HAR-BOT

PROJECT REPORT

submitted by

ABHISHEK BABU T	**TCR19EC004**
ASHIK M M	**TCR19EC019**
KARTHIK S	**TCR19EC035**
SIVAPRASAD A S	**TCR19EC051**

to
APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the Degree
of
Bachelor of Technology
in
Electronics & Communication Engineering

Department of Electronics & Communication Engineering
Government Engineering College Thrissur
May 2023

DECLARATION

We undersigned hereby declare that the project report **HAR-BOT** submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala, is a bonafide work done by us under supervision of Prof. VIJEESH V. This submission represents our ideas in our own words and where ideas or words of others have been included, We have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

ABHISHEK BABU T

Place:

ASHIK M M

Date:

KARTHIK S

SIVAPRASAD A S

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Government Engineering College Thrissur

CERTIFICATE

This is to certify that the report entitled **HAR-BOT** is submitted by students named **ABHISHEK BABU T (TCR19EC004) , ASHIK M M(TCR19EC019) , KARTHIK S(TCR19EC035) , SIVAPRASAD A S (TCR19EC051)** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Electronics & Communication Engineering is a bonafide record of the project work carried out by him/her under my/our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Project Guide Project Coordinator Head of Department

DEPARTMENT VISION

To become a nationally acclaimed Department of higher learning and research that will serve as a source of knowledge and expertise in Electronics & Communication Engineering.

DEPARTMENT MISSION

1. To provide quality education in the area of Electronics and Communication Engineering, to produce innovative and ethically driven professionals adept at dealing with a globally competitive environment, for the welfare of the nation.
2. To inculcate inquisitiveness in young graduates thereby persuading them to undertake research in emerging areas of Electronics and Communication Engineering.

ACKNOWLEDGMENT

I wish to record my indebtedness and thankfulness to all who helped me prepare this Project Report titled **HAR-BOT** and present it in a satisfactory way. I am especially thankful to my guide and supervisor Prof. VIJEESH V for giving me valuable suggestions and critical inputs in the preparation of this report. I am also thankful to Dr. SINITH M S, Head of Department of Electronics & Communication Engineering for encouragement.

Any attempt of gratitude would be incomplete without the mention of the project coordinators Prof. RIYAS K S and Prof. NISSA SURLING for their suggestions and solutions during times of dilemma. I also thank all the staff of the department, for their whole- hearted support and cooperation. Finally, I would like to acknowledge my deep sense of gratitude to all well-wishers and friends who helped me directly and indirectly to continue with this work.

ABHISHEK BABU T

ASHIK M M

KARTHIK S

SIVAPRASAD A S

ABSTRACT

In India, agriculture has traditionally been a labor-intensive industry. However, there is a need to investigate autonomous alternatives in place of conventional ways in order to cater to the fast-expanding population in the face of rising labor expenses. One such procedure in agriculture is harvesting. The greatest care should be taken because picking fruits too early could lower output. This is more obvious, particularly in terrace and greenhouse farming. We, therefore, introduce our ”HAR-BOT” in order to make that process more effective and economical. HAR-BOT is a mobile robot arm manipulator that can locate and choose ripe fruits like tomatoes. Har-bot will navigate a path without running into any other objects. Using a camera module, it should be able to identify ripe tomatoes and determine their pose. The robot intends to pick ripe tomatoes and put them in a basket using a robotic arm. A smartphone app that is installed remotely can control and monitor how Harbot is operating.

CONTENTS

LIST OF FIGURES

ABBREVATIONS

- SCARA – Selective Compliance Assembly Robot Arm
- IK – Inverse Kinematics
- FK – Forward Kinematics
- ML – Machine learning
- YOLO – You Only Look Once
- DOF – Degree of Freedom Arm
- LFR – Line Follower Robot
- CNN – Convolutional Neural Network
- SSD – Single Shot MultiBox Detector
- IR – Infra Red
- HARBOT – Harvestiong Robot
- SVM – Support Vector Machine
- UI – User Interface
- DB – Data Base
- API – Application Interface
- LIPO – Lithium Polymer
- PWM – Pulse Width Modulation
- RPM – Revolution per Minute
- HDMI – High Defenition Multimedia Interface
- LAN – Local Area Network
- FFC – Flexible Flat Cable

Chapter 1

INTRODUCTION

1.1 GENERAL BACKGROUND

In India, agriculture has traditionally been a labor-intensive industry. However, there is a need to investigate autonomous alternatives in place of conventional ways in order to cater to the fast-expanding population in the face of rising labor expenses. One such procedure in agriculture is harvesting. The greatest care should be taken because picking fruits too early could lower output. This is more obvious, particularly in terrace and greenhouse farming. Fruit harvesting robots can be used to harvest a variety of different types of fruit, including apples, pears, peaches, and citrus. They are typically more efficient and accurate than manual labor, and they can work longer hours without getting tired. A harvesting robot's capabilities and unique design determine its range of operation. While some harvesting robots are more adaptable and can handle a range of crops, others are built to handle just one type of crop. While certain harvesting robots are better suited to certain types of terrain, others can adapt to a wider range of settings. A harvesting robot's capabilities may also be influenced by its size, mobility, and the accessibility of necessary infrastructures, such as recharging stations and maintenance facilities.

1.2 OBJECTIVES OF THE PROJECT

- To implement an Automated harvesting solution
- Labor-saving

- Low-cost harvesting
- Increase productivity
- Integrating the latest technologies in daily life
- To make affordable robots
- To gain application level knowledge in modern technologies
- Integrating technologies like python programming, computer vision,networking, robotics,power electronics and micro controller to build a cost effective solution

1.3 PURPOSE AND NEED OF THE PROJECT

Harvesting robots are designed to perform the labor-intensive tasks involved in harvesting crops. They can be used to harvest a variety of crops, including fruits, vegetables, and grains. There are several benefits to using harvesting robots:

- **Increased efficiency:** Harvesting robots can work 24 hours a day, 7 days a week, without getting tired or needing breaks. This allows them to harvest crops faster and more consistently than human workers.
- **Reduced labor costs:** Harvesting robots can significantly reduce the labor costs associated with crop harvesting. This is especially beneficial in regions where labor is scarce or expensive.
- **Improved food safety:** Harvesting robots can minimize the risk of contamination by reducing the need for human handling of the crops.
- **educed impact on the environment:** R Harvesting robots can minimize the environmental impact of crop harvesting by reducing the need for chemical pesticides and herbicides.

Overall, harvesting robots can provide a number of benefits for farmers and the agriculture industry, including increased efficiency, reduced labor costs, improved food safety, and a reduced impact on the environment. So making such robot at affordable price will help to popularize robots among modern people.

1.4 SCOPE OF THE PROJECT

Crop harvesting is carried out by automated equipment known as harvesting robots. Overall, the scope of a harvesting robot is determined by its capabilities and the needs of the farming operation in which it is used. One potential challenge to the adoption of fruit harvesting robots is their high cost, which may limit their use to larger commercial farming operations. However, as technology improves and the cost of these systems decreases, it is possible that smaller farmers may also begin to adopt fruit harvesting robots. Our project is intended to serve farming in confined areas like terraces, greenhouses, etc for timely harvesting of the fruit. Currently, small-scale robots like floor cleaning robots, serving robots, etc are becoming common in households. Like that this robot will also have the potential to be a part of a people's farm or house.

1.5 FEASIBILITY

To begin harvesting, the planned system may be remotely operated using a smartphone application. It works nicely on flat surfaces such as a patio or a greenhouse. All hardware control may be accomplished with the Raspberry Pi and the Python programming language. Because the system is built with components that are readily available in the surrounding environment, the cost of manufacture may be significantly lowered when compared to other robots on the market.

Chapter 2

REQUIREMENT ANALYSIS

The process of refining, modelling, and specifying the already identified user needs is known as requirements analysis and validation. This stage of development includes the systematic use of tried-and-true ideas, techniques, languages, and tools for the efficient analysis, documenting, and ongoing evolution of user demands as well as the definition of a system's external behaviour to meet those needs. This chapter explains the technique used for requirement elicitation and the user requirements that were as a result acquired. Following validation using an appropriate approach, the criteria are also completed.

2.1 METHOD OF REQUIREMENT ELICITATION

To elicit the requirements, we adopted certain techniques which helped us to identify the expectations from users, requirements at the system level, and expected behavior of various subsystems. The techniques we employed were:

2.1.1 Group Discussion

Examined several solutions to the issue description. Discussed the advantages and disadvantages of existing options. Each group member addressed the topic from a different angle and gathered pertinent data from potential product end users by conducting interviews, brief discussions, etc. Based on it, the group members engage in discussion.

2.1.2 Brain Storming

Through brainstorming meetings, the potential solutions to the requirements were chosen. At this point, the best potential answer is chosen. One problem shouldn't lead to another once it has been solved. Therefore, the development of a basic sensor-based IoT system is proposed.

2.1.3 Literature Survey

We investigated projects that were comparable to the intended application in order to improve implementation and strengthen the robot's design. We looked at several academic papers that discussed problems with robot design and arm actuation. The components required to determine the robot's motion, the end effector system's design, and the system's weight were estimated using the results of this field of study.

Our group looked at previously established robotic systems that were used to pick fruits. Several articles and research have been written on the development of a three- a dimensional robotic arm that would be used to harvest fruits growing in space. The end effector system uses a scissor to cut the branches on which the fruits grow, and the fruit is subsequently placed in the basket attached to the body. A camera module is employed to detect fruits on the farm. Color and shape detection are both done with the camera module. Application of image processing techniques [6, 15, 8] is utilized to detect the fruit and then move the robot to the location where they grow. The position of the fruits in the field was determined using a visual algorithm that was built. A fruit-picking robot was guided by this information. One fruit at a time was collected by the robot. The robot's the success rate was reported to be 60%. In a very recent paper [9], the system was powered by a 12-volt Lithium-Ion (LiFePO4) battery.

In our design, the robotic arm can move up and down and can rotate 180 degrees from the base so that fruits can be placed in the container. In order to pluck the fruits, we close the gripper hand connected with blades. In [10], the author has designed an arm that can move vertically and horizontally in one plane, which restricts the access of fruits in other planes. [11] employs a 4-DoF robotic arm with 0-180 degree base rotation, forward-backwards movement, up-down motion, and 0-180 degree gripper opening. However, this makes picking the fruits more effective.

Object identification [12, 13] and tracking [**?**, 15] is a well-studied field with numerous practical applications. The primary purpose of object detection systems is to recognise the desired elements in a picture, as well as to determine their position in the current world and categorise them. Machine learning algorithms [16, 17, 12] such as Scale Invariant Feature Transform (SIFT), Random Sample Consensus (RANSAC), and Convolutional Neural Network are used in vision-based software systems to increase detection and tracking accuracy (CNN). The use of neural networks necessitates sophisticated technology as well as access to a large database, resulting in a complex system. Such technologies are more difficult to operate remotely and to put into practise in the real world.

Here object detection is performed using YOLOv5.[1] From the detected object we collect the pixel coordinate and convert it into real-world coordinates using a linear transformation. Along with that with the help of OpenCV, we perform depth estimation. By these methods, we will obtain real-world coordinates. These coordinates are converted to angles for rotating motors by applying inverse kinematics. The motors are rotated based on the angle and take appropriate positions. Remote access for starting the robot motion is done with the help of a firebase. Path for the motion of the robot can be created with the support of a line follower system. A Line follower-based path detection will be most suited for applications like fruit harvesting.

2.2 USER REQUIREMENTS

Some of the requirements that is expected by the users are :

- Automated harvesting solution
- Labor-saving
- Low-cost harvesting
- Increase productivity
- Remote connectivity

2.3 PROJECT REQUIREMENTS

So by considering user requirements, the project is trying to achieve requirements such as remotely controlled automated harvesting solution for terrace and green house by integrating technologies like python programming, computer vision,networking, robotics,power electronics and micro controller to build a cost effective solution.

