Web Based System For Precision Agriculture Applications In 5G-Based Internet Of Things

Graduation Project Thesis
Class of 2023

Aswan University
Department of Computer Engineering in Electrical Engineering (CE)

Principal Supervisors:	Dr. Montasser M. Ramadan
	Dr. Hamada Ahmed Hamada
	Dr. Adel Fathy Khalifa
	Dr. Mostafa Mohamed Abbas
Associate Supervisor:	Eng. Hossam Badry
Students:	Mohamed Nabil
	Yusuf Khaled
	Ziad Hussien
Submission:	11th July 2023

STUDENTS CONTRIBUTION

Implementation and Performance Evaluation

By:

(1) Mohamed Nabil

(2) Yousef Khaled

(3) Ziad Hussien

Chapter	Title	Contributors
1	Introduction	Mohamed Nabil Yousef Khaled Ziad Hussien
2	Web Development	Yousef Khaled
3	Embedded System	Mohamed Nabil
4	Mobile Development	Ziad Hussien
5	Deep Learning	Ziad Hussien
6	Results	Mohamed Nabil Yousef Khaled Ziad Hussien
7	Conclusion and Future work	Mohamed Nabil Yousef Khaled Ziad Hussien

Abstract

Before the advent of precision agriculture, farmers faced numerous challenges that hindered their ability to maximize crop yields and minimize losses. These problems included inefficient use of resources such as water, fertilizers, and pesticides, resulting in increased costs and environmental pollution. Additionally, traditional farming practices often relied on guesswork or subjective assessments, leading to inaccurate predictions of crop growth and yield. This made it difficult for farmers to make informed decisions about their farming practices, resulting in suboptimal crop yields and reduced profitability. Moreover, traditional farming practices were often labor-intensive, requiring significant manual labor and time. This led to low productivity and increased costs. Overall, the challenges associated with traditional farming practices highlighted the need for more efficient and precise agricultural techniques, which led to the development of precision agriculture. When trying to find a solution to this issue. The main goal of the proposed system project is to address this issue, and it does so by combining Internet of Things (IoT), Machine Learning (ML) technologies and Web Services. The implementation strategy is based on the use of a low-cost camera and sensor system to identify the type of plant, determine whether it is sick, as well as its solid status and make decisions in accordance with that status. Displaying this information on a website with some features to help the user in managing his fields remotely and easily.

Acknowledgments

We sincerely thank Allah for his mercies and for guiding us to the successful completion of this endeavor. We are extremely grateful to our project supervisors Dr. Montasser, Dr. Hamada, Dr. Adel Fathy and Dr. Mostafa Abbas for all their helpful suggestions, words of support, conversations, and never-ending ideas. A special thank you to Eng. Hossam Badry and Eng. Abdelrahman Yehia for their outstanding efforts in providing the team with guidance and advice throughout the project.
We also want to thank all of our teachers from the computer and communication departments who did their best to share their knowledge of two of the most popular engineering fields with us. Finally, we would want to express our gratitude to our families for their love and support, without which we would not have been able to achieve academic success.

Contents

Figures

Tables

1 Introduction

Precision agriculture is a farming management concept that involves observing, measuring, and responding to inter- and intra-field variability in crops. In other words, precision agriculture seeks to optimize crop production by utilizing modern information technologies to provide, process, and analyze multisource data of high spatial resolution. It is a management strategy that takes into account the temporal and spatial variability of crops to improve the sustainability of agricultural production. This approach to farming has gained popularity in recent years, as it offers a range of benefits to farmers, including increased crop yields, improved efficiency, and reduced costs. One of the most significant advantages of precision agriculture is the ability to increase crop yields. By utilizing precision agriculture technologies such as tractor guidance, farmers can improve on-farm efficiencies and optimize the use of land, water, fuel, and fertilizer. The use of precision instruments also enables farmers to process data faster and more accurately, reducing waste and improving overall crop health. Additionally, precision agriculture provides farmers with current, historical, and future weather analytics to make informed decisions and increase crop yield. Another advantage of precision agriculture is the improved efficiency and reduced costs associated with farming. By using precision agriculture technologies, farmers can better understand the nutrient levels and soil types across their fields, allowing them to adjust their inputs to match the actual crop needs. This approach to farming also reduces the need for manual labor and can produce healthier crops and higher yields. Ultimately, precision agriculture offers a sustainable approach to farming that benefits both farmers and the environment.

Artificial intelligence (AI) is one of the major research solutions in software development due to its rapid technological advancement and vast application area. The central concept of AI in agriculture is flexibility, rapid performance, accuracy, and cost viability. Artificial intelligence in agriculture not only assists farmers in utilizing their farming skills but also leads farmers to increase returns and improve quality while spending less money. AI-based wireless sensor technology improves the efficiency of all sectors and addresses issues faced by many sectors in the agriculture industry, such as crop harvesting, irrigation, and soil content sensitivity. AI technology allows for the diagnosis of plant diseases, pests, and malnutrition on farms, and AI sensors can monitor and control agricultural parameters.

Agriculture is a vital sector of the global economy, providing food and other essential resources to millions of people around the world. However, farming is also a complex and challenging endeavor that requires careful planning, management, and optimization. In recent years, advances in technology have opened up new opportunities for

farmers to improve their practices and increase efficiency. One of the most promising areas of innovation in agriculture is smart farming, which combines sensors, data analytics, and automation to optimize crop production and reduce waste. Smart farming systems allow farmers to monitor and control various environmental factors that affect plant growth and health, such as temperature, humidity, soil moisture, and light intensity. By collecting and analyzing data in real-time, smart farming systems can help farmers make informed decisions and take proactive measures to prevent crop damage and improve yields.

Our proposed system consists of 3 main components as illustrated if Fig.1. In our project, we explore the design and implementation of an embedded system for a smart farm using a Raspberry Pi. Our system integrates various sensors and actuators to monitor and control the environmental factors in a farm. We selected the Raspberry Pi because it's a versatile and affordable platform that can be easily customized and integrated with various sensors and controllers. Our embedded system includes sensors to measure temperature, humidity, soil moisture, and light intensity, as well as actuators to control irrigation and ventilation systems. The data collected from these sensors are processed and analyzed using machine learning algorithms to generate insights and recommendations for farmers. Our system provides a web-based interface for farmers to access real-time data and control the different components of the system from anywhere.

In addition to providing real-time monitoring and control, our system also includes a data logging and visualization feature that enables farmers to track and analyze historical data. This feature helps farmers identify trends and patterns in the data and make informed decisions for future crop planning and management. Overall, our project aims to demonstrate the potential of embedded systems and IoT technologies in agriculture and provide an affordable and scalable solution for small and medium-sized farms. By leveraging the power of data analytics and automation, we hope to help farmers improve their practices, reduce waste, and increase yields.

1.1 Traditional Agriculture

Two distinct methods of farming have arisen in recent years: traditional agriculture and precision agriculture. For generations, farmers have utilized traditional farming methods, which involve managing entire fields using manual labor and local conditions. Contrarily, farmers can employ information technology-based precision agriculture techniques and practices to better assess the state of their farms and make more focused decisions. The use of Variable-Rate Application (VRA), which permits inputs

to be delivered in amounts particular to each area of the field rather than being applied uniformly across the entire field, is the primary distinction between these two systems. Compared to traditional agriculture, precision agriculture has several benefits. One study claims that because precision farming has the potential to improve profitability, productivity, and sustainability, it will revolutionize farm management. Precision farming lowers agricultural expenses, boosts production, and guarantees that crops get the proper amount of water and nutrients. Additionally, precision agriculture makes it possible to utilize fertilizers in precisely the right amounts and at precisely the right locations, which boosts productivity and lowers waste. Precision agriculture does, however, deviate from traditional farming practices and may call for a substantial technological investment. Traditional agriculture also has benefits and drawbacks of its own. Traditional agriculture uses small-scale, non-mechanized farming methods because they allow for spatially varied treatments and can be more environmentally sustainable. Traditional farming methods, on the other hand, might not be as effective as precision agriculture and might demand more labor and resources. Precision agriculture can play a critical part in satisfying this demand because traditional agriculture may not be able to keep up with the rising food demand. The decision between precision agriculture and traditional agriculture ultimately comes down to the particular requirements and objectives of the farmer.

1.2 Related Work

Murugamani and Shitharth [1] developed a system to detect and control cotton leaf diseases. This paper comprises several aspects, including leaf disease detection, remote monitoring systems depending on the server, moisture and temperature sensing, and soil sensing. **CNN-based transfer learning** model was used for the preprocessed images. The study dealt with 4 diseases **Bacterial blight**, **Alternaria**, **Grey mildew**, and **Cerespora**. They used different types of artificial intelligence are used for detecting leaf diseases such as **Random Forest** which achieved accuracies $98.34\%, 92.45\%, 91.56\%, and 97.45\%$ respectively in the four diseases. Naive Bayes was another model and its results were close to RF as it got $75.99\%, 77.34\%, 72.45\%$, and 69.34% accuracies. Support Vector Machine(SVM) was the third option, with $98.34\%, 92.45\%, 91.56\%$, and 97.45% it got the best accuracies in the study.

Indian farmers continue to struggle with selecting the correct crop for the right biological and non-biological elements. This article discusses developing crop recommendation systems [2]. By examining the data, they extract the most important attributes using techniques like principal component analysis (PCA), and linear discriminant analysis (LDA). These extracted features are used to train models like Naive Bayes, Random

Forest, KNN, etc. using training data and performance is evaluated on test data using techniques such as cross-validation, accuracy, RMSE, precision, and recall. The best-performing model is selected and subsequently used to classify and recommend crops. The different models were applied to different test datasets. For Uttar Pradesh state's dataset, the accuracies were 40.59% for **Naive Bayes**, 78.481% for **Logistic Regression**, 80.739% for **Decision Table**, and 84.174% for **Random Forest**. On the other hand, for Karnataka state the values were 63.635% for **Naive Bayes**, 69.354% for **Logistic Regression**, 72.327% for **Decision Table**, and 75.598% for **Random Forest**

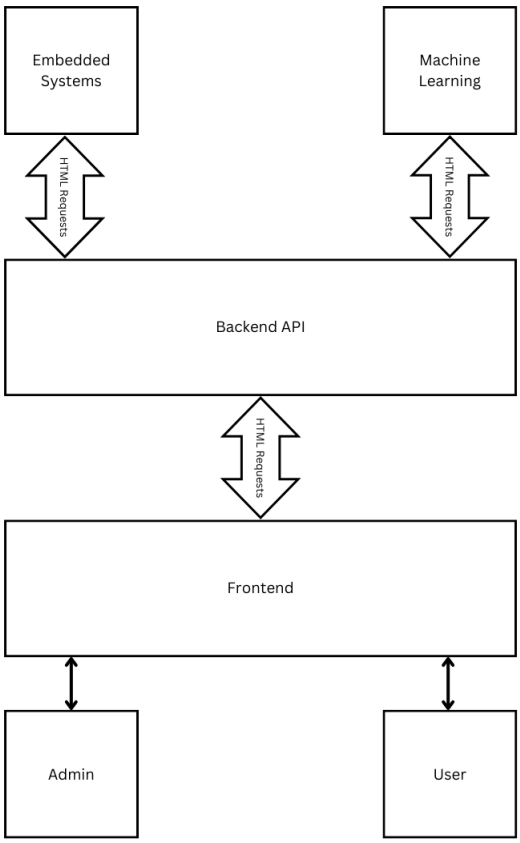


Figure 1 PA Block Diagram

2 Web Development

This web application is designed to provide users with a seamless and efficient experience while accessing the features and functionalities of the platform. The front-end of the application is built using Next.js, which allows for the creation of highly optimized and dynamic user interfaces which is shown in Fig.2. On the other hand, the back-end is built using Express.js and is designed to operate as an API that can be accessed by different clients such as web, mobile, or desktop applications. This approach ensures that the application is highly scalable and adaptable to meet the needs of its users. The use of an API-based architecture enables the application to integrate with other services, systems, and databases, thereby making it a highly flexible and versatile platform. With its responsive design, intuitive interface, and reliable performance, this web application is poised to provide users with an exceptional experience that meets their needs and exceeds their expectations.

2.1 Back End

2.1.1 Back End Development tools

Back-end applications are developed using several programming languages and development tools. Here are some of the commonly used tools:

(1) **Node.js:** This is a cross-platform, open-source back-end JavaScript runtime environment that executes JavaScript code outside a web browser. It is built on the V8 engine, which makes it efficient and fast.[3]

(2) **Express.js:** This is a fast, unopinionated, and minimalist web application framework for Node.js. It provides a robust set of features for web and mobile applications, including routing, middleware support, and template engines.[4]

(3) **MongoDB:** This is a NoSQL database program that stores data in the form of documents. It is an open-source, document-oriented database that offers high performance and scalability.[5]

(4) **MongoDB Atlas:** This is a cloud database service built and run by the team behind MongoDB. It is used to deploy, operate, and scale MongoDB in the cloud while ensuring availability, scalability, and compliance with the most demanding data security and privacy standards.[6]

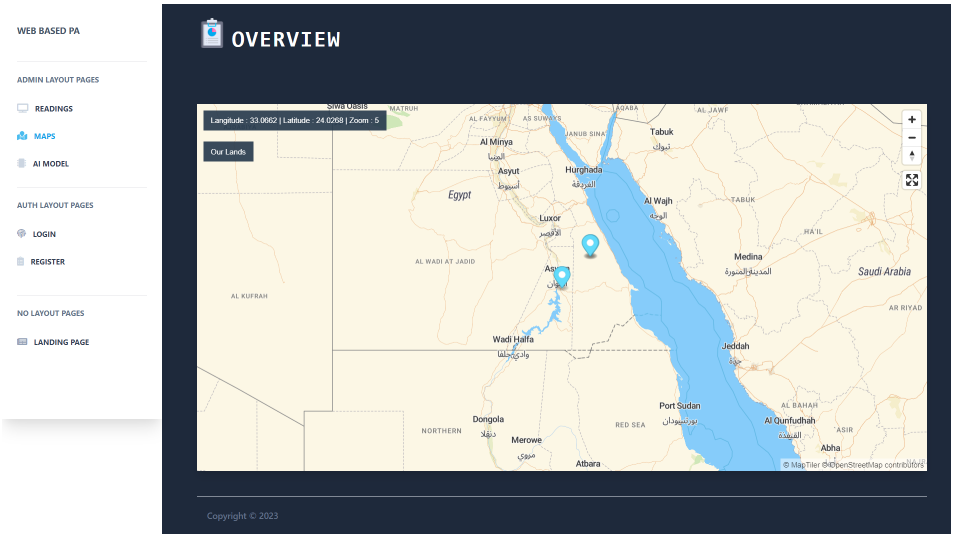
(5) **Visual Studio Code:** This is a source-code editor that is feature-rich and made by Microsoft. It offers support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.[7]

(6) **Postman:** This is an API client that is used to create, share, test, and document APIs. It enables users to run HTTP requests and read their responses, making it easier to test and debug APIs.[8]

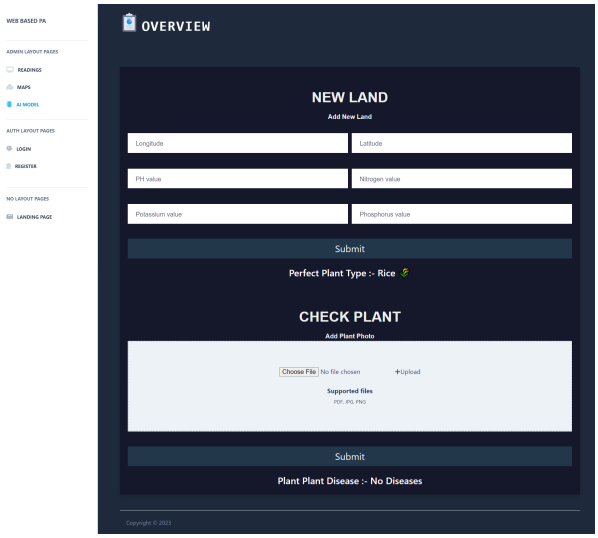
(7) **GitHub:** This is a web-based graphical interface and Git repository hosting service. It provides a platform for team members to share their work on the project and collaborate effectively.[9]

(8) **Render.com:** This is a cloud hosting platform that simplifies the deployment and scaling of web applications. It supports various programming languages and frameworks, including Node.js and Express.js, and provides a streamlined deployment process with built-in scalability and automatic SSL certificate management.[10]

(a) Reading Overview Page



(b) Maps Page



(c) AI Model Page

Figure 2 Web Page Sections

2.1.2 Back End and Core Development

In this section, we will discuss the implementation of the back-end process. The architecture utilized in this stage is MVC (Model-View-Controller). Additionally, we will delve into the details of the Express.js framework, the database, and the RESTful API.

(1) **Explaining MVC Architecture:** The Model-View-Controller (MVC) is a software application pattern that segregates the input, processing, and output components of an application into the following parts:

- **Models:** Responsible for managing data and business logic.
- **Controllers:** Handle the user interface and application logic.
- **Views:** Manage graphical user interface objects and presentation.[11][12]

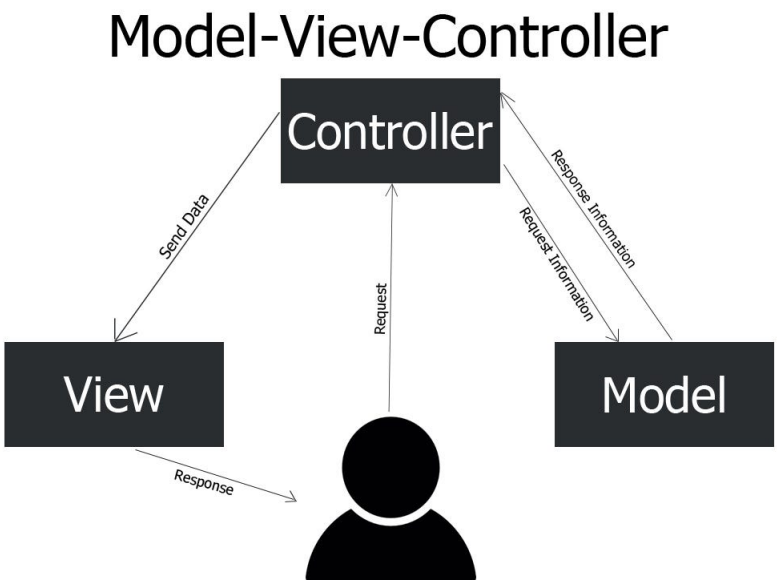


Figure 3 MVC Architectural Pattern

(2) **Express.js Framework:** Express.js is a JavaScript framework used for building web applications and APIs. In conjunction with MongoDB, a NoSQL database, and Mongoose, an Object Data Modeling (ODM) library for MongoDB, it provides a powerful and flexible platform for back-end development. The combination of Express.js and MongoDB (with Mongoose) follows the MVC pattern and offers several advantages:

(a) **Authentication:** Implementing authentication techniques in Express.js is made simple. JSON Web Tokens (JWT) can be used for authentication, allowing the server to send authentication details (tokens) to verify active users.[13][14]

(b) **Security:** Passwords can be securely hashed using bcrypt hashing algorithm.

(c) **Flexibility:** Express.js allows for the seamless integration of MongoDB as the database, offering a flexible schema and the ability to store complex data structures.

(d) **Scalability:** MongoDB's scalable architecture, combined with the lightweight and minimalist nature of Express.js, enables the development of highly scalable and efficient applications.

(e) **Object Data Modeling:** Mongoose, an ODM library for MongoDB, provides a convenient way to define schemas, models, and relationships between data entities in an application. It simplifies data validation, querying, and manipulation.

By utilizing Express.js with MongoDB, developers can leverage the benefits of a robust and efficient back-end solution. It provides a solid foundation for creating RESTful APIs and building web applications that are secure, scalable, and flexible.[15][16]

(3) **Database:** Databases form the backbone of modern applications by providing a structured and organized approach to storing and managing data. Whether it's an e-commerce platform, social media network, or even a personal blog, databases are at the core of data-driven applications, enabling seamless data storage, retrieval, and analysis. MongoDB, a leading NoSQL database, has gained widespread adoption due to its scalability, flexibility, and ability to handle large volumes of unstructured and semi-structured data. By utilizing a document-oriented data model, MongoDB eliminates the need for predefined schemas, allowing developers to easily adapt to evolving application requirements.

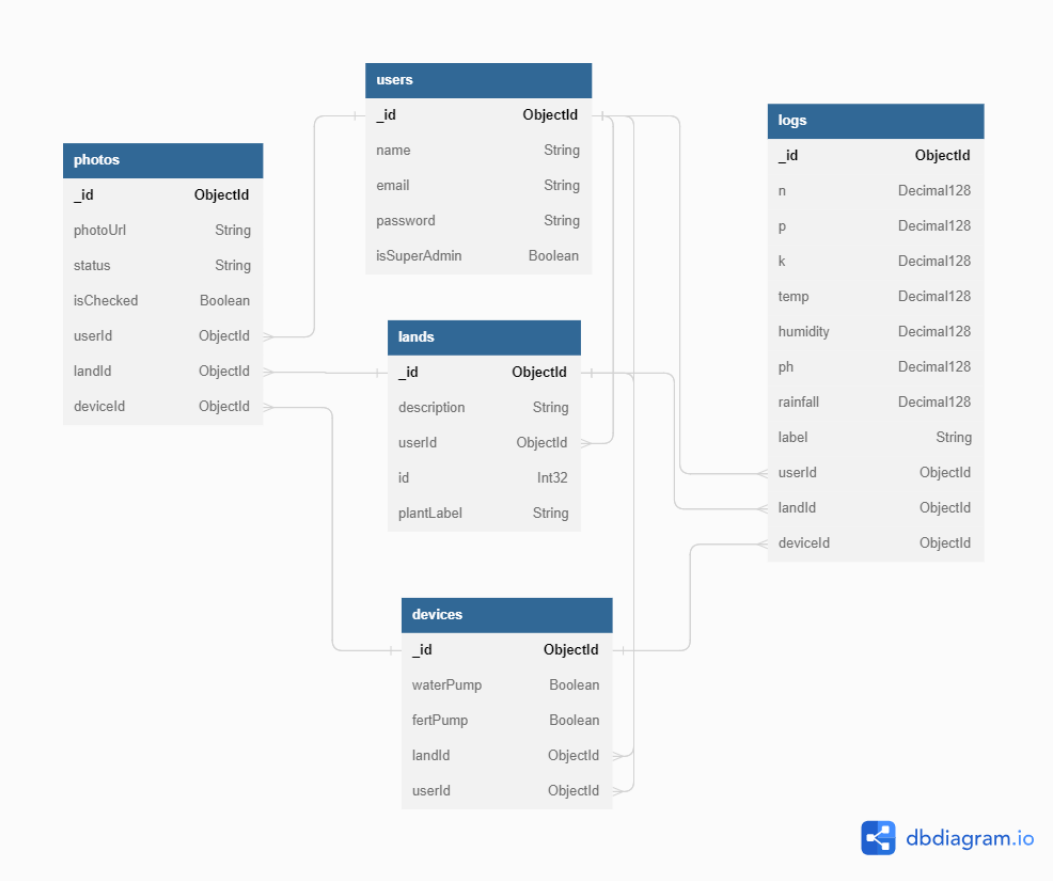


Figure 4 PA Schema

A well-designed database schema is crucial for efficient data management. This entity-relationship diagram (ERD) represents a database design for the precision agriculture application implemented using MongoDB. Here's a description of each collection and its fields:

(a) **Collection Users:** This collection stores information about users of the precision agriculture system. It has the following fields:

- **_id:** A unique identifier for each user stored as an ObjectId.
- **name:** The name of the user is stored as a string.
- **email:** The email address of the user is stored as a string.
- **password:** The password of the user is stored as a string.
- **isSuperAdmin:** A boolean indicating whether the user is a super admin.

(b) **Collection Lands:** This collection stores information about lands or plots associated with users. It has the following fields:

- **_id:** A unique identifier for each land stored as an ObjectId.
- **description:** The description of the land is stored as a string.

- **userId:** A reference to the user who owns the land, stored as an ObjectId.
- **id:** An identifier for the land stored as an Int32.
- **plantLabel:** The label or name associated with the plants in the land is stored as a string.

(c) **Collection Devices:** This collection stores information about users of the precision agriculture system. It has the following fields:

- **_id:** A unique identifier for each device stored as an ObjectId.
- **waterPump:** A boolean indicating the state of a water pump for the device.
- **fertPump:** A boolean indicating the state of a fertilizer pump for the device.
- **landId:** A reference to the land associated with the device, stored as an ObjectId.
- **userId:** A reference to the user who owns the device, stored as an ObjectId.

(d) **Collection Logs:** This collection stores logs or records of various measurements related to precision agriculture. It has the following fields:

- **_id:** A unique identifier for each log stored as an ObjectId.
- **n, p, k, temp, humidity, ph, rainfall:** Decimal128 fields representing different measurements.
- **label:** The label of the preferable plant for cultivation according to the log (by ML) stored as a string.
- **userId:** A reference to the user associated with the log, stored as an ObjectId.
- **landId:** A reference to the user associated with the log, stored as an ObjectId.
- **deviceId:** A reference to the user associated with the log, stored as an ObjectId.

(e) **Collection Photos:** This collection stores information about photos related to precision agriculture. It has the following fields:

- **_id:** A unique identifier for each photo stored as an ObjectId.
- **photoUrl:** The URL or location of the photo is stored as a string.
- **status:** The status of the photo (by ML) is stored as a string.
- **userId:** A reference to the user associated with the photo, stored as an ObjectId.

- **landId:** A reference to the land associated with the photo, stored as an ObjectId.
- **deviceId:** A reference to the device associated with the photo, stored as an ObjectId.

Main sequence of manipulating database through the API:
Here is a sequence of steps for using the API:

(a) **Sign Up:**

- Send a PUT request to the API endpoint for user registration.
- Include the user's name, email, and password in the request body.
- The API will create a new user in the "users" table and return a success message.

(b) **Sign In:**

- Send a POST request to the API endpoint for user authentication.
- Include the user's email and password in the request body.
- The API will validate the credentials and return a JSON Web Token (JWT) as a response.

(c) **Create a Land:**

- Send a POST request to the API endpoint for land creation.
- Include the land description, an ID for the land, and a plant label in the request body.
- The API will create a new entry in the "lands" table, associating it with the corresponding user.

(d) **Add a Device:**

- Send a POST request to the API endpoint for device creation.
- Include the device's water pump and fertilizer pump status, the land ID (from the created land), and the user ID (from the signed-in user) in the request body.
- The API will create a new entry in the "devices" table, associating it with the corresponding land and user.

(e) **Set Up Embedded System:**

- Retrieve the device ID from the API response after adding the device.

- Configure the embedded system using the device ID obtained.
- This step involves programming the embedded system to send logs and photos to the API using the appropriate API endpoints.

(f) **Send Logs and Photos:**

- Configure the embedded system to periodically send logs and photos to the API endpoints.
- For each log or photo, send a POST request to the corresponding API endpoint.
- Include the necessary data such as measurements, labels, URLs, etc., along with the device ID.
- The API will store the logs and photos in the respective tables ("logs" and "photos") and associate them with the user, land, and device.

(g) **Change pumps status of a device:**

- Obtain the device ID associated with the specific device for which you want to change the pump status.
- Send a PUT request to the appropriate API endpoint, including the device ID in the request URL or body.
- In the request payload, specify the new fertPump and waterPump values with their desired statuses (e.g., true for on or false for off).

Throughout the process, access to certain API endpoints and actions, such as creating lands or devices, may require authentication using the JWT obtained during the sign-in process. Additionally, authorization checks may be implemented to ensure that only super admins can perform certain privileged operations (such as getting all logs from all users), a user can be promoted to super admin by directly accessing the database and changing the flag in user collection.

