A

Project Report

On

"AGRICULTURAL AID FOR TOMATO HARVESTING (AAT)"

Submitted in partial fulfillment of the requirement for the award of a Degree of

BACHELOR OF TECHNOLOGY

Submitted By

Konam Pavan Kartheek (18K91A04A2)

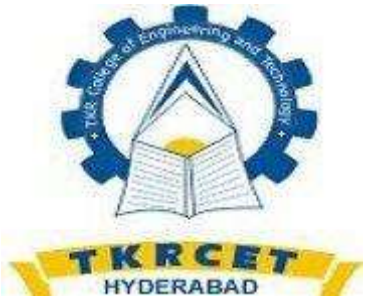
J. Manichandra (18K91A0478)

Garlapati Rohith Reddy (18K91A0464)

Under The Guidance Of

Dr. M. Girish Kumar

Associate Professor, ECE Dept.



2018-2022

TKR COLLEGE OF ENGINEERING AND TECHNOLOGY

[AUTONOMOUS]

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

(Accredited By NBA, Approved by AICTE)

Accredited by NAAC with an "A" grade

(MEDBOWLI, MEERPET, SAROORNAGAR, HYDERABAD – 500079)

TKR COLLEGE OF ENGINEERING AND TECHNOLOGY
(Sponsored by TKR Educational Society, Approved by AICTE, Affiliated by JNTUH)
Autonomous, Accredited by NAAC with 'A' Grade. Accredited by NBA
Medbowli, Meerpet, Saroornagar, Hyderabad - 500 097
Phone: 9100377790, e-mail: info@tkrcet.ac.in website: www.tkrcet.ac.in

College Code : K9

CERTIFICATE

This is to certify that the Major Project entitled **"AGRICULTURAL AID FOR TOMATO HARVESTING (AAT)"**

Submitted

By

KONAM PAVAN KARTHEEK	**18K91A04A2**
J. MANICHANDRA	**18K91A0476**
GARLAPATI ROHITH REDDY	**18K91A0464**

In partial fulfillment of the requirements for the degree of Bachelor of Technology in Electronics & Communication Engineering by the JNTU, Hyderabad during the academic year 2021-2022.

Dr. M. GIRISH KUMAR	**Dr. P. GAYATRI**	**Dr. D. NAGESHWAR RAO**
Associate Professor	Associate Professor	Head of the Department
Co-Ordinator, Internal guide	Co-Ordinator	ECE

External Examiner:

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance have crowned our efforts with success.

We extend my deep sense of gratitude to Principal **Dr. D. V. Ravi Shankar** TKR College of Engineering and Technology, Meerpet, for permitting us to undertake this project.

We are indebted to **Dr. D. Nageshwar Rao**, Professor, and Head of the Department, of Electronics and Communication Engineering, TKR College of Engineering and Technology, Meerpet, for his support and guidance throughout our project.

We are indebted to our guide, **Dr. Girish Kumar**, Associate Professor, Department of Electronics and Communication Engineering, TKR College of Engineering and Technology, Meerpet, for his constant support and guidance throughout our project.

We are indebted to the project coordinators, **Dr. P. Gayathri**, Associate Professor, Electronics and Communication Engineering, TKR College of Engineering and Technology, Meerpet for their support and guidance throughout our project.

Finally, we express thanks to one and all who have helped us in completing this project report. Furthermore, we would like to thank my family and friends for their moral support and encouragement.

Submitted by:

Konam Pavan Kartheek	18K91A04A2
J. Manichandra	18K91A0478
Garlapati Rohith Reddy	18K91A0464

DECLARATION

We hereby declare that the major project report entitled "**AGRICULTURAL AID FOR TOMATO HARVESTING (AAT)**" is done under the esteemed guidance of **Dr. Girish Kumar** Associate Professor, Department of Electronics and Communication Engineering, TKR College of Engineering and Technology, is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering.

This is a record of bonafide work carried out by us at TKR College of Engineering and Technology and the results embodied in this project has not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree.

Submitted by:

Konam Pavan Kartheek	18K91A04A2
J. Manichandra	18K91A0478
Garlapati Rohith Reddy	18K91A0464

TKR COLLEGE OF ENGINEERING AND TECHNOLOGY
(Sponsored by TKR Educational Society, Approved by AICTE, Affiliated by JNTUH)
Autonomous, Accredited by NAAC with 'A' Grade. Accredited by NBA
Medbowli, Meerpet, Saroornagar, Hyderabad - 500 097
Phone: 9100377790, e-mail: info@tkrcet.ac.in website: www.tkrcet.ac.in

College Code : K9

PLAGIARISM REPORT

This is to certify that the Major Project entitled **"AGRICULTURAL AID FOR TOMATO HARVESTING (AAT)"**

Submitted

By

KONAM PAVAN KARTHEEK	**18K91A04A2**
J. MANICHANDRA	**18K91A0476**
GARLAPATI ROHITH REDDY	**18K91A0464**

Is Checked for Plagiarism and the similarity obtained is 10%.

Dr. M. GIRISH KUMAR	**Dr. P. GAYATRI**	**Dr. D. NAGESHWAR RAO**
Associate Professor	Associate Professor	Head of the Department
Co-Ordinator, Internal guide	Co-Ordinator	ECE

External Examiner:

DrillBit

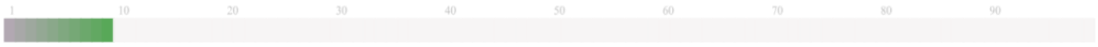
The Report is Generated by DrillBit Plagiarism Detection Software

Submission Information

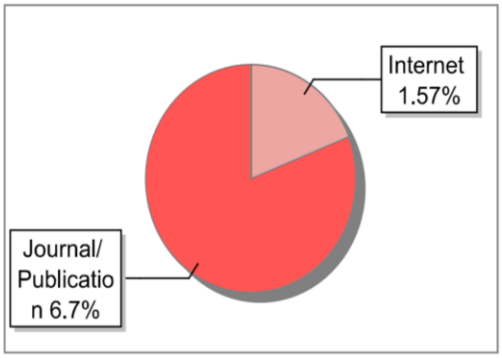
Author Name	BATCH 2 ECE-B
Title	AGRICUTURAL AID FOR TOMATO HARVESTING
Paper/Submission ID	533528
Submission Date	2022-05-28 16:04:25
Total Pages	72
Document type	Dissertation

Result Information

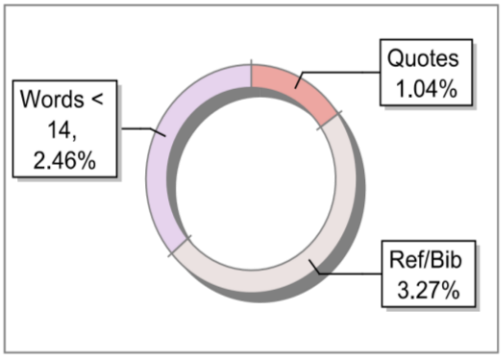
Similarity **10 %**



Sources Type



Report Content



Exclude Information

Quotes	Not Excluded
References/Bibliography	Not Excluded
Sources: Less than 14 Words Similarity	Not Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

A Unique QR Code use to View/Download/Share Pdf File

ABSTRACT

Fruit and vegetable harvesting robots have been widely studied and developed in recent years. However, commercial tomato harvesting robots remain a challenge despite extensive research due to the cost and difficulty. In this project, we propose an efficient tomato harvesting robot that combines the principle of 3D perception, Manipulation, and an end effector. For this robot, tomatoes are detected based on deep learning, after which 3D coordinates of the target crop are extracted and sent to the user. The user has a live feed of the crop and an analysis of the fruit of whether it is red or green and the user can specify the robot to harvest the fruit or not. In recent days, agricultural value has been decreased to a large extent, becoming a threat to existence. To overcome this problem digital farming is the best option for digital farming. Promoting digital farming might be the answer to current challenges in the agriculture industry by employing a robot to provide continuous information about its deployed area and gives the right analysis of many aspects of farming. Using this technology farming can be done as a game that can be controlled from anywhere. The robot consists of a four-wheel independent steering system, a 5-DOF harvesting system, a navigation system, and a binocular stereo vision system. The four-wheel independent steering system was capable of providing a low-speed steering control of the robot based on Ackerman steering geometry. The novelty of this project is that this method is simple, and image processing is kept simple to accommodate processors' limited computation resources.

DOMAIN: Harvesting Robot, Machine Learning.

TABLE OF CONTENTS

TABLE OF FIGURES

1. INTRODUCTION

1.1 INTRODUCTION

Today's world demands professionals that combine scientific and technological competencies with soft and social skills. The demand for agricultural goods throughout the world is increasing at an unprecedented rate.[1] In the next ten years, an estimated 50% increase in agricultural production will be required to feed and clothe the world's population. Promoting digital farming might be the solution to this challenge, such as the lack of interest in this industry, the ever-growing population that demands more food supplies, and the requirement in keeping quality and quantity checked all the time. Increased automation and robotization are critical in the agriculture business to accommodate rising demand and compensate for manpower shortages.

Automation of agriculture tasks has improved all phases of the industrial process from pre-harvest to post-harvest stages. Agricultural chores are also frequently physically hard and repetitious. Employing a robot is considered to be a digital farming project. The robot can give constant information about the farm or land where it is deployed and gives an accurate analysis of the aspects of farming. Furthermore, multi-annual crops, like apples, tomatoes, and grapes, need that the plant isn't broken throughout gathering. Selective harvesting means harvesting and regenerating crops which is an intensive and expensive task in the automation of the harvesting industry.

All the types of robots are possible to be farming robots, such as a drone that can spray pesticides and pick fruits from tall tress effectively. A robot can be effective during harvesting since it can monitor and pick the product at right time based on the input criteria. Arm robot manipulators and the mobile robot can be used as harvester robots. However, picking a robotic arm manipulator is perfect for the job since it can grab any objects. To increase the accuracy the robot needs an eye to detect, track, cut, and grab the product. Availability of the electronics is easier in the present world so making the robot has been simpler and more effective.

Development has brought a great change in technology, camera size is also getting smaller so that it can be mounted to the robot without adding significant weight to it. On Arm robots, the camera has mainly two types of applications, eye in hand and eye to hand. Eye in hand means attaching the camera to the robot, and eye to hand means

attaching the camera elsewhere. Both have advantages and disadvantages. Moreover, the eye in hand has a more natural function as an eye but it is prone to occlusion. The camera is utilized by image processing to differentiate which one to grab and cut and which ones to ignore.

1.2 LITERATURE SURVEY

In Japan, England, France, Italy, Israel, and other countries, a variety of tomato harvesting robots have been created. Kyoto University developed a tomato harvesting robot with a 5-degree of freedom (DOF) manipulator. Okayama University developed a 7-DOF robot consisting of a moving system, vision, end effector, manipulator, and a control system that detects both red and green tomatoes. Later new tomato harvesting robot consisting of a vision system and a rotating arm was developed. However, the time needed from recognition to pitching was about 15-20 s per tomato, with a success rate of 50%-70%. Moreover, the Institute of Agricultural Mechanization Science of Korea also developed a series of Tomato harvesting robots.

In China, there has been significant progress made in the research of tomato harvesting robots. Current robots employed are not intelligent enough and the success rate does not meet the expectations. The picking arms are generally replaced with industrial robotic arm manipulators which are of high cost and complexity to control. The time taken to pick fruit is around 30sec to estimate the fruit to harvest or not due to which the harvest rate is less than 90%. The fruits are identified mainly by color characteristics and geometrical shape. These factors are affected by light and environmental variables and are difficult to differentiate nonetheless, these issues are resolved using a tomato harvesting robot that is used in the greenhouse.

Tomato gathering robots will choose and pluck tomatoes of bound colors in the dark by employing an electric lamp. The robots will work night in pilotless greenhouses beneath these twinkling lights. These types of robots are being developed by Panasonic. Japan is working greatly in the field of harvesting robots and some of the Japanese scientists and robotics talks as below. Japanese farmer and scientist say, "*We began by mimicking individuals doing farm chores and harvesting. We utilized rings instead of blades and devised a way of plucking the tomatoes without touching them directly, much like we would on a farm.*" [2]Mr. Toshima says, "*Previously if a tomato was partially obscured by a leaf or a stem, the robot couldn't tell it was a tomato. The robot*

was able to distinguish tomatoes by memorizing photos that only featured bits of tomatoes." The answers to these issues are discussed in these projects.

1.3 MOTIVATION

The major cultivation states in India are Bihar, Karnataka, Uttar Pradesh, Orissa, Maharashtra, Telangana, Madhya Pradesh, Andhra Pradesh, and West Bengal for tomatoes. In the years 2018-2021 the cost of tomatoes has increased nearly to 10000 rupees for 1 ton and the usage has increased nearly above 20% of the regular usage. The estimated usage will be further increased and cultivation will be decreased in the next two years. The main factors in the reduction of cultivation are the high cost of harvesting and regular monitoring. A human can pick a tomato in 2 to 3 seconds whereas a robot can pick a tomato in 5 to 6 sec, but the human can work only for a limited time and the robot can work for about hours without a break. Using a robot to handle this problem has been attempted, however, the expensive cost of industrial arm manipulators has created difficulty for impoverished farmers, and manipulating complex mechanisms have made the task more difficult for ignorant farmers. Household agriculture has increased in the states of Karnataka and Bihar on a large scale. The major household cultivation plant is the tomato, but the regular monitoring and harvesting had created a disappointment in the result over the years 2015 to 2021.

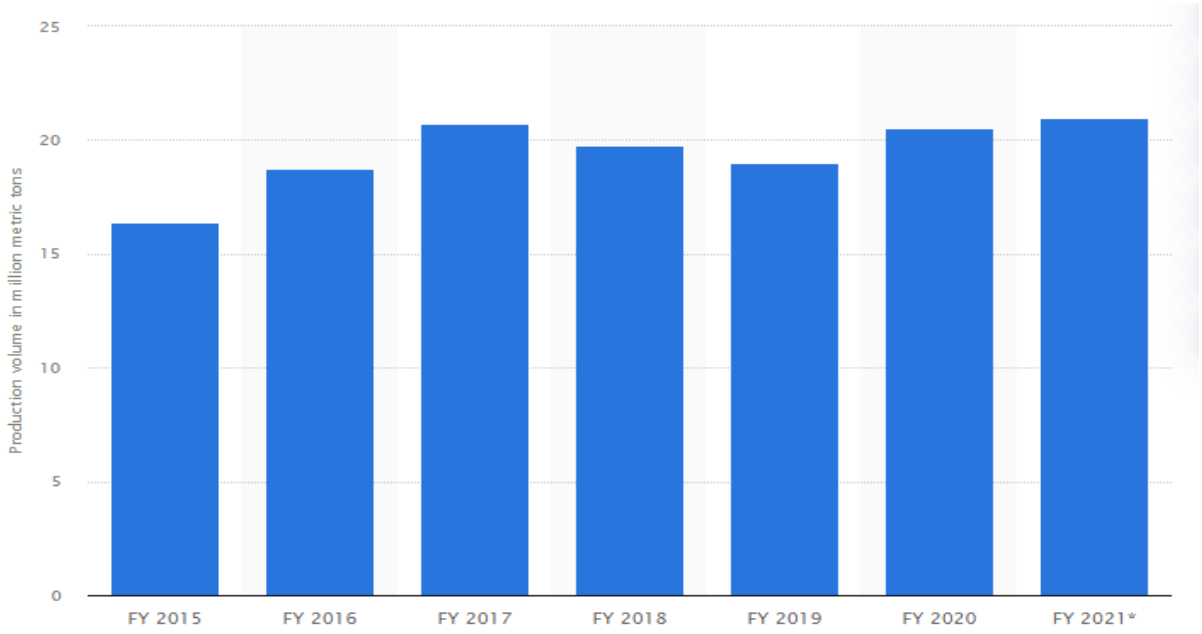


Figure 1 Tomato Production from the year 2015 to 2021

1.4 OBJECTIVE

In this project, we are going to create a harvesting robot for tomatoes at an affordable rate for poor farmers. This robot is designed mainly for household agriculture. The robot classifies the tomatoes into ripe and unripe based on the color configurations. The bot is employed with a robotic arm manipulator that helps in harvesting the fruit. The arm manipulator consists of 5- degrees of freedom, which enables the robot to move in different angles and directions, creating ease in harvesting. To allow the robot to move, we are wielding DC motors with circular treads for dexterity.

