

Plant Moisture Detector Using Python

Mr. Dibya Ranjan Giri

Student, Dept. of CSE,
GIFT Autonomous, Bhubaneswar

Mr. Shibangi Dhal

Student, Dept. of CSE,
GIFT Autonomous, Bhubaneswar

Prof. Rumana Hasinullah Shaikh

Assistant Professor, Dept. of CSE,
GIFT Autonomous, Bhubaneswar

Abstract— The Plant Moisture Detector using Python is an intelligent soil monitoring system developed to automate plant care and improve water management efficiency. The system combines soil moisture sensing hardware with Python-based software to monitor real-time moisture conditions and provide accurate irrigation recommendations. Traditional watering methods often depend on manual observation and estimation, which can lead to overwatering, underwatering, and unnecessary water wastage. This project addresses these issues by implementing a smart monitoring system capable of collecting, processing, storing, and visualizing soil moisture data.

The proposed system uses a soil moisture sensor connected to a microcontroller such as Arduino or NodeMCU. The sensor continuously measures the moisture content present in the soil and transmits the data to a Python application through serial communication. The Python program processes the incoming sensor values, converts them into readable moisture percentages, and displays the information through a graphical user interface. The system also supports threshold-based alerts that notify users when the soil becomes too dry or excessively wet.

Additionally, the platform includes data storage and visualization features using SQLite and Matplotlib libraries. Historical moisture readings can be stored, analyzed, and represented through graphs to understand moisture trends over time. The project can further be integrated with automated irrigation systems, IoT-based monitoring platforms, and cloud databases for advanced agricultural applications.

The major objective of this project is to create a cost-effective, user-friendly, and efficient smart farming solution that reduces water wastage, improves plant health, and minimizes manual effort. By integrating sensors, microcontrollers, Python programming, and real-time monitoring techniques, the Plant Moisture Detector demonstrates the practical application of automation and smart agriculture technologies.

The system combines both hardware and software components to create a complete real-time monitoring solution. A soil moisture sensor is inserted into the soil near plant roots to measure the amount of moisture present in the soil. The sensor readings are collected by a microcontroller such as Arduino or NodeMCU, which converts analog signals into digital values. These values are transmitted to a Python application through serial communication for further processing and analysis. The Python backend processes the incoming data, calculates moisture percentages, compares values with predefined thresholds, and determines whether the soil condition is dry, optimal, or excessively wet.

In addition to real-time monitoring, the system includes data storage and visualization capabilities. Moisture readings along with timestamps are stored in a SQLite database, allowing users to maintain historical records and analyze moisture trends over time. Python libraries such as Matplotlib and Pandas are used to generate graphical charts and moisture analysis reports. The

project also supports threshold-based alert systems that notify users whenever irrigation is required. Furthermore, the system can be integrated with relay modules and water pumps for automatic irrigation functionality.

The major objective of this project is to provide a cost-effective, scalable, and user-friendly smart irrigation solution suitable for home gardening, nurseries, greenhouses, and small-scale agriculture. The proposed system demonstrates the practical implementation of automation, embedded systems, data analysis, and smart farming technologies using Python.

Keywords: Plant Moisture Detector, Python, Soil Moisture Sensor, Smart Irrigation, Arduino, IoT, Data Visualization, Automation.

I. INTRODUCTION

Agriculture and gardening play an essential role in human life, food production, and environmental sustainability. Proper irrigation management is one of the most important factors that influences plant growth, crop productivity, and soil quality. In traditional farming and gardening practices, watering decisions are generally based on manual observation and experience. However, these methods are often inaccurate and inefficient, especially under varying environmental conditions.

Improper watering can negatively affect plant health in multiple ways. Overwatering can damage plant roots, reduce oxygen supply in the soil, and encourage fungal diseases. Underwatering, on the other hand, can cause dehydration, reduced crop yield, and poor plant growth. Additionally, manual watering methods lead to unnecessary water wastage, which is a major concern in modern agriculture due to increasing water scarcity.

With the advancement of smart technologies and automation systems, intelligent agricultural solutions are becoming increasingly important. Soil moisture monitoring systems provide real-time information about the water content in soil and help users make accurate irrigation decisions. These systems support sustainable farming practices and improve water conservation.

The Plant Moisture Detector using Python is developed as a smart monitoring system that combines hardware and software technologies for automated soil moisture analysis. The system uses a soil moisture sensor connected to a microcontroller such as Arduino or NodeMCU to measure soil moisture levels continuously. The collected data is transmitted to a Python-based application for processing, storage, visualization, and alert generation.