(4) **RESTful API:** In this section, we will discuss strategies for documenting code and accessing the RESTful API through the server URL.[17] The API is accessible via the following root URL: **http://<host>:<port>/path/** In the development phase, the host and port correspond to **"localhost:8080"** as the application was initially developed locally. However, once the PA API is deployed, it can be accessed at URL: **https://pa-api.onrender.com**. Will described the routes used by the front-end to access the database, as follows:

- **Back-end Endpoints**
  - **Authentication**:
    Endpoint to **SIGNUP** a new user in the system:
    (a) Path: /api/auth/signup
    (b) Method: PUT
    (c) Parameters: name, email, password
    (d) Response: A successful request will result in HTTP 201, message (User created.), userId
    Parameter miss will result in HTTP 422.

    Endpoint to **LOGIN** a user or admin in the system:
    (a) Path: /api/auth/login
    (b) Method: POST
    (c) Parameters: email, password
    (d) Response: A successful request will result in HTTP 200, the token used for authentication, userId
    A parameter miss will result in HTTP 401.
  - **Super admin getters:**
    API endpoint to get a **LIST** of all photos accessible by super admin:
    (a) Path: /api/allPhotos
    (b) Method: GET
    (c) Parameters: none
    (d) Response: A successful request will result in an "HTTP 200" Status Code, message (All photos fetched.), photos which is an array with photo objects, totalItems for pagination.
    A request for a user who is not a super admin will result in an "HTTP 401" Status Code.

    Endpoint to get a **LIST** of all logs accessible by super admin:
    (a) Path: /api/allLogs
    (b) Method: GET
    (c) Parameters: none
    (d) Response: A successful request will result in an "HTTP 200" Status Code, message (All photos fetched.), pho-

tos which is an array with photo objects, totalItems for pagination.

A request for a user who is not a super admin will result in an "HTTP 401" Status Code.

Endpoint to get a **LIST** of all devices accessible by super admin:

(a) Path: /api/allDevices

(b) Method: GET

(c) Parameters: none

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (All photos fetched.), photos which is an array with photo objects, totalItems for pagination.

A request for a user who is not a super admin will result in an "HTTP 401" Status Code.

Endpoint to get a **LIST** of all lands accessible by super admin:

(a) Path: /api/allLands

(b) Method: GET

(c) Parameters: none

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (All photos fetched.), photos which is an array with photo objects, totalItems for pagination.

A request for a user who is not a super admin will result in an "HTTP 401" Status Code.

– **Lands:**

Endpoint to get a **LIST** of lands with pagination parameters:

(a) Path: /api/lands

(b) Method: GET

(c) Parameters: none

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Lands fetched.), lands

which is an array with land objects, totalItems for pagination.

A request for a non-authorized user will result in an "HTTP 401" Status Code.

Endpoint to get a **SPECIFIC** land by ID:

(a) Path: /api/lands/:landId

(b) Method: GET

(c) Parameters: landId

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Land fetched.), land object.

A request for a non-authorized user will result in an "HTTP 401" Status Code.

Endpoint to **ADD** a new land to the set:

(a) Path: /api/lands

(b) Method: POST

(c) Parameters: land data

(d) Response: A successful request will result in an "HTTP 201" Status Code, message (Created successfully.), land object.

A parameter miss will result in an "HTTP 422" Status Code.

Endpoint to **DELETE** a specific land by ID:

(a) Path: /api/lands/:landId

(b) Method: DELETE

(c) Parameters: landId

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Land deleted.)

A request for a non-authorized user will result in an "HTTP 401" Status Code.

Endpoint to update a specific land:

(a) Path: /api/updateLands

(b) Method: POST

(c) Parameters: landId, updated land data

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Updated successfully.)

A request for a non-authorized user will result in an "HTTP 401" Status Code.

– **Devices:**

Endpoint to get a **LIST** of devices with pagination parameters:

(a) Path: /api/devices

(b) Method: GET

(c) Parameters: none

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Devices fetched.), devices which is an array with device objects, totalItems for pagination.

A request for a non-authorized user will result in an "HTTP 401" Status Code.

Endpoint to get a **SPECIFIC** device by ID:

(a) Path: /api/devices/:deviceId

(b) Method: GET

(c) Parameters: deviceId

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Device fetched.), device object.

A request for a non-authorized user will result in an "HTTP 401" Status Code.

Endpoint to **ADD** a new device to the set:

(a) Path: /api/devices

(b) Method: POST

(c) Parameters: device data, landId

(d) Response: A successful request will result in an "HTTP 201" Status Code, message (Created successfully.), device object.

A parameter miss will result in an "HTTP 422" Status Code.

Endpoint to **DELETE** a specific device by ID:

(a) Path: /api/devices/:deviceId
(b) Method: DELETE
(c) Parameters: deviceId
(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Device deleted.).

A request for a non-authorized user will result in an "HTTP 401" Status Code.

Endpoint to **update** a specific device:

(a) Path: /api/updateDevice
(b) Method: POST
(c) Parameters: deviceId, updated device data
(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Updated successfully.).

A request for a non-authorized user will result in an "HTTP 401" Status Code.

– **Logs:**

Endpoint to get a **LIST** of logs with pagination parameters:

(a) Path: /api/logs
(b) Method: GET
(c) Parameters: none
(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Logs fetched.), devices which is an array with device objects, totalItems for pagination.

A request for a non-authorized user will result in an "HTTP 401" Status Code.

Endpoint to get a **SPECIFIC** log by ID:

(a) Path: /api/logs/:logId
(b) Method: GET
(c) Parameters: logId

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Log fetched.), log object.
A request for a non-authorized user will result in an "HTTP 401" Status Code.

Endpoint to **ADD** a new log to the set:

(a) Path: /api/logs

(b) Method: POST

(c) Parameters: log data

(d) Response: A successful request will result in an "HTTP 201" Status Code, message (Created successfully.), log object.
A parameter miss will result in an "HTTP 422" Status Code.

Endpoint to **DELETE** a specific log by ID:

(a) Path: /api/logs/:logsId

(b) Method: DELETE

(c) Parameters: logsId

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Log deleted.).
A request for a non-authorized user will result in an "HTTP 401" Status Code.

Endpoint to get a **LIST** of logs for a specific land with pagination parameters:

(a) Path: /api/landLogs/:landId

(b) Method: GET

(c) Parameters: landId

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Logs fetched.), logs which is an array with log objects, totalItems for pagination.
A request for a non-authorized user will result in an "HTTP 401" Status Code.

– **Photos:**
Endpoint to get a **LIST** of photos with pagination parameters:

(a) Path: /api/photos

(b) Method: GET

(c) Parameters: none

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Photos fetched.), devices which is an array with photo objects, totalItems for pagination.

A request for a non-authorized user will result in an "HTTP 401" Status Code.

Endpoint to get a **SPECIFIC** photo by ID:

(a) Path: /api/photos/:photoId

(b) Method: GET

(c) Parameters: photoId

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Photo fetched.), photo object.

A request for a non-authorized user will result in an "HTTP 401" Status Code.

Endpoint to **ADD** a new photo to the set:

(a) Path: /api/photos

(b) Method: POST

(c) Parameters: photo file, deviceId

(d) Response: A successful request will result in an "HTTP 201" Status Code, message (Created successfully.), photo object.

A parameter miss will result in an "HTTP 422" Status Code.

Endpoint to **DELETE** a specific photo by ID:

(a) Path: /api/photos/:photoId

(b) Method: DELETE

(c) Parameters: photoId

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Photo deleted.).

A request for a non-authorized user will result in an "HTTP 401" Status Code.

Endpoint to get a **LIST** of photo for a specific land with pagination parameters:

(a) Path: /api/landPhotos/:landId

(b) Method: GET

(c) Parameters: landId

(d) Response: A successful request will result in an "HTTP 200" Status Code, message (Photos fetched.), logs which is an array with log objects, totalItems for pagination.

A request for a non-authorized user will result in an "HTTP 401" Status Code.

3 Embedded System

3.1 Introduction

The embedded system for your innovative farm is divided into three main parts: the lighting part, the irrigation part, and the camera part.

The lighting part includes a light sensor that measures the amount of light in the farm and controls the lighting system accordingly. Based on the data collected by the sensor, the system can adjust the lighting to provide optimal conditions for the plants to grow. This can include adjusting the intensity and color of the light to mimic natural sunlight and promote photosynthesis.

The irrigation part includes a soil moisture sensor and other sensors and a pump that controls the water supply to the plants. The soil moisture sensor measures the moisture level in the soil and sends data to the system. The system can then use this data to determine when to turn the pump on or off, based on the moisture needs of the plants. This ensures that the plants are getting the right amount of water and helps prevent over-watering or under-watering. The camera part includes a camera module that captures images of the farm and sends them to the system for analysis. The system can use image processing and machine learning algorithms to identify plant issues or anomalies, such as pests or diseases. This can help farmers proactively prevent crop damage and improve yields [18].

Overall, the different parts of the embedded system work together to create a comprehensive and automated solution for smart farming. By integrating different sensors and controllers, the system provides real-time monitoring and control of the environmental factors that affect plant growth and health. This can help farmers optimize their practices, reduce waste, and increase yields. All of these values of sensors are displayed on a display screen throw over periods of time.

Table 2 Components used in this simulation

Component	Type	Model	Count
Raspberry pi	Single-board computer	Raspberry pi 4	1
Humidity and temperature sensor	Digital Sensor	DHT11	1
Light sensor	Digital Sensor	S1133 SI photodiode	1
Water level sensor	Analog Sensor	SKU:EB0045619	1
Soil moisture sensor	Digital Sensor	FC-28 (resistive soil moisture sensor)	1

PH sensor	Analog Sensor	SKU SEN0161	1
CO2 sensor	sensor	Adafruit CCS811	1
Analog to digital converter (ADC)	IC	MCP3008 8-channel 10-bit	1
Display screen	LCD screen		1
wires			
DC motor driver		L298N	1
DC motor			
Capacitors	Various capacities according to its usage		
breadboard			
PIR sensor			
Rain sensor			

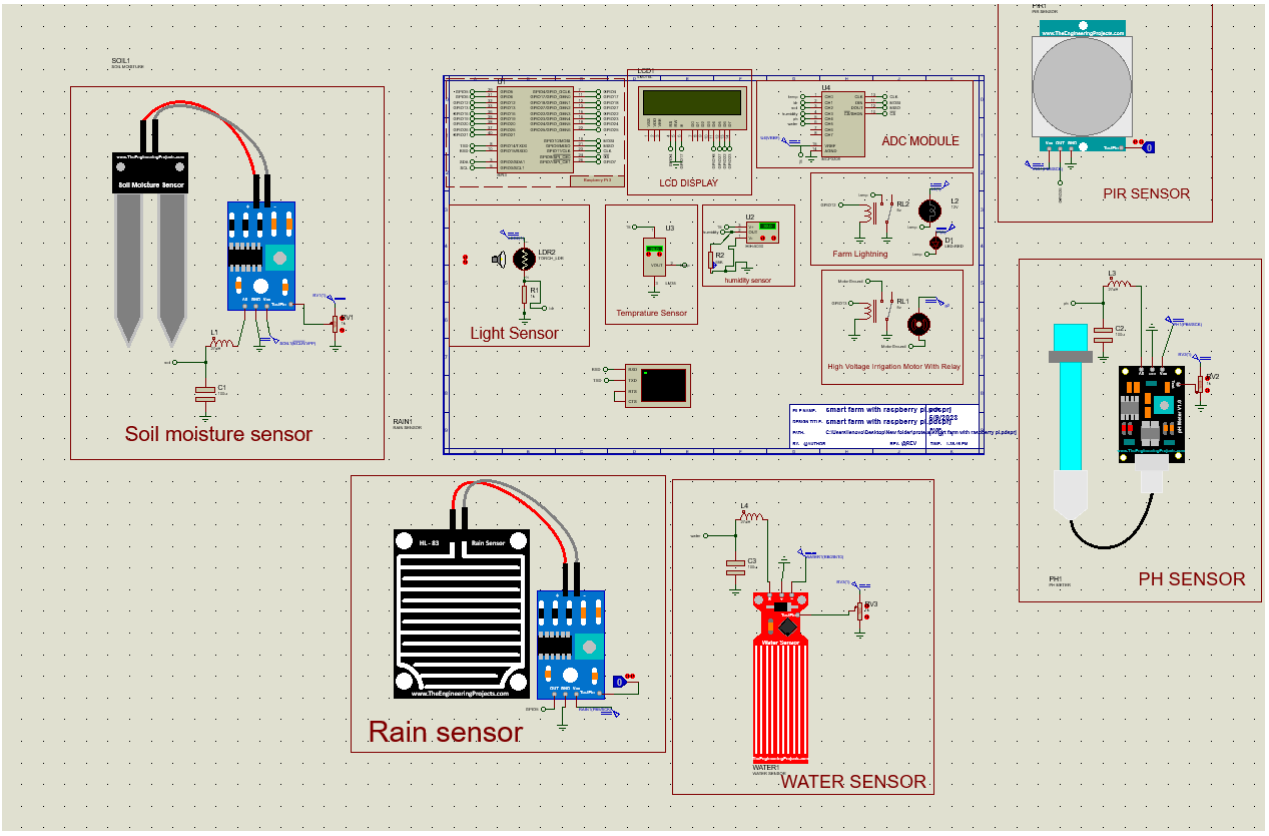


Figure 5 Simulation Circuit

3.2 Illustration for the embedded system components:

We used a software called Proteus version 8.9. This software is paid software but this software contains a lot of libraries for sensors and other components also you can find many libraries for a lot of sensors on the internet. You can use this software

for raspberry pi and Arduino simulation because it supports coding by Python and C languages and also supports coding by a flow chart.

Note: If you want to use raspberry pi you should use Proteus version 8.9 or above.

3.2.1 Basic components:

Raspberry Pi: In this simulation, we use raspberry pi 3 to take control of all the components of the embedded system In this specific project of building an embedded system for a smart farm, using a Raspberry Pi can be more efficient than using an Arduino for several reasons.

Firstly, the Raspberry Pi's processing power and memory capacity are better suited to handle the complex and data-intensive tasks required in a smart farming system. The system requires the processing of large amounts of sensor data, which need to be analyzed in real-time to make decisions on irrigation, lighting, and other environmental factors. The Raspberry Pi's more powerful processor and greater memory capacity make it more efficient at processing and analyzing this data.

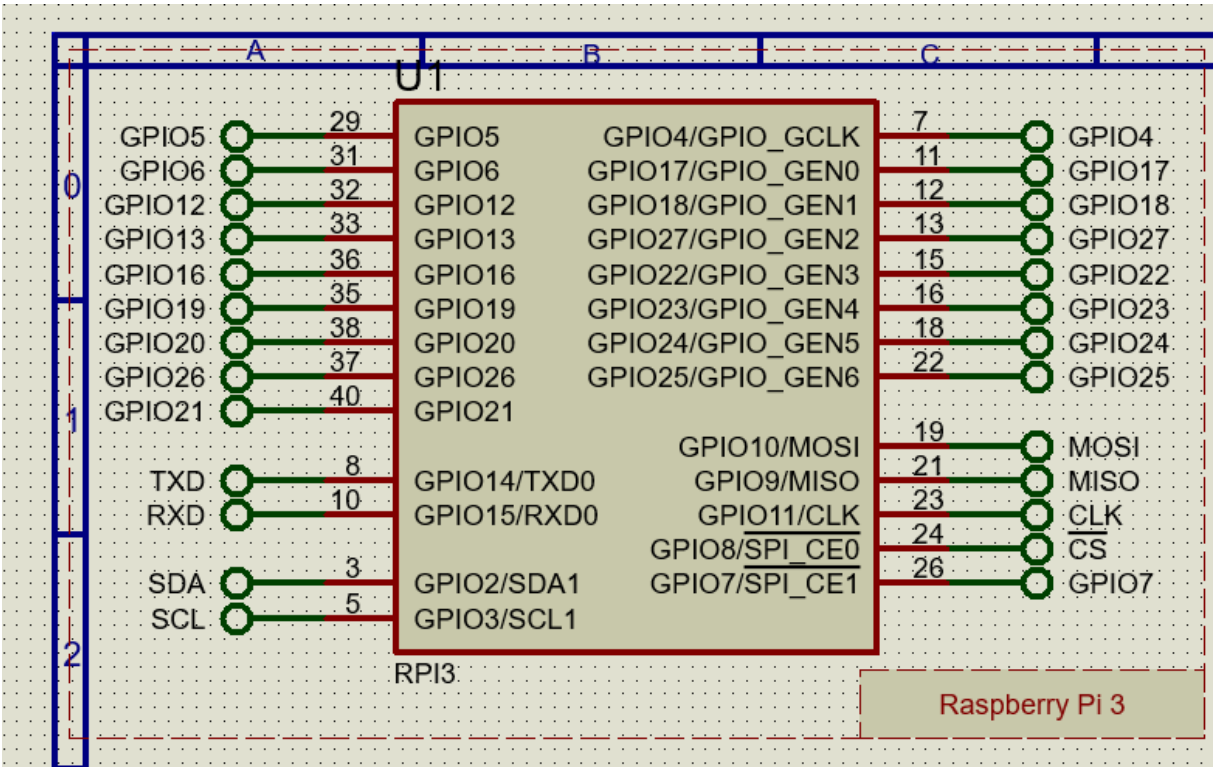


Figure 6 Raspberry Pi

Secondly, the Raspberry Pi's built-in networking capabilities, including Wi-Fi and Ethernet support, make it easier to access and control the system from remote

locations. In a smart farming system, it is important to have the ability to monitor and control the farm from anywhere, and the Raspberry Pi's networking capabilities make this more efficient and convenient.

Thirdly, the Raspberry Pi's compatibility with a wide range of programming languages, libraries, and software tools makes it easier to develop and customize the system to suit specific needs. This is particularly important in a smart farming system, where different sensors and actuators may need to be integrated and controlled in different ways.

Finally, the Raspberry Pi's support for advanced graphics and visualization capabilities can be useful in a smart farming system. For example, the system can display graphs and charts that provide a visual representation of the sensor data, making it easier to identify patterns and trends that may not be immediately apparent from the raw data [19].

Overall, in this project, the Raspberry Pi's processing power, memory capacity, networking capabilities, programming flexibility, and graphics capabilities make it a more efficient and effective choice than an Arduino for building an embedded system for a smart farm [20].

Display screen: This is a display screen that displays the values that we get from the sensors and any messages that refer to some actions according to the values of the sensors.

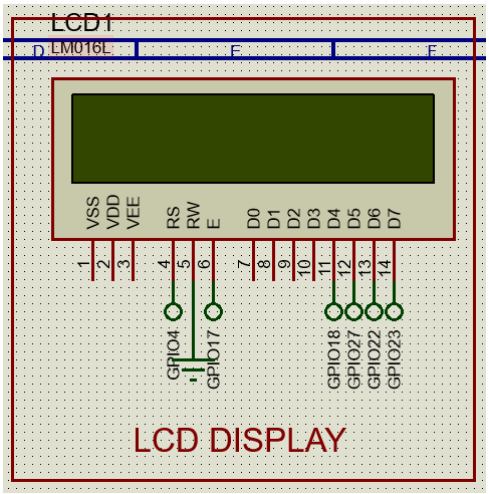


Figure 7 LCD Display

ADC module: The analog-to-digital converter (ADC) module is a key component in many embedded systems, as it allows analog signals to be converted into digital signals that can be processed by a microcontroller or other digital device.

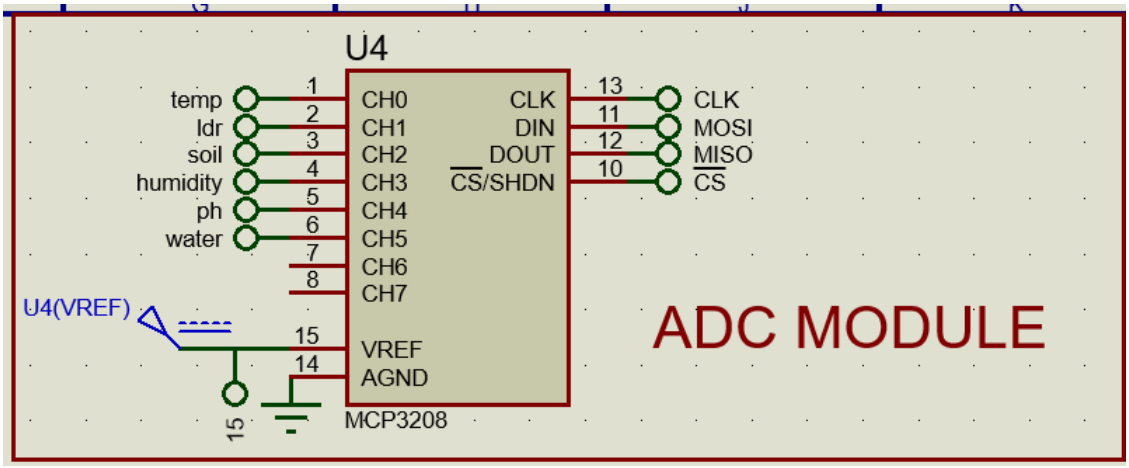


Figure 8 ADC Module

There are several benefits to using an ADC module in an embedded system:

- Improved accuracy: Analog signals are subject to noise, distortion, and other factors that can affect their accuracy. By converting analog signals to digital signals, it is possible to eliminate some of these sources of error and achieve more accurate measurements.
- Increased resolution: Digital signals can represent a wider range of values than analog signals, allowing for higher resolution measurements. For example, a $12 - bit$ ADC can represent $2^12(4096)$ distinct values, while an analog signal can only represent a continuous range of values.
- Compatibility with digital systems: Many microcontrollers and other digital devices are designed to work with digital signals, and may not be able to process analog signals directly. By using an ADC module to convert analog signals to digital signals, it is possible to integrate analog sensors and other devices into a digital system.
- Flexibility: ADC modules can be configured to measure a wide range of input signal types, including voltage, current, temperature, and more. This allows them to be used in a variety of applications, from industrial control systems to medical devices.

Overall, the ADC module is an essential component in many embedded systems, allowing analog signals to be processed and integrated with digital systems in a wide range of applications.

3.2.2 Lighting part:

The lighting part of the embedded system for the smart farm is designed to optimize plant growth and health by providing the right amount and quality of light. The system

includes a light sensor that measures the amount of light in the farm and a lighting system that can be controlled based on the sensor data. The system uses machine learning algorithms to analyze the data and adjust the lighting to provide optimal conditions for the plants to grow. The lighting system includes LED lights that can be adjusted in terms of intensity and color to mimic natural sunlight. The system can be programmed to provide different lighting conditions based on the stage of plant growth, with different light wavelengths and intensities to promote photosynthesis and other biological processes.

The system is designed to be flexible and adaptable, with options for manual control or automatic control based on sensor data. The sensor data is processed and analyzed in real time, allowing the system to respond quickly to changes in the environment and adjust the lighting accordingly.

Overall, the lighting part of the embedded system for the smart farm is an essential component in optimizing plant growth and health. By providing the right amount and quality of light, the system can help farmers improve their crop yields and reduce waste. The system's flexibility and adaptability make it a valuable tool for smart farming practices, enabling farmers to optimize their practices and achieve better results.

Light sensor: We use a photoresistor sensor for light intensity. A photoresistor (also known as a light-dependent resistor or LDR) is a type of sensor that changes its resistance in response to changes in light intensity. As the light intensity increases, the resistance of the photoresistor decreases, and as the light intensity decreases, the resistance increases. Photoresistor sensors are widely used in a variety of applications, including light meters, automatic street lights, and camera exposure control systems. They are also commonly used in smart farming systems, where they can be used to monitor the amount of light in a greenhouse or other farm environment.

In a smart farming system, a photoresistor sensor can be integrated with a microcontroller or other digital device to measure the light intensity and adjust the lighting system accordingly. For example, if the light intensity is too low, the microcontroller can turn on additional lights or adjust the intensity of existing lights to provide optimal conditions for plant growth. Likewise, if the light intensity is too high, the microcontroller can reduce the intensity of the light to prevent damage to the plants.

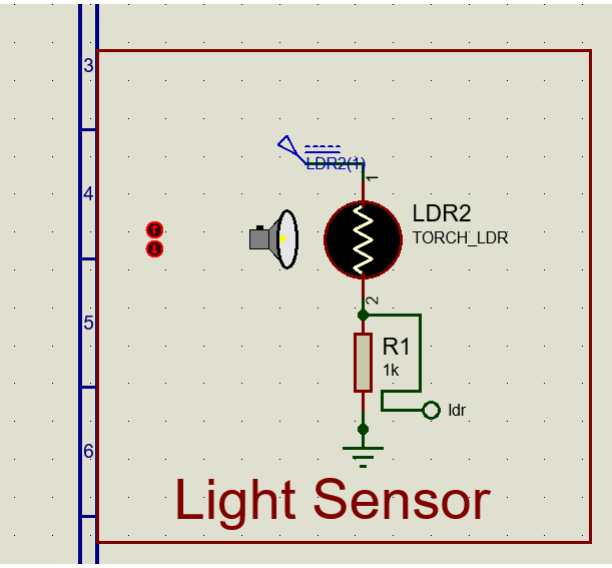


Figure 9 Light Sensor

Photoresistor sensors are relatively inexpensive and easy to use, making them a popular choice for many embedded systems. However, they do have some limitations, including sensitivity to temperature changes and a limited dynamic range. To overcome these limitations, some systems may use more advanced sensors, such as photodiodes or phototransistors, which offer higher sensitivity and a wider dynamic range [21].

PIR sensor: For detecting motion we use a PIR sensor. A **PIR (Passive Infrared)** sensor is a type of motion sensor that detects changes in infrared radiation emitted by moving objects. It works by detecting the differences in temperature between an object and its surrounding environment. PIR sensors are commonly used in security systems, lighting control systems, and other applications where motion detection is required. In a smart farming system, PIR sensors can be used to detect the presence of animals or humans in the farm environment, as well as to monitor the movement of plants and other objects. The PIR sensor consists of two components: a pyroelectric sensor and a Fresnel lens. The pyroelectric sensor is made of a crystalline material that generates a voltage when exposed to changes in temperature, while the Fresnel lens is used to focus the infrared radiation onto the sensor. When a moving object enters the sensor's field of view, the infrared radiation emitted by the object is focused onto the pyroelectric sensor, causing it to generate a voltage. The sensor then sends a signal to a microcontroller or other digital device, which can be programmed to perform a specific action, such as turning on a light or sounding an alarm.

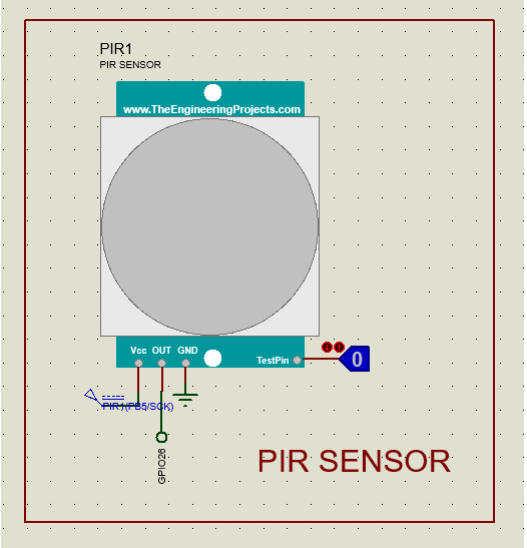


Figure 10 PIR Sensor

PIR sensors are low-power and relatively inexpensive, making them a popular choice for many embedded systems. However, they do have some limitations, including a limited detection range and sensitivity to changes in temperature and humidity. To overcome these limitations, some systems may use additional sensors or employ advanced signal processing techniques to improve the accuracy and reliability of the motion detection system [22].

Farm lights: These are some lights that are used on the farm.

