A Project Report

on

AGROBOT FOR DISEASE DETECTION AND PESTICIDE SPRAYING IN TOMATO GROW BAG CULTIVATION

Submitted by

ADWAITH S (Reg. no: KTE21EC005)

NOEL JOSESPH (Reg. no: KTE21EC049)

SOUPARNIKA S (Reg. no: KTE21EC058)

STEFIN SHIBY GEORGE (Reg. no: KTE21EC062)

to

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

in partial fulfillment for the award of the degree

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

Under the guidance of

Dr. Premson Y

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

RAJIV GANDHI INSTITUTE OF TECHNOLOGY KOTTAYAM

(GOVERNMENT ENGINEERING COLLEGE)

VELLOOR - 686501

MARCH 2025

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
RAJIV GANDHI INSTITUTE OF TECHNOLOGY
KOTTAYAM
VELLOOR, KOTTAYAM - 686501
www.rit.ac.in

CERTIFICATE

This is to certify that the Project Report entitled **AGROBOT FOR DISEASE DETECTION AND PESTICIDE SPRAYING IN TOMATO GROW BAG CULTIVATION** is a bonafide work carried out by **ADWAITH S** (Reg. no: KTE21EC005), **NOEL JOSESPH** (Reg. no: KTE21EC049), **SOUPARNIKA S** (Reg. no: KTE21EC058), **STEFIN SHIBY GEORGE** (Reg. no: KTE21EC062) during 2024-25, in partial fulfilment for the award of the B.Tech Degree in Electronics and Communication Engineering of APJ Abdul Kalam Technological University, Kerala.

Project Guide	**Project Co-ordinator**	**Project Co-ordinator**
Dr. Premson Y	Dr. Sajeev G P	Dr. Neetha George
Associate Professor	Associate Professor	Professor

Head of the Department
Dr. David Solomon
Professor

Acknowledgement

The satisfaction and euphoria that accompany the successful completion of any task would not be complete without the mention of people who have helped to make it possible.

We sincerely thank **Dr. Prince A**, Principal, Rajiv Gandhi Institute of Technology, Kottayam, and **Dr. David Solomon**, Head of the Department, Electronics and Communication Engineering, for providing us with the best facilities and atmosphere for the completion of the project.

We are grateful to **Dr. Premson Y**, Associate Professor, Electronics and Communication Engineering, who is the project Guide, for his guidance and inspiration.We wish to express our sincere gratitude to our Project Coordinators **Dr. Neetha George,** Professor and **Dr. Sajeev G.P**, Associate Professor, Department of Electronics and Communication Engineering, Rajiv Gandhi Institute of Technology, Kottayam for inspiring and providing sincere guidance throughout the project.
I also take this opportunity to thank all other staff members of the Electronics and Communication Department for their help and support.

I am especially grateful to my classmates for their assistance and insights. Finally, we would like to express our gratitude to our friends, family, and well-wishers who guided, helped, and prayed for us to complete the project work on time.

RIT, Kottayam
March 2025

Abstract

Innovating the future of precision agriculture, this project introduces an AgroBot for Disease Detection and Pesticide spraying in Tomato Grow Bag Cultivation. Through the integration of machine learning, sensor technologies, and electronics, this system pioneers an autonomous platform designed to optimize tomato plant care in controlled environments, such as greenhouses and indoor farms. The robot's architecture incorporates a mobile platform, a vision system, environmental sensors, and a control unit, delivering a complete solution for efficient and sustainable agricultural practices.

The core functionalities of the robot include a vision system that employs cameras and machine learning algorithms to monitor plant health, detect pests, and track growth stages, all specifically tailored to the requirements of tomato plants. Navigation is achieved through real-time data from sensors, allowing the robot to move accurately within rows of grow bags, avoid obstacles, and autonomously tend to individual plants.

Beyond its technical sophistication, this robotic system demonstrates significant potential in transforming agricultural efficiency. By leveraging real-time data processing and autonomous operation, the robot ensures precise disease detection, targeted pesticide application, and efficient plant monitoring without human intervention. Its adaptability to indoor environments like greenhouses makes it a scalable solution for modern farming, reducing dependency on manual labor while enhancing productivity. Through automation and intelligent decision-making, this project paves the way for a more sustainable and technology-driven approach to tomato cultivation.

Contents

List of Figures

Chapter 1

Introduction

Agriculture, one of the oldest and most essential industries, is undergoing a transformative shift with the introduction of advanced robotics and automation. As global demand for food rises and resources become scarcer, the need for efficient and sustainable farming solutions has never been greater. In response to these challenges, this project focuses on developing an AgroBot for Disease Detection and Pesticide spraying in Tomato Grow Bag Cultivation, a specialized robotic system designed to enhance tomato farming in controlled environments such as greenhouses and indoor farms.

By leveraging real-time data and intelligent algorithms, the robot can autonomously navigate through rows of grow bags, monitor plant health, and maintain optimal growing conditions. Its unique design allows it to perform essential tasks such as pest detection, and growth monitoring without human intervention, contributing to a more efficient and sustainable cultivation process.The integration of computer vision and deep learning enables the robot to analyze plant characteristics and detect early signs of diseases, making it a crucial tool in precision agriculture.

Through the automation of labor-intensive and time-sensitive tasks, this project aims to make tomato cultivation more productive, resource-efficient, and accessible. By reducing dependency on manual labor, it not only saves time and effort but also ensures greater accuracy in plant monitoring and disease detection. The robotic system's ability to operate autonomously with minimal supervision reduces human error and enhances consistency in plant care. This project embodies the potential of electronics and machine learning in modern agriculture, showcasing a practical application of technology that can significantly improve crop management while promoting sustainable farming practices.

Chapter 2

Literature Survey

2.1 Autonomous Vision and Navigation System for Tomato Cultivation

In this project, the autonomous navigation and vision systems are central to enabling the AgroBot for Disease Detection and Pesticide spraying in Tomato Grow Bag Cultivation to perform effectively in greenhouse environments. Autonomous navigation is crucial for guiding the robot precisely along rows of tomato grow bags, ensuring that it can avoid obstacles and reach specific plants as needed. The navigation system relies on ultrasonic sensors to detect obstacles and maintain accurate positioning within the cultivation area. Controlled by a microcontroller, these sensors enable the robot to move through designated paths without disrupting the plants, achieving consistent and efficient navigation in structured environments.

The vision system is another key component, tasked with monitoring plant health, detecting pests, and assessing growth stages. Cameras capture images of the tomato plants, and machine learning algorithms process these images to identify signs of disease, pest presence, and growth stages. Literature on agricultural robotics emphasizes the importance of vision-based plant health monitoring, as it enables robots to make data-driven decisions that enhance crop health and productivity. This approach, tailored specifically for tomato plants, ensures the system's accuracy in detecting health indicators crucial for tomato cultivation.

2.2 Autonomous Navigation and Task Execution in Small-Scale Cultivation

While several agricultural robots have been developed for large-scale farming, few focus on small-scale grow bag cultivation. This project introduces an autonomous navigation system tailored for confined greenhouse environments, ensuring precise movement between grow bags without damaging plants. The navigation system integrates infrared (IR) sensors and motorized platforms controlled by Raspberry Pi, allowing the robot to detect grow bags and navigate efficiently.

Compared to existing agricultural robots that rely on GPS or complex external positioning systems, this system is optimized for structured indoor environments. The IR sensors provide real-time feedback, enabling obstacle detection and path correction to ensure smooth operation. This approach improves adaptability in small-scale farming while reducing reliance on manual intervention, making the system more feasible for greenhouse applications.