Python is widely used in automation, Internet of Things (IoT), and data analysis applications due to its simplicity, flexibility, and extensive library support. In this project, Python is used for serial communication, data processing, graphical visualization, database management, and alert generation. The system provides a userfriendly interface where users can monitor current moisture levels, track historical moisture trends, and receive notifications whenever watering is required.

Agriculture plays a critical role in human survival, food production, environmental sustainability, and economic development. One of the most important factors that directly influences agricultural productivity and plant health is irrigation management. Proper watering ensures healthy root development, nutrient absorption, and balanced plant growth. However, in traditional farming and gardening methods, irrigation is generally controlled manually based on visual observation or fixed schedules. Such methods are often inaccurate because soil moisture conditions continuously change depending on weather, temperature, humidity, soil type, and plant requirements.

The proposed system can be used in home gardening, nurseries, greenhouses, and small-scale agricultural applications. It reduces manual effort, improves irrigation efficiency, and encourages smart farming practices. The project also demonstrates the practical integration of sensors, embedded systems, Python programming, and data visualization technologies in real-world applications.

In many situations, farmers and gardeners are unable to determine the exact amount of water required by plants. As a result, plants are often overwatered or underwatered. Overwatering can damage roots, reduce oxygen availability in the soil, encourage fungal growth, and waste significant amounts of water. Underwatering can lead to dehydration, poor crop quality, stunted growth, and reduced agricultural productivity. Water scarcity is becoming a major global concern, making efficient irrigation systems extremely important in modern agriculture.

With advancements in embedded systems, automation, sensors, Internet of Things (IoT), and software technologies, smart irrigation systems have become increasingly popular. Soil moisture monitoring systems help users analyze the real-time water content present in soil and make accurate watering decisions. These systems contribute to water conservation, automation, and sustainable farming practices. Modern smart farming techniques use sensors, microcontrollers, cloud platforms, and data analytics to improve agricultural efficiency and reduce manual labor.

The Plant Moisture Detector using Python is developed as a smart monitoring system that combines hardware components with Python-based software technologies to automate soil moisture analysis. The proposed system uses a soil moisture sensor to measure the water content present in the soil continuously. The sensor is connected to a microcontroller such

as Arduino or NodeMCU, which collects moisture readings and sends them to the Python application through serial communication. The Python program processes the incoming data, calculates moisture percentages, stores readings in a database, and displays graphical analysis using visualization libraries.

Python is selected as the primary programming language because of its simplicity, flexibility, extensive library support, and compatibility with hardware communication systems. Libraries such as pySerial, Tkinter, Matplotlib, Pandas, and SQLite are used to implement real-time monitoring, graphical visualization, database storage, and user interface functionalities. The system also includes threshold-based alert mechanisms that notify users whenever the soil becomes too dry or excessively wet.

The proposed project can be used in home gardens, nurseries, greenhouses, indoor farming environments, and small-scale agricultural applications. It reduces manual effort, improves irrigation efficiency, conserves water resources, and promotes healthy plant growth. The project demonstrates the practical application of smart farming concepts using sensors, microcontrollers, Python programming, and data visualization technologies.

II. EXISTING APPROACHES

Traditional irrigation systems mainly rely on manual inspection and scheduled watering techniques. Farmers and gardeners generally estimate soil moisture conditions through visual observation or fixed watering intervals. Although these methods are simple and inexpensive, they often lead to inaccurate watering decisions and inefficient resource utilization.

Several existing soil monitoring systems have been developed to improve irrigation management. Basic moisture detection systems use analog sensors connected directly to display units or buzzers. These systems only provide real-time readings and lack advanced functionalities such as data storage, graphical analysis, automation, and remote monitoring.

Modern smart irrigation systems integrate IoT devices, wireless communication modules, and cloud platforms for remote monitoring and automation. These systems use technologies such as Wi-Fi modules, cloud databases, mobile applications, and machine learning algorithms to analyze environmental conditions. While such systems provide advanced features, they are often expensive, complex, and difficult for small-scale users or students to implement.

The rapid growth of smart agriculture and precision farming technologies has significantly increased the importance of automated irrigation and soil monitoring systems. Traditional irrigation methods mainly rely on manual watering schedules and visual inspection of soil conditions. Farmers and gardeners generally estimate the amount of water required by plants based on experience rather than accurate scientific measurements. Although these methods are simple and

inexpensive, they often result in inefficient water usage, poor irrigation management, and inconsistent plant growth.

