SMART AGRICULTURE USING MACHINE LEARNING AND IOT

SYNOPSIS

The project "**Smart Agriculture Using Machine Learning and IoT**" proposes an innovative solution to empower farmers with real-time data on crucial environmental parameters such as temperature and soil moisture. The advent of Internet of Things (IoT) technology has sparked a revolution across various domains of everyday life, imbuing them with smart and intelligent capabilities. IoT encompasses a network of interconnected devices that form a self-configuring system. The ongoing advancement in Intelligent Smart Farming IoT devices is reshaping agricultural production by not only augmenting its efficiency but also rendering it more cost-effective and reducing wastage.

The primary objective of this report is to propose an IoT-based Smart Agriculture System tailored to empower farmers with real-time data on critical environmental parameters such as temperature and soil moisture. The system facilitates the utilization of a Machine Learning model for recommending suitable crop yields through a web application. This system aims to facilitate efficient monitoring of agricultural conditions, thereby empowering farmers to enhance both the yield and quality of their produce. The proposed IoT-based Smart Agriculture System outlined in this project leverages Microcontroller Technology in conjunction with various sensors and a Wi-Fi module to generate live data feeds accessible on the Web.

TABLE OF CONTENTS

1. INTRODUCTION

1.1. PROJECT OVERVIEW

The advent of Internet of Things (IoT) technology in agriculture marks a profound transformation in farming methodologies, ushering in an era of smart agriculture or precision farming. This innovative approach harnesses advanced IoT devices and sensors to gather real-time data, empowering farmers with valuable insights to optimize their operations for enhanced efficiency and productivity.

A key advantage of IoT in agriculture lies in its ability to provide farmers unprecedented visibility into various aspects of their farming environment. By deploying sensors to monitor parameters like soil moisture, temperature, and humidity, farmers gain comprehensive insights into field conditions. This data-driven approach facilitates precise irrigation scheduling, ensuring crops receive optimal water levels. Additionally, early detection of plant diseases or pests enables proactive measures to safeguard crops and minimize losses.

Furthermore, IoT enables remote monitoring and control of agricultural equipment, reducing manual intervention and labor-intensive tasks. Automated systems such as robotic harvesters and drones equipped with sensors and cameras perform tasks like crop monitoring and precision spraying with remarkable accuracy and efficiency, saving time and labor costs. Beyond operational efficiency, IoT promotes sustainable farming practices and environmental stewardship. Precise resource management minimizes waste and reduces the environmental footprint of farming operations. IoT facilitates data-driven decision-making and predictive analytics, empowering farmers to optimize practices and maximize profitability. Analyzing historical data and real-time sensor readings enables farmers to identify trends, predict yields, and optimize planting schedules. This predictive approach minimizes risks associated with weather fluctuations and market volatility. By leveraging data and connectivity, farmers can overcome challenges, increase productivity, and ensure food security for a growing global population. As IoT continues to evolve, its potential to revolutionize agriculture and shape the future of farming remains limitless.

2. SYSTEM ANALYSIS

System analysis is the overall analysis of the system before implementation and for arriving at a precise solution.

2.1. EXISTING SYSTEM

The existing smart IoT-based agriculture system currently in use incorporates a network of interconnected devices and sensors deployed across agricultural fields. These devices collect real-time data on critical environmental parameters such as temperature and soil moisture levels. The data collected by the sensors is transmitted to a central server where it is processed and analyzed using data analytics techniques. Farmers can access the analyzed data through a web-based interface or a mobile application, allowing them to monitor agricultural conditions remotely and make informed decisions about irrigation scheduling, fertilization, and pest management.

2.2. PROPOSED SYSTEM

The proposed IoT-based Smart Agriculture System builds upon the existing system by integrating advanced features and technologies to further enhance its capabilities and effectiveness. In the proposed system, in addition to monitoring temperature and soil moisture levels, Machine Learning models are employed to predict suitable crop yields based on historical data and current environmental conditions. This predictive capability empowers farmers to make proactive decisions about crop planning and resource allocation, ultimately leading to improved yield and quality of produce.

Furthermore, the proposed system utilizes Microcontroller Technology along with various sensors and a Wi-Fi module to generate live data feeds accessible on a web application. This enhances the accessibility and usability of the system, allowing farmers to easily access real-time data from anywhere with an internet connection.

3. SYSTEM CONFIGURATIONS

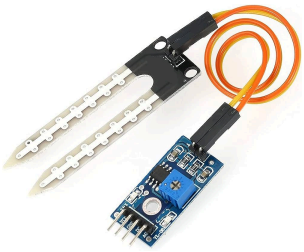
3.1. HARDWARE SPECIFICATION

Processor	:	11th Gen Intel Core i5
Hard disk	:	256 GB
RAM	:	8 GB
Keyboard	:	104 Standard keys
Mouse	:	Optimal Mouse

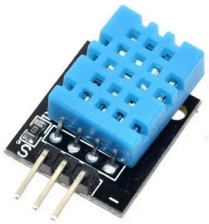
COMPONENTS

Microcontroller (ESP8266 NodeMcu): The ESP8266 NodeMCU is a versatile microcontroller board with integrated Wi-Fi capabilities, making it ideal for IoT projects requiring wireless connectivity. Its small form factor and GPIO pins enable easy integration with various sensors and actuators, facilitating rapid development of smart devices and applications.

I2C LCD Display: It is a type of liquid crystal display (LCD) that communicates with other devices using the I2C (Inter-Integrated Circuit) protocol. It consists of a display module connected to a microcontroller via I2C communication lines, allowing for easy integration into projects with minimal wiring.

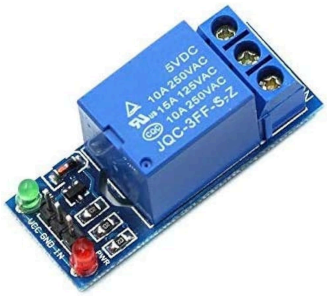


Soil Moisture Sensor: A soil moisture sensor is a device that measures the moisture content in soil, providing crucial data for efficient irrigation and plant health management in agricultural and gardening applications. These sensors typically employ conductivity or capacitance principles to detect soil moisture levels accurately and are commonly used in conjunction with microcontrollers for automated watering systems.



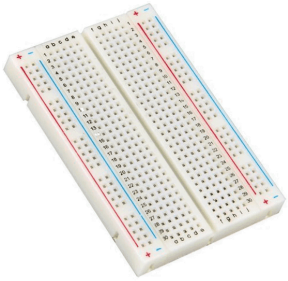
Temperature and Humidity (DHT11 Sensor): The DHT11 is a basic, ultra-low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the air surrounding the plant and sends out a digital signal on the data pin.