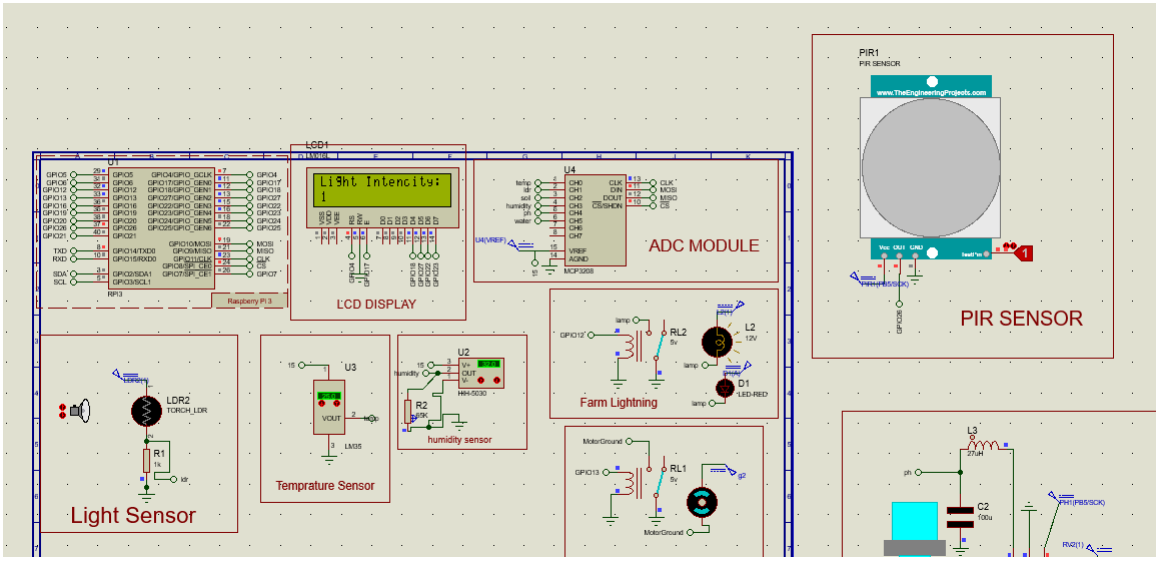


Figure 11 Farm Lighting

Algorithm of lighting part:

We used a light sensor and PIR sensor in this part to control Farm lights Firstly we used an LDR sensor to get the light intensity value, As we see in Fig.12 our display screen displays the value of light intensity value which is 1 this mean that the light is very low and the farm is dark now which may mean that it is night or there is something that prevents the light from our farm.

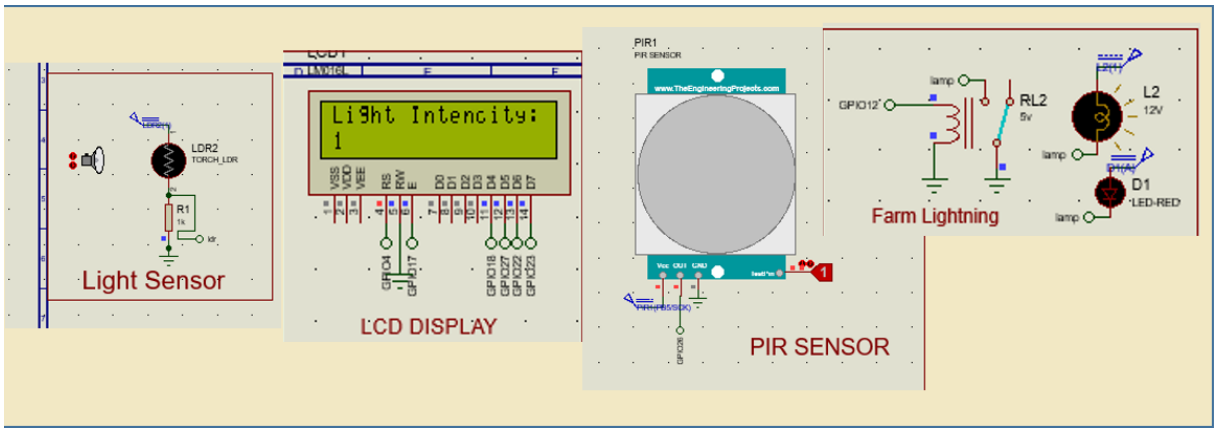


Figure 12 Raspberry Pi detects the value of light intensity from light sensors

secondly, the PIR sensor is used to detect if there are some persons inside the farm, as we see in Fig.13 the sensor has a logic state that equals 1 which means that the state of founding persons inside the farm is true.

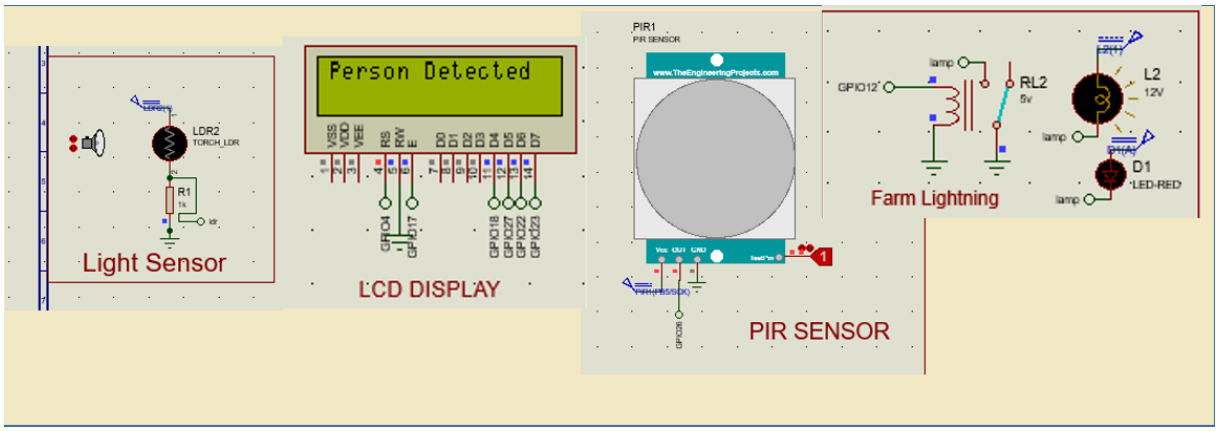


Figure 13 PIR detected there is a person inside the farm

The coding algorithm is that if the light intensity value is lower than 250 And there are some people on the farm so we should turn on the Farm lights As you see in Fig.14 the lights are turned on and a message was displayed on the screen referring that farm lights being turned on Overall this part detects if there are some persons on the farm

and then turns on the lights if the light is low this part also sends the values of the lights to the web server so the AI model can deal with it.

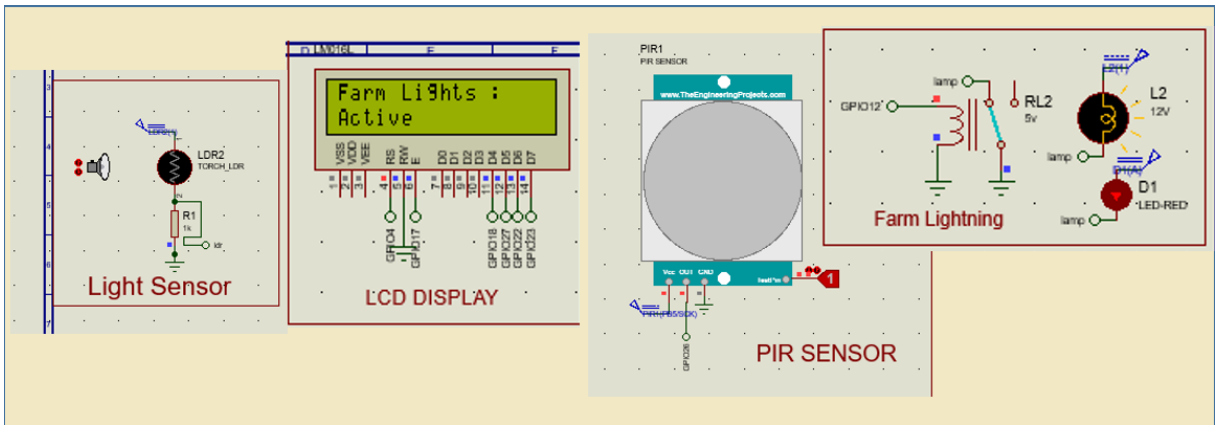


Figure 14 Light value is lower than 250 and there are persons inside the farm

3.2.3 Irrigation part:

The irrigation part of a smart farm project is an essential component for ensuring the growth and health of plants. In a smart farm, the irrigation system is automated and can be controlled based on environmental data, such as temperature, humidity, and soil moisture levels. This allows for more efficient use of water resources and can help reduce water waste.

The irrigation system includes a network of soil moisture sensors that are placed at various locations on the farm. These sensors measure the moisture content of the soil and send data to a microcontroller or other digital device for analysis. The microcontroller can then activate the irrigation system when the soil moisture levels are below a certain threshold.

The irrigation system can be designed to be flexible, with options for manual control or automatic control based on sensor data. The system can also be programmed to provide different amounts of water based on the stage of plant growth, with different irrigation schedules for different crops.

In addition to the soil moisture sensors, the irrigation system may also include weather sensors that measure temperature, humidity, and other environmental factors. This data can be used to adjust the irrigation schedule to account for changes in weather conditions and to optimize water usage.

The irrigation system can be powered by solar panels and can be designed to be energy-efficient, using water pumps and other components that consume minimal

power. The system can also be designed to be scalable, allowing for expansion as the farm grows and new crops are added.

Overall, the irrigation part of a smart farm project is an essential component for ensuring the efficient use of water resources and the growth and health of plants. By using advanced sensors and automation technology, a smart irrigation system can help farmers optimize their water usage and improve their crop yields.

Temperature sensor: Temperature sensors play a critical role in smart farms by providing accurate and timely information about the temperature of the farm environment. They are used to measure the temperature of the air, soil, and water, and this information can be used to make informed decisions about crop management. One of the main jobs of temperature sensors in a smart farm is to monitor the temperature of the soil. Soil temperature can have a significant impact on crop growth and development, and different crops have different temperature requirements. In addition to monitoring temperature, temperature sensors can also be used to detect temperature changes that may indicate a problem in the farm environment. For example, sudden drops in temperature may indicate a malfunction in the heating or cooling systems, while sudden increases in temperature may indicate a fire or other safety hazard [23].

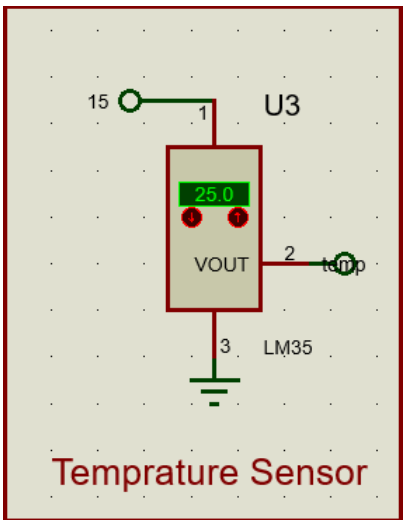


Figure 15 Temprature Sensor

Overall, the job of temperature sensors in a smart farm is to provide accurate, real-time information about the temperature of the farm environment, which can be used to optimize crop growth and ensure the safety and efficiency of farm operations. By using temperature sensors in conjunction with other types of sensors and data analytics tools, farmers can create a comprehensive view of their

farm environment, and make data-driven decisions to improve their crop yields and profitability.

Humidity sensor: Humidity sensors are an important component of smart farm systems. They are used to measure the amount of moisture in the air and soil, which is critical for plant growth and health. Humidity sensors can also be used to monitor the humidity levels in the greenhouse or other growing environments and to control the ventilation and cooling systems to maintain optimal growing conditions. Humidity sensors can be used in a variety of ways in a smart farm system. For example, they can be used to monitor the humidity levels in the greenhouse or other growing environments and to control the ventilation and cooling systems to maintain optimal growing conditions. They can also be used to monitor the humidity levels in the soil and to control irrigation systems to ensure that plants receive the optimal amount of water. Overall, the job of humidity sensors in a smart farm is to provide accurate and timely information about the humidity levels in the farm environment, which can be used to optimize crop growth and ensure the efficiency and safety of farm operations. By using humidity sensors in conjunction with other types of sensors and data analytics tools, farmers can create a comprehensive view of their farm environment, and make data-driven decisions to improve their crop yields and profitability [24]. In our project, we use low voltage humidity sensors, which are a type of humidity sensor that operate at low voltages, typically less than 5 volts. Low voltage humidity sensors work by measuring the electrical capacitance or resistance of a material, which varies with the amount of moisture present in the environment. They typically use a microcontroller or other electronic circuit to measure the capacitance or resistance and convert this into a humidity reading.

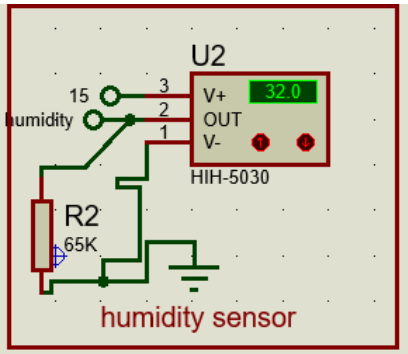


Figure 16 Humidity Sensor

One advantage of low-voltage humidity sensors is that they consume less power than other types of humidity sensors, which makes them ideal for use in battery-powered or low-power applications. They are also typically smaller and more

compact than other types of sensors, which makes them easier to integrate into smart farm systems.

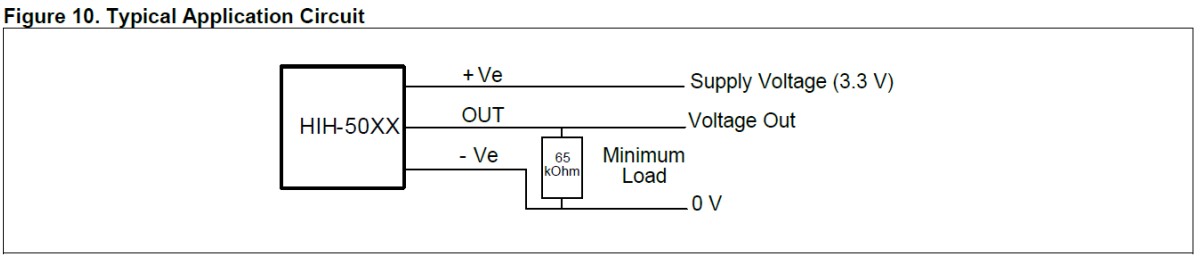


Figure 17 Typical Application Circuit

According to the datasheet of these sensors, we see the typical application circuit which we implement in our system.

Soil moisture sensor: Soil moisture sensors measure the amount of water in the soil they work by measuring the electrical conductivity of the soil, which changes as the soil moisture content changes. There are several types of soil moisture sensors available, including resistive sensors, capacitance sensors, and time-domain reflectometry (TDR) sensors.

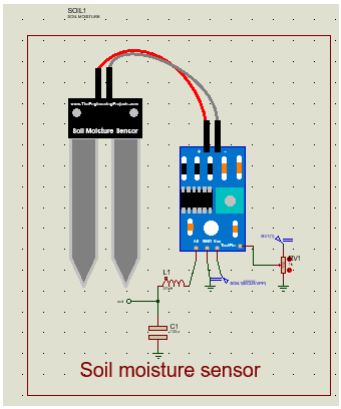


Figure 18 Soil Moisture Sensor

Resistive sensors work by measuring the resistance of a soil sample between two electrodes. As the soil moisture content increases, the resistance decreases, and vice versa. Capacitance sensors work by measuring the change in capacitance between two electrodes as the soil moisture content changes. TDR sensors work by measuring the time it takes for an electromagnetic pulse to travel through the soil and reflect back to the sensor. The moisture content of the soil affects the speed of the electromagnetic pulse, allowing the sensor to calculate the soil moisture content. Overall, soil moisture sensors are an important tool for anyone involved in agriculture or gardening, as they can help ensure that plants

receive the right amount of water at the right time, leading to healthier plants and improved crop yields [25].

Water Level sensor: A water level sensor is an electronic device that is used to detect and measure the level of water in a tank, reservoir, well, or any other container. The sensor provides accurate information about the water level, which can be used for a variety of purposes, such as monitoring water usage, preventing overflow, and controlling the filling and draining of the container.

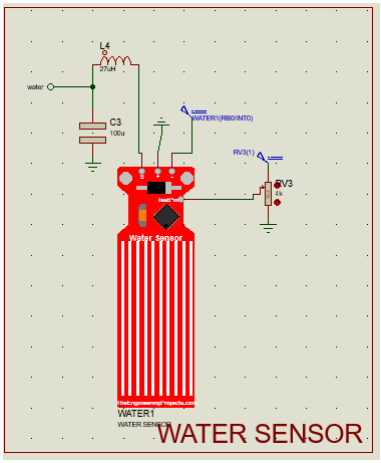


Figure 19 Water Sensor

There are various types of water level sensors available, including float sensors, ultrasonic sensors, pressure sensors, and capacitive sensors. Each type of sensor operates on a different principle and has its advantages and disadvantages, in this project, we use float sensors. Float Sensors: These sensors use a float that rises and falls with the water level. As the float moves, it activates a switch that sends a signal to the control system [26].

Rain Drop sensor: Rain drop sensors are electronic devices that are used to detect the presence of rain or moisture. They are commonly used in weather monitoring systems, irrigation systems, and other applications where it is important to detect the presence of rain. Raindrop sensors work by measuring the electrical conductivity of the water droplets that fall on the sensor's surface. When rain falls on the sensor's surface, it creates a conductive path between two or more electrodes on the sensor. The sensor then detects this change in conductivity and sends a signal to the control system.

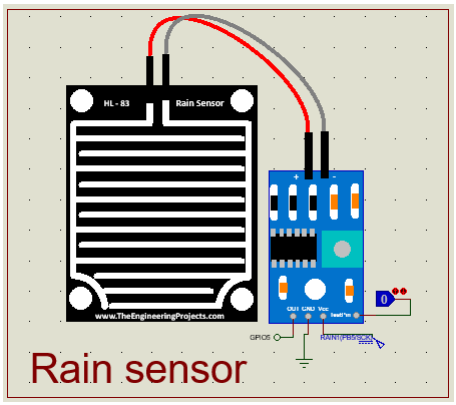


Figure 20 Rain Sensor

There are two types of raindrop sensors: analog and digital. Analog sensors measure the resistance between two or more electrodes on the sensor, and this resistance changes as the water droplets fall on the sensor. Digital sensors, on the other hand, use a microcontroller to measure the time it takes for the conductive path to be established between the electrodes. This information is then used to calculate the amount of rain that has fallen [27].

PH sensor: A **PH** sensor is an electronic device that is used to measure the acidity or alkalinity of a liquid solution. PH is a measure of the concentration of hydrogen ions **(H+)** in a solution, and it is an important parameter in many industrial, environmental, and biological applications. **PH** is the unit of measure that describes the degree of acidity or alkalinity. It is measured on a scale of 0 to 14. A PH sensor works by measuring the voltage or potential difference between two electrodes placed in the solution being measured. The electrodes are typically made of glass and are coated with a special gel that is sensitive to changes in PH. When the electrodes come into contact with a solution, the gel on the electrode surface interacts with the ions in the solution, creating a potential difference that is measured by the sensor.

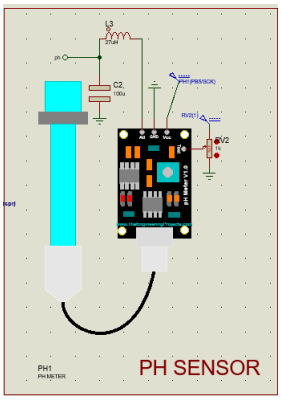


Figure 21 PH Sensor

There are several types of pH sensors available on the market, including glass electrode sensors, ion-selective field-effect transistor **(ISFET)** sensors, and optical sensors. Glass electrode sensors are the most common type of pH sensor and are widely used in laboratory and industrial applications. **ISFET** sensors and optical sensors are newer technologies that offer advantages such as smaller size, lower power consumption, and higher accuracy [28].

Irrigation motor: This is a motor that is used to irrigate the farm according to the reading of the sensors which are used in this part of our project.

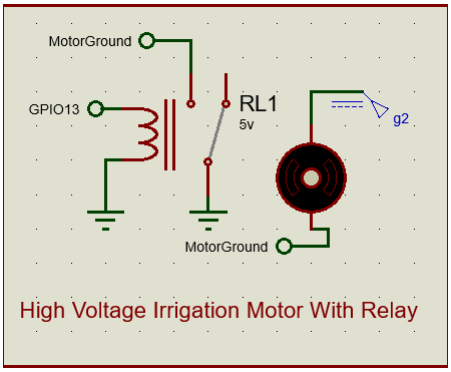


Figure 22 Irrigation Motor

Algorithm of irrigation part:

This part of the farm is used to irrigate our farm according to specific algorithms. So in this part, we use a temperature sensor first to check the temperature value. Another sensor we used in this part is the water level sensor that is used to detect the value of the water level on our farm. Also, we used a soil moisture sensor to check the soil moisture value. Finally, we used a raindrop sensor to check if there is a raindrop or not. **So how irrigation occurs? First** temperature sensor checks the value of the temperature, if the value is high (higher than $30\,℃$ according to our algorithm) then another sensor is check its value here which is the soil moisture sensor, this sensor checks the value of the moisture of farms soil if this value is low (lower than 100 according to our algorithm), then the system turns on the irrigation motor. So after the system turned on the motor, the system must check the value of the water level in our farm, so we used a water level sensor to check this value, if this value is high (higher than 300 according to our algorithm) then the system turns off the motor. All values of our sensors are displayed on the display screen. Also, our system uses a raindrop sensor to check if there is rain on our farm, if there is rain then our system turns off the motor, so we can save some power and save the usage of water. Also when rain drop is turned on it displays rain warning on our screen. **Finally**, when rain is stopped our system checks the values of our sensors and then decides if the system needs to turn on the motor

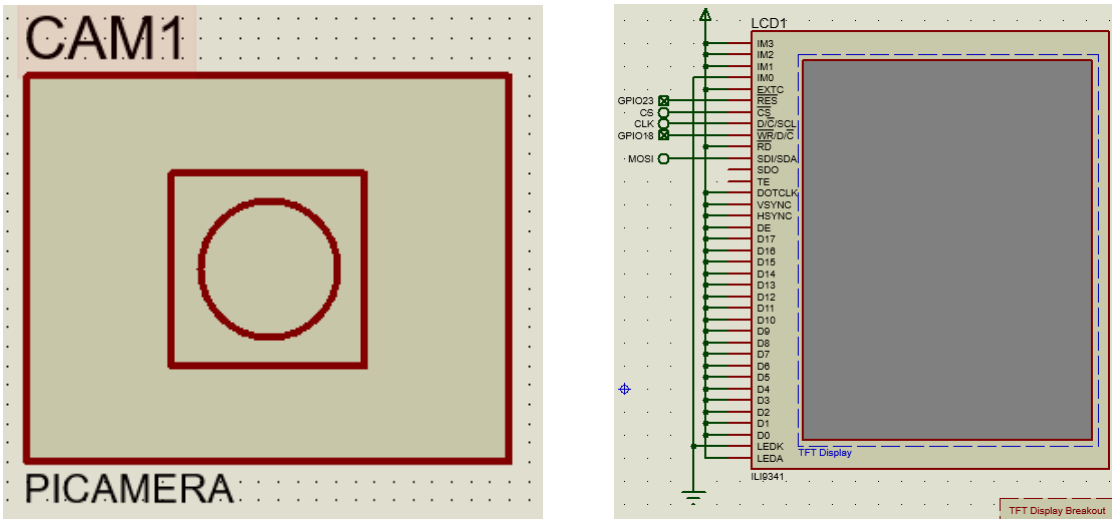
again or not. Humidity sensors and PH sensors are not fully used in the irrigation part, but we use the values that are measured from these sensors in the artificial intelligence part, which may affect our irrigation, as well as they are displayed on our screen and also displayed on the website.

3.2.4 Camera part:

In this part, cameras can play an important role in monitoring various aspects of the farm. Cameras can be used to capture images or videos of crops and can provide valuable information to farmers and other stakeholders. This part of our project is important for crop monitoring. By installing cameras in strategic locations throughout the farm, farmers can capture images or videos of their crops and use the part of artificial intelligence which is related to the embedded system part in this project to analyze the data. This can help identify early signs of plant diseases, nutrient deficiencies, and other issues that may impact crop yield or quality. With this information, farmers can take proactive measures to address the problems and optimize their crop production. This part consists of 2 components:

(1) Camera: This component is used to get some images or sometimes to get videos about our crops then it will be sent to the cloud.

(2) Display screen for photos: This component is used to see the images that are taken.
Note: This component is used only for the simulation but in fact, we will display our images on our website or application instead of this screen.



(a) PI Camera (b) Display Screen for simulation

Figure 23 Camera part components

Algorithm of irrigation part:

When the system is on the camera captures an image for the crops every 30 minutes and displayed it then it will send it to the cloud to get it on the website and deal with it by AI model. Also, the camera is linked with a mobile application which we can control with this app.

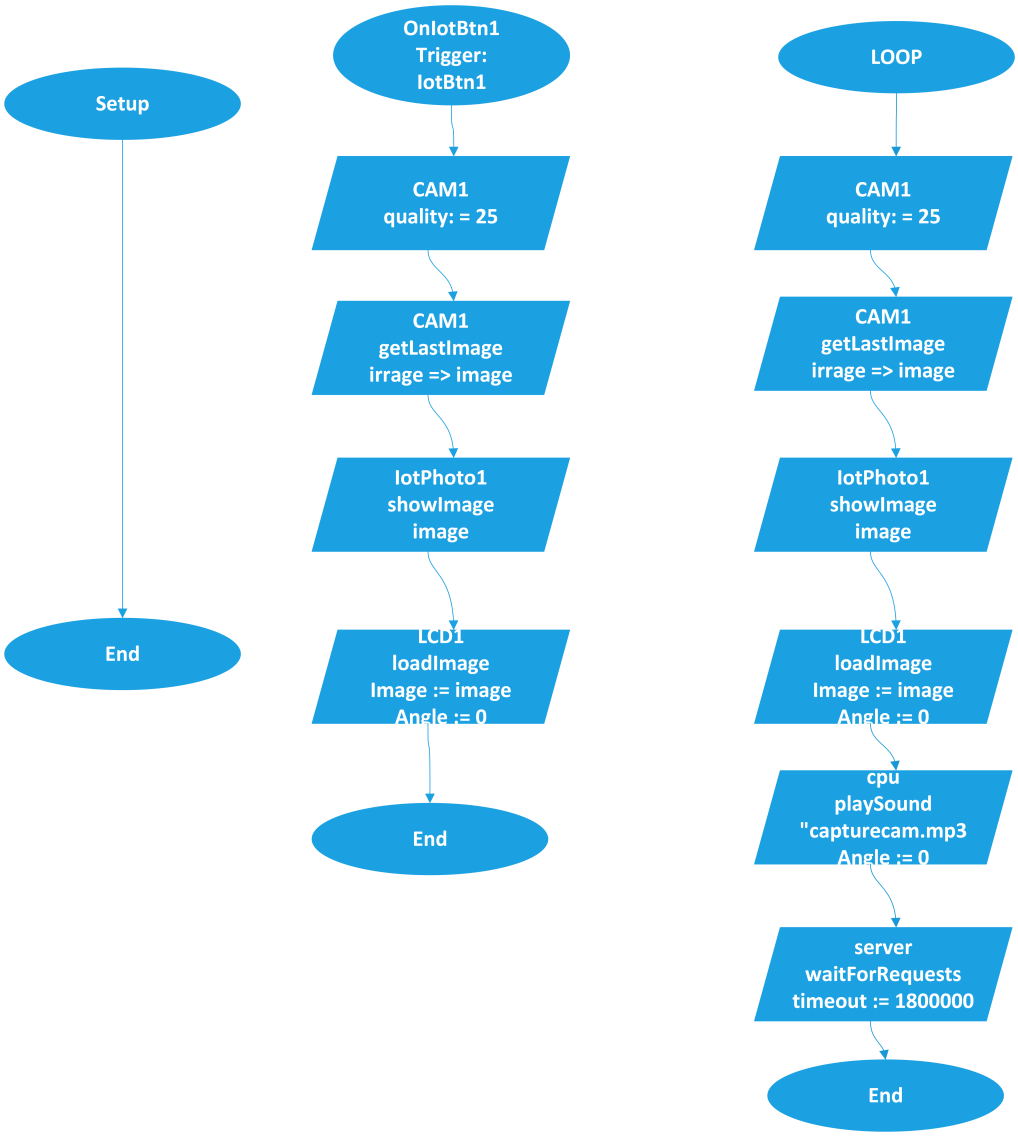


Figure 24 Camera system flowchart

4 Mobile Development

4.1 Android

People are known to constantly utilize their mobile phones, hence initiatives that rely on smartphones have grown in popularity. Because smartphones are simpler, less expensive, and more entertaining. How do you feel about utilizing a mobile app to get services or spend time? Fortunately, apps are available for free in the app store. Consider clicking on a mobile touchscreen and saving many hours of your time. However, it has some development and security challenges.

