GAYATRI VIDYA PARISHAD

COLLEGE FOR DEGREE AND PG COURSES (A)

(Affiliated to Andhra University)

Rushikonda, Visakhapatnam

Department of Computer Science

VISION

"Creating human excellence for a better society"

MISSION

"Unfold into a world class organization with strong academic and research base, producing responsible citizens to cater to the changing needs of the society."

AI-Powered IoT System for Plant Disease Detection And Fertilizer Precision Spraying

A project report submitted in partial fulfillment of

the requirements for the award of the Degree of

Master of Science in Computer Science

Submitted by

KOVIRI SANDEEP

(Regd.No:PG232406030)

Under the Guidance of

Mr. CH.S.S. RAJESH PATNAIK

Assistant Professor

Department of Computer Science

GAYATRI VIDYA PARISHAD

COLLEGE FOR DEGREE AND PG COURSES

(Autonomous)

(Affiliated to Andhra University)

Rushikonda, Visakhapatnam.

2023-2025

GAYATRI VIDYA PARISHAD

COLLEGE FOR DEGREE AND PG COURSES (A)

(Affiliated to Andhra University)

Rushikonda, Visakhapatnam

Department of Computer Science

CERTIFICATE

This is to certify that the project report titled "**AI-Powered IoT System for Plant Disease Detection and Fertilizer Precision Spraying"** is the bonafide record of project work carried out by **KOVIRI SANDEEP (**Regd. No. PG232406030**)** as a student of this college, during the academic year 2024-2025, in partial fulfillment of the requirement for the award of the degree of Master of Computer Science.

Project Guide
Mr CH.S.S. RAJESH PATNAIK

Director of MCA
Prof. I. S. Pallavi

External Examiner

DECLARATION

I, KOVIRI SANDEEP here by declares that the project report titled "**AI-Powered IoT System for Plant Disease Detection and Fertilizer Precision Spraying",** is an original work done at **Gayatri Vidya Parishad College for Degree and PG Courses (Autonomous), Visakhapatnam,** submitted in partial fulfillment of the requirements for the award of Master of Computer Science , Gayatri Vidya Parishad College for Degree and PG Courses (A), affiliated to Andhra University. I assure that this projectis not submitted in any other University or college.

KOVIRI SANDEEP

ACKNOWLEGMENT

I consider it as a privilege to thank all those who helped me a lot for successful completion of the project "**AI-Powered IoT System for Plant Disease Detection and Fertilizer Precision Spraying".**

I would like to thank **Principal of Gayatri Vidya Parishad College for Degree and PG Courses (A), Prof.K.S.Bose,** who has provided full-fledged lab and infrastructure for successful completion of my project.

I would like to thank **Prof. I. S. Pallavi, Director of MCA**, who has very obliged in responding to every request though she is busy with her hectic schedule of administration and teaching.

I would like to thank our ever-accommodating my project guide **Mr. CH.S.S. RAJESH PATNAIK, Assistant Professor**, **Department of Computer Science** who has very obliged in responding to every request though he is busy with her hectic schedule of teaching.

I thank all the **Teaching and Non-teaching staff** who has been a constant source of support and encouragement during the study tenure.

KOVIRI SANDEEP

AI-Powered IoT System for Plant Disease Detection And Fertilizer Precision Spraying

CONTENTS

ABSTRACT

ABSTRACT

This project develops an **IoT-based system** to detect plant diseases and automate fertilizer spraying. An **ESP32-CAM** captures plant images, analyzed by a **machine learning model** for disease detection. If detected, **stepper motors** and a **CoreXY mechanism** position the sprayer, while a **soil moisture sensor** ensures proper irrigation.

Real-time data **is** uploaded to a **cloud platform** like Blynk allowing farmers to monitor disease detection, spraying, and soil moisture remotely via a **mobile app or web dashboard**, improving efficiency and crop health.

INTRODUCTION

1. INTRODUCTION

`

1.1 Introduction to Internet of Things

The Internet of things (IoT) refers to the concept of extending Internet connectivity beyond conventional computing platforms such as personal computers and mobile devices, and into any range of traditionally "dumb" or non-internet-enabled physical devices and everyday objects. Embedded with electronics, Internet connectivity, and other forms of hardware (such as sensors), these devices can communicate and interact with others over the Internet, and they can be remotely monitored and controlled. The definition of the Internet of things has evolved due to convergence of multiple technologies, real-time analytics, machine learning, commodity sensors, and embedded systems. Traditional fields of embedded systems, wireless sensor networks, control systems, automation (including home and building automation), and others all contribute to enabling the Internet of things. In the consumer market, IoT technology is most synonymous with products pertaining to the concept of the "smart home", covering devices and appliances (such as lighting fixtures, thermostats, home security systems and cameras, and other home appliances) that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smart phones and smart speakers.

1.1.1 Applications

The extensive set of applications for IoT devices is often divided into consumer, commercial, industrial, and infrastructure spaces.

- **Consumer Applications**

A growing portion of IoT devices are created for consumer use, including connected vehicles, home automation, wearable technology (as part of Internet of Wearable Things (IoT), connected health, and appliances with remote monitoring capabilities.

Smart Home: IoT devices are a part of the larger concept of home automation, which can include lighting, heating and air conditioning, media and security systems. Long term benefits could include energy savings by automatically ensuring lights and electronics are turned off. A smart home or automated home could be based on a platform or hubs that

Control smart devices and appliances. There are also dedicated smart home hubs that aSre offered as standalone platforms to connect different smart home products and these include Amazon Echo, Google Home.

- **Commercial Applications**

Building and Home Automation: IoT devices can be used to monitor and control the mechanical electrical and electronic systems used in various types of buildings (e.g., public and private, industrial, institutions, or residential) in home automation and building automation systems. In this context, three main areas are being covered in literature:

- The integration of the Internet with building energy management systems in ordertocreate energy efficient and IOT-driven "smart buildings"
- The possible means of real-time monitoring for reducing energy consumptionandmonitoring occupant behavior.
- The integration of smart devices in the built environment and how they might toknow how to use in further applications

- **Industrial Applications**

Manufacturing: The IoT can realize the seamless integration of various manufacturing devices equipped with sensing, identification, processing, communication, actuation, and networking capabilities. Based on such a highly integrated smart cyber physical space, it opens the door to create whole new business and market opportunities for manufacturing. Network control and management of manufacturing equipment, asset and situation management, or manufacturing process control bring the IoT within the realm of industrial applications and smart manufacturing as well. The IoT intelligent systems enable rapid manufacturing of new products, dynamic response to product demands, and real-time optimization of manufacturing production and supply chain networks, by networking machinery, sensors and control systems together. Digital control systems to automate process controls, operator tools and service information systems to optimize plant safety and security are within the purview of the IoT. But it also extends itself to asset management via predictive maintenance, statistical evaluation,and measurements to maximize reliability. Smart industrial management systems can also be integrated with the Smart Grid, thereby enabling real-time energy optimization. Measurements,

Automated controls, plant optimization, health and safety management, and other functions are provided by a large number of networked sensors. The term industrial Internet of things (IoT) is often encountered in the manufacturing industries, referring to the industrial subset of the IoT. IoT in manufacturing could generate so much business value thatit will eventually lead to the Fourth Industrial Revolution, so the so- called Industry 4.0. It is estimated that in the future, successful companies will be able to increase their revenuethrough Internet of things by creating new business models and improve productivity, exploit analytics for innovation, and transform workforce. The potential of growth by implementing IoT may generate $12 trillion of global GDP by 2030.

- **Agriculture:** There are numerous IoT applications in farming such as collecting data on temperature, rainfall, humidity, wind speed, pest infestation, and soil content. This data can be used to automate farming techniques, take informed decisions to improve quality and quantity, minimize risk and waste, and reduce effort required to manage crops.For example, farmers can now monitor soil temperature and moisture from afar, and evenapply IoT-acquired data to precision fertilization programs.

- **Infrastructure Applications:** Monitoring and controlling operations of sustainable urban and rural infrastructures like bridges, railway tracks and on- and offshore wind-farms is a key application of the IoT. The IoT infrastructure can be used for monitoring any events or changes in structural conditions that can compromise safety and increase risk. The IoT can benefit the construction industry by cost saving, time reduction, better quality workday, paperless workflow and increase in productivity. It can help in taking faster decisions and save money with Real-Time Data Analytics. It can also be used for scheduling repair and maintenance activities in an efficient manner, by coordinating tasks between different service providers and users of these facilities. IoT devices can also be used to control critical infrastructure like bridges to provide access to ships. Usage of IoT devices for monitoring and operating infrastructure is likely to improve incident management and emergency response coordination, and quality of service, uptimes and reduce costs of operation in all infrastructure related areas.

1.1.2 Advantages of IoT:

- It can assist in the smarter control of homes and cities via mobile phones. It enhances security and offers personal protection.
- By automating activities, it saves us a lot of time.
- Information is easily accessible, even if we are far away from our actual location, and it is updated frequently in real time.
- Electric Devices are directly connected and communicate with a controller computer, such as a cell phone, resulting in efficient electricity use. As a result, there will be no unnecessary use of electricity equipment.
- Personal assistance can be provided by IoT apps, which can alert you to your regular plans.
- It is useful for safety because it senses any potential danger and warns users. For example, GM On Star, is a integrated device that system which identifies a car crash or accident on road. It immediately makes a call if an accident or crash is found.
- It minimizes human effort because IoT devices connect and communicate with one another and perform a variety of tasks without the need for human intervention.
- Patient care can be performed more effectively in real time without the need for a doctor's visit. It gives them the ability to make choices as well as provide evidence-based care.
- Asset tracking, traffic or transportation tracking, inventory control, delivery, surveillance, individual order tracking, and customer management can all be Made more cost-effective with the right tracking system.

1.1.3 Introduction to Blynk – A Powerful IoT Platform:

Blynk is a powerful Internet of Things (IoT) platform designed to help developers and hobbyists build applications that can control and monitor hardware devices remotely using smartphones. It allows easy interaction between physical devices such as microcontrollers (ESP32, Arduino, NodeMCU, etc.) and the Blynk mobile app, providing a seamless interface for real-time control and data visualization. The main advantage of Blynk lies in its user-friendly mobile app, which lets users design custom dashboards using widgets like buttons, sliders, charts, and gauges without writing any complex UI code..

The architecture of Blynk consists of three major components: the Blynk app (user interface), the Blynk server (cloud or local), and the Blynk library that runs on the hardware. When a user interacts with the mobile app, commands are sent to the server, which in turn communicates with the connected device through the internet. Each device is identified using a unique Auth Token generated by the app, which ensures secure communication between the cloud and hardware. Blynk also supports virtual pins, which allow you to assign specific functions or data streams within your code for smoother widget-device interaction.

One of Blynk's key strengths is its wide compatibility with different hardware platforms and connectivity methods (Wi-Fi, Ethernet, GSM), allowing it to fit into various applications including smart farming, home automation, environmental monitoring, and robotics. It also supports integration with other services such as IFTTT, Google Sheets, and HTTP APIs, enabling users to trigger cloud-based actions or store data externally. Developers can use the Blynk Console (web dashboard) to manage devices, users, and data from a central place. With built-in features for real-time notifications, automation, geofencing, and device sharing, Blynk makes it easy to scale from a small prototype to a large deployment. Its support for secure SSL connections, customizable apps, and graphical analytics adds to its robustness. Overall, Blynk provides a simple yet comprehensive platform to bring IoT ideas to life without the need for developing custom applications from scratch.

1.1.4 Introduction to Machine Learning

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models to unable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data.

I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

1.1.5 Categories of Machine Leaning

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modelling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modelling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for

more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

As its name suggests, supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Let's understand supervised learning with an example. Suppose we have an input dataset of cats and dog images. So, first, we will provide the training to the machine to understand the images, such as the shape & size of the tail of cat and dog. Shape of eyes, colour, height (dogs are taller, cats are smaller), etc. After completion of training, we input the picture of a cat and ask the machine to identify the object and predict the output. Now, the machine is well trained, so it will check all the features of the object, such as height, shape, color, eyes, ears, tail, etc., and find that it's a cat. So, it will put it in the Cat category.

1.1.6 Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, Al is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn. The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

1.1.7 Challenges in Machines Learning

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of Al as whole still has a long way to go. The reason behind is that ML. has not been able to overcome number of challenges. The challenges that ML is facing currently are-

Quality of Data - Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming Task - Another challenge faced by ML. models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of Specialist Persons - As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No Clear Objective for Formulating Business Problems - Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of Overfitting & Underfitting - If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of Dimensionality - Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in Deployment - Complexity of the ML model makes it quite difficult to be deployed in real life.

1.1.8 Applications of Machines Learning

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of Al and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML

- Emotion analysis
- Sentiment analysis

- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

1.1.9 Machine Learning Techniques

Artificial Intelligence (AI) and Machine Learning (ML) encompass a wide range of techniques and methodologies that enable machines to perform tasks that typically require human intelligence. These techniques are at the heart of many modern applications, from voice assistants and chatbots to advanced medical diagnostics and autonomous vehicles.

One of the fundamental techniques in AI is the use of algorithms to process and analyze data. These algorithms can be broadly categorized into traditional AI algorithms and those used in machine learning. Traditional AI algorithms include search algorithms, which are used to navigate through data to find specific information or solutions to problems. Examples include the A* search algorithm, which is used in pathfinding and graph traversal, and the minimax algorithm, which is used in decision-making processes, particularly in game theory and competitive environments.

Machine learning algorithms, on the other hand, are designed to enable systems to learn from data and improve their performance over time. These algorithms can be divided into three main types: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training a model on a labeled dataset, meaning that each training example is paired with an output label. The model learns to map inputs to the correct output based on this training data. Common algorithms used in supervised learning include linear regression, logistic regression, support vector

machines (SVM), and decision trees. Neural networks, a more advanced form of supervised learning, are particularly powerful for tasks involving complex and high-dimensional data, such as image and speech recognition.

Unsupervised learning deals with unlabeled data, meaning the algorithm tries to find patterns and relationships within the data without any explicit instructions on what to look for. Clustering algorithms, such as k-means and hierarchical clustering, are used to group similar data points together. Dimensionality reduction techniques, like principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE), are used to reduce the number of variables under consideration and to visualize high-dimensional data. Unsupervised learning is particularly useful for exploratory data analysis and for finding hidden patterns or intrinsic structures in the data.

Reinforcement learning is a type of machine learning where an agent learns to make decisions by performing actions in an environment to maximize some notion of cumulative reward. Unlike supervised learning, where the correct answer is provided during training, reinforcement learning relies on a reward signal to evaluate the actions taken by the agent. The agent receives feedback in the form of rewards or penalties and uses this feedback to learn the best strategy or policy to achieve its goals. This approach is highly effective in dynamic environments where the optimal actions are not always apparent and must be discovered through trial and error. Reinforcement learning has been successfully applied to a wide range of problems, including robotics, game playing, and autonomous driving.

Another critical technique in AI and ML is the use of neural networks, which are designed to simulate the way the human brain processes information. Neural networks consist of layers of interconnected nodes, or neurons, that process data in a hierarchical manner. The most basic form of a neural network is the feedforward neural network, where information flows in one direction from the input layer to the output layer. More complex architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have been developed to handle specific types of data and tasks. CNNs are particularly effective for image processing tasks due to their ability to capture spatial hierarchies in images, while RNNs are suited for sequential data, such as time series or natural language, because they can maintain information about previous inputs through their recurrent connections.

Deep learning, a subset of machine learning, refers to the use of neural networks with many layers (hence "deep") to model complex patterns in data. The increased depth of these networks allows them to learn more abstract and high-level features from raw data, which has led to significant advancements in fields such as computer vision, natural language processing, and speech recognition. Training deep learning models typically requires large amounts of data and computational resources, but the results have been groundbreaking, enabling the development of AI systems that can outperform humans in certain tasks.

Overall, the techniques in AI and ML are diverse and continually evolving, driven by advances in computational power, availability of data, and innovative research. These techniques form the backbone of intelligent systems that are transforming industries and shaping the future of technology.