Motion (PIR Sensor): A Passive Infrared (PIR) sensor detects motion by measuring changes in infrared radiation emitted by objects in its field of view. It is commonly used in security systems, lighting controls, and automation applications to detect human movement. PIR sensors are characterized by their low cost, simple operation, and ability to operate effectively in various lighting conditions.



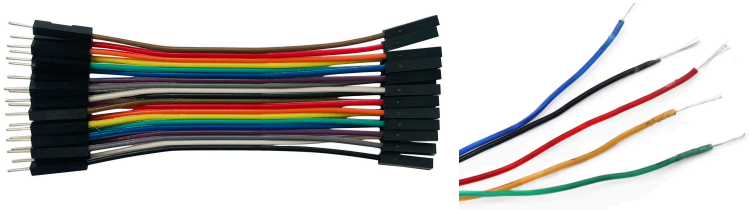
Relay Module (5V): It is an electromechanical switch that is activated by applying a 5-volt signal to its coil. It is used to control higher voltage circuits with lower voltage signals. Relays provide isolation between the control signal and the load, ensuring safety and reliability in circuitry.

Water Pump (DC Motor): A DC motor water pump is a device designed to move water efficiently using a direct current (DC) motor. It consists of a motor connected to an impeller, which creates suction to draw water in and then propels it through an outlet—commonly used in applications such as irrigation, hydroponics, aquariums, and cooling systems due to their compact size, low power consumption, and versatility.



Breadboard: It is a solderless prototyping board used to create and test electronic circuits. It consists of a grid of holes into which electronic components can be inserted and interconnected using jumper wires. Breadboards are commonly used for rapid circuit prototyping and experimentation due to their versatility and ease of use.

Battery: A battery is a portable energy storage device that converts chemical energy into electrical energy. It consists of one or more electrochemical cells, which generate a flow of electrons through a circuit when connected to an external load.



Wires (Jumper wires and Straight wires): They are used to establish electrical connections between different elements on a breadboard or a circuit board. Jumper wires are typically flexible with connectors at both ends, facilitating easy insertion into breadboard holes, while straight wires are rigid and can be soldered onto PCBs or inserted directly into terminals. Both types of wires help create circuits, enabling the flow of electricity between components for proper functionality.

3.2. SOFTWARE SPECIFICATION

Development Environment	**:**	Windows 11
Third party tool	**:**	Arduino Software, Blynk Application, Visual Studio
Coding Language	**:**	C++, Python

4. SOFTWARE DESCRIPTION

4.1. ARDUINO SOFTWARE:

The Arduino Software (IDE) is a user-friendly integrated development environment designed for programming Arduino boards. It provides a simple and intuitive interface for writing, compiling, and uploading code to Arduino microcontrollers. With a range of built-in libraries and examples, the Arduino IDE caters to beginners and advanced users alike, making it accessible for projects spanning from simple blinking LEDs to complex robotics and IoT applications. Additionally, its open-source nature allows for extensive customization and community-driven support, fostering a vibrant ecosystem for hardware prototyping and experimentation.

One of its standout functionalities is the library manager, which simplifies the installation and management of libraries. These libraries, repositories of pre-written code, extend the capabilities of Arduino projects significantly. Moreover, the IDE integrates a serial monitor, facilitating real-time communication with Arduino boards, crucial for debugging and monitoring project behavior.

While not as intricate as dedicated version control systems, the Arduino IDE provides basic integration with tools like Git, aiding code management. Users can also customize the environment to their preferences, adjusting settings and utilizing plugins to expand functionality beyond the default offerings. Additionally, debugging tools like breakpoints and variable inspection support users in identifying and resolving issues within their code.

4.2. BLYNK APPLICATION:

The Blynk application is a versatile platform that enables users to easily create IoT projects and control connected devices using a smartphone or tablet. With a simple drag-and-drop interface, users can design custom dashboards to monitor sensor data, control actuators, and receive

notifications remotely. Blynk supports a wide range of hardware platforms and communication protocols, making it suitable for various IoT applications, including home automation, smart agriculture, and industrial monitoring. Its cloud-based infrastructure ensures seamless connectivity and allows users to access their projects from anywhere in the world. Overall, Blynk provides an intuitive solution for building interactive and connected IoT systems without the need for extensive coding knowledge.

4.3. MACHINE LEARNING:

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Python has Prebuilt Libraries like Numpy for scientific computation, Scipy for advanced computing, and Pybrain for machine learning (Python Machine Learning) making it one of the best languages For Artificial Intelligence (AI).

Python developers around the world provide comprehensive support and assistance via forums and tutorials making the job of the coder easier than any other popular languages. Python is platform-independent and is hence one of the most flexible and popular choices for use across different platforms and technologies with the least tweaks in basic coding.

4.3.1. PYTHON LIBRARIES:

ML libraries are available in many programming languages, but Python being the most user-friendly and easy-to-manage language, and having a large developer community, is best suited for machine learning purposes and that's why many ML libraries are being written in Python.

Pandas:

Pandas are very useful and very fast with large datasets. Its speed exceeds that of Numpy when the records are more than 50k. It is the best library when it comes to data cleaning because it provides interactiveness like Excel and speed like NumPy. As known the most significant part of data analysis and Machine Learning is the data cleaning, processing, and analyzing where Pandas helps very effectively.

Scikit-Learn:

Scikit-learn is mostly focused on various data modeling concepts like regression, classification, clustering, model selections, etc. The library is written on the top of Numpy, Scipy, and Matplotlib. It is an open-source and commercially usable library that is also very easy to understand.

Regression models such as Linear Regression, Ridge Regression, Lasso Regression, KNN Model, Decision Tree, SVM, and Random Forest were used from this library.