Chapter 3

DESIGN AND IMPLEMENTATION

3.1 ARCHITECTURAL DESCRIPTION

An architectural description is a document or set of documents that describe the architecture of a system. It provides a high-level overview of the system's structure, components, and interactions, and helps to ensure that the system meets its functional and non-functional requirements.An architectural description typically includes an overview of the system and its purpose, and outlines the scope of the architectural description, section describes the overall vision for the system's architecture, including its goals and objectives, and how it will meet the system's requirements, section describes the design patterns and architectural styles that are used in the system, and how they are implemented etc.By providing a clear and comprehensive overview of the system's architecture, an architectural description helps to ensure that the system meets its requirements and can be easily understood and maintained over time.

HAR-BOT's architectural design here comprises of a movable base capable of navigating difficult terrain and uneven surfaces. The foundation may be fitted with wheels or tracks for movement and may be built to work in a variety of environments, such as a patio or green house.The robot will normally have one arm or manipulators attached to the base, which will be utilised to grip and handle the product. The arms might be outfitted with a pi camera to assist in identifying the crop and ensuring precise harvesting.Aside from the hardware, the harvesting robot will need specialised software to coordinate its motions and activities. This software may include algorithms for identifying the location and ripeness of the produce, as well as machine learning models to improve the accuracy of the harvesting process over time.Overall, the design of a harvesting robot is focused on optimizing the efficiency and accuracy of the harvesting process, while reducing the need for human labor and minimizing damage to the produce.

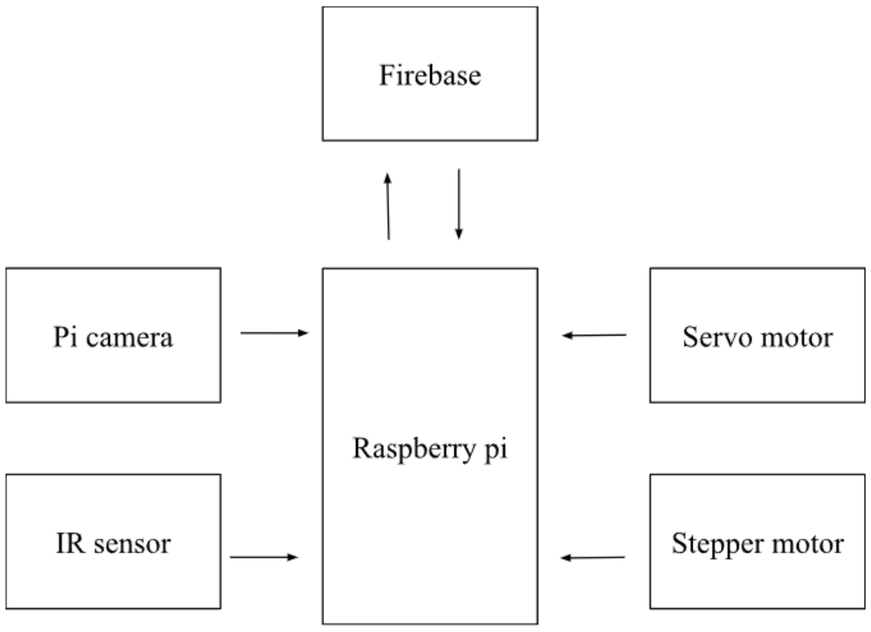


Figure 3.1: Block diagram

3.2 DESIGN DESCRIPTION

This is the design of HAR-BOT we intended to build. It has three degrees of freedom (DOF). They are:-

- One for up and down motion
- One for planar elbow motion
- One for the planar motion of gripper hand

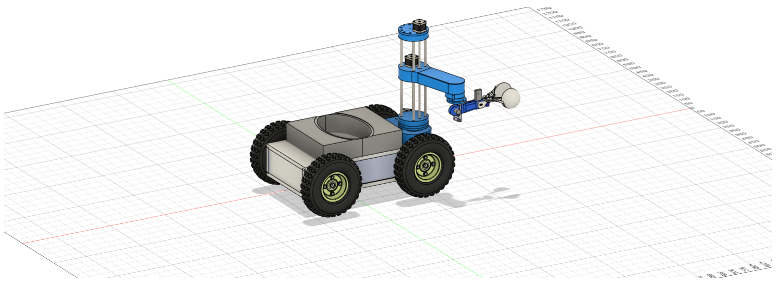


Figure 3.2: Design of HAR-BOT

We also have a closing and opening motion for the gripper, which helps to cut and collect the fruit. The path for the motion of the robot is supposed to be built with line follower to techniques to avoid reachability issues during plucking the fruits.

All the designs displayed here is created with the help of the cad software named fusion360. It possible to desing a robot with different views using this software. A robot can be viewed in different angles for better understanding of its working and its structure. HAR-BOT viewed in different angles is shown below

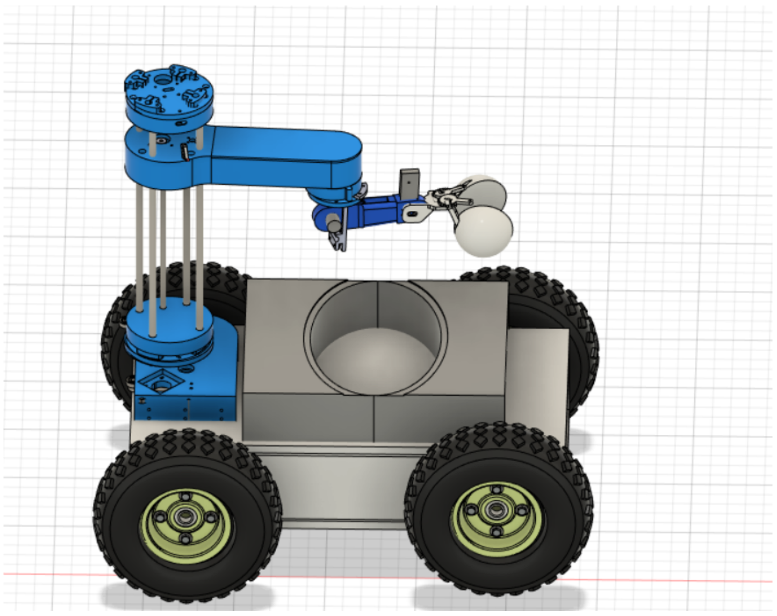


Figure 3.3: top view

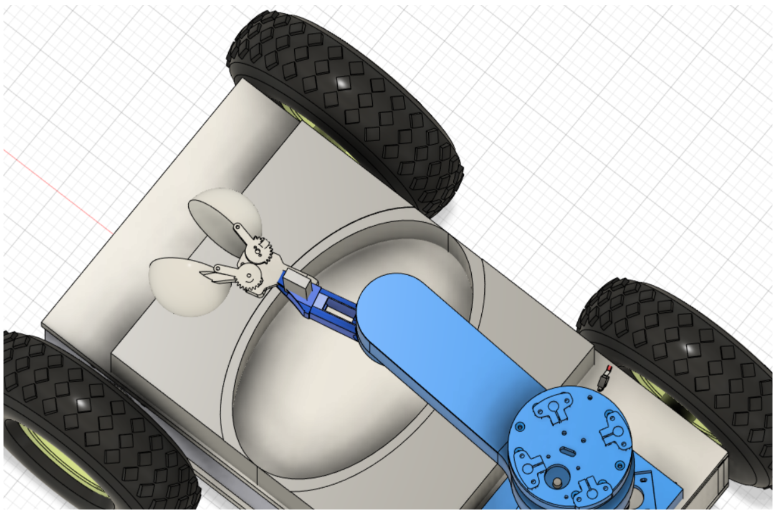


Figure 3.4: gripper view

3.3 DECOMPOSITION DESCRIPTION

The System is divided into 4 module based on the functionality of the system, that is it is divided into module for each of the functionality the system would provide. The Parts are :

- **Real time object detection :** Mobilenet ssd and opencv are used for real-time object detection. A MobileNet feature extractor is followed by a detection module in the MobileNet SSD architecture. A lightweight convolutional neural network (CNN) extracts features from an input picture using the feature extractor. The detection module then utilises these properties to detect things. MobileNet SSD is a popular choice for real-time object identification on mobile devices because it can achieve excellent accuracy while keeping quick inference times. The Pi camera is used to identify objects in real time by interacting with the Raspberry Pi.

- **Arm manipulator :** For picking matured tomatoes, a scara robot with three degrees of freedom is employed as an arm manipulator. The stepper motor drives the arms. Inverse kinematics is used to select the angles for stepper rotation. To handle tomatoes, a gripper mechanism is attached at the tip and is operated by a servo motor.

- **Line follower :** A line follower is used to allow the robot to move smoothly across a garden with several plants set in a consistent pattern. The IR sensor is utilised to keep the robot on course. The gear motor is rotated based on the IR sensor's measured values.
- **Remote Access :** Remote access entails commanding HAR-BOT from a remote place.This is accomplished through the usage of a mobile application in which HAR-BOT is exclusively linked to registered users. Data exchanged between the bot and users is handled by Firebase, which provides real-time database support. The smartphone application allows the user to start the harbot and get updates on the fruits harvested.

3.4 WORKFLOW

1. Start the HARBOT using a mobile app
2. Detect plant using specific stops in line follower path
3. Stop motion
4. Check for ripened tomato by moving down the arm
5. If detected obtain its real-world coordinates
6. Apply coordinates to inverse kinematic solver
7. Rotate motors according to the output of the inverse kinematic solver
8. Close the gripper and collect fruit
9. Drop fruit in a basket and gain initial state
10. Go back to step 4 and repeat

11. If arm reached the bottom, then go to the top

12. Continue motion of vehicle

3.5 CIRCUIT DIAGRAM

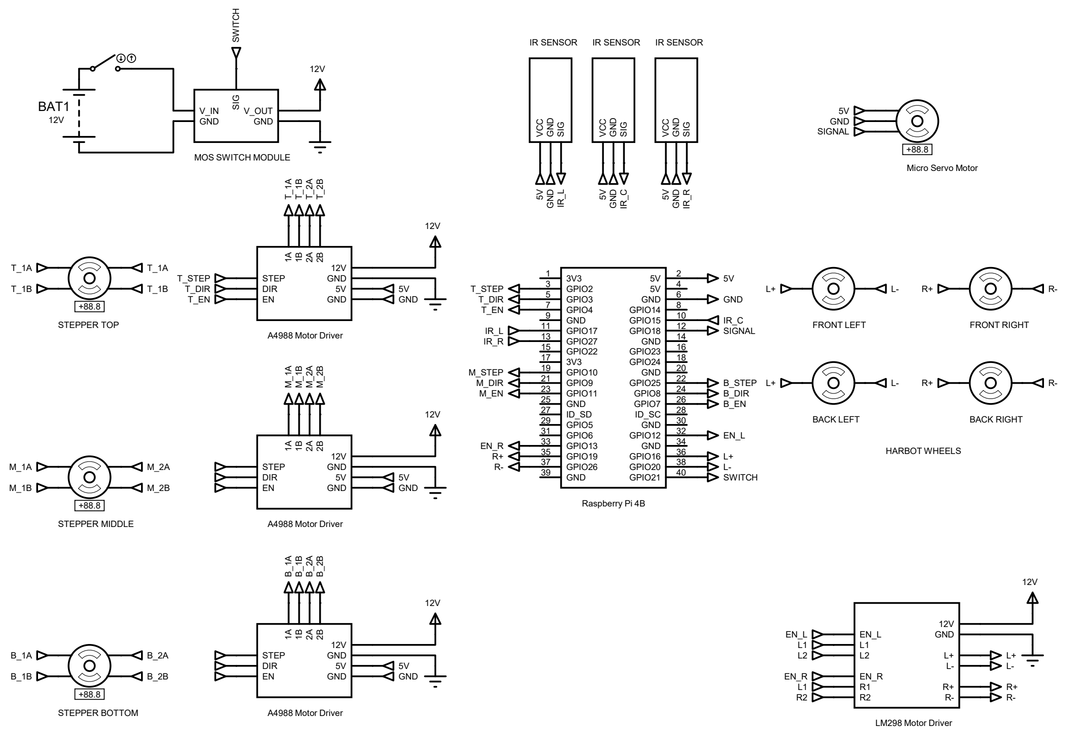


Figure 3.5: Circuit diagram

Chapter 4

REAL TIME OBJECT DETECTION

4.1 COMPUTER VISION

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

Every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point (i.e. the center) are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found.