4.2 Installation

Before beginning to write Android code, which consists mostly of the Java programming language, the XML graphical user interface, and resource files. To begin developing an Android app, you must first prepare the necessary tools. You will be relieved to learn that you can begin developing Android applications on any of the operating systems listed below. Microsoft Windows, Mac OS, and Linux are all options. The second aspect is that all of the tools needed to develop Android applications are free and can be downloaded from the Internet. The following is a list of software that you will require before beginning your Android application programming.

(1) Android SDK.

(2) Java Runtime Environment (JRE).

(3) Android Studio or Eclipse IDE.

(4) Java JDK 17.0.6 or later version.

To install Android Studio on your Mac, follow these steps:

(1) Launch the Android Studio DMG file.

(2) Drag and drop Android Studio into the Applications folder, then launch Android Studio.

(3) Choose whether to import previous Android Studio settings, then click OK.

(4) Complete the Android Studio Setup Wizard, which includes downloading the Android SDK components that are required for the development:

- Once downloaded Android Studio, then double-click the Android Studio dmg file to start the installation. Once you double-click, you will see a screen like the below image [29].

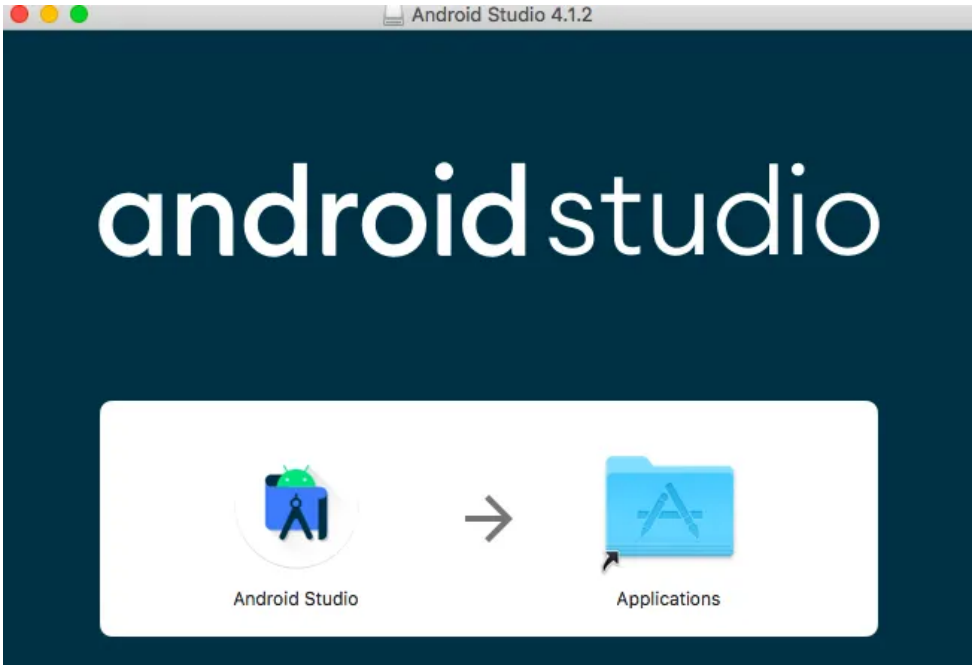


Figure 25 Step 1

- Simply drag the Android Studio to the Application Folder. It will copy the file and start the installation process. Then a security pop-up will appear. To proceed with the installation, click the open button.

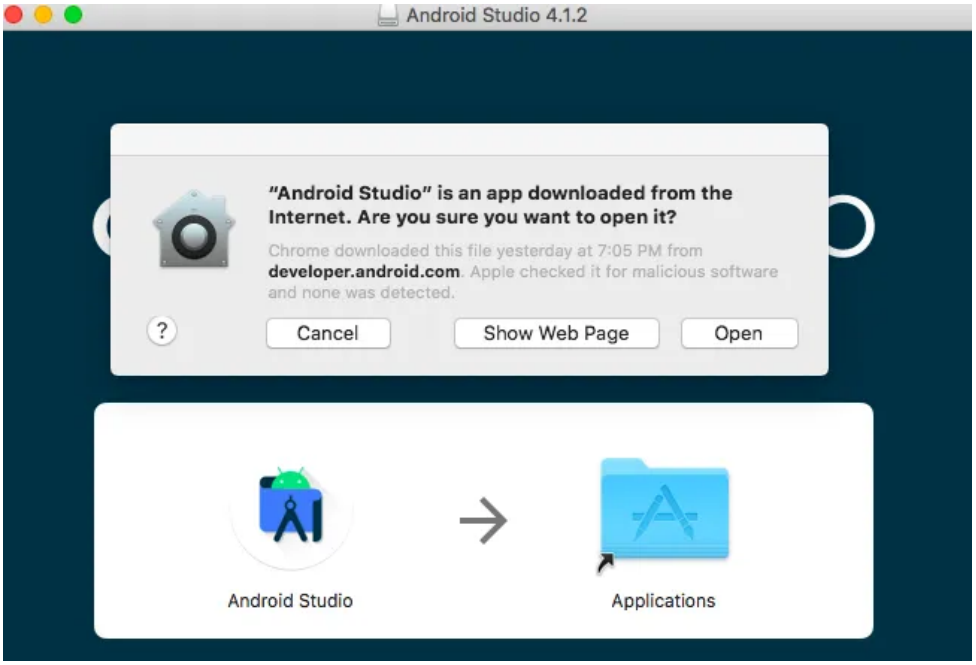


Figure 26 Step 2

- Then select the option to not import settings.

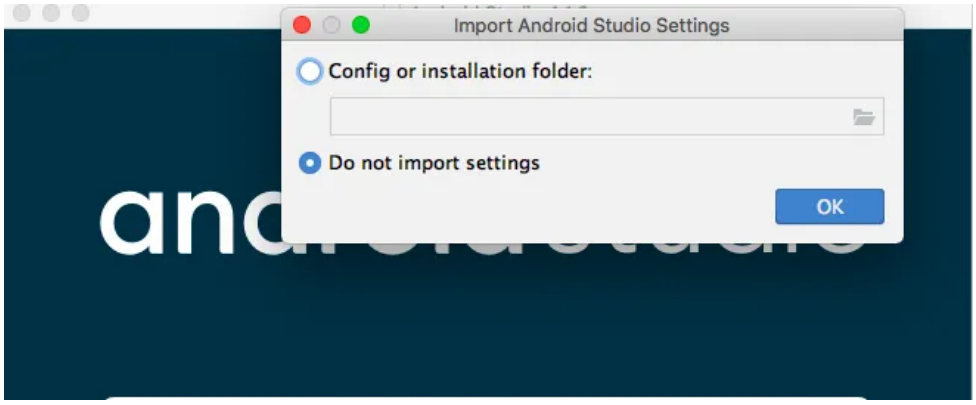


Figure 27 Step 3

- The configuration page will thereafter take some time to load. In the setup wizard, click Next.

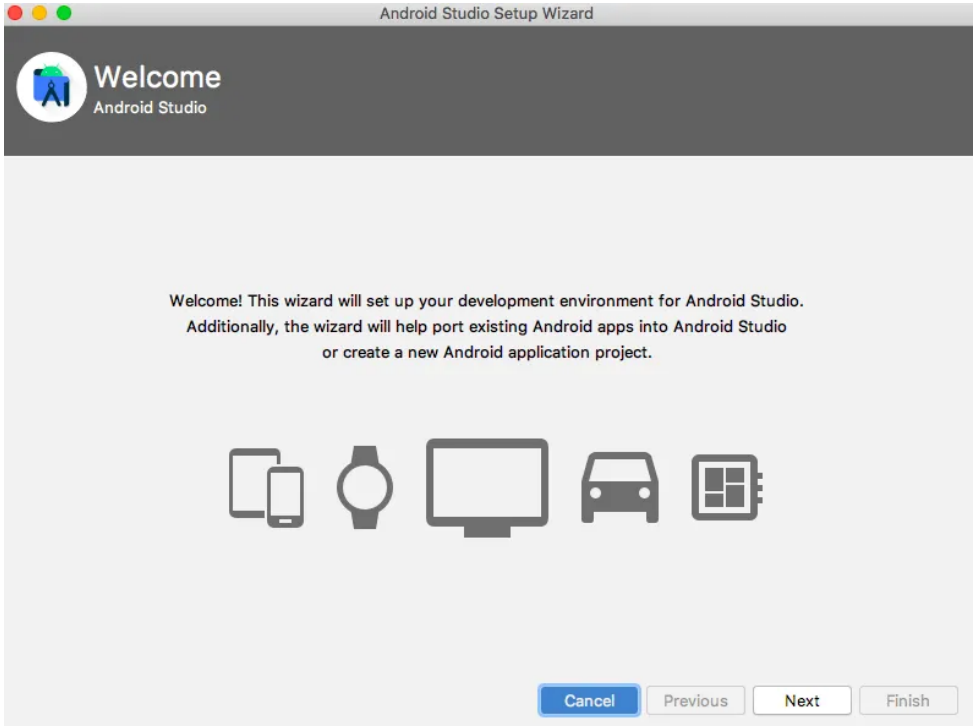


Figure 28 Step 4

- Select Standard and then press the Next button.

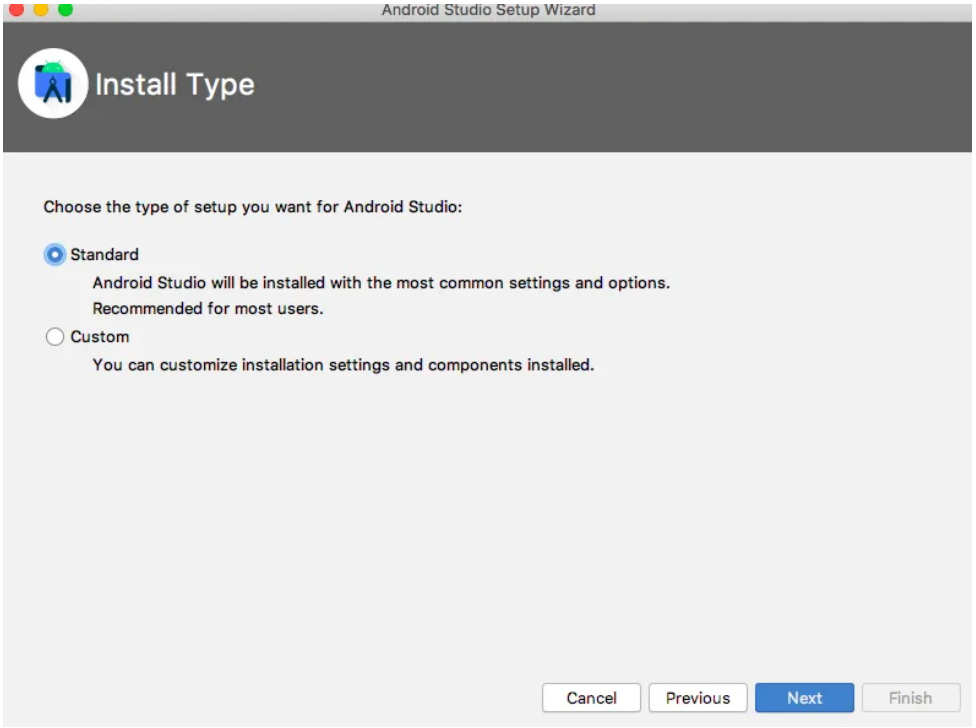


Figure 29 Step 5

- To finish the installation, click the Finish button. It will take some time depending on your internet connection speed. Because, during the last step of the installation, it will download some required files from the internet.

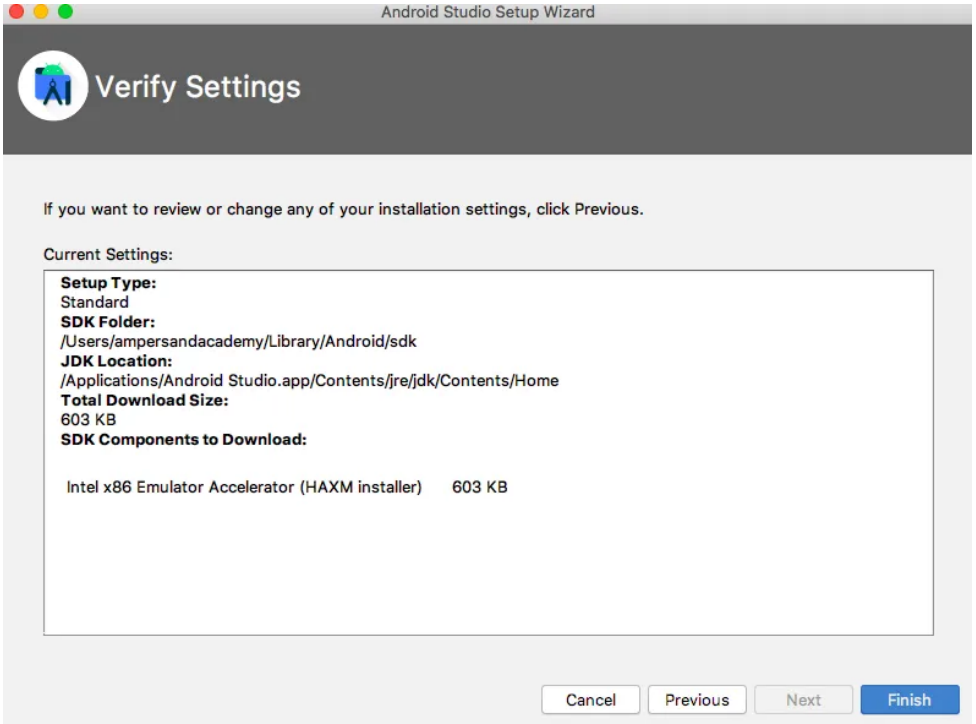


Figure 30 Step 6

4.3 Creating a New Android Project

(1) Once the installation is completed, you can see a screen like below, or if you open the Android Studio from the Launchpad. Click on **Create New Project** [30]

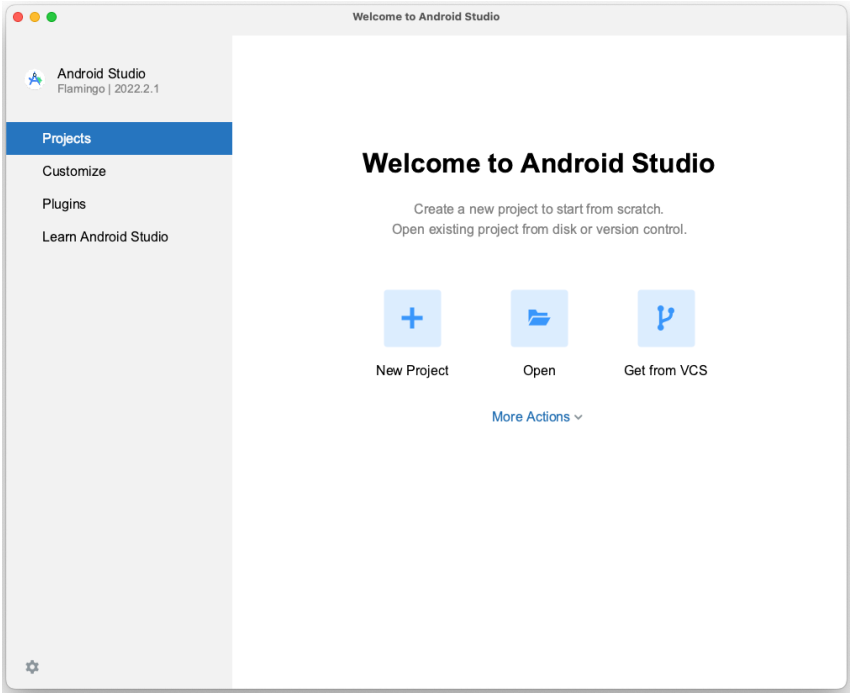


Figure 31 Step 1

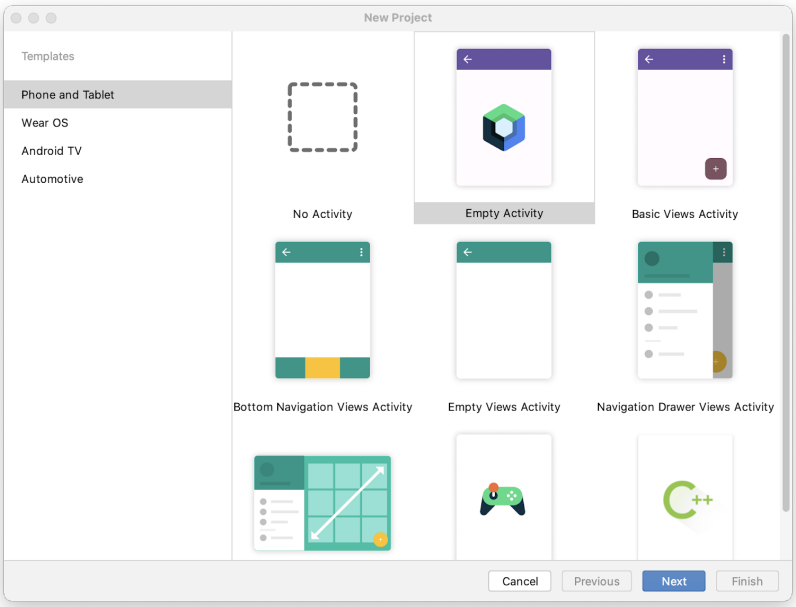


Figure 32 Step 2

The New Project window opens with a list of templates provided by Android Studio. In Android Studio, a project template is an Android project that provides the blueprint for a specific type of app. Templates create the structure of the project and the files needed for Android Studio to build your project. The template that you choose provides a starter code to get you going faster.

(2) Make sure the Phone and Tablet tab is selected.

(3) Click the Empty Activity template to select it as the template for your project. The Empty Activity template is the template to create a simple project that you can use to build a Compose app. It has a single screen and displays the text "Hello Android!".

(4) Click Next. The New Project dialog opens. This has some fields to configure your project.

(5) Configure your project as follows:
The Name field is used to enter the name of your project, for this codelab type "Precision Agriculture". Leave the Package name field as is. This is how your files will be organized in the file structure. In this case, the package name will be com.example.greetingcard. Leave the Save location field as is. It contains the location where all the files related to your project are saved.

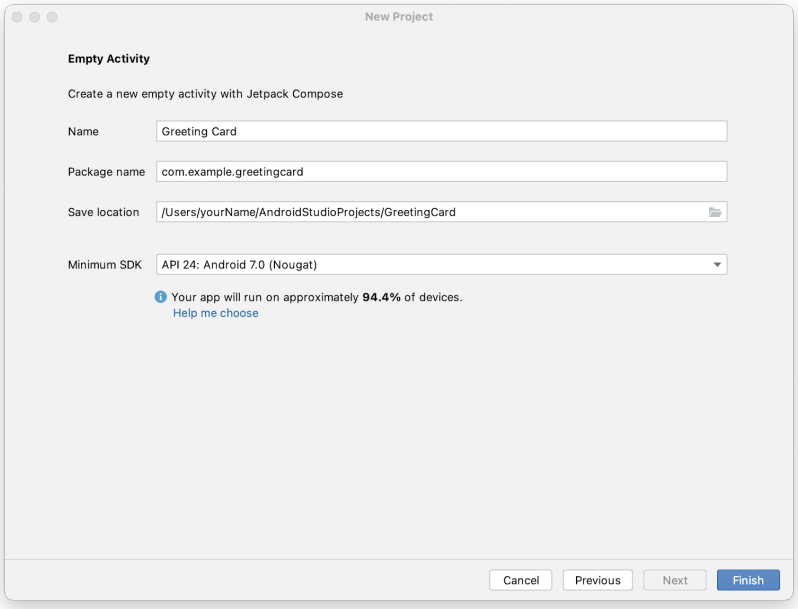


Figure 33 Step 3

Take note of where that is on your computer so that you can find your files. Select API 24: Android 7.0 (Nougat) from the menu in the Minimum SDK

field. Minimum SDK indicates the minimum version of Android that your app can run on.

(6) Click Finish. This may take a while - this is a great time to get a cup of tea! While Android Studio is setting up, a progress bar and message indicate whether Android Studio is still setting up your project.

In the MainActivity class, we declared variables for the ImageView, TextView, Bitmap, and Button UI elements used. To process the photos, we used the **openCV 4.6.0** library **tflite**. By incorporating the **CNN** tflite model tflite into the application and applying the same data preparation in model training before providing it as a tensor shape to the CNN model.

The model's size exceeded $200MB$ after converting it to tflite version. We trained the model again with the quantization conversion technique to handle these limitations.

5 Deep Learning

5.1 Artificial Intelligence

Computer science includes the field of artificial intelligence. When pioneers in the field of computer science began to wonder whether it was possible to make computers "think," the phenomenon first emerged. We're continuing to delve into this question today. Understanding some related concepts, like intellect, is necessary to comprehend AI.

The definition of artificial intelligence is

> "The science of making machines do things that require intelligence if done by men." (MIT's Marvin Minsky)

It is a group of computing technologies that draw their inspiration from the way neurons in the human brain and body sense, absorb information, and behave. It can also be defined as a system that can learn new concepts and tasks, a system that can reason and come to useful conclusions about the world, a system that can understand different natural languages, a system that can perceive and comprehend a visual scene, and a system that can carry out other tasks requiring human intelligence.
Artificial intelligence (AI) can generally be divided into three categories:

(1) **Artificial Narrow Intelligence (ANI):** Also referred to as weak AI, ANI is made to carry out a single task or a group of related tasks. Examples of this type of AI are language translation, image recognition, and recommendation systems. It is the most prevalent type of AI now in use. ANI systems are not able to learn or generalize outside of their unique mission.

(2) **Artificial General Intelligence (AGI):** AGI, sometimes referred to as strong AI or human-level AI is programmed to carry out any cognitive task that a human is capable of. AGI systems are intended to be able to learn and reason across a variety of areas and to generalize their knowledge across domains. AGI does not automatically imply consciousness or self-awareness, though.

(3) **Artificial Superintelligence (ASI):** is a hypothetical type of AI that is capable of outpacing human intelligence in every way possible. ASI is also referred to as singularity. In addition to being able to recursively self-improve,

ASI would be able to solve issues that are currently beyond the capabilities of humans and would see an exponential rise in intelligence. There is disagreement over whether ASI is feasible or even desirable, hence it is merely hypothetical currently.

5.2 Applications of AI

(1) **Healthcare:** Several applications of AI are being utilized to enhance patient care and results, including the identification of early disease symptoms, the creation of individualized treatment regimens, and the analysis of medical imagery.

(2) **Finance:** Fraud detection, risk management, and investment analysis have all benefited from the usage of AI. For instance, AI-driven chatbots are utilized to give clients individualized financial advice.

(3) **Transportation:** Autonomous vehicle development, traffic flow optimization, and supply chain management are all made possible by AI.

(4) **Manufacturing:** AI is used to increase productivity and efficiency in manufacturing operations including supply chain optimization, quality control, and predictive maintenance.

(5) **Education:** AI is being utilized to customize learning opportunities and raise student performance. AI-powered tutoring systems, for instance, are being used to deliver adaptive learning experiences catered to specific student needs.

(6) **Virtual reality and augmented reality** experiences are two examples of new forms of entertainment that AI is being utilized to create. AI is also used to customize content recommendations, enhance search, and facilitate discovery.

AI is having a big impact on society and changing how we work and live. Here are a few examples of how AI is influencing society:

(1) AI's potential to eliminate jobs is one of the main worries. There is a chance that many employments will become mechanized as robots become more adept at carrying out tasks that humans once completed. Certain industries, especially those that rely heavily on repetitive or routine tasks, could experience significant job losses because of this.

(2) To collaborate with and create AI technology, new skills and training will be needed as AI continues to develop. This indicates that there will be an increase in the need for individuals who possess expertise in fields like data science, machine learning, and natural language processing.

(3) Privacy and Security: Because AI mainly relies on data, issues with data privacy and security are becoming more crucial. Personal information runs the risk of being used for unethical objectives like discrimination or spying.

(4) Fairness and Bias: The data that AI systems are taught determines how unbiased they are. AI systems run the potential of sustaining prejudice and discrimination, especially if the data used to train them is biased or deficient.

5.3 Machine Learning

Machine learning has emerged as a revolutionary field within the realm of artificial intelligence (AI). With its ability to analyze vast amounts of data and uncover patterns, machine learning has become an invaluable tool across various domains. Its applications, and its potential implications for the future.
Machine learning refers to the process by which computer systems are programmed to learn from data and make predictions or decisions without explicit human intervention. It is a subset of AI that focuses on developing algorithms capable of analyzing data, recognizing patterns, and making informed decisions or predictions based on that data. As articulated by Arthur Samuel in 1959, machine learning is about

> “Giving computers the ability to learn without being explicitly programmed"
> (Samuel, 1959)

There are several types of machine learning algorithms, each designed to tackle specific problems and extract meaningful insights:

(1) **Supervised Learning:** A model is trained using labeled data through supervised learning, where the input features are linked to corresponding output labels. The computer gains knowledge from this labeled data to accurately forecast or categorize brand-new, untainted data. Decision trees, support vector machines (SVM), and neural networks are supervised learning techniques.

(2) **Unsupervised Learning:** Unsupervised learning works with unlabeled data with the intention of revealing underlying patterns or structures. Without using labels, the program finds clusters, correlations, or anomalies. Clustering

algorithms like k-means and hierarchical clustering, as well as dimensionality reduction techniques like principal component analysis (PCA), are common unsupervised learning techniques.

(3) **Reinforcement Learning:** Reinforcement learning includes teaching an agent to interact with the environment and discover the best course of action through making mistakes. Reward or punishment feedback is sent to the agent, influencing its decision-making. Numerous fields, including robotics, autonomous systems, and game playing, have effectively used reinforcement learning. In their thorough book on reinforcement learning published in 2018, Sutton and Barto discuss algorithms like policy gradients and Q-learning.

(4) **Semi-Supervised Learning:** Semi-supervised learning combines elements of both supervised and unsupervised learning. It leverages a small amount of labeled data along with a larger set of unlabeled data to improve model performance. This approach is particularly useful when labeling large datasets becomes time-consuming or expensive.