4.4. TKINTER:

Tkinter is a standard GUI (Graphical User Interface) toolkit for Python, providing a set of tools to create desktop applications with a graphical interface. It is based on the Tk GUI toolkit and allows developers to design and build user-friendly applications using widgets like buttons, labels, text boxes, and more. Tkinter is included with most Python installations, making it readily available for developers to create cross-platform desktop applications with ease

5. SYSTEM DESIGN

5.1. SYSTEM FLOW

The proposed architecture for the system is as follows:

1. Connect to the ESP8266 Microcontroller
2. Measure the sensor values
3. Check the moisture levels
4. If the level is below the desired threshold, turn ON the water pump
   - 4.1. using manual switch
   - 4.2. using the toggle button in Blynk App.
5. Live Data shared in Blynk App

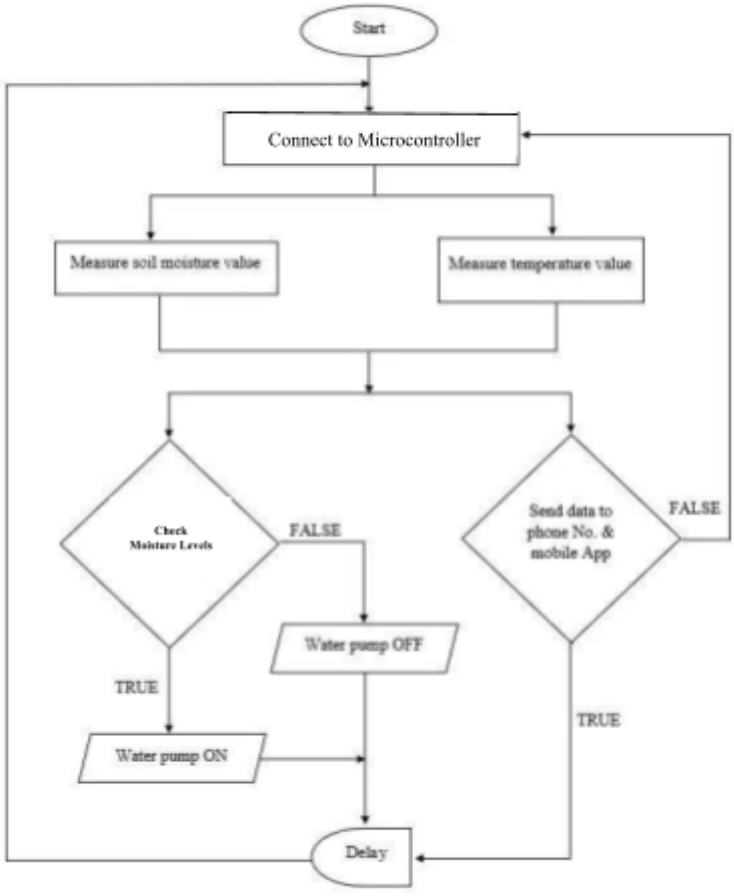


Fig 1: System Flow Diagram

5.2. CIRCUIT DIAGRAM

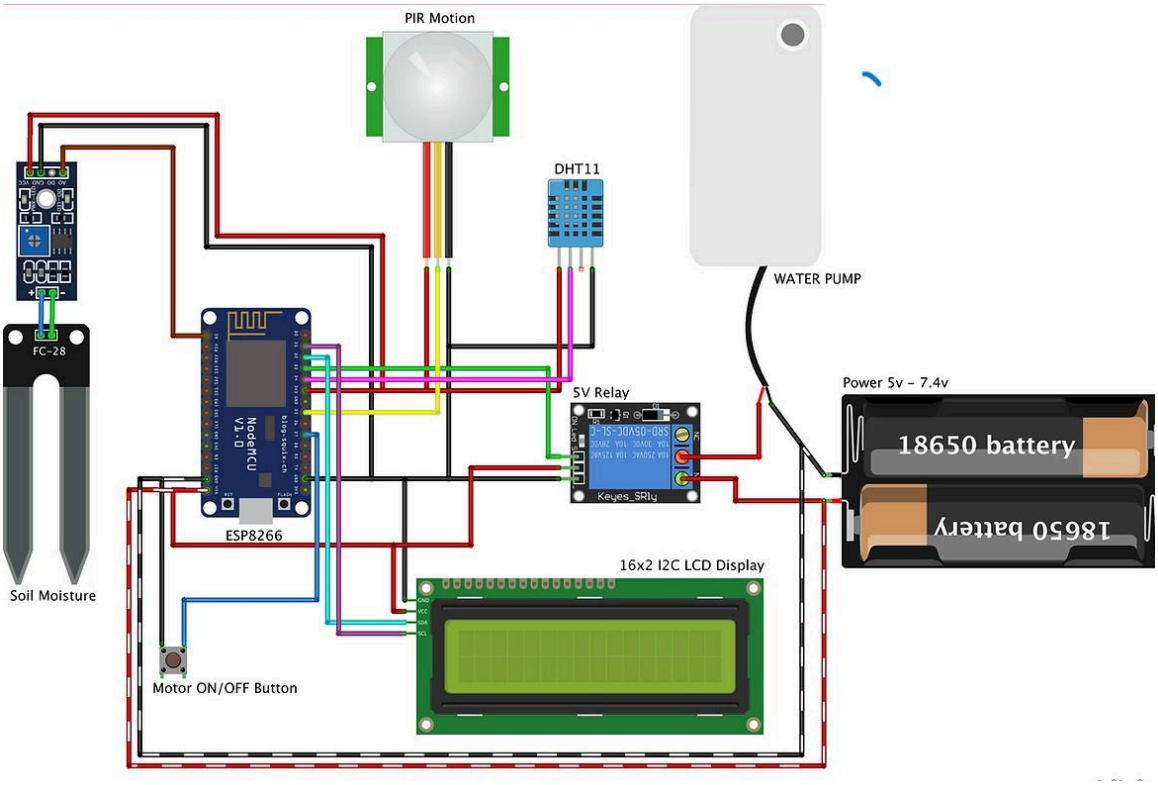


Fig 2: Circuit Diagram

The system integrates various components including sensors (DHT11 for temperature and humidity, soil moisture sensor, PIR motion sensor), a relay for controlling water pumps, and a push-button for manual control. These hardware elements are connected to an ESP8266 microcontroller, which facilitates communication with the Blynk cloud platform over Wi-Fi. The ESP8266 receives sensor data, processes it, and sends relevant information to the Blynk app for remote monitoring and control. Additionally, the system incorporates an LCD to provide real-time feedback on environmental conditions and system status.