Methods for object detection generally fall into either neural network-based or non-neural approaches. For non-neural approaches, it becomes necessary to first define features using one of the methods below, then using a technique such as support vector machine (SVM) to do the classification. On the other hand, neural techniques are able to do end-to-end object detection without specifically defining features, and are typically based on convolutional neural networks (CNN). Neural network approaches includes Region Proposals, Single Shot MultiBox Detector(SSD), You Only Look Once (YOLO) & Retina-Net.

4.1.1 Ripe Fruit Detection

Humans depend on their vision quality to check whether the fruit is ripe or unripe. They grade the maturity level of a fruit based on their vision based features that lead to inaccuracy, inconsistency and inefficiency in the results. There are numerous methods to detect the ripeness of fruits. For detecting ripe tomatoes we have used Mobilenet SSD, a CNN based object detection model due to its high speed and good accuracy. SSD algorithm has gained popularity because of its superior performance over the aforementioned object detection techniques.

4.1.2 Mobilenet SSD

SSD (Single Shot MultiBox Detector) is a popular algorithm in object detection. It's generally faster than Faster RCNN. The MobileNet network architecture is a special class of convolutional neural models that are built using depth-wise separable convolutions and are therefore more lightweight in terms of their parameter count and computational complexity.

- Speed: This algorithm improves the speed of detection because it can predict objects in real-time.
- High accuracy: Mobilenet SSD is a predictive technique that provides accurate results with minimal background errors.
- Learning capabilities: The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection.

4.1.3 Algorithm

For detecting ripe tomatoes, we assembled a dataset and trained a custom Mobilenet SSD v3 model to recognize the objects in our dataset. To do so we will take the following steps:

1. Create a Dataset: Gather a dataset of images and label our dataset: In order to train our custom model, we need to assemble a dataset of representative images with bounding box annotations around the objects that we want to detect. And we need our dataset to be in Mobilenet SSD v3 format.
2. Select a Model: Select a pretrained model to start training from.
3. Train: Train the model on custom Dataset by specifying dataset, batch-size, image size and either pretrained, or randomly initialized.
4. Run test inference to view our model at work

4.2 CENTROID ESTIMATION

Along with real-time object detection, image localization is also performed by the SSD model. Here we locate the exact position of the ripe tomatoes by drawing a bounding box around it (Bounding Box Regression). The SSD model produces an output vector consisting of the following:

- The confidence score: Confidence score represents the presence of an object in the bounding box. It is a probability of prediction i.e how confident the model is in predicting.
- Bounding Box Coordinates: It consists of the coordinates of the bounding box i.e x_{max}, x_{min}, y_{max} & y_{max}

The centroid of the bounding box (c_x, c_y) can be found using the equation:

$$c_x = \frac{x_{\text{min}} + x_{\text{max}}}{2} \tag{4.1}$$

$$c_y = \frac{y_{\text{min}} + y_{\text{max}}}{2} \tag{4.2}$$

4.3 DEPTH ESTIMATION

Distance measurement between a robot and the object is needed to control the action of the robot such as grabbing an object or even avoiding obstacles. There are many methods to estimate the distance such as ultrasonic ranging, laser ranging, and vision based ranging. Vision based techniques have the merit of its low cost. To estimate distance of ripe tomatoes using camera, we use Triangle similarity algorithm to estimate the depth of the tomatoes from the camera.
The camera generates a one-to-one relationship between the object and the image. Using this principle, we can deduce a relationship between known parameters: focal length (f), width of tomato in the image plane (r), and width of tomato in the object plane (R) and unknown parameter distance from the camera to the object(d).

Using the principle of Similarity of Triangles, we can obtain the formulas as follows:

$$\frac{f}{d} = \frac{r}{R} \tag{4.3}$$

$$f = d * \frac{r}{R} \tag{4.4}$$

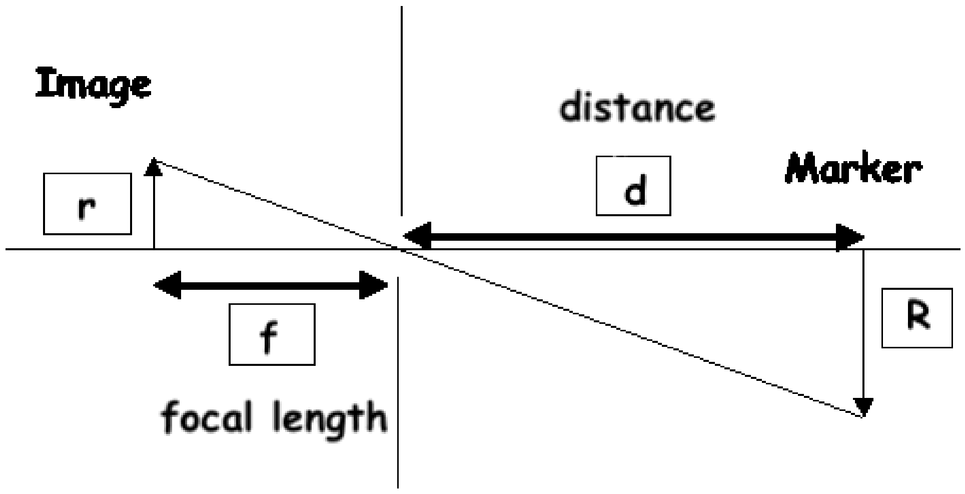


Figure 4.1: Principle of the method

$$d = f * \frac{R}{r} \quad cm \tag{4.5}$$

OpenCV can be used to estimate the focal length after taking some images of ripe tomatoes with camera, the result will be intrinsic parameters: focal length and optical center and extrinsic parameters: rotation and translation vectors of the camera. The focal length is estimated using the equation (5.4) and the values is obtained in terms of pixels. The depth of the object is found using the equation (5.5) obtained in cm.

Chapter 5

ROBOTIC ARM MANIPULATOR

5.1 INTRODUCTION

A robotic arm manipulator is a particular kind of robot made to move items or things around in a particular setting, such a manufacturing floor or warehouse. It typically consists of a number of joints and linkages coupled by motors, sensors, and other electronics, allowing it to precisely move and control items. These robots are frequently employed in production and assembly processes as well as in other industries where activities must be carried out repeatedly and accurately. They can be programmed to carry out a wide range of functions, from straightforward pick-and-place procedures to more intricate assembly processes.

5.2 INVERSE KINEMATICS

Robotics and computer graphics employ the technique of inverse kinematics to determine the joint angles required for a robotic arm to achieve a particular position. It is the opposite of forward kinematics, which determines the robotic arm's final position based on the joint angles. The joint angles required to move the robotic arm's end effector to a specific point, which can be described in terms of Cartesian coordinates (x, y, and z), or polar coordinates, are calculated using inverse kinematics (distance from origin, angle of inclination, angle of rotation). For a wide variety of robotic systems, such as manipulators, legged robots, and humanoid robots, inverse kinematics techniques can be used to calculate the joint angles.

5.2.1 Inverse Kinematics in two degree of freedom

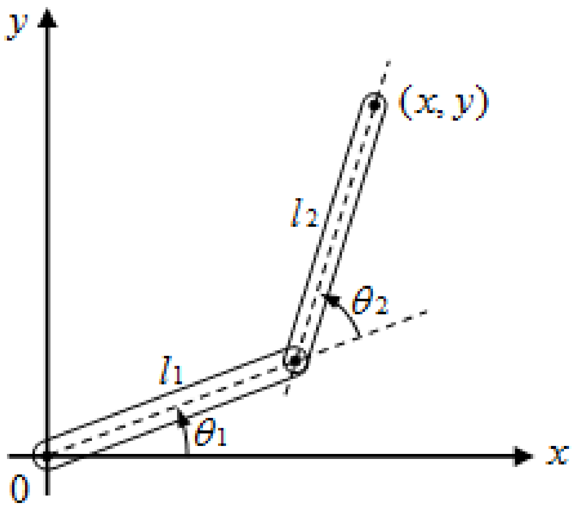


Figure 5.1: Figure showing 2DOF arm

A two degree of freedom (DOF) robotic arm's inverse kinematics can be determined using straightforward geometrical techniques. The two DOF arm is made up of two joints, each of which has one degree of rotational flexibility. In terms of its Cartesian coordinates, the end effector's position (the point at the end of the arm) can be characterised (x, y).

5.2.2 Inverse kinematics calculations for two degree of freedom

To calculate the joint angles necessary to reach a desired end effector position (x, y), we can use the following steps:

1. Calculate the distance d between the arm's base and end effector (the point where the arm is attached to the body of the robot). The Pythagorean theorem can be used to determine this distance: d = sqrt(x2 + y2).

2. The angle theta1 between the x-axis and the line from the arm's base to the

end effector should be calculated. The inverse tangent function can be used to determine this angle: theta1 = atan2 (y, x).

3. Determine the angle theta2 between the arm and the line linking its base to its end effector. The law of cosines can be used to determine this angle: theta2 = acos((l1 + d - l2 2) / (2 * l1 * d)). The two links of the arm's length, l1 and l2, are given in this instance.

The joint angles theta1 and theta2 can then be used to position the arm to reach the desired end effector position (x, y).

Let theta1=q1,theta2=q2, l1=a1,l2=a2

$$cos(q_2) = \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1a_2}$$

$$q_2 = cos^{-1}(\frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1a_2})$$

$$q_1 = tan^{-1}(\frac{y}{x}) - tan^{-1}\frac{a_2 sin(q_2)}{a_1 + a_2 cos(q_2)}$$

5.2.3 Implementation of Inverse kinematics

The numerical computing environment and programming language MATLAB (short for ”Matrix Laboratory”) were created and developed by MathWorks. It is widely used in a variety of industries, such as engineering, science, economics, and finance, and is especially well-suited for using matrices, arrays, and mathematical functions. MATLAB's robust calculator-style syntax, integrated graphical capabilities, and extensive library of functions and toolboxes for fields like signal processing, optimization, and statistics are just a few of its standout features.

The method of determining the joint angles required for a robot to attain a specific position is known as inverse kinematics.MATLAB is a very useful tool which help us to simulate the invesre kinematics algorithms virtually. This helps the designers to look deep into working of proposed model without actually building it. It is a significant robotics issue that is employed in activities like path planning and manipulation. Use the ”ikine” function from the Robotics Toolbox in MATLAB to solve the inverse kinematics problem. The joint angles associated with a given end-effector position are determined by this function using an iterative numerical approach. The joint angle solution is given as a column vector by the ”ikine” function. In the event that there are numerous solutions, the function returns each one in a matrix, with each column denoting a different answer. The ”ikcon” function from the Robotics Toolbox can also be used to tackle the inverse kinematics issue with restrictions on the joint angles. This can be helpful, for instance, if you wish to prevent robot configurations that might make it unique or force it to crash with its surroundings.

Chapter 6

LINE FOLLOWER

6.1 INTRODUCTION

This chapter discusses the details of the methodology adopted for the incorporation of line follower robot in HARBOT for guiding it through the garden.