1.1.10 Future Directions

Artificial Intelligence (AI) and Machine Learning (ML) have profoundly impacted numerous sectors, revolutionizing how we approach problem-solving and decision-making. Their influence extends across industries such as healthcare, finance, transportation, and entertainment, driving significant advancements and efficiencies. In healthcare, AI and ML are transforming diagnostic and treatment processes. AI-driven diagnostic tools can analyze medical images with remarkable accuracy, often outperforming human radiologists in detecting diseases like cancer. Machine learning models predict patient outcomes and recommend personalized treatment plans based on vast amounts of historical data. These technologies are also accelerating drug discovery by identifying potential drug candidates and predicting their efficacy, significantly reducing the time and cost involved in bringing new medications to market. Moreover, AI-powered wearable devices continuously monitor patients' vital signs, enabling early detection of health issues and timely medical interventions.

The financial sector has also witnessed substantial changes due to AI and ML. These technologies enhance fraud detection by identifying unusual patterns in transaction data, thereby preventing fraudulent activities. In trading, machine learning algorithms analyze market data to forecast stock prices and inform investment strategies, often executing trades at high speeds and with greater precision than human traders.

Furthermore, AI-driven chatbots and virtual assistants provide personalized customer service, handling routine inquiries and transactions, which frees up human agents for more complex tasks.

Transportation is another field greatly impacted by AI and ML. Autonomous vehicles, which rely on sophisticated machine learning models to navigate and make real-time decisions, promise to reduce accidents caused by human error and improve traffic flow. AI algorithms optimize logistics and supply chain management by predicting demand, managing inventory, and routing deliveries efficiently. These advancements lead to cost savings and enhanced customer satisfaction through faster and more reliable services.

In the entertainment industry, AI and ML are used to create personalized user experiences. Streaming services like Netflix and Spotify employ machine learning algorithms to analyze users' viewing and listening habits, recommending content tailored to individual preferences. AI-driven tools are also used in content creation, such as generating realistic graphics in video games or producing original music and scripts. Additionally, sentiment analysis tools gauge audience reactions to movies, shows, and advertisements, helping creators and marketers refine their content to better meet audience expectations.

The impact of AI and ML extends beyond these sectors, influencing areas such as education, agriculture, and environmental conservation. In education, adaptive learning platforms use machine learning to tailor educational content to students' individual learning styles and paces, improving learning outcomes. In agriculture, AI-powered systems monitor crop health, optimize irrigation, and predict yields, enhancing productivity and sustainability. Environmental conservation efforts benefit from AI's ability to analyze data from sensors and satellite images, tracking wildlife populations and detecting illegal activities like poaching and deforestation.

Looking to the future, AI and ML are poised to drive further innovations and societal changes. One key area of development is the advancement of explainable AI (XAI), which aims to make AI systems more transparent and understandable to humans. As AI systems become more complex, ensuring that their decision-making processes are interpretable and trustworthy is crucial, particularly in high-stakes domains like healthcare and finance. Another promising direction is the integration of AI with the

Internet of Things (IoT). IoT devices generate vast amounts of data, and AI can analyze this data to derive insights and make intelligent decisions in real-time. This synergy has applications in smart cities, where AI-powered systems manage energy consumption, traffic flow, and public safety, creating more efficient and livable urban environments.

AI and ML will also play a significant role in addressing global challenges such as climate change and pandemics. Machine learning models can predict climate patterns, optimize renewable energy sources, and improve disaster response strategies. In public health, AI can assist in monitoring disease outbreaks, developing vaccines, and managing healthcare resources more effectively. Ethical considerations and regulatory frameworks will be critical as AI and ML continue to evolve. Ensuring that these technologies are developed and deployed responsibly, with attention to issues such as bias, privacy, and job displacement, will be essential to maximizing their benefits while mitigating potential risks. Collaborative efforts between governments, industry, and academia will be necessary to create policies and standards that promote ethical AI development and use.

1.1.11 Challenges and Limitations of Machine Learning

Artificial Intelligence (AI) and Machine Learning (ML) have made significant strides in recent years, but they still face several challenges and limitations. Understanding these issues is crucial for developing more robust and ethical AI systems.

1. Technical Challenges

- Overfitting and Underfitting: One of the key challenges in ML is achieving a balance between overfitting and underfitting. Overfitting occurs when a model learns the noise in the training data rather than the underlying pattern, resulting in poor performance on new, unseen data. Underfitting happens when a model is too simple to capture the complexity of the data. Finding the right model complexity and regularization techniques is essential for optimal performance.
- Scalability: As datasets grow larger and more complex, scaling ML algorithms becomes challenging. Training models on massive datasets requires substantial computational resources and time. Ensuring that algorithms can efficiently handle large-scale data while maintaining performance is a significant hurdle.
- Data Quality and Quantity: The effectiveness of ML models depends heavily on

the quality and quantity of the data. Inadequate or biased data can lead to inaccurate or skewed predictions. Data preprocessing, cleaning, and augmentation are critical to ensure that models are trained on high-quality data that accurately represents the problem domain.

2. Data Privacy and Security

- Privacy Concerns: AI systems often require access to large amounts of personal data, raising concerns about data privacy. Ensuring that sensitive information is protected and used responsibly is a major challenge. Techniques like anonymization and federated learning are being developed to address privacy concerns, but balancing privacy with model effectiveness remains a complex issue.
- Data Security: AI systems are vulnerable to security threats such as adversarial attacks, where malicious inputs are designed to fool the model into making incorrect predictions. Ensuring the robustness of AI systems against such attacks and protecting them from data breaches is crucial for maintaining trust and security.

3. Ethical and Bias Issues

- Algorithmic Bias: AI and ML systems can inadvertently perpetuate or even exacerbate existing biases present in the training data. For instance, biased training data can lead to biased outcomes in areas such as hiring, law enforcement, and credit scoring. Identifying and mitigating biases in AI models is essential for promoting fairness and equity.
- Ethical Considerations: The deployment of AI systems raises various ethical concerns, including the potential for misuse, impacts on employment, and decision-making transparency. Ensuring that AI systems are designed and used ethically involves addressing questions about accountability, transparency, and the broader societal impact of AI technologies.

4. Interpretability and Explain ability

- Black-Box Nature: Many advanced ML models, particularly deep learning models, operate as "black boxes," meaning their internal decision-making processes are not easily interpretable. This lack of transparency can hinder trust and make it difficult to understand how decisions are made. Developing

techniques for model interpretability and explain ability is essential for ensuring that AI systems are transparent and their decisions can be understood and justified.

- Model Interpretability: For AI systems to be widely accepted and trusted, it is crucial that their predictions and decision-making processes are interpretable by humans. Efforts to enhance model interpretability involve creating methods and tools that provide insights into how models arrive at their conclusions, which is especially important in high-stakes domains like healthcare and finance.

5. Societal Impact and Public Perception

- Job Displacement: The automation of tasks through AI and ML can lead to job displacement, as machines and algorithms increasingly perform tasks previously done by humans. Addressing the impact on employment and developing strategies for workforce retraining and support are important for mitigating the negative effects of automation.
- Public Perception: Public perception of AI and ML can vary, with concerns about the potential misuse of technology and its impact on daily life. Building public trust involves transparent communication about how AI systems work, their benefits, and their limitations.

1.2 Introduction to AI-Powered IoT System for Plant Disease Detection and Fertilizer Precision Spraying

Agricultural productivity is heavily impacted by plant diseases, which can spread rapidly if not identified and treated early. Traditional methods of plant health monitoring often rely on manual inspection, which is time-consuming, labor-intensive, and prone to human error. As a result, there is a growing need for a more intelligent, automated, and efficient solution to safeguard crops and improve yield.

This project proposes an AI-powered IoT-based system that integrates an ESP32-CAM module for real-time plant image capture, Edge Impulse for machine learning-based disease detection, and a CoreXY mechanism controlled by a second ESP32 board for fertilizer precision spraying. By leveraging computer vision and wireless communication between the two ESP32 boards, the system can autonomously detect plant diseases, identify affected areas, and precisely apply fertilizers or pesticides using pump-controlled relays. Additionally, soil moisture sensors enable conditional watering, and the entire system can be monitored remotely via cloud platforms.

➢ Key Features

1. **AI-Based Disease Detection**: Uses a trained convolutional neural network (CNN) model on Edge Impulse to identify plant leaf diseases in real time from camera images.
2. **CoreXY Mechanism for fertilizer Precision Spraying**: Employs stepper motors and a CoreXY system to move the spray nozzle precisely to the affected area, minimizing chemical waste.
3. **Dual ESP32 Architecture**: Separates processing tasks—ESP32-CAM handles detection while the main ESP32 controls motion and spraying—communicating over Wi-Fi.
4. **Smart Irrigation Control**: Incorporates a soil moisture sensor and water pump to automate watering only when needed.
5. **IoT Monitoring and Alerts**: Integrates data logging and real-time monitoring, optionally to a cloud platform, for performance tracking and remote alerts.

- ingtechnologies like ultrasonic, infrared, or magnetic sensors.
- **Real-time Data Collection**: Sensors continuously transmit data on parking space availability to a centralized cloud platform.
- **Cloud-based Management**: Data from sensors is processed and stored in the cloud, enabling real-time monitoring and analysis of parking occupancy.
- **Mobile App Integration**: Users can access real-time information on available parking spaces via mobile apps, allowing for convenient and efficient parking spot location.
- **Dynamic Pricing and Reservation**: Some systems offer features for dynamic pricing based on demand or allow users to reserve parking spots in advance.
- **Navigation and Guidance**: Actuators such as electronic signs or mobile app directions guide drivers to available parking spots, optimizing the parking process.
- **Analytics and Insights**: Utilizes data analytics to generate insights into parking patterns, which can inform urban planning decisions and optimize parking resource allocation.
- **Scalability and Flexibility**: Systems are scalable to accommodate varying parking lot sizes and can be integrated with existing infrastructure for seamless deployment.
- **Security and Privacy**: Ensures data security protocols are in place to protect sensitive information about parking availability and user locations.
- **Environmental Impact**: Reduces traffic congestion and emissions by minimizing the time spent searching for parking, contributing to a more sustainable urban environment.

1.2.1 Introduction to Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing. Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers-

students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments.

1.2.2 Why Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use forbeginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux.Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics Designers and architects build interactive prototypes, musicians and artists use it for installations andto experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn newthings. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step-by-step instructions of a kit, or sharing ideas online with other members of the Arduino community. There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Net media's BX-24, Phid gets, MIT's Handy board, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage forteachers, students, and interested amateurs over other systems:

- **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50

- **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX,

and Linux operating systems. Most microcontroller systems are limited to Windows.

- **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

1.2.3 ESP32

The ESP32 microcontroller has emerged as a cornerstone in the realm of embedded systems and IoT applications. Boasting a dual-core processor and robust wireless connectivity via Wi-Fi and Bluetooth, it offers a powerful platform for developers to create innovative solutions. Its extensive array of peripherals—including GPIO pins, UART, SPI, I2C, ADC, DAC, and more—facilitates versatile interfacing with various sensors, actuators, and other components. This versatility, coupled with low power consumption and advanced security features, makes the ESP32 well-suited for applications ranging from smart home devices and industrial automation to wearable technology and sensor networks. Supported by a vibrant development ecosystem that includes Express if's ESP-IDF and Arduino IDE, the ESP32 continues to drive advancements in connectivity, efficiency, and functionality across diverse industries.

- Different ESP32 modules

  - ESP32-WROOM-DA
  - ESP32-WROOM-32E
  - ESP32-WROOM-32UE
  - ESP32-WROOM-32D
  - ESP32-WROOM-32U
  - ESP32-SOLO-1
  - ESP32-WROVER-E
  - ESP32-WROVER-IE

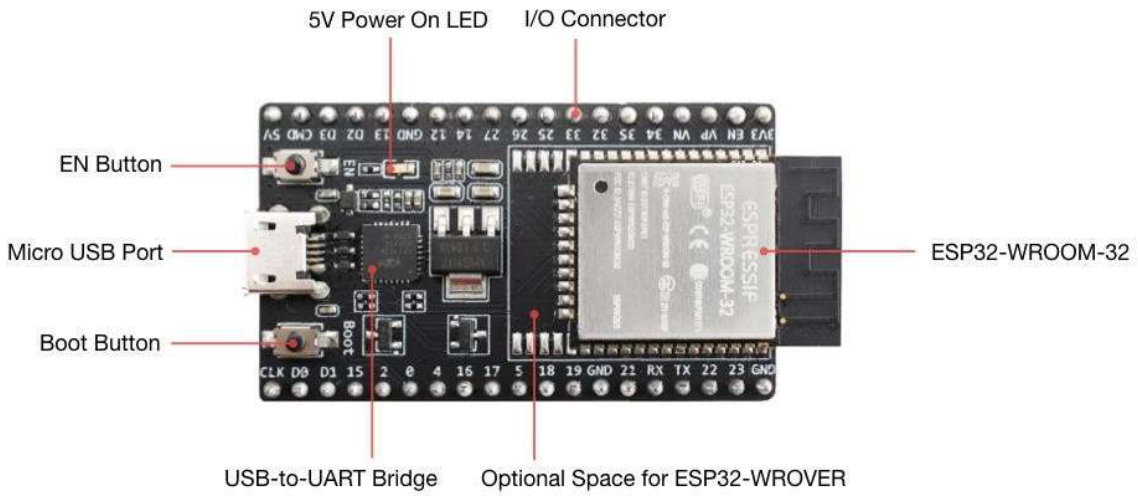

Figure 1.1: ESP32

1.2.4 Pin Configuration of ESP32

➢ **Power Pins**:

- **VIN**: Input voltage range is 5V. Used to power the ESP32 module.
- **3V3**: Output 3.3V voltage regulator. Used to power external components.
- **GND**: Ground pins for power and signal ground.

➢ **Analog Pins (ADC)**:

- **ADC1_0** to **ADC1_8**: Analog input pins with 12-bit resolution.
- **ADC2_0** to **ADC2_10**: Additional analog input pins with 12-bit resolution.

➢ **Digital GPIO Pins**:

- **GPIO0** to **GPIO39**: General purpose digital input/output pins. These can be configured as input or output, with support for internal pull-up or pull-down resistors.

➢ **Special Function Pins**:

- **UART Pins (TX, RX)**: Typically GPIO1 and GPIO3 or GPIO17 and GPIO16.
- **SPI Pins (SCK, MOSI, MISO, CS)**: Typically GPIO18, GPIO23, GPIO19, and GPIO5 respectively.
- **I2C Pins (SDA, SCL)**: Typically GPIO21 (SDA) and GPIO22 (SCL).
- **PWM Output Pins**: Available on GPIO0, GPIO2, GPIO4, GPIO12, GPIO13, GPIO14, GPIO15, GPIO25, GPIO26, GPIO27, etc.
- **Touch Sensor Pins**: Capacitive touch sensor inputs on GPIO4, GPIO0, GPIO2,

GPIO15, GPIO13, GPIO12, GPIO14, GPIO27, GPIO33, GPIO32.

➢ **Other Pins**:

- **EN**: Enable pin to power on the ESP32 module.
- **BOOT**: Boot mode selection pin.
- **IO0**: Input/output pin used for programming and boot mode selection.
- **IO33, IO32**: Additional digital input/output pins.