5.3. RECOMMENDATION SYSTEM

The architecture for the recommendation system is as follows

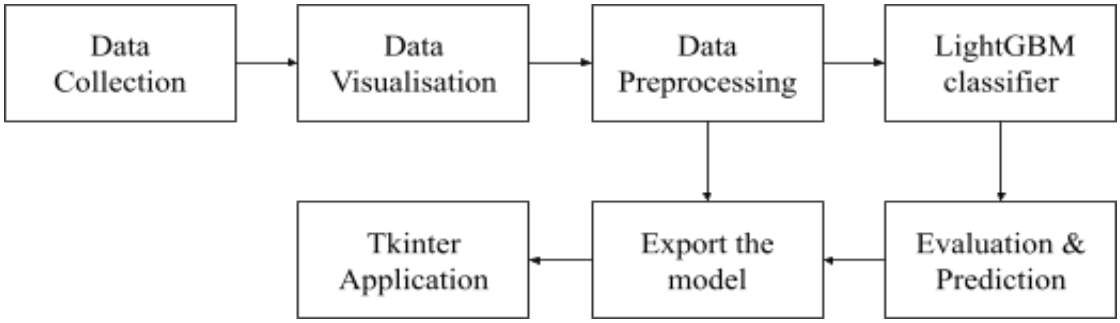


Fig 3: Recommendation System Architecture

A comprehensive analysis of agricultural data has been conducted, leveraging a variety of visualization techniques, data preprocessing methods, and machine learning models. The workflow encompasses data exploration through visualization, including the analysis of soil nutrient content, NPK ratios for crops and fruits, and correlations between features. Following data exploration, preprocessing steps involve splitting the dataset into independent and target variables, as well as into training and testing sets. Subsequently, a LightGBM classifier model is built and trained on the training data, with evaluation conducted on the test set to assess model performance and identify potential overfitting. Further analysis includes the construction of a confusion matrix and the generation of a classification report to gauge the model's accuracy and effectiveness. Finally, the trained model enables predictions for new data points, facilitating crop recommendations based on specific feature values.

6. SYSTEM IMPLEMENTATION

6.1. IOT WORKFLOW

The provided code is for an IoT-based smart agriculture monitoring system using the Blynk platform. Here's a breakdown of the workflow:

1. Library and Configuration Setup:

- Include necessary library files for Blynk, WiFi connection, DHT sensor, and LCD display.
- Define Blynk authentication token, WiFi SSID, and password.
- Initialize the LCD and DHT sensor.

2. Pin Definitions:

- Define the pins to which various components are connected, such as soil moisture sensor, PIR motion sensor, relay, and push button.

3. Variable Initialization:

- Initialize variables to track the state of the relay, push button, and PIR sensor toggle value.

4. Setup Function:

- Initialize serial communication for debugging purposes.
- Initialize the LCD, set its backlight on, and display an initialization message.
- Set pin modes for components like the PIR sensor, relay, and push button.
- Begin Blynk connection with the provided authentication token, WiFi credentials, and Blynk cloud server.
- Set up Blynk timer intervals to periodically execute specific functions for reading sensor values and checking the physical button state.

5. Loop Function:

- Check the state of the PIR sensor toggle value. If motion detection is enabled, call the PIRsensor() function to monitor motion.

- Update the LCD to indicate the status of motion detection and water pump.
- Run Blynk and timer functions to maintain communication with the Blynk server and execute scheduled tasks.

6. Sensor Reading Functions:

- DHT11 Sensor: Read temperature and humidity values from the DHT11 sensor, send the data to Blynk, and display it on the LCD.
- SoilMoisture Sensor: Read soil moisture values from the soil moisture sensor, send the data to Blynk, and display it on the LCD.
- PIR Sensor: Read motion detection values from the PIR sensor, log events to Blynk, and control the LED widget on the Blynk app.

7. Blynk Functions:

- Update the PIR sensor toggle value based on the Blynk app widget state.
- Synchronize the virtual button state with the relay state when Blynk connects.
- Update the relay state based on the virtual button state from the Blynk app.

8. Physical Button Handling:

Monitor the state of the physical push button and toggle the relay state accordingly. Update the virtual button state on the Blynk app.

This workflow enables the monitoring of temperature, humidity, soil moisture, and motion detection in agricultural environments using IoT devices and the Blynk platform. Additionally, it provides control over a water pump relay via both a physical button and the Blynk app.

6.2. RECOMMENDATION SYSTEM WORKFLOW

1. Data Import and Overview:

- Import necessary libraries for data manipulation and visualization: pandas, numpy, matplotlib, seaborn, and plotly.
- Load the dataset from a CSV file named "Crop_recommendation.csv" into a data frame named `cropdf`.
- Examine the structure of the dataset by displaying the first few rows to understand its columns and values.
- Check for any missing values in the dataset to ensure data integrity.

	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

Fig 1: Dataset Preview

2. Data Visualization:

- Perform various visualizations to gain insights into the dataset.
- Create bar charts, scatter plots, and pie charts to visualize different aspects of the data, such as nitrogen, phosphorus, and potassium content in the soil for different crops.
- Compare NPK ratios for different crops and fruits to understand their nutritional requirements.
- Visualize the correlation between different features using a heatmap to identify any relationships between them.