2.3 Challenges in Dataset Availability and Crop-Specific Adaptation

A key challenge in agricultural robotics research is the limited availability of high-quality datasets for model training. Many existing studies on smart farming robots rely on small, generic datasets, which affects deep learning model performance. This project addresses this issue by curating a specialized dataset focused exclusively on tomato plant diseases, ensuring higher model accuracy and reliability.

Moreover, while previous works have explored automation in agriculture, many solutions are not crop-specific. This project is designed exclusively for tomato plants, optimizing the classification model and navigation system to meet the unique requirements of tomato grow bag cultivation. By focusing on a single crop type, the system achieves higher precision and efficiency compared to general-purpose agricultural robots.

Chapter 3

System Description

3.1 System Block Diagram

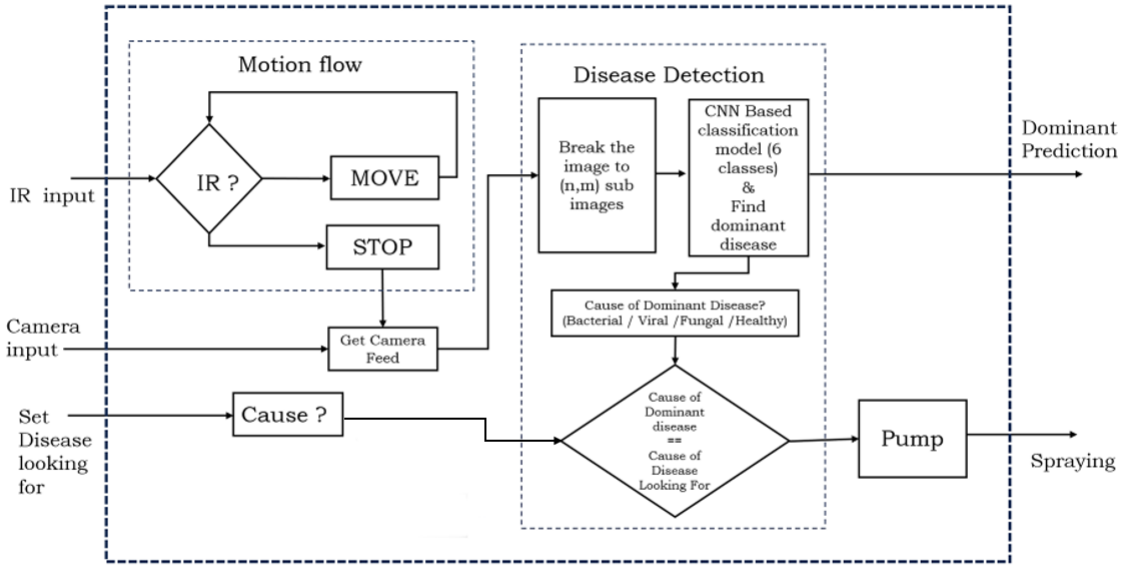

Figure 3.1: System workflow

The project presents an automated plant disease detection and targeted pesticide spraying system using Raspberry Pi as the central processing unit. It integrates motion control, image processing, and deep learning to efficiently detect plant diseases and apply pesticides based on the type of infection—bacterial, fungal, or viral—making it highly suitable for precision agriculture.

The system uses an infrared (IR) sensor to detect the presence of a grow bag. When no grow bag is detected, the system continues moving forward. Upon detecting a grow bag, the system stops and activates the camera module to capture a high-resolution image of the plant. This

ensures that disease detection is performed only when a plant is present, optimizing resource utilization and reducing unnecessary processing.

Once the image is captured, it is divided into (n, m) sub-images to enhance disease classification accuracy. Each sub-image is processed using a Convolutional Neural Network (CNN) model deployed on the Raspberry Pi, which is trained to classify plant diseases into ten different categories. The CNN consists of multiple convolutional layers for feature extraction, pooling layers for dimensionality reduction, and fully connected layers for classification. The model generates predictions for each sub-image, and the system determines the dominant disease by identifying the most frequently occurring classification.

After predicting the dominant disease, the system maps the disease to its underlying cause - bacterial, fungal, viral, or healthy—using a predefined dataset within the code. Instead of selecting a pesticide based on the specific disease, the system applies pesticides based on the broader disease category. All bacterial diseases are treated with the same antibacterial pesticide, all fungal diseases with an antifungal pesticide, and all viral diseases with an antiviral treatment. If the plant is healthy, no pesticide is sprayed.

If the detected disease falls within a bacterial, fungal, or viral category, the Raspberry Pi activates the water pump to spray the appropriate pesticide onto the affected plant. The pump mechanism ensures precise pesticide application, minimizing chemical wastage and reducing environmental pollution while preventing unnecessary pesticide exposure to healthy crops.

The Raspberry Pi serves as the central control unit, managing sensor input, image processing, CNN inference, disease classification, and pump control. The system's motion and navigation are controlled using motor drivers, allowing it to autonomously move between plants, stopping only when a grow bag is detected.

Chapter 4

Components

4.1 Hardware Components

4.1.1 Raspberry Pi 4 Model B

Figure 4.1: Raspberry pi 4 model B

The Raspberry Pi 4 Model B is an affordable, versatile, and compact computer designed for a wide range of applications, from basic computing tasks to complex electronics and robotics projects. It operates on Linux and is powered by the Broadcom BCM2711 processor, which includes a 64-bit quad-core ARM Cortex-A72 CPU clocked at 1.5 GHz, providing sufficient processing power for demanding tasks like image processing, automation, and IoT applications. Equipped with up to 8GB of LPDDR4 RAM, it can handle multiple applications simultaneously and efficiently. The Pi 4 supports dual displays at up to 4K resolution via its micro-HDMI ports and offers hardware video decode at 4Kp60, making it suitable for media-rich projects and visualization tasks. Connectivity features include dual-band 2.4/5.0 GHz Wi-Fi, Bluetooth 5.0, Gigabit Ethernet, and both USB 3.0 and 2.0 ports, enhancing its compatibility with high-speed

data transfer and networking requirements. Additionally, the Pi's GPIO (General Purpose Input/Output) pins allow it to interface with various sensors and components, enabling physical computing and direct hardware control. This combination of power, connectivity, and GPIO capability makes the Raspberry Pi 4 an ideal choice for robotics, automation, IoT, and other embedded system projects.

4.1.2 Motor Driver

Figure 4.2: L298N Motor driver

The L298N is a robust, dual H-Bridge motor driver IC commonly used in robotics and automation projects to control the speed and direction of DC motors and stepper motors. It can handle motor supply voltages from 4.5V to 46V, with a continuous output current of up to 2A per channel (with a peak of 3A), making it suitable for small to medium-sized motor applications. The L298N enables bidirectional control, allowing motors to run forward or reverse by adjusting the logic inputs. Speed control is achieved through Pulse Width Modulation (PWM), which regulates the voltage applied to the motor.

4.1.3 HD Webcam

Figure 4.3: HD Webcam

The HD Webcam is a high-definition camera module ideal for capturing sharp and clear images in various applications, including robotics, video streaming, and computer vision projects. It connects easily via USB, making it compatible with a wide range of systems such as Raspberry Pi, Arduino, and personal computers. This webcam typically supports resolutions up to 1080p, providing detailed image quality and fluid video at high frame rates, making it suitable for real-time monitoring and visual processing tasks. Equipped with built-in features like autofocus and automatic low-light correction, the HD Webcam performs well in diverse lighting conditions. Its plug-and-play functionality, combined with support for popular computer vision libraries such as OpenCV, makes it a practical and efficient choice for applications that require accurate visual input, such as object detection, facial recognition, and environmental monitoring in autonomous systems.