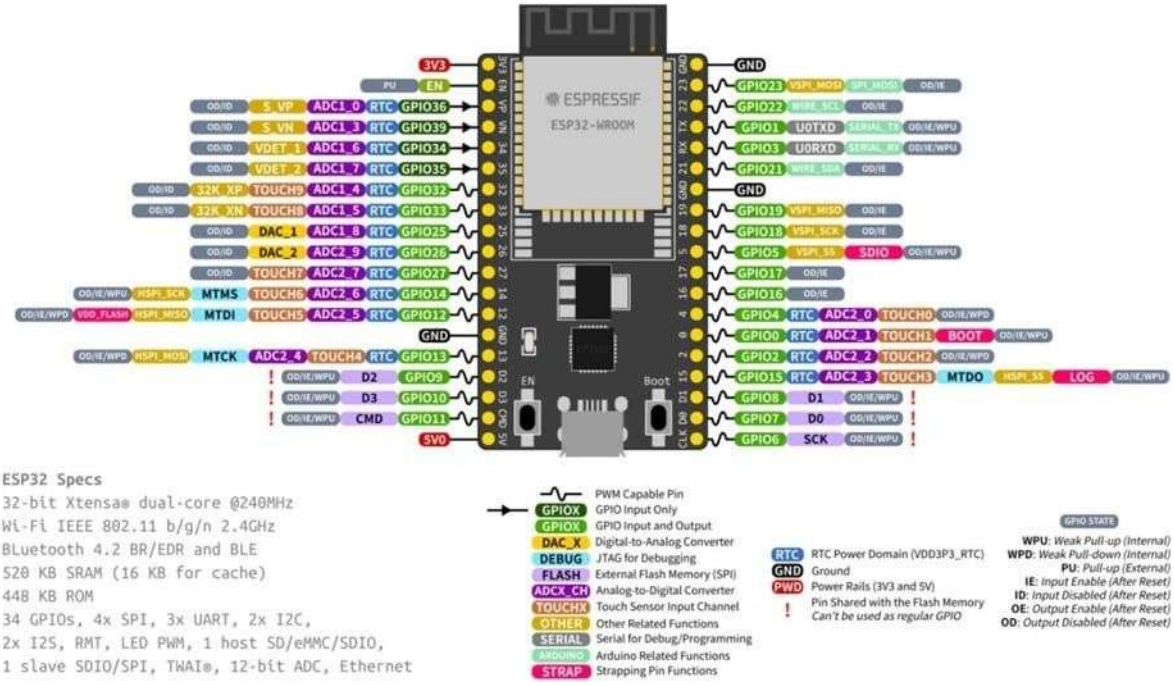

Figure 1.2: Pin Configuration ESP32

1.2.5 Siren:

Figure 1.3: Siren

- A siren is an electromechanical or electronic acoustic device that produces a loud sound to signal alerts or warnings. Sirens are commonly used in industrial, safety, and emergency systems, and are increasingly adopted in smart IoT-based systems for audible alerts. Traditionally used in fire alarms and emergency vehicles, modern sirens are now integrated into automated monitoring and response setups.
- The siren operates by converting electrical energy into sound energy, typically using a diaphragm or speaker mechanism. When activated through a control signal (usually from a microcontroller like ESP32), the siren emits a loud tone that draws immediate attention. In smart systems, it is triggered based on sensor feedback or alert conditions. The pitch, duration, and pattern of the sound can be modulated for different types of alerts. Sirens in embedded systems are compact, energy-efficient, and capable of delivering high-decibel outputs reliably.
- Sirens offer multiple advantages in automated agriculture systems, including immediate audio alerting of users in case of abnormal events such as detection of diseased plants, low soil moisture, or system malfunctions. This enhances real-time responsiveness and safety in field operations. However, care must be taken to manage power supply and triggering logic to avoid false alarms or unnecessary noise pollution.

Working:

In the project, the siren is controlled by the ESP32. When a diseased plant is detected or a specific condition is met (e.g., extreme dryness), the ESP32 activates the siren via a digital output pin. This serves as an audible indicator to notify nearby users or farmers for manual intervention or attention. The siren remains on for a programmed duration or until the situation is resolved.

1.2.6 ESP32-CAM

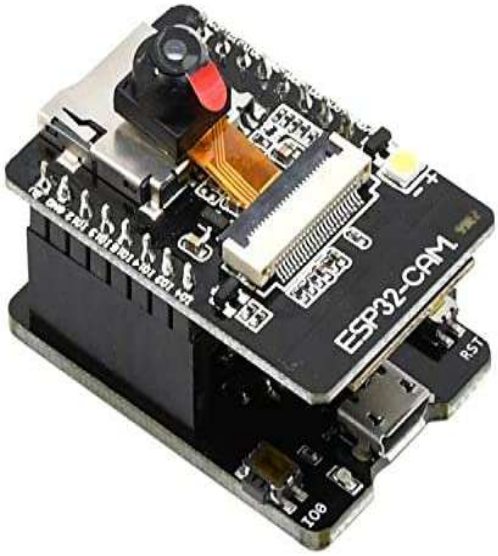

Figure 1.4: ESP32-CAM

The ESP32-CAM is a compact development board that integrates an ESP32-S chip, OV2640 camera module, microSD card slot, and Wi-Fi/Bluetooth connectivity. It's ideal for AIoT applications such as image classification, object detection, and surveillance. It has limited GPIOs as many are used internally for the camera.

Key Pin Configuration:

- GPIO 0: Used to flash new code (must be pulled LOW during programming)
- GPIO 1 (U0TXD) & GPIO 3 (U0RXD): UART for serial communication (used during programming)
- GPIO 16 & GPIO 17: Connected to the microSD card
- GPIO 12, 13, 14: Used for internal PSRAM
- GPIO 4, 5, 15: Can be used as general-purpose IO
- GPIO 33, 34, 35, 36, 39: Input-only pins
- GPIO 32: Can be used as input or output
- 3.3V and GND: Power supply pins

The module requires a 5V input and uses onboard voltage regulators. The onboard camera (OV2640) connects directly to the ESP32 through SCCB/I2C and parallel lines. Care must be taken while using GPIOs as many are shared with the camera and flash. It's a powerful board suitable for edge AI and IoT image-based projects.

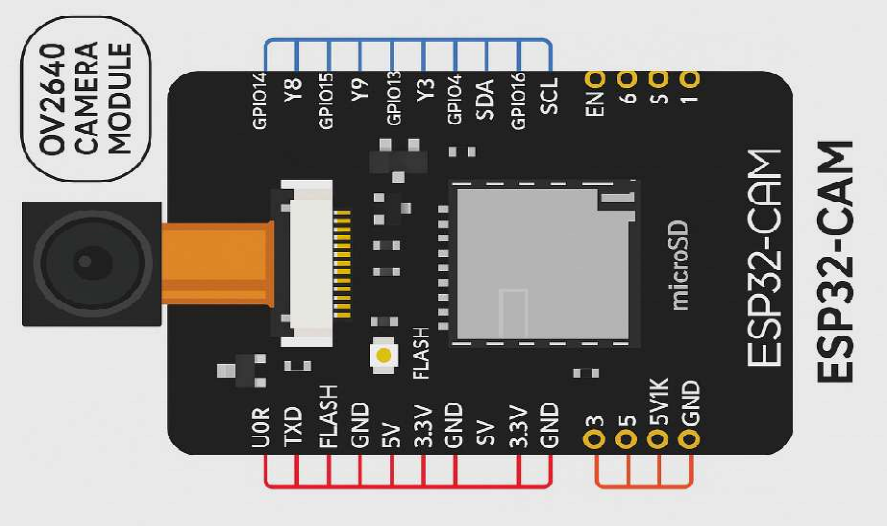

Figure 1.5: Working ESP32-CAM

1.2.7 **Stepper Motor**

A stepper motor is a type of brushless DC motor that divides a full rotation into a large number of equal steps. Unlike regular DC motors, which spin continuously when power is applied, stepper motors move in discrete steps. This makes them ideal for applications requiring precise position control.

There are two main types of stepper motors: unipolar and bipolar.

- **Working Principle:**

A stepper motor operates by energizing coils in a specific sequence, which causes the rotor (magnetized shaft) to align with the energized stator teeth.

Each pulse of current causes the motor to rotate by a precise angle, known as the step angle.

This makes stepper motors excellent for applications requiring accurate angular movement, such as moving a camera module or sprayer nozzle along a track (e.g., your CoreXY setup).

- **Stepper Motor Pins:**

A typical bipolar stepper motor has 4 wires that connect to two internal coils:

- Coil A → A+, A−
- Coil B → B+, B−

These are connected to a motor driver like the A4988.

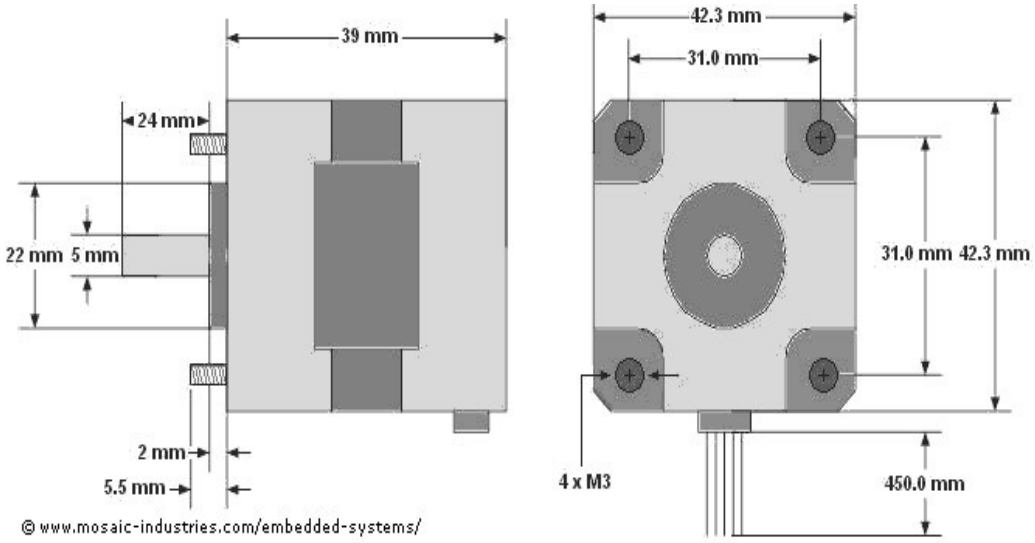

Figure 1.6: Stepper motor

1.2.8 Soil Moisture Sensor

A soil moisture sensor is an electronic device used to measure the volumetric water content in soil. It is widely used in smart farming and gardening projects to monitor the hydration level of plants and trigger actions such as automatic watering.

There are two common types:

Analog soil moisture sensor (basic resistive type)

Capacitive soil moisture sensor (more reliable, corrosion-resistant)

You are likely using the capacitive soil moisture sensor for better durability in long-term applications.

- **Pin Configuration (for Capacitive Soil Moisture Sensor – V1.2 or similar):**

VCC – Power Supply (typically 3.3V or 5V from ESP32 board)
GND – Ground
AOUT – Analog Output (voltage varies with moisture level)

- **Working Principle:**

The capacitive sensor measures soil moisture by detecting changes in the dielectric constant of the soil (which varies with water content).

When the soil is dry, the output voltage is higher.

When the soil is wet, the output voltage drops.

The ESP32 reads this analog value using its ADC (Analog to Digital Converter) to determine moisture levels.

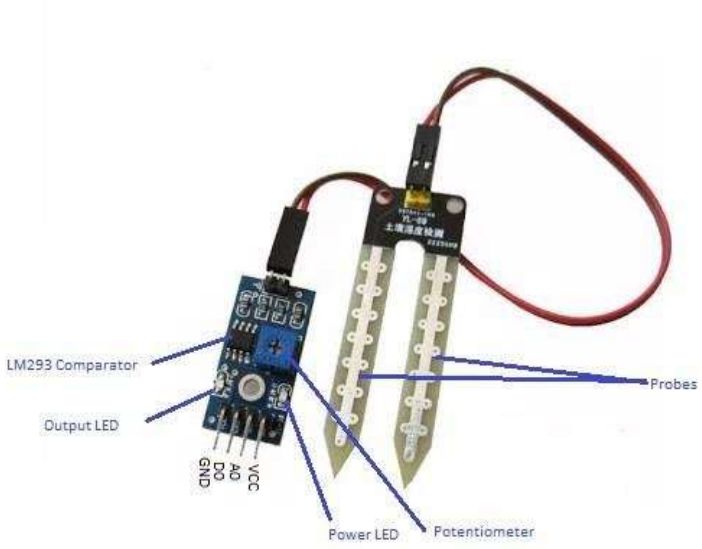

Figure 1.7: Soil Moisture Sensor

- The sensor is placed near the plant roots.
- It continuously monitors soil moisture.
- If the moisture level drops below a threshold, the ESP32 triggers a relay that activates a water pump to irrigate the plant.
- This automation ensures healthy plant growth and saves water.

1.2.9 Relay module:

A relay module is an electrically operated switch that allows a low-power microcontroller like an ESP32 or Arduino to control high-power AC or DC loads such as motors, lights, or pumps. When a digital HIGH or LOW signal is sent from a microcontroller to the IN1/IN2 pin of the relay module, it energizes the internal electromagnet, toggling the mechanical switch inside the relay. This process creates a physical click and allows the connected load to be powered ON or OFF.

Relay modules often include opto-isolators to protect the microcontroller from voltage spikes, enhancing safety. A single relay module will have one control input,

while dual-channel modules have IN1 and IN2 to control two separate loads. The relay's switching terminals are typically labeled as NO (Normally Open), NC (Normally Closed), and COM (Common). When the relay is inactive, COM is connected to NC. When activated, COM connects to NO, thus completing the circuit. These modules are essential in home automation, IoT-based agriculture systems (like yours for controlling pumps), and industrial automation where isolation and control of high-voltage devices are required.

Figure 1.8: relay module

Pin Configuration of a 2-Channel Relay Module:

VCC – Power supply input pin (typically 5V or 3.3V, depending on the relay type).

GND – Ground connection.

IN1 – Signal input pin to control Relay 1 (connected to microcontroller digital pin).

IN2 – Signal input pin to control Relay 2 (for dual-channel module).

1.2.10 Jumper Wires

Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with breadboards and other prototyping tools in order to make it easy to change a circuit as need though jumper wires come in a variety of colors, the colors don't actually mean anything. This means that a red jumper wire is technically the same as a

black one. But the colors can be used to your advantage in order to differentiate between types of connections, such as ground or power.

Figure 1.9: Jumper Wires

1.2.11 DHT11 Temperature and Humidity Sensor:

The DHT11 is a widely used digital sensor that measures both ambient temperature and relative humidity. It's cost-effective, compact, and ideal for weather monitoring, greenhouse control, and smart agriculture systems.

- **Working Principle:**

  The sensor uses a thermistor to measure temperature and a capacitive humidity sensor to detect moisture in the air. The data is then transmitted digitally to a microcontroller like the ESP32.

- **Pin Configuration (Typical 3-Pin Version of DHT11):**

  VCC – Power Supply (3.3V to 5V)
  DATA – Serial Data Output (connect to a digital pin on ESP32)
  GND – Ground

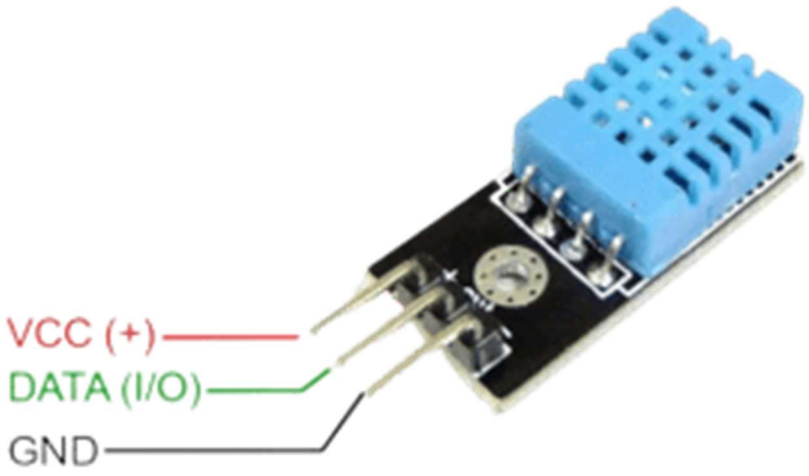

Figure 1.10: DHT11

- **Features and Advantages:**

Measures temperature: 0–50 °C (±2 °C accuracy)
Measures humidity: 20–90% RH (±5% accuracy)
Digital output – no need for ADC
Low power consumption
Simple to integrate with libraries like DHT.h in Arduino

In an IoT-based agriculture system, the DHT11 can monitor environmental conditions that affect plant health. You can log data to the cloud or use it to control irrigation and ventilation automatically.

1.2.12 Mini Water Pump:

A water motor pump in IoT-based automation projects is typically controlled using a relay module, as it operates on higher voltages (e.g., 12V or 24V) and cannot be directly driven by microcontrollers like ESP32. The relay acts as a switch that allows the low-power signal from the microcontroller to turn the pump ON or OFF.

- **Pin Configuration (when used with a relay module):**

VCC: Connects to the 5V or 3.3V pin of the microcontroller to power the relay module.

GND: Ground connection to complete the circuit**.**

IN (Input): Signal pin connected to any GPIO pin of the microcontroller (e.g., D25 or D26 of ESP32) to trigger the relay**.**

Relay Output (NO**, NC, COM):** These terminals are used to connect the pump.