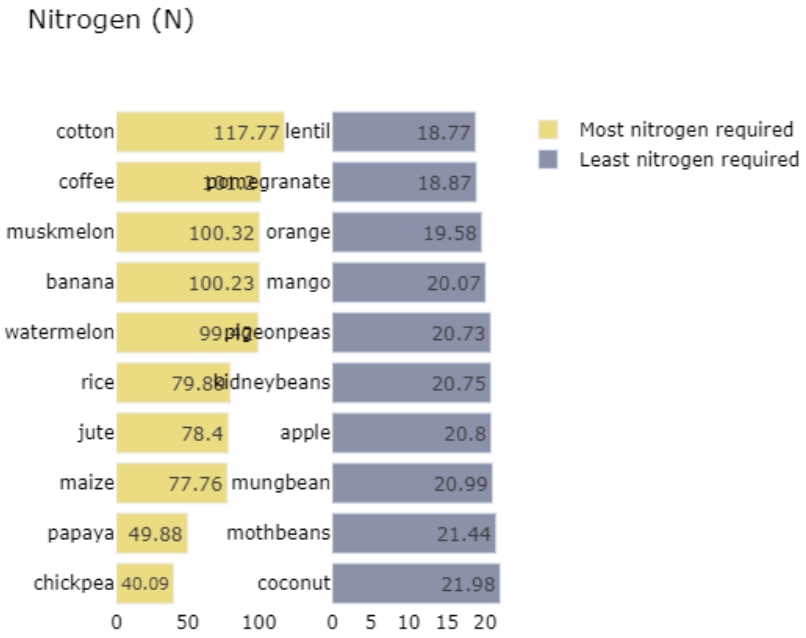


Fig 2: Nitrogen Analysis

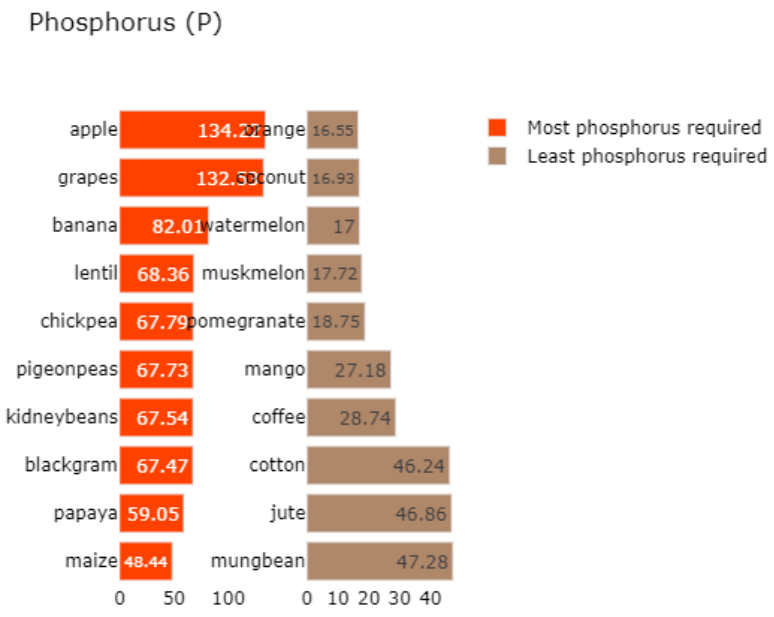


Fig 3: Phosphorus Analysis

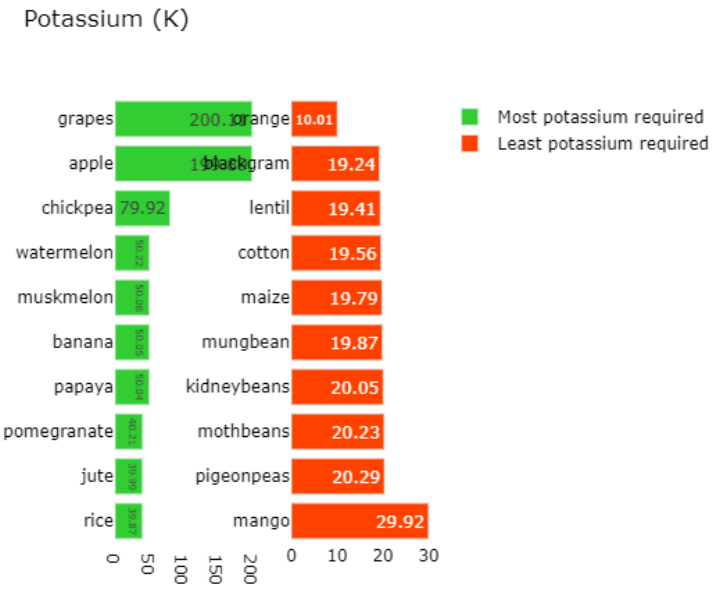


Fig 4: Potassium Analysis

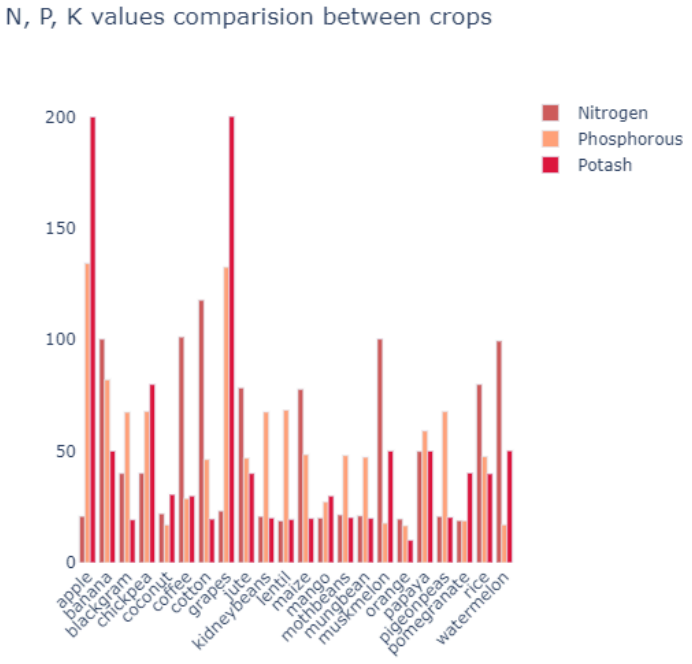


Fig 5: Comparison of N, P, and K values between crops

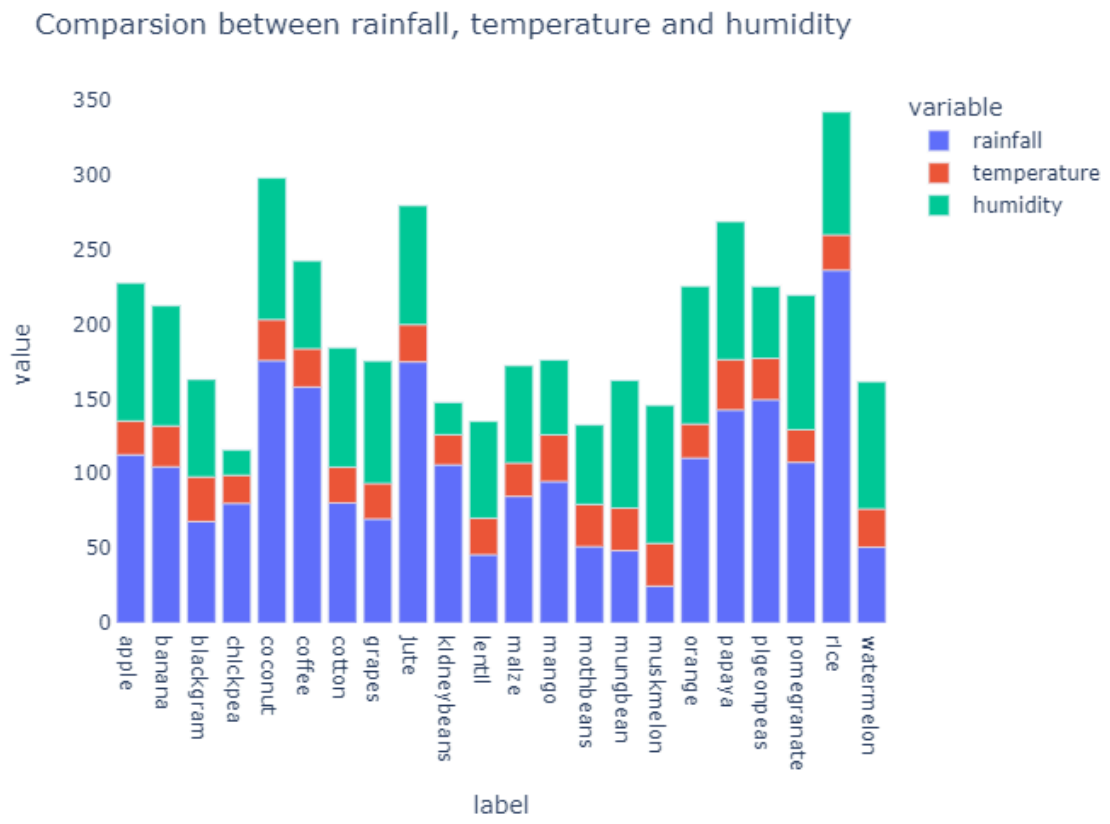


Fig 6: Comparison between rainfall, temperature, and humidity

3. Data Preprocessing:

- Split the dataset into independent variables (`X`), which include features like nitrogen, phosphorus, potassium, temperature, humidity, pH, and rainfall, and the target variable (`y`), which represents the crop label.
- Further, split the data into training and testing sets using the `train_test_split` function to evaluate the model's performance.