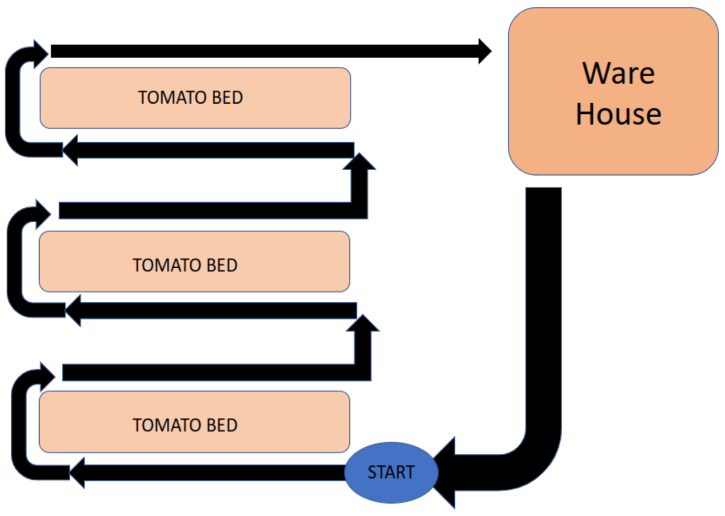


Figure 6.1: Figure showing path of line follower in the garden

The line fallowing robot is one of the self-operating robots. That detects and fallows a line drawn on the area. The line is indicated by white line on a block surface or block line on a white surface. This system must be sense by the line. This application is depends upon the sensors. Here we are using two sensors for path detection purpose. That is proximity sensor and IR sensor. The proximity sensor used for path detection and IR sensor used for obstacle detection. These sensors mounted at front end of the robot. The microcontroller is an intelligent device the whole circuit is controlled by the microcontroller. In our project the raspberry pi act as the brain of the line follower.

6.2 WORKING PRINCIPLE

We use the behaviour of light on the black and white surface. The white colour reflects all the light that falls on it, whereas the black colour absorbs the light. In this line follower robot, we use IR transmitters and receivers (photodiodes). They are used to send and receive the lights. When IR rays fall on a white surface, it is reflected towards IR receiver, generating some voltage changes. When IR rays fall on a black surface, it is absorbed by the black surface, and no rays are reflected; thus, the IR receiver doesn't receive any rays. In this project, when the IR sensor senses a white surface, an Raspberry Pi gets 1 (HIGH) as input, and when it senses a black line, an Raspberry Pi gets 0 (LOW) as input. Based on these inputs, an Raspberry Pi provides the proper output to motor drivers according to program logic. We use 3 IR sensors in HARBOT. The program configuration logic is shown in the figure.

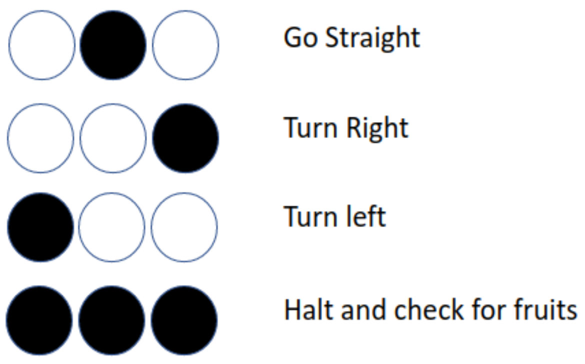


Figure 6.2: Figure showing logic of line follower

The circles on left indicates the output of photodiodes. White circles indicates the output is high that is a white line is detected and the black circle indicates a black is observed by the photodiode. Combinations of output that are valid along with the program logic are shown in the figure. Any other combinations of output from photodiodes will lead to stopping of HARBOT. HARBOT is meant to stop when detecting black colour on three of the photodiodes since we place markers near plants on the line of the line follower to indicate the presence of a plant.

Chapter 7

REMOTE ACCESS

7.1 INTRODUCTION

This chapter discusses the details of the methodology adopted for the implementation of remote access of HARBOT using an Android application. The application for remote access feature is developed for android OS using Kodular Creator. It contains user interface for displaying information and controlling HARBOT. The application was connected with a remote database provided by google firebase. The firebase will act as a relay between the android application and Raspberry pi board embedded in HARBOT.

7.2 APPLICATION DEVELOPMENT

7.2.1 Kodular Creator

The main component of the online suite Kodular is an online web application that enables the creation of apps without the need for coding knowledge. Kodular Creator is an MIT App Inventor distribution, which means it is based on the open source App Inventor project even though it has many advantages over App Inventor. Kodular Creator, however, is not an open source project, and its source code is kept on GitHub in a secure environment. The apps in Kodular are built as a combination of various Components , with each individual Component being used for a specific purpose. One component may be used to design the User Interface(UI) of the app, for example a Button component, while others may be used for performing actions like communicating to a database, saving an image to the Android device's folder etc. for example the FirebaseDB component. Blocks are ones which are used describe how to do a task. The way in which the Components respond to various actions and events in the app is designed using the Blocks. The Component's behaviour is configured using Blocks

7.2.2 Fire base database component

Firebase Database component is a non-visible component that communicates with a Firebase to store and retrieve information. This component helps the application to connect and manipulate firebase database. The component requires firebase database URL and the web API key provided by Firebase. API key ensures the security of stored data, such that Firebase cannot be acessed without these combinations.

7.3 GOOGLE FIREBASE

Firebase is a set of hosting services for any type of application (Android, iOS, Javascript, Node.js, Java, Unity, PHP, C++ ...). It offers NoSQL and real-time hosting of databases, content, social authentication (Google, Facebook, Twitter and Github), and notifications, or services, such as a real-time communication server. Firebase's first product was the Firebase Realtime Database, an API that synchronizes application data across iOS, Android, and Web devices, and stores it on Firebase's cloud. The product assists software developers in building real-time, collaborative applications.

7.4 WORKING PRINCIPLE

We have used Pyrebase package for interfacing Raspberry Pi with the Firebase. Pyrebase is a simple python wrapper for the Firebase API.

1. Authentication

   In order to ensure security and support multiple users, the app prompts the user to enter a username and password during the sign-in process. These credentials are then compared to the ones stored in the Firebase server. If a matching set of credentials is found, the app proceeds to allocate a personalized space within Firebase for the HARBOT user.

   By verifying the credentials against the stored data, the app ensures that only authorized users can access their individual data. This approach adds a layer of security to protect sensitive information and prevent unauthorized access.

   Once the user clicks the sign-in button, the app communicates with the Firebase server to validate the provided credentials. If the username and password combination is a match, the app establishes a connection with the user's allocated databucket within Firebase.

   The databucket serves as a dedicated storage space for the HARBOT user, where they can store, retrieve, and manage their data securely. This personalized space allows users to have their own isolated environment within the larger Firebase ecosystem.

   With this setup, each user can access their own databucket and interact with their data without interfering with other users' information. This personalized allocation ensures privacy and data integrity, as users can only access and modify their own data within Firebase.

   Overall, this process of username and password verification, coupled with personalized databuckets in Firebase, promotes enhanced security and tailored experiences for the HARBOT app users.

2. Turning ON HARBOT using App

   After the user successfully logs in from the login screen, the databucket in Firebase is updated to correspond to that specific user. Within the databucket, several variables are set to facilitate the functionality of the HARBOT app.

   One variable stored in the databucket is used to track the current status of the Raspberry Pi. When the Raspberry Pi is turned off or the program is not running, the app displays a ”Power OFF” message on the screen. In this state, the user is unable to toggle the status of HARBOT. This feature ensures that the user is aware of the Raspberry Pi’s status and prevents them from attempting to control HARBOT when it is not operational.

   Additionally, the databucket contains a variable for storing the current request from the app. This variable is updated as the user interacts with the application, specifically by clicking on the status card. The user can toggle the status of HARBOT between sleep mode and harvest mode, or vice versa, by modifying the request variable in Firebase.

   The Raspberry Pi constantly monitors the app request status within the databucket. Whenever there is a change in the app request status, the Raspberry Pi responds accordingly by switching the MOSFET switch module on or off, based on the received request. This mechanism ensures that the Raspberry Pi is synchronized with the user’s actions in the app and enables the desired mode for HARBOT.

   By updating the request variable in Firebase and having the Raspberry Pi monitor and respond to these changes, the app provides real-time control over HARBOT’s status. This integration between the app and the Raspberry Pi allows for seamless interaction and ensures that the robot operates as per the user's preferences.

Overall, this system architecture enables users to manage HARBOT's status effectively and ensures that the Raspberry Pi and app remain synchronized in controlling the robot's operational mode.

3. Features in the App

   (a) Secure Login and Access to user Space

   (b) Number of Fruits harvested by HARBOT can be monitored through the App

   (c) The user can view and toggle the status of the HARBOT directly through the App. That is the user can toggle HARBOT from sleep mode to Harvest mode or vice versa

   (d) The user will be able to automate the robot to harvest in particular duration of the day or week

   (e) The user can signout and clear app caches through app settings

   (f) The User can easily enable or disable the biometric login through app settings menu

   (g) App updates can be checked at our github page

Chapter 8

HARDWARE COMPONENTS

8.1 LITHIUM POLYMER BATTERY

Figure 8.1: Flipo 3300mAh 3S 40C/80C (11.1V) Lithium Polymer Battery

The Flipo 3S 40C/80C Lithium polymer 3300mah battery Pack (LiPo) is well-known for its performance, dependability, and low cost. So it comes as no surprise that the Flipo lipo battery is beneficial in drones or other multirotor systems, as well as health and fitness equipment. The 3300mAh battery Pack (LiPo) provides full capacity at a price that everyone can afford, and we guarantee a high-quality product and excellent customer service. Above all, the Flipo 3S 40C/80C

3300mAh battery Pack (LiPo) comes with heavy-duty discharge leads, which minimise resistance and allow for high current loads. Flipo batteries can withstand the severe extremes of aerobatic flying and remote-controlled vehicles. Each bundle includes gold-plated connections and JST-XH type balance connectors. In order to guarantee excellent dependability, all Flipo Lithium Polymer battery packs are assembled utilising IR match cells.

8.2 STEPPER MOTOR

NEMA17 Stepper Motors are frequently found in CNC machines, hard drives, and linear actuators. The motor has six lead wires and a rated voltage of twelve volts. It may be run at a lower voltage, but the torque will be reduced. These motors have a step angle of 1.8 deg., which means that it has 200 steps each revolution and each step covers a 1.8°, resulting in a high degree of control. These motors operate on 12V and so have a high torque. So, if you're searching for a small, easy-to-use stepper motor with a lot of torque, this is the one for you.

Figure 8.2: nema 17 stepper motor

8.3 SERVO MOTOR

A micro servo motor, also known as an SG90, is a small, light-weight server motor with a high output power. The servo revolves approximately 180 degrees (90 degrees in each direction) and acts similarly to bigger servos. Any servo code, hardware, or library may control these servos. Because it may fit in small locations, it is ideal for beginners who do not wish to build a motor controller with feedback and a gear box. PWM is a way of rotating servo motors. PWM generates analogue signals using digital input devices such as microcontrollers.The resultant signal will be a succession of square wave pulses.This implies that the wave might be high or low at any time.

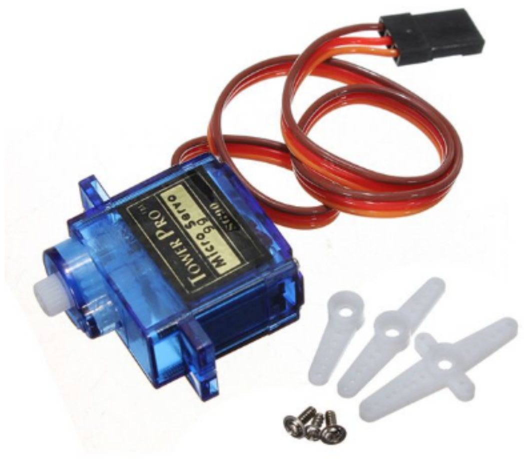


Figure 8.3: servo motor

8.4 GEAR MOTOR

DC Motor - 60RPM - 12Volts Geared motors are typically made up of a basic DC motor and a gearbox. This may be employed in all-terrain robots as well as a wide range of robotic applications. These motors include a 3 mm threaded drill hole in the centre of the shaft, making it easy to attach them to wheels or other mechanical assemblies. 60 RPM 12V DC geared motors are frequently used in robotics applications. Very simple to use and comes in typical sizes. The most popular L298N H-bridge module with onboard voltage regulator motor driver can be used with this motor, which has a voltage range of 5 to 35V DC, or you can select the most precise motor diver module from our Motor divers category based on your specific needs. The shaft has a nut and threads for easy connection, as well as an internally threaded shaft for easy connection to the wheel.DC Geared motors with a strong metal gearbox for heavy-duty applications, available in a wide RPM range, and perfect for robots and industrial applications.Very simple to use and comes in typical sizes. The shaft has a nut and threads for easy connection, as well as an internally threaded shaft for easy connection to the wheel.