COM **(Common):** Connects to one terminal of the pump power supply.

NO (Normally Open): Connects to the pump's positive terminal; the pump runs when the relay is triggered.

NC (Normally closed): Used if the pump should stay ON by default (rare in plant watering systems).

- **Working:**

When the ESP32 sends a HIGH signal to the relay's IN pin, the relay switches ON, connecting the power supply to the water motor and activating it. In this we are using two motor pump's one for fertilizer spraying and another for irrigation. This mechanism allows automatic control of watering based on inputs such as soil moisture and plant disease detection. Always ensure proper isolation between the pump's power circuit and the microcontroller using an opto-isolated relay module for safety.

Figure 1.11: Water pump

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 INTRODUCTION:

Agriculture remains the backbone of many economies, yet it faces persistent challenges in optimizing crop health and productivity. One of the major issues is the timely detection and management of plant diseases, which, if not identified early, can lead to substantial crop losses. Traditional manual monitoring methods are often labor-intensive, time-consuming, and prone to human error. With the advancement of Artificial Intelligence (AI) and the Internet of Things (IoT), it is now possible to automate these processes efficiently. This project proposes an AI-powered IoT system that leverages an ESP32-CAM module integrated with a TinyML model trained via Edge Impulse for real-time plant disease detection. The system can analyze captured images of leaves to determine whether a plant is healthy or infected. Upon identifying a diseased plant, the system triggers an automated mechanism to spray fertilizers or pesticides at the detected location. Additionally, a soil moisture sensor monitors hydration levels, activating a water pump when required. CoreXY mechanisms, controlled via stepper motors and A4988 drivers, guide the sprayer precisely. The system logs data on the Blynk cloud for remote monitoring and control. This integration reduces manual labor, ensures efficient resource utilization, and enhances crop yield. The solution offers scalability for large farms and precision for small gardens. Its ability to work offline with on-device AI inference makes it suitable even in low-connectivity rural areas. In essence, this project bridges modern AI with smart farming practices, promoting sustainable agriculture.

Circuit Diagram:

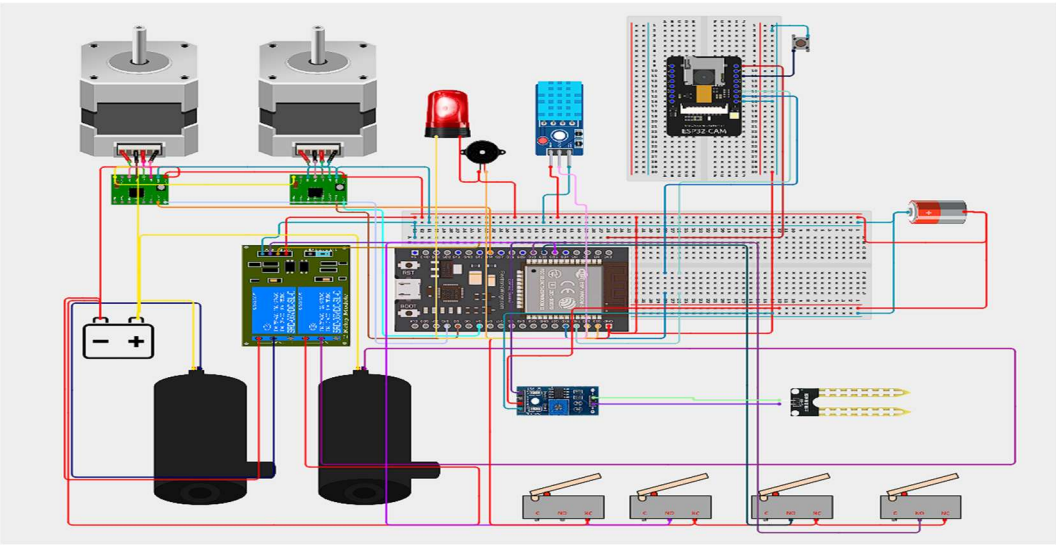

Figure 2.1: Circuit Diagram

2.2 History

The use of technology in agriculture has evolved significantly over the past decades. Initially, plant disease management relied solely on the experience and manual observation of farmers, often leading to inaccurate diagnosis and delayed treatment. As technology advanced, researchers began exploring image processing techniques using digital cameras and computer vision to identify visible symptoms of diseases. However, these methods required manual data transfer and lacked portability. The introduction of mobile applications provided some relief but still depended heavily on internet connectivity and server-side AI processing. Around 2015, the concept of Precision Agriculture began to gain momentum, integrating IoT sensors for environmental data collection. By the late 2010s, Tiny Machine Learning (TinyML) enabled the deployment of lightweight AI models directly on edge devices like microcontrollers. Simultaneously, platforms like Edge Impulse democratized the training and deployment of ML models for embedded systems. The ESP32-CAM, a low-cost Wi-Fi-enabled microcontroller with camera capabilities, became popular among developers for AI + IoT projects. In the recent past, farmers have started adopting these technologies to monitor soil, weather, and crop conditions. However, combining all these advancements into a compact, cost-effective, automated system for both disease detection and precision treatment has remained a challenge. This project emerges at the intersection of these developments, offering a solution that incorporates real-time edge AI, cloud monitoring, and automated actuation for disease management and irrigation.

2.3 Survey

Several research papers and studies have explored plant disease detection using image processing techniques and AI models. A study published in the International Journal of Computer Applications used a convolutional neural network (CNN) to classify tomato plant diseases with over 90% accuracy. However, the system required a high-end computer for processing and was not feasible for field deployment.

A project implemented using Raspberry Pi and OpenCV libraries achieved decent results in capturing and analyzing leaf images. While the Raspberry Pi offered local processing, the overall system was expensive and required more power, making it less ideal for battery-powered or remote agricultural setups.

In 2021, a team introduced an IoT-based smart irrigation system using soil moisture sensors and cloud-based decision-making. Though efficient in water usage, the system did not incorporate disease detection or AI-based analysis. Thus, it lacked comprehensive plant health monitoring.

Edge Impulse has been increasingly used by hobbyists and developers for deploying machine learning models on microcontrollers. Many prototypes focused on gesture recognition or object detection using the ESP32-CAM but did not fully explore agricultural applications such as disease recognition and spraying systems.

Current market solutions like drones for pesticide spraying are costly and not accessible to small-scale farmers. Our system fills this gap by offering an affordable, scalable, and efficient disease detection and precision spraying platform, making smart farming practical even for low-budget deployments**.**

2.4 Problem Statement

Traditional agricultural practices struggle with early detection and precise treatment of plant diseases, especially over large farming areas. Manual inspection is not only time-consuming but often inaccurate, leading to delayed action and reduced crop yields. The overuse or mistargeted use of fertilizers and pesticides further worsens the situation by damaging healthy plants and the environment. Additionally, inconsistent irrigation due to lack of moisture sensing causes either dehydration or waterlogging of crops. There is a pressing need for a cost-effective, automated system that can monitor plant health in real-time, identify diseased areas accurately, and apply treatment precisely. The system should also optimize water usage and be accessible for small to medium-scale farmers. Moreover, it must function reliably in rural areas with minimal connectivity..

2.5 Existing System

The existing systems for plant monitoring are largely manual or semi-automated. Some solutions use drones or mobile apps to capture images of crops, but these require significant human intervention, high costs, and stable internet connectivity for processing and analysis. Other systems offer only irrigation control based on soil moisture without disease detection. Standalone AI systems are not tailored for agriculture and often lack mobility or precision. Furthermore, existing setups may not support real-time response, precise treatment, or integration with cloud services for remote access.

Limitations:

High Cost and Complexity: Many existing plant disease detection systems are expensive and involve complex components or setups, making them unsuitable for small-scale or rural farmers.

Dependence on Internet and High-End Hardware: Most systems require continuous internet access and powerful processing units, which are not always available or affordable in rural or remote areas.

Low Accessibility for Non-Technical Users: The technical knowledge required to operate these systems creates a barrier for farmers who lack expertise in electronics or software.

Lack of Real-Time Spraying Action: Current solutions typically do not include real-time precision spraying mechanisms, which are essential to control the disease as soon as it is detected.

Poor Scalability and Energy Inefficiency: Many systems are not optimized for scalability or low power consumption, making them less viable for long-term use in the field.

2.6 Proposed System

The proposed system introduces an AI-powered, IoT-enabled smart farming prototype designed for real-time plant disease detection and automatic response. Using the ESP32-CAM and Edge Impulse's TinyML capabilities, the system captures leaf images and performs on-device classification to detect diseases. If disease is detected, an ESP32-controlled CoreXY system positions the sprayer at the exact location to apply fertilizer or pesticide. A soil moisture sensor continuously checks the hydration level, triggering a water pump when needed. All activity and sensor data are sent to the Blynk cloud platform, allowing remote monitoring and control via smartphone. The entire setup runs on affordable components and can be solar-powered for field use.

Advantages:

- Accurate detection using trained ML models.
- Automated spraying reduces human labor.

- Precision targeting minimizes pesticide wastage.
- Remote monitoring via Blynk enhances accessibility.

2.7 Requirement analysis

The system's design requires analyzing both software and hardware requirements to ensure reliable operation in field conditions. At its core, the ESP32-CAM handles image acquisition and disease detection through a lightweight AI model. A second ESP32 controls movement of the CoreXY mechanism and acts based on the classification result. Real-time data such as soil moisture levels are gathered and fed into the system to decide irrigation needs. Relay modules are used to control pumps that spray fertilizer or water the plants. The stepper motors and their A4988 drivers provide accurate control of the sprayer's movement. Blynk's cloud platform serves for remote interaction and visualization. Edge Impulse is used for model training and code export. Adequate power management, potentially through batteries or solar panels, is vital. The system must also account for environmental robustness such as protection from dust, rain, and high temperatures.

2.7.1 Functional Requirements

The functional requirements for the project outline the specific behaviour and functionalities that the system will exhibit to meet user needs and achieveits intended purpose. These requirements focus on what the system should do, describing the interactions between the system and its users, as well as the system's operations.

INPUT:

Leaf Image Capture: High-resolution images of plant leaves captured using the ESP32-CAM module. These images are used as input for disease classification using the onboard TinyML model.

• **Disease Detection via ML Model:** Trained machine learning model from Edge Impulse processes captured images on-device to classify plant health (e.g., Healthy or Diseased with Black Spots).

• **Soil Moisture Sensor Reading:** Real-time soil moisture data is collected to determine whether the plant requires watering, helping to avoid under- or over-irrigation.

• **Limit Switch Signals**: Inputs from limit switches placed at the edges of the CoreXY frame to prevent stepper motors from moving beyond physical boundaries and to reset system to home position.

• **ESP32-CAM to ESP32 Communication:** Wi-Fi-based communication from the camera unit to the main ESP32 controller, transferring detection results and triggering movement commands for the CoreXY system.

• **User Control via Blynk App:** Optional user commands via the Blynk mobile app to view system status, override actions, or initiate manual watering/spraying.

OUTPUT:

- **Disease Classification Result:** Output from the machine learning model indicating whether the detected plant leaf is healthy or infected, and optionally its location.
- **CoreXY Movement Control:** Directional and positional movement commands sent to stepper motors via A4988 drivers to align the sprayer accurately above the target plant.
- **Fertilizer or Water Pump Activation:** Based on disease detection or soil moisture status, relays are activated to control pumps for spraying fertilizer or watering the plant.
- **Real-time Notifications via Blynk:** Updates and alerts about plant status, disease detection, pump operation, and system faults sent to the user's mobile device.
- **LED/Buzzer Siren Activation:** Visual (LED) and audio (buzzer) indicators activated as a warning when disease is detected and until spraying is completed.
- **• Data Logging to Cloud:** Sensor readings, classification results, and device actions are uploaded to the Blynk cloud for remote monitoring and future analysis.

➢ **Hardware Requirements:**

- ESP32-CAM (for image capture and disease detection)
- ESP32 (for motion control and relay activation)
- A4988 Motor Drivers with Stepper Motors (2 Nos.)
- Soil Moisture Sensor
- Relay Modules (2-channel)
- Fertilizer & Water Pumps
- Limit Switches
- Power Supply / Battery / Solar Setup
- CoreXY Frame

➢ **Software Requirements:**

- Arduino IDE for firmware development
- Edge Impulse Studio for ML model training
- Blynk IoT Platform for remote monitoring
- Embedded C/C++ for ESP32 firmware

2.7.2 Non-Functional Requirements

Non-functional requirements describe the overall qualities and attributes of the Smart Pill Reminder system, focusing on how the system performs its functions rather than what specific functions it performs. These requirements ensure the system is usable, reliable, and efficient.

- **Performance**: The system should respond to user requests within 2 seconds. Update parking space availability status in real-time with a maximum latency of 5 seconds.

- **Scalability**: The system should support up to 10,000 concurrent users without performance degradation. Allow easy addition of new parking locations andsensors without major architectural changes.

- **Reliability**: Ensure 99.9% system uptime. Implement failover and redundancy mechanisms to handle server failures and ensure continuous operation.

- **Security**: Use end-to-end encryption for all data transmissions. Implement multi-factor authentication for user accounts.
- **Usability**: The user interface should be intuitive and user-friendly for both web and mobile applications.

UMLMODELING

3. UML MODELING

3.1 Introduction to UML

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software system. UML was created by the Object Management Group (OMG) and UML 1.0specification drift was proposed to the OMG in January 1997.OMG is continuously making efforts to create a truly industry standard.

- UML stands for Unified Modeling Language.
- UML is different from the other common programming language such as C++, java, etc.
- UML is a pictorial language used to make software blueprints.
- UML can be described as a general-purpose visual modeling language to visualize, specify, construct and document software system.
- Although UML is generally used to model software system, it is not limited within this boundary. It is generally used to model software system as well. For example, the process flows in a manufacturing unit, etc.
- UML is not a programming language, but tools can be used to generate code in various language using UML diagrams. UML has a direct relation with object-oriented analysis and design. After some standardization, UML has become an OMG standard.

3.2 Goals of UML

- UML aimed to create a universal modeling language for all types of systems, emphasizing simplicity and widespread applicability.
- UML diagrams cater not only to developers but also to business users, general audiences, and anyone interested in understanding complex systems.
- UML integrates with development processes rather than being a standalone methodology, enhancing system development and understanding.
- UML provides a straightforward approach to model and visualize both software and non-software systems in modern, intricate environments.

3.3 UML Standard Diagrams:

The elements are like components which can be associated in diverse ways to make a complete UML picture, which is known as diagram. Thus, it is very important to understand the different diagrams to implement the knowledge in real-life system. Any complex system is best understood by making some kind of diagrams or pictures. These diagrams have a better impact on our understanding. If we look around, we will realize that the diagram is not a new concept but it is used widely in different forms in different industries. We prepare UML diagram to understand the system in a better and simple way. A single diagram is not enoughto cover all the aspects of the system. UML defines various kinds of diagrams to cover most of the aspects of a system. You can also create your own set of diagrams to meet your requirements. Diagrams are generally made in an incremental and iterative way. There are two broad categories of diagram and they are again divided into subcategories:

- Structural Diagrams
- Behavioral Diagrams

3.3.1 Structural Diagrams

The structural diagram represents the static aspect of the system. These static aspects represent those parts of a diagram, which forms the main structure and are therefore stable. These static parts are represented by classes, interfaces, object, components, and nodes. The four structural diagrams are:

- Class Diagram
- Object Diagram
- Component Diagram
- Deployment Diagram

3.3.2 Behavioral Diagrams

Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered. Behavioral diagram captures the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts ofa system. UML has the following five types of behavioral diagrams:

- Use case Diagram
- Sequence Diagram
- Activity Diagram

3.3.3.1 Use Case Diagram:

Use case describes the behavior of the system as seen from the actor's point of view. A use case diagram can portray then different types of users of a system and the many ways that they interact with system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well. Actors initiate the use cases for accessing system's functionality. When actors and use cases exchange information, they are said to Communicate. To describe a use case, we use a template composed of six fields:

- **Use Case Name:** The name of the use case.
- **Participating Actors:** The actors participating in the particular use case.
- **Entry Condition:** Condition for initiating the use case.
- **Flow of events:** Sequence of steps describing the functioning of Use case.
- **Exit Condition:** Condition for terminating the use case.
- **Quality Requirements:** Requirements that do not belong to the use case but constraint the functionality of the system.