4. Model Building and Training:

- Initialize the LightGBM classifier, a gradient-boosting framework known for efficiency and performance on large datasets.
- Train the classifier on the training dataset using the `fit` method.

4.1. Light Gradient Boosting Machine (LightGBM):

LightGBM, short for Light Gradient Boosting Machine, is a powerful machine learning algorithm known for its efficiency and effectiveness, particularly in handling large-scale datasets. Developed by Microsoft, LightGBM is based on the gradient boosting framework, which sequentially builds an ensemble of weak learners (decision trees) to make predictions.

What sets LightGBM apart is its novel implementation of gradient boosting, employing a technique called Gradient-Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). These optimizations significantly reduce memory usage and training time while maintaining high accuracy, making LightGBM well-suited for both classification and regression tasks.

One of the key advantages of LightGBM is its ability to handle categorical features directly, eliminating the need for pre-processing such as one-hot encoding. This feature simplifies the workflow and can lead to more efficient models, especially in scenarios with a large number of categorical variables. Moreover, LightGBM supports parallel and distributed computing, allowing for efficient training on multi-core CPUs and distributed computing frameworks like Apache Spark. This scalability makes LightGBM suitable for processing vast amounts of data in real-world applications.

In terms of performance, LightGBM often outperforms other popular gradient-boosting implementations in terms of training speed and predictive accuracy. Its flexibility in hyperparameter tuning also contributes to its widespread adoption in competitions and real-world machine-learning projects.

5. Model Evaluation:

- Make predictions on the test data using the trained model.
- Calculate the accuracy of the predictions using the `accuracy_score` function.
- Compare the accuracy of the model on both the training and testing datasets to check for overfitting or underfitting.

LightGBM Model accuracy score	0.9894
Training-set accuracy score	1.0000
Training set score	1.0000
Test set score	0.9894

Table 1: Accuracy Score

6. Confusion Matrix and Classification Report:

- Construct a confusion matrix, which provides a summary of correct and incorrect predictions.
- Generate a classification report, including metrics such as precision, recall, F1-score, and support for each class, providing a detailed understanding of the model's performance.

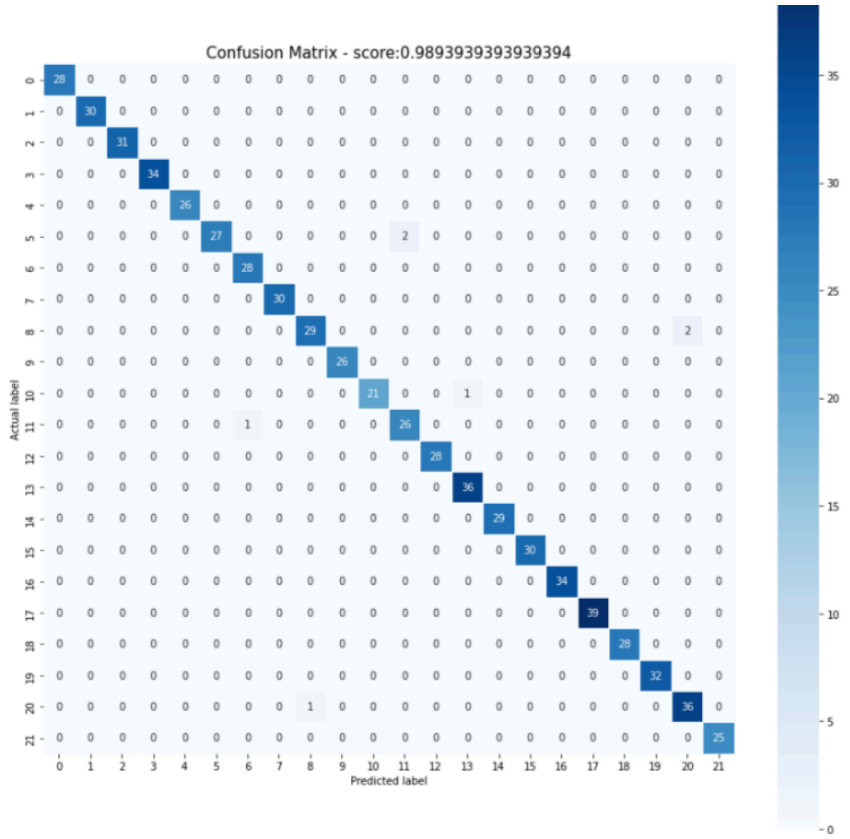


Fig 7: Confusion Matrix

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	28
banana	1.00	1.00	1.00	30
blackgram	1.00	1.00	1.00	31
chickpea	1.00	1.00	1.00	34
coconut	1.00	1.00	1.00	26
coffee	1.00	0.93	0.96	29
cotton	0.97	1.00	0.98	28
grapes	1.00	1.00	1.00	30
jute	0.97	0.94	0.95	31
kidneybeans	1.00	1.00	1.00	26
lentil	1.00	0.95	0.98	22
maize	0.93	0.96	0.95	27
mango	1.00	1.00	1.00	28
mothbeans	0.97	1.00	0.99	36
mungbean	1.00	1.00	1.00	29
muskmelon	1.00	1.00	1.00	30
orange	1.00	1.00	1.00	34
papaya	1.00	1.00	1.00	39
pigeonpeas	1.00	1.00	1.00	28
pomegranate	1.00	1.00	1.00	32
rice	0.95	0.97	0.96	37
watermelon	1.00	1.00	1.00	25
accuracy			0.99	660
macro avg	0.99	0.99	0.99	660
weighted avg	0.99	0.99	0.99	660

Fig 8: Classification Report

7. Prediction:

- Demonstrate how to use the trained model to make predictions for new data points.
- Input specific feature values for a new data point (e.g., temperature, humidity, pH, and soil Moisture) and use the model to predict the corresponding crop label.

This workflow covers the entire process of data analysis, modeling, evaluation, and prediction, providing valuable insights and recommendations for crop selection based on various soil and environmental factors.