Figure 8.4: gear motor

8.5 IR SENSOR

The IR Sensor module has a great adaptive capability of the ambient light, having a pair of infrared transmitter and receiver tubes, the infrared emitting tube to emit a certain frequency, encounters an obstacle detection direction (reflecting

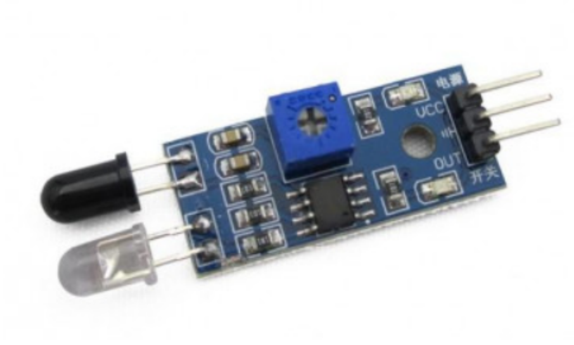


Figure 8.5: IR sensor module

surface), infrared reflected back to the receiver tube receiving, after a comparator circuit processing, the green LED lights up, while the signal output will output digital signal (a low-level signal), through the potentiometer knob. The detection range of the sensor can be modified by the potentiometer, with minimum interference, easy to construct and easy to use features, and it can be extensively used for robot obstacle avoidance, obstacle avoidance vehicle assembly line count, and black-and-white line tracking, among other things.

8.6 A4988 STEPPER MOTOR DRIVER

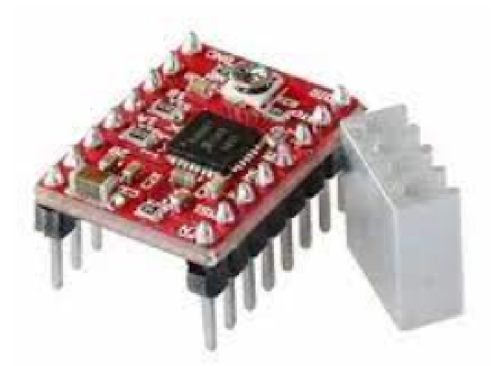


Figure 8.6: a4988 motor driver

The A4988 Stepper Motor Driver is a comprehensive micro-stepping motor driver with a built-in converter that is simple to use. It runs between 8 and 35 volts and can provide up to 1 amp per phase without a heat sink or forced air flow (it is rated for 2 amps per coil with adequate extra cooling). The A4988 Stepper Motor Driver contains a fixed off-time current regulator that can operate in slow or mixed decay mode. The converter is critical to the simple implementation of the A4988. There are no phase sequence tables, high-frequency control interface programming, or anything else. The use of the A4988 interface is ideal for when a complicated microprocessor is unavailable or overloaded. The A4988's chopping control automatically picks the current decay mode (slow or mixed) throughout the stepping process. The mix decay current control approach reduces audible motor noise while increasing step precision and lowering power consumption.

8.7 L298 DC MOTOR DRIVER

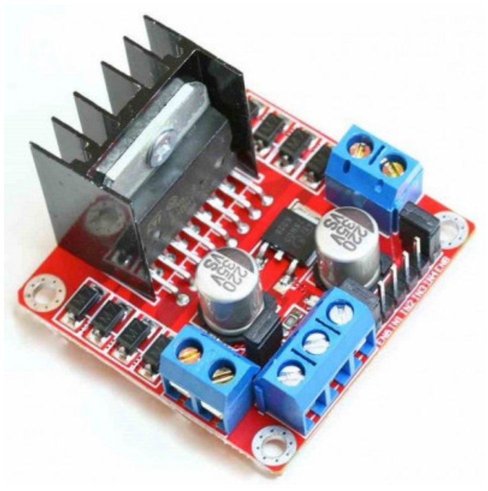


Figure 8.7: L298 DC motor driver

This L298-based Motor Driver Module is a high-power motor driver that is ideal for operating DC and stepper motors. It employs the well-known L298 motor driver IC and features an integrated 5V regulator that can give power to an external circuit. It has the ability to control up to four DC motors or two DC motors with directional and speed control. This motor driver is ideal for robotics and mechatronics applications, as well as controlling motors from microcontrollers, switches, relays, and other devices. Ideal for controlling DC and Stepper motors in micro mice, line following robots, robot arms and other applications.An H-Bridge is a circuit that can drive current in either polarity and is controlled by PWM. The duration of an electrical pulse may be controlled via pulse width modulation. Consider the brush of a motor to be a water wheel, and electrons to be streaming drops of water. The voltage would be the continual flow of water over the wheel; the more water flowing, the greater the voltage. Motors are rated at specific voltages and can be damaged if the voltage is applied too strongly or abruptly lowered to slow the motor down. As a result, PWM. Consider the water wheel analogy: imagine water striking it in pulses but at a steady flow. The longer the pulses, the quicker the wheel turns; the shorter the pulses, the slower the wheel turns. PWM-controlled motors will last considerably longer and be more dependable.

8.8 RASPBERRY PI 4B

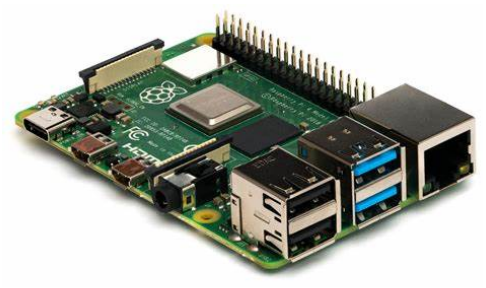


Figure 8.8: Raspberry Pi 4B

The Raspberry Pi 4 Model B is the most recent addition to the popular Raspberry Pi computer line. It outperforms the previous-generation Raspberry Pi 3 Model B+ in terms of CPU speed, multimedia performance, memory, and connection while maintaining backwards compatibility and power consumption. The Raspberry Pi 4 Model B offers desktop performance equivalent to entry-level x86 PC systems to the end user. A high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI ports, hardware video decode at up to 4Kp60, 4GB of RAM, dual-band 2.4/5.0GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on) are among the key features of this product. The dual-band wireless LAN and Bluetooth modules have modular compliance certification, which allows the board to be incorporated into end devices with much reduced compliance testing, reducing both cost and time to market.

8.9 PI CAMERA

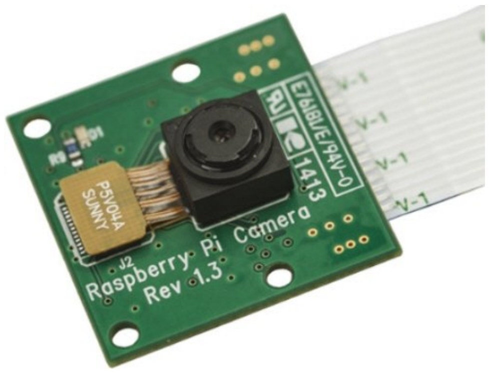


Figure 8.9: Pi Camera

5 megapixel native resolution sensor-capable of 2592 x 1944 pixel static images supporting 1080p30, 720p60 and 640x480p60/90 video. Camera is supported in the latest version of Raspbian, Raspberry Pi's preferred operating system. The camera is interfaced with the Raspberry Pi via FCC cable.

8.10 RADIAL BEARINGS

A radial bearing is a form of rolling-element bearing that sustains loads that are perpendicular to the rotation axis. A deep-groove ball bearing, or simply a ball bearing, is another name for it. Radial bearings are normally made up of an inner and outer ring, as well as a number of balls kept in place by a cage. The inner and outer rings are often constructed of steel, whereas the balls might be made of

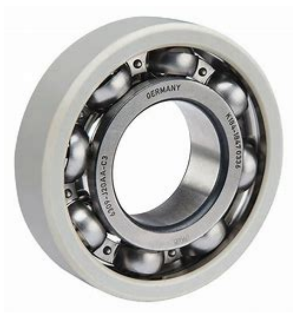


Figure 8.10: Radial Bearings

steel, ceramic, or plastic. The balls roll along the raceways generated on the rings as the inner and outer rings are rotated relative to each other, minimising friction and permitting smooth and efficient rotation.Radial bearings are widely utilised in a variety of applications such as electric motors, automobile engines, industrial gear, and home appliances. They are reasonably priced, simple to maintain, and have a long service life.

8.11 THRUST BEARINGS

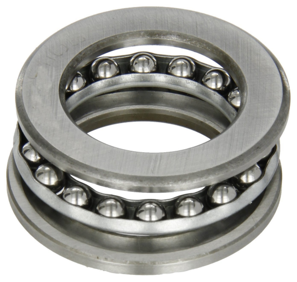


Figure 8.11: Thrust Bearings

An axial load, or force acting parallel to the axis of rotation, is what a thrust bearing is intended to handle. Thrust bearings are particularly intended to manage the sorts of forces that occur when an item is pushed or dragged along its axis, in contrast to other types of bearings like radial bearings, which are built to handle radial loads. Typically, thrust bearings are made up of two bearing plates, or ”washers,” and a number of rolling components, such as balls or rollers, are positioned in the space in between. The two washers may move in relation to one another with little resistance thanks to the rolling components, which aids in distributing the axial load uniformly throughout the bearing surface.

8.12 LINEAR BEARINGS

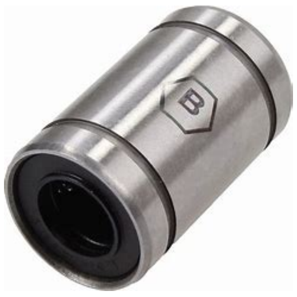


Figure 8.12: Linear Bearings

Linear bearings are mechanical components used to allow linear motion, or motion along a straight line, in machinery and equipment. They are designed to support and guide a moving part, such as a linear shaft or rod, with minimal friction and wear.