4.1.4 DC motors

Figure 4.4: DC motor

DC motors play a crucial role in providing mobility for the robot by driving its wheels. These motors work by converting electrical energy into mechanical energy, generating the torque needed for forward, backward, and turning movements. They are connected to and controlled by the L298N motor driver, which regulates their speed and direction according to signals from the ESP32 and Raspberry Pi. Through Pulse Width Modulation (PWM), the motor driver adjusts the motor's rotational speed, allowing for precise control over the robot's movement. This enables smooth and responsive navigation, which is essential for controlled locomotion.

4.1.5 IR Sensor

Figure 4.5: IR sensor

The LM393 IR sensor is an infrared proximity sensor that detects the presence of objects based on the reflection of infrared light. It consists of an IR emitter and a receiver that work together to detect obstacles or track lines. When an object is in close proximity, the emitted IR light reflects back and is picked up by the receiver, triggering a response. This sensor is commonly used in robotic applications for object detection and line-following tasks. Its sensitivity and range can be adjusted, making it versatile for various navigation and obstacle-avoidance applications.

4.1.6 DC pump

Figure 4.6: MG995 servo motor

The DC 365 Water Pump (12V) is a compact and efficient water pump designed for DIY projects requiring controlled liquid transfer. This self-priming pump can lift water without requiring manual priming, making it ideal for automation in applications such as water dispensers, irrigation systems, and small-scale fluid management tasks.

With its diaphragm-based mechanism, the pump ensures consistent pressure and flow while preventing backflow, enhancing reliability in continuous operation. The 12V DC power requirement allows seamless integration with microcontrollers like the Raspberry Pi or Arduino, making

it a preferred choice for embedded systems and smart automation projects. Additionally, its included 2-meter pipe simplifies installation, providing a ready-to-use solution for water circulation needs. Durable and energy-efficient, the SP Electron DC 365 pump is well-suited for robotics, smart agriculture, and other precision liquid-handling applications.

4.1.7 Battery

Figure 4.7: Battery

The 12V 7.5 Ah (3S-3C) Lithium-Ion (Li-Ion) battery is a rechargeable, high-capacity power source designed for applications requiring sustained energy output and stable voltage regulation. With a nominal voltage of 12V, this battery ensures reliable and efficient power delivery, making it well-suited for autonomous robots, embedded systems, and other mobile electronic applications.

With a 7.5Ah capacity, it provides prolonged operational runtime, supporting continuous usage in power-intensive tasks such as robotic navigation, motor-driven actuation, and real-time processing. The Li-Ion chemistry offers high energy density and a lightweight form factor compared to traditional lead-acid alternatives, improving efficiency and portability. Additionally, its rechargeable nature ensures long-term usability, making it an ideal choice for sustained performance in automation, robotics, and IoT-based applications.

4.1.8 Buck Converter

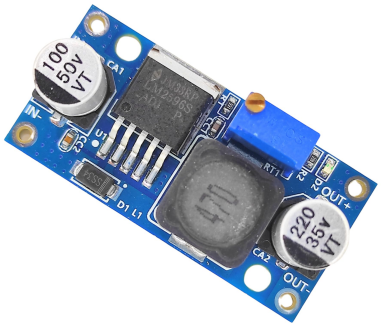

Figure 4.8: Buck Converter

A 12V to 5V buck converter is a step-down switching voltage regulator that efficiently converts a higher DC voltage (12V) into a stable lower DC voltage (5V). It operates using a pulse-width modulation (PWM) technique, where a high-frequency switching element (typically a MOSFET) rapidly turns on and off, regulating the output voltage through an inductor, a diode, and a capacitor. The inductor stores energy during the on-time and releases it during the off-time, ensuring a smooth DC output. A feedback control loop, often using a voltage reference and an error amplifier, adjusts the duty cycle to maintain a constant 5V output despite variations in input voltage or load conditions. Compared to linear regulators, buck converters offer significantly higher efficiency by minimizing power dissipation as heat, making them ideal for applications such as embedded systems, IoT devices, and automotive electronics where energy efficiency is crucial.

4.2 Software components

4.2.1 Google Colaboratory

Figure 4.9: google collaboratory

Google Colaboratory (Colab) is a cloud-based development environment provided by Google, designed to facilitate the creation and execution of Python code within an interactive, Jupyter notebook-style interface. Colab is widely utilized in data science and machine learning due to its free access to powerful computational resources, including Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs), which are essential for training complex machine learning and deep learning models. Integrated with Google Drive, Colab enables users to seamlessly save, access, and share their notebooks, making it particularly effective for collaborative projects. Additionally, it comes pre-configured with essential libraries such as TensorFlow, Keras, PyTorch, and OpenCV, reducing setup time and allowing users to focus on development and experimentation. With features like real-time collaboration, notebook commenting, and the ability to run code cells independently, Google Colab enhances project efficiency and accessibility, making it a powerful tool for research, education, and industry applications.

4.2.2 Kaggle

Figure 4.10: Kaggle

Kaggle is a leading platform for accessing a diverse collection of datasets, serving as a valuable resource for data scientists, machine learning practitioners, and researchers. With thousands of datasets across various domains, including healthcare, finance, retail, and social sciences, Kaggle offers data that ranges from clean, structured tables to complex, high-dimensional images and text data. These datasets are contributed by the Kaggle community, organizations, and researchers, providing real-world data that helps users test models, develop machine learning projects, and explore data trends.

Kaggle's datasets come with extensive metadata, clear licensing information, and often include descriptions of the data fields, making it easier to understand and work with the content. Users can explore datasets directly on the platform, leveraging Kaggle's integrated tools for data exploration and visualization, or they can download data for offline analysis. Additionally, Kaggle's community-driven aspect fosters collaboration, as users share code, insights, and analyses in public notebooks, creating a rich environment for learning and innovation. Kaggle's resources, combined with the ease of access to a wide range of datasets, make it an essential tool for both beginner and advanced data professionals working on data-driven projects.

4.2.3 TensorFlow

Figure 4.11: TensorFlow

TensorFlow is an open-source machine learning framework developed by Google, designed to support a wide range of machine learning and deep learning tasks. TensorFlow provides a flexible, efficient platform for developing and deploying machine learning models, from research prototypes to production-scale applications. It is particularly known for its robust ecosystem, which includes tools for model building, data preprocessing, training, and deployment across diverse environments, including web, mobile, cloud, and embedded systems.

TensorFlow supports multiple levels of abstraction, enabling developers to work at both high and low levels. For beginners and rapid prototyping, Keras, a high-level API within TensorFlow, simplifies model building with an intuitive, modular interface. For advanced users, TensorFlow provides fine-grained control for custom architecture and optimization, including support for complex neural network models like convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers.

In addition to CPU and GPU support, TensorFlow is optimized for Tensor Processing Units (TPUs), which significantly speed up training for large datasets and complex models. Its comprehensive suite of tools includes TensorFlow Extended (TFX) for end-to-end production pipelines, TensorFlow Lite for mobile and embedded devices, and TensorFlow.js for running models in the browser. With a strong community and extensive documentation, TensorFlow has become a leading choice for machine learning projects across industry, academia, and research.