We are utilizing deep learning algorithms for tomato classification. We are using the Inception V3 algorithm for training the model used to classify the plant. For the robotic arm manipulator movement, we are using the servo motor available at ease and a low cost for easy replacement. We developed an interface to control the robot over a region through the internet and can have a live video stream of the crop. To make the work easier and fast we are using raspberry pi which behaves as the heart and brain of the system. The programming is carried out using python for greater comprehension and future growth.

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The present tomato harvesting robots can pick up the tomatoes in the range of 4 to 5 seconds and cover an area of around 4 to 5 acres of land in about 40 to 45 mins. The robot is capable of picking the fruit over different heights from 10cm to 100cm above the ground. But this robot costs around 3 to 4 lakhs and each robot is different from the other to handle. Any damage to the product inside the robot is difficult to repair or replace and a high sense of knowledge is required to handle the robot. The robot consumes about the power consumed by two air conditioners and not using the robot for a long time can create permanent damage to the robot. The availability of these robots is very scarce and manufacturing these robots is far more difficult.

2.2 DISADVANTAGES OF THE EXISTING SYSTEM

- Present robots are too costly.
- Difficult to maintain and handle.
- Not suitable for household agriculture.
- Power consumption is more.
- Available in limited regions.
- Repairs and replacements are not available.
- Manipulations cannot be made to the existing work style.

2.3 PROPOSED SYSTEM

In this project, we are going to design a robot with low cost and mutate into the user's need by a simple procedure. We are going to implement deep learning algorithms to classify the ripe and unripe tomatoes. We are using the 3D printed robotic arm manipulator parts for cost-cutting and to use them reliably. For the movement of the robot, we are going to use DC motors with alloy wheels for easy movement in sand and farming land. For better accuracy, we are using a digital cam that is readily available in the market. We are making use of the easily available equipment to make the robot user-friendly and easy for the user to replace the parts in case of damage. We are designing the robot in a way that can be mainly used in household agriculture. We maintained the control only to the user and there is no automation, this is to increase interest in the field of agriculture for the future generations. We are designing in a way

that agriculture is portrayed as a game for children and a healthy habit for the adults to increase the agriculture percentage with the growing population.

2.4 ADVANTAGES OF THE PROPOSED SYSTEM

- Cost Friendly.
- Easy to handle and maintain.
- Suitable for Household Agriculture.
- Less Power Consumption.
- Easy repairs and readily replaceable.
- Manipulations can be made easily.
- Can be controlled only by the user.

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

Functional requirements specify the services the system must provide, the behavior of the system to particular inputs, and the system behavior in particular situations. Functional requirements can also specify the activities not expected from the system. Functional requirements are captured in use cases.

Non Functional requirements specify the constraints on the services or functions offered by the system. These generally include timing constraints and also constraints on the development process and standards. These specify the quality attribute of the software. Non-Functional Requirements deal with issues like scalability, maintainability, performance, portability, security, reliability, and many more. Non-Functional Requirements address vital issues of quality for software systems. As functional requirements indicate what a system must do, non-functional requirements support them and determine how the system must perform. These features are usually expressed as constraints or criteria that define a level of freedom for developers or users.[3] These requirements apply to the complete system.

Domain requirements originate from the application domain of the system and indicate the characteristics and constraints of the domain. These can be functional or non-functional requirements.

For this project, the functional requirements are that the system must be able to analyze the captured image based on the trained dataset values. The non-functional requirements are that the system is constrained by the speed of the processor and the amount of memory available to store the data. Some other non-functional requirements for this project are reduced manpower, user-friendly, and improved performance.

3.2 SOFTWARE REQUIREMENTS

Any programming language that supports deep learning algorithms and has the designing interfaces to process images can be used. For designing the robotic arm manipulator parts, we used Fusion 360 and for training the database we used Google Colab. Although any software that supports the training of the model and 3D model design is desirable, depending on the system configuration. The programming language

chosen must simplify the task of implementing the various aspects of the application. As such, we identified the following software requirements.

- **Operating System:** Raspbian 64Bit
- **Programming Language:** Python 3.7
- **Tool Kits:** Python Standard Library, Numpy, Tensorflow, PIP, etc.

3.3 SOFTWARE TECHNOLOGIES

3.3.1 PYTHON

Python is presently the most extensively used multi-purpose, high-position programming language. Python allows programming in Object-Acquainted and Procedural paradigms. Python programs generally are lower than the other programming languages like Java, and C. Programmers have to class fairly lower and the indentation demand of the language makes them readable all the time. Python language is being used by almost all tech-giant companies like- Google, Microsoft, Amazon, Facebook, Instagram, etc... The firmness of Python is a large collection of standard libraries that can be used for different applications like Machine Learning, Web frameworks like Django, Web Scraping like Scrapy, Test Frameworks, Multimedia, and Image processing OpenCV.

Python downloads with an expansive library and it contain law for colorful purposes like regular expressions, unit-testing, threading databases, CGI, image manipulation, and more. Python can be extended to other languages as you can write your code in C or C# and can convert it into python or vice versa. Complimentary to extensibility, Python is embeddable as well.[4] The language's simplicity and expansive libraries render programmers more productive than languages like Java and C. Also, the fact that you need to write less and get further effects done. Since Python forms the base of new platforms like Raspberry Pi, it finds the unborn bright for the Internet of Effects. Because it isn't similar to a circumlocutory language, reading Python is more like reading English. This is the reason why it's so easy to learn, understand, and law. It also doesn't need curled braces to define blocks, and indentation is obligatory. This further aids the readability of the law. These are some advantages of Python.

Python is easy and simple but it has its downsides like Speed, Weak Browsers, Design Restrictions, etc. Python can raise run-time errors since it is dynamically typed. Speed of execution isn't a problem unless it is the focal point of your project.

3.3.2 DEEP LEARNING

Deep Learning or in short terms mentioned as deep literacy. It is a machine learning fashion that teaches computers to do what comes naturally to humans learn by illustration. Deep literacy is a crucial technology behind driverless buses, enabling them to fete a stop sign or to distinguish a rambler from a lamppost. It's the key to voice control in consumer bias like phones, tablets, TVs, and hands-free speakers. Deep literacy is getting lots of attention recently and for good reason. It’s achieving results that weren't possible before. A computer model learns to perform bracket tasks directly from images, textbooks, or sounds in deep literacy. Deep literacy models can achieve state-of-the-art delicacy, occasionally exceeding mortal- position performance. Models are trained using a large set of labeled or unlabelled data and neural network infrastructures containing numerous layers.

But the question arises here why it matters the most. In a word, delicacy. Deep literacy achieves recognition delicacy in advanced situations than ever ahead. This helps consumer electronics meet stoner prospects, and it's pivotal for safety-critical operations like driverless buses. Recent advances in deep literacy have better to the point where deep literacy outperforms humans in some tasks like classifying objects in images. While deep literacy was first theorized in the 1980s, there are two main reasons it has only lately come useful. Firstly deep literacy requires large quantities of labeled data. For illustration, driverless auto development requires millions of images and thousands of hours of videotape. Secondly, deep Literacy requires substantial calculating power. High-performance GPUs have a resemblant armature that's effective for deep literacy. When combined with clusters or pall computing, development brigades reduce a deep literacy network training time from weeks to hours or lower. Utmost deep literacy styles use neural network infrastructures, which is why deep literacy models are frequently appertained to as deep neural networks. The term “deep” generally refers to the number of retired layers in the neural network. Traditional neural networks only contain 2-3 retired layers, while deep networks can have as numerous as 150. Deep literacy models are trained by using large sets of labeled data and neural network infrastructures that learn features directly from the data without the need for homemade point birth.

One of the most popular deep neural networks is convolutional neural networks (CNN or ConvNet).[5] A CNN convolves learned features with input data, and uses 2D convolutional layers, making this armature well suited to recycling 2D data, similar to images. CNN excludes the need for homemade point birth, so you don't need to identify features used to classify images. The CNN works by rooting features directly from images. The applicable features aren't trained; they're learned while the network trains on a collection of images. This automated point birth makes deep literacy models largely accurate for computer vision tasks similar to object bracket. A machine learning workflow starts with applicable features being manually uprooted from images. The features are also used to produce a model that categorizes the objects in the image. With a deep literacy workflow, applicable features are automatically uprooted from images. In addition, deep literacy performs “end-to-end literacy” – where a network is given raw data and a task to perform, similar to bracket, and it learns how to do this automatically. Another crucial difference is deep literacy algorithms gauge with data, whereas shallow literacy converges. Shallow literacy refers to machine literacy styles that table at a certain position of performance when you add further exemplifications and training data to the network. A crucial advantage of deep learning networks is that they frequently continue to ameliorate as the size of your data increases. These are the main reasons that I used CNN deep learning networks to train my model.

3.3.3 FUSION 360

Fusion 360 is a pall-grounded 3D modeling, CAD, CAM, CAE, and PCB software platform for product design and manufacturing. Design and mastermind products to insure aesthetics, form, fit, and function. Reduce the impact of design, engineering, and PCB changes and ensure manufacturability with simulation and generative design tools.[6] Directly edit being features or model institutions with the only truly integrated CAD-CAM software tool. Fusion 360 enables you to design painlessly with flexible 3D CAD software. The electronic simulations for this are flawless 3D simulation software, produce 3D models and pretend in a single software tool with Fusion 360. We can incontinently turn your 3D models into 2D delineations with Fusion 360. Fusion 360 is simpler than Solidworks, but still an important software in its own right. It is easier to learn and grasp. Fusion 360 is recommended for functional prints as Autodesk rolls out due to regular updates. After all, they are both free for us potterers. Fusion 360 is created by Autodesk. Fusion 360 mainly focuses on 3D

modeling and manufacturing. Fusion 360 focuses more on accessible design and manufacturing of mechanical systems, making it a good choice for professional work but especially popular with individualities, including potterers and scholars. Fusion 360 also supports interpretation control as well as import and import of all major CAD train types. The Fusion 360 software's wide range of features gives us a protean tool that is popular with masterminds and professional contrivers and with hobbyhorse printers and start-ups. We used this software for the creation of a robotic arm manipulator design.

3.3.4 MODULES

Tensorflow

Tensorflow is an end-to-end open-source platform for machine literacy. It has a comprehensive, flexible ecosystem of tools, libraries, and community coffers that lets experimenters push the state-of-the-art in ML, and gives inventors the capability to fluently make and emplace ML-powered operations. TensorFlow provides a collection of workflows with intuitive, high-position APIs for both newcomers and experts to produce machine literacy models in multitudinous languages. Inventors have emplaced models on several platforms similar to waiters, in the pall, on mobile and edge bias, in cybersurfers, and on numerous other JavaScript or Python platforms. This enables inventors to go from model structure and training to deployment much more fluently. Tensorflow was developed by Google as an initiative for its research and production which was later converted into an open-source platform.

TensorFlow allows inventors to produce dataflow graphs — structures that describe how data moves through a graph, or a series of processing bumps. Each knot in the graph represents a fine operation, and each connection or edge between bumps is a multidimensional data array or tensor. Python is easy to learn and work with and provides accessible ways to express how high-position abstractions can be coupled together. Bumps and tensors in TensorFlow are Python objects, and TensorFlow operations are themselves Python operations. The factual calculation operations, still, aren't performed in Python. The libraries of metamorphoses that are available through TensorFlow are written as high-performance C binaries. Python just directs business between the pieces and provides high-position programming abstractions to hook them together. TensorFlow operations can be run on utmost any target that’s accessible an original machine, a cluster in the pall, iOS and Android bias, CPUs, or GPUs. However, you can run TensorFlow on Google’s custom TensorFlow Processing Unit (TPU)

silicon for further acceleration, if you use Google's pall. Performing, However, the performing models created by TensorFlow on utmost any device where will be used to serve prognostications. We trained the model using TensorFlow on Google Colaboratory.

Keras

Keras is one of the leading high-position neural network APIs. It's written in Python and supports multiple back-end neural network calculation engines. Keras was created to be stoner-friendly, modular, easy to extend, and to work with Python. The API was "designed for mortal beings, not machines," and "follows stylish practices for reducing cognitive cargo." Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can combine to produce new models. New modules are simple to add, as new classes and functions.[7] Models are defined in Python law, not separate model configuration lines.

The biggest reasons to use Keras stem from its guiding principles, primarily the bone about being stoner-friendly. Beyond ease of literacy and ease of model structure, Keras offers the advantages of broad relinquishment, support for a wide range of product deployment options, integration with at least five back-end machines (TensorFlow, CNTK, Theano, MXNet, and PlaidML), and strong support for multiple GPUs and distributed training. Plus, Keras is backed by Google, Microsoft, Amazon, Apple, Nvidia, Uber, and others. Keras models can be stationed across a vast range of platforms and maybe further than any other deep literacy frame. That includes iOS, via CoreML (supported by Apple); Android, via the TensorFlow Android runtime; in a cyber surfer, viaKeras.js and WebDNN; on Google Cloud, via TensorFlow-Serving; in a Python web app back end; on the JVM, via DL4J model import; and on Raspberry Pi.