Several conventional soil moisture monitoring systems have been developed over the years to improve irrigation practices. Basic systems use analog moisture sensors connected directly to LEDs, buzzers, or simple display units. These systems provide only realtime moisture readings and do not support data storage, trend analysis, remote monitoring, or automation features. In many cases, users must manually monitor sensor outputs and operate irrigation systems themselves.

Modern smart irrigation systems integrate Internet of Things (IoT) technologies, wireless communication modules, cloud databases, and mobile applications to provide advanced monitoring capabilities. These systems use Wi-Fi modules such as ESP8266 or GSM modules for transmitting sensor data to cloud servers. Mobile applications allow users to monitor soil conditions remotely and control irrigation systems from smartphones. Some advanced systems also include machine learning algorithms to predict irrigation requirements based on environmental conditions.

Despite these advancements, many existing smart irrigation systems have several limitations. Large-scale IoT-based systems are often expensive, complex to configure, and difficult to maintain. Many systems require stable internet connectivity, cloud subscriptions, and advanced technical knowledge. Small-scale farmers, students, and home gardeners may find such systems difficult to implement due to high installation and maintenance costs.

Another major limitation of traditional and existing systems is the lack of efficient data visualization and historical analysis. Many low-cost systems only display current moisture values without storing data for future reference. Without historical analysis, users cannot study moisture trends, watering patterns, or plant water requirements over time.

The proposed Plant Moisture Detector using Python addresses these challenges by combining sensor-based monitoring, Pythonbased data processing, visualization tools, database management, and threshold-based automation in a single cost-effective solution. The system is designed to be user-friendly, scalable, and suitable for educational projects as well as real-world smart farming applications.

Table-I: Comparison of Traditional Systems and Proposed System

Feature	Traditional Tools	Proposed System
Moisture Monitoring	Manual Observation	Real-Time Sensor Monitoring
		Threshold-Based Intelligent Control
		SQLite Database Storage
		Python GUI Dashboard
	Fixed Schedule	Graphical Visualization
Watering Control	Not Available	Supports Automatic Irrigation
Data Storage	Not Available	
User Interface	Limited	
Visualization	Not Available	
Automation	Not Available	

io n	Ma nu al Wa teri ng
---------	------------------------------------

Existing systems suffer from several limitations:

- Lack of real-time monitoring and automation.
- Inability to store historical moisture data.
- No graphical visualization or trend analysis.
- High implementation cost for advanced IoT systems.
- Limited user-friendly interfaces.
- Poor scalability and flexibility.

The proposed Plant Moisture Detector using Python addresses these issues by integrating sensor-based monitoring, Python-based data processing, database management, visualization tools, and threshold-based alerts within a single cost-effective platform.

III. PROPOSED SYSTEM ARCHITECTURE

The proposed Plant Moisture Detector system follows a layered architecture that combines hardware sensing components with software-based processing and visualization modules. The architecture is designed to provide efficient data collection, real-time monitoring, secure storage, and userfriendly interaction.

The Soil Moisture Sensor Layer continuously measures the water content present in the soil. The sensor operates based on conductivity or capacitance principles and generates analog or digital signals corresponding to the moisture level.

The Microcontroller Layer, implemented using Arduino or NodeMCU, receives sensor signals and converts them into digital values. The microcontroller performs initial processing and transmits the readings to the Python application through serial communication.

The Communication Layer establishes data transfer between the hardware and software components. USB serial communication or wireless communication modules such as ESP8266 can be used for transmitting moisture readings.

The Python Backend Layer is responsible for reading sensor data, processing moisture values, validating readings, applying threshold conditions, and generating alerts. Python libraries such as pySerial, Pandas, and NumPy are used for efficient data handling.

The Database Storage Layer stores moisture readings, timestamps, and plant information using SQLite or MySQL databases. This layer helps maintain historical records for trend analysis and future reference.

The Visualization and User Interface Layer displays moisture levels, status indicators, and graphical charts through Python GUI frameworks such as Tkinter. Visualization libraries like Matplotlib are used to generate moisture trend graphs.

The Alert and Automation Layer compares moisture readings with predefined thresholds. When the soil becomes too dry or too wet, the system triggers notifications or activates an automatic water pump through a relay module.