Chapter 9

TESTS AND RESULTS

9.1 REAL TIME OBJECT DETECTION

9.1.1 Software simulation

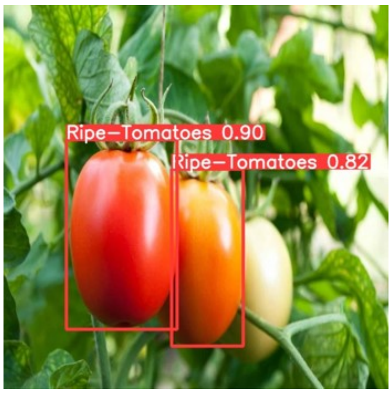
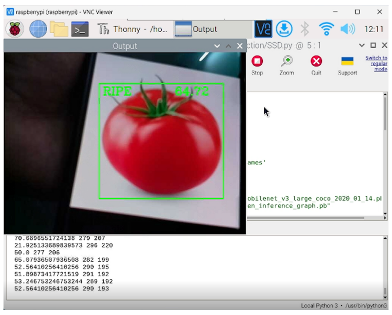


Figure 9.1: Object Detection

9.1.2 Hardware Simulation



```
ames'
obilenet_v3_large_coco_2020_01_14.pl
en_inference_graph.pb"
```

```
70.6896551724138 279 207
21.925133689839573 296 220
50.0 277 206
65.07936507936508 282 199
52.56410256410256 290 195
51.89873417721519 291 192
53.246753246753244 289 192
52.56410256410256 290 193
```

Figure 9.2: Object Detection using Raspberry Pi

9.2 ROBOTIC ARM MANIPULATOR

9.2.1 Software simulation

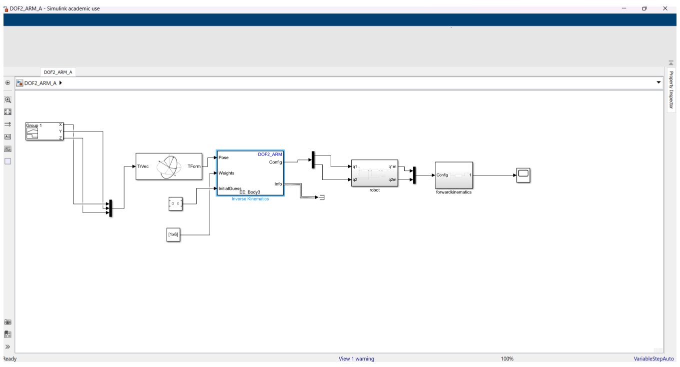


Figure 9.3: Simulink model for IK simulation

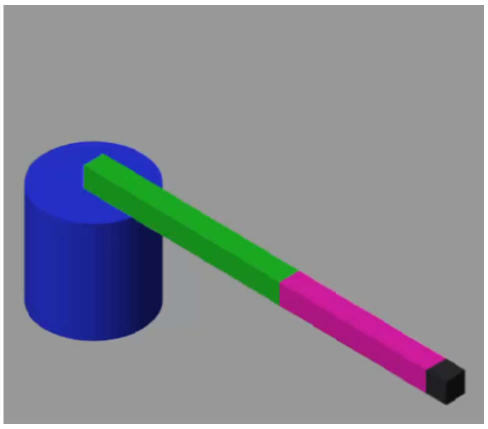
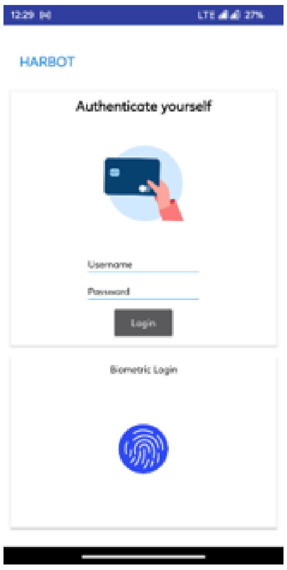


Figure 9.4: Simscape scara model

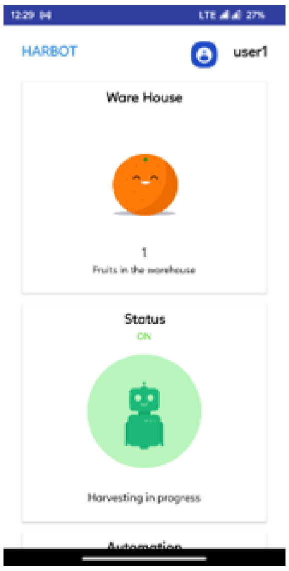
9.2.2 Hardware simulation

In order to test the hardware, three stepper motors are rotated to the desired angle using inverse kinematics. Two stepper motors, one in the middle and one in the bottom, are rotated by integrating with the pi camera using inverse kinematics. The stepper at the top is spin at certain stages at regular intervals to determine whether or not the fruit is ripe.

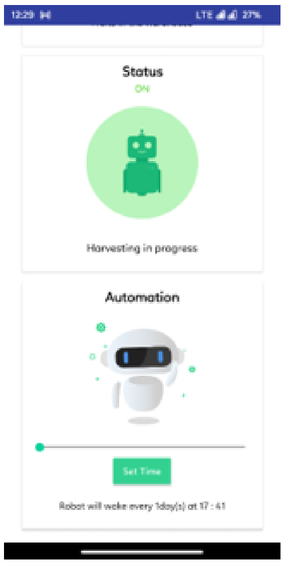
9.3 MOBILE APPLICATION INTERFACE



(a) Login Screen

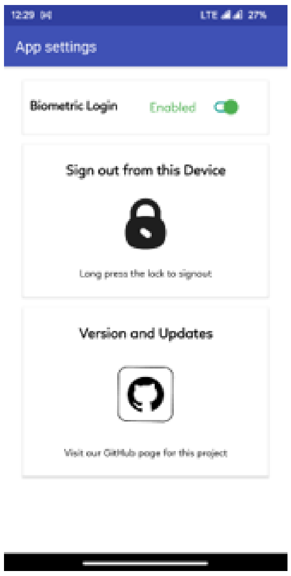


(b) Home Screen



Robot will wake every 1day(s) at 17 : 41

(c) Home Screen



Long press the lock to signout

Visit our GitHub page for this project

(d) Settings

Figure 9.5: Application Interface

9.4 LINE FOLLOWER

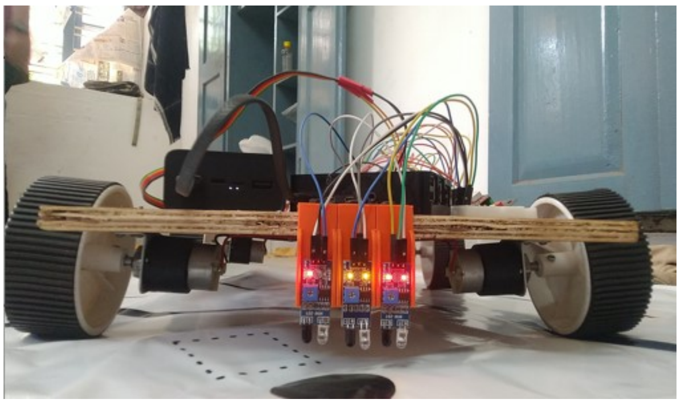


Figure 9.6: Line Follower front view

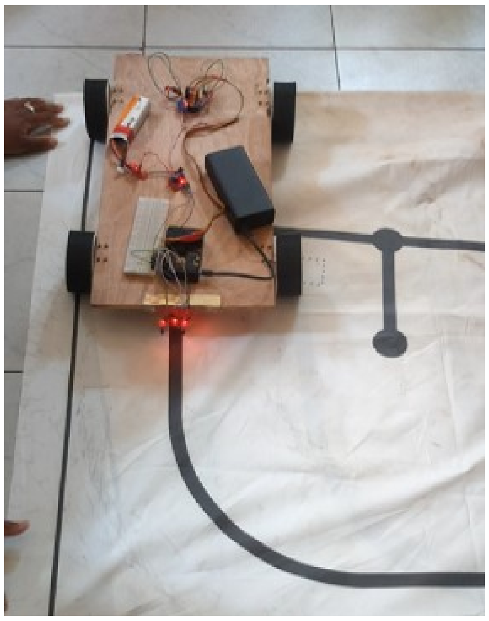


Figure 9.7: Line Follower

9.5 OVERALL RESULT

9.5.1 Circuit designed

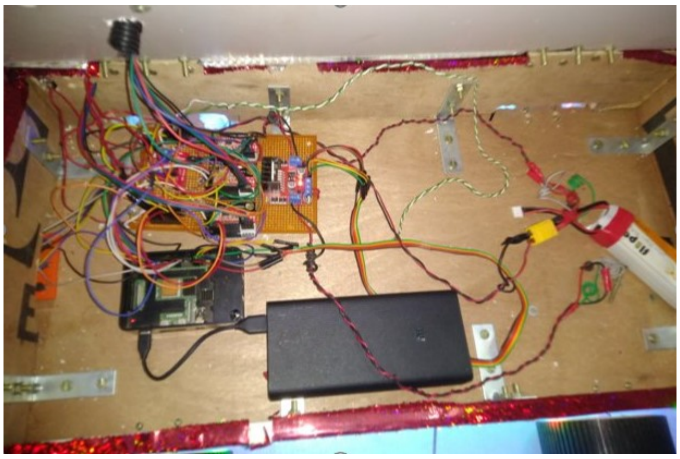


Figure 9.8: PCB Circuit

9.5.2 Gripper Mechanism

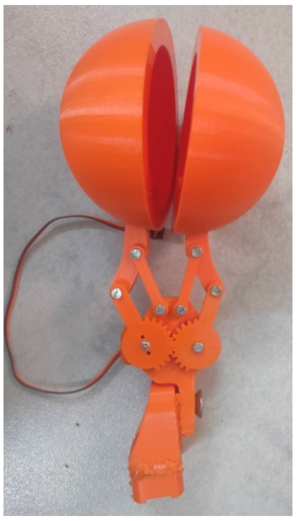


Figure 9.9: Designed Gripper

9.5.3 Final prototype

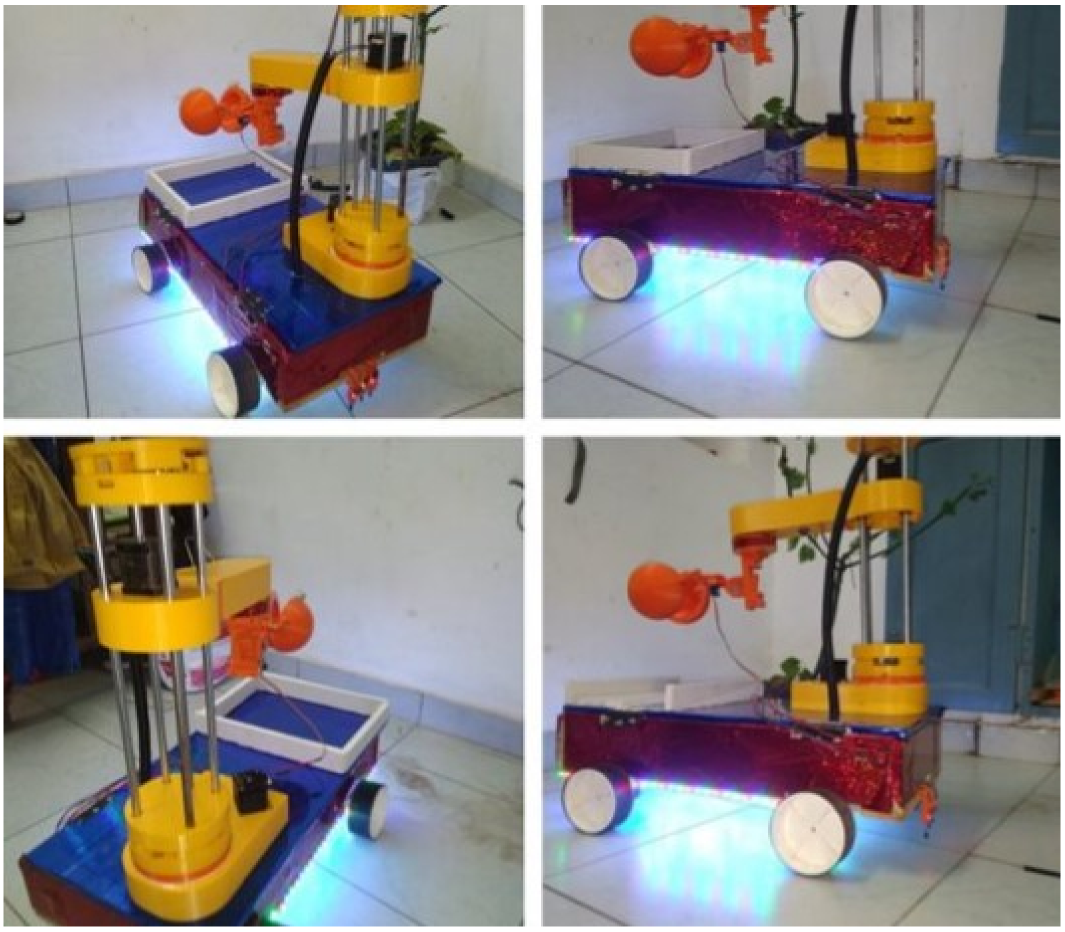


Figure 9.10: Final Prototype

Chapter 10

PROJECT SCHEDULING & DISTRIBUTION OF WORK

10.1 SCHEDULE

- **September :** Finalized idea for project, Detailed Literature Survey and analysis of existing market technologies.
- **October :** Analysed performance of various object detection models and selected YOLOv5 model for object detection. Created a custom dataset and performed object detection on dataset. Designed robotic arm with 2 DOF in Fusion 360. Designed circuit for Line follower.
- **November :** Simulated the robotic arm using python. Prepared an animated video of arm simulation. Performed Real-time object detection using YOLOv5. Developed a mobile application for remotely accessing the HARBOT.
- **December :** Estimated centroid and depth of the detected tomatoes. Familiarized with Raspberry pi. Turned on a LED using mobile app using Firebase. Tested the stepper motors using arduino uno.Started work of prototype of proposed model for line follower.
- **January :** Interfaced Stepper motors, raspberry pi camera and microservo motor with raspberry pi. Assembled the robotic arm. Deployed object detection model on raspberry pi.