NumPy

NumPy is a Python library that allows you to do scientific computing. It's a Python library that includes a multidimensional array object, colorful deduced objects (similar to masked arrays and matrices), and a variety of routines for performing fast array operations, such as fine, logical, shape manipulation, sorting, opting, I/O, separate Fourier transforms, introductory direct algebra, introductory statistical operations, arbitrary simulation, and more. The ndarray object is the nucleus of the NumPy

package. This wraps n-dimensional arrays of homogenous data types, with a variety of operations done in collected law for speed. The differences between NumPy arrays and ordinary Python sequences are significant. Unlike Python lists, which may expand indefinitely, NumPy arrays have a fixed size when created. Increasing or decreasing the size of a ndarray will produce a new array and cancel the original. The rudiments in a NumPy array are all needed to be of the same data type, and therefore will be the same size in memory.[8] The exception bone can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized rudiments. NumPy arrays grease advanced fine and other types of operations on large figures of data. Generally, similar operations are executed more efficiently and with lower law than is possible using Python's erected-in sequences.

A growing plethora of scientific and fine Python-grounded packages are using NumPy arrays; though these generally support Python-sequence input, they convert similar input to NumPy arrays previous to processing, and they frequently output NumPy arrays. In other words, to efficiently use important maybe indeed most of the moment's scientific/ fine Python-grounded software, just knowing how to use Python's erected-in sequence types is inadequate- bone also needs to know how to use NumPy arrays.

PIL

Python Imaging Library is the context that helps to perform image processing for the Python Interpreter. This provides a wide range of file support, efficient internal representation, and high image processing capabilities. This library is designed for fast access to the data stored in the image in the form of pixels format. It helps in creating, manipulating, and storing the images. This library supports a large number of image formats like jpg, jpeg, png, etc. The Imaging Library for Python was discontinued in 2011 and a new project is initiated with the name Pillow, a forked repository of PIL. In our project, the captured images from the webcam are processed using this library. For the dataset training, the images are performed through various operations and the result is viewed with the help of this library.

Operating System

The Operating System (full form of OS) library is used to perform several operations related to the system like creating, deleting, and accessing a file or folder in

the system. This library helps in the data gathering and various other formats of information present in the working system using python or any other programming language. This library is used to load the dataset trained weights and also to interlink the codes from one file to another like interlinking the training dataset and test dataset. OS library also helps to change the directory of the present working directory it also enables a view of the different files in the system using simple functions. We used this library to store the trained dataset in a preferred location and also to load it. We also used this library to provide access to the dataset images for training and testing purposes.

Open CV

Open Source Computer Vision Library is an open-source computer vision and machine learning software library.[9] This library was built to create an easy interface for the users to provide a platform for computer vision and other machine learning concepts which are interlinked to the camera. This helps in accelerating the use of machine perception. This is used by business persons to utilize and modify their code. This library simplifies the work of image processing. Using Open CV we can manipulate the images and videos it also helps to capture the images or videos which are defined as frames per image. We can even detect the handwritten text, and faces of the people, and animals, and identify the closet and internal items. In today's world, the use of Open CV has increased to a large extent and it is one of the leading libraries in the fields of Machine Learning, Artificial Intelligence, and other image processing platforms. It was initiated by Intel and later supported by multiple companies. This is a free source for both educational and commercial.

Open CV is supported by different APIs and multiple programming languages like C, C++, Python, and MATLAB (Matrix Laboratory). It also supports various operating systems like Windows, Mac OS, Linux, IOS, and Android. The main aim of this library is "to create real-time applications for computational efficiency". With consolidation with other libraries like NumPy, the images are processed in the form of an array. The images are stored in the form of vectors so that mathematical operations are performed undeniably. Latterly the Open CV has diverse applications like Facial recognition, Egomotion Estimation, Gesture recognition, Human-Computer Interaction, Motion Understanding, Object Detection, Mobile robotics, Stereopsis stereo vision, Structure from motion, Motion tracking, Augmented reality, automated

inspection, and surveillance, etc. We are using Open CV for the dataset image processing as it is free and other advantages like vacuity of numerous tutorials and comity with leading rendering languages.

Flask

Flask is a web framework that lets you develop web operations fluently. It is easy as it makes use of the microframework which doesn't include the Object Relational Director or similar features. Flask library is a wrapper around Werkzeug and Jinga. It has numerous features like URL routing, template machine, etc. It has similar features to Django but this is veritably Pythonic. It is designed to keep the operation of designing a web frame simple and scalable. Rather than abstraction subcaste for database support, flask supports multiple extensions to add similar capabilities to the operation. Using flask is easy due to its easy interaction with the support devices and it enables the user to create a display web page and access the webpage using simple and few lines of code. On top of that, it's veritably unequivocal, which increases readability. We used Flask to create a controller page to control the robot from a distant location and also we can view the live feed of the data on this platform where it made easier to create and control.

General Purpose Input Output (GPIO)

Raspberry-gpio-python or RPi.GPIO is the python library that helps to interface the GPIO pins on the Raspberry Pi. It buttresses referring to the GPIO pins either the physical PINs on the GPIO connector or using the Broadcom chip-specific PINs. For instance, pin 24 is BCM channel GPIO 8. Both the methods work fine but usage of the board number has slightly a higher advantage as the channel names may vary in the future. To use the Raspberry Pi we need to use the GPIO pins as they are the only ones that interact with the outer world and software. Using GPIO we can address the pins on and off depending on the usage. We can either give a high power or the Analog power contingent on the application. To control the motors and sensors we connect them to the GPIO pins and enable them on the required instance. We are making use of this library to control the actuators for the movement of the arm manipulator and robot.

Time

Time is the library or the module which provides various time-related operations. For different Operating Systems, the time function gives the different

format of time preceding the date. We can also convert the date-time format into our required format and perform operations on the date. Using this library we can set a timer for 10 minutes to run the code. We can also declare a period to run the section of code. The time function returns the date and time as a tuple of data. The time function is used in various coding languages like C, C#, Java, and Python. In Raspberry Pi, the time function is utilized in an additional method like delaying the work using simple functions like delay or sleep which helps the system to relieve the overheating of the device. We used this library to run the motors for a particular period and also used to store the time of the images captured.

Filetype

Filetype is a module in python which is a small and dependency package to infer file type, Multipurpose Internet Mail Extensions. This allows us to know the format of a file and verify whether it is in the required format or not. Filetype module has numerous features like it is simple, friendly API, supports a wide variety of file types, file discovery by kind or extension or MIME type, fast, can process large files, can pass a list of bytes, cross-platform file recognition, and protectorate free. We are using this module in our testing as well as training code to verify whether the captured images or stored images are of the required format or not. We can also check the trained model file extension as files may have the same names. It is important to check the file type as in a large database multiple files have the same names with different extensions and we need to load only the required format of data. This module also helps us to remove the unrequired file type by identifying them. Using this library we can also load the images from the Internet using URL in the required format.

Matplotlib

Matplotlib is an extensive library of NumPy. This library is versatile. It is supported in different machines like MATLAB, and Python. This is a data visualization tool and graphical plotting tool. A few lines of code are sufficient to generate a data visual plot. Matplotlib consists of two API overlays, pyplot API and object-oriented API. The Object-Oriented API is more customizable and powerful but difficult to use than pyplot. This library was built to replace MATLAB. When we create a plot using this library it creates a user interface and menu structure. We used Matplotlib to plot the various values like accuracy, loss, pre-trained loss, pre-trained accuracy, and bias, over the trained dataset to observe the result legibly.

4. HARDWARE REQUIREMENT SPECIFICATIONS

4.1 HARDWARE REQUIREMENTS

The hardware parts specifications for the movement of the robot may vary with the requirement of the user. However, the application requires basic hardware support for execution on a single test system. The memory requirements depend on the actual database and its distributional features. The disk space and CPU speed are also factors that determine the response time of the application.

For test and simulation purposes of the software, we identified the following hardware requirements:

- **Processor**: Quad-core Cortex-A72
- **Speed**: 1.4 GHz
- **RAM**: 1 GB (min)
- **Hard Disk**: 16 GB

4.2 MACHINERY

Raspberry Pi

Raspberry Pi is the device that acts as the brain for this project. It is a pocket-size computer that is entrapment into a computer examiner or Television and uses a standard keyboard and mouse. Raspberry Pi was launched by Raspberry Pi Foundation. It's an able little device that enables people of all periods to explore computing and to learn how to program in languages like Scratch and Python. It’s able of doing everything you’d anticipate a desktop computer to do, from browsing the internet and playing high-description videotapes, to making spreadsheets, word-processing, and playing games. It can interact with the outside world and has been used in a wide array of digital maker systems, from music machines and parent sensors to rainfall stations and twittering birdhouses with infra-red cameras. There were many releases worldwide and from the year 2012 to 2022 there were a total of 13 releases and each release has an update in one or the other manner. Compared to the computer raspberry pi have very minor differences as raspberry pi does not include a hard disk but has an SD card slot that acts as a hard disk for the device. Each variant has different pins and slots for the cables like VGA, HDMI, USB, etc.

To interact with the outer world the raspberry pi has 40 pins of which 26 are General Purpose Input Output Pins. These pins enable the sensors to on and off according to the user-defined functions. Raspberry Pi is highly recommended as it is low-cost, compatible, and attainable. We can use the Raspberry Pi according to our needs and mostly the proper selection of devices is mandatory. We selected the model Raspberry Pi 4 for our project as it is easy to handle and satisfies our requirements. We employed the 8GB RAM Raspberry Pi device for faster communication and interaction. We exercised 14 GPIO pins and one USB slot for connecting the motors and the camera. We have installed the Raspbian 64Bit software as it is pre-employed with python software and have the ability to rein in the Tensorflow latest version.

DC Motors

A DC motor is any motor within a class of electrical machines whereby direct current electrical power is converted into mechanical power. Most frequently, this type of motor relies on forces that glamorous fields produce. Anyhow of the type, DC motors have some kind of internal medium, which is electronic or electromechanical. In both cases, the direction of current inflow in part of the motor is changed periodically. The speed of a DC motor is controlled using a variable force voltage or by changing the strength of the current within its field windings. While lower DC motors are generally used in the timber of appliances, tools, toys, and machine mechanisms, similar to electric auto seats, larger DC motors are used in hoists, elevators, and electric vehicles.

A 12v DC motor is small and affordable, yet important enough to be used for numerous applications. One specific of a 12v DC motor is the operating voltage. When a motor is powered by batteries, low operating voltages are generally preferred since smaller cells are needed to gain the specified voltage. Still, at advanced voltages, electronics to drive a motor are generally more effective. Although the operation is possible with volts as low as 1.5 that go up to 100, the most common are the 6v DC motor, 12v DC motor, and 24v DC motor. We are making use of four 12v DC motors for the movement of the robot from one location to another and it can move in four directions left, right, forward, backward, and stop for switching off the motors.

Servo Motors

Servo motors are motors that can rotate through steps using a servo mechanism. It is a linear or rotary actuator that allows for precise control of linear or angular

positions, velocity, and acceleration. It is usually controlled for a specific angular rotation using a typical closed-loop feedback control system. Inside a servo motor, the motor is coupled to a sensor for positional feedback to know the angle of movement. There are various types of Servo motors for different applications. They vary with the rotational speed and also the amount of weight they can bear. The main reason to use a servo motor is that we can define the amount of angular rotation, unlike the DC motor. In this project, we used two types of servo motors namely Mg90S and Mg996 for the robotic arm movement.

The MG996R is an upgraded interpretation of the notorious MG995 servo and features upgraded shock-proofing and a redesigned PCB and IC control system that make it much more accurate than its precursor. It can rotate roughly 120 degrees, 60 degrees in each direction. It has metal gear due to which durability increases. It has a torque of around 11kg/cm. We can also use the Mg995 but we used three Mg996 (Metal Gear) servo motors for the bottom hand, base, and hand joint of the robotic arm.

Mg90S is a replacement for the SG90 servo motor. Both the servo motors have similar workstyle but Mg90S has an upper hand due to the metal gear which makes them durable. These motors operate on the voltage value of 4.8V to 6V and stall torque of 1.8Kg/cm. It can rotate from about 0 degrees to 180 degrees. We used this motor as it is less weight and small in size to fit in the robotic arm. We used three Mg90S motors for the upper hand, jaw, and pincers of the robotic arm.

HD Camera (WebCam)

A webcam is a device that can capture images and videos. The Raspberry Pi model has an internal slot for the camera which supports the Raspberry Pi Cam but the resolution of the camera is too low. To get higher pixel clarity for image processing we used an HD cam with a USB connector through which it is connected to the Raspberry Pi. In general terminology, a webcam is a microdevice for capturing images wired or wireless. The features of the webcam are mostly dependent on the software operating system. The characteristics of a webcam include cost and resolution. To meet the requirements of the device for capturing images we utilized TVS WC 103 cam.

TVS WC 103 cam has multiple features which highly satisfy the need for image capturing properties. It can capture video at 30FPS (Frame per second) and has a resolution of 1080 pixels. It also has a field of view of 60 degrees. The length of the

cable made it easier to connect the device and the LEDs made it easy to understand the working of the camera. Any other camera which can capture the images of the required resolution and low cost can replace this device.

Motor Driver

Motor drivers are devices that can control the motor. A DC motor needs high voltage that a microprocessor cannot produce so we use an IC for the motor which is called a Motor driver IC. When a microprocessor sends a signal the motor driver converts the signal and sends it to the motor. The motor driver forms an H-bridge to control the motor based on the signal received. There is a legion of motor drivers and each has its specifications. An L293D motor driver is used to drive at 4.5 V and the motors of high voltage cannot be used for this motor driver. An L293D motor driver IC can interact with 4 motors at a time. Though the L293D motor driver has numerous advantages it cannot run the motors above 500 RPM. To overcome this problem we used two L298N motor drivers to control 4 motors.

L298N has a similar working style as L293D but it has some extra features like it can control the direction along with the speed. It has an extra advantage in that it creates a dual H-bridge to connect. There are various other devices imposed on the motor driver to avoid heat damage like a resistor, capacitors, voltage regulators, and Power LED. There is an extra advantage that when the input supply is less than 12V the internal circuitry will be powered by the 78M05 voltage regulator. This motor driver can run the motors of 1000RPM and it is cost-friendly. It also has the ability to temperature protection.

Robotic Arm

The parts of the robotic arm are designed using Fusion 360 according to our needs. The robotic arm is designed in a way that the motor holding slots are perfectly designed for the motor that we are using. For printing the robotic arm we used plastic material due to availability and low cost. The printing is not performed on the complete arm at a time but the arm was subdivided into parts and printed individually so that any damage on any part can be easily replaceable. The printed parts are combined using the screws provided by the motors while bought and the slots were given in proper design for the screws.

The 5 Degrees of freedom designed robotic arm can move at 5 different angles. The arm is simulated on the axis of design and made remarkably easy for any user to design the parts. The parts of the robotic arm are classified into Circular Base, Gear Arm, Gripper Base, Grip, Gear Arm 2, Rotating Waste, and three types of arms. The slots of the Circular base, Rotating waste are bigger than the other servo motor slots. The arranged model of the 5 Degree of Freedom robotic arm looks similar to the below picture.