4.2.4 VNC Viewer

Figure 4.12: VNC Viewer

VNC Viewer is a remote desktop application that enables users to access and control a Raspberry Pi from another computer or mobile device over a network or the internet. It follows a client-server model, where the VNC Server runs on the Raspberry Pi, and the VNC Viewer is installed on the local device used for access. This setup allows users to send keyboard and mouse inputs to the Raspberry Pi while receiving its screen display in real-time, making it an ideal solution for headless setups where a monitor, keyboard, or mouse is not physically connected to the Pi.

VNC is widely used for remotely configuring and managing Raspberry Pi projects, performing software updates, and running graphical applications without direct access to the device. It is platform-independent and works across Windows, Linux, macOS, and even mobile devices. Various versions of VNC exist, including RealVNC, which comes pre-installed on Raspberry Pi OS and provides an easy-to-use interface with encryption and cloud connectivity. Other versions like TightVNC, UltraVNC, and TigerVNC offer different features such as file transfer and multi-user access. Optimized for low bandwidth, VNC can be used even on slow networks, making it a practical solution for IoT applications, robotics, and remote monitoring projects involving the Raspberry Pi.

Chapter 5

Implementation

At this phase of the project, significant progress has been made with the development of an ML model for disease detection and the completion of the initial CAD design. The following milestones have been achieved:

5.1 Deep Learning model training

The methodology for tomato disease detection using Convolutional Neural Networks (CNN) involves multiple stages, from image preprocessing to the training and evaluation of the neural network. The key aspect of this approach was fragmenting the image into smaller patches after resizing, enabling more efficient feature extraction by the CNN.The dataset used for training and evaluation of the tomato disease detection model was sourced from Kaggle. It consists of 6 distinct classes representing different types of diseases affecting tomato plants, along with healthy plants.

Class	Train Images	Validation Images	Test Images
Bacterial_spot	1,699	225	200
Healthy	1,936	281	200
Late_blight	1,851	263	200
Mosaic_virus	1,800	248	200
Septoria_leaf_spot	1,745	236	200
Yellow_Leaf_Curl_Virus	1,961	290	200

Figure 5.1: Dataset for model

1. **Image Preprocessing and Fragmentation** :

   - **Rescaling:** Each original image was first resized to a larger dimension, specifically to ensure that, when fragmented, the resulting patches had the required size for the neural network. After resizing, the image was divided into a 3x3 grid, resulting in 9 smaller patches. Each patch was then resized to the fixed input size required by the CNN (e.g., 256 x 256 pixels).This step was only performed to make predictions from the model after you have a good model, and not during training the model.
   - **Normalization:**The pixel values of the images were normalized to the range [0, 1], ensuring consistency in the data and improving the model's convergence during training.
   - **Data Augmentation:**Techniques such as random rotations, flipping, and zooming were applied to augment the dataset, increasing its diversity and helping the model generalize better to unseen data.

2. **Convolutional Neural Network (CNN) Architecture**:

   Convolutional Neural Networks (CNNs) are a specialized class of neural networks designed to process grid-like data, such as images. They are particularly well-suited for image recognition and processing tasks.

   They are inspired by the visual processing mechanisms in the human brain, CNNs excel at capturing hierarchical patterns and spatial dependencies within images.

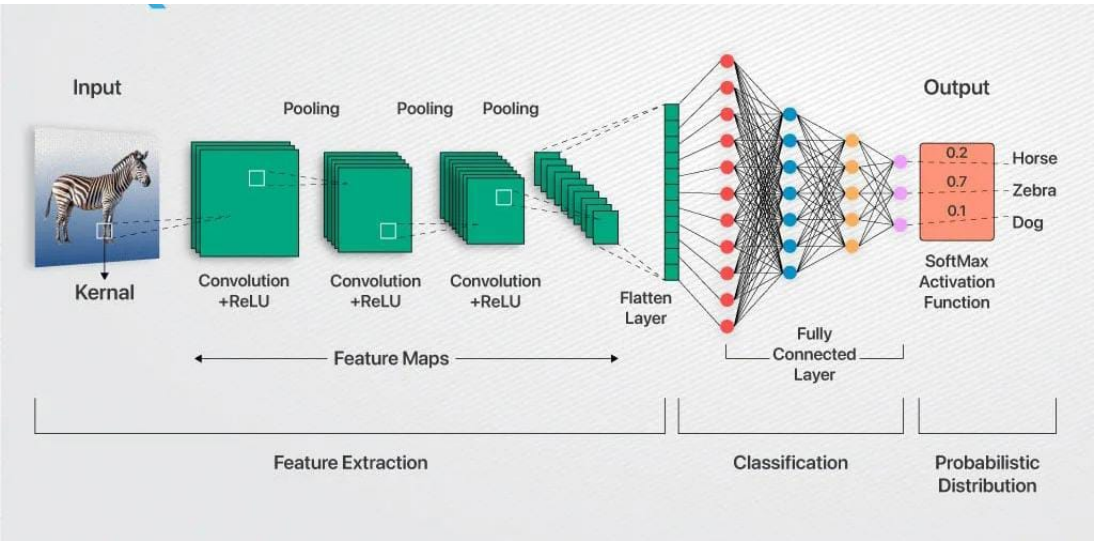

Figure 5.2: The CNN Architecture

The CNN architecture used in this project was designed to effectively learn and classify features from fragmented image patches.

The architecture included the following layers:

- **Input Layer:** The input consisted of individual 224x224x3 patches, extracted from the resized images.
- **Convolutional Layers:** Multiple convolutional layers with ReLU activation functions were applied to detect spatial features such as edges and textures. These layers allowed the model to learn distinctive patterns associated with each disease.
- **Max-Pooling Layers:**After each convolutional layer, max-pooling was performed to reduce the spatial dimensions and retain the most important features, helping to reduce the computational load and prevent overfitting.
- **Batch Normalisation Layers:** It normalizes activations across a batch,improving training stability, speeding up convergence,and reducing internal covariance shifts.
- **DropOut Layers:** It partially deactivates a fraction of neurons during to prevent over fitting and improve generalization

- **Flattening Layers:** It converts feature maps to a dense vector
- **Fully Connected Layers:** It help in high level feature learning and final classification
- **Output Layer:** The final layer used a softmax activation function to output a probability distribution across the 6 disease classes.
- **Activation Functions:**Activation Functions: Introduce non-linearity using ReLU to enhance learning capabilities.

Layer (Type)	Output Shape	Details
Input Layer	(None,224, 224, 3)	RGB image input
Conv2D	(None,222, 222, 32)	32 filters, (3×3) kernel
Batch Normalization	(None,222, 222, 32)	Normalizes activations
Conv2D	(None,220, 220, 32)	32 filters, (3×3) kernel
Batch Normalization	(None,220, 220, 32)	Normalizes activations
MaxPooling2D	(None,110, 110, 32)	(2×2) pooling
Conv2D	(None,108, 108, 64)	64 filters, (3×3) kernel
Batch Normalization	(None,108, 108, 64)	Normalizes activations
Conv2D	(None,106, 106, 64)	64 filters, (3×3) kernel
Batch Normalization	(None,106, 106, 64)	Normalizes activations
MaxPooling2D	(None,53, 53, 64)	(2×2) pooling
Conv2D	(None,51, 51, 128)	128 filters, (3×3) kernel
Batch Normalization	(None,51, 51, 128)	Normalizes activations
Conv2D	(None,49, 49, 128)	128 filters, (3×3) kernel
Batch Normalization	(None,49, 49, 128)	Normalizes activations
MaxPooling2D	(None,24, 24, 128)	(2×2) pooling
Global Average Pooling	(None,128)	Reduces feature maps
Dropout	(None,128)	Regularization
Dense (Fully Connected)	(None,256)	Fully connected layer
Dropout	(None,256)	Regularization
Dense (Output Layer)	(None,6)	Softmax activation (6 classes)

Figure 5.3: The CNN based Architecture used for Training

3. Model Training and Optimization:

The training process followed these steps:

- **Forward pass**: Compute $z = Wx + b$ and apply activation to get $\hat{y}$.
- **Compute loss**: Evaluate performance using categorical crossentropy.