Use Case Diagram:

Use Case Diagram for AI-Powered IoT System for Plant Disease Detection and Fertilizer Precision Spraying

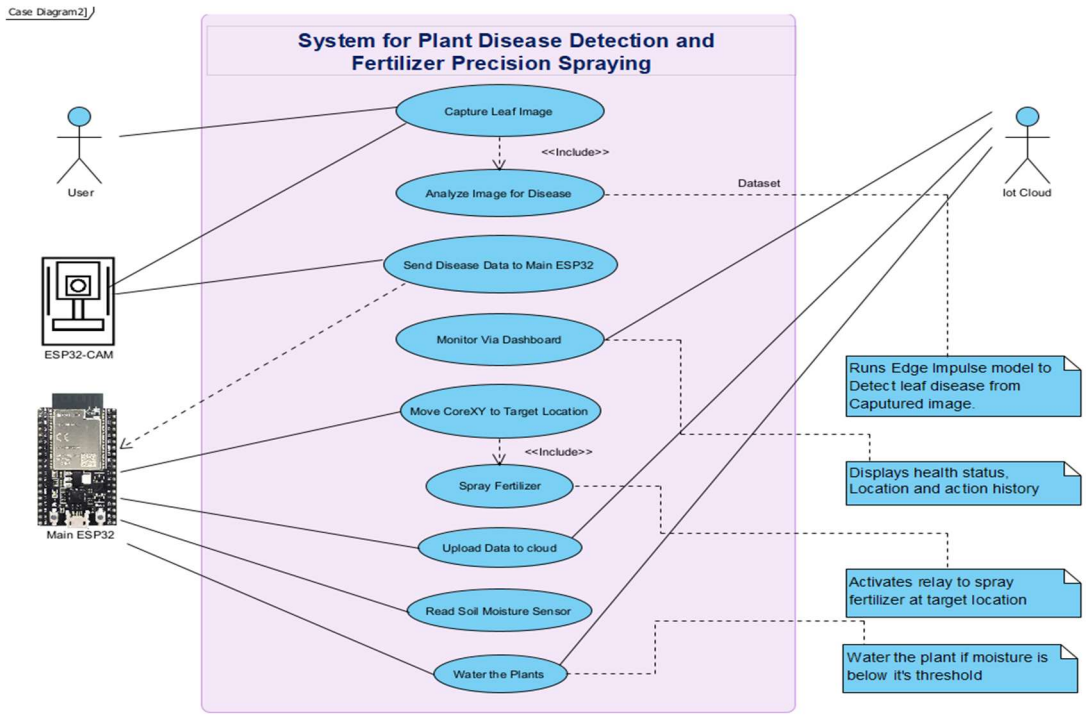

Figure 3.1: Use Case Diagram

Description:

The AI-Powered IoT System for Plant Disease Detection and Fertilizer Precision Spraying is designed to automate the identification and treatment of diseased plants. Using an ESP32-CAM module trained with an Edge Impulse TinyML model, the system captures images of plant leaves, analyzes them for signs of disease, and communicates results to a main ESP32 controller. The CoreXY mechanism then navigates the fertilizer nozzle to the affected plant location for precise spraying. A soil moisture sensor further enables irrigation control. Real-time updates are sent to the Blynk IoT cloud, providing users with live monitoring via a dashboard..

Actors:

The main actors are the User and the IoT System (comprising ESP32-CAM, Main ESP32, and Blynk Cloud). The User interacts with the system through a mobile dashboard to initiate plant scanning, monitor plant health, and receive real-time updates

regarding disease detection and watering status. The IoT System handles image capturing, disease classification using an AI model, controls the CoreXY mechanism for movement, and manages the operation of pumps and sensors. It also ensures the automatic uploading of health data and sensor readings to the Blynk cloud, providing the user with an intuitive monitoring experience and enabling remote management of the plant care system.

There are two main actors:

1. User
2. IoT Cloud (Blynk platform)

• **User**: The User initiates the scanning operation and views real-time plant health status through a mobile app connected to the Blynk dashboard. The User receives notifications regarding disease detection, fertilizer spraying, and irrigation updates.

• **IoT Cloud:** The Blynk IoT platform acts as a cloud dashboard to monitor system activity, receive disease classification results, and track sensor-based actions like watering. It also enables remote interaction and analytics logging.

Use Case for Capture & Analyze Image

Use Case ID	UC001
Use case Name	Capture & Analyze Image
Participating Actors	ESP32-CAM, User
Entry Condition	User starts scanning via mobile app or button trigger.
Flow of Events	• The ESP32-CAM captures the image of the plant leaf. • The captured image is processed locally using an Edge Impulse model. • The leaf is classified as healthy or diseased. • Results are sent to the main ESP32.
Exit Condition	The image analysis is completed and decision data is transferred.

Table 3.1: Capture & Analyze Image

Use case for Send Data and Monitor via Cloud:

Use Case ID	UC002
Use case Name	Send Disease Data and Monitor
Participating Actors	ESP32-CAM, ESP32, IoT Cloud
Entry Condition	Disease detection is completed.
Flow of Events	• ESP32-CAM sends disease classification data to main ESP32. • ESP32 uploads the data to the IoT Cloud via Wi-Fi. • The User receives real-time updates on Blynk mobile app.
Exit Condition	Data is stored and displayed in the Blynk cloud dashboard.

Table 3.2: Send Data and Monitor via Cloud

Use case for Movement and Spraying Fertilizer:

Use Case ID	UC003
Use case Name	Move CoreXY and Spray Fertilizer
Participating Actors	Main ESP32, Stepper Motors
Entry Condition	Disease-positive result is received
Flow of Events	• • Main ESP32 calculates the affected plant's coordinate. • CoreXY mechanism moves to the target location. • Relay 1 is activated to start the fertilizer pump. • Fertilizer is sprayed for a specified duration.
Exit Condition	Spraying is completed and location is logged.

Table 3.3: Movement and Spraying Fertilizer

Use case for Soil Moisture Detection and Watering:

Use Case ID	UC004
Use case Name	Detect Soil Moisture and Water
Participating Actors	Main ESP32, Soil Moisture Sensor
Entry Condition	Spraying completed or dry plant detected
Flow of Events	• Soil moisture sensor checks moisture level of plant base. • If level is below threshold, Relay 2 activates water pump. • Moisture readings are uploaded to cloud dashboard.
Exit Condition	Watering is completed if needed.

Table 3.4: Soil Moisture Detection and Watering

Use case for Access & Control via App:

Use Case ID	UC005
Use case Name	Access Control Mobile Dashboard Access Control
Participating Actors	User, IoT Cloud
Entry Condition	User logs in or opens Blynk app
Flow of Events	• User accesses dashboard via app • Views disease history, real-time monitoring and controls • Can trigger scan, reset or emergency stop manually
Exit Condition	System status and control are reflected in app interface

Table 3.5: Access & Control via App

3.3.2.2 Sequence Diagram:

Sequence diagrams depict interactions between objects in a system through a timeline of messages exchanged. Each message triggers operations within receiving objects, influencing their behavior and interactions with other objects. Arguments accompanying messages provide data necessary for executing operations, binding parameters to the receiving object's functions. These diagrams help visualize the flow of operations and communication paths, aiding in the understanding and design of complex system behaviors. Sequence diagrams are invaluable for clarifying the order of operations, dependencies between objects, and the overall dynamics of object-oriented systems during development and analysis phases.

Sequence diagram for AI-Powered IoT System for Plant Disease Detection and Fertilizer Precision Spraying

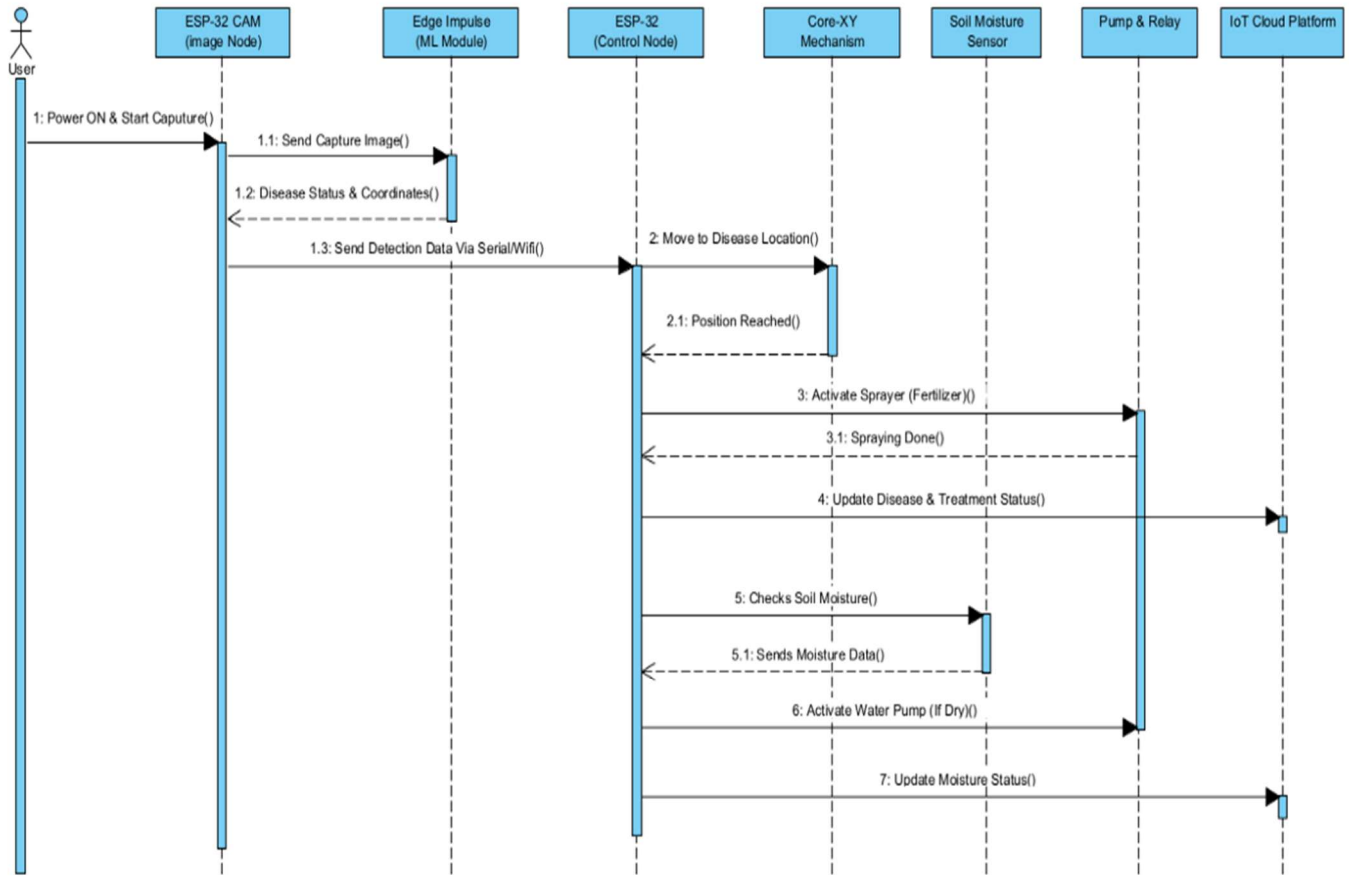

Figure 3.2: Sequence Diagram

Description:

The sequence diagram illustrates the real-time interaction flow within the AI-Powered IoT System for Plant Disease Detection and Fertilizer Precision Spraying. It begins with the user powering on the ESP32-CAM, which captures plant images and sends them to the Edge Impulse ML module for disease analysis. Upon detection, the disease status and coordinates are sent to the main ESP32 controller. The ESP32 then directs the CoreXY mechanism to move to the affected plant's location. Once the position is reached, it activates the relay to spray fertilizer, and then logs the treatment status to the cloud. Following spraying, the ESP32 reads soil moisture using a sensor. If the moisture level is below the threshold, the system activates the water pump to irrigate the plant. Finally, the updated moisture status is uploaded to the IoT cloud platform for remote monitoring and record keeping. This automated sequence ensures precise disease treatment and efficient water management

Activity Diagram:

An Activity Diagram in UML (Unified Modeling Language) is a type of behavioral diagram that represents the dynamic aspects of a system by showing the flow of control or data from one activity to another. It is widely used to model the workflow of a system, business process, or algorithm. The diagram begins with an initial node and proceeds through various actions or activities connected by control flows, ending at a final node. Activity diagrams often include decision nodes, merge nodes, forks, and joins to illustrate branching and parallel processes. In the context of an IoT-based plant disease detection system, an activity diagram helps visualize the sequence of tasks—such as capturing images, analyzing them, detecting disease, and activating actuators. It allows developers and stakeholders to understand how the system behaves in response to different inputs and states. Swim lanes can be used to assign responsibilities to different hardware or software components. This makes activity diagrams useful for both system design and debugging. Overall, they offer a clear visual representation of the system's logic and workflow.

3.3.2.4 Activity diagram:

Activity Diagram for AI-Powered IoT System for Plant Disease Detection and Fertilizer Precision Spraying

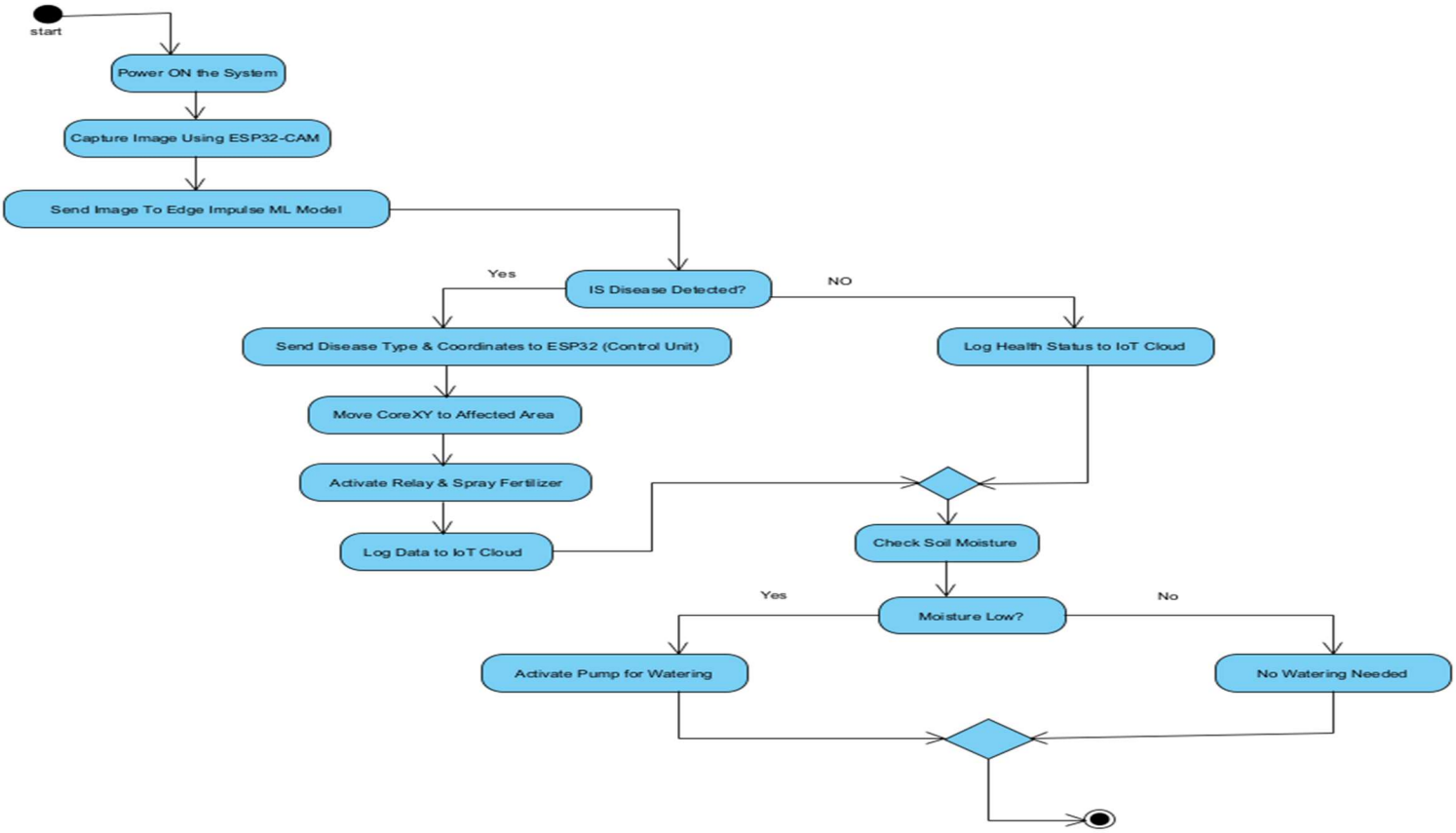

Figure 3.3: Activity Diagram

Description:

The activity diagram for the "AI-Powered IoT System for Plant Disease Detection and Fertilizer Precision Spraying" outlines the complete operational flow of the system. It begins with powering on the system, followed by capturing plant images using the ESP32-CAM. These images are then sent to the Edge Impulse machine learning model for analysis. If a disease is detected, the system sends the disease type and coordinates to the main ESP32 controller. The CoreXY mechanism is then activated to move to the infected plant location, and fertilizer spraying is initiated using a relay-controlled pump. The treatment details are logged to the IoT cloud platform. If no disease is found, the plant's health status is logged, and the system proceeds to check soil moisture levels. Based on the sensor reading, if the soil is dry, the pump is activated to water the plant; otherwise, watering is skipped. This diagram effectively represents the intelligent automation and decision-making process within the system.