Wheels

A wheel is an indirect element that's intended to rotate on an axle bearing. The wheel is one of the crucial factors of the wheel and axle which is one of the six simple machines. Wheels, in confluence with axles, allow heavy objects to be moved fluently easing movement or transportation while supporting a cargo, or performing labor in machines. Wheels are also used for other purposes, similar to a boat's wheel, steering wheel, potter's wheel, and flywheel.[10] Common exemplifications are plants in transport operations. A wheel greatly reduces disunion by easing stir by rolling together with the use of axles. For the bus to rotate, a moment needs to be applied to the wheel about its axis, either by way of graveness or by the operation of another external force or necklace. Using the wheel, Sumerians constructed a gimmick that spins complexion as a potter shapes it into the asked object.

For the accurate movement of the robot, we used the plastic empowered wheels aligned with the rubber layer of threading around the wheels for better movement in the fields. It will be more effective when we use standard wheels or the spike wheels which can even move in rough sand but to empower the wheels with the movement on land and water is not quite possible. The wheels get affected easily so proper maintenance is mandatory. While running the robot on wetland we need to make sure the sand is not stuck between the wheels which may damage the run system.

Power Supply

Power sources are the key items to running a device. We cannot use the AC power supply as the robot moves from one location to another. We are using two different power sources for the robot one is the 12V DC power supply and the other is a power bank. The power bank powers the Raspberry Pi whereas the 12V power supply powers the motors. These items are used as they are easily available. Power banks are

commonplace and with our adding use of battery-powered outfits. They're effectively a movable bowl. All they need is a USB charging interface. Power banks come in a variety of shapes and sizes and suit numerous different people and their needs in recent times, the use of power banks has risen significantly as they give a veritably accessible and easy system of charging smartphones and other biases when down from mains power. The name power bank can be likened to a fiscal bank where finances can be deposited, stored, and withdrawn when demanded.

12V power inventories are one of the most common power inventories in use moment. In general, a 12VDC affair is attained from a 120VAC or 240VAC input using a combination of mills, diodes, and transistors. 12V power inventories can be of two types 12V regulated power inventories and 12V limited power inventories.12 V regulated power inventories come in three styles Switching regulated AC to DC, Linear regulated AC to DC and Switching regulated DC to DC. Switching regulated 12VDC power inventories, occasionally appertained to as SMPS power inventories, switchers, or switched-mode power inventories, regulate the 12VDC affair voltage using a complex high frequency switching fashion that employs palpitation range modulation and feedback. Linear-regulated 12VDC power inventories regulate the affair using a dissipative regulating circuit. They're extremely stable, have veritably low ripple, and have no switching frequentness to produce EMI.

5. SYSTEM DESIGN

5.1 SOFTWARE ARCHITECTURE

We trained the software using different algorithms of which the Inception V3 which is a part of the Convolutional Neural Network had given the highest accuracy. We collected the data of various tomatoes and classified them into two classes Plant and Unripe. They are pre-processed in which the images with noise are filtered and the later the features of the plant and unripe are extracted and the extracted features are processed through the algorithm and a model is created and deployed over the weights. These features are stored as weights which the trained model over our system. Later the trained model is used to detect the feature whether it is Plant or Unripe.

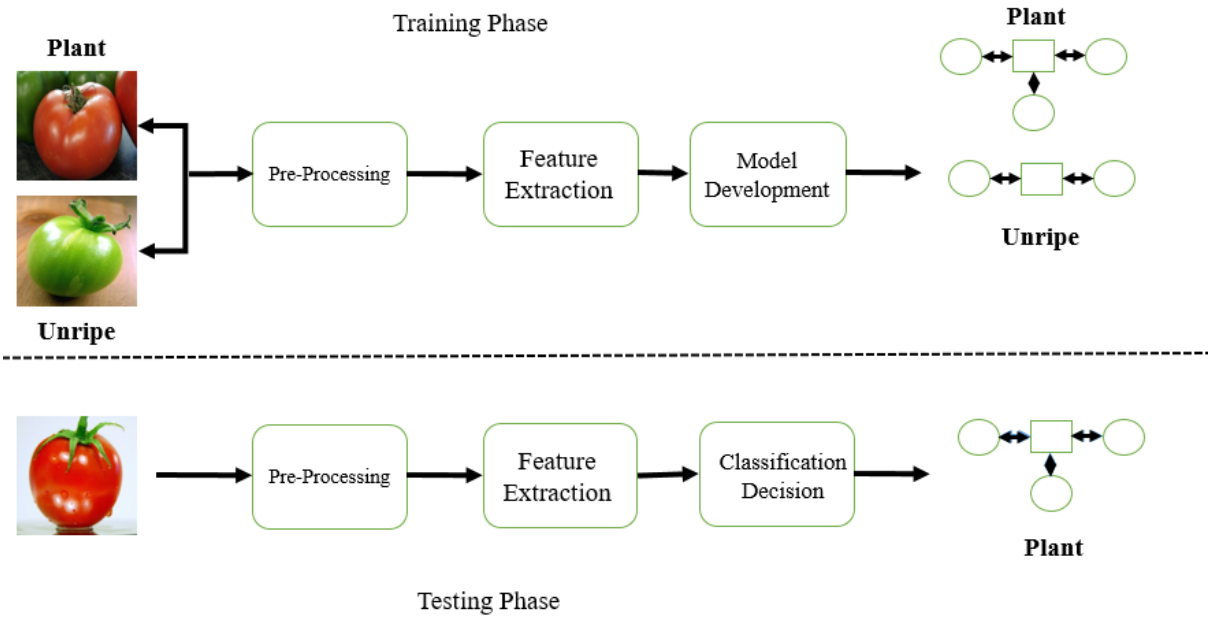


Figure 2 Software Architecture

5.2 HARDWARE ARCHITECTURE

All the motors and motor drivers along with the camera are attached to the brain of the system which is Raspberry Pi. The Raspberry Pi creates a web terminal through which the user can connect and control the robot. The user-defined functions or operations have been processed, and the Raspberry Pi then powers the motors in accordance with the user-defined operation. The User will have a live feed through the camera where the image is always processed from the raspberry pi from frame to frame with an FPS (Frames per second) of 10. The user and the raspberry pi should be on the

same network for the connection to establish and double-check the internet speed is high to avoid network losses and slow movement or damage to the robot. The DC motors are connected to the motor driver which activates the motor clockwise or anti-clockwise according to the user-defined functions.

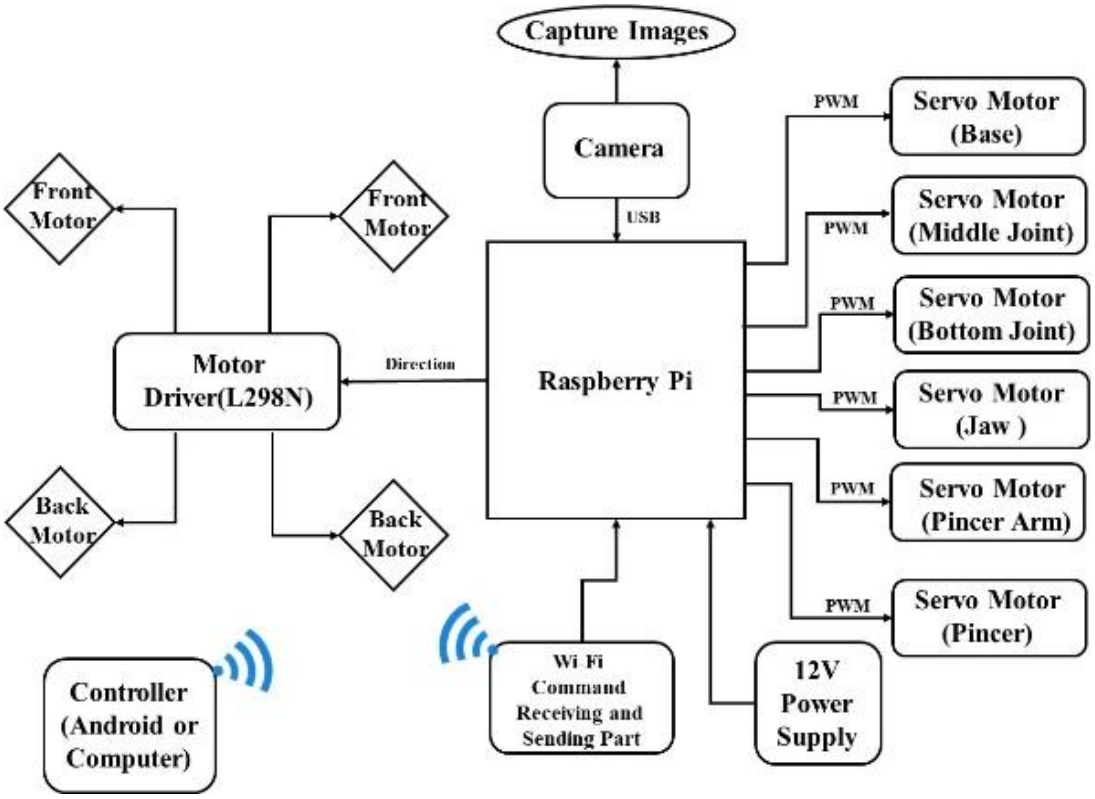


Figure 3 Hardware Architecture

5.3 MODULE DESCRIPTION

The model we trained is on over Convolutional Neural Network Algorithm of which we used Inception V3. The model has been trained over the dataset and classified. We have collected a total of around 500 images of each class and pre-processed each image into an array of required formats where noise is been eliminated. The Inception V3 is the updated model of Inception V2 which consists of several layers of convolution. The several layers of convolution make sure the image is processed properly and noise is been eliminated even for a small level. We converted the processed images into RGB and extracted the color levels and made the model train over the values on the classes the model is trained and saved which can be used as a trained model for the testing process. A better understanding of the model can be

understood by the description of the algorithm used. A bit of clarity can be found in the Unified Modeling Language diagrams (UML) below.

UML is an ultramodern approach to modeling and establishing software. It’s one of the most popular business process modeling ways. It's grounded on diagrammatic representations of software factors. As the adage says “a picture is worth a thousand words”. By using visual representations, we are suitable to more understand possible excrescencies or crimes in software or business processes. The software mastermind can represent an analytical model using the modeling memorandum, which is regulated by a set of syntactic-semantic and realistic criteria. Any complex system is best understood by making some kind of plates or filmland. These plates have a better impact on our understanding. However, we will if we look around, we realize that the plates aren't a new conception but are used extensively in different forms in different diligence. We prepare UML plates to understand the system in a better and simple way. A single illustration isn't enough to take care of all parts of the system. UML defines colorful kinds of plates to cover the utmost aspects of a system.

5.4 STATE CHART DIAGRAM

Statechart diagrams are used to model the dynamic nature of a system.[11] It specifies several states of an item across time, and these states are altered by events. It illustrates how control flows from one state to the next. In the below-manifested diagram, we declared the states over which the robot process through. For better and easy understanding, we divided the states over which each part of the robot process into 3 divisions. Where the top division deals with the robot moving from one place to other. The second section is about the capture state and the states it travels through on its way to the terminus. The third section is on arm movement and how it is processed.

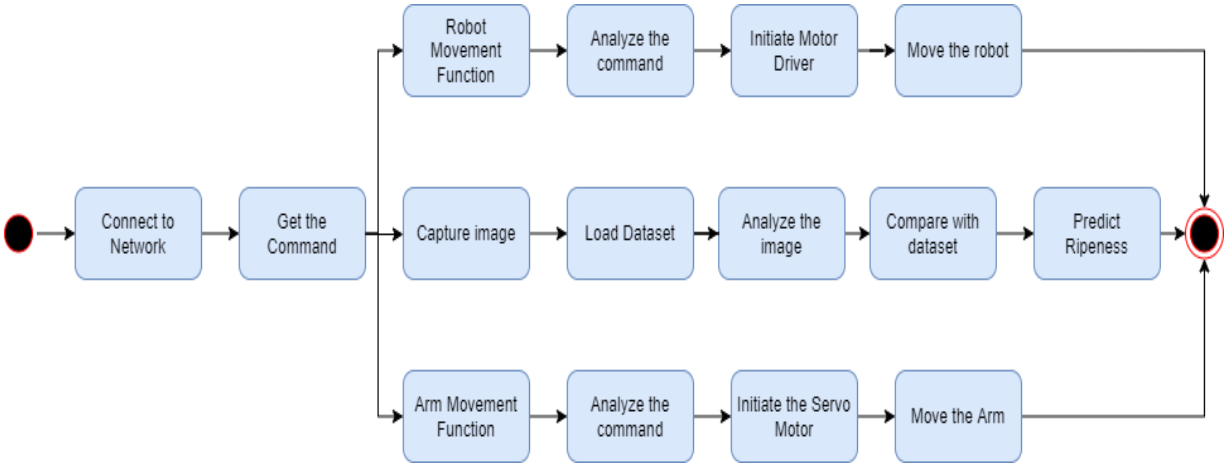


Figure 4 State Chart Diagram

5.5 SEQUENCE DIAGRAM

A sequence diagram in the Unified Modeling Language (UML) is a type of business diagram that describes how processes interact and in what order. It's a Communication Sequence Map concept. Sequence plates are occasionally called event plates, event scripts, and timing plates. We classified the sequences of operations into two different types where one deals with the sequence of capture function operations and the other deals with the robot and arm movement operation sequences.

Figure 5 is the sequence of operations over which the capture function is processed and the description of the sequence of operations is mentioned in the diagram. When the user calls the Capture function from the web page the below sequences of operations are performed. Starting from capturing the image to the detect ripeness all the operations are sequentially and these operations are not visible to the user he can only view the result.

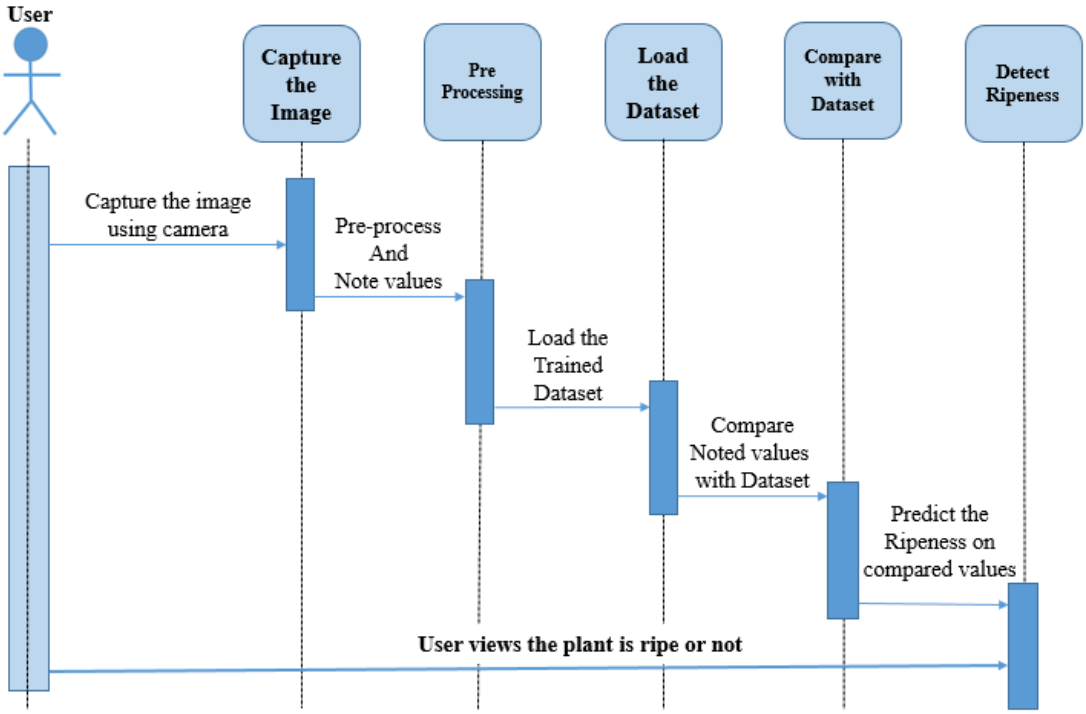


Figure 5 Sequence Diagram of Capture

Figure 6 is the sequential representation of the movement of the motors in the robot. It deals with the movement of the robot and arm manipulator. When the user calls a function that is related to the movement of the arm or robot the user can only view the movement but for successful movement, the below sequence of operations is need to be processed flawlessly. We declared a function for each operation given to the user. The system checks the data whether the user called the function or it is an error and analyzes it and later initializes and runs the motors.