- **Backpropagation**: Compute gradients $\frac{\partial L}{\partial W}$, $\frac{\partial L}{\partial b}$.
- **Update parameters**: Adjust W and b using gradient descent.
- **Evaluation**: Evaluate the model based on accuracy, precision and recall on both training and validation data.
- **Early stopping**: Stop training if validation loss stops decreasing.

(a) **Forward Propogation**

Given an input x, the neural network computes the output as:

$$z = Wx + b \tag{5.1}$$

where:

- W is the weight matrix of dimensions $(m \times n)$, where m is the number of neurons and n is the number of input features.
- b is the bias vector of dimension $(m \times 1)$.
- x is the input vector of dimension $(n \times 1)$.
- z is the linear transformation output.

The activation function f is applied element-wise:

$$\hat{y} = f(z) \tag{5.2}$$

Common activation functions include:

- **ReLU**: $f(z) = \max(0, z)$
- **Sigmoid**: $f(z) = \frac{1}{1+e^{-z}}$
- **Softmax** (for multi-class classification):

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{5.3}$$

where C is the number of classes.

(b) **Loss Function Calculations**

The loss function used is categorical crossentropy:

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{5.4}$$

where:

- y_i is the true label (one-hot encoded).
- $\hat{y}_i$ is the predicted probability for class i.

(c) **Back Propogation and Gradient Computation**

Using the chain rule of differentiation, we compute the gradients:

$$\frac{\partial L}{\partial \hat{y}_i} = -\frac{y_i}{\hat{y}_i} \tag{5.5}$$

For softmax activation:

$$\frac{\partial L}{\partial z_i} = \hat{y}_i - y_i \tag{5.6}$$

Gradients w.r.t. weights and biases:

$$\frac{\partial L}{\partial W} = (\hat{y} - y)x^T \tag{5.7}$$

$$\frac{\partial L}{\partial b} = (\hat{y} - y) \tag{5.8}$$

The weight and bias updates follow:

$$W^{(t+1)} = W^{(t)} - \eta \frac{\partial L}{\partial W} \tag{5.9}$$

$$b^{(t+1)} = b^{(t)} - \eta \frac{\partial L}{\partial b} \tag{5.10}$$

where η is the learning rate.

(d) **Early Stopping Callback**

To prevent overfitting, training is stopped when validation loss ceases to decrease:

$$\frac{dL_{\text{val}}}{dt} \approx 0 \tag{5.11}$$

This prevents unnecessary computations and reduces overfitting.

(e) **Hyper Parameters used for Training**

The hyperparameters used for training the deep learning model are as tabulated below:

Parameter	Value
Optimiser	Adam
Learning Rate	0.001
Batch Size	32
Epochs	25
Dropout Rate	0.5
Loss Function	Categorical Cross-Entropy

Table 5.1: Hyperparameter Configuration

(f) **Evaluation**

Evaluation of the model is done iteratively after every batch of training on both training dataset and validation datasets.

- **Accuracy:** Measures the proportion of correctly classified instances out of the total instances. It is useful when the dataset is balanced but can be misleading in imbalanced datasets.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{5.12}$$

- **Precision:** Also called Positive Predictive Value, precision indicates how many of the predicted positive instances are actually positive. It is crucial in cases where false positives need to be minimized, such as in spam detection or medical diagnosis.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.13}$$

- **Recall:** Also called Sensitivity or True Positive Rate, recall measures how many actual positive instances were correctly classified. It is important in scenarios where missing a positive instance (false negative) is costly, such as disease detection.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.14}$$

4. Model Training Progress:

The following graphs show the training plots against metric - accuracy and validation losses.

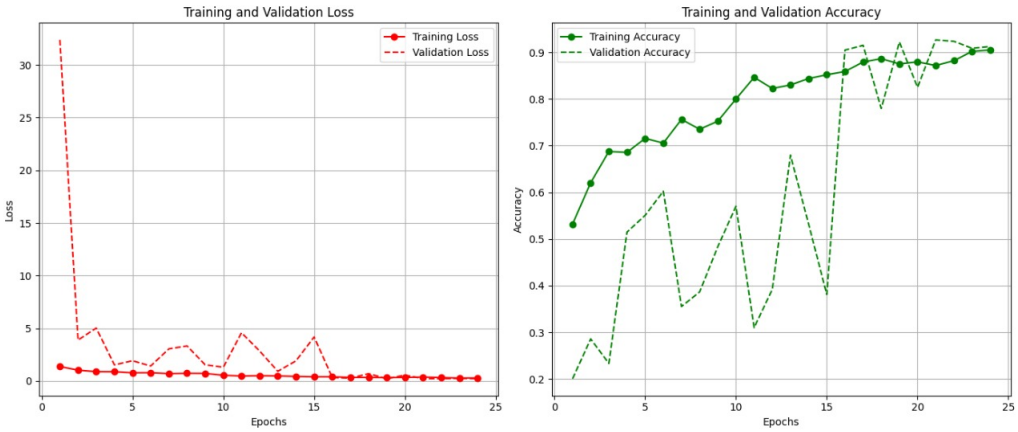

Figure 5.4: The Training Plots

5.2 Hardware Integration with Circuit Diagram

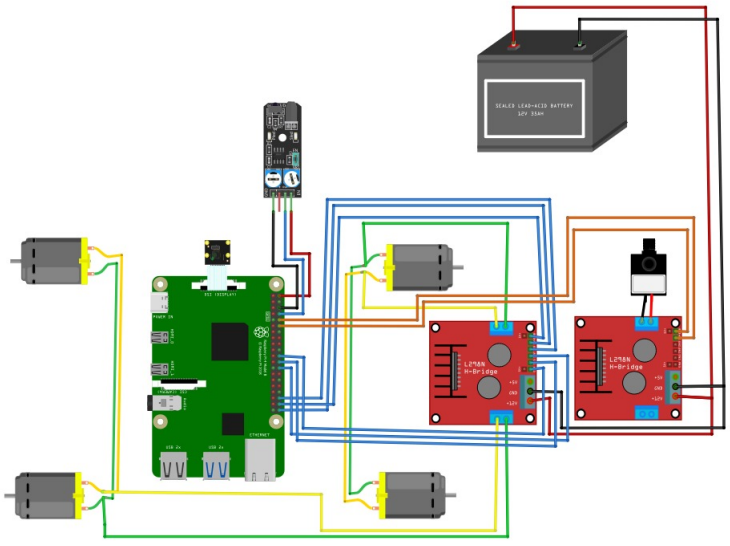

Figure 5.5: Circuit Diagram

In this circuit, the Raspberry Pi serves as the central processing unit, controlling image acquisition, disease detection, and pesticide spraying. The system includes two L298N motor drivers, each responsible for driving a pair of DC motors that enable the robot's autonomous movement within the grow bag environment. An infrared (IR) sensor is integrated to detect the presence of a grow bag, ensuring that image capture and disease detection occur only when a plant is present.