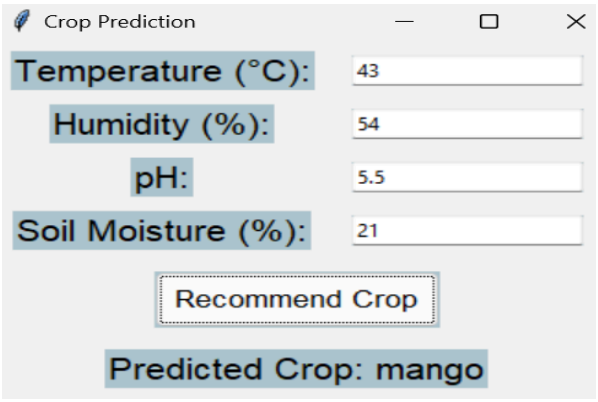


Fig 9: Output

7. SCOPE FOR FUTURE ENHANCEMENT

To unify the IoT application with the crop recommendation system, we propose leveraging live sensor data as input for real-time crop recommendations via an Android application. This integration offers an opportunity to enhance field monitoring efficiency by incorporating additional sensors like pH meters, water level sensors, and rain sensors. These enhancements aim to provide more comprehensive and accurate data for optimizing crop selection and management practices. Furthermore, future iterations could explore incorporating machine learning algorithms to analyze sensor data trends and provide proactive recommendations for crop cultivation and resource allocation.

8. CONCLUSION

In conclusion, the integration of an IoT application with a crop recommendation system presents a promising avenue for enhancing agricultural practices. By leveraging live sensor data and advanced algorithms, farmers can receive real-time recommendations tailored to their specific field conditions. The addition of sensors such as pH meters, water level sensors, and rain sensors further augments the system's capability to provide comprehensive monitoring and insights. This project not only offers immediate benefits in optimizing crop selection and management but also lays the foundation for future enhancements, including the implementation of machine learning algorithms for proactive decision-making in agriculture. Overall, the convergence of IoT technology and crop recommendation systems holds great potential to revolutionize modern farming practices for increased efficiency and sustainability.

9. BIBLIOGRAPHY

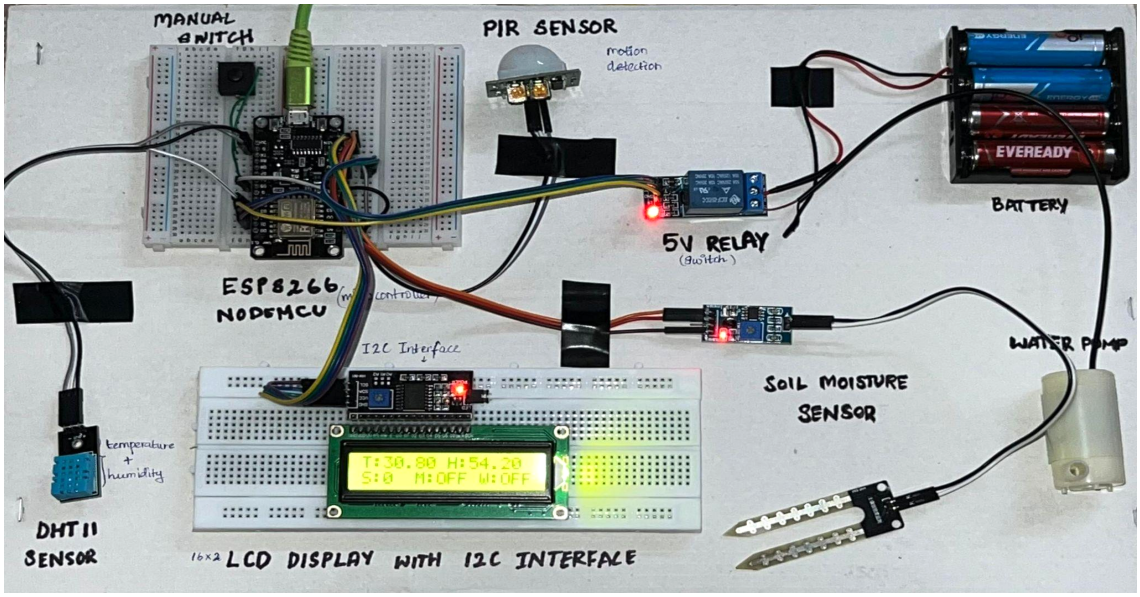
REFERENCES:

1. A. Pravin, T. Prem Jacob and P. Asha (2018) ‘Enhancement of plant monitoring using IoT’, International Journal of Engineering and Technology, Volume 7 (3.27).
2. Monirul Islam Pavel, Syed Mohammed Kamruzzaman, Sadman Sakib Hasan, Saifur Rahman Sabuj (2019) ‘An IoT-based plant health monitoring system implementing using Image Processing’, IEEE 4th International Conference on Computer and Communication systems.
3. Cuelogic, “Role of Python in Artificial Intelligence (AI)”, June 23, 2016. https://www.cuelogic.com/blog/role-of-python-in-artificial-intelligence
4. “Python Libraries for Machine Learning”, Zenesys, July 30, 2021. https://www.zenesys.com/blog/top-10-python-libraries-for-machine-learning
5. Ariyaratne, U. & Yasaswin, Diyon & Deelaka, L. & Herath, H.. (2022). IoT Smart Plant Monitoring, Watering, and Security System.
6. S. M. PANDE, P. K. RAMESH, A. ANMOL, B. R. AISHWARYA, K. ROHILLA and K. SHAURYA, "Crop Recommender System Using Machine Learning Approach," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2021, pp. 1066-1071, doi: 10.1109/ICCMC51019.2021.9418351.

