"IoT based Smart Irrigation System"

A Project Report Submitted to

Rajiv Gandhi Proudyogiki Vishwavidyalaya

Towards Partial Fulfillment for the Award of

Bachelor of Technology
in
Computer Science & Information Technology

Submitted by:

Pratham Mukati (0827CI221102)
Liza Bhor (082CI221083)
Manasvi Ghune (0827CI221086)
Mohit Rajput (0827CI221094)

Guided by:

Dr.Ajit Jain
Prof. Ashwinee Gadwal
CSIT Department

Acropolis Institute of Technology & Research, Indore
July- Dec 2024

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

2022-2026

DECLARATION

We hereby declare that the work, which is being presented in this project entitled "**IoT Based Smart Irrigation System**" in partial fulfillment of the requirements for the award of degree of **Bachelor ofTechnology in Computer Science and Information Technology**, is authentic record of work carried out by us.

Place: CSIT, Indore

Date:

Signature of Student

Pratham Mukati
0827CI221102
Liza Bhor
0827CI221083
Manasvi Ghune
0827CI221086
Mohit Rajput
0827CI221094

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

2022-2026

RECOMMENDATION

This is to certify that the work embodied in this project entitled **"IoT Based Smart Irrigation System"** submitted by **Pratham Mukati (0827CI221102), Liza Bhor (0827CI221083), Manasvi Ghune (0827CI1086), Mohit Rajput (0827CI1094)** is a satisfactory account of the Bonafide work done under the supervision of **Prof. Nidhi Nigam & Prof. Ashwinee Gadwal**, is recommended towards partial fulfillment for the award of the Bachelor of Technology in Computer Science & Information Technology degree by Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal.

Prof. Nidhi Nigam
Prof. Ashwinee Gadwal

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

2022-2026

CERTIFICATE

The Project entitled **"IoT Based Smart Irrigation System"** submitted by **Pratham Mukati (0827CI221102), Liza Bhor (0827CI221083), Manasvi Ghune (0827CI1086), Mohit Rajput (0827CI1094)** has been examined and is hereby approved towards partial fulfillment for the award of **Bachelor of Technology** in **Computer Science & Information Technology,** for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed, or conclusion drawn therein, but approve the project only for the purpose for which it has been submitted.

Prof. Nidhi Nigam
Prof. Ashwinee Gadwal
Date:

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

2022-2026

STUDENT UNDERTAKING

This is to certify that project entitled **"IoT Based Smart Irrigation System"** has developed by us under the supervision of **Prof. Nisha Rathi & Prof. Ashwinee Gadwal**. The whole responsibility of the work done on this project is ours. The sole intension of this work is only for practical learning and research.

We further declare that to the best of our knowledge, this report does not contain any part of any work which has been submitted for the award of any degree either in this University or in any other University / Deemed University without proper citation and if the same work found then we are liable for explanation to this.

Pratham Mukati	**Liza Bhor**	**Manasvi Ghune**	**Mohit Rajput**
Date:	**Date:**	**Date:**	**Date:**

ACKNOWLEDGEMENT

We thank the almighty Lord for giving me the strength and courage to sail out through the tough and reach on shore safely.

We owe a debt of sincere gratitude, deep sense of reverence and respect to our mentors **Prof. Nisha Rathi** and **Prof. Ashwinee Gadwal**, AITR, Indore for their motivation, sagacious guidance, constant encouragement, vigilant supervision and valuable critical appreciation throughout this project work, which helped us to successfully complete the project on time.

We express profound gratitude and heartfelt thanks to **Prof. (Dr.) Shilpa Bhalerao HOD CSIT**, AITR Indore, for her support, suggestion and inspiration for carrying out this project. We would be failing in our duty if we do not acknowledge the support and guidance received from **Prof. (Dr.) S. C. Sharma, Director**, AITR, Indore whenever needed. We take the opportunity to convey my regards to the **Management of the Acropolis Institute, Indore** for extending academic and administrative support and providing us with all necessary facilities for project to achieve our objectives.

We are grateful to our parents and family members who have always loved and supported us unconditionally.

Pratham Mukati	**Liza Bhor**	**Manasvi Ghune**	**Mohit Rajput**
Date:	**Date:**	**Date:**	**Date:**

ABSTRACT

Water scarcity is a growing concern worldwide, making efficient irrigation systems essential for sustainable agriculture. Traditional irrigation methods often lead to excessive water consumption and inefficient distribution, contributing to resource wastage. This project proposes an IoT-based Smart Irrigation System that automates water supply by continuously monitoring real-time environmental parameters. The system utilizes soil moisture sensors, temperature sensors, weather data APIs, and microcontrollers to assess soil conditions and determine optimal irrigation levels. An automated control mechanism ensures that water is supplied only when necessary, reducing human intervention and preventing overwatering or underwatering.