The system architecture consists of the following major layers:

- Soil Moisture Sensor Layer
- Microcontroller Processing Layer
- Communication Layer
- Python Backend Layer
- Database Storage Layer
- Visualization and User Interface Layer
- Alert and Automation Layer

The Soil Moisture Sensor Layer continuously measures the water content present in the soil. The sensor operates based on conductivity or capacitance principles and generates analog or digital signals corresponding to the moisture level.

The Microcontroller Layer, implemented using Arduino or NodeMCU, receives sensor signals and converts them into digital values. The microcontroller performs initial processing and transmits the readings to the Python application through serial communication.

The Communication Layer establishes data transfer between the hardware and software components. USB serial communication or wireless communication modules such as ESP8266 can be used for transmitting moisture readings.

The Python Backend Layer is responsible for reading sensor data, processing moisture values, validating readings, applying threshold conditions, and generating alerts. Python libraries such as pySerial, Pandas, and NumPy are used for efficient data handling.

The Database Storage Layer stores moisture readings, timestamps, and plant information using SQLite or MySQL

databases. This layer helps maintain historical records for trend analysis and future reference.

The Visualization and User Interface Layer displays moisture levels, status indicators, and graphical charts through Python GUI frameworks such as Tkinter. Visualization libraries like Matplotlib are used to generate moisture trend graphs.

The Alert and Automation Layer compares moisture readings with predefined thresholds. When the soil becomes too dry or too wet, the system triggers notifications or activates an automatic water pump through a relay module.

The Soil Moisture Sensor Layer continuously measures the water content present in the soil. The sensor operates based on conductivity or capacitance principles and generates analog or digital signals corresponding to the moisture level.

The Microcontroller Layer, implemented using Arduino or NodeMCU, receives sensor signals and converts them into digital values. The microcontroller performs initial processing and transmits the readings to the Python application through serial communication.

The Communication Layer establishes data transfer between the hardware and software components. USB serial communication or wireless communication modules such as ESP8266 can be used for transmitting moisture readings.

The Python Backend Layer is responsible for reading sensor data, processing moisture values, validating readings, applying threshold conditions, and generating alerts. Python libraries such as pySerial, Pandas, and NumPy are used for efficient data handling.

The Database Storage Layer stores moisture readings, timestamps, and plant information using SQLite or MySQL databases. This layer helps maintain historical records for trend analysis and future reference.

The Visualization and User Interface Layer displays moisture levels, status indicators, and graphical charts through Python GUI frameworks such as Tkinter. Visualization libraries like Matplotlib are used to generate moisture trend graphs.

The Alert and Automation Layer compares moisture readings with predefined thresholds. When the soil becomes too dry or too wet, the system triggers notifications or activates an automatic water pump through a relay module.

Processing Layer	Arduino NodeMCU	Data acquisition and conversion
Communication Layer	USB / Wi-Fi	Data transmission
Backend Layer	Python	Data processing and logic
Database Layer	SQLite	Data storage and retrieval
Visualization Layer	Tkinter, Matplotlib	Graphical interface and charts
Automation Layer	Relay Module	Automatic irrigation control

IV. METHODOLOGY

The development methodology of the Plant Moisture Detector using Python follows a systematic approach that includes data collection, preprocessing, analysis, visualization, and automation.

1. Data Collection

The soil moisture sensor is inserted into the soil near the plant roots to measure water content continuously. The sensor outputs analog signals corresponding to moisture levels. These signals are received by the microcontroller and transmitted to the Python application using serial communication.

2. Data Preprocessing

The raw sensor readings may contain fluctuations or noise due to environmental conditions. Python-based filtering and averaging techniques are applied to improve data accuracy and stability.

3. Moisture Analysis

The processed moisture values are converted into readable percentages and compared with predefined threshold levels. Based on the threshold conditions, the soil status is categorized as:

- Dry
- Optimal • Wet

4. Data Storage

The system stores moisture readings along with timestamps in a SQLite database. Historical data helps users analyze moisture patterns over time.

5. Visualization

Graphs and charts are generated using Matplotlib to represent moisture trends visually. The GUI dashboard displays current moisture levels, plant status, and historical graphs.

6. Alert Generation

Whenever moisture levels fall below the minimum threshold or exceed the maximum threshold, the system generates alerts through the user interface or buzzer notifications.

Table II: Technology Stack and Functional Operations

Layer	Technology Used	Functional Operation
Sensor Layer	Soil Moisture Sensor	Soil moisture measurement

7. Automation Support

The system can optionally control a water pump using a relay module. Automatic irrigation is triggered when soil moisture falls below the specified threshold.

