows basic movement commands based on object detection. Integrating autonomous navigation using SLAM (Simultaneous Localization and Mapping) would allow the robot to move independently within a greenhouse or field.
- The development of AI-driven motion planning algorithms could optimize harvesting efficiency and minimize unnecessary movements.

5. Improved Hardware Design for Scalability

- A more lightweight and modular robotic arm could be designed to increase flexibility in handling different crop types.
- The addition of battery-powered operation with energy-efficient components could enable deployment in larger agricultural fields without constant external power sources.

6. Internet of Things (IoT) and Cloud-Based Monitoring

- The integration of IoT-based sensors could allow real-time monitoring of environmental conditions, such as temperature, humidity, and crop growth patterns.
- Cloud-based data collection and AI-driven analytics could optimize harvesting schedules based on real-time crop status.

7. Testing and Deployment in Real-World Agricultural Fields

- Expanding the testing environment beyond greenhouses to open-field conditions would validate the system's robustness against weather variations and unpredictable terrain.
- Collaboration with agricultural industries and research institutions could facilitate large-scale deployment and continuous improvement.

The implementation of robotic automation in agricultural harvesting presents a promising solution to labor shortages, increased demand for food production, and the need for precision agriculture. This study successfully demonstrated that a vision-based robotic system can efficiently detect and harvest crops, reducing manual labor and improving efficiency.

However, agricultural robotics remains a rapidly evolving field, and continuous research is required to overcome technical challenges, scalability issues, and real-world deployment constraints. Future advancements in computer vision, AI, robotics, and IoT technologies will play a crucial role in shaping the next generation of autonomous farming systems.

By further developing intelligent, adaptive, and cost-effective robotic solutions, the agricultural industry can move toward fully automated and sustainable farming practices, ensuring higher productivity, reduced waste, and improved food security worldwide.

References

[1] P. M. Blok et al., "Vision based fruit recognition and positioning technology for robotic harvesting," Computers and Electronics in Agriculture, vol. 75, no. 1, pp. 1–12, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0168169923006464

[2] Agriculture robotic arm for apple harvesting. [Online]. Available: https://www.gettyimages.com/videos/fruit-picking-robot

[3] A. Koirala et al., "Deep learning – Method overview and review of use for fruit detection and yield estimation," Computers and Electronics in Agriculture, vol. 162, pp. 219–234, 2019. [Online]. Available: https://www.researchgate.net/publication/332493627

[4] Y. Tang et al., "Three-dimensional perception of orchard banana central stock enhanced by adaptive multi-vision technology," Computers and Electronics in Agriculture, vol. 177, 105708, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0168169920302957

[5] C. Lehnert et al., "Performance improvements of a sweet pepper harvesting robot in protected cropping environments," Journal of Field Robotics, vol. 35, no. 6, pp. 930–945, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21973

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, 2017. [Online]. Available: https://dl.acm.org/doi/10.1145/3065386

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, pp. 436–444, 2015. [Online]. Available: https://www.nature.com/articles/nature14539

[8] M. A. Hossain and R. Chellali, "A survey on computer vision-based automated fruit harvesting systems," Journal of Imaging, vol. 6, no. 9, 2020. [Online]. Available: https://www.mdpi.com/2313-433X/6/9/92

[9] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," Computers and Electronics in Agriculture, vol. 147, pp. 70–90, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169917314710

[10] L. Bottou, "Stochastic Gradient Descent Tricks," in Neural Networks: Tricks of the Trade, Springer, 2012. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-35289-8_25

[11] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” arXiv preprint, 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[12] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/

[13] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” arXiv preprint, Apr. 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[14] G. Jocher et al., “YOLOv5 by Ultralytics,” GitHub repository, 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[15] OpenCV Team, “Trackbar for HSV thresholding in OpenCV,” LearnOpenCV, 2019. [Online]. Available: https://learnopencv.com/color-spaces-in-opencv-cpp-python

[16] A. S. Martin, “Segmentation and Classification with HSV,” Medium – NeuroSapiens, Oct. 2020. [Online]. Available: https://medium.com/neurosapiens/segmentation-and-classification-with-hsv-8f2406c62b39

[17] Valentinos Papaiakovou, “Deep Learning Model for Plant Detection and Harvesting”, Diploma thesis, School of Electrical & Computer Engineering, Technical University of Crete, Chania, Greece, 2025. (Unpublished).