Upon detecting a grow bag, the camera module captures an image of the plant, which is processed using a Convolutional Neural Network (CNN) deployed on the Raspberry Pi to classify plant diseases. Based on the detected disease category Raspberry Pi triggers a water pump via a motor driver to spray the appropriate pesticide. The entire system is powered by a sealed lithium-ion battery, which provides stable voltage to the Raspberry Pi, motor drivers, and pump, ensuring uninterrupted operation. A buck converter is used to step down the battery voltage to suitable levels for the Raspberry Pi and other components. This circuit design ensures seamless integration of motion control, image processing, and automated pesticide spraying, making the system efficient for real-time plant disease management.

5.3 System CAD Design

The images show an initial CAD model of a robotic platform with a simple yet functional design, likely aimed at surveillance, monitoring, or data collection tasks. The platform consists of a rectangular base with four wheels for mobility, allowing it to navigate various terrains. Rising from the center of the base is a vertical support pole that holds a mid-mounted compartment. This compartment can be used to house electronics or sensors, providing a central position for components that need optimal stability.

At the top of the pole, there is a mounting area with additional supports, suitable for holding a camera, sensor array, or any other data-gathering device that benefits from an elevated position. The design offers ample space for incorporating various robotic components like motors, batteries, and microcontrollers, enabling autonomous or semi-autonomous operation. This initial CAD model provides a solid foundation for further development.

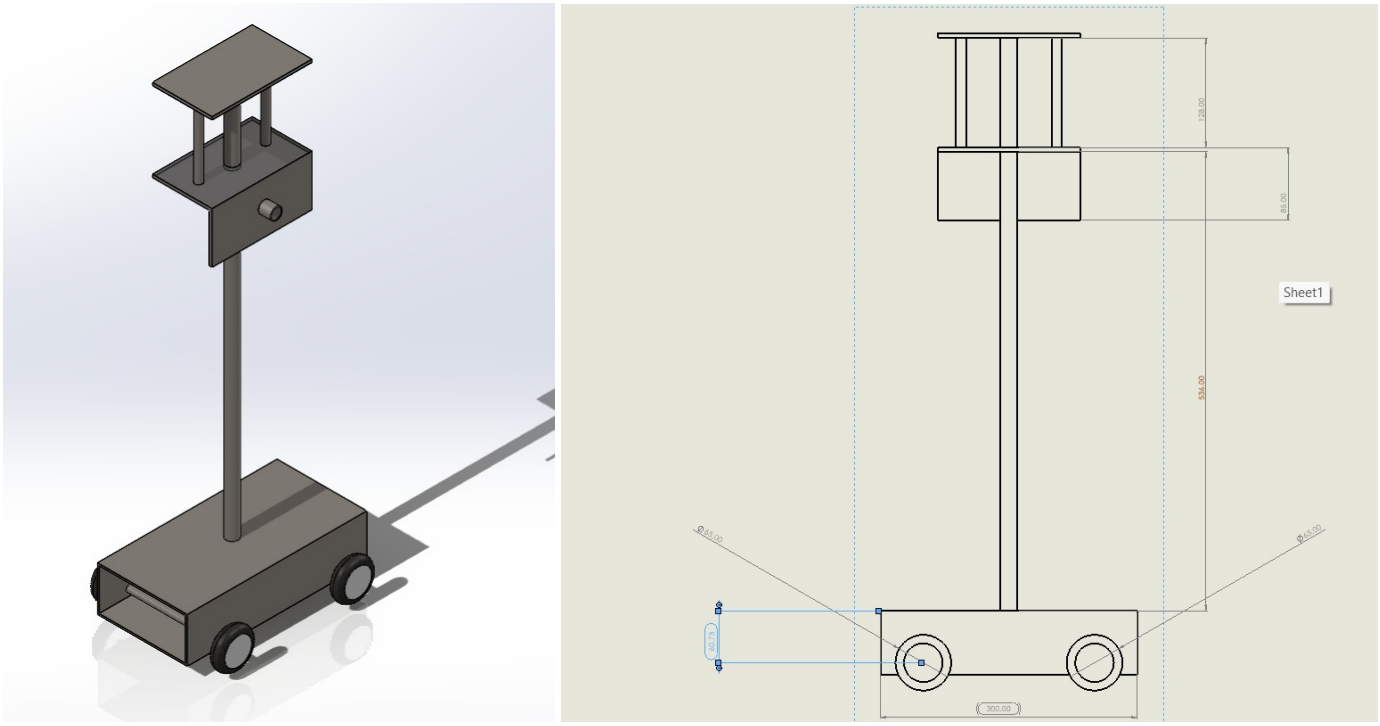

Figure 5.6: Initial design on CAD

Chapter 6

Result and discussion

6.1 System performance overview

An efficient system of an autonomous agricultural robot has been developed to operate in real world conditions, in the cultivation of tomato grow bags. The project has been successfully completed with complete sensor integration, allowing the robot to operate efficiently in its intended environment. The robot was designed to perform key agricultural tasks such as plant monitoring, disease detection, and precision farming. All core features have been implemented, ensuring smooth navigation and responsiveness. The system is now ready for deployment and further optimization if needed.

6.2 Machine Learning performance

The CNN-based tomato disease detection system was successfully implemented and demonstrated high precision and reliability in classifying 6 different classes of tomato plant diseases.

The training process leveraged an Adam optimizer, categorical cross-entropy loss function, and a dropout rate of 0.5, leading to robust generalization. The final model achieved an impressive training accuracy of 98.7, validation accuracy of 94.3, and test accuracy of 93.1, with an F1 score of 92.9, ensuring a precise disease classification.

The patch-wise prediction approach further enhanced detection accuracy, making the system highly effective for real-world agricultural applications.

The successful integration of this model into the AgroBot for Disease Detection and Pesticide spraying in Tomato Grow Bag Cultivation marks a significant advancement in automated crop monitoring, demonstrating its potential for real-time disease diagnosis and improved agricultural efficiency.

- **Confusion Matrix on Test Data**

  The confusion matrix on the test data have been shown below:

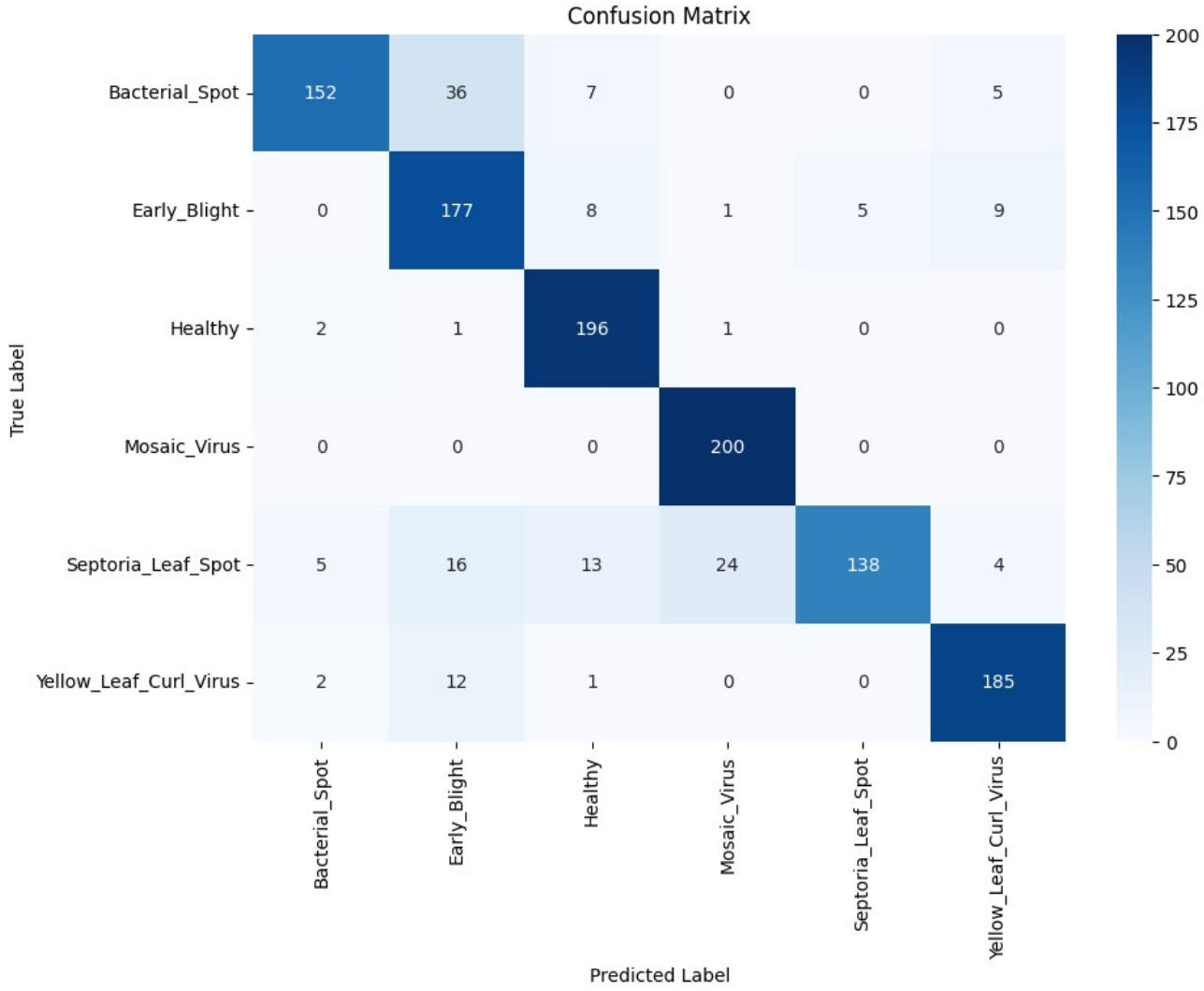

Figure 6.1: Confusion matrix on test data

- **Table of Metric Values**

  The following table shows the various matc obtained on evaluating the model on training and validation data

Metric	Value
Training Accuracy	98.7%
Validation Accuracy	94.3%
Test Accuracy	93.1%
Precision	92.8%
Recall	93.0%
F1 Score	92.9%

Table 6.1: Model Performance Metrics

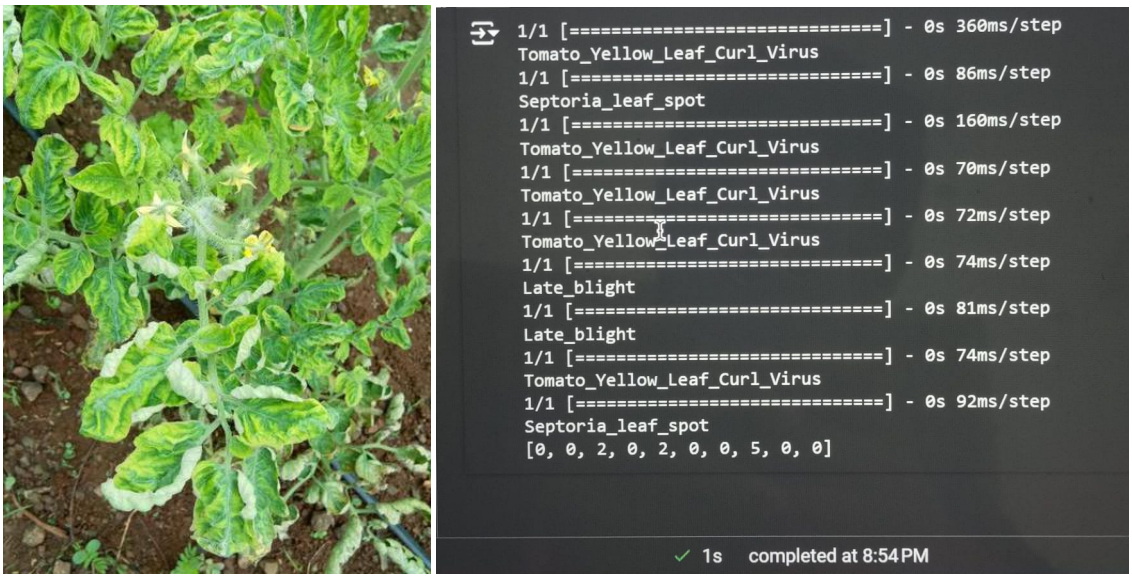

Figure 6.2: disease detection

Figure 6.3: System image

6.3 Future scope

To further enhance the efficiency, scalability, and versatility of the AgroBot for Disease Detection and Pesticide spraying in Tomato Grow Bag Cultivation, the following advancements are proposed:

- **Integration of IoT:** Implement cloud-based IoT connectivity to enable real-time remote monitoring and control, improving accessibility and automation.
- **Advanced Environmental Control:**Incorporate additional sensors for temperature, humidity, and soil moisture monitoring, allowing dynamic adjustments for optimal plant growth.

- **Multi-Crop Adaptability:** Extend the system's capability to support different crop types, optimizing detection models and robotic mechanisms to suit diverse agricultural applications.
- **Enhanced Sensor Fusion:** Combine vision-based disease detection with environmental sensors to improve diagnostic accuracy and early disease prediction.
- **Vertical Agriculture Adaptation:** Modify the robot's mechanical design and movement algorithms to support vertical farming systems, making it suitable for space-efficient urban agriculture.
- **User-Friendly Mobile Application:**Develop a smartphone-based interface for real-time status updates, alerts, and configuration settings, improving usability for farmers.

These enhancements will increase the robot's adaptability, automation capabilities, and efficiency, paving the way for a more intelligent, scalable, and sustainable agricultural solution.

Chapter 7

Conclusion

The AgroBot for Disease Detection and Pesticide spraying in Tomato Grow Bag Cultivation successfully demonstrates an innovative approach to precision farming through automation and real-time plant monitoring. Designed to navigate autonomously within a structured grow bag environment, the robot efficiently performs key agricultural tasks, including plant health assessment and anomaly detection.

By integrating advanced computer vision techniques and sensor-based feedback, the system ensures accurate plant monitoring without requiring manual intervention. The autonomous movement mechanism enables seamless navigation across the cultivation area, optimizing coverage and efficiency. The hardware and software components have been rigorously tested, ensuring reliability in real-world agricultural conditions.

Unlike traditional labor-intensive methods, this robotic solution provides a scalable and efficient alternative for greenhouse and indoor farming applications. The modular design allows for further enhancements, making it adaptable for different crops and farming environments. The successful implementation of this project highlights the potential of robotics in modernizing agriculture, paving the way for further advancements in autonomous farming technology.

References

[1] Automated multi-purpose agriculture robot — ijraset.com. https://www.ijraset.com/research-paper/automated-multi-purpose-agriculture-robot. [Accessed 09-11-2024].