Workflow of the System

1. Soil moisture sensor detects moisture level.
2. Microcontroller collects sensor data.
3. Data is transferred to Python application.
4. Python processes and analyzes readings.
5. Data is stored in the database.
6. Visualization dashboard displays moisture information.
7. Alerts or irrigation actions are triggered.

V. SYSTEM DESIGN AND IMPLEMENTATION

The implementation of the Plant Moisture Detector integrates hardware and software modules to create a fully functional smart monitoring system.

Hardware Components

The hardware setup includes:

- Soil Moisture Sensor
- Arduino UNO / NodeMCU
- Jumper Wires
- Breadboard
- Relay Module (Optional)
- Water Pump (Optional)
- USB Cable

The soil moisture sensor is connected to the analog input pins of the microcontroller. The microcontroller continuously reads moisture values and transmits them to the Python application.

Software Components

The software implementation is carried out using Python. The major software tools and libraries include:

- Python IDE (VS Code / PyCharm)
- Arduino IDE
- pySerial
- Tkinter
- Matplotlib
- SQLite3
- Pandas

Python Implementation

The Python program performs the following operations:

- Reads sensor data from serial port.
- Converts analog readings into moisture percentages.
- Displays moisture information through GUI.
- Stores readings in SQLite database.
- Generates graphs for moisture analysis.

- Triggers alerts when thresholds are crossed.

Database Design

The SQLite database stores the following information:

Field Name	Description
Plant_ID	Unique plant identifier
Plant_Name	Name of plant
Moisture_Level	Current moisture percentage
Status	Dry / Optimal / Wet
Timestamp	Date and time of reading

User Interface Design

The graphical user interface developed using Tkinter includes:

- Moisture percentage display
- Plant status indicator
- Moisture trend graph
- Alert notifications
- Data history section

VI. RESULT AND ANALYSIS

The Plant Moisture Detector using Python was tested under multiple environmental conditions to evaluate its performance, accuracy, reliability, and efficiency. The testing process included monitoring soil moisture under dry, moderately wet, and highly wet soil conditions. Multiple sensor readings were collected continuously and analyzed using the Python application.

The performance evaluation of the proposed Plant Moisture Detector system demonstrated stable and efficient operation during continuous monitoring. The integration of Python-based data processing, real-time visualization, and threshold-based alert generation improved irrigation accuracy and reduced manual intervention. The system effectively maintained consistent monitoring performance while supporting efficient water conservation and healthy plant growth management.

During experimental testing, the system successfully detected variations in soil moisture levels and generated accurate moisture percentage values. The soil moisture sensor responded effectively to changes in soil water content and transmitted data correctly to the microcontroller. The Arduino successfully converted analog sensor readings into digital values and transferred them to the Python backend through serial communication.

The Python application processed the incoming data efficiently and displayed moisture information through a graphical user interface. The system categorized soil conditions into three major states:

- Dry Soil Condition
- Optimal Soil Condition
- Wet Soil Condition

Threshold values were predefined within the Python application to identify these conditions accurately. Whenever the moisture level dropped below the minimum threshold, the system generated an alert indicating that watering was required. Similarly, if the moisture level exceeded the maximum threshold, the system displayed a warning message indicating excessive moisture.

The graphical visualization module developed using Matplotlib successfully displayed moisture trends over time. Users could monitor real-time moisture variations and analyze historical moisture patterns through line graphs and charts. These graphs helped users understand irrigation frequency and plant water requirements more effectively.

The SQLite database successfully stored moisture readings along with timestamps, allowing long-term data analysis and record management. Historical data could be retrieved efficiently for visualization and reporting purposes.

Performance Analysis Table

Soil Condition	Moisture Percentage	System Response
Dry Soil	10% – 25%	Watering Required
Moderate Soil	40% – 60%	Optimal Condition
Wet Soil	70% – 90%	Excess Water Warning

The system demonstrated several operational advantages during testing:

- Accurate soil moisture detection.
- Real-time data monitoring.
- Reduced water wastage.
- User-friendly graphical interface.
- Efficient database storage.
- Easy integration with automation systems.
- Low hardware and software cost.
- Reliable performance in different soil conditions.

The project also reduced manual effort significantly because users no longer needed to inspect soil conditions repeatedly. Real-time alerts helped improve irrigation management and supported better plant health maintenance.