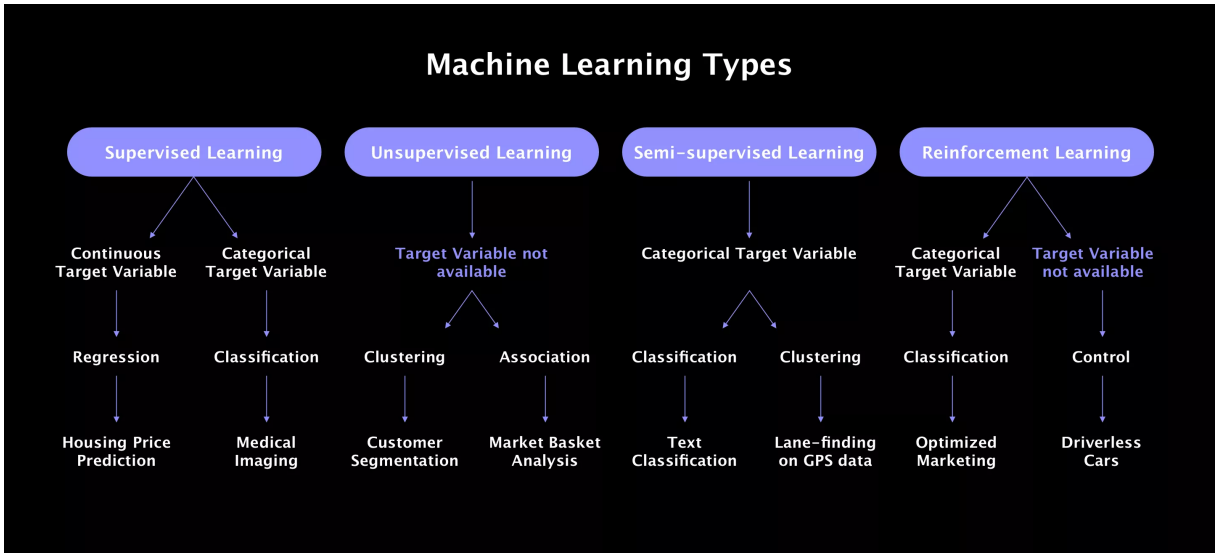


Figure 34 Machine Learning Types

5.4 Deep Learning

Through the use of neural networks, deep learning, a subset of machine learning, has allowed computers to learn intricate patterns and representations, revolutionizing the area of artificial intelligence. There are different types of deep learning architectures and methods that have revolutionized numerous fields.

(1) **Convolutional Neural Networks (CNNs):** Convolutional Neural Networks, or CNNs, have revolutionized image and video processing tasks. CNNs are specifically designed to analyze visual data by using layers such as convolutional and pooling layers. These architectures excel at tasks such as image classification, object detection, and image segmentation.

(2) **Recurrent Neural Networks (RNNs):** Recurrent Neural Networks, or RNNs, are designed to process sequential data, where the order of inputs matters. RNNs maintain an internal memory to process past information, making them suitable for tasks such as natural language processing, speech recognition, and language translation.

(3) **Generative Adversarial Networks (GANs):** Generative Adversarial Networks, or GANs, are composed of two neural networks: a generator and a discriminator. GANs excel at generating realistic synthetic data by learning from existing data distributions. They have been widely used for image synthesis, text generation, and video generation.

(4) **Autoencoders:** Autoencoders are neural network architectures used for unsupervised learning and dimensionality reduction. They consist of an encoder that compresses input data into a low-dimensional representation and a decoder that reconstructs the original input from the compressed representation. Autoencoders have applications in image denoising, anomaly detection, and feature learning.

(5) **Reinforcement Learning with Deep Neural Networks:** Combining deep learning with reinforcement learning has resulted in significant advancements in areas such as game playing and robotics. Deep reinforcement learning models use neural networks to approximate value functions or policies, enabling agents to make optimal decisions in dynamic environments.

5.5 Computer Vision

Computer vision is a field of study and technology that focuses on enabling computers to gain an understanding of digital images or videos in a way similar to how humans perceive and interpret visual information. It involves the development of algorithms and techniques to extract meaningful information from visual data, allowing machines to analyze, recognize, and make decisions based on visual inputs.

Computer vision relies on a wide range of techniques and tools, such as deep learning, artificial neural networks, image analysis, image segmentation, feature detection,

motion tracking, and more. These techniques play a crucial role in enabling computers to recognize shapes, analyze them, and extract information. One of the key goals of computer vision is to enable machines to interpret and understand the content of images or videos, similar to how humans can recognize objects, scenes, and patterns. This involves tasks such as image classification, object detection, image segmentation, and scene understanding. Computer vision algorithms can be trained using machine learning techniques, such as deep learning, which require large amounts of labeled data to learn and generalize from.

Applications of computer Vision are so highly diffused

(1) **Autonomous vehicles:** Computer vision plays a crucial role in enabling self-driving cars to perceive their surroundings and make real-time decisions based on the detected objects, lanes, traffic signs, and pedestrians.

(2) **Surveillance and security:** Computer vision systems are used for video surveillance, monitoring public spaces, detecting anomalies, and identifying objects or individuals of interest.

(3) **Medical imaging:** Computer vision algorithms are utilized in medical imaging technologies such as X-ray, MRI, and CT scans to assist in the diagnosis and analysis of diseases, identifying tumors, and assisting in surgical procedures.

(4) **Augmented reality (AR) and virtual reality (VR):** Computer vision enables the overlaying of digital information or virtual objects onto the real world, creating immersive AR and VR experiences.

(5) **Robotics:** Computer vision allows robots to perceive and interact with their environment, enabling tasks such as object recognition, grasping and manipulation, and navigation.

Image filtering, feature extraction, pattern recognition, and machine learning algorithms are a few examples of computer vision approaches that rely on different mathematical and statistical models. Python and other computer languages as well as libraries like OpenCV, TensorFlow, and PyTorch are frequently used to construct these methods.

5.6 Plant detection and Classification

5.6.1 Data description

This dataset is recreated using offline augmentation from the original dataset. This dataset consists of about **87K rgb** images of healthy and diseased crop leaves which are categorized into **38** different classes [31]. The total dataset is divided into **80/20** ratio of training and validation set preserving the directory structure. A new directory containing **33** test images is created later for prediction purposes.
Sources: From Kaggle - From GitHub

Figure 35 Sample of images in the training dataset

5.6.2 Methodology

CNN Model:

CNN is a type of neural network model which allows us to extract higher representations for the image content. Unlike classical image recognition where you define the image features yourself, CNN takes the images raw pixel data, trains the model, then extracts the features automatically for better classification.

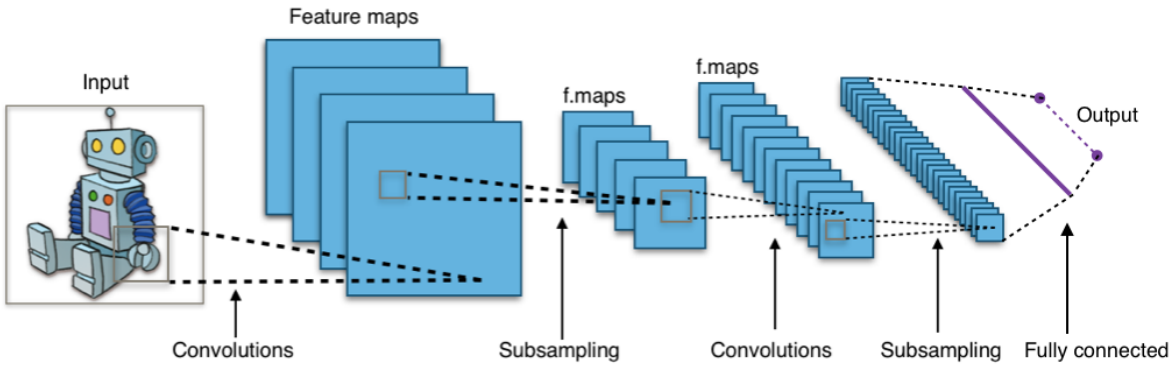


Figure 36 CNN Architecture

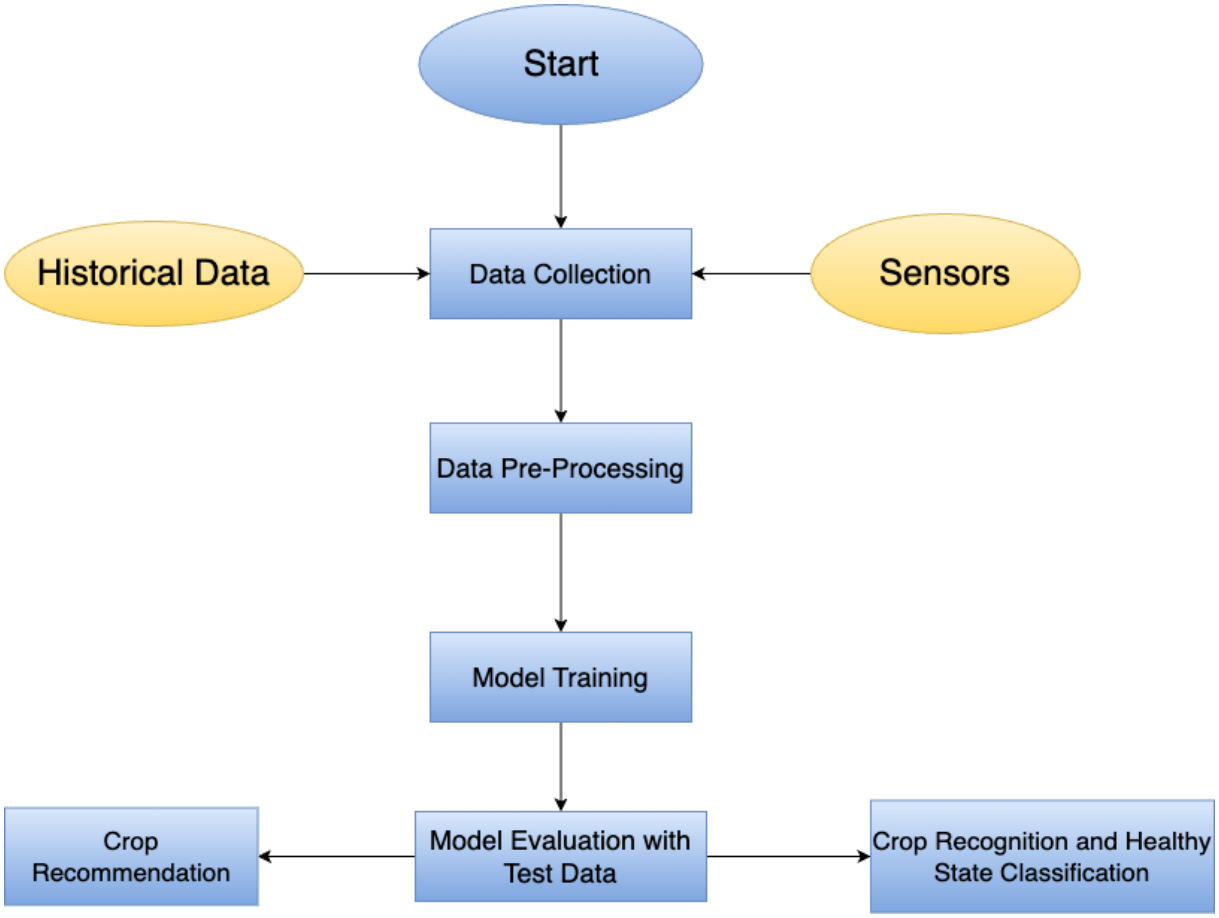


Figure 37 Block diagram for proposed model sequence and controlling the sent variables

Blocks of code:

- **Data Preparing:** After clustering the data set we have **70295** for training and **17572** for validation images and **38** classes.

- **Data Preprocessing:** Using **(256, 256)** as image size in our data and splitting it into train, validation, and test data. After that, we had data augmentation to increase the diversity of our training set.

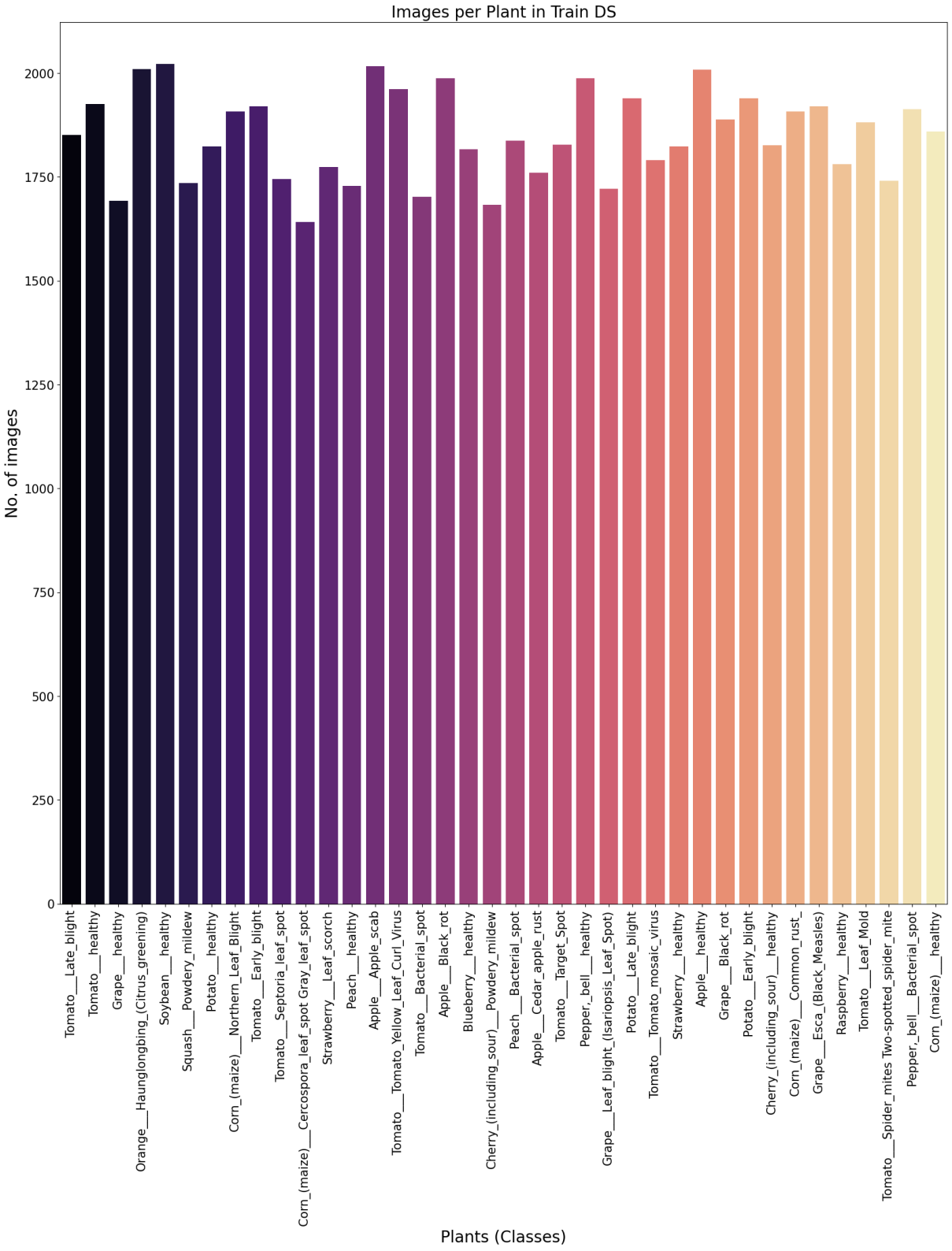


Figure 38 Bar plot for the number of images for each plant in the training dataset

The validation set was a random image from the training images and the test images were split from the full dataset with the same image shape size after the preprocessing.

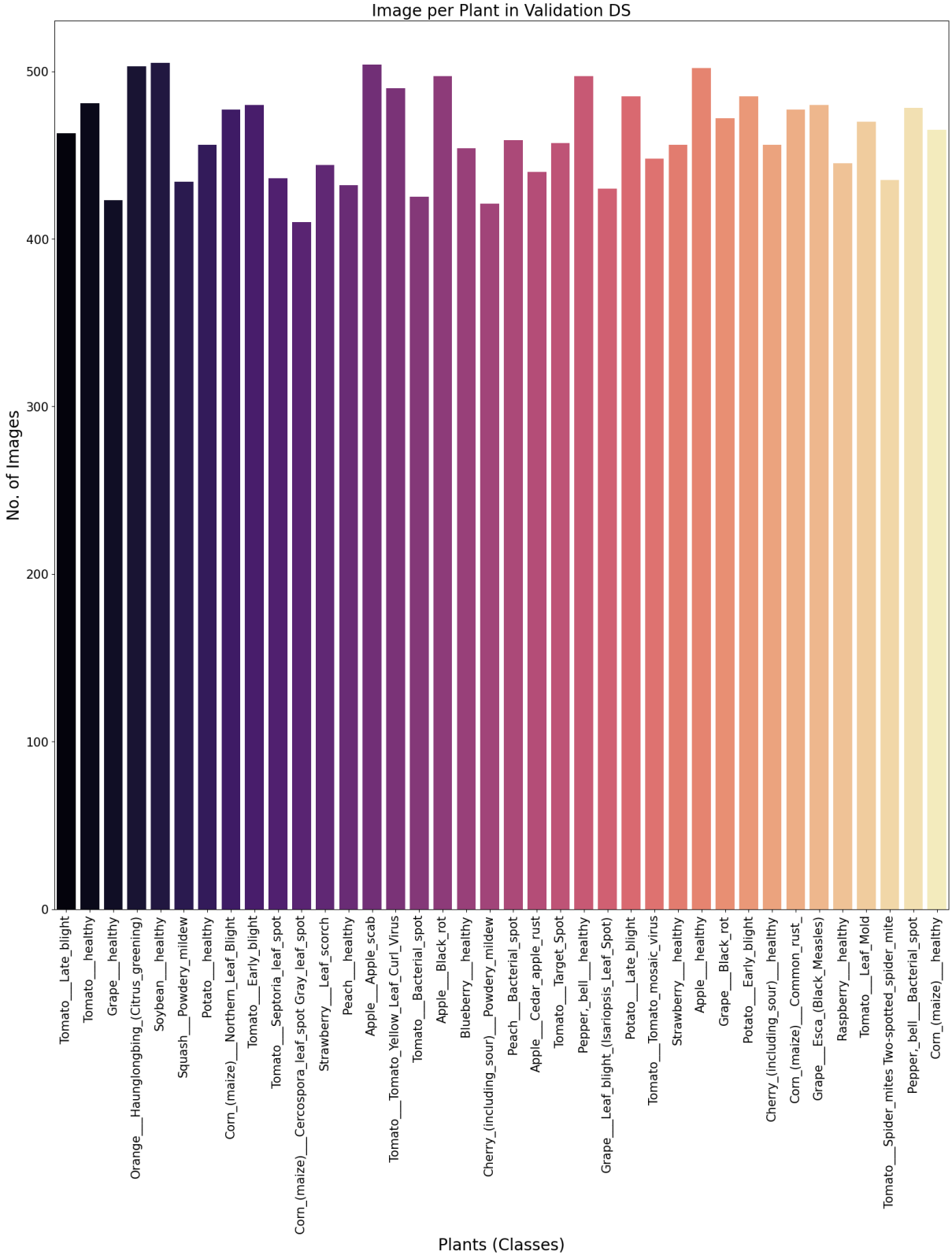


Figure 39 Bar plot for the number of images for each plant in the validation dataset

The proposed model training:

After training a CNN Model with input shape **(256, 256, 3)**, RELU activation function in the hidden layers, a Softmax activation function in the output Dense layer to classify the output to the **38** classes.
By using **32** filters in the input layer and by increasing this number through the layers until we get **1024** filters.With the combination with Adam optimizer, sparse categorical cross-entropy loss and a **0.0001** learning rate value and **20** epochs training.

Total Parameters Accuracy	**53,510,982%**
Trainable Parameters	**53,510,982%**
Non-Trainable Parameters	**0%**

Table 3 Parameters counter in the proposed model

Because of the model size, we retrained it again using the post-training quantization technique [32]. Post-training quantization offers several benefits in machine learning applications:

(1) **Reduced Memory Footprint:** Quantizing the model reduces the memory requirements compared to the original floating-point representation. This is especially beneficial for deploying models on resource-constrained devices with limited memory capacity, such as mobile phones, IoT devices, and edge devices.

(2) **Lower Computational Requirements:** Quantized models perform computations using lower-precision fixed-point operations, which typically require fewer computational resources than floating-point operations. This results in faster inference times and reduced energy consumption, making it more efficient for real-time and power-constrained applications.

(3) **Improved Model Deployment:** Post-training quantization enables the deployment of larger and more complex models on devices with limited resources. By reducing the model size and computational requirements, it becomes feasible to deploy advanced machine learning models in edge computing scenarios where there are constraints on memory, processing power, and energy consumption.

(4) **Faster Inference:** Quantized models can often be executed more quickly than their floating-point counterparts due to the reduced precision and smaller memory footprint. This is particularly important in latency-sensitive applica-

tions where fast response times are crucial, such as real-time object detection, speech recognition, and autonomous driving.

(5) **Compatibility with Hardware Acceleration:** Many hardware accelerators, such as specialized chips or neural network processors, are optimized for lower-precision operations. By quantizing the model, it becomes compatible with these accelerators, allowing for even faster and more efficient inference.

(6) **Cost Reduction in Cloud Deployments:** Post-training quantization can also have cost benefits in cloud-based deployments. By reducing the model size and computational requirements, it decreases the amount of computation resources needed, resulting in lower infrastructure costs for running and serving the model.

(7) **Maintaining Competitive Accuracy:** While quantization introduces a loss of precision, modern techniques like quantization-aware training and fine-tuning help mitigate this accuracy drop. With careful optimization and fine-tuning, quantized models can often achieve competitive accuracy compared to their full-precision counterparts, making post-training quantization a viable option without sacrificing performance significantly.

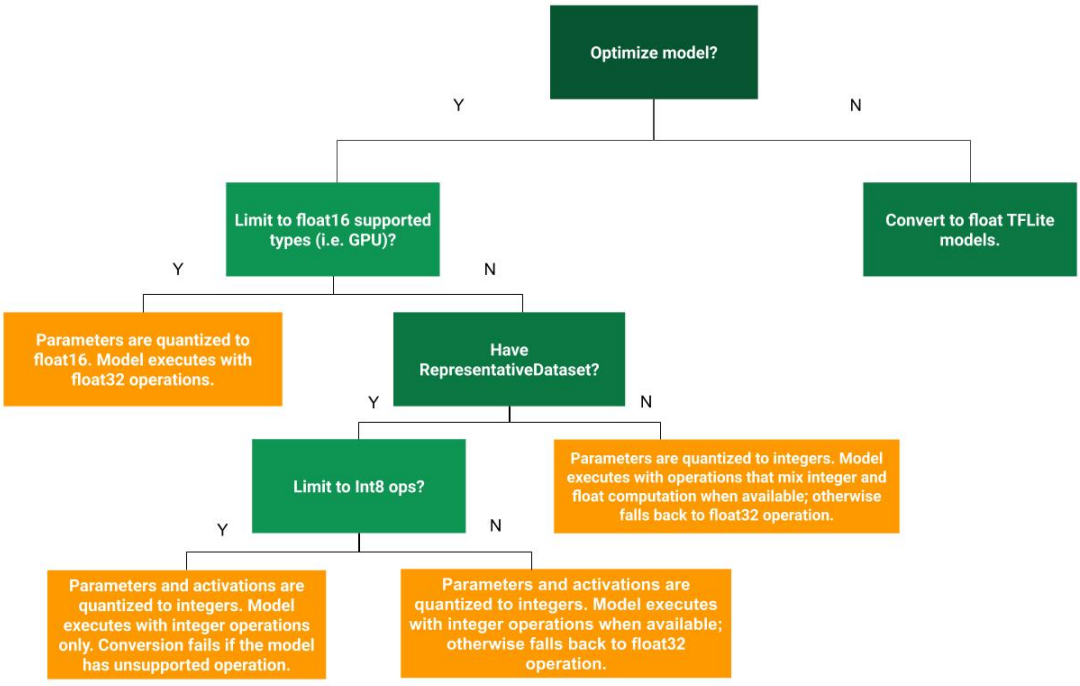


Figure 40 Post-training Quantization

5.7 Crop Recommendation System

5.7.1 Data description

This dataset was built by augmenting datasets of rainfall, climate and fertilizer data available for India.

Data fields:

N - ratio of Nitrogen content in soil
P - ratio of Phosphorous content in soil
K - ratio of Potassium content in soil
temperature - temperature in degrees Celsius
humidity - relative humidity in %
ph - ph value of the soil
rainfall - rainfall in mm

The dataset has **8** columns **7** features and a label that recommends the most suitable crops to grow in a particular farm based on various parameters as it's obvious in Fig.41. With **2200** records and **22** different types of crops we trained our model.

By applying the Exploratory Data Analysis [33] on the dataset we spotted the anomalies and discovered some patterns between the features. Fortunately, our dataset has no null values or unreal data records.

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

Figure 41 Sample of Crop Dataset

The mean values are almost converged in the first three features. Normalization won't has a great effect on the data as it's in a small range so the calculations to detect the

weights and bias in each different model won't be affected.
The plants in the dataset are grown in different environments, places, and seasons so the recorded data has a great varying to cover more than one plant it will be more useful if we have more records for these plants.

With a hundred records for each plant, we trained our models which led to underfitting in the model. All the features' values were numerical so we didn't need to use encoding except in the label which has the recommendation result in object type so it converted to float-64 type.

	N	P	K	temperature	humidity	ph	rainfall
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.463655
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.958389
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.211267
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.551686
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.867624
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.267508
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117

Figure 42 Crop dataset numerical decription

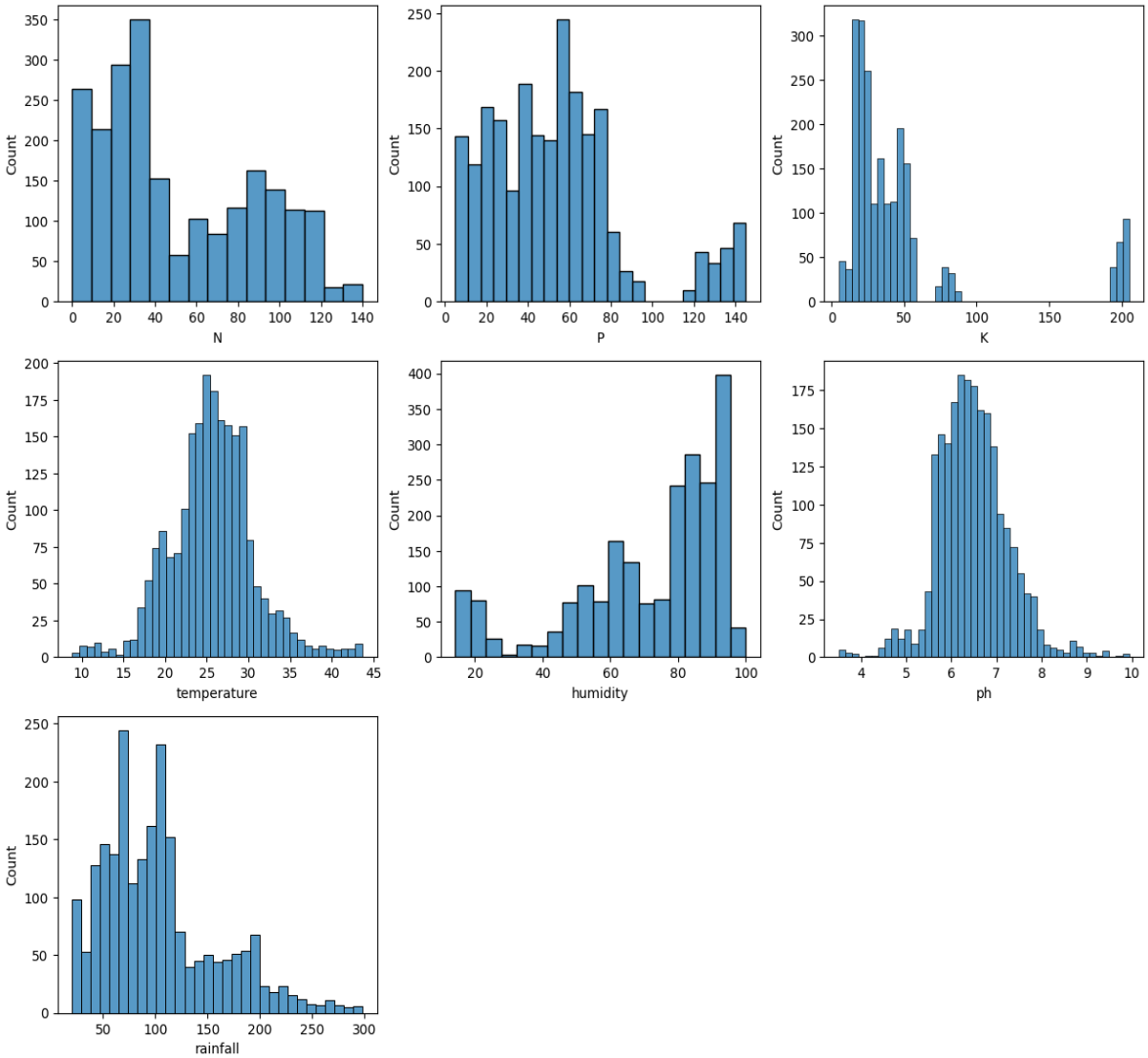


Figure 43 Quantity of each value in each feature

The data values have different distributions as shown in Fig.43, Temperature and PH almost have a Gaussian curve. K has a great gap in the range of 100 to 180 which means it has a lower effect on updating the weight and bias values in this range.