DESIGN

4. DESIGN

System Design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. In System design, developers:

- Define design goals of the project
- Decompose the system into smaller sub systems
- Design hardware/software strategies
- Design persistent data management strategies
- Design global control flow strategies
- Design access control policies and
- Design strategies for handling boundary conditions. System design is not algorithmic. It is decomposed of several activities. They are:
- Identify Design Goals
- Design the initial subsystem decomposition
- Refine the subsystem decomposition to address the design goals.

System Design is the transformation and analyzes a model into a system design model. Developers define the design goals of the project and decompose the system into smaller subsystems that can be realized by individual teams. Developers also select strategies for building the system, such as the hardware/software platform on which the system runs, the result of the system design is model that includes a clear description of each of these strategies,subsystem decomposition.

4.1 Design Goals

Design goals are the qualities that the system should focus on. Many design goals can be inferred from the non-functional requirements or from the application domain.

- **Readable**: The arrangement of fonts and words in order to make written content flow in asimple and easy to read manner.
- **Usability:** This targets for the percentage of users who clearly understands the interface ofthe application
- **Reliable:** Ensures the system perform a specified function within a given system for andexpected lifecycle.
- **Reusable**: Reusing several components previously used makes it quicker and easier todesign a new project.
- **Extendable**: It is a measure of the ability to extend a system and the level of effort required to implement extension.
- **Flexible:** The capacity to support multiple functions without altering the architecture of the system.
- **Efficient:** Efficiency means that the project does not waste any resources like timeand memory in order to process the application.

4.2 Algorithm:

Step 1: Power on the system and initialize all hardware components:

- ESP32-CAM
- ESP32 (control board)
- CoreXY motion system
- Stepper motors, relays, pumps
- Soil moisture sensor

Step 2: System remains in HOME position and waits for a START command from the Blynk mobile/web dashboard.

Step 3: Upon receiving the START command from Blynk, system begins plant health inspection cycle.

Step 4: The ESP32-CAM and fertilizer nozzle move together to the first plant position.

Step 5: ESP32-CAM captures the image of the plant leaf.

Step 6: Captured image is processed locally using a TinyML model (Edge Impulse) to detect plant health status.

Step 7: If the plant is healthy: move directly to the next plant.

If the plant is diseased:

ESP32 receives coordinates and detection result

Fertilizer nozzle sprays only on the affected plant using Relay 1

Log the action to Blynk platform

Step 8: Repeat Steps 4 to 7 for each plant until the last plant is scanned.

Step 9: System checks soil moisture using the soil sensor.

- If soil is dry, activate water pump via Relay 2.
- Log moisture data to Blynk.

Step 10: Once all plants are scanned and necessary actions are taken, the camera and nozzle return to the HOME position.

Step 11: System waits in STANDBY mode for the next START command from Blynk to begin a new scan cycle.

❖ **Overview Screen of AI-Powered IoT System for Plant Disease Detection and Fertilizer Precision Spraying**

Figure 4.1: Overview

4.3 What makes up an Arduino?

ARDUINO IDE

You decided to go and buy yourself an Arduino, but once it arrived, you realized you have no idea what to do with it. Do not panic, for help is at hand! In this how-to, we will look at how to get started with Arduino microcontroller boards. We'll cover software

installation, as well as connecting and configuring the Arduino IDE.

You Will Need

- USB B Cable
- Windows 10, Windows 8, Windows 7, Mac, or Linux OS
- Arduino IDE

4.3.1 Download the IDE

First, you must download the IDE and install it. Start by visiting Arduino's software page .The IDE is available for most common operating systems, including Windows, Mac OS X, and Linux, so be sure to download the correct version for your OS. If you are using Windows 7 or older, do not download the Windows app version, as this requires Windows 8.1 Or Windows 10.

Once the installer has downloaded, go ahead and install the IDE. Chances are you will want to enable all options on the installer, including any USB drivers and libraries, but do make sure to read the EULA!

The Arduino IDE

The Arduino IDE is incredibly minimalistic, yet it provides a near-complete environment for most Arduino-based projects. The top menu bar has the standard options, including "File" (new, load save, etc.), "Edit" (font, copy, paste, etc.), "Sketch" (for compiling and programming), "Tools" (useful options for testing projects), and "Help". The middle section of the IDE is a simple text editor that where you can enter the program code. The bottom section of the IDE is dedicated to an output window that is used to see the status of the compilation, how much memory has been used, any errors that were found in the program, and various other useful messages.

Figure 4.2: Arduino IDE

The Arduino IDE in its default state.

Projects made using the Arduino are called sketches, and such sketches are usually written ina cut-down version of C++ (a number of C++ features are not included). Because programming a microcontroller is somewhat different from programming a computer, there are a number of device-specific libraries (e.g., changing pin modes, output data on pins, reading analog values, and timers). This sometimes confuses users who think Arduino is programmed in an "Arduino language." However, the Arduino is, in fact, programmed in C++.It just uses unique libraries for the device. The 6 Buttons while more advanced projects will take advantage of the built-in tools in the IDE, most projects will rely on the six buttons found below the menu bar.

The button bar

The **check mark** is used to verify your code. Click this once you have written your code.

1. The **arrow** uploads your code to the Arduino to run.
2. The **dotted paper** will create a new file.
3. The **upward arrow** is used to open an existing Arduino project.
4. The **downward arrow** is used to save the current file.
5. The far right button is a **serial monitor**, which is useful for sending data from the Arduino to the PC for debugging purposes.

There are plenty of other features available to consider on the IDE. But, having used many different types of microcontrollers and having been involved in multiple programming environments, it is shocking how simple the Arduino and its IDE is! In less than two minutes, you can get a simple C++ program uploaded onto the Arduino and have it.

CODING

5. CODING

The goal of coding or programming phase is to translate the design of the system produced during the phase into code in a given programming language, which can be executed by a computer and the performs the computation specified by the design.

The coding phase affects both testing and maintenance. The goal of coding is not to reduce the implementation cost, but the goal should be to reduce the cost of later phase. Inother words, the goal is not to simplify the job of programmer. Rather the goal should be to simplify the job of the tester and maintainer.

5.1 Coding Approach:

There are two major approaches for coding any software system. They are top-Down approach and bottom-up approach.

Bottom-up approach can suit for developing the object-oriented systems. During system design phase of reduce the complexity. We decompose the system into appropriate number of subsystems, for which objects can be modelled independently. These objects exhibit the way the subsystems perform their operations.

Once object have been modeled they are implemented by means of coding. Even though related to the same system as the objects are implemented of each other the Bottom-Up approach is more suitable for coding these objects.

In this approach, we first do the coding of objects independently and then we integrate these modules into one system to which they belong. In this project, top-Down approach is followed. For registration and Login. User will click and stores the intruder information intensely.

5.2 Information Handling:

Any software system requires some amount of information during its operation selection of appropriate data structures can help us to produce the code so that objects of the system can better operate with the available information decreased complexity.

In this project, Encryption and decryption will not be possible if the image fields are vacant. System will not have any default values. User must specify each secret file name in encryption and locate all required operations in decryption.

5.3 Programming Style:

Programming style deals with act of rules that a programmer must follow so that the characteristics of coding such as Traceability, Understands the ability, Modifiability, and Extensibility can be satisfied.

In the current system, we followed the coding rules for naming the variables and methods. The system is developed in a very interactive and users friendly manner.

5.4 Verification and Validation:

Verification is the process of checking the product built is right. Validation is the process of checking whether the right product is built. During the Development of the system coding for the object as been thoroughly verified from various aspects regarding their design,in the way they are integrated and etc. The various techniques that have been followed for validation discussed in testing the current system. Validations applied to the entire system at two levels.

5.5 Form level validation:

Validations of all the inputs given to the system at various points in the forms are validated while navigating to the next form. System raises appropriate custom and predefined exceptions to alert the user about the errors occurred or likely to occur.

Validations at the level of individual controls are also applied whenever necessary. System pops up appropriate and sensuous dialogs whenever necessary.

5.6 SOURCE CODE:

```
#define BLYNK_TEMPLATE_ID "TMPL3cUsEpof5"
#define BLYNK_TEMPLATE_NAME "SANDEEP"
#define BLYNK_AUTH_TOKEN "Qrkf9sE1KHke8HMsdQo0wivLPfnaGpdd"

#include <BlynkSimpleEsp32.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <DHT.h>

// COREXY
#define STEP_PIN_1 12
#define DIR_PIN_1 13
#define STEP_PIN_2 14
#define DIR_PIN_2 27
#define FORWARD HIGH
#define BACKWARD LOW

// peripherals
#define FERTILIZER_PUMP_PIN 25
#define WATER_PUMP_PIN 26
#define RED_LED_PIN 22
#define BLUE_LED_PIN 21
#define BUZZER_PIN 23
#define LIMIT_RIGHT 32
#define LIMIT_LEFT 33
#define LIMIT_TOP 34
#define LIMIT_BOTTOM 35
#define SOIL_MOISTURE_PIN 39
#define DHTPIN 0
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
// Blynk
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "High";
char pass[] = "12345678";

bool autoModeActive = false;
```

```
bool manualModeActive = false;
bool sirenActive = false;
int currentX = 0, currentY = 0;

const char* cam_url_detect = "http://192.168.32.154/detect";
const char* cam_url_stream = "http://192.168.32.154/stream";

void verifyHomePosition() {
  Serial.println("Verifying Home Position...");

  digitalWrite(DIR_PIN_1, HIGH); digitalWrite(DIR_PIN_2, LOW);
  while (digitalRead(LIMIT_BOTTOM) != LOW) {
    digitalWrite(STEP_PIN_1, HIGH); digitalWrite(STEP_PIN_2, HIGH);
    delayMicroseconds(800);
    digitalWrite(STEP_PIN_1, LOW); digitalWrite(STEP_PIN_2, LOW);
    delayMicroseconds(800);
  }
  delay(300);

  digitalWrite(DIR_PIN_1, LOW); digitalWrite(DIR_PIN_2, LOW);
  while (digitalRead(LIMIT_LEFT) != LOW) {
    digitalWrite(STEP_PIN_1, HIGH); digitalWrite(STEP_PIN_2, HIGH);
    delayMicroseconds(800);
    digitalWrite(STEP_PIN_1, LOW); digitalWrite(STEP_PIN_2, LOW);
    delayMicroseconds(800);
  }
  currentX = 0; currentY = 0;
  Serial.println("✅ Home verified");
}

void moveTo(int x, int y) {
  int dx = x - currentX;
  int dy = y - currentY;

  if (dx != 0) {
    digitalWrite(DIR_PIN_1, dx > 0 ? BACKWARD : FORWARD);
    digitalWrite(DIR_PIN_2, dx > 0 ? FORWARD : BACKWARD);
    for (int i = 0; i < abs(dx); i++) {
      digitalWrite(STEP_PIN_1, HIGH); digitalWrite(STEP_PIN_2, HIGH);
```

```
    delayMicroseconds(800);
    digitalWrite(STEP_PIN_1, LOW); digitalWrite(STEP_PIN_2, LOW);
    delayMicroseconds(800);
   }
  }

  if (dy != 0) {
   digitalWrite(DIR_PIN_1, dy > 0 ? FORWARD : BACKWARD);
   digitalWrite(DIR_PIN_2, dy > 0 ? FORWARD : BACKWARD);
   for (int i = 0; i < abs(dy); i++) {
    digitalWrite(STEP_PIN_1, HIGH); digitalWrite(STEP_PIN_2, HIGH);
    delayMicroseconds(800);
    digitalWrite(STEP_PIN_1, LOW); digitalWrite(STEP_PIN_2, LOW);
    delayMicroseconds(800);
   }
  }

  currentX = x; currentY = y;
  delay(500);
}

bool isDiseasedLeaf() {
  HTTPClient http;
  http.begin(cam_url_detect);
  int httpCode = http.GET();
  if (httpCode == 200) {
   String payload = http.getString();
   return payload.indexOf("whiteDot's") >= 0;
  }
  return false;
}

void checkPlant(int x, int y, String name) {
  if (!autoModeActive) return;

  moveTo(x, y);
  delay(2000);

  if (isDiseasedLeaf()) {
```

```
    Serial.println(name + " diseased, fertilizing");
    Blynk.virtualWrite(V4, name + " diseased (getting fertilizer)");

    // Fertilizer pump ON
    digitalWrite(FERTILIZER_PUMP_PIN, LOW);

    // Start siren pulse for 4 seconds
    unsigned long startTime = millis();
    bool ledState = false;

    while (millis() - startTime < 4000) {
      ledState = !ledState;
      digitalWrite(RED_LED_PIN, ledState ? HIGH : LOW);
      digitalWrite(BLUE_LED_PIN, ledState ? LOW : HIGH);
      digitalWrite(BUZZER_PIN, ledState ? HIGH : LOW);
      Blynk.virtualWrite(V11, ledState);  // Blynk siren indicator
      delay(300);
    }

    // Stop fertilizer, siren, buzzer
    digitalWrite(FERTILIZER_PUMP_PIN, HIGH);
    digitalWrite(RED_LED_PIN, LOW);
    digitalWrite(BLUE_LED_PIN, LOW);
    digitalWrite(BUZZER_PIN, LOW);
    Blynk.virtualWrite(V11, 0);

  } else {
    Serial.println(name + " healthy");
    Blynk.virtualWrite(V4, name + " healthy");
  }
}

// BLYNK callbacks
BLYNK_WRITE(V20) {  // auto mode
  autoModeActive = param.asInt();
  if (autoModeActive) {
    manualModeActive = false;
  }
}
```

```
BLYNK_WRITE(V21) {  // manual mode
  manualModeActive = param.asInt();
  if (manualModeActive) {
    autoModeActive = false;
  }
}
BLYNK_WRITE(V10) {  // image capture
  if (manualModeActive && param.asInt()) {
    Blynk.virtualWrite(V30, cam_url_stream);
  }
}
BLYNK_WRITE(V12) { if(manualModeActive && param.asInt())
moveTo(currentX,currentY+200); }
BLYNK_WRITE(V13) { if(manualModeActive && param.asInt())
moveTo(currentX,currentY-200); }
BLYNK_WRITE(V14) { if(manualModeActive && param.asInt()) moveTo(currentX-
200,currentY); }
BLYNK_WRITE(V15) { if(manualModeActive && param.asInt())
moveTo(currentX+200,currentY); }
BLYNK_WRITE(V16) { if(manualModeActive) digitalWrite(FERTILIZER_PUMP_PIN,
!param.asInt()); }
BLYNK_WRITE(V17) { if(manualModeActive) digitalWrite(WATER_PUMP_PIN,
!param.asInt()); }
BLYNK_WRITE(V18) { // emergency stop
  digitalWrite(STEP_PIN_1, LOW); digitalWrite(STEP_PIN_2, LOW);
  digitalWrite(FERTILIZER_PUMP_PIN, HIGH);
  digitalWrite(WATER_PUMP_PIN, HIGH);
}
void soilMoistureTask(void *pv) {
  for (;;) {
    int val = analogRead(SOIL_MOISTURE_PIN);
    Blynk.virtualWrite(V0, val);
    Blynk.virtualWrite(V1, val > 2000 ? "Soil dry" : "Soil wet");
    digitalWrite(WATER_PUMP_PIN, val > 2000 ? LOW : HIGH);
    vTaskDelay(5000/portTICK_PERIOD_MS);
  }
}

void dhtTask(void *pv) {
  for(;;) {
```

```
    float t = dht.readTemperature();
    float h = dht.readHumidity();
    Blynk.virtualWrite(V2, t);
    Blynk.virtualWrite(V3, h);
    vTaskDelay(5000/portTICK_PERIOD_MS);
  }
}
void setup() {
  Serial.begin(115200);
  pinMode(STEP_PIN_1, OUTPUT); pinMode(DIR_PIN_1, OUTPUT);
  pinMode(STEP_PIN_2, OUTPUT); pinMode(DIR_PIN_2, OUTPUT);
  pinMode(FERTILIZER_PUMP_PIN, OUTPUT); digitalWrite(FERTILIZER_PUMP_PIN,
HIGH);
  pinMode(WATER_PUMP_PIN, OUTPUT); digitalWrite(WATER_PUMP_PIN, HIGH);
  pinMode(RED_LED_PIN, OUTPUT); pinMode(BLUE_LED_PIN, OUTPUT);
pinMode(BUZZER_PIN, OUTPUT);
  pinMode(LIMIT_LEFT, INPUT_PULLUP); pinMode(LIMIT_BOTTOM,
INPUT_PULLUP);
  dht.begin();
  WiFi.begin(ssid, pass);
  Blynk.begin(auth, ssid, pass);
  verifyHomePosition();
  xTaskCreatePinnedToCore(soilMoistureTask,"soilTask",4096,NULL,1,NULL,1);
  xTaskCreatePinnedToCore(dhtTask,"dhtTask",4096,NULL,1,NULL,1);
}
void loop() {
  Blynk.run();
  if (autoModeActive) {
    checkPlant(0, 200, "P1");
    checkPlant(900, 200, "P2");
    checkPlant(1900, 200, "P3");
    checkPlant(0, 1500, "P4");
    checkPlant(900, 1500, "P5");
    checkPlant(1900, 1500, "P6");
    moveTo(0,0);
    Blynk.virtualWrite(V4,"Auto cycle complete");
    delay(10000);
  }
}
```