- **February :** Completed arm assembly. Performed object detection using Mobilenet SSD on raspberry pi.
- **March :** Designed and developed circuit for overal functionality, Developed mobile application for controlling robot.
- **April :** Integrated all modules to build up final version. Tested at each stage for obtaining desired output.
- **May :** Tested and corrected errors of overall functionality, Performed decoration works to improve aesthetics of robot.

10.2 WORK DISTRIBUTION

1. **Abhishek Babu T (TCR19EC004)**

   Realtime Object detection, Centroid & Depth Estimation.

2. **Ashik M M (TCR19EC019)**

   Line Follower, Mobile app & Remote access.

3. **Karthik S (TCR19EC035)**

   Line Follower & Circuit Design.

4. **Sivaprasad A S (TCR19EC051)**

   Harbot Design & Simulation,Scara arm & Inversre kinematics.

Chapter 11

CONCLUSION AND FUTURE WORK

11.1 CONCLUSION

In summary, HAR-BOT have streamlined and improved the efficiency of the harvesting process, revolutionising the agricultural sector. The amount of physical labour needed to harvest crops has been greatly decreased thanks to these sophisticated tools, increasing output and lowering costs for farmers. Robots may labour around the clock by automating the harvesting process, increasing the efficiency and precision of crop collecting.The development of HAR-BOT has also solved a number of issues affecting the agriculture industry, including labour shortages and growing labour prices. These robots relieve human employees of monotonous and physically taxing chores, allowing them to concentrate on more intricate and sophisticated agricultural jobs.Aside from that, harvesting robots have shown to be incredibly versatile and adaptive, able to harvest a variety of crops, including grains, fruits, and vegetables. These robots can recognise ripe food, handle fragile objects with care, and navigate through complicated farming conditions thanks to cutting-edge sensor technology and cognitive algorithms.Additionally, the employment of harvesting robots has advanced sustainable agricultural methods. These robotic harvesters save waste by streamlining the harvesting procedure. They may also gather information on crop health and productivity, allowing farmers to choose the right amounts of irrigation, fertiliser, and pesticides.Although harvesting robots have many advantages, there are still certain difficulties to be solved. There is still room for development in terms of technological constraints, such as the capacity to manage fluctuating crop conditions and adapt to different agricultural locations. Additionally, small-scale farmers may find it difficult to afford the initial outlay required to purchase and maintain harvesting robots.

In conclusion, HAR-BOT have become a transformative technology in agriculture, bringing enhanced sustainability, decreased labour needs, and higher efficiency. These robots have the potential to revolutionise crop harvesting as robotics and artificial intelligence continue to improve, boosting productivity and ensuring a more sustainable future for the agricultural sector.

11.2 FUTURE SCOPE

- Can add sophisticated gripper mechanism
- Can improve the height of device by providing proper support
- Can deliver the details of false fruits to user

REFERENCES

[1] **S. M. Mangaonkar, R. Khandelwal, S. Shaikh, S. Chandaliya and S. Ganguli**,"Fruit Harvesting Robot Using Computer Vision," 2022 International Conference for Advancement in Technology (ICONAT), 2022

[2] **Onishi, Y., Yoshida, T., Kurita, H. *et al***, An automated fruit harvesting robot by using deep learning. Robomech J 6, 13 (2019).

[3] **R. K. Megalingam, G. V. Vivek, S. Bandyopadhyay and M. J. Rahi**, "Robotic arm design, development and control for agriculture applications," 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), 2017,

[4] https://www.mathworks.com/help/robotics/inverse-kinematics.html

[5] https://robotacademy.net.au/lesson/inverse-kinematics-for-a-2-joint-robot-arm-using-geometry/

[6] **I. Benkhaled, I. Marc, and D. Lafon,** "Colour contrast detection in chromaticity diagram: A new computerized colour vision test," 2017 IEEE Western New York Image and Signal Processing Workshop (WNYISPW), Nov. 2017.

[7] **G. Chandan, A. Jain, H. Jain, and Mohana** , "Real Time Object Detection and Tracking Using Deep Learning and OpenCV," 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), Jul. 2018.

[8] **R. OommenThomas and K. Rajasekaran**, “Remote Monitoring and Control of Robotic Arm with Visual Feedback using Raspberry Pi,” International Journal of Computer Applications, vol. 92, no. 9, pp. 29–32, Apr. 2014.

[9] **S. Salvi, V. Sahu, R. Kalkundre, and O. Malwade**, “Autonomous Tomato Harvester Using Robotic Arm and Computer Vision,” Intelligent Communication, Control, and Devices, pp. 75–90, 2021.

[10] **F. Taqi, F. Al-Langawi, H. Abdulraheem, and M. El-Abd** , “A cherry tomato harvesting robot,” 2017 18th International Conference on Advanced Robotics (ICAR), Jul. 2017.

[11] **R. Yenorkar and U. M. Chaskar**, “GUI Based Pick and Place Robotic Arm for Multipurpose Industrial Applications,” 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Jun. 2018.

[12] **X. Liu, B. Dai, and H. He** , “Real-time object segmentation for visual object detection in dynamic scenes,” 2011 International Conference of Soft Computing and Pattern Recognition (SoCPaR), Oct. 2011.

[13] **B. Gupta, A. Chaube, A. Negi, and U. Goel**, “Study on Object Detection using Open CV - Python,” International Journal of Computer Applications, vol. 162, no. 8, pp. 17–21, Mar. 2017.

[14] **Krutika a Veerapur, Ganesh V. Bhat.** , Colour Object Tracking on Embedded Platform Using Open CV

[15] **G. Chandan, A. Jain, H. Jain, and Mohana**, “Real Time Object Detection and Tracking Using Deep Learning and OpenCV,” 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), Jul. 2018.

[16] **A. Raghunandan, Mohana, P. Raghav, and H. V. R. Aradhya**, “Object Detection Algorithms for Video Surveillance Applications,” 2018 International Conference on Communication and Signal Processing (ICCSP), Apr. 2018.

[17] **C. H. Low, M. K. Lee, and S. W. Khor**, “Frame Based Object Detection–An Application for Traffic Monitoring,” 2010 Second International Conference on Computer Research and Development, 2010.

APPENDIX

.1 MAIN

```
from Line_Follower import *
import RPi.GPIO as IO
IO.setmode(IO.BCM)
import signal

###############################

IO.setup(IR_Left, IO.IN)
IO.setup(IR_Center, IO.IN)
IO.setup(IR_Right,IO.IN)
IO.setup(en_l,IO.OUT)
IO.setup(en_r,IO.OUT)
IO.setup(M_Left1,IO.OUT)
IO.setup(M_Left2,IO.OUT)
IO.setup(M_Right1,IO.OUT)
IO.setup(M_Right2,IO.OUT)
IO.setup(switch,IO.OUT)

###############################

def keyboardInterruptHandler(signal, frame):
    print("KeyboardInterrupt␣(ID:␣{})␣has␣been␣caught.
␣␣␣␣Cleaning␣up...".format(signal))
    FDB_set("APP_Status", "OFF")
    FDB_set("RPi_Status", "OFF")
    FDB_set("Power", "OFF")
    IO.cleanup()
    print("Exiting the program")
    exit(0)
signal.signal(signal.SIGINT, keyboardInterruptHandler)
FDB_set("Power", "ON")

while True :
    STATUS = status()
    while STATUS == "\"ON\"" or STATUS == "ON":
        control(90)
        #FDB_set("Fruits", 1) #code for uploading fruit count
        STATUS = status()
    #stop()

while False :
    forward(100)
```

.2 CONFIG

```
def config():
    config = {
      "apiKey": "AIzaSyBBeg75pP2pimr2uhY7am4R_kCD_41Lou4",
      "authDomain": "fir-iot-d86c8-default-rtdb",
      "databaseURL": "https://fir-iot-d86c8-default-rtdb.firebaseio.com/",
      "storageBucket": "HARBOT"
    }
    return config
```

.3 LINE FOLLOWER

```
import RPi.GPIO as IO
import time
import pyrebase
from config import *
from UT_main import *

IO.setwarnings(False)
IO.setmode(IO.BCM)

#IR_setup
IR_Left = 17
IR_Center = 15
IR_Right = 27
IO.setup(IR_Left, IO.IN)
IO.setup(IR_Center, IO.IN)
IO.setup(IR_Right,IO.IN)
global IR_out
global prev_Status
prev_Status = "null"
global sharp_turn
sharp_turn = "null"

#motor_setup
en_l = 12
M_Left1 = 16
M_Left2 = 20
en_r = 13
M_Right1 = 19
M_Right2 = 26
IO.setup(en_l,IO.OUT)
IO.setup(en_r,IO.OUT)
IO.setup(M_Left1,IO.OUT)
IO.setup(M_Left2,IO.OUT)
IO.setup(M_Right1,IO.OUT)
IO.setup(M_Right2,IO.OUT)
l = IO.PWM(en_l,1000)          #create PWM instance with frequency
r = IO.PWM(en_r,1000)          #create PWM instance with frequency

#FIREBASE
config = config()
firebase = pyrebase.initialize_app(config)
user = "user1"
storageBucket = "HARBOT/" + user

#Remote_switch
switch = 21
IO.setup(switch,IO.OUT)

def forward(dutycycle):
    l.start(dutycycle)                          #start PWM of required Duty Cycle
    r.start(dutycycle)                          #start PWM of required Duty Cycle
    IO.output(M_Left2,True)
    IO.output(M_Left1,False)
    IO.output(M_Right2,False)
    IO.output(M_Right1,True)
    print("FORWARD")

def left(dutycycle):
    l.start(dutycycle)                          #start PWM of required Duty Cycle
    r.start(dutycycle)                          #start PWM of required Duty Cycle
    #l.start(80)                                #start PWM of required Duty Cycle
    IO.output(M_Left1,True)
    IO.output(M_Left2,False) #False
    IO.output(M_Right1,True) #TRUE
    IO.output(M_Right2,False)
    print("Left Turn")
    harvest()
```

```
def right(dutycycle):
    l.start(dutycycle)                          #start PWM of required Duty Cycle
    r.start(dutycycle)                          #start PWM of required Duty Cycle
    IO.output(M_Left1,False)
    IO.output(M_Left2,True) # Forward
    IO.output(M_Right1,False)
    IO.output(M_Right2,True) #Reverse
    print("Right Turn")

def stop(dutycycle):
    IO.output(en_l,False)                   #start PWM of required Duty Cycle
    IO.output(en_r,False)
    IO.output(M_Left1,True)
    IO.output(M_Right1,True)
    IO.output(M_Left2,True)
    IO.output(M_Right2,True)
    harvest()

def IRread():
    global IR_out
    IR_out = str(IO.input(IR_Left)) + str(IO.input(IR_Center)) +
    str(IO.input(IR_Right))
    #01 : Frontleft 10: Right front
    print(IR_out)
    # return IR_out

def status():
    database = firebase.database()
    ProjectBucket = database.child(storageBucket)
    Status = ProjectBucket.child("APP_Status").get().val()
    #print(Status)
    global prev_Status
    if prev_Status != Status:
        if str(Status) == "OFF" or str(Status) == "\"OFF\"" :
            IO.output(en_l,False)            #start PWM of required Duty Cycle
            IO.output(en_r,False)
            FDB_set("RPi_Status", "OFF")
            print("HARBOT␣is␣now␣OFF.")
            IO.output(switch, IO.LOW)
        else:
            FDB_set("RPi_Status", "ON")
            print("HARBOT␣now␣is␣ON.")
            IO.output(switch, IO.HIGH)
    #global prev_Status
    prev_Status = Status
    return Status

def FDB_set(key, value):
    database = firebase.database()
    ProjectBucket = database.child(storageBucket)
    Status = ProjectBucket.child(key).set(value)

def control(speed):
    IRread()
    global sharp_turn
    #FORWARD
    if IR_out == "010":
        forward(speed)
    #TURN RIGHT
    elif IR_out == "001":
        right(speed)
        sharp_turn = "right"
    elif IR_out == "011":
        right(speed)
    #TURN LEFT
```

```
        elif IR_out == "100":
            left(speed)
        elif IR_out == "110" :
            left(speed)
            sharp_turn = "left"
        elif IR_out == "000":
            if sharp_turn == "right":
                right(speed)
            if sharp_turn == "left":
                left(speed)
            else:
                stop(100)
        else:
            stop(100)
IO.cleanup() #resets defined pins as input
```