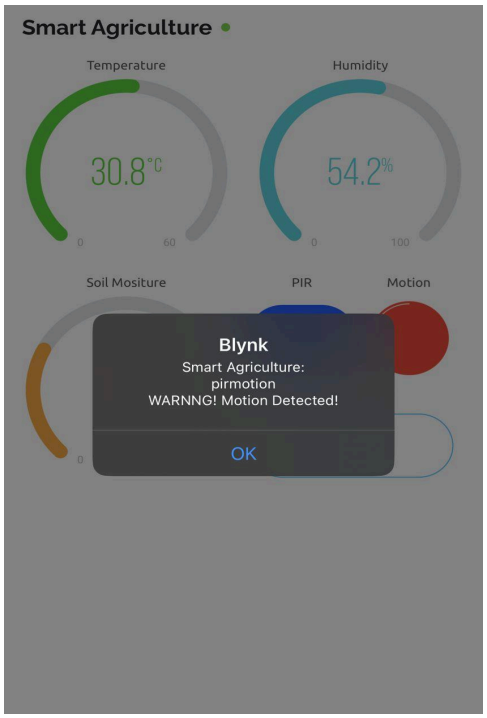
10. APPENDIX

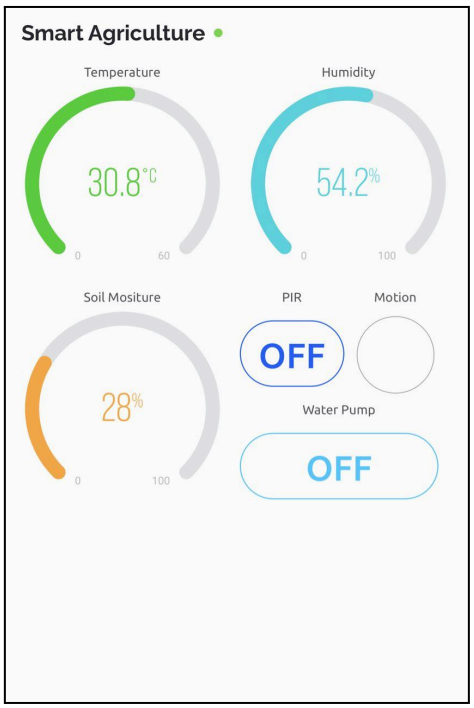
A. SCREENSHOTS:

IOT Working System:



Blynk App:





B. SAMPLE CODE

Arduino IDE(C++):

```
// Blynk IOT Smart Agriculture Monitoring System

/* Connections
Relay. D3
Btn.   D7
Soil.  A0
PIR.   D5
SDA.   D2
SCL.   D1
Temp.  D4
*/

//Include the library files
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPL3QrM6R0ym"
#define BLYNK_TEMPLATE_NAME "Smart Agriculture"
#define BLYNK_AUTH_TOKEN "x_zCFUSrb8Henyq6Ih1PWQgq07XEeRnB"
#include <LiquidCrystal_I2C.h>
#include <ESP8266WiFi.h>
#include <Wire.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>
#include <math.h>


char auth[] = "x_zCFUSrb8Henyq6Ih1PWQgq07XEeRnB";  //Blynk Auth token
```

```
char ssid[] = "URNANBAN";  //WIFI SSID
char pass[] = "zamil@321";  //WIFI Password

// Initialize the LCD display
LiquidCrystal_I2C lcd(0x27, 16, 2);

// DHT sensor setup
DHT dht(D4, DHT11); // DHT11 Temperature Sensor

// Component pins
#define soil A0     // Soil Moisture Sensor
#define PIR D5      // PIR Motion Sensor
#define RELAY_PIN_1 D3   // Relay
#define PUSH_BUTTON_1 D7   // Button
#define VPIN_BUTTON_1 V12

// Variables
int relay1State = HIGH;
int pushButton1State = LOW;
int PIR_ToggleValue;

// Initialize Blynk timer
BlynkTimer timer;

void setup() {
  Serial.begin(9600);
  lcd.begin();
  lcd.backlight();
  pinMode(PIR, INPUT);

  pinMode(RELAY_PIN_1, OUTPUT);
```

```
  digitalWrite(RELAY_PIN_1, relay1State);

  pinMode(PUSH_BUTTON_1, INPUT_PULLUP);

  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass, "blynk.cloud", 80);
  dht.begin();

  lcd.setCursor(0, 0);
  lcd.print("  Initializing  ");
  for (int a = 5; a <= 10; a++) {
    lcd.setCursor(a, 1);
    lcd.print(".");
    delay(500);
  }
  lcd.clear();
  lcd.setCursor(11, 1);
  lcd.print("W:OFF");

  // Call the functions
  timer.setInterval(100L, soilMoistureSensor);
  timer.setInterval(100L, DHT11sensor);
  timer.setInterval(500L, checkPhysicalButton);
}

void loop() {
  if (PIR_ToggleValue == 1) {
    lcd.setCursor(5, 1);
    lcd.print("M:ON ");
    PIRsensor();
  } else {
    lcd.setCursor(5, 1);
```

Visual Studio (Python):

```
import tkinter as tk
from tkinter import ttk
import joblib
# Load the pre-trained model
model = joblib.load('crop_prediction_model.pkl')
# Initialize Tkinter window
root = tk.Tk()
root.title("Crop Prediction")
root.geometry("340x255")

# Function to predict crop and update UI
def predict_crop():
    # Fetching data from Tkinter entry fields
    temp = float(temp_entry.get())
    humidity = float(humidity_entry.get())
    ph = float(ph_entry.get())
    soil_moisture = float(soil_moisture_entry.get())

    # Creating prediction data
    predict_data = [[temp, humidity, ph]]

    # Predicting crop
    predicted_crop = model.predict(predict_data)[0]

    # Display predicted crop
    result_label.config(text="Predicted Crop: " + predicted_crop)

style = ttk.Style()
style.configure('TLabel', background='#AEC6CF', font=('Arial', 15))
style.configure('TEntry', background='#AEC6CF', font=('Arial', 20))
```

```
# Add labels and entry fields for input parameters
temp_label = ttk.Label(root, text="Temperature (°C):")
temp_label.grid(row=0, column=0, padx=10, pady=5)
temp_entry = ttk.Entry(root)
temp_entry.grid(row=0, column=1, padx=10, pady=5)
humidity_label = ttk.Label(root, text="Humidity (%):")
humidity_label.grid(row=1, column=0, padx=10, pady=5)
humidity_entry = ttk.Entry(root)
humidity_entry.grid(row=1, column=1, padx=10, pady=5)
ph_label = ttk.Label(root, text="pH:")
ph_label.grid(row=2, column=0, padx=10, pady=5)
ph_entry = ttk.Entry(root)
ph_entry.grid(row=2, column=1, padx=10, pady=5)
soil_moisture_label = ttk.Label(root, text="Soil Moisture (%):")
soil_moisture_label.grid(row=4, column=0, padx=10, pady=5)
soil_moisture_entry = ttk.Entry(root)
soil_moisture_entry.grid(row=4, column=1, padx=10, pady=5)

# Add button to predict crop
predict_button = ttk.Button(root, text="Recommend Crop", command=predict_crop, style='Custom.TButton')
predict_button.grid(row=5, columnspan=2, padx=10, pady=10)
# Add label to display predicted crop
result_label = ttk.Label(root, text="", background='#AEC6CF', font=('Arial', 15))
result_label.grid(row=6, columnspan=2, padx=10, pady=5)
# Configure custom style for the button
style.configure('Custom.TButton', background='#AEC6CF', font=('Arial', 13), padding=6, width=12)
root.mainloop()
```