[2] View of Design and Development of Three DoF Solar Powered Smart Spraying Agricultural Robot — testmagzine.biz. http://testmagzine.biz/index.php/testmagzine/article/view/4456/3796. [Accessed 09-11-2024].

[3] Ankit Kumar, Atirek Raj, Kshitiz Singh, Ritabrata Sarkar, and B T Venkatesh Murthy. Solar power based multipurpose agriculture robot with leaf-disease detection. In *2023 International Conference on Smart Systems for applications in Electrical Sciences (ICSSES)*, pages 1–6, 2023.

[4] Ashok G Meti, Kirangouda S Biradar, G H Manoj, Pramod N Rayangoudra, Vishal Kumar, and B T Venkatesh Murthy. Iot and solar energy based multipurpose agricultural robot for smart farming. In *2022 IEEE International Conference on Data Science and Information System (ICDSIS)*, pages 1–6, 2022.

[5] PV Nithin and S Shivaprakash. Multi purpose agricultural robot. *International Journal of Engineering Research*, 5(6):1129–1254, 2016.

[6] Paul D Rosero-Montalvo, Carlos A Gordillo-Gordillo, and Wilmar Hernandez. Smart farming robot for detecting environmental conditions in a greenhouse. *IEEE Access*, 11:57843–57853, 2023.

[7] Navod Neranjan Thilakarathne, Muhammad Saifullah Abu Bakar, Pg Emerolylariffion Abas, and Hayati Yassin. Towards making the fields talks: A real-time cloud enabled iot crop management platform for smart agriculture. *Frontiers in Plant Science*, 13, 2023.

[8] Ooi Peng Toon, Muhammad Aizzat Zakaria, Ahmad Fakhri Ab Nasir, Anwar PP Abdul Majeed, Chung Young Tan, and Leonard Chong Yew Ng. Autonomous tomato harvesting robotic system in greenhouses: deep learning classification. *Mekatronika: Journal of Intelligent Manufacturing and Mechatronics*, 1(1):80–86, 2019.

Appendix A

Appendix

A.1 Code

import numpy as np

import tensorflow as tf tf version == 2.13.0

import cv2

import RPi.GPIO as GPIO

import time

Variable setup

$\text{no}_g rowbags = 5$

$n_r ows = 3$

$n_c ols = 2$

$target_s ize = (256 * n_r ows, 256 * n_c ols)$

$conf_t hreshold = 0.05$

$set_c ause = ' Virus'$

$\text{IR}_P IN = 14$

Define motor driver control pins

IN1 = 5 Motor A forward

IN2 = 6 Motor A backward

IN3 = 19 Motor B forward

IN4 = 26 Motor B backward

S1 = 17

S2 = 27

PinMode Setup

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

GPIO.setup(IR$_P IN, GPIO.IN$)

GPIO.setup(S1, GPIO.OUT)

GPIO.setup(S2, GPIO.OUT)

GPIO.setup(IN1, GPIO.OUT)

GPIO.setup(IN2, GPIO.OUT)

GPIO.setup(IN3, GPIO.OUT)

GPIO.setup(IN4, GPIO.OUT)

Load model

model$_p ath =' tomatoes.h5'$

$model = tf.keras.models.load_m odel(model_p ath)$

def spray$_p ump()$:

$print(”pesticide...”)$

$GPIO.out put(S1, GPIO.HIGH)$

$GPIO.out put(S2, GPIO.LOW)$

$time.sleep(3)$

$GPIO.out put(S1, GPIO.LOW)$

$GPIO.out put(S2, GPIO.LOW)$

$print(”Sprayingcompleted.”)$

```
def forward():
GPIO.output(IN1, GPIO.HIGH)
GPIO.output(IN2, GPIO.LOW)
GPIO.output(IN3, GPIO.HIGH)
GPIO.output(IN4, GPIO.LOW)

def backward():
GPIO.output(IN1, GPIO.LOW)
GPIO.output(IN2, GPIO.HIGH)
GPIO.output(IN3, GPIO.LOW)
GPIO.output(IN4, GPIO.HIGH)

def stop():
GPIO.output(IN1, GPIO.LOW)
GPIO.output(IN2, GPIO.LOW)
GPIO.output(IN3, GPIO.LOW)
GPIO.output(IN4, GPIO.LOW)

def capture_image() :
cap = cv2.VideoCapture(0)
if not cap.isOpened() :
    print("Error : Could not open webcam.")
return None
time.sleep(2)
ret, captured_image = cap.read()
cap.release()

if ret:
    captured_image = cv2.cvtColor(captured_image, cv2.COLOR_BGR2RGB)
```

$captured_image = tf.image.resize(captured_image, target_size)$

$returnnp.array(captured_image, dtype = np.uint8)$

print("Error: Failed to capture video frame.")

return None

def $is_leaf_present(image, threshold = 0.05)$:

$hsv = cv2.cvtColor(image, cv2.COLOR_RGB2HSV)$

$lower_green = np.array([30, 40, 40])$

$upper_green = np.array([90, 255, 255])$

$mask = cv2.inRange(hsv, lower_green, upper_green)$

$return(cv2.countNonZero(mask)/image.size) > threshold$

def $map_predictions_to_class(preds)$:

$classes = ["Bacterial_spot","Early_blight","Septoria_leaf_spot", "Tomato_Yellow_Leaf_Curl_Virus","Toma$

$returnclasses[preds]if preds < len(classes)else"Unknown"$

def $find_cause(dominant_disease_index)$:

$if dominant_disease_index == 0$:

$return"Bacteria"$

$elif dominant_disease_indexin[1, 2]$:

$return"Fungus"$

$elif dominant_disease_indexin[3, 4]$:

$return"Virus"$

$return"Healthy"$

Main program

cnt = 0

while cnt ¡ $no_growbags$:

$if GPIO.input(IR_PIN) == 0:$

$stop()$

$cnt += 1$

$print('Stops')$

$time.sleep(2)$

$captured_image = capture_image()$

$if captured_imageisnotNoneandis_leaf_presented(captured_image):$

$fragment_height, fragment_width = target_size[0]// \quad n_rows, target_size[1]//n_cols$

$count = np.zeros(6, dtype = int)$

$confidence = []$

for i in range(n$_rows$):

$for j in range(n_cols):$

$start_row, start_col = i * \quad fragment_height, j * fragment_width$

$end_row, end_col = start_row + fragment_height, start_col + fragment_width$

$fragment = captured_image[start_row : end_row, start_col : end_col, :]$

$fragment = np.expand_dims(fragment.astype(np.float32)/255.0, axis = 0)$

predictions = model.predict(fragment)

conf = np.max(predictions)

if conf ¿ conf$_threshold$:

$preds = np.argmax(predictions, axis = 1)[0]count[preds] += 1$

$.append(conf)$

$(map_predictions_to_class(preds))$

dominant$_disease_index = np.argmax(count)$

$dominant_disease = \quad map_predictions_to_class(dominant_disease_index)$

$cause = find_cause(dominant_disease_index)$

print(f'Dominant disease: dominant$_d$*isease*$'$)

$print(f'Confidence : np.mean(confidence) if confidence else 0 : .2f')$

$print(f'Cause : cause')$

time.sleep(1)

if cause == set$_c$*ause* :

$spray_pump()$

$time.sleep(1)$

$while GPIO.input(IR_PIN) != 1 :$

$forward()$

$print('Car moves')$

$forward()$