N feature has the most normally distributed data so it can be considered the main feature in our dataset. [34]

PH value also refers to the used water and the effect of using pesticides and fertilizers integrated with the water on plant growth.

The relation between each feature with other show how the dependencies on each feature and how can each feature benefits from other ones and what are the best

ranges we can mix to have the best recommendation. As it's shown in Fig.44 we considered the N feature as the basic feature in a **hue** view plot. This could help to generate new features in the future or cover the shortage in the dataset records. [1]

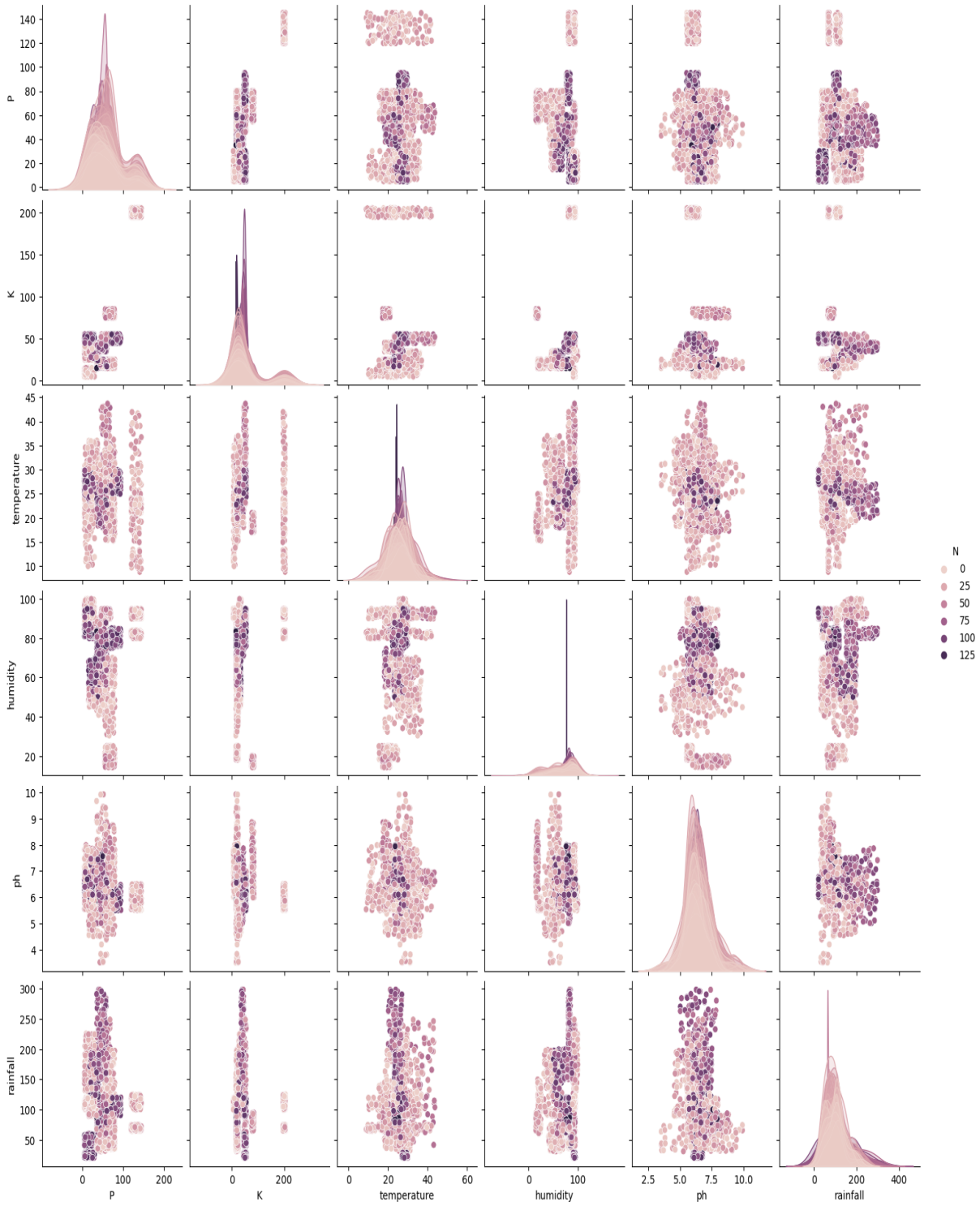


Figure 44 Relationship between features

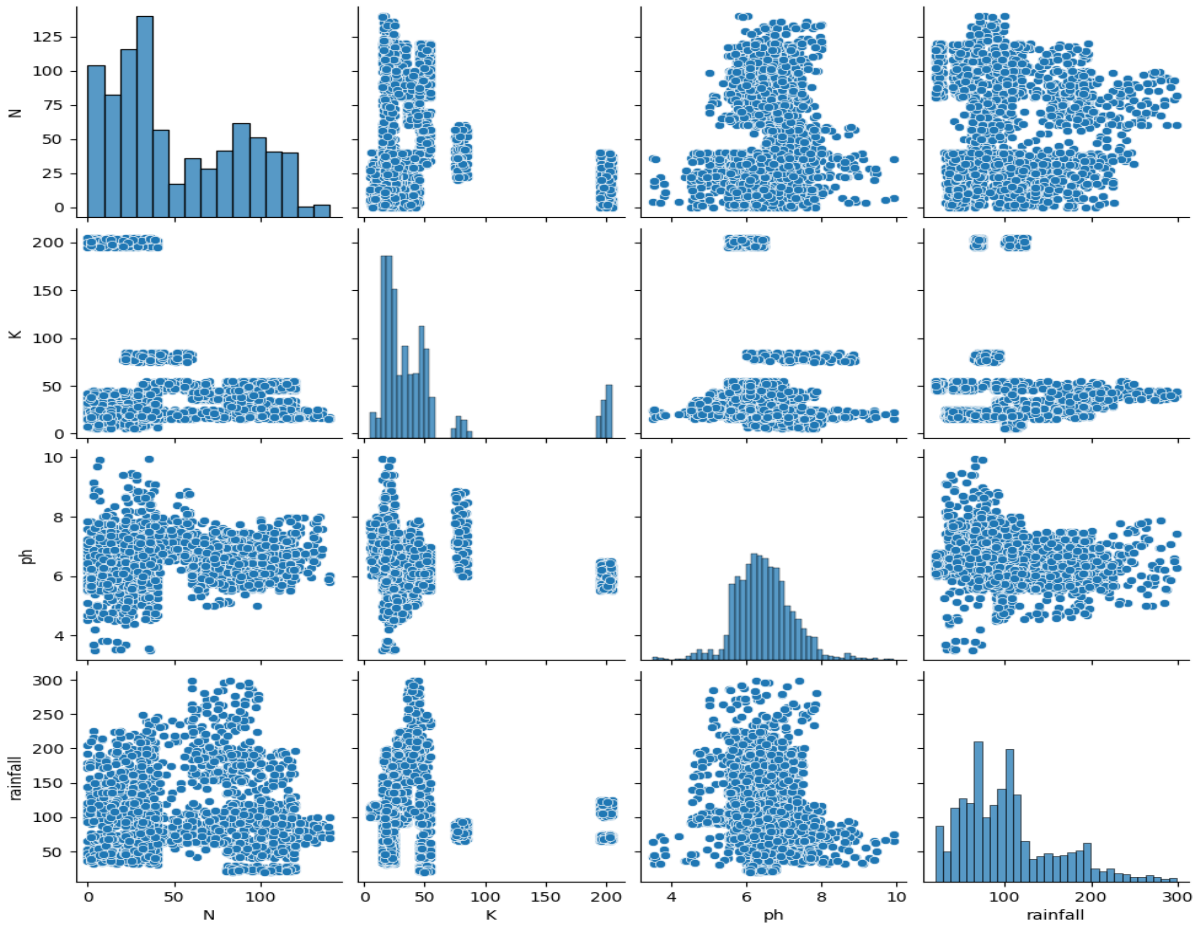


Figure 45 Relationship between dedicated feature

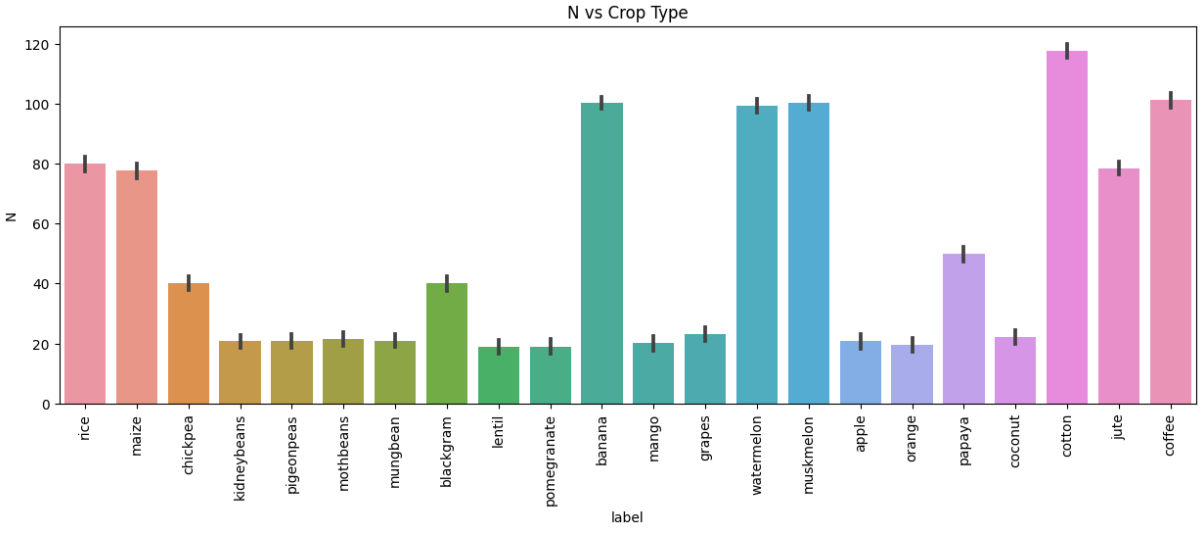


Figure 46 N vs Crop type

In the main view, each feature has an effect on the plant type in more than one way, in itself as it's shown in Fig.46, Fig.47 and Fig.48 also in the combination with other

feature as we illustrated it before adding to this the range of its values as if it has a big range value it will cover the effect of other features in calculating the bias and weights.

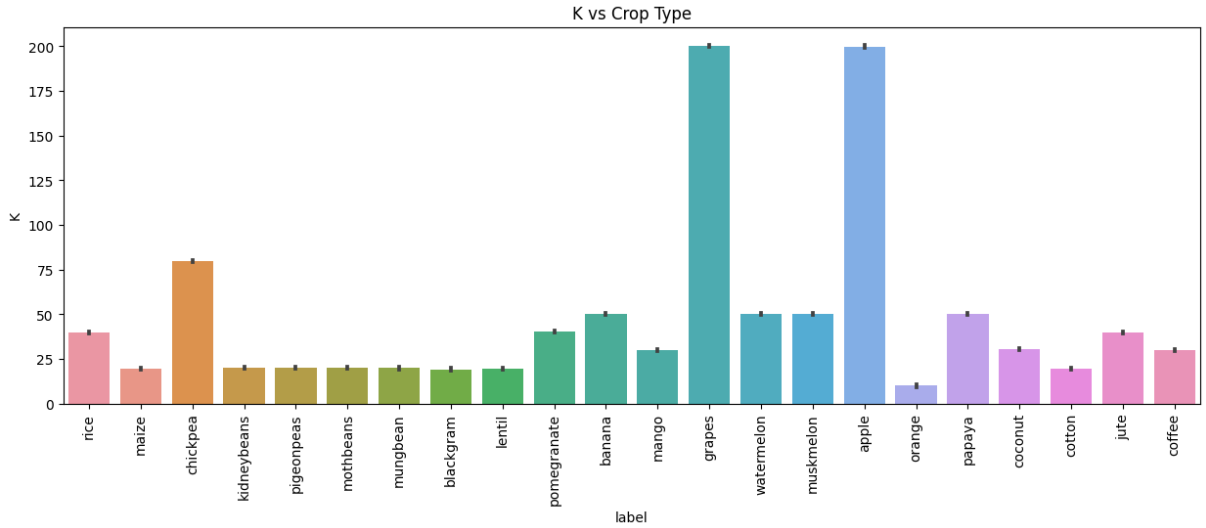


Figure 47 K vs Crop type

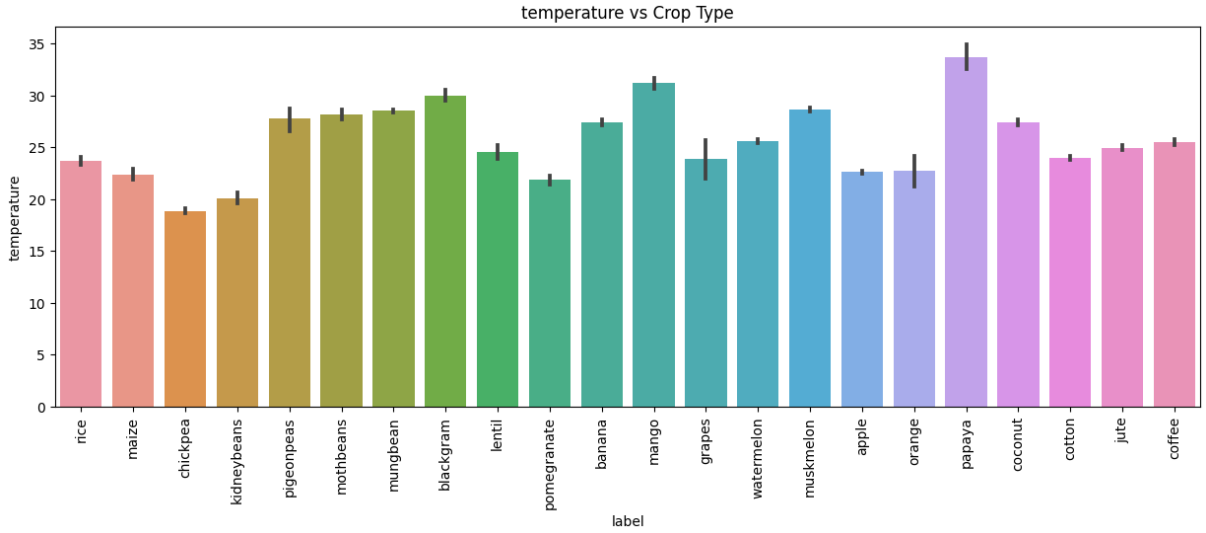


Figure 48 Temperature vs Crop type

6 Results

6.1 Embedded System

(1) **Test 1:** As we see in Fig.49, we test some readings as you see in the upper figure, Temperature is $25℃$, soil moisture value is 505 and water level value is 950. According to these readings, the motor will be off because the temperature is lower than 30, soil moisture is higher than 100 and the water level is higher than 300.

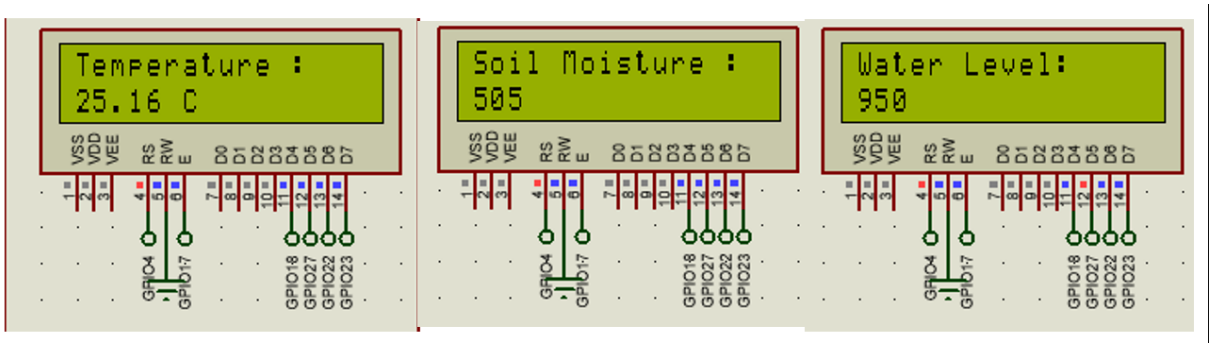
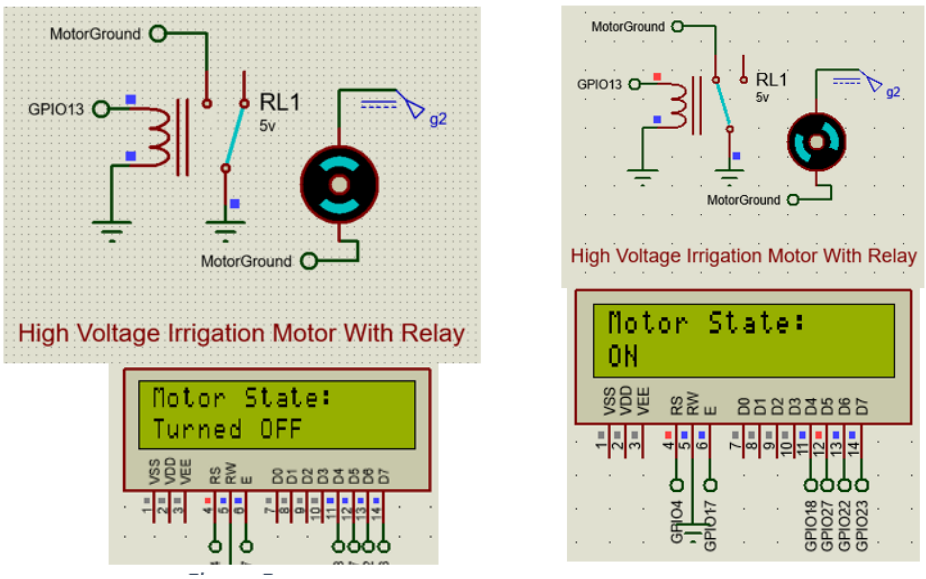


Figure 49 Test1 results



(a) Test1 Motor State (b) Test2 Motor State

Figure 50 Motor's States Results

(2) **Test 2:** In Fig.51 we will test another reading as you see. Temperature is $35℃$, soil moisture value is 47 and water level value is 38. According to these readings, the motor will be turned on because the temperature is higher than 30, soil moisture is lower than 100 and the water level is lower than 300.

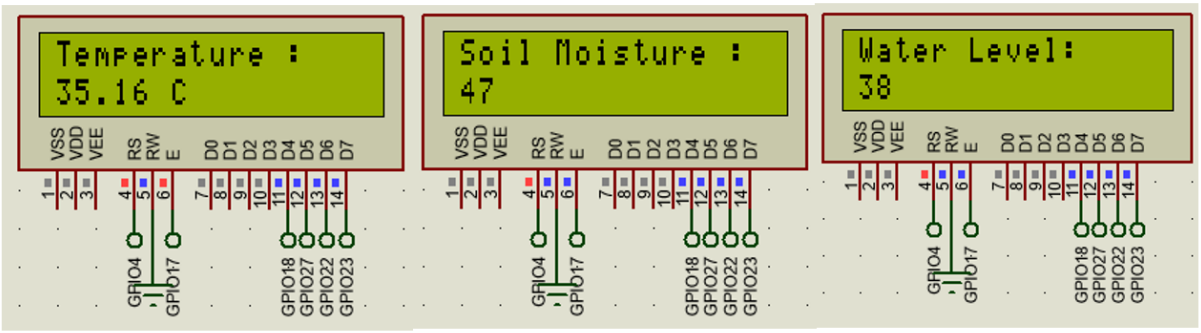
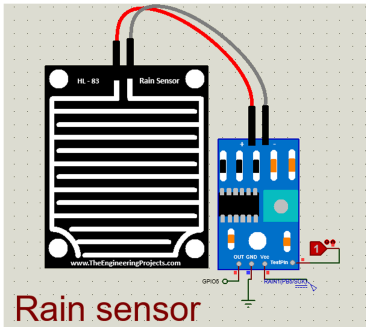
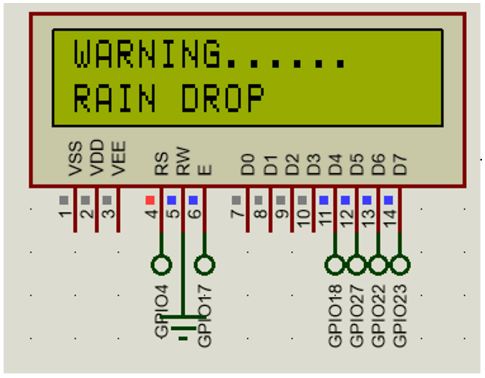


Figure 51 Test2 results

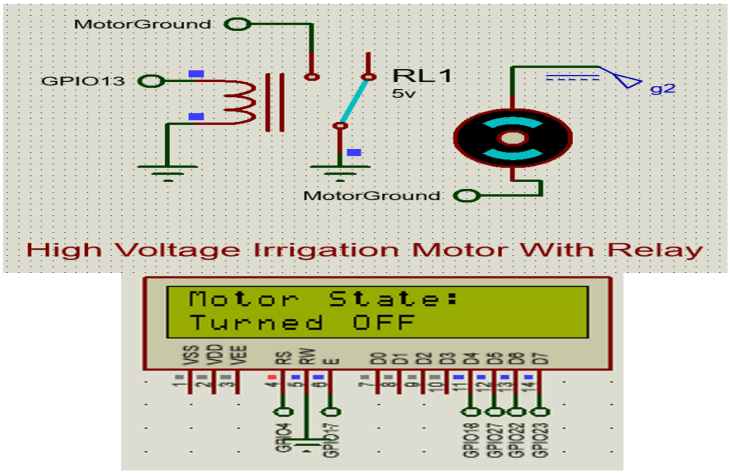
(3) **Test 3:** In this test, we use the same readings in the previous test but this time we simulate that there is a raindrop, so the raindrop sensor will display a warning message on the display screen as you see in the figure then it will ignore all readings and turn off the motor.



(a) Rain sensor have 1 logic state on test pen to simulate that there is rain



(b) Rain sensor send rain drop warning message



(c) Motor turned off

Figure 52 Test3 Results

6.2 Mobile Development (Android)

6.2.1 Results:

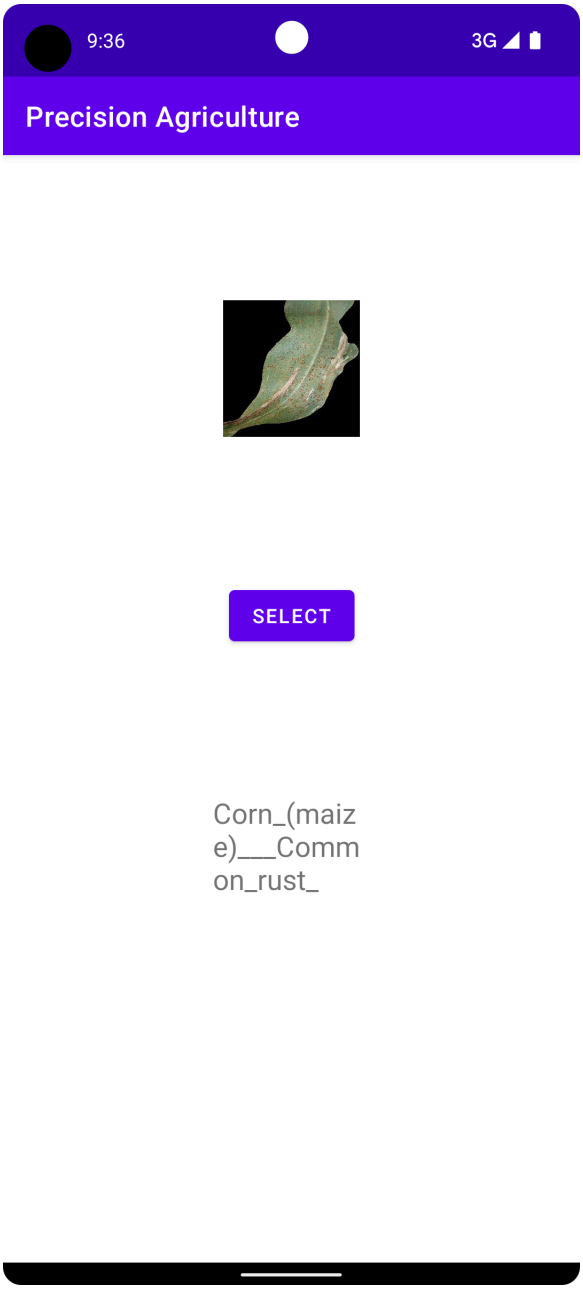


Figure 53 Mobile Application prediction sample

6.3 Deep Learning

6.3.1 Plant detection and classification

Results:

After this training, we get **99.15%** as a training accuracy, **97.75%** as a test accuracy, **0.0279** as a training loss, and **97.75** as a Sensitivity and Precision.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{6.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{6.2}$$

$$Sensitivity(Recall) = \frac{TP}{TP + FN} \tag{6.3}$$

Train Accuracy	**99.15%**
Test Accuracy	**97.75%**
Precision Score	**97.75%**
Recall Score	**97.75%**

Table 4 CNN-Model accuracy values

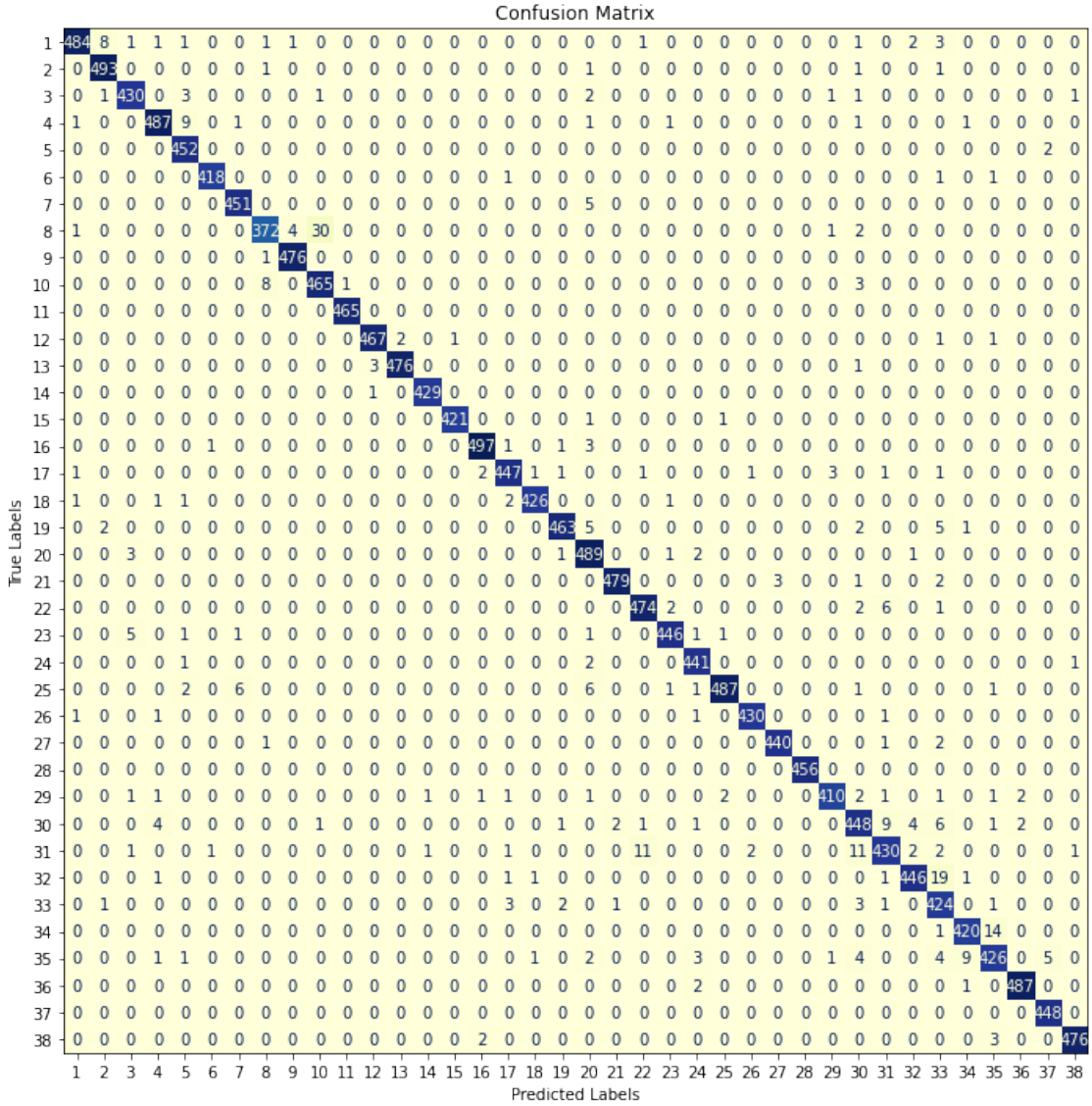


Figure 54 Confusion Matrix

With 10 epochs in addition to the preprocessing, they were enough to get good accuracy and the values of the true positive and true negative in the correlation matrix at Fig.54 were satisfied as the wasnt great misclassification in the predictions.