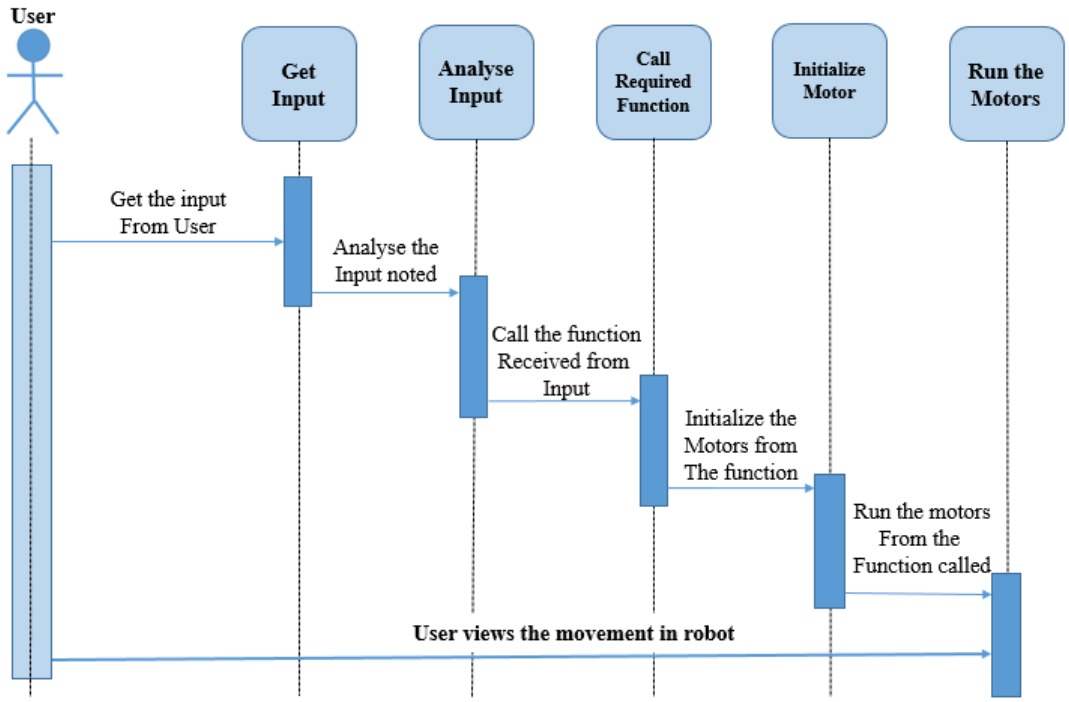


Figure 6 Sequence Diagram of Motors

5.6 USE CASE DIAGRAM

A use case illustration is a sort of behavioral illustration specified by and derived from a Use-case analysis in the Unified Modelling Language (UML). Its purpose is to present a graphical overview of the functionality handed by a system in terms of actors, their pretensions (represented as use cases), and any dependencies between those use cases. A use case illustration's principal aim is to explain which system functions are executed for which actor. The locations of the system's actors can be shown. There are simply four different cases in the robot function and they are mentioned in the below diagram. Each use case has its operations which are discussed before and the operations are performed according to it. The user can only view the below four cases and handle the operations of the four cases Robot Movement, Capture Image, Detecting Ripe and Unripe, and ARM movement.

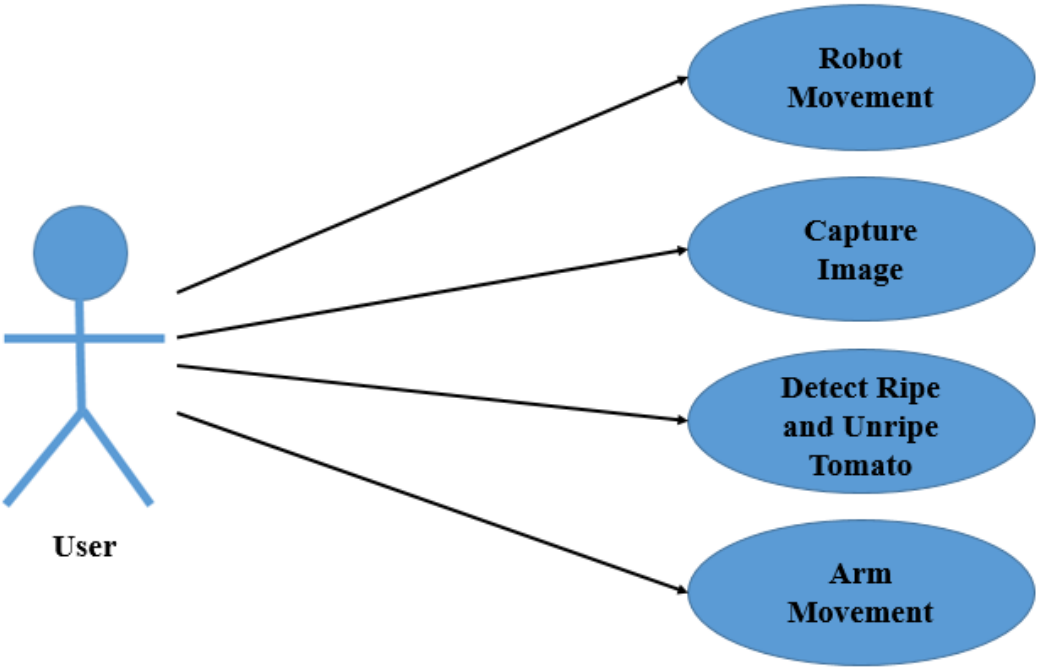


Figure 7 Use Case Diagram

5.7 PROCESS FLOW DIAGRAM

A Process Flow Diagram (PFD) is a type of flowchart that illustrates the connections between major factors at an artificial factory. It's most frequently used in chemical engineering and process engineering, though its generalities are occasionally applied to other processes as well. It’s used to validate a process, ameliorate a process, or model a new bone. Depending on its use and content, it may also be called a Process Flow Chart, Flowsheet, Block Flow Diagram, Schematic Flow Diagram, Macro Flowchart, Top-down Flowchart, Pipeline, and Instrument Diagram, System Flow Diagram, or System Diagram. They use a series of symbols and memos to depict a process. The symbols vary in different places, and the plates may range from simple, hand-drawn scribbles or sticky notes to professional-looking plates with expandable detail, produced with software. For a clear understanding and easy representation, we divided the process flow diagram into two further parts where those process flow diagrams are declared as the functions in the main process flow diagram. The capture function is explained in the main flow diagram itself.

In the below process flow diagram the Robot movement and Arm movement are the functions where each function has its process flow diagram which is further described. The main point of the process is the establishment of the connection between the user and the robot which is defined as the IP connection. The Stop function in the flow diagram expresses the stop for all the motors but not the live feed. The stop function is one way similar to the end as it is the final step of the process.

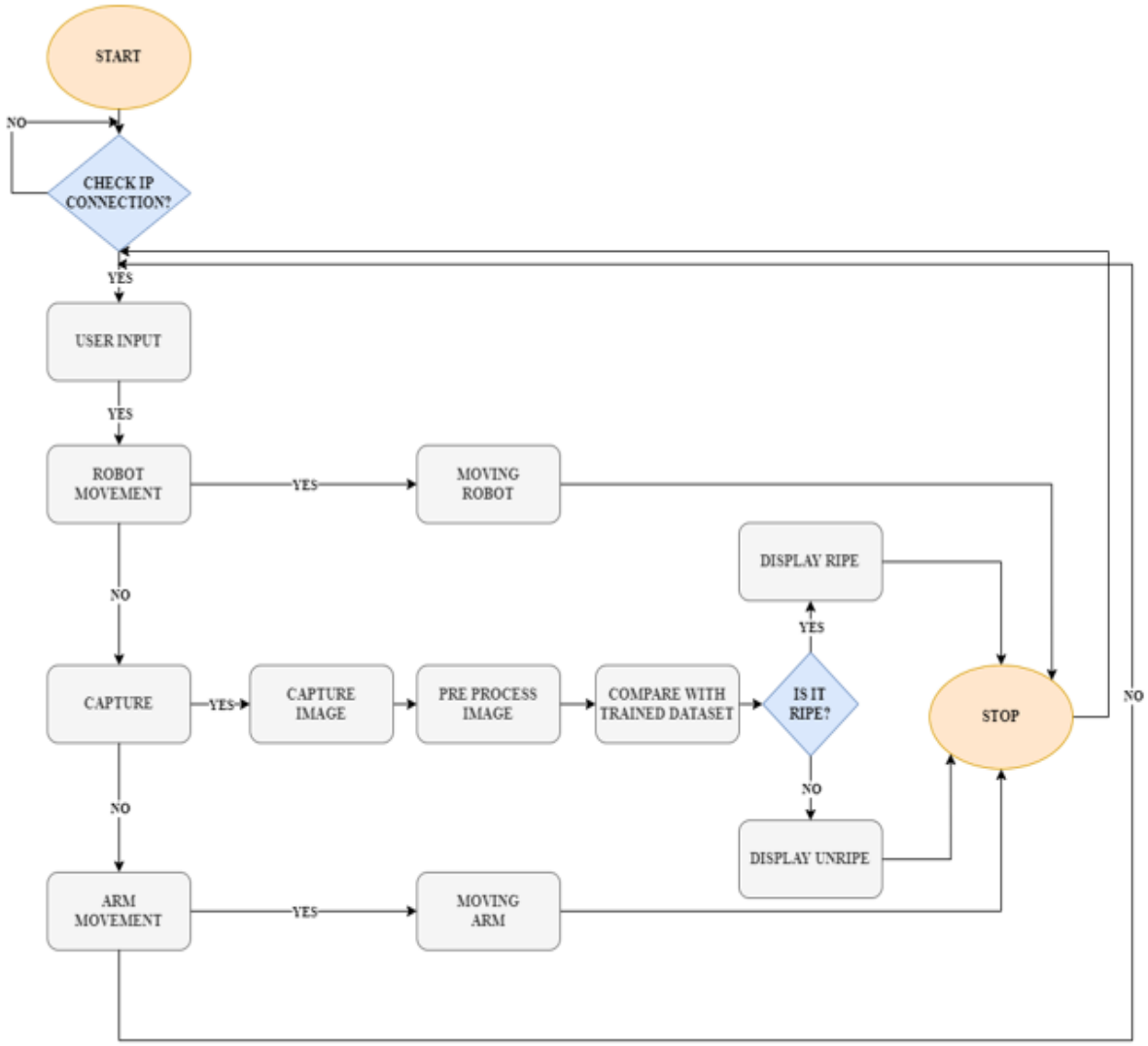


Figure 8 Process Flow Diagram of Robot

The moving robot deals with the DC motors and the below diagram describes the process flow of the robot movement which depends on DC motors. All the motors are connected to the motor driver and the functions called are processed in Raspberry Pi and those functions are sent to the motor driver as a command to on or off depending on the function called. This process gives a clear explanation of the moving robot function declared in the main process flow diagram.

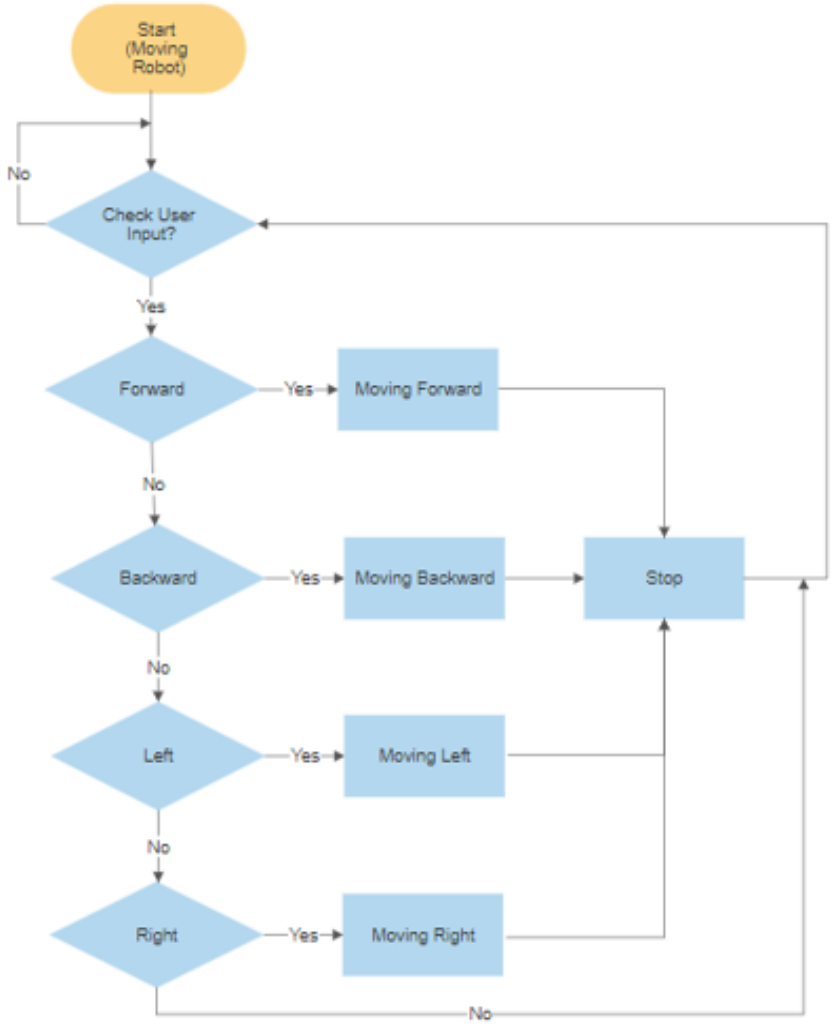


Figure 9 Process Flow Diagram of Moving Robot

The below-described process flow diagram is the clear process of the moving arm function declared in the robot process flow diagram. There are six servo motors where each has its priority in the project and each motor has two different operations one is to increase the angle and the other to decrease. When the increasing function or decreasing function is called the servo angle changes with a difference of 2 degrees to ensure that even a small movement can be made and observed.