ESP32-CAM Module Code:

```
#include <WiFi.h>
#include <WebServer.h>
#include <esp_camera.h>
#include <leafDiseaseDetection1_inferencing.h>
#include "edge-impulse-sdk/dsp/image/image.hpp"

// ==== CAMERA CONFIGURATION: AI Thinker ====
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

// ==== WIFI CREDENTIALS ====
const char* ssid = "High";
const char* password = "12345678";
const char* deviceName = "ESP32-CAM-LEAF";

// ==== CAMERA SETTINGS ====
static camera_config_t camera_config = {
  .pin_pwdn       = PWDN_GPIO_NUM,
  .pin_reset      = RESET_GPIO_NUM,
  .pin_xclk       = XCLK_GPIO_NUM,
  .pin_sscb_sda   = SIOD_GPIO_NUM,
  .pin_sscb_scl   = SIOC_GPIO_NUM,
  .pin_d7         = Y9_GPIO_NUM,
```

```
  .pin_d6       = Y8_GPIO_NUM,
  .pin_d5       = Y7_GPIO_NUM,
  .pin_d4       = Y6_GPIO_NUM,
  .pin_d3       = Y5_GPIO_NUM,
  .pin_d2       = Y4_GPIO_NUM,
  .pin_d1       = Y3_GPIO_NUM,
  .pin_d0       = Y2_GPIO_NUM,
  .pin_vsync    = VSYNC_GPIO_NUM,
  .pin_href     = HREF_GPIO_NUM,
  .pin_pclk     = PCLK_GPIO_NUM,
  .xclk_freq_hz = 20000000,
  .ledc_timer   = LEDC_TIMER_0,
  .ledc_channel = LEDC_CHANNEL_0,
  .pixel_format = PIXFORMAT_JPEG,
  .frame_size   = FRAMESIZE_QVGA,
  .jpeg_quality = 12,
  .fb_count     = 1,
  .fb_location  = CAMERA_FB_IN_PSRAM,
  .grab_mode    = CAMERA_GRAB_WHEN_EMPTY
};

// ==== VARIABLES ====
WebServer server(80);
uint8_t* snapshot_buf;

#define EI_CAMERA_RAW_FRAME_BUFFER_COLS 320
#define EI_CAMERA_RAW_FRAME_BUFFER_ROWS 240
#define EI_CAMERA_FRAME_BYTE_SIZE 3

// ==== AI Inference Image Capture ====
bool ei_camera_capture(uint32_t img_width, uint32_t img_height, uint8_t* out_buf) {
  camera_fb_t* fb = esp_camera_fb_get();
  if (!fb) return false;

  bool converted = fmt2rgb888(fb->buf, fb->len, PIXFORMAT_JPEG, snapshot_buf);
  esp_camera_fb_return(fb);

  if (!converted) return false;
```

```
  if ((img_width != EI_CAMERA_RAW_FRAME_BUFFER_COLS) || (img_height != EI_CAMERA_RAW_FRAME_BUFFER_ROWS)) {
    ei::image::processing::crop_and_interpolate_rgb888(
      out_buf,
      EI_CAMERA_RAW_FRAME_BUFFER_COLS,
      EI_CAMERA_RAW_FRAME_BUFFER_ROWS,
      out_buf,
      img_width,
      img_height
    );
  }
  return true;
}

// ==== Helper for Edge Impulse Classifier ====
static int ei_camera_get_data(size_t offset, size_t length, float* out_ptr) {
  size_t pixel_ix = offset * 3;
  for (size_t i = 0; i < length; i++) {
    out_ptr[i] = (snapshot_buf[pixel_ix + 2] << 16) + (snapshot_buf[pixel_ix + 1] << 8) + snapshot_buf[pixel_ix];
    pixel_ix += 3;
  }
  return 0;
}

// ==== /DETECT ====
void handleDetect() {
  snapshot_buf = (uint8_t*)malloc(EI_CAMERA_RAW_FRAME_BUFFER_COLS * EI_CAMERA_RAW_FRAME_BUFFER_ROWS * EI_CAMERA_FRAME_BYTE_SIZE);
  if (!snapshot_buf) {
    server.send(500, "text/plain", "Failed to allocate buffer");
    return;
  }

  if (!ei_camera_capture(EI_CLASSIFIER_INPUT_WIDTH, EI_CLASSIFIER_INPUT_HEIGHT, snapshot_buf)) {
    server.send(500, "text/plain", "Image capture failed");
    free(snapshot_buf);
    return;
  }
```

```
  ei::signal_t signal;
  signal.total_length = EI_CLASSIFIER_INPUT_WIDTH *
EI_CLASSIFIER_INPUT_HEIGHT;
  signal.get_data = &ei_camera_get_data;

  ei_impulse_result_t result = { 0 };
  EI_IMPULSE_ERROR res = run_classifier(&signal, &result, false);

  if (res != EI_IMPULSE_OK) {
    server.send(500, "text/plain", "Classifier failed");
    free(snapshot_buf);
    return;
  }

  String status = "Healthy";
  for (uint16_t i = 0; i < EI_CLASSIFIER_LABEL_COUNT; i++) {
    if (String(ei_classifier_inferencing_categories[i]) == "whiteDot's" &&
result.classification[i].value > 0.6) {
      status = "whiteDot's";
      break;
    }
  }

  free(snapshot_buf);
  server.send(200, "text/plain", status);
}

// ==== /STREAM ====
void handleStream() {
  WiFiClient client = server.client();

  String response = "HTTP/1.1 200 OK\r\n";
  response += "Content-Type: multipart/x-mixed-replace; boundary=frame\r\n\r\n";
  client.print(response);

  while (client.connected()) {
    camera_fb_t* fb = esp_camera_fb_get();
    if (!fb) continue;
```

```
    client.printf("--frame\r\nContent-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n", fb->len);
    client.write(fb->buf, fb->len);
    client.print("\r\n");

    esp_camera_fb_return(fb);
    delay(100);
  }
}

// ==== /CAPTURE ====
void handleCapture() {
  camera_fb_t* fb = esp_camera_fb_get();
  if (!fb) {
    server.send(500, "text/plain", "Camera capture failed");
    return;
  }
  server.send_P(200, "image/jpeg", (char*)fb->buf, fb->len);
  esp_camera_fb_return(fb);
}

// ==== SETUP ====
void setup() {
  Serial.begin(115200);

  Serial.println();
  Serial.print("🔌 Device: ");
  Serial.println(deviceName);
  Serial.print("🔗 Connecting to WiFi: ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println();
  Serial.println("✅ WiFi connected");
```

```
  Serial.print("📶 IP address: ");
  Serial.println(WiFi.localIP());

  if (esp_camera_init(&camera_config) != ESP_OK) {
    Serial.println("❌ Camera init failed!");
    return;
  }

  server.on("/detect", handleDetect);
  server.on("/stream", handleStream);
  server.on("/capture", handleCapture);   // snapshot
  server.begin();
  Serial.println("🌐 Web server started");
}


// ==== LOOP ====
void loop() {
  server.handleClient();
}
```

TESTING

6 .TESTING

Testing is the process of finding differences between the expected behavior specifiedby system models and the observed behavior of the system. Testing is a critical role in quality assurance and ensuring the reliability of development and these errors will be reflected in the codes the application should be thoroughly tested and validated.

Unit testing finds the differences between the object design model and its corresponding components. Structural testing finds differences between the system design model and asubset of integrated subsystems. Functional testing finds differences between the use case model and the system.

Finally, performance testing, finds differences between non-functional requirements and actual system performances. Form modeling point of view, testing is the attempt of falsification of the system with respect to the system models. The goal of testing is to design tests that exercise defects in the system and to reveal problems.

6.1 Testing Activities

Testing a large system is a complex system is a complex activity and like any complex activity. It must be breaking into smaller activities. Thus, incremental testing was performed on the project i.e., components and subsystems of the system were tested separately before integrating them to from the subsystem for system testing.

6.2 Types of testing

Unit Testing:

Unit testing is essential to ensure reliability and accuracy. First, each parking sensor's connectivity to the IoT network must be verified to confirm proper integration. Next, the sensors should be tested for data accuracy to ensure they correctly detect and report vehicle presence or absence. It is also crucial to validate the communication protocol, ensuring that data transmission from the sensors to the central server is both reliable and consistent. Additionally, the system must be checked for database integrity to confirm that the server correctly updates and stores parking status data. Lastly, the user interface should be tested to ensure that the app accurately displays real-time parking availability to users.

Integration Testing:

Integration testing using IoT involves several critical steps to ensure seamless operation. First, the end-to-end data flow must be verified, ensuring data is transmitted smoothly from parking sensors to the central server and subsequently to the user app. Next, the interaction and synchronization between the parking sensors, server, and database need to be tested to confirm they work together correctly. Real-time updates are also crucial; therefore, it's essential to ensure that any changes in parking status are accurately reflected across all system components. Finally, the system's ability to handle errors must be tested, ensuring it can effectively manage issues like sensor disconnections or network failures, and recover gracefully from such disruptions.

Functional Testing:

Functional testing focuses on verifying that all system features work as intended. This includes ensuring that parking sensors accurately detect vehicle presence and absence, and that this data is correctly processed and displayed in the user app. The system's reservation functionality must be tested to confirm that users can book parking spaces and that these reservations are properly reflected in the system. Additionally, payment processing should be tested to ensure transactions are secure and correctly recorded. Overall, functional testing ensures that each feature of the smart parking system operates correctly and provides a seamless user experience.

Performance Testing:

Performance testing is essential to ensure the system operates efficiently under various conditions. This involves evaluating the system's response time to detect and report vehicle presence, and ensuring the central server can handle large volumes of data from numerous sensors without delays. The system's scalability is tested to verify it can support an increasing number of users and parking spots without performance degradation. Additionally, stress tests are conducted to assess the system's stability and reliability under peak load conditions, ensuring that it maintains functionality during high traffic periods.

Accuracy Testing:

Accuracy testing focuses on verifying that parking sensors correctly detect vehicle presence and absence. This involves comparing sensor data against actual parking occupancy to ensure high detection precision. The test also checks for false positives and negatives to minimize errors. Ensuring accurate data transmission to the central server and user app is crucial for reliable real-time parking availability information.

Usability Testing:

Usability testing ensures the system is user-friendly and intuitive. This involves evaluating the user interface of the app for ease of navigation, clarity of information, and efficiency in booking and paying for parking spots. Feedback from real users is collected to identify any issues and improve overall user experience, ensuring the system meets users' needs effectively.

Security Testing:

Security testing involves ensuring the protection of data and system integrity. This includes verifying secure communication between sensors, servers, and user applications to prevent data breaches. Tests are conducted to identify vulnerabilities in the system, such as unauthorized access, data tampering, and potential cyber-attacks. Ensuring robust authentication and encryption mechanisms is critical to safeguarding user data and maintaining the overall security of the smart parking system.

Black box Testing:

Black box testing involves evaluating the system's functionality without examining its internal code or structure. Testers focus on verifying that the system correctly handles input data, such as vehicle detection and parking reservations, and produces the expected output, like accurate parking availability and booking confirmations. The goal is to ensure that the system meets functional requirements and performs as intended from a user's perspective.

Whitebox Testing:

White box testing involves examining the internal logic and code of the system. This includes testing algorithms for vehicle detection, data processing, and communication protocols to ensure they function correctly and efficiently. The aim is to identify and fix potential issues within the code, such as logical errors or security vulnerabilities, ensuring the system operates as intended at a technical level.

6.3 Testing Plan:

A robust testing plan for the AI-Powered IoT System for Plant Disease Detection and Fertilizer Precision Spraying is crucial to ensure reliability and accuracy in real-world agricultural settings. Unit testing will be conducted on each individual hardware component including the ESP32-CAM, ESP32 controller, relay modules, CoreXY motors, soil moisture sensor, and water pumps to verify their basic functionality. Integration testing will validate communication between the ESP32-CAM and the main ESP32 module over Wi-Fi, coordination between disease detection results and movement of the CoreXY, as well as the relay activation logic. Functional testing will verify that the camera captures plant images correctly, the AI model processes them accurately, and the system performs fertilizer spraying and watering based on plant health and soil moisture. Performance testing will assess how quickly images are analyzed and actions executed under continuous operation. Accuracy testing will verify correct disease detection by the ML model and correct location mapping. Usability testing will ensure that farmers can monitor plant status and issue commands via the Blynk dashboard intuitively. Security testing will be done to protect communication over Wi-Fi and secure access to cloud-based dashboards. Regression testing will ensure firmware updates do not affect previously functional modules. Acceptance testing will confirm that the system performs plant monitoring, diagnosis, spraying, and watering as intended across all 6 plant stations.

6.4 Test Cases

Test Case for Disease Detection and Reporting:

Test Case ID	TC001
Description	Validate that the ESP32-CAM captures an image and sends it to the ML model for disease detection.
Steps:	• Power on the system. • Place a diseased plant in front of the ESP32-CAM. • Check if the ESP32-CAM captures the image. • Confirm the image is processed by the Edge Impulse ML model. • Check whether disease status and coordinates are sent to the main ESP32.
ExpectedResult	The system detects disease, classifies it, and sends accurate location and type to the ESP32.

Table 6.1: Disease Detection and Reporting

Test Case for Movement to Target Location:

Test Case ID	TC002
Description	Validate that the CoreXY system moves to the specified coordinates.
Steps	• Receive target coordinates from the ESP32. • Initiate CoreXY movement to location. • Check if CoreXY reaches correct position without collision.
Expected Results	CoreXY moves precisely to the diseased plant location for treatment.

Table 6.2: Movement to Target Location

Test case for Fertilizer Spraying::

Test Case ID	TC003
Description	Validate that the fertilizer is sprayed correctly at the diseased plant.
Steps:	• Simulate disease detection. • Allow system to move to target. • Confirm relay activates the correct pump.
Expected Results	The correct pump is activated and fertilizer is sprayed over diseased plant.