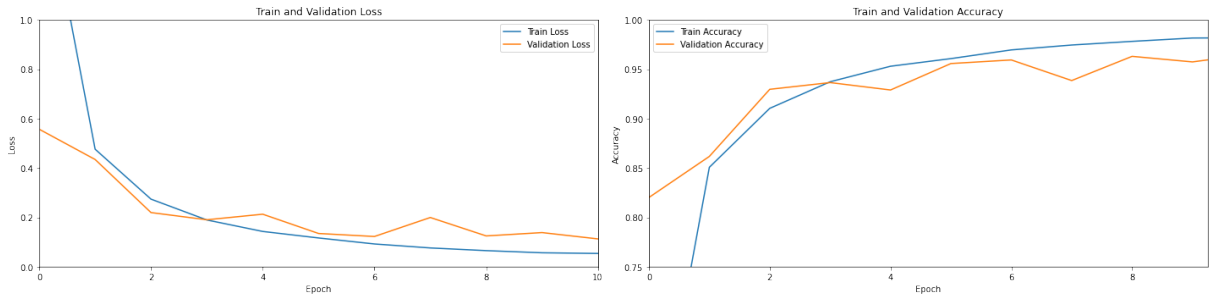


Figure 55 Plot of Loss and Accuracy values

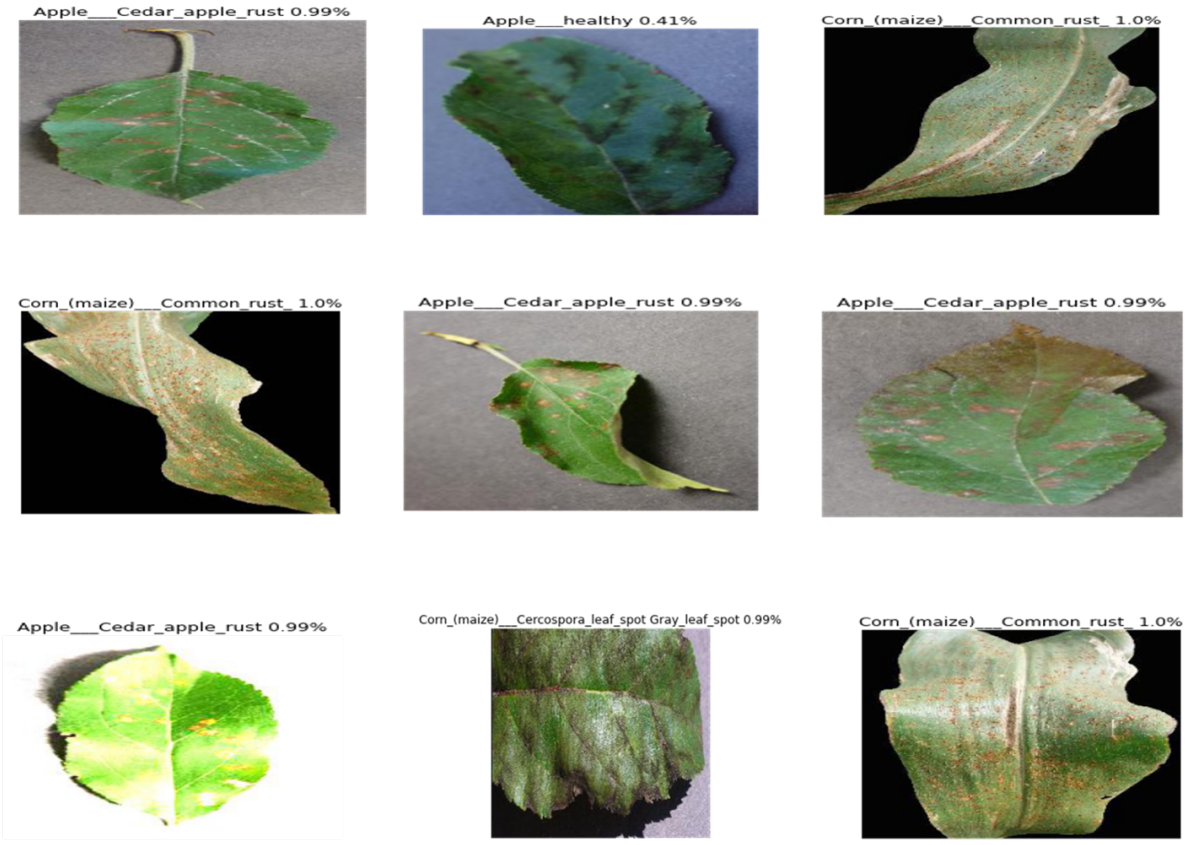


Figure 56 Samples of the proposed model predictions

As its evident from Fig.56, in each prediction we get the predicted value of the classification of the plant with its disease if it has a percentage value of the assurance of the forecast.

6.3.2 Crop Recommendation System:

Results:

The models need more data, the available data we have hasn't enough records to train the models.
The five different models give converged results. The best model was **Naive Bayes** with **99.55%** accuracy

Model	Accuracy
Decision Tree	**90.68%**
Naive Bayes	**99.55%**
SVM	**97.72%**
Logistic Regression	**95%**
RF	**99.09%**

Table 5 Models accuracies values

The different models were saved in pickle files to integrate them into web and mobile applications.

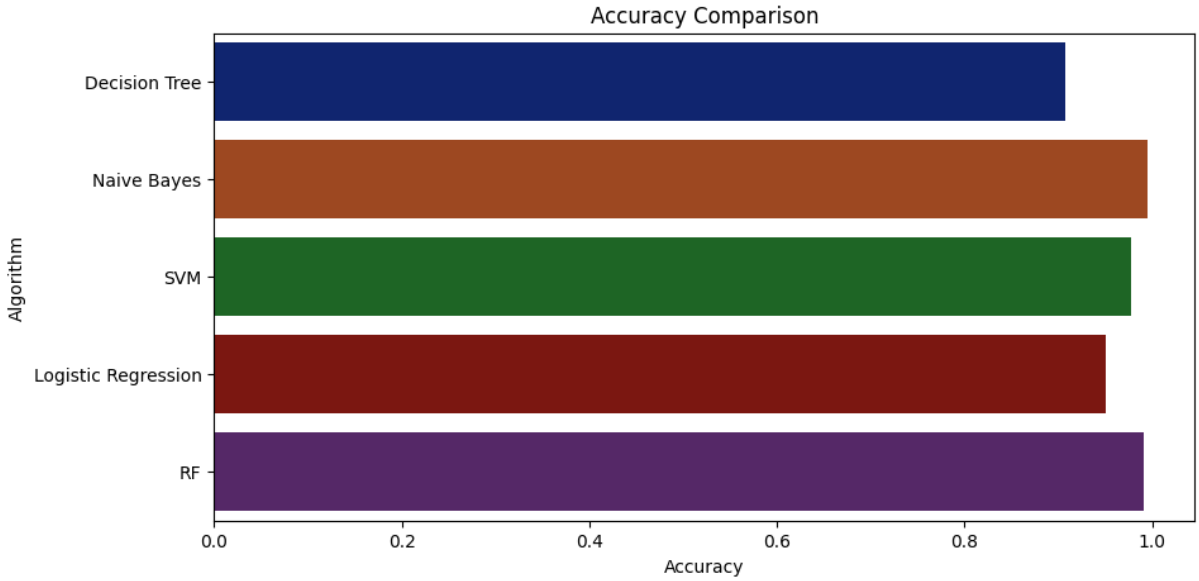


Figure 57 Different Models Accuracies Plot

7 Conclusion and Future work

Precision agriculture is currently having a significant impact on crop selection and cultivation. This proposed model exhibits a high level of accuracy and precision while requiring minimal prediction time. Gathering additional data on new crops, particularly within our region and continent, has the potential to extend the benefits to a greater number of countries.

In the realm of precision agriculture projects, embedded systems can serve as powerful tools. By integrating sensors, controllers, and other hardware components, embedded systems enable real-time monitoring and control of various aspects of farm operations, such as irrigation, fertilization, and pest management. This integration empowers farmers to optimize crop production, minimize waste, and reduce their environmental footprint.

Embedded systems also facilitate the collection and analysis of large volumes of data, offering valuable insights into crop growth and health. By harnessing advanced analytics and machine learning algorithms, farmers can identify patterns and trends within the data, allowing them to make well-informed decisions and enhance overall farm performance.

In conclusion, the utilization of embedded systems in precision agriculture projects holds the potential to enhance crop yields, decrease costs, and promote sustainability. Therefore, it represents a promising area for innovation and investment for both farmers and agricultural technology companies.

To further augment precision agriculture, a web application specifically designed for this purpose integrates diverse agricultural data sources. This web application enables seamless crop monitoring and analysis, provides decision support tools, enables remote control and automation, offers data visualization and reporting capabilities, and empowers farmers with actionable insights. With its user-friendly interface, this web application enhances resource allocation, crop quality, and yield, all while promoting sustainability within the field of agriculture.

References

[1] C. Murugamani, S. Shitharth, S. Hemalatha, *et al.*, "Machine learning technique for precision agriculture applications in 5g-based internet of things", *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[2] R. Katarya, A. Raturi, A. Mehndiratta, and A. Thapper, "Impact of machine learning techniques in precision agriculture", in *2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE)*, IEEE, 2020, pp. 1–6.

[3] L. P. Chitra and R. Satapathy, "Performance comparison and evaluation of node.js and traditional web server (iis)", in *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, 2017, pp. 1–4. DOI: 10.1109/ICAMMAET.2017.8186633.

[4] E. Brown, *Web development with node and express: leveraging the JavaScript stack*. O'Reilly Media, 2019.

[5] S. Bradshaw, E. Brazil, and K. Chodorow, *MongoDB: the definitive guide: powerful and scalable data storage*. O'Reilly Media, 2019.

[6] *Get started with atlas*, 2023. [Online]. Available: https://docs.atlas.mongodb.com/getting-started.

[7] A. Del Sole, "Introducing visual studio code", in *Visual Studio Code Distilled: Evolved Code Editing for Windows, macOS, and Linux*. Berkeley, CA: Apress, 2021, pp. 1–15. DOI: 10.1007/978-1-4842-6901-5_1. [Online]. Available: https://doi.org/10.1007/978-1-4842-6901-5_1.

[8] D. Westerveld, *API Testing and Development with Postman: A practical guide to creating, testing, and managing APIs for automated software testing*. Packt Publishing Ltd, 2021.

[9] G. Lomidze, *Git and GitHub: The Complete Git and GitHub Course*. Packt Publishing Ltd, 2020.

[10] *Docs*, 2023. [Online]. Available: https://render.com/docs.

[11] A. Leff and J. Rayfield, "Web-application development using the model/view/controller design pattern", in *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*, 2001, pp. 118–127. DOI: 10.1109/EDOC.2001.950428.

[12] *Mvc design pattern*, 2023. [Online]. Available: https://www.geeksforgeeks.org/mvc-design-pattern/.

[13] S. Ahmed and Q. Mahmood, "An authentication based scheme for applications using json web token", in *2019 22nd International Multitopic Conference (INMIC)*, 2019, pp. 1–6. DOI: 10.1109/INMIC48123.2019.9022766.

[14] *Json web tokens*, 2023. [Online]. Available: https://auth0.com/docs/secure/tokens/json-web-tokens.

[15] P. Sturgeon and L. BOHILL, *Build APIs you wont hate*. LeanPub, https://leanpub.com/build-apis-youwont-hate, 2016.

[16] S. Allamaraju, *Restful web services cookbook: solutions for improving scalability and simplicity*. " O'Reilly Media, Inc.", 2010.

[17] R. T. Fielding, *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000.

[18] L. Rover, *Raspberry pi or arduino: When to choose which?*, https://www.leorover.tech/post/raspberry-pi-or-arduino-when-to-choose-which, accessed 2023.

[19] Opensource.com, *Raspberry pi resources*, https://opensource.com/resources/raspberry-pi, accessed 2023.

[20] Sixfab, *Smart agriculture projects using raspberry pi*, https://sixfab.com/smart-agriculture-projects-using-raspberry-pi/, accessed 2023.

[21] E. Power, *Photo resistor: Resistor guide*, https://eepower.com/resistor-guide/resistor-types/photo-resistor/, accessed 2023.

[22] Adafruit, *Pir (passive infrared) proximity motion sensor*, https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/overview, accessed 2023.

[23] E. Rite, *Temperature sensor probe types: How it works applications*, https://www.encardio.com/blog/temperature-sensor-probe-types-how-it-works-applications, accessed 2023.

[24] ScienceDirect, *Humidity sensor*, https://www.sciencedirect.com/topics/materials-science/humidity-sensor, accessed 2023.

[25] SemiconductorForU, *Soil moisture sensor: Working applications*, https://www.semiconductorforu.com/soil-moisture-sensor-working-its-applications/, accessed 2023.

[26] L. M. Engineers, *Water level sensor arduino tutorial*, https://lastminuteengineers.com/water-level-sensor-arduino-tutorial/, accessed 2023.

[27] Robocraze, *What is a raindrop sensor?*, https://robocraze.com/blogs/post/what-is-a-raindrop-sensor, accessed 2023.

[28] O. Engineering, *Ph meter*, https://www.omega.com/en-us/resources/ph-meter, accessed 2023.

[29] Codex, *Install android studio on macos and create a project*, 2023. [Online]. Available: https://medium.com/codex/install-android-studio-on-macos-and-create-a-project-fb8780d6f868.

[30] Google, *Basic android kotlin compose: First app*, 2023. [Online]. Available: https://developer.android.com/codelabs/basic-android-kotlin-compose-first-app#1.

[31] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[32] T. Lite, *Post-training quantization*, 2023. [Online]. Available: https://www.tensorflow.org/lite/performance/post_training_quantization.

[33] P. Patil, *Exploratory data analysis*, 2023. [Online]. Available: https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15.

[34] A. Sharma, A. Jain, P. Gupta, and V. Chowdary, "Machine learning applications for precision agriculture: A comprehensive review", *IEEE Access*, vol. 9, pp. 4843–4873, 2020.

Appendices

A Web Development Code

```
1    /**
2
3 app.js
4 This file is the entry point for the API application built using Express
      .js framework.
5 It sets up the server, database connection, middleware, and defines the
      routes for handling incoming requests.
6 */
7
8 // Import required modules
9 const path = require('path');
10 const express = require('express');
11 const bodyParser = require('body-parser');
12 const mongoose = require('mongoose');
13 const multer = require('multer');
14 const helmet = require('helmet');
15
16 // Import routers
17 const projectRoutes = require('./routes/project');
18 const authRoutes = require('./routes/auth');
19
20 // Set up database connection URI
21 const MONGODB_URI = `mongodb+srv://${process.env.MONGO_USER}:${process.
      env.MONGO_PASS}\@cluster0.bs6du.mongodb.net/${process.env.
      MONGO_DEF_DB}?retryWrites=true&w=majority`;
22
23 // Initialize Express application
24 const app = express();
25
26 // Set up security middleware
27 app.use(helmet());
28
29 // Configure multer for handling file uploads
30 const fileStorage = multer.diskStorage({
31     destination: (req, file, cb) => {
32         cb(null, 'images');
33     },
34     filename: (req, file, cb) => {
35         cb(null, new Date().toISOString().replace(/:/g, '.') + '-' +
      originalname);
36     },
37 });
38
39 // Configure photos filter
40 const fileFilter = (req, file, cb) => {
41     switch (file.mimetype) {
42         case 'image/png':
43         case 'image/jpg':
44         case 'image/jpeg':
45         case 'image/webp':
46             cb(null, true);     //accept the file
47             break;
```

```
48          default:
49              cb(null, false);    //reject the file
50      }
51  };
52
53  // Configure body parsers
54  app.use(bodyParser.json());
55  app.use(multer({storage: fileStorage, fileFilter}).single('image'));
56  app.use('/images', express.static(path.join(__dirname, 'images')));
57
58  //  Set up CORS Headers - Cross-Origin Resource Sharing
59  app.use((req, res, next) => {
60      res.setHeader('Access-Control-Allow-Origin', '*');
61      res.setHeader('Access-Control-Allow-Methods', 'OPTIONS, GET, POST,
    PUT, PATCH, DELETE');
62      res.setHeader('Access-Control-Allow-Headers', 'Origin, X-Requested-
    With, Content-Type, Accept, Authorization, method');
63      next();
64  });
65
66  // Set up routes
67  app.use('/auth', authRoutes);
68  app.use(projectRoutes);
69
70  // Route to handle requests to the API root domain
71  app.get('/', (req, res, next)=>{
72      // Send a welcome message to users accessing the root domain
73      return res.status(200).send(`
74          <h1>Welcome to the Precision Agriculture API!</h1>
75          <p>This API provides endpoints for accessing and managing
    precision agriculture data. It offers a range of features and
    functionalities to optimize agricultural processes, improve crop
    yield, and enhance farming efficiency.</p>
76          <p>Please refer to the API documentation for detailed
    information on available endpoints and how to interact with the API
    .</p>
77          <h2>Happy farming!</h2>
78      `);
79  })
80
81  // Error handler middleware
82  app.use((error, req, res, next)=>{
83      console.log(error);
84      const status = error.statusCode || 500; //setting default value of
    500
85      const message = error.message;
86      const data = error.data;
87      res.status(status).json({message: message, data});
88  });
89
90  // Connect to the database and start the server
91  mongoose
92      .connect(MONGODB_URI)
93      .then(result => {
94          app.listen(process.env.PORT || 8080); //localhost:8080 in
    development
95          console.log('connected');
96      })
```

```
97      .catch(err => {
98          console.log(err);
99      });
100
101
102
103 /**
104
105 user.js
106 This file defines the user model for the API application using Mongoose.
107 */
108
109 // Import required modules
110 const mongoose = require('mongoose');
111
112 const Schema = mongoose.Schema;
113
114 // Define user schema
115 const userSchema = new Schema({
116     name:{
117         type: String,
118         required: true
119     },
120     email:{
121         type: String,
122         required: true
123     },
124     password:{
125         type: String,
126         required: true
127     },
128     isSuperAdmin:{
129         type: Boolean,
130         default: false,
131     },
132 },{ timestamps: true });
133
134 // Export the user model
135 module.exports = mongoose.model('User', userSchema);
136
137
138 /**
139
140 auth.js
141 This file defines the authentication routes for the API application
        using Express.js.
142 It handles user signup and login requests.
143 */
144
145 // Import required modules
146 const express = require('express');
147
148 const {
149     body
150 } = require('express-validator');
151
152 const User = require('../models/user');
153
```

```
154 const authController = require('../controllers/auth');
155
156 const router = express.Router();
157
158 // /auth/signup => PUT
159 router.put('/signup', [
160         body('email')
161             .isEmail()
162             .withMessage('Please enter a valid email.')
163             .custom((value, {req}) => {
164                 return User.findOne({email: value})
165                     .then(userDoc => {
166                         if (userDoc) {
167                             return Promise.reject('Email already in use'
    );
168                         }
169                     })
170         })
171         .normalizeEmail(),
172         body('password')
173             .trim()
174             .isLength({min: 5}), //minimum length
175         body('name')
176             .trim()
177             .not()
178             .isEmpty()
179
180     ],
181     authController.signup);
182
183 // auth/login => POST
184 router.post('/login', authController.login);
185
186 module.exports = router;
187
188
189 /**
190
191 auth.js
192 This file contains the authentication controller functions for the API
    application.
193 It includes the signup and login functions.
194 */
195
196 // Import required modules
197 const bcrypt = require('bcryptjs');
198 // const nodemailer = require('nodemailer');
199 // const sendgridTransport = require('nodemailer-sendgrid-transport');
200 const jwt = require('jsonwebtoken');
201
202 const {
203     validationResult
204 } = require('express-validator');
205
206 const User = require('../models/user');
207
208 //Handle user signup
209 exports.signup = (req, res, next) => {
```

```
210     const errors = validationResult(req);
211     if(!errors.isEmpty()){
212         const error =  new Error('Validation failed.');
213         error.statusCode = 422;
214         error.data = errors.array();
215         throw error;
216     }
217     const name = req.body.name;
218     const email = req.body.email;
219     const password = req.body.password;
220     bcrypt.hash(password, 12)
221     .then(hashedPassword => {
222         const user = new User({
223             name,
224             email,
225             password: hashedPassword
226         });
227         return user.save();
228     })
229     .then(result =>{
230         res.status(201).json({message: 'User created.', userId: result._id});
231 
232     })
233     .catch(err=>{
234         if (!err.statusCode) {
235             err.statusCode = 500;
236         }
237         next(err);
238     });
239 };
240 
241 //Handle user login
242 exports.login = (req, res, next) => {
243     const password = req.body.password;
244     const email = req.body.email;
245     let loadedUser;
246     User.findOne({email})
247     .then(user => {
248         if(!user){
249             const error =  new Error('A user with this email could not be found.');
250             res.statusCode = 401;
251             throw error;
252         }
253         loadedUser = user;
254         return bcrypt.compare(password, user.password);
255     })
256     .then(isEqual => {
257         if(!isEqual){
258             console.log("not successful");
259             const error =  new Error('Wrong password');
260             res.statusCode = 401;
261             throw error;
262         }
263         const token = jwt.sign({
264             email: loadedUser.email,
265             id: loadedUser._id.toString(),
```

```
            isSuperAdmin: loadedUser.isSuperAdmin
        },
        process.env.JWT_SECRET,
        { expiresIn: '1h' }
        );
        console.log("successful");
        res.status(200).json({token, userId: loadedUser._id.toString()
    });
     })
     .catch(err=>{
         if (!err.statusCode) {
             err.statusCode = 500;
         }
         next(err);
     });
};


/**

is-auth.js
This file defines the authentication middleware for the API application.
It checks if the user is authenticated by verifying the JWT token in the
     Authorization header.
*/

// Import required modules
const jwt = require('jsonwebtoken');

module.exports = (req, res, next) => {
     const authHeader = req.get('Authorization');
     if (!authHeader){
         const error = new Error('Not authenticated.')
         error.statusCode = 401;
         throw error;
     }

     const token = authHeader.split(' ')[1];
     //The setup in front-end must be: headers: {Authorization: 'Bearer '
     + this.props.token}
     let decodedToken;
     try{
         decodedToken= jwt.verify(token, process.env.JWT_SECRET)
     } catch (err) {
         err.statusCode =500;
         throw err
     }
     if (!decodedToken){
         const error = new Error('Not authenticated.')
         error.statusCode = 401;
         throw error;
     }
     req.userId = decodedToken.id;
     req.isSuperAdmin = decodedToken.isSuperAdmin;
     next();
};
```