.4 DETECTION MAIN

```
from ut import *
from Stepper import *
from Circle_Color_Detection import *

def harvest():
    a=0
    while a<=4800:
        red=color()
        if red==0:
            TOP('clk',800)
        if red==1:
            getObjects()
            time.sleep(2)
        a=a+800
    if a>4800:
        TOP('ant',4800)
```

.5 CIRCLE COLOR DETECTION

```
import numpy as np
import cv2
import time

def color():
    cap = cv2.VideoCapture(0)
    cap.set(3,640)
    cap.set(4,480)
    x=0
    red=0
    while x<50:
        # Capture frame-by-frame
        ret, captured_frame = cap.read()
        output_frame = captured_frame.copy()
        # Convert original image to BGR, since Lab is only available from BGR
        captured_frame_bgr = cv2.cvtColor(captured_frame, cv2.COLOR_BGRA2BGR)
        # First blur to reduce noise prior to color space conversion
        captured_frame_bgr = cv2.medianBlur(captured_frame_bgr, 3)
        # Convert to Lab color space, we only need to check one
        #channel (a-channel) for red here
        captured_frame_lab = cv2.cvtColor(captured_frame_bgr, cv2.COLOR_BGR2Lab)
        # Threshold the Lab image, keep only the red pixels
        # Possible yellow threshold: [20, 110, 170][255, 140, 215]
        # Possible blue threshold: [20, 115, 70][255, 145, 120]
        captured_frame_lab_red = cv2.inRange(captured_frame_lab,
        np.array([20, 150, 150]), np.array([190, 255, 255]))
        # Second blur to reduce more noise, easier circle detection
        captured_frame_lab_red=cv2.GaussianBlur(captured_frame_lab_red
        , (5, 5), 2, 2)
        # Use the Hough transform to detect circles in the image
        circles = cv2.HoughCircles(captured_frame_lab_red, cv2.HOUGH_GRADIENT,
        1, captured_frame_lab_red.shape[0] / 8, param1=100, param2=18,
        minRadius=5, maxRadius=60)
        # If we have extracted a circle, draw an outline
        # We only need to detect one circle here, since there will only be
        #one reference object
        if circles is not None:
            circles = np.round(circles[0, :]).astype("int")
            cv2.circle(output_frame, center=(circles[0, 0], circles[0, 1]),
            radius=circles[0, 2], color=(0, 255, 0), thickness=2)
            red=1
        # Display the resulting frame, quit with q
        cv2.imshow('frame', output_frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
        x=x+1
    cap.release()
    #cv2.destroyAllWindows()
    return red
    # When everything done, release the capture
```

.6 RIPE DETECTION

```
import cv2
from inversek import inverse
from Stepper import TOP
thres = 0.30 # Threshold to detect object
known_distance= 40
tom_width=4
```

```
l=[]
c=[]
a=0

def passval(c,l):
    l1=len(c)
    l2=len(l)
    s1=sum(c)
    s2=sum(l)
    x=int((s1/l1)*0.05+2-10)
    d=int(s2/l2+2.5)
    print(x,d)
    inverse(x,d)

classNames= []
classFile = '/home/admin/Documents/Object_Detection/coco.names'
with open(classFile,'rt') as f:
    classNames = f.read().rstrip('\n').split('\n')
configPath = "/home/admin/Documents/Object
Detection/ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt"
weightsPath = "/home/admin/Documents/Object_Detection/frozen_inference_graph.pb"
net = cv2.dnn_DetectionModel(weightsPath,configPath)
net.setInputSize(320,320)
net.setInputScale(1.0/ 127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)
p=0

def getObjects():
    cap = cv2.VideoCapture(0)
    cap.set(3,640)
    cap.set(4,480)
    k=0
```

```
success ,img = cap.read()
classIds , confs , bbox = net.detect(img, confThreshold=thres)
#print(classIds ,bbox)
if len(classIds) != 0:
    for classId , confidence ,box in zip(classIds.flatten(),confs.flatten()
    ,bbox):
        if classId == 53:
            p=1
            while k!=30:
                cx= int(box[0]+box[2]/2)
                cy= int(box[1]+box[3]/2)
                w=box[2]
                focal_length = 700
                distance = (tom_width * focal_length) / w
                cv2.rectangle(img,box,color=(0,255,0),thickness=2)
                cv2.putText(img,classNames[classId -1].upper(),
                (box[0]+10,box[1]+30),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
                cv2.putText(img,str(round(confidence*100,2)),
                (box[0]+200,box[1]+30),cv2.FONT_HERSHEY_COMPLEX,1(0,255,0),2)
                print(distance ,cx*0.05)
                k=k+1
                c.append(cx)
                l.append(distance)
                if k==29:
                    print(k)
                    passval(c,l)

cv2.imshow('Output',img)
cv2.waitKey(1)
```

.7 INVERSE KINEMATICS

```
from numpy import *
from Stepper import TOP,MID,BOTTOM
from Servo_Motor import *

a1 = 24  # length of link a2 in cm
a2 = 16  # length of link a4 in  cm
# Desired Position of End effector
#x = 5
#y = 3

def inverse(x,d):
    d1=sqrt((0-a1)**2+(0-a2)**2)
    d2=sqrt((0-x)**2+(0-d)**2)
    if(d<40):
    # Equations for Inverse kinematics
        r = sqrt(x**2+d**2)  # eqn 1
        alpha = arccos((a2**2+a1**2-r**2)/(2*a1*a2))  # eqn 2
        theta_2=pi-alpha
        a2_sin_theta_2=sqrt(1-cos(theta_2**2))
        a2_cos_theta_2=a2*cos(theta_2)
        side_length=a1+a2_cos_theta_2
        beta=arctan2(a2_sin_theta_2 ,side_length)
        theta_1 = (arctan2(d,x))-beta
        theta_2 = rad2deg(theta_2)
        theta_1=rad2deg(theta_1)
        print(theta_1)
        s1=int(8.88*theta_1)
        s2=int(8.88*theta_2)
        print(s1)
        print(s2)
        MID('ant',s2)
        tb=BOTTOM('ant',s1)
        print(tb)
        if tb==1:
            MID('clk',s2)
            BOTTOM('clk',s1)
            open1()
    else:
        print("un reachable")
```

.8 STEPPER MOTOR

```
import RPi.GPIO as GPIO
from RpiMotorLib import RpiMotorLib
import time
from Servo_Motor import *
GPIO.setmode(GPIO.BCM)

T_STEP = 2 # Step GPIO Pin
T_DIR = 3 # Direction (DIR) GPIO Pin
T_EN = 4 # enable pin (LOW to enable)
M_STEP = 10 # Step GPIO Pin
M_DIR = 9 # Direction (DIR) GPIO Pin
M_EN = 11 # enable pin (LOW to enable)
B_STEP = 25 # Step GPIO Pin
B_DIR = 8 # Direction (DIR) GPIO Pin
B_EN = 7 # enable pin (LOW to enable)

# Declare a instance of class pass GPIO pins numbers and the motor type
Top = RpiMotorLib.A4988Nema(T_DIR, T_STEP, (21,21,21), "DRV8825")
GPIO.setup(T_EN,GPIO.OUT) # set enable pin as output
# Declare a instance of class pass GPIO pins numbers and the motor type
Mid = RpiMotorLib.A4988Nema(M_DIR, M_STEP, (21,21,21), "DRV8825")
GPIO.setup(M_EN,GPIO.OUT) # set enable pin as output
# Declare a instance of class pass GPIO pins numbers and the motor type
```

```
Bot = RpiMotorLib.A4988Nema(B_DIR, B_STEP, (21,21,21), "DRV8825")
GPIO.setup(B_EN,GPIO.OUT) # set enable pin as output

###########################
# Actual motor control
###########################

def TOP(dir,s3):
    if (dir == "ant"):
        GPIO.output(T_EN,GPIO.LOW) # pull enable to low to enable motor
        Top.motor_go(False, # True=Clockwise, False=Counter-Clockwise
                     "Full" , # Step type (Full,Half,1/4,1/8,1/16,1/32)
                     s3, # number of steps
                     .0005, # step delay [sec]
                     False, # True = print verbose output
                     .05) # initial delay [sec]
        print('ATOP')
        #time.sleep(5)
    if (dir == "clk"):
        GPIO.output(T_EN,GPIO.LOW) # pull enable to low to enable motor
        Top.motor_go(True, # True=Clockwise, False=Counter-Clockwise
                     "Full" , # Step type (Full,Half,1/4,1/8,1/16,1/32)
                     s3, # number of steps
                     .0005, # step delay [sec]
                     False, # True = print verbose output
                     .05) # initial delay [sec]

        print("Top")

def MID(dir,s2):
    if (dir == "ant"):
        GPIO.output(M_EN,GPIO.LOW) # pull enable to low to enable motor
        Mid.motor_go(False, # True=Clockwise, False=Counter-Clockwise
                     "Full" , # Step type (Full,Half,1/4,1/8,1/16,1/32)
                     s2, # number of steps
                     .0005, # step delay [sec]
                     False, # True = print verbose output
                     .05) # initial delay [sec]
        print('AMid')
    if (dir == "clk"):
        GPIO.output(M_EN,GPIO.LOW) # pull enable to low to enable motor
        Mid.motor_go(True, # True=Clockwise, False=Counter-Clockwise
                     "Full" , # Step type (Full,Half,1/4,1/8,1/16,1/32)
                     s2, # number of steps
                     .0005, # step delay [sec]
                     False, # True = print verbose output
                     .05) # initial delay [sec]

        print("MID")

def BOTTOM(dir,s1):
    if (dir == "ant"):
        GPIO.output(B_EN,GPIO.LOW) # pull enable to low to enable motor
        Bot.motor_go(False, # True=Clockwise, False=Counter-Clockwise
                     "Full" , # Step type (Full,Half,1/4,1/8,1/16,1/32)
                     s1, # number of steps
                     .0005, # step delay [sec]
                     False, # True = print verbose output
                     .05) # initial delay [sec]
        print('ABOT')
        close()
        return 1

    if (dir == "clk"):

        GPIO.output(B_EN,GPIO.LOW) # pull enable to low to enable motor
        Bot.motor_go(True, # True=Clockwise, False=Counter-Clockwise
                     "Full" , # Step type (Full,Half,1/4,1/8,1/16,1/32)
                     s1, # number of steps
                     .0005, # step delay [sec]
                     False, # True = print verbose output
                     .05) # initial delay [sec]

        print("Bot")
```

.9 SERVO MOTOR

```
from gpiozero import AngularServo
from time import sleep
servo = AngularServo(18, min_pulse_width=0.0006, max_pulse_width=0.0023)

def close():
    servo.angle = 90
    print("90")
    sleep(2)
    servo.angle = 90
    print("0")
    sleep(2)

def open1():
    servo.angle = -90
    print("-90")
    sleep(2)
    servo.angle = -90
    print("-90")
    sleep(2)
```