Table 6.3: Fertilizer Spraying

Test case for Soil Moisture Monitoring and Watering:

Test Case ID	TC004
Description	Validate that watering occurs only when soil moisture is below the threshold.
Steps	• Insert the soil moisture sensor into dry soil. • Wait for the ESP32 to read sensor values. • Observe if the water pump is activated.
Expected Result	If soil is dry, water pump activates; otherwise, no watering occurs.

Table 6.4: Soil Moisture Monitoring and Watering

Test case for Cloud Data Update:

Test Case ID	TC005
Description	Validate that system updates disease, treatment, and moisture status to IoT cloud.
Steps	• Run disease detection and watering cycle. • Open Blynk dashboard. • Check for updated disease, moisture, and treatment data.
Expected Result	Real-time status is reflected accurately in the cloud dashboard.

Table 6.5: Cloud Data Update

Test case for Manual Override via Blynk:

Test Case ID	TC006
Description	Validate that the user can manually start pump or move system via Blynk.
Steps	• Open Blynk dashboard. • Press manual pump ON button. • Observe response on ESP32 board.
Expected Result	Manual commands are executed by hardware without error.

Table 6.6: Manual Override via Blynk

SCREENS

7. SCREENS

The below screens shows initial position of ESP32-CAM.

Figure 7.1: Home position

Figure 7.2: moved to second position

Description:

The ESP32-CAM module is mounted on a CoreXY mechanism that allows precise movement between multiple plant positions. It can seamlessly travel from its home location to any designated plant, such as Plant 2, for scanning or monitoring purposes. This movement is controlled by the main ESP32 board based on pre-defined coordinates. The flexibility of motion enables the system to cover all plant positions efficiently.

The below screens shows Healthy plants:

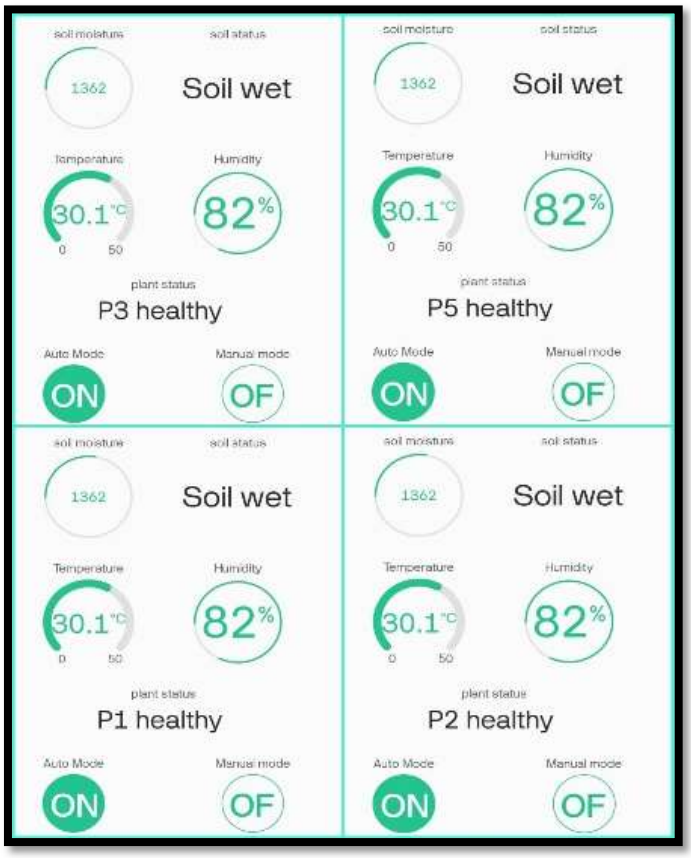

Figure 7.3 Healthy plants detected

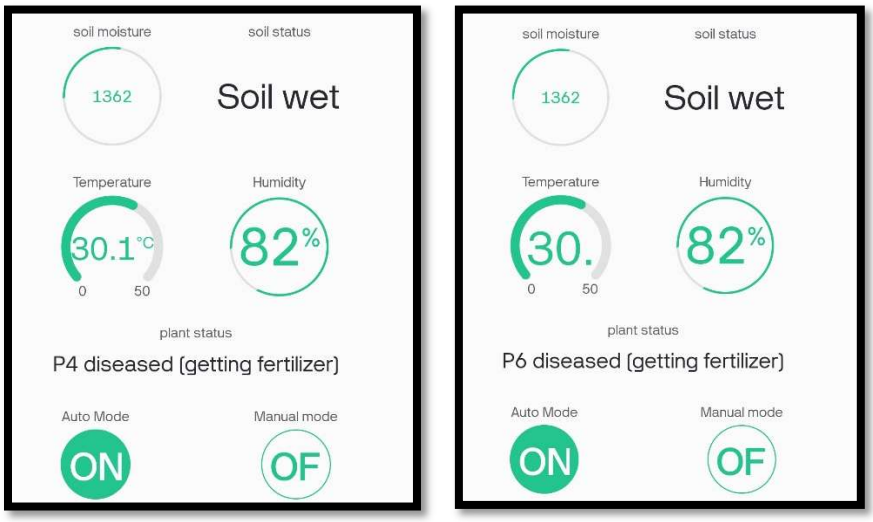

Figure 7.4 Disease plants detected

Description:

The ESP32-CAM captures images of each plant and sends them for disease analysis using a TinyML model. Based on the AI inference, the plant is classified as either healthy or diseased. The detection result is then communicated to the main ESP32 controller. The status of each plant—Healthy or Diseased—is updated in real time on the Blynk dashboard. This allows remote monitoring of plant health through a mobile application..

The below screen shows fertilizer spraying

Figure 7.5 fertilizer spraying

Description:

The ESP32-CAM module successfully captured an image and identified a diseased leaf at Plant 4 and Plant 6. Based on the coordinates received from the AI model, the CoreXY mechanism moved precisely from the home position to the Plants location. The system then activated the relay to spray fertilizer at the target plant. After spraying, updated treatment status was logged to the IoT cloud.

The below screen shows Blynk Dashboards

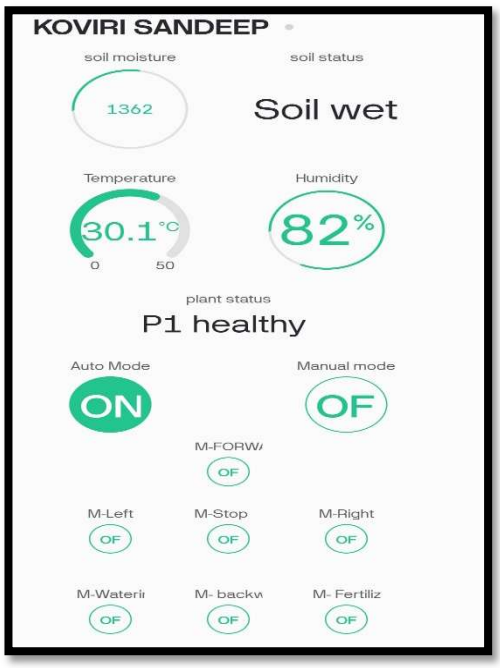

Figure 7.6 Blynk Dashboard in mobile

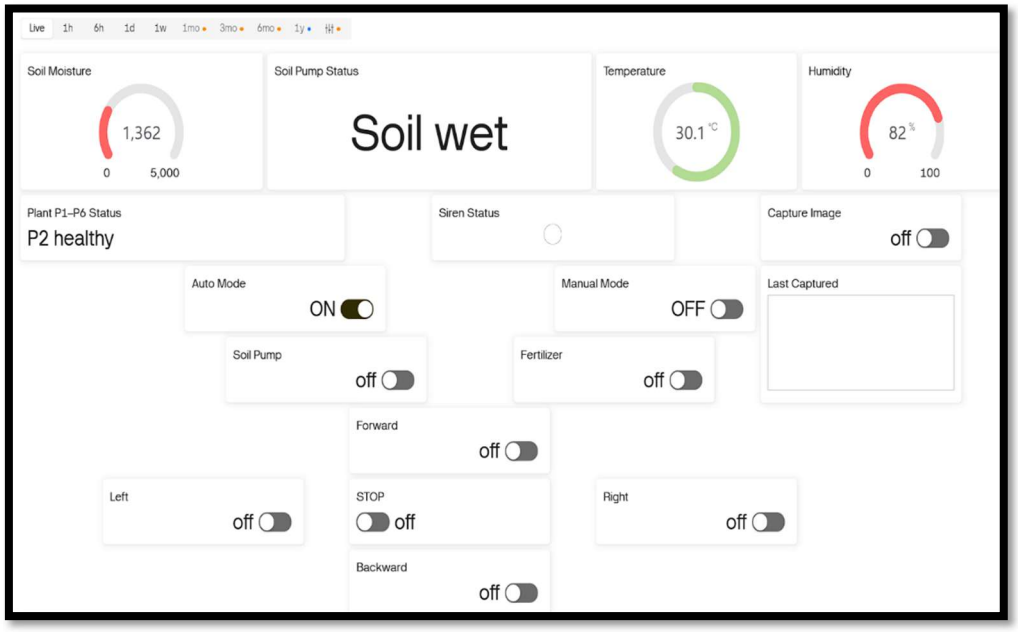

Figure 7.7 Blynk Dashboard in Web

Description:

The Blynk dashboard, available on both mobile and web platforms, displays real-time environmental parameters such as temperature, humidity, and soil moisture. It shows the current soil condition (e.g., "Soil Wet") and plant health status (e.g., "P1 Healthy", "P2 Healthy"). The dashboard allows users to switch between Auto and Manual operation modes for system control. In manual mode, users can command directional movement and activate fertilizer or water pumps. This interface enables convenient and centralized monitoring and control of the plant health system.

The below screen shows Soil moisture Status

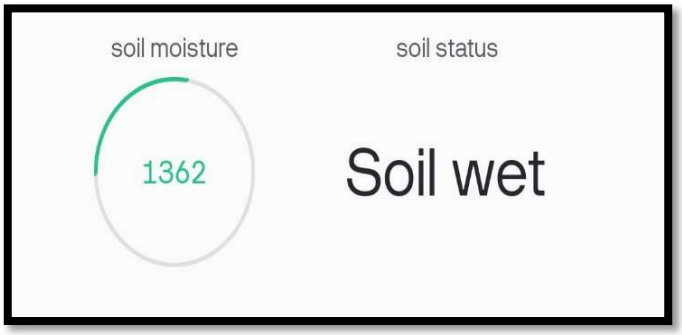

Figure 7.8 Soil Wet

Figure 7.9 Soil Dry

Description:

The soil moisture levels are effectively monitored and displayed on both the Blynk mobile and web dashboards. When the moisture value is low (e.g., 1362), the system labels the status as "Soil Wet," indicating sufficient moisture for plant health. Conversely, when the moisture value rises above the set threshold (e.g., 4050), the status updates to "Soil Dry," signaling the need for irrigation. These real-time updates enable precise decision-making for watering schedules. The system ensures plants receive optimal moisture, thereby supporting healthy growth and efficient resource usage.

The below screen shows DHT11

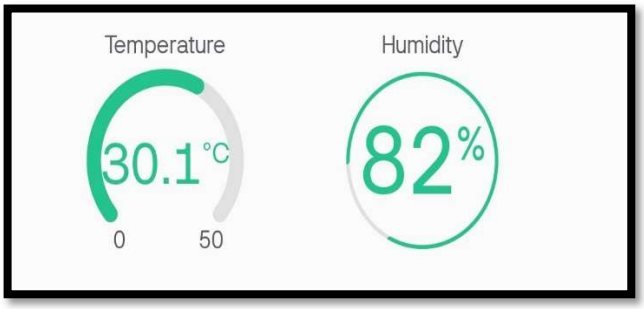

Figure 7.10 Temperature and Humidity

Description:

The DHT sensor in the system accurately measures and displays temperature and humidity data in real time on the Blynk dashboard. In this case, the temperature is shown as 30.1°C, indicating warm ambient conditions. The humidity level is recorded at 82%, reflecting a high-moisture environment suitable for plant growth. These readings help monitor the climate around the plants to ensure optimal growing conditions. The live display allows users to make timely decisions on ventilation or irrigation based on environmental changes.

CONCLUSION

8. CONCLUSION

The AI-Powered IoT System for Plant Disease Detection and Fertilizer Precision Spraying enhances agricultural productivity by combining real-time monitoring, machine learning, and precision spraying. This system minimizes pesticide wastage, improves disease detection accuracy, and reduces the need for manual intervention.

Future improvements could include integrating AI-based predictive analytics for early disease forecasting and expanding the system's functionality to cover large-scale farms with multiple crop varieties. By leveraging IoT and automation, this project paves the way for sustainable and efficient smart farming solutions.

FUTURE SCOPE

9. FUTURE SCOPE

- **Integration with Drone Systems:** Future versions of the system can be extended to work with autonomous drones for aerial surveillance and disease detection across larger farmlands, enhancing coverage and reach.
- **Advanced Disease Classification:** Incorporate more sophisticated AI/ML models trained on broader datasets to detect a wider range of plant diseases and predict their stages, severity, and spread.
- **Dynamic Precision Spraying:** Implement real-time dosage calibration based on disease severity, leaf size, and plant type, reducing chemical usage and increasing treatment accuracy.
- **Predictive Analytics for Crop Health:** Leverage cloud-stored historical data to predict disease trends and suggest preventive actions, offering farmers guidance before diseases even appear.
- **Real-time Voice Alerts and Mobile Notifications:** Add functionality to send real-time SMS, app, or voice alerts to farmers with diagnosis results, treatment actions, and irrigation status.
- **Expansion to Multi-Plant and Multi-Row Systems:** Scale the CoreXY and image processing system to monitor and treat larger grids of plants or multi-crop rows using advanced path planning.
- **Autonomous Navigation System:** Replace or enhance the CoreXY mechanism with AGVs (Automated Guided Vehicles) or robotic arms for autonomous navigation and dynamic route optimization.
- **Integration with Weather Forecasting APIs:** Combine weather data with disease detection and watering decisions to avoid spraying during windy or rainy periods and optimize irrigation.
- **Solar-Powered Deployment**: Incorporate solar panels for energy autonomy, making the system sustainable and deployable in remote agricultural fields without power supply.
- **Open Source API and Mobile Dashboard:** Provide an open API for third-party developers to build mobile apps or integrate with larger farm management platforms to further empower farmers.

REFERENCE

10. REFERENCES

10.1 Academic Papers and Journals

- Kamilaris, A. Kartakoullis, and F. X. Prenafeta-Boldú, “A Review on the Practice of Big Data Analysis in Agriculture,” Computers and Electronics in Agriculture, vol. 143, pp. 23–37, 2017.

- H. Jadhav and A. H. Bhagat, “Plant Disease Detection Using Image Processing and Machine Learning,” International Journal of Innovative Science and Research Technology, Vol. 4, Issue 4, pp. 645–649, April 2019.

- S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, “Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification,” Computational Intelligence and Neuroscience, Hindawi, 2016.

- M. K. Wankhade, G. N. Shinde, “Detection and Classification of Plant Leaf Diseases Using Machine Learning,” International Journal of Computer Applications, Vol. 180, No. 33, pp. 7–11, 2018.

- N. Patel, D. Shah, H. Bheda, “IoT-Based Smart Agriculture Monitoring and Disease Detection System,” International Journal of Computer Sciences and Engineering, Vol. 6, No. 10, pp. 513–517, October 2018.

- Thakur, B. Jaiswal, “Precision Agriculture Using IoT and Image Processing Techniques,” International Journal of Advanced Research in Computer and Communication Engineering, Vol. 8, Issue 3, March 2019.

- G. Singh, D. Kumar, R. Pandey, “Smart Farming Using IoT, a Solution for Optimizing Water Use in Agriculture,” 2018 4th International Conference on Computing Communication and Automation (ICCCA), pp. 1–4.

10.2 Web References

- https://youtu.be/h4HQlSZROQ8
- https://youtu.be/RCtVxZnjPmY
- https://youtu.be/bZIKVaD3dRk
- https://youtu.be/0oG6OF1SXto
- https://youtu.be/lqPyH3eR45Y?si=fZZ2uyfj2mYv4DFB
- https://youtu.be/_y9vX5iPckQ?si=LE9Zcf58d9k0eQ7A

APPENDIX

11. APPENDIX

List of Figures

List of Tables