[18] Petros Panigirakis, ” Design, construction, and programming of a robotic arm for Efficient Fruit Picking in Greenhouses”, Diploma thesis, School of Production Engineering and Management, Technical University of Crete, Chania, Greece, 2025. (Unpublished).

[19] "Stepper Motors Basics," Texas Instruments. Available: https://www.ti.com/lit/an/slva980/slva980.pdf

[20] STMicroelectronics, “Understanding Stepper Motor Fundamentals,” Application Note CD00274221, 2011. [Online]. Available: https://www.st.com/resource/en/application_note/cd00274221-motor-control-primer-stmicroelectronics.pdf

[21] External View of a NEMA17 Bipolar Stepper Motor - Pololu Robotics and Electronics, “Stepper Motor: Bipolar, 200 Steps/Rev, 42×38mm, 4V, 1.2 A/Phase,” [Online]. Available: https://www.pololu.com/product/2267

[22] D. Micić, “Stepper Motors and Arduino – The Ultimate Guide,” HowToMechatronics, [Online]. Available: https://howtomechatronics.com/tutorials/arduino/stepper-motors-and-arduino-the-ultimate-guide/

[23] Arduino, “Arduino Mega 1280 Board Overview,” Arduino.cc, [Online]. Available: https://docs.arduino.cc/hardware/mega-1280

[24] RepRap, “RAMPS 1.4 - RepRap Arduino Mega Pololu Shield,” RepRap Wiki, [Online]. Available: https://reprap.org/wiki/RAMPS_1.4

[25] Arduino Mega 1280: Pinout and Functional Overview - M. Kisse, “Arduino Mega Pinout & Functions,” Weebly Blog, [Online]. Available: https://mkisse.weebly.com/blog/arduino-mega-pinout-functions

[26] RAMPS 1.4 Shield Pinout Diagram for Arduino Mega Integration - Osoyoo, “RepRap 3D Printer Circuit Connection Graph,” Osoyoo, Jul. 3, 2016. [Online]. Available: https://osoyoo.com/2016/07/03/reprap-3d-printer-circuit-connection-graph/

[27] Pololu, “A4988 Stepper Motor Driver Carrier,” [Online]. Available: https://www.pololu.com/product/1182

[28] Allegro Microsystems, “A4988: DMOS Microstepping Driver with Translator and Overcurrent Protection,” [Online]. Available: https://www.allegromicro.com/en/products/motor-drivers/stepper-motor-drivers/a4988

[29] Pinout Diagram of the A4988 Stepper Motor Driver Module - “Interface A4988 Stepper Motor Driver with Arduino,” Circuit Digest, [Online]. Available: https://circuitdigest.com/microcontroller-projects/interface-a4988-stepper-motor-driver-with-arduino

[30] Circuit Digest, “Interface A4988 Stepper Motor Driver with Arduino,” [Online]. Available: https://circuitdigest.com/microcontroller-projects/interface-a4988-stepper-motor-driver-with-arduino

[31] SparkFun Electronics, “How to Use a Stepper Motor,” [Online]. Available: https://learn.sparkfun.com/tutorials/hobby-motor-and-stepper-motor-drivers/all

[32] Schematic Diagram of the A4988 Stepper Motor Driver - “Schematic Diagram of the A4988 Stepper Motor Driver,” Pololu, [Online]. Available: https://www.pololu.com/picture/view/0J3359

[33] "A4988 Stepper Motor Driver Module," Envistia Mall Support. [Online]. Available: https://support.envistiamall.com/kb/a4988-stepper-motor-driver-module/

[34] M. Margolis, AccelStepper Library for Arduino. [Online]. Available: https://www.airspayce.com/mikem/arduino/AccelStepper/