B Embedded System Code

```
import spidev
import time
import RPi.GPIO as GPIO
import pio
import Ports
import os



GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)


pio.uart = Ports.UART()  # Define serial port


# Open SPI bus
spi = spidev.SpiDev()
spi.open(0, 0)



# define_pins

redlamp_pin = 32
motor_pin = 33
temp_pin = 11
temp_channel = 0
ldr_channel = 1
soil_channel = 2
humidity_channel = 3
ph_channel = 4
water_channel = 4
rain_pin = 29
pir_pin = 37
acid_pin=35
alkile_pin=36

# difine lcd pins

LCD_RS = 7
LCD_E = 11
LCD_D4 = 12
LCD_D5 = 13
LCD_D6 = 15
LCD_D7 = 16


# setup pins

GPIO.setup(redlamp_pin, GPIO.OUT)  # red lamp and motor
GPIO.setup(motor_pin, GPIO.OUT)  # red lamp and motor
GPIO.setup(acid_pin, GPIO.OUT)  # acide
GPIO.setup(alkile_pin, GPIO.OUT)  # alkile
GPIO.setup(rain_pin, GPIO.IN)
GPIO.setup(pir_pin, GPIO.IN)
```

```
57
58
59  # Timing constants
60  E_PULSE = 0.0005
61  E_DELAY = 0.0005
62  delay = 1
63
64  # All of lcd interface
65
66  # setup lcd pins
67  GPIO.setup(LCD_E, GPIO.OUT)  # E
68  GPIO.setup(LCD_RS, GPIO.OUT)  # RS
69  GPIO.setup(LCD_D4, GPIO.OUT)  # DB4
70  GPIO.setup(LCD_D5, GPIO.OUT)  # DB5
71  GPIO.setup(LCD_D6, GPIO.OUT)  # DB6
72  GPIO.setup(LCD_D7, GPIO.OUT)  # DB7
73
74  # Define some device constants
75  LCD_WIDTH = 16    # Maximum characters per line
76  LCD_CHR = True
77  LCD_CMD = False
78  LCD_LINE_1 = 0x80  # LCD RAM address for the 1st line
79  LCD_LINE_2 = 0xC0  # LCD RAM address for the 2nd line
80
81
82  '''
83  Function Name :lcd_init()
84  Function Description : this function is used to initialized lcd by
        sending the different commands
85  '''
86
87
88  def lcd_init():
89    # Initialise display
90    lcd_byte(0x33, LCD_CMD)  # 110011 Initialise
91    lcd_byte(0x32, LCD_CMD)  # 110010 Initialise
92    lcd_byte(0x06, LCD_CMD)  # 000110 Cursor move direction
93    lcd_byte(0x0C, LCD_CMD)  # 001100 Display On,Cursor Off, Blink Off
94    lcd_byte(0x28, LCD_CMD)  # 101000 Data length, number of lines, font
        size
95    lcd_byte(0x01, LCD_CMD)  # 000001 Clear display
96    time.sleep(E_DELAY)
97
98
99  '''
100 Function Name :lcd_byte(bits ,mode)
101 Fuction Name :the main purpose of this function to convert the byte data
        into bit and send to lcd port
102 '''
103
104
105 def lcd_byte(bits, mode):
106   # Send byte to data pins
107   # bits = data
108   # mode = True  for character
109   #        False for command
110
111   GPIO.output(LCD_RS, mode)  # RS
```

```
112
113   # High bits
114   GPIO.output(LCD_D4, False)
115   GPIO.output(LCD_D5, False)
116   GPIO.output(LCD_D6, False)
117   GPIO.output(LCD_D7, False)
118   if bits & 0x10 == 0x10:
119     GPIO.output(LCD_D4, True)
120   if bits & 0x20 == 0x20:
121     GPIO.output(LCD_D5, True)
122   if bits & 0x40 == 0x40:
123     GPIO.output(LCD_D6, True)
124   if bits & 0x80 == 0x80:
125     GPIO.output(LCD_D7, True)
126
127   # Toggle 'Enable' pin
128   lcd_toggle_enable()
129
130   # Low bits
131   GPIO.output(LCD_D4, False)
132   GPIO.output(LCD_D5, False)
133   GPIO.output(LCD_D6, False)
134   GPIO.output(LCD_D7, False)
135   if bits & 0x01 == 0x01:
136     GPIO.output(LCD_D4, True)
137   if bits & 0x02 == 0x02:
138     GPIO.output(LCD_D5, True)
139   if bits & 0x04 == 0x04:
140     GPIO.output(LCD_D6, True)
141   if bits & 0x08 == 0x08:
142     GPIO.output(LCD_D7, True)
143
144   # Toggle 'Enable' pin
145   lcd_toggle_enable()
146
147
148 '''
149 Function Name : lcd_toggle_enable()
150 Function Description:basically this is used to toggle Enable pin
151 '''
152
153
154 def lcd_toggle_enable():
155   # Toggle enable
156   time.sleep(E_DELAY)
157   GPIO.output(LCD_E, True)
158   time.sleep(E_PULSE)
159   GPIO.output(LCD_E, False)
160   time.sleep(E_DELAY)
161
162
163 '''
164 Function Name :lcd_string(message,line)
165 Function  Description :print the data on lcd
166 '''
167
168
169 def lcd_string(message, line):
```

```
  # Send string to display

  message = message.ljust(LCD_WIDTH, " ")

  lcd_byte(line, LCD_CMD)

  for i in range(LCD_WIDTH):
    lcd_byte(ord(message[i]), LCD_CHR)


# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
def ReadChannel(channel):
  adc = spi.xfer2([1, (8+channel) << 4, 0])
  data = ((adc[1] & 3) << 8) + adc[2]
  return data


# Function to calculate temperature from
# TMP36 data, rounded to specified
# number of decimal places.
def ConvertTemp(data, places):

  # ADC Value
  # (approx)  Temp  Volts
  #    0      -50    0.00
  #   78      -25    0.25
  #  155        0    0.50
  #  233       25    0.75
  #  310       50    1.00
  #  465      100    1.50
  #  775      200    2.50
  # 1023      280    3.30

  temp = ((data * 330)/float(1023))
  temp = round(temp, places)
  return temp

 # ----------------------------------------


def ConvertHumidity(data, tempoo, places):

  # RH = ((((data/1023)*5)-0.8)/3.1)*100
  # RH = round(RH,places)

   # rhvoltage = (data/512)*2.5; # are converted into relaive humidity
   percentage

   # RH = ((rhvoltage/3.3)-0.1515)/0.00636;
   # RH = round(RH,places)

   # from data sheet

   # v=3.3*(0.00636*(data) + 0.1515 )

   # v = round(v,places)

```

```
    RH = data * (1.0546 - (0.00216 * tempoo))
    RH = round(RH, places)
    return RH
 
 
 
 
#------------------------------------------------

# Define function to convert raw ADC value to pH value
def convert_to_ph(raw_adc_value):
    # Insert your inversion formula here
    # This will depend on the specific pH sensor/meter you are using
    # Consult the datasheet for the sensor/meter for the inversion formula
    ph_value = 7 - ((raw_adc_value * 5) / 1024)   # Example inversion formula
    return ph_value

# Define delay between readings
delay = 5
lcd_init()
lcd_string("welcome ", LCD_LINE_1)


time.sleep(2)


# infinite loop
while 1:

  rain_data = GPIO.input(rain_pin)

  pir_data = GPIO.input(pir_pin)

  # Print out results

   lcd_init()
   lcd_string("Light Intencity: ", LCD_LINE_1)
   ldr_level = ReadChannel(ldr_channel)
   lcd_string(str(ldr_level), LCD_LINE_2)
   pio.uart.println("light : "+str(ldr_level))
   time.sleep(0.1)

   if ((pir_data == True)):
    lcd_init()
    lcd_string("Person Detected", LCD_LINE_1)
    time.sleep(0.1)

    if (ldr_level < 250):
     lcd_init()
     lcd_string("Farm Lights :", LCD_LINE_1)
     lcd_string("Active ", LCD_LINE_2)
     GPIO.output(redlamp_pin, True)  # red lamp and motor
     time.sleep(0.1)

    else:
     lcd_init()
```

```
283       lcd_string("Farm Lights :", LCD_LINE_1)
284       lcd_string("Turned Off ", LCD_LINE_2)
285       GPIO.output(redlamp_pin, False)  # red lamp and motor
286       time.sleep(0.1)
287
288     else:
289      lcd_init()
290      lcd_string("No Person :", LCD_LINE_1)
291      lcd_string("Lights Default ", LCD_LINE_2)
292      GPIO.output(redlamp_pin, False)  # red lamp and motor
293      time.sleep(0.1)
294
295     lcd_init()
296     lcd_string("Temperature : ", LCD_LINE_1)
297     temp_level = ReadChannel(temp_channel)
298     temp = ConvertTemp(temp_level, 2)
299     lcd_string(str(temp)+" C", LCD_LINE_2)
300     time.sleep(0.1)
301
302     lcd_init()
303     lcd_string("Humidity : ", LCD_LINE_1)
304     humidity_level = ReadChannel(humidity_channel)
305     humidity = ConvertHumidity(humidity_level, temp, 2)
306     lcd_string(str(humidity)+" %RH", LCD_LINE_2)
307     time.sleep(0.1)
308
309     lcd_init()
310     lcd_string("Soil Moisture : ", LCD_LINE_1)
311     soil_level = ReadChannel(soil_channel)
312     lcd_string(str(soil_level), LCD_LINE_2)
313     time.sleep(0.1)
314
315     lcd_init()
316     lcd_string("Water Level: ", LCD_LINE_1)
317     water_level = ReadChannel(water_channel)
318     lcd_string(str(water_level), LCD_LINE_2)
319     time.sleep(0.1)
320
321   lcd_init()
322   lcd_string("PH Level: ", LCD_LINE_1)
323   ph_level = ReadChannel(ph_channel)
324   ph_value = convert_to_ph(ph_level)
325   lcd_string(str(ph_value), LCD_LINE_2)
326
327
328   time.sleep(0.1)
329
330    if (rain_data == True):
331     lcd_init()
332     lcd_string("WARNING......", LCD_LINE_1)
333     lcd_string("RAIN DROP ", LCD_LINE_2)
334
335     GPIO.output(motor_pin, False)
336     time.sleep(0.1)
337
338    if (temp > 30) and (soil_level < 100) and (rain_data != True) and (
       water_level < 300):
339     lcd_init()
```

```
    lcd_string("Motor State:  ", LCD_LINE_1)
    lcd_string('ON', LCD_LINE_2)
    GPIO.output(motor_pin, True)
    time.sleep(0.1)

   else:
    lcd_init()
    lcd_string("Motor State:  ", LCD_LINE_1)
    lcd_string('Turned OFF', LCD_LINE_2)
    GPIO.output(motor_pin, False)
    time.sleep(0.1)

  if (ph_value < 6.5):
    lcd_init()
    lcd_string("Alkile Motor:  ", LCD_LINE_1)
    lcd_string('ON', LCD_LINE_2)
    #GPIO.output(alkile_pin, True)
    time.sleep(0.1)
    lcd_init()
    lcd_string("Acide Motor:  ", LCD_LINE_1)
    lcd_string('OFF', LCD_LINE_2)
    GPIO.output(acid_pin, False)
    time.sleep(0.1)

  if (ph_value > 6.5):
    lcd_init()
    lcd_string("Acide Motor:  ", LCD_LINE_1)
    lcd_string('ON', LCD_LINE_2)
    GPIO.output(acid_pin, True)
    time.sleep(0.1)
    lcd_init()
    lcd_string("Alkile Motor:  ", LCD_LINE_1)
    lcd_string('OFF', LCD_LINE_2)
    GPIO.output(alkile_pin, False)
    time.sleep(0.1)
```

C Mobile Development Code

C.1 Layout XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http:
    //schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        app:layout_constraintBottom_toTopOf="@+id/button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.5"
        tools:srcCompat="@tools:sample/avatars" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Select"
        app:layout_constraintBottom_toTopOf="@+id/textView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView"
        app:layout_constraintVertical_bias="0.5" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="112dp"
        android:layout_height="231dp"
        android:text="TextView"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button"
        app:layout_constraintVertical_bias="0.5" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

C.2 Main Java Code:

```
package com.example.precisionagriculture;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import android.annotation.SuppressLint;
import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import com.example.precisionagriculture.ml.OptimzedModel;
import org.opencv.android.OpenCVLoader;
import org.tensorflow.lite.DataType;
import org.tensorflow.lite.support.image.TensorImage;
import org.tensorflow.lite.support.tensorbuffer.TensorBuffer;
```

```
19 import java.io.IOException;
20 import java.nio.ByteBuffer;
21 import java.util.Arrays;
22
23 public class MainActivity extends AppCompatActivity {
24
25     ImageView imgView;
26     TextView tv;
27     Bitmap img;
28     Button select;
29
30     @SuppressLint("MissingInflatedId")
31     @Override
32     protected void onCreate(Bundle savedInstanceState) {
33         super.onCreate(savedInstanceState);
34         setContentView(R.layout.activity_main);
35         imgView = findViewById(R.id.imageView);
36         tv = findViewById(R.id.textView);
37         select = findViewById(R.id.button);
38
39         if(OpenCVLoader.initDebug()) Log.d("LOADED","success");
40         else Log.d("LOADED","err");
41
42
43         select.setOnClickListener(view -> {
44             Intent intent= new Intent(Intent.ACTION_GET_CONTENT);
45             intent.setType("image/*");
46             startActivityForResult(intent,100);
47
48             tv.setText("");
49         }); }
50
51     @Override
52     protected void onActivityResult(int requestCode, int resultCode,
   @Nullable Intent data) {
53         super.onActivityResult(requestCode, resultCode, data);
54
55         if(requestCode==100) {
56             assert data != null;
57             imgView.setImageURI(data.getData());
58
59             Uri uri = data.getData();
60             try {
61                 img = MediaStore.Images.Media.getBitmap(this.
   getContentResolver(), uri);
62             } catch (IOException e) {
63                 throw new RuntimeException(e);
64             } }
65
66         img=Bitmap.createScaledBitmap(img,256,256,true);
67
68         try {
69             OptimzedModel model = OptimzedModel.newInstance(
   getApplicationContext());
70
71             // Creates inputs for reference.
72             TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(
   new int[]{1, 256, 256, 3}, DataType.FLOAT32);
```

```
            TensorImage tensorImage = new TensorImage(DataType.FLOAT32);
            tensorImage.load(img);
            ByteBuffer byteBuffer = tensorImage.getBuffer();
            inputFeature0.loadBuffer(byteBuffer);

            // Runs model inference and gets result.
            OptimzedModel.Outputs outputs = model.process(inputFeature0)
    ;
            TensorBuffer outputFeature0 = outputs.
    getOutputFeature0AsTensorBuffer();

            // Releases model resources if no longer used.
            model.close();

            int[] out = outputFeature0.getIntArray();

            String[] diseases = {"Apple___Apple_scab", "
    Apple___Black_rot", "Apple___Cedar_apple_rust", "Apple___healthy", "
    Blueberry___healthy", "Cherry_(including_sour)___Powdery_mildew", "
    Cherry_(including_sour)___healthy", "Corn_(maize)
    ___Cercospora_leaf_spot Gray_leaf_spot",
                    "Corn_(maize)___Common_rust_", "Corn_(maize)
    ___Northern_Leaf_Blight", "Corn_(maize)___healthy",
                    "Grape___Black_rot", "Grape___Esca_(Black_Measles)",
     "Grape___Leaf_blight_(Isariopsis_Leaf_Spot)",
                    "Grape___healthy", "Orange___Haunglongbing_(
    Citrus_greening)", "Peach___Bacterial_spot", "Peach___healthy",
                    "Pepper,_bell___Bacterial_spot", "Pepper,
    _bell___healthy", "Potato___Early_blight", "Potato___Late_blight",
                    "Potato___healthy", "Raspberry___healthy", "
    Soybean___healthy", "Squash___Powdery_mildew", "
    Strawberry___Leaf_scorch",
                    "Strawberry___healthy", "Tomato___Bacterial_spot", "
    Tomato___Early_blight", "Tomato___Late_blight",
                    "Tomato___Leaf_Mold", "Tomato___Septoria_leaf_spot",
     "Tomato___Spider_mites Two-spotted_spider_mite",
                    "Tomato___Target_Spot", "
    Tomato___Tomato_Yellow_Leaf_Curl_Virus", "
    Tomato___Tomato_mosaic_virus", "Tomato___healthy"};
            tv.setText(Arrays.toString(out));
                for (int i = 0; i < out.length; i++) {

                    if (out[i] == 1)
                        tv.setText(diseases[i]);
                }
        }catch (IOException e) {
            System.out.println("Crash");
        } } }
```

C.3 TF Lite Java Code:

```
    try {
    OptimzedModel model = OptimzedModel.newInstance(context);

```

```
    // Creates inputs for reference.
    TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int
    []{1, 256, 256, 3}, DataType.FLOAT32);
    inputFeature0.loadBuffer(byteBuffer);

    // Runs model inference and gets result.
    OptimzedModel.Outputs outputs = model.process(inputFeature0);
    TensorBuffer outputFeature0 = outputs.
    getOutputFeature0AsTensorBuffer();

    // Releases model resources if no longer used.
    model.close();
} catch (IOException e) {
    // TODO Handle the exception
}
```

D Deep Learning Code

D.1 Plant Disease Detection Code

```
# Commented out IPython magic to ensure Python compatibility.
import os
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import glob as glob
import itertools
import seaborn as sns
from itertools import chain
from tqdm import tqdm, tqdm_notebook, tnrange
from sklearn.metrics import precision_score, accuracy_score,
    recall_score, confusion_matrix, ConfusionMatrixDisplay
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
    img_to_array, load_img
from tensorflow.keras.models import Sequential, Model, load_model,
    save_model
from tensorflow.keras.layers import (Conv2D, MaxPooling2D, Dense,
    Flatten, Activation)
from tensorflow.keras import backend as K

# \%matplotlib inline

#data_path = "/kaggle/input/new-plant-diseases-dataset/"
train_path = "/kaggle/input/new-plant-diseases-dataset/New Plant
    Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)/
    train"
val_path = "/kaggle/input/new-plant-diseases-dataset/New Plant Diseases
    Dataset(Augmented)/New Plant Diseases Dataset(Augmented)/valid"
test_path = "/kaggle/input/new-plant-diseases-dataset/test/test"
```

```
classes = os.listdir(train_path)

train_num, val_num = {}, {}

for each in classes:
    train_num[each] = len(os.listdir(os.path.join(train_path, each)))
    val_num[each] = len(os.listdir(os.path.join(val_path, each)))
img_per_class = pd.DataFrame(train_num.values(), train_num.keys(),
   columns=["Image per Class"])
img_per_class

val_img_per_class = pd.DataFrame(val_num.values(), val_num.keys(),
   columns=['Val images per Class'])

train_data =keras.utils.image_dataset_from_directory(train_path ,
                                          image_size=(256, 256))
val_data = keras.utils.image_dataset_from_directory(val_path,
                                        image_size=(256, 256))

rescale = keras.layers.Rescaling(scale=1.0/255)

train_gen = train_data.map(lambda image,label:(rescale(image),label))
val_gen  = val_data.map(lambda image,label:(rescale(image),label))
test_gen = val_data.map(lambda image,label:(rescale(image),label))

model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same'
   , input_shape=(256, 256, 3)),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same'
   ),
    tf.keras.layers.MaxPooling2D(3, 3),

    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same'
   ),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same'
   ),
    tf.keras.layers.MaxPooling2D(3, 3),

    tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same
   '),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same
   '),
    tf.keras.layers.MaxPooling2D(3, 3),

    tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same
   '),
    tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same
   '),

    tf.keras.layers.Conv2D(512, (5, 5), activation='relu', padding='same
   '),
    tf.keras.layers.Conv2D(512, (5, 5), activation='relu', padding='same
   '),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dropout(.5),

```

```
    tf.keras.layers.Dense(38, activation='softmax')
])

lr = 0.0001
model.compile(loss='sparse_categorical_crossentropy',
                optimizer=keras.optimizers.Adam(learning_rate=lr),
                metrics=["accuracy"])

model.summary()

history = model.fit(train_gen, validation_data=val_gen, epochs=20,
    verbose=1)

labels = []
predictions = []
for x,y in val_gen:
    labels.append(list(y.numpy()))
    predictions.append(tf.argmax(model.predict(x),1).numpy())

print("Train Accuracy  : {:.2f} \%".format(history.history['accuracy'
    ][-1]*100))
print("Test Accuracy   : {:.2f} \%".format(accuracy_score(labels,
    predictions) * 100))
print("Precision Score : {:.2f} \%".format(precision_score(labels,
    predictions, average='micro') * 100))
print("Recall Score    : {:.2f} \%".format(recall_score(labels,
    predictions, average='micro') * 100))

Li = ['Apple___Apple_scab', 'Apple___Black_rot', '
    Apple___Cedar_apple_rust', 'Apple___healthy', 'Blueberry___healthy',
    'Cherry_(including_sour)___Powdery_mildew', 'Cherry_(including_sour)
    ___healthy', 'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot',
      'Corn_(maize)___Common_rust_', 'Corn_(maize)
    ___Northern_Leaf_Blight', 'Corn_(maize)___healthy',
      'Grape___Black_rot', 'Grape___Esca_(Black_Measles)', '
    Grape___Leaf_blight_(Isariopsis_Leaf_Spot)',
      'Grape___healthy', 'Orange___Haunglongbing_(Citrus_greening)', '
    Peach___Bacterial_spot', 'Peach___healthy',
      'Pepper,_bell___Bacterial_spot', 'Pepper,_bell___healthy', '
    Potato___Early_blight', 'Potato___Late_blight',
      'Potato___healthy', 'Raspberry___healthy', 'Soybean___healthy', '
    Squash___Powdery_mildew', 'Strawberry___Leaf_scorch',
      'Strawberry___healthy', 'Tomato___Bacterial_spot', '
    Tomato___Early_blight', 'Tomato___Late_blight',
      'Tomato___Leaf_Mold', 'Tomato___Septoria_leaf_spot', '
    Tomato___Spider_mites Two-spotted_spider_mite',
      'Tomato___Target_Spot', 'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
    'Tomato___Tomato_mosaic_virus', 'Tomato___healthy']

# predicting an image
import os
import matplotlib.pyplot as plt
from keras.preprocessing import image
import numpy as np
files = [os.path.join(test_path,p) for p in sorted(os.listdir(test_path)
    )]
for i in range(0,10):
    image_path = files[i]
```

```
    new_img =keras.utils.load_img(image_path, target_size=(256, 256))
    img = keras.utils.img_to_array(new_img)
    img = np.expand_dims(img, axis=0)
    img = img/255
    prediction = model.predict(img)
    probabilty = prediction.flatten()
    max_prob = probabilty.max()
    index=prediction.argmax(axis=-1)[0]
    class_name = Li[index]
    #ploting image with predicted class name
    plt.figure(figsize = (4,4))
    plt.imshow(new_img)
    plt.axis('off')
    plt.title(class_name+" "+ str(max_prob)[0:4]+"\%")
    plt.show()

def get_file_size(file_path):
    size = os.path.getsize(file_path)
    return size

def convert_bytes(size, unit=None):
    if unit == "KB":
        return print('File size: ' + str(round(size / 1024, 3)) + '
   Kilobytes')
    elif unit == "MB":
        return print('File size: ' + str(round(size / (1024 * 1024), 3))
    + ' Megabytes')
    else:
        return print('File size: ' + str(size) + ' bytes')

TF_LITE_MODEL_FILE_NAME = "plant_cnn_model.tflite"
tf_lite_converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = tf_lite_converter.convert()

tflite_model_name = TF_LITE_MODEL_FILE_NAME
open(tflite_model_name, "wb").write(tflite_model)

convert_bytes(get_file_size(TF_LITE_MODEL_FILE_NAME), "KB")

import tensorflowjs as tfjs
tfjs.converters.save_keras_model(model, 'plant_cnn_model_tfjs')

import shutil
shutil.make_archive("/kaggle/working/plant_cnn_model_tfjs", 'zip', '/
   kaggle/working/')
```

D.2 Crop Recommendation Code

```
# Importing libraries

from __future__ import print_function
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
7  import seaborn as sns
8  from sklearn.metrics import classification_report
9  from sklearn import metrics
10 from sklearn import tree
11 import warnings
12 warnings.filterwarnings('ignore')
13
14 import numpy as np
15 import pandas as pd
16 from sklearn.linear_model import LogisticRegression
17 from scipy.interpolate import make_interp_spline
18 import matplotlib.pyplot as plt
19 import seaborn as sns
20 from sklearn.metrics import accuracy_score as acc
21 from sklearn.preprocessing import StandardScaler
22 from sklearn.model_selection import train_test_split
23 from sklearn.metrics import confusion_matrix
24 import plotly.graph_objects as go
25 from sklearn.utils import shuffle
26 from sklearn.metrics import mean_absolute_error
27 from sklearn.metrics import mean_squared_error
28 from sklearn.preprocessing import LabelEncoder
29 from sklearn.metrics import r2_score
30 from sklearn.datasets import make_classification
31 from sklearn.multioutput import MultiOutputClassifier
32 from sklearn.ensemble import RandomForestClassifier
33
34 from google.colab import drive
35 drive.mount('/content/drive')
36
37 df = pd.read_csv('/content/drive/MyDrive/test-ml/crop_recommendation.csv
       ')
38
39 all_columns = df.columns[:-1]
40
41 df  = shuffle(df,random_state=5)
42
43 """### Seperating features and target label"""
44
45 features = df[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall'
       ]]
46 target = df['label']
47 #features = df[['temperature', 'humidity', 'ph', 'rainfall']]
48 labels = df['label']
49
50 # Initialzing empty lists to append all model names and corresponding
       names
51 acc = []
52 model = []
53
54 # Splitting into train and test data
55
56 from sklearn.model_selection import train_test_split
57 Xtrain, Xtest, Ytrain, Ytest = train_test_split(features,target,
       test_size = 0.2,random_state =2)
58
59 """# Decision Tree"""
60
```

```
from sklearn.tree import DecisionTreeClassifier

DecisionTree = DecisionTreeClassifier(criterion="entropy",random_state
    =2,max_depth=5)

DecisionTree.fit(Xtrain,Ytrain)

predicted_values = DecisionTree.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Decision Tree')
print("DecisionTrees's Accuracy is: ", x*100)

print(classification_report(Ytest,predicted_values))

from sklearn.model_selection import cross_val_score

# Cross validation score (Decision Tree)
score = cross_val_score(DecisionTree, features, target,cv=5)

score

"""### Saving trained Decision Tree model"""

import pickle
# Dump the trained Naive Bayes classifier with Pickle
DT_pkl_filename = 'DecisionTree.pkl'
# Open the file to save as pkl file
DT_Model_pkl = open(DT_pkl_filename, 'wb')
pickle.dump(DecisionTree, DT_Model_pkl)
# Close the pickle instances
DT_Model_pkl.close()

"""# Guassian Naive Bayes"""

from sklearn.naive_bayes import GaussianNB

NaiveBayes = GaussianNB()

NaiveBayes.fit(Xtrain,Ytrain)

predicted_values = NaiveBayes.predict(Xtest)
x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Naive Bayes')
print("Naive Bayes's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))

# Cross validation score (NaiveBayes)
score = cross_val_score(NaiveBayes,features,target,cv=5)
score

"""### Saving trained Guassian Naive Bayes model"""
# Dump the trained Naive Bayes classifier with Pickle
NB_pkl_filename = 'NBClassifier.pkl'
# Open the file to save as pkl file
NB_Model_pkl = open(NB_pkl_filename, 'wb')
```

```
pickle.dump(NaiveBayes, NB_Model_pkl)
# Close the pickle instances
NB_Model_pkl.close()

"""# Support Vector Machine (SVM)"""

from sklearn.svm import SVC
# data normalization with sklearn
from sklearn.preprocessing import MinMaxScaler
# fit scaler on training data
norm = MinMaxScaler().fit(Xtrain)
X_train_norm = norm.transform(Xtrain)
# transform testing dataabs
X_test_norm = norm.transform(Xtest)
SVM = SVC(kernel='poly', degree=3, C=1)
SVM.fit(X_train_norm,Ytrain)
predicted_values = SVM.predict(X_test_norm)
x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('SVM')
print("SVM's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))

# Cross validation score (SVM)
score = cross_val_score(SVM,features,target,cv=5)
score

#Saving trained SVM model
# Dump the trained SVM classifier with Pickle
SVM_pkl_filename = 'SVMClassifier.pkl'
# Open the file to save as pkl file
SVM_Model_pkl = open(SVM_pkl_filename, 'wb')
pickle.dump(SVM, SVM_Model_pkl)
# Close the pickle instances
SVM_Model_pkl.close()

"""# Logistic Regression"""

from sklearn.linear_model import LogisticRegression

LogReg = LogisticRegression(random_state=2)

LogReg.fit(Xtrain,Ytrain)

predicted_values = LogReg.predict(Xtest)

x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('Logistic Regression')
print("Logistic Regression's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))

# Cross validation score (Logistic Regression)
score = cross_val_score(LogReg,features,target,cv=5)
score

```

```
"""### Saving trained Logistic Regression model"""
# Dump the trained Naive Bayes classifier with Pickle
LR_pkl_filename = 'LogisticRegression.pkl'
# Open the file to save as pkl file
LR_Model_pkl = open(LR_pkl_filename, 'wb')
pickle.dump(LogReg, LR_Model_pkl)
# Close the pickle instances
LR_Model_pkl.close()

"""# Random Forest"""

from sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(n_estimators=20, random_state=0)
RF.fit(Xtrain,Ytrain)

predicted_values = RF.predict(Xtest)

x = metrics.accuracy_score(Ytest, predicted_values)
acc.append(x)
model.append('RF')
print("RF's Accuracy is: ", x)

print(classification_report(Ytest,predicted_values))

# Cross validation score (Random Forest)
score = cross_val_score(RF,features,target,cv=5)
score

"""### Saving trained Random Forest model"""
# Dump the trained Naive Bayes classifier with Pickle
RF_pkl_filename = 'RandomForest.pkl'
# Open the file to save as pkl file
RF_Model_pkl = open(RF_pkl_filename, 'wb')
pickle.dump(RF, RF_Model_pkl)
# Close the pickle instances
RF_Model_pkl.close()

"""## Accuracy Comparison"""

accuracy_models = dict(zip(model, acc))
for k, v in accuracy_models.items():
    print (k, '-->', v)

"""## Making a prediction"""

data = np.array([[104,18, 30, 23.603016, 60.3, 6.7, 140.91]])
prediction = RF.predict(data)
print(prediction)

data = np.array([[83, 45, 60, 28, 70.3, 7.0, 150.9]])
prediction = RF.predict(data)
print(prediction)
```