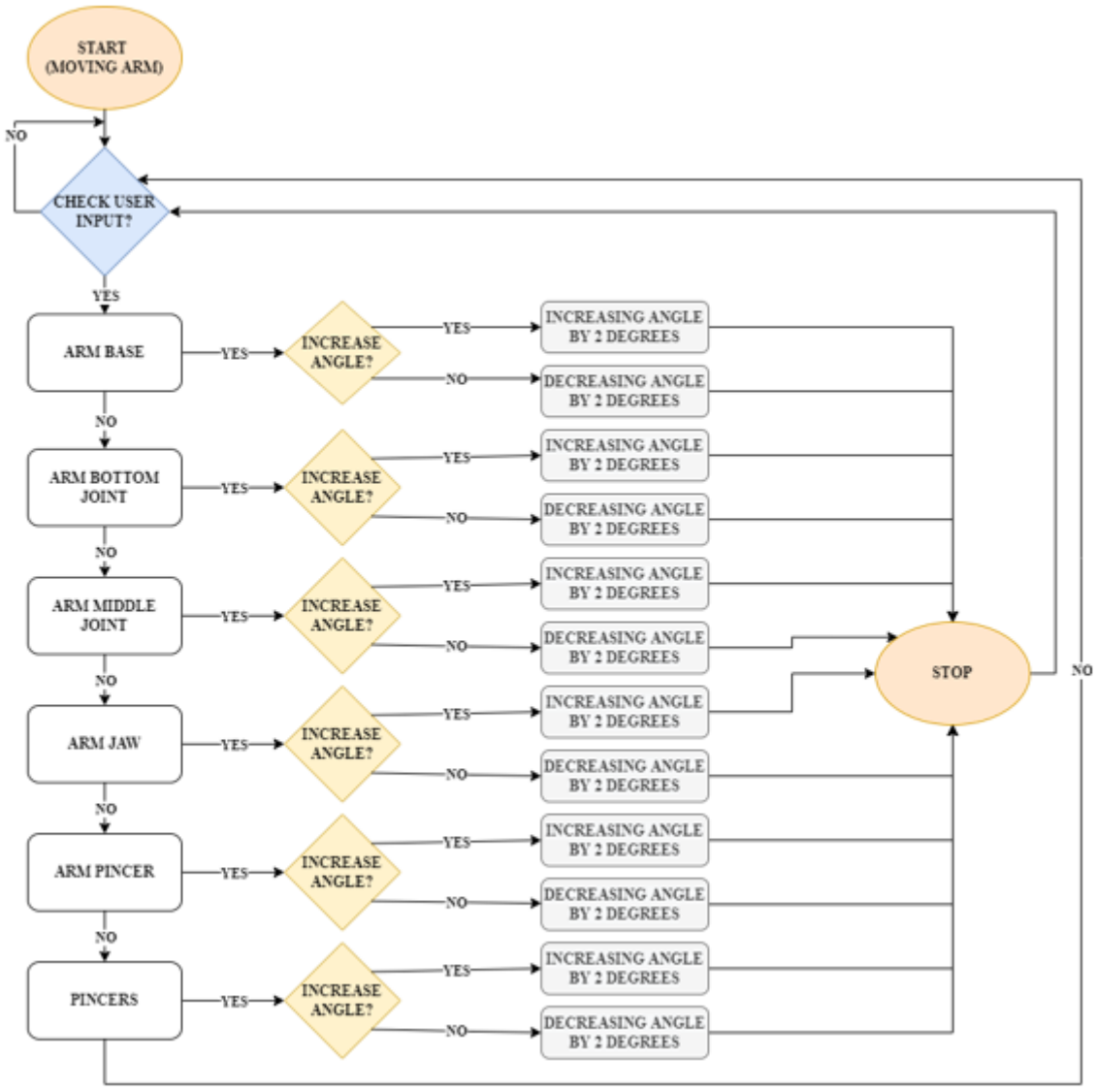


Figure 10 Process Flow Diagram of Moving Arm

6. IMPLEMENTATION

6.1 ALGORITHMS USED

INCEPTION V3

Largely performant deep neural networks need to be large. For a neural network to be considered large, it had to have several further layers within the network and units within these layers. Convolutional neural networks profit from rooting features at varying scales. The natural mortal visual cortex functions by relating patterns at different scales, which accumulates to form larger comprehensions of objects. Thus multi-scale convnet has the eventuality to learn further. Large networks are prone to overfitting, and chaining multiple convolutional operations together increases the computational cost of the network.[12] To make the Inception network and module a reality, the experimenters designed intuitive convnet infrastructures. When multiple deep layers of convolutions were used in a model it redounded in the overfitting of the data. To avoid this from passing the commencement V1 model uses the idea of using multiple pollutants of different sizes in the same position. Therefore, in the commencement models, rather than having deep layers, we've resemblant layers, making our model wider rather than deeper. The Inception model is made up of multiple inception modules. The introductory module of the Inception V1 model is made up of four resemblant layers.1 × 1 convolution, 3 × 3 convolutions, 5 × 5 convolutions, and 3 × 3 maximum pooling. The process of transubstantiating an image by applying a kernel over each pixel and its original neighbors across the entire image is called convolution. Pooling is the process used to reduce the confines of the point chart. There are different types of pooling but the most common bones are uttermost pooling and average pooling. This model is called as Naïve form.

One of the downsides of this naive form is that indeed the 5 × 5 convolutional subcaste is computationally enough precious i.e. time- consuming and requires high computational power. To overcome this the authors added a 1 × 1 convolutional subcaste before each convolutional subcaste, which results in reduced confines of the network and brisk calculations. The Inception V3 is just the advanced and optimized interpretation of the Inception V1 model. The Inception V3 model used several ways for optimizing the network for better model adaption. It has advanced effectiveness. It has a deeper network compared to the Inception V1 and V2 models, but its speed is not

compromised. It's computationally less precious. It uses supplementary Classifiers as regularizes. The Inception v3 model was released in the time 2015, it has an aggregate of 42 layers and a lower error rate than its forerunners. The major variations done on the Inception V3 model are Factorization into Lower Complications, Spatial Factorization into Asymmetric Complications, Mileage of Auxiliary Classifiers, and Effective Grid Size Reduction.

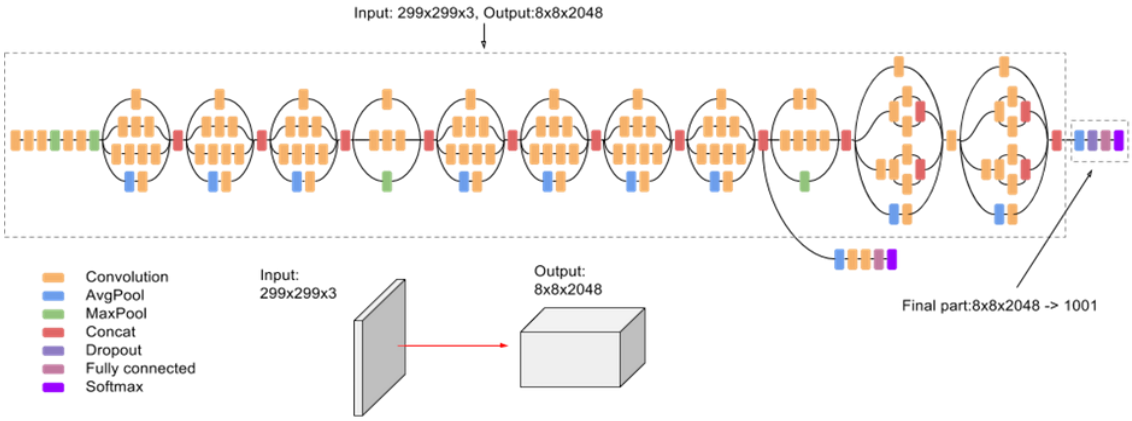


Figure 11 Inception V3 Architecture

The Inception V3 model architecture is based on the above factors and each image is classified over the above levels. Factorized Convolutions help to reduce the computational effectiveness as it reduces the number of parameters involved in a network. It also keeps a check on the network's effectiveness. Lower convolutions replacing bigger convolutions with lower convolutions surely leads to briskly training. Say a 5 × 5 sludge has 25 parameters; two 3 × 3 pollutants replacing a 5 × 5 convolution has only 18 (3 * 3 3 * 3) parameters rather. A supplementary classifier is a small CNN fitted between layers during training, and the loss incurred is added to the main network loss. In GoogLeNet supplementary classifiers were used for a deeper network, whereas in Inception v3 a supplementary classifier acts as a regularizer. Grid size reductions are performed by max pooling. The above operations are performed on the collected dataset and the accuracy and loss of the model are churned out below in figures 12 and 13.

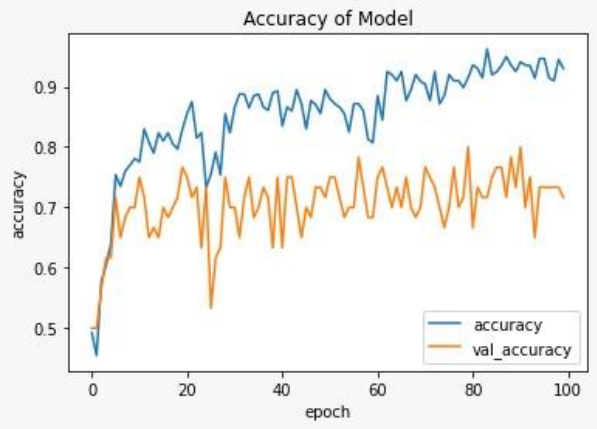


Figure 12 Accuracy of the Trained Model

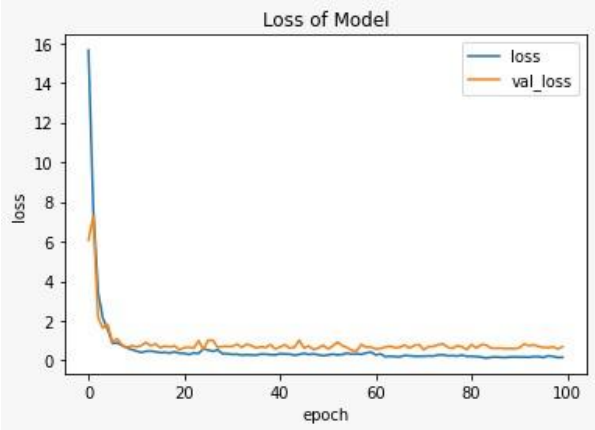


Figure 13 Loss of the Trained Model

6.2 ENCIPHER

6.2.1 TRAINING

IMAGE GENERATION AND AUGMENTATION

For training the model for the detection of ripe and unripe tomatoes we collected 500 images. The collected images are augmented using the function image_gen_w_aug. In this function, the images are rescaled and classified which are later processed for training. This function returns three values regarding training, validation, and testing. In this process, the images are rescaled and the required image is generated.

```
def image_gen_w_aug(train_parent_directory):
    train_datagen = ImageDataGenerator(rescale=1/255,
                                       rotation_range = 30,
```

```
                    zoom_range = 0.2,
                    width_shift_range=0.1,
                    height_shift_range=0.1,
                    validation_split = 0.15)
    test_datagen = ImageDataGenerator(rescale=1/255)
    train_generator = train_datagen.flow_from_directory(train_parent_directory,
                                        target_size = (75,75),
                                        batch_size = 40,
                                        class_mode = 'categorical',
                                        subset='training')
    val_generator = train_datagen.flow_from_directory(train_parent_directory,
                                          target_size = (75,75),
                                          batch_size = 7,
                                          class_mode = 'categorical',
                                          subset = 'validation')
    test_generator = test_datagen.flow_from_directory(train_parent_directory,
                                       target_size=(75,75),
                                       batch_size = 7,
                                       class_mode = 'categorical')
    return train_generator, val_generator, test_generator
train_dir = os.path.join('/content/train')
train_generator, validation_generator, test_generator = image_gen_w_aug(train_dir)
```

CLASSIFICATION AND PRE-TRAINING

In this section, we are checking the classified dataset into plant and unripe where the images are displayed sequentially depending on their classes. The images are divided on the level of batches and the weights are stored and calculated.

```
import numpy as np

CLASS_NAMES = np.array(['plant','unripe'], dtype='<U10')

import matplotlib.pyplot as plt

def show_batch(image_batch, label_batch):

  plt.figure(figsize=(25,20))

  for n in range(8):

     ax = plt.subplot(1,8,n+1)

     plt.imshow(image_batch[n])

     plt.title(CLASS_NAMES[label_batch[n]==1][0].title())

     plt.axis('off')

image_batch, label_batch = next(train_generator)

show_batch(image_batch, label_batch)
```

We are training the model using the Inception V3 algorithm which is a part of the Convolutional Neural Network where the images are reshaped into the dimensions 75*75*3 and the weights are stored from the imagenet function.

```
pre_trained_model = InceptionV3(input_shape = (75, 75, 3),

                                include_top = False,

                                weights = 'imagenet')
```

The images are classified as shown in Figure 14.

Figure 14 Classified Images Output

TRAINING AND SAVING THE MODEL

The model is retrained using the pre trained model for better accuracy. Model output for TL function trains the model based on the classification weights using the dense layer and softmax functions.

```
def model_output_for_TL (pre_trained_model, last_output):
    x = Flatten()(last_output)
    # Dense hidden layer
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.2)(x)
    # Output neuron.
    x = Dense(2, activation='softmax')(x)
    model = Model(pre_trained_model.input, x)
    return model
```

The pre-trained model accuracy and the compared values are represented using the below code. The loss and accuracy are calculated using categorical cross-entropy.

```
for layer in pre_trained_model.layers:
  layer.trainable = False
last_layer = pre_trained_model.get_layer('mixed3')
last_output = last_layer.output
model_TL = model_output_for_TL(pre_trained_model, last_output)
```

```
model_TL.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

model_TL.summary()
```

Training a model over large number of epochs increases the model accuracy rate and also helps to identify the faults in the dataset model and the training algorithm. We trained this model over 100 epochs and saved the model to the name called rps_model.

```
history_TL = model_TL.fit(
    train_generator,
    steps_per_epoch=5,
    epochs=100,
    verbose=1,
    validation_data = validation_generator)

tf.keras.models.save_model(model_TL,'rps_model.hdf5')
```

6.2.2 TESTING THE MODEL

The trained model is tested with two different methods one is by loading the saved model and the other is to test it by the model function in which the training code is mandatory. However, both the test cases give the exact output but the implementation of loading a saved model and testing is easier as it need not be run all the training code again.