Although the system performed efficiently, some limitations were identified during testing. Sensor readings occasionally fluctuated due to soil composition, temperature variations, and electrical noise. Proper sensor calibration is therefore important for achieving maximum accuracy. Additionally, wireless

communication and cloud integration were not included in the current implementation and can be added in future versions.

Overall, the experimental results confirmed that the proposed Plant Moisture Detector using Python is an effective, reliable, and cost-efficient smart irrigation solution suitable for home gardening, greenhouse monitoring, educational projects, and small-scale agricultural applications.

The graphical visualization module successfully displayed moisture trends over time. The database storage mechanism preserved historical records efficiently, enabling long-term moisture analysis.

Advantages Observed

- Accurate moisture detection.
- Real-time monitoring capability.
- Reduced water wastage.
- Improved plant health management.
- User-friendly graphical interface.
- Low-cost implementation.

Limitations

- Sensor readings may vary depending on soil type.
- Requires continuous power supply.
- Wireless functionality requires additional modules.

VII. FUTURE SCOPE

The future scope of the Plant Moisture Detector using Python is very promising in the field of smart agriculture and automation. The system can be enhanced by integrating IoT technology for remote monitoring through mobile applications and cloud platforms. Future versions may include wireless communication modules such as Wi-Fi and Bluetooth for realtime data access. Artificial Intelligence and Machine Learning algorithms can also be implemented to predict irrigation requirements based on weather conditions and soil patterns. Solar-powered monitoring systems, automatic irrigation control, and large-scale greenhouse integration can further improve efficiency, reduce water wastage, and support sustainable farming practices in modern agriculture.

Future improvements of the Plant Moisture Detector system can focus on enhancing accuracy, scalability, and automation capabilities. Advanced sensors with higher precision can be integrated to improve moisture detection under different soil conditions. The project can also be connected with weather forecasting APIs to make intelligent irrigation decisions based on rainfall predictions and temperature analysis. Cloud databases and dashboard systems can allow farmers to monitor multiple

plants remotely from any location. Integration with smart home systems and voice assistants can further increase usability. These advancements will help create a fully automated, energy-efficient, and intelligent irrigation system suitable for modern smart farming applications.

The Plant Moisture Detector system can be further enhanced with several advanced features and technologies.

Future improvements include:

- Integration with IoT platforms for remote monitoring.
- Cloud-based data storage and analytics.
- Mobile application support for notifications.
- AI-based predictive irrigation systems.
- Solar-powered monitoring units.
- Wireless sensor networks for large-scale farming.
- Integration with weather forecasting systems.
- Voice-controlled smart irrigation support.

The project has strong potential in smart agriculture, greenhouse monitoring, precision farming, and sustainable irrigation management.

VIII. CONCLUSION

The Plant Moisture Detector using Python is an effective and intelligent solution for monitoring soil moisture and improving irrigation efficiency. The system successfully integrates soil moisture sensing hardware with Python-based software technologies to provide real-time monitoring, graphical visualization, threshold-based alerts, and historical data storage.

The project reduces manual effort, minimizes water wastage, and supports healthy plant growth through accurate moisture analysis. The use of Python libraries, microcontrollers, and sensors demonstrates the practical implementation of automation and smart agriculture concepts.

The developed system is cost-effective, scalable, and suitable for home gardening, nurseries, and small-scale agricultural applications. With future enhancements such as IoT integration, cloud connectivity, and AI-based irrigation control, the project can evolve into a complete smart farming solution.

REFERENCES

1. Rajalakshmi, P., and Devi, S., "Smart Irrigation System Using IoT," International Journal of Engineering Research and Technology, 2021.
2. Sharma, R., "Soil Moisture Monitoring Techniques for Smart Farming," Journal of Agricultural Technology, 2020.
3. Python Software Foundation, "Python Documentation," Available: <https://www.python.org>
4. Arduino Documentation, "Arduino UNO User Guide," Available: <https://www.arduino.cc>
5. Patel, K., "Automation in Agriculture Using Embedded Systems," International Conference on Smart Farming, 2022.
6. Kumar, A., and Singh, R., "IoT Based Soil Monitoring and Smart Irrigation," IEEE Access, 2021.
7. McKinney, W., "Python for Data Analysis," O'Reilly Media, 2018.
8. Hunter, J., "Matplotlib: A 2D Graphics Environment," Computing in Science & Engineering, 2007.
9. SQLite Documentation, "SQLite Database System," Available: <https://www.sqlite.org>
10. Gupta, S., "Smart Farming Technologies and Water Conservation," International Journal of Computer Applications, 2023.