TESTING USING TRAINING MODEL

This is the testing code using the model function. In this testing methodology, we need to load the image using the image URL to maintain it user free. The loaded image is pre-processed and sent to the function and the predictions are predicted accurately. This model will have high accuracy but the process will be very slow as it needs to run all the code again.

```
import PIL

import numpy as np
```

```
from matplotlib import pyplot

from PIL import Image, ImageOps

size = (75,75)

rps = np.array([1, 2, 3], dtype = '<U10')

url = input(" Please enter a url : ")

picture = Image.open(url)

pyplot.imshow(picture)

picture = ImageOps.fit(picture, size, Image.ANTIALIAS)

picture = picture.convert('RGB')

picture = np.asarray(picture)

picture = (picture.astype(np.float32) / 255.0)

picture_reshape = picture[np.newaxis,...]

y_pred = model_TL.predict(picture_reshape)

#y_pred = rps[y_pred[0]]

print(f'The predicted output is : {y_pred}')

if np.argmax(y_pred) == 0:

  print("It's a Plant")

elif np.argmax(y_pred) == 1:

  print("It's unripe")
```

The Output of the testing model using the trained model is in figure 15.

```
Please enter a url : /content/train/plant/1110409179_a988627dbd.jpg
The predicted output is : [[0.8240221  0.17597781]]
It's a Plant
```

Figure 15 Output using Trained Model

TESTING USING SAVED MODEL

This is the testing code using the pre-trained saved model. In this testing methodology, we need to load the image using the location. The loaded image is pre-processed and sent to the function in which the image is converted into an array and float values on which the predictions are predicted accurately. This model will have average accuracy but the process will be very fast as it needs to load only the pre-trained saved model.

```
import numpy as np

import tensorflow as tf

import filetype

from PIL import Image, ImageOps

def import_and_predict(image_data, model):

    size = (75,75)

    image = ImageOps.fit(image_data, size, Image.ANTIALIAS)

    image = image.convert('RGB')

    image = np.asarray(image)
```

```
        image = (image.astype(np.float32) / 255.0)
        img_reshape = image[np.newaxis,...]
        prediction = model.predict(img_reshape)
        return prediction
model = tf.keras.models.load_model('rps_model.hdf5')
print("Ripe and Unripe Tomato Classifier")
file = "/home/pi/open/testimage1.jpg"
if filetype.is_image(file):
    image = Image.open(file)
    prediction = import_and_predict(image, model)
    if np.argmax(prediction) == 0:
        print("It is Ripe!")
    elif np.argmax(prediction) == 1:
        print("It is Unripe!")
    print("Probability (0: Plant, 1: Unripe)")
    print(prediction)
else:
    print("You haven't uploaded an image file")
```

The Output of the testing model using the saved model is in figure 16.

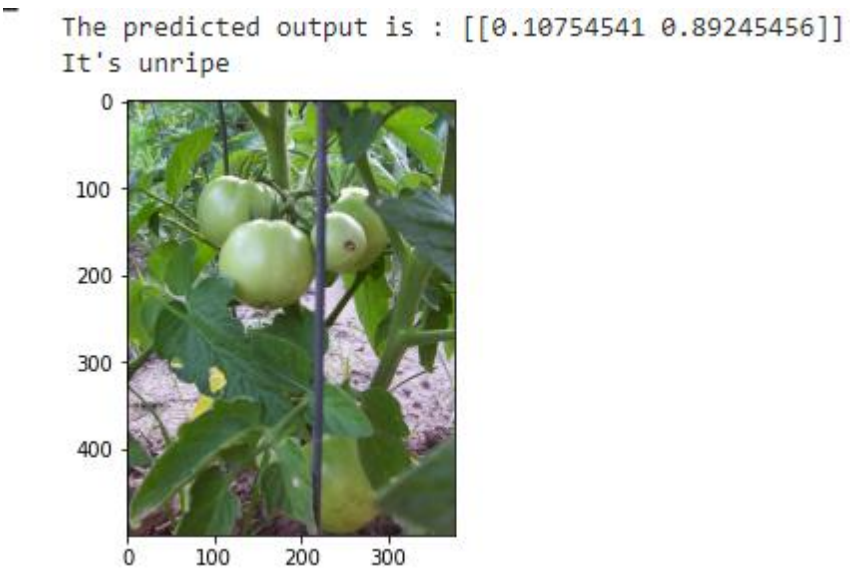


Figure 16 Output Using Saved Model

6.2.3 ROBOT CODE

For the ease of movement of the control of the robot, we designed different functions which function in different ways. We firstly initialized all the libraries, loaded the trained model, declared variables for the motors, declared the function of motors, and we set the thrust of the motors to 50%. This will be the initialization part of the functioning code and this will run inside the Raspberry Pi continuously. Any function is executed only when it is called.

```
from flask import Flask

from flask import Flask, render_template, Response, request

import RPi.GPIO as GPIO

import time

import os

import datetime

import sys

import subprocess

import cv2

import numpy as np
```

```
import tensorflow as tf

import filetype

from PIL import Image, ImageOps

global capture

capture=0

model = tf.keras.models.load_model('rps_model.hdf5')

app = Flask(__name__,template_folder='templates')

x1=3

x2=3

x3=3

x4=3

x5=3

x6=3

Motor1A = 24

Motor1B = 23

Motor1E = 25

Motor2A = 14

Motor2B = 15

Motor2E = 18

Motor3A = 8

Motor3B = 7

Motor3E = 21

Motor4A = 12

Motor4B = 16
```

```
Motor4E = 20

servoPIN1 =27

servoPIN2=22

servoPIN3 =17

servoPIN4 =10

servoPIN5 =9

servoPIN6 =11

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)# GPIO Numbering

GPIO.setup(Motor1A,GPIO.OUT)

GPIO.setup(Motor1B,GPIO.OUT)

GPIO.setup(Motor1E,GPIO.OUT)

GPIO.setup(Motor2A,GPIO.OUT)

GPIO.setup(Motor2B,GPIO.OUT)

GPIO.setup(Motor2E,GPIO.OUT)

GPIO.setup(Motor3A,GPIO.OUT)

GPIO.setup(Motor3B,GPIO.OUT)

GPIO.setup(Motor3E,GPIO.OUT)

GPIO.setup(Motor4A,GPIO.OUT)

GPIO.setup(Motor4B,GPIO.OUT)

GPIO.setup(Motor4E,GPIO.OUT)

GPIO.setup(servoPIN1, GPIO.OUT)

GPIO.setup(servoPIN2, GPIO.OUT)

GPIO.setup(servoPIN3, GPIO.OUT)
```

```
GPIO.setup(servoPIN4, GPIO.OUT)

GPIO.setup(servoPIN5, GPIO.OUT)

GPIO.setup(servoPIN6, GPIO.OUT)

p1 = GPIO.PWM(servoPIN1, 50)

p2 = GPIO.PWM(servoPIN2, 50)

p3 = GPIO.PWM(servoPIN3, 50)

p4 = GPIO.PWM(servoPIN4, 50)

p5 = GPIO.PWM(servoPIN5, 50)

p6 = GPIO.PWM(servoPIN6, 50)

pwm1=GPIO.PWM(Motor1E,100)

pwm2=GPIO.PWM(Motor2E,100)

pwm3=GPIO.PWM(Motor3E,100)

pwm4=GPIO.PWM(Motor4E,100)

pwm1.start(50)

pwm2.start(50)

pwm3.start(50)

pwm4.start(50)

p1.start(0)

p2.start(0)

p3.start(0)

p4.start(0)

p5.start(0)

p6.start(0)

print ("Done")
```

Import and predict function loads the image declared and analyze it. It converts the image into array and float values. If there is an error in the image it filters out and compares the float values with the loaded dataset and sends the prediction output.

```
def import_and_predict(image_data, model):
    size = (75,75)
    image = ImageOps.fit(image_data, size, Image.ANTIALIAS)
    image = image.convert('RGB')
    image = np.asarray(image)
    image = (image.astype(np.float32) / 255.0)
    img_reshape = image[np.newaxis,...]
    prediction = model.predict(img_reshape)
    return prediction
```

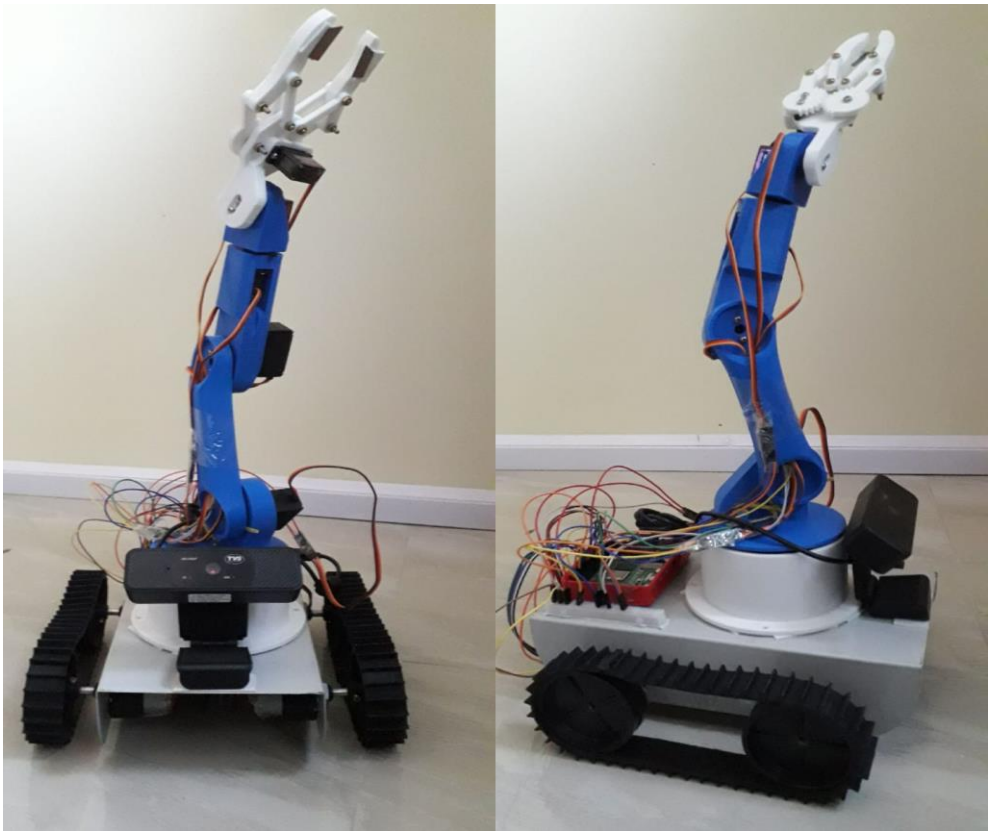


Figure 17 Overview of Designed Robot

Index function will be executed when the user connects to the web page and it returns the unripe HTML code as initially there is no detection of tomato.

```
@app.route("/")
def index():
    return render_template('index1.html')
```

The forward function moves the robot in the forward direction using the four DC motors at a 50% threshold value. This function works for 2 seconds and stops. This function is executed when the user clicks on the forward button.

```
@app.route('/A')
def forward():
    data1="A"
    GPIO.output(Motor1A,GPIO.HIGH)
    GPIO.output(Motor1B,GPIO.LOW)
    GPIO.output(Motor1E,GPIO.HIGH)
    GPIO.output(Motor2A,GPIO.HIGH)
    GPIO.output(Motor2B,GPIO.LOW)
    GPIO.output(Motor2E,GPIO.HIGH)
    GPIO.output(Motor3A,GPIO.HIGH)
    GPIO.output(Motor3B,GPIO.LOW)
    GPIO.output(Motor3E,GPIO.HIGH)
    GPIO.output(Motor4A,GPIO.HIGH)
    GPIO.output(Motor4B,GPIO.LOW)
    GPIO.output(Motor4E,GPIO.HIGH)
    time.sleep(2)
    stop()
```

```
    return render_template('index1.html')
```

The backward function moves the robot in the backward direction using the four DC motors at a 50% threshold value. This function works for 2 seconds and stops. This function is executed only when the user clicks on the backward button.

```
@app.route('/a')
def backward():
    data1="a"
    GPIO.output(Motor1A,GPIO.LOW)
    GPIO.output(Motor1B,GPIO.HIGH)
    GPIO.output(Motor1E,GPIO.HIGH)
    GPIO.output(Motor2A,GPIO.LOW)
    GPIO.output(Motor2B,GPIO.HIGH)
    GPIO.output(Motor2E,GPIO.HIGH)
    GPIO.output(Motor3A,GPIO.LOW)
    GPIO.output(Motor3B,GPIO.HIGH)
    GPIO.output(Motor3E,GPIO.HIGH)
    GPIO.output(Motor4A,GPIO.LOW)
    GPIO.output(Motor4B,GPIO.HIGH)
    GPIO.output(Motor4E,GPIO.HIGH)
    time.sleep(2)
    stop()
    return render_template('index1.html')
```

The left function moves the robot in the left direction using the four DC motors at a 50% threshold value. This function works for 2 seconds and stops. This function is executed only when the user clicks on the left button.

```
@app.route('/B')
def left():
    data1="B"
    GPIO.output(Motor1A,GPIO.LOW)
    GPIO.output(Motor1B,GPIO.LOW)
    GPIO.output(Motor1E,GPIO.LOW)
    GPIO.output(Motor2A,GPIO.HIGH)
    GPIO.output(Motor2B,GPIO.LOW)
    GPIO.output(Motor2E,GPIO.HIGH)
    GPIO.output(Motor3A,GPIO.LOW)
    GPIO.output(Motor3B,GPIO.LOW)
    GPIO.output(Motor3E,GPIO.LOW)
    GPIO.outemplacedor4A,GPIO.HIGH)
    GPIO.output(Motor4B,GPIO.LOW)
    GPIO.output(Motor4E,GPIO.HIGH)
    time.sleep(2)
    stop()
    return render_template('index1.html')
```

The right function moves the robot in the right direction using the four DC motors at a 50% threshold value. This function works for 2 seconds and stops. This function is executed only when the user clicks on the right button.

```
@app.route('/b')
def right():
    data1="b"
    GPIO.output(Motor1A,GPIO.HIGH)
```

```
    GPIO.output(Motor1B,GPIO.LOW)
    GPIO.output(Motor1E,GPIO.HIGH)
    GPIO.output(Motor2A,GPIO.LOW)
    GPIO.output(Motor2B,GPIO.LOW)
    GPIO.output(Motor2E,GPIO.LOW)
    GPIO.output(Motor3A,GPIO.HIGH)
    GPIO.output(Motor3B,GPIO.LOW)
    GPIO.output(Motor3E,GPIO.HIGH)
    GPIO.output(Motor4A,GPIO.LOW)
    GPIO.output(Motor4B,GPIO.LOW)
    GPIO.output(Motor4E,GPIO.LOW)
    time.sleep(2)
    stop()
    return render_template('index1.html')
```

The stop function stops all the motors and loads the image from the location and checks whether the file is in image format or not. It also checks for the ripeness in the image and returns the related webpage on the prediction.

```
@app.route('/C')
def stop():
    data1="C"
    GPIO.output(Motor1A,GPIO.LOW)
    GPIO.output(Motor1B,GPIO.LOW)
    GPIO.output(Motor1E,GPIO.LOW)
    GPIO.output(Motor2A,GPIO.LOW)
    GPIO.output(Motor2B,GPIO.LOW)
```

```
    GPIO.output(Motor2E,GPIO.LOW)
    GPIO.output(Motor3A,GPIO.LOW)
    GPIO.output(Motor3B,GPIO.LOW)
    GPIO.output(Motor3E,GPIO.LOW)
    GPIO.output(Motor4A,GPIO.LOW)
    GPIO.output(Motor4B,GPIO.LOW)
    GPIO.output(Motor4E,GPIO.LOW)
    file = "/home/pi/Trail/test.jpg"
    if filetype.is_image(file):
        image = Image.open(file)
        prediction = import_and_predict(image, model)
        if np.argmax(prediction) == 0:
            return render_template('index2.html')
        elif np.argmax(prediction) == 1:
            return render_template('index1.html')
    else:
        return render_template('index2.html')
```

Bottom servo functions one and two increase and decreases the angle of the base motor of the robotic arm manipulator. It changes the angle by one degree and these functions are executed only when the bottom servo buttons are pressed.

```
@app.route('/d')
def botomservo1():
    data1="d"
    global x1
    x1=x1+1
```

```
    p1.ChangeDutyCycle(x1)
    return render_template('index1.html')
@app.route('/e')
def botomservo2():
    data1="e"
    global x1
    x1=x1-1
    p1.ChangeDutyCycle(x1)
    return render_template('index1.html')
```

First-hand functions one and two increase and decrease the angle of the first-hand motor of the robotic arm manipulator. It changes the angle by one degree and these functions are executed only when the first-hand buttons are pressed.

```
@app.route('/f')
def firsthand1():
    data1="f"
    global x2
    x2+=1
    p2.ChangeDutyCycle(x2)
    return render_template('index1.html')
@app.route('/g')
def firsthand2():
    data1="g"
    global x2
    x2=x2-1
    p2.ChangeDutyCycle(x2)
```

```
    return render_template('index1.html')
```

Second-hand functions one and two increase and decrease the angle of the second-hand motor of the robotic arm manipulator. It changes the angle by one degree and these functions are executed only when the second-hand buttons are pressed.

```
@app.route('/h')
def secondhand1():
    data1="h"
    global x3
    x3+=1
    p3.ChangeDutyCycle(x3)
    return render_template('index1.html')
@app.route('/i')
def secondhand2():
    data1="i"
    global x3
    x3-=1
    p3.ChangeDutyCycle(x3)
    return render_template('index1.html')
```

Third-hand functions one and two increase and decrease the angle of the third-hand motor of the robotic arm manipulator. It changes the angle by one degree and these functions are executed only when the third-hand buttons are pressed.

```
@app.route('/j')
def thirdhand1():
    data1="j"
    global x4
```

```
    x4+=1
    p4.ChangeDutyCycle(x4)
    return render_template('index1.html')
@app.route('/k')
def thirdhand2():
    data1="k"
    global x4
    x4-=1
    p4.ChangeDutyCycle(x4)
    return render_template('index1.html')
```

Fourth-hand functions one and two increase and decreases the angle of the fourth-hand motor of the robotic arm manipulator. It changes the angle by one degree and these functions are executed only when the fourth-hand buttons are pressed. This movement helps in the movement of the cutter arm.

```
@app.route('/l')
def forthhand1():
    data1="l"
    global x5
    x5+=1
    p5.ChangeDutyCycle(x5)
    return render_template('index1.html')
@app.route('/m')
def forthhand2():
    data1="m"
    global x5
```

```
    x5-=1
    p5.ChangeDutyCycle(x5)
    return render_template('index1.html')
```

Cutter functions one and two increase and decreases the angle of the cutter motor of the robotic arm manipulator. It changes the angle by one degree and these functions are executed only when the cutter buttons are pressed. This movement helps in cutting the ripe tomatoes.

```
@app.route('/n')
def cuter1():
    data1="n"
    global x6
    x6+=1
    p6.ChangeDutyCycle(x6)
    return render_template('index1.html')
@app.route('/o')
def cuter2():
    data1="o"
    global x6
    x6-=1
    p6.ChangeDutyCycle(x6)
    return render_template('index1.html')
```

Gen frames function is executed when the user clicks on the capture button. It initializes the camera to capture the frame at that movement of time and save it to a location with a .jpg extension.

```
camera = cv2.VideoCapture(1)
def gen_frames():
```

```
global out, capture
while True:
    success, frame = camera.read()
    if success:
        if(capture):
            print(capture)
            cv2.imwrite('/home/pi/Trail/test.jpg', frame)
            capture=0
        try:
            ret, buffer = cv2.imencode('.jpg', cv2.flip(frame,1))
            frame = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
        except Exception as e:
            pass
    else:
        pass
```

The video feed function returns the live video frame to frame to the user and this function runs continuously without any initialization. This function starts when the user connects to the terminal or web page.

```
@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame')
```

Tasks functions are executed in the backend it helps the system whether the user has clicked on the button or not. This function is executed always and checks for the click on the capture button.

```
@app.route('/requests',methods=['POST','GET'])
def tasks():
    global switch,camera
    if request.method == 'POST':
        if request.form.get('click') == 'Capture':
            global capture
            capture=1
    elif request.method=='GET':
        return render_template('index1.html')
    return render_template('index1.html')
```

This main function runs for one time and starts all the other functions in the code.

```
if __name__ == "__main__":
    print ("Strat")
    app.run(host='192.168.43.175', port=5010)
```

6.2.4 HTML CODE

We designed the web interface for the user to interact with the robot. The robot controls are designed on the webpage. We designed two Html pages for two different cases, one is when there is ripe tomato and the other is unripe. Below code is the HTML template of the unripe tomato code, this page is returned when the user captures a picture using the capture function and detects an unripe tomato or identification of no tomato.

```
<!DOCTYPE html>

<html>

<h1>AAT</h1>

<h2>Robot Controls</h2>

<h3> Unripe</h3>

<br>

<a href=\A><button>forward</button></a>

<a href=\a><button>backward</button></a><br/>

<a href=\B><button>left</button></a>

<a href=\b><button>right</button></a><br/>

<a href=\C><button>stop</button></a><br/>

<a href=\d><button>bottomservo left</button></a>

<a href=\e><button>bottomservo right</button></a><br/>

<a href=\f><button>firsthand left</button></a>

<a href=\g><button>firsthand right</button></a><br/>

<a href=\h><button>secondhand left</button></a>

<a href=\i><button>secondhand right</button></a><br/>

<a href=\j><button>thirdhand left</button></a>

<a href=\k><button>thirdhand right</button></a><br/>

<a href=\l><button>forthhand left</button></a>

<a href=\m><button>forthhand right</button></a><br/>

<a href=\n><button>cutter left</button></a>

<a href=\o><button>cutter right</button></a><br/>

<form method="post" action="{{ url_for('tasks') }}">
```

```
<input type="submit" value="Capture" name="click"/>
</form>
<img src="{{ url_for('video_feed') }}" height="80%">
</html>
```

Below code is the HTML template of the ripe tomato code, this page is returned when the user captures a picture using the capture function and detects a ripe tomato. This page may even be returned when the capture function fails and the previously captured tomato is ripe. Each hypertext reference refers to a control button for the robot.

```
<!DOCTYPE html>
<html>
<h1>AAT</h1>
<h2>Robot Controls</h2>
<h3> Ripe</h3>
<br>
<a href=\A><button>forward</button></a>
<a href=\a><button>backward</button></a><br/>
<a href=\B><button>left</button></a>
<a href=\b><button>right</button></a><br/>
<a href=\C><button>stop</button></a><br/>
<a href=\d><button>bottomservo left</button></a>
<a href=\e><button>bottomservo right</button></a><br/>
<a href=\f><button>firsthand left</button></a>
<a href=\g><button>firsthand right</button></a><br/>
<a href=\h><button>secondhand left</button></a>
<a href=\i><button>secondhand right</button></a><br/>
```

```
<a href=\j><button>thirdhand left</button></a>

<a href=\k><button>thirdhand right</button></a><br/>

<a href=\l><button>forthhand left</button></a>

<a href=\m><button>forthhand right</button></a><br/>

<a href=\n><button>cutter left</button></a>

<a href=\o><button>cutter right</button></a><br/>

<form method="post" action="{{ url_for('tasks') }}">

<input type="submit" value="Capture" name="click"/>

</form>

<img src="{{ url_for('video_feed') }}" height="80%">

</html>
```

ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- Cost Friendly.
- Easy to handle and maintain.
- Suitable for Household Agriculture.
- Less Power Consumption.
- Easy repairs and readily replaceable.
- Manipulations can be made easily.
- Can be controlled only by the user.

DISADVANTAGES

- Unstable.
- Delay between the user and the system is high.
- Not suitable for large acres of land.
- Security is breachable.
- Regular Monitoring is required.
- Limited range of control.

APPLICATIONS

- Household Agriculture
- Green House Agriculture
- Picking Tomatoes
- Regular Monitoring
- Ripeness Detection

CONCLUSION

We designed a robot that aids in tomato harvesting. The robot is capable of picking the ripe and unripe fruits on command. It has the ability to capture the tomatoes and predict the ripeness level by comparing the array values of the picture to the trained dataset. The robot can move from place to place and gives a live feed of data, along with that it can also pick a tomato using the robotic arm and helps in harvesting. The model for detection of ripeness is trained on the Inception V3 algorithm where the model is trained over large epochs and has greater accuracy. The trained model can be used anywhere in machine learning concepts and can also be used to increase the model accuracy. The robot is capable of cutting the tomato root node without producing damage to the tomato.

FUTURE SCOPE

In the next ten years, robotics will replace people in the agricultural industry. Agriculture will be mechanized, and all crops will be cultivated without human intervention, resulting in an output rate that is ten times higher than it is now. The harvesting robot will be automated and employed in all types of farming, from small-scale farming to large-scale farming. As agriculture becomes a pastime in the future, the tomato-picking robot will be employed in every home. By 2050, farming will be considered as important as feeding a pet. The robotic hands that will pick the tomatoes will be mechanized, and a suction mechanism will be used. The categorization in the machine learning model will be enhanced so that it can even estimate the period of maturity and the amount of water a plant requires for optimum growth. The future of agriculture is in the imaginations of robots and their creators.

BIBLIOGRAPHY

1. [1] *Selective Harvesting Robotics: Current Research, Trends, and Future Directions | SpringerLink*.(n.d.). Retrieved 11 May 2022, from https://link.springer.com/article/10.1007/s43154-020-00034-1
    1. [2] Liu, G., Mao, S., & Kim, J. H. (2019). A mature-tomato detection algorithm using machine learning and color analysis. Sensors, 19(9), 2023.
2. [3] L. Jafari, 'What Are Non-Functional Requirements? Types and Examples', *WINaTALENT | Blog*, May 31, 2020. https://winatalent.com/blog/2020/05/what-are-non-functional-requirements-types-and-examples/ (accessed May 11, 2022).
3. [4] 'Advantages and Disadvantages of Python - How it is dominating Programming World', *DataFlair*, Jan. 02, 2018. https://data-flair.training/blogs/advantages-and-disadvantages-of-python/ (accessed May 11, 2022).
4. [5]'Deep Learning – Inherent Tech'. https://inherenttech.com/deep-learning/ (accessed May 11, 2022).
5. [6] 'Fusion 360 | 3D CAD, CAM, CAE & PCB Cloud-Based Software | Autodesk'. https://www.autodesk.in/products/fusion-360/overview?panel=buy&term=1-YEAR (accessed May 11, 2022).
6. [7] 'Home - Keras 2.0.2 Documentation'. https://faroit.com/keras-docs/2.0.2/ (accessed May 11, 2022).
7. [8]'What is NumPy? — NumPy v1.23.dev0 Manual'. https://numpy.org/devdocs/user/whatisnumpy.html (accessed May 11, 2022).
8. [9]'About', *OpenCV*. https://opencv.org/about/ (accessed May 11, 2022).
9. [10] 'Wheel', *Wikipedia*. May 10, 2022. Accessed: May 11, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wheel&oldid=1087082626
10. [11] 'UML - Statechart Diagrams'. https://sceweb.sce.uhcl.edu/helm/WEB-TOC-UML/Tutorial/uml__statechart_diagrams.html (accessed May 11, 2022).
11. [12] 'Deep Learning: Understanding The Inception Module | by Richmond Alake | Towards Data Science'. https://towardsdatascience.com/deep-learning-understand-the-inception-module-56146866e652 (accessed May 11, 2022).
12. S. N. Kulkarni and S. Kumar Singh, "Object Sorting Automated System using Raspberry Pi," 2018 3rd International Conference on Communication and Electronics Systems (ICCES), 2018, pp. 217-220, https://doi.org/10.1109/CESYS.2018.8724056.

13. J. -D. Lee, W. -C. Li, J. -H. Shen and C. -W. Chuang, "Multi-robotic arms automated production line," 2018 4th International Conference on Control, Automation and Robotics (ICCAR), 2018, pp. 26-30, https://doi.org/10.1109/ICCAR.2018.8384639.
14. P. Siagian and K. Shinoda, "Web based monitoring and control of robotic arm using Raspberry Pi," 2015 International Conference on Science in Information Technology (ICSITech), 2015, pp. 192-196, https://doi.org/10.1109/ICSITech.2015.7407802.
15. G. Aceto, V. Persico, and A. Pescapé, "A Survey on Information and Communication Technologies for Industry 4.0: State-of-the-Art, Taxonomies, Perspectives, and Challenges," IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3467-3501, 2019, https://doi.org/10.1109/COMST.2019.2938259.
16. Ko M H, Ryuh B S, Kim K C, Suprem A, Mahalik N P. Autonomous greenhouse mobile robot driving strategies from system integration perspective: review and application. Transactions on Mechatronics. America: IEEE/ASME, 2014; pp.1–12.
17. Xiong C, Xiaoman C, Zou X 2017 A Method for Identification and Matching of the Picking Point for Mature Litchi under Structural Environment Journal of Applied Biotechnology & Bioengineering Vol 3 Issue 6 p 4.
18. Stoelen M, Krzysztof K, Tejada V F, Heiberg N, Balaguer C and Korsaeth A 2015 Low-Cost Robotics for Horticulture: A Case Study on Automated Sugar Pea Harvesting 10th European Conference on Precision Agriculture (ECPA) DOI: 10.3920/978-90-8686-814-8 34.
19. Duckett T, Pearson S, Blackmore S and Grieve B, 2018 Agriculture Robotics: The Future of Robotic Agriculture, UK-RAS Network Robotics & Autonomous System ISSN 2391-4414.