By leveraging wireless communication, cloud-based data storage, and remote monitoring, the system allows users to access real-time data and control irrigation settings via a web or mobile application. Additionally, weather predictions help optimize irrigation schedules, further enhancing efficiency. The integration of machine learning algorithms can improve decision-making over time by analysing historical data patterns.

This project aims to promote water conservation, reduce operational costs, and improve crop yield, making it highly beneficial for farmers, greenhouse operators, and plant nurseries. With its potential scalability, the system can be adapted for both small-scale gardening and large-scale agricultural applications, contributing to the development of smart farming and sustainable water management practices.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER1

Chapter 1: Introduction

1.1Overview

An IoT-based Smart Irrigation System is designed to optimize the process of watering plants through the integration of modern technology and automation. This system offers an innovative solution to the challenges associated with traditional watering methods, ensuring precise water management tailored to the needs of plants. Using the NodeMCU ESP8266 as the core component, the system continuously monitors soil moisture levels and automates the irrigation process, thereby conserving water and enhancing plant health.

The implementation of IoT in this system allows for real-time monitoring and control, making it accessible and convenient for users through mobile or web interfaces. With the growing demand for efficient water management and plant care solutions, such systems represent a significant step forward in combining technology with sustainability to deliver smarter irrigation practices.

1.2Existing System

Numerous IoT-based smart irrigation systems have been developed, primarily focusing on large-scale agricultural applications. These systems are designed to optimize water usage across vast fields and are equipped with advanced features such as weather integration, crop-specific irrigation schedules, and large-area soil moisture monitoring. Their primary goal is to enhance productivity, conserve water, and reduce labor requirements in agricultural settings.

1.3Problem Statement

Effective water management is a critical aspect of plant care, yet traditional irrigation methods often lead to overwatering, underwatering, and inefficient use of resources. These issues can harm plant health, waste water, and require constant manual intervention. Furthermore, individuals with busy schedules or limited knowledge of plant care face challenges in maintaining optimal watering routines. Existing solutions are either designed for large-scale agricultural applications or are not tailored for small-scale setups, such as houseplants or compact gardens.This highlights the need for a cost-effective, automated, and user-friendly system that leverages Internet of Things (IoT) technology to monitor soil moisture, automate irrigation, and provide real-time control and monitoring. Such a solution would promote sustainable water use and simplify plant care for individuals, ensuring healthier plants with minimal effort.

1.4 Proposed System

The proposed system is an **IoT-based smart irrigation solution** tailored for small-scale applications such as home gardens, potted plants, and compact green spaces. It is designed to provide an affordable, efficient, and user-friendly way to automate plant watering with minimal effort.

Key features of the system include:

- **Real-Time Monitoring**: Soil moisture sensors continuously monitor the soil's condition to determine when watering is necessary.
- **Automatic Irrigation**: The system triggers irrigation only when required, preventing overwatering and conserving water.
- **Cost-Effectiveness**: Unlike large-scale agricultural solutions, this system is affordable and optimized for small setups, making it accessible to a wider audience.
- **Ease of Use**: The system is easy to install and maintain, requiring minimal technical expertise.
- **IoT Integration**: Users can remotely monitor and control the system through a mobile or web application, offering convenience and flexibility for plant care.

By addressing the gap in existing irrigation technologies, the proposed system ensures healthier plants, sustainable water usage, and effortless management for small-scale environments.

1.5Need and Scope

Efficient water management is crucial for healthy plant growth, yet traditional methods often lead to overwatering, underwatering, and wastage, especially in small-scale setups like houseplants and home gardens. These spaces lack accessible and affordable smart irrigation solutions, leaving users reliant on manual effort or costly, large-scale systems. To address this, the project aims to develop a compact IoT-based smart irrigation system tailored for small-scale applications. The system automates watering by monitoring soil moisture and activating irrigation only when needed, ensuring optimal water use and healthier plants. With IoT integration, users can remotely monitor and control the system via a mobile or web app, providing convenience and flexibility. Affordable, easy to install, and adaptable to various plant types, this solution simplifies plant care while promoting sustainable and efficient water management.

1.6 ReportOrganization

- **Chapter 1** states the overview of the project while discussing the existing systems in today’s scenario. Describe the problem statement and how we have given our proposed solution considering all the shortcomings of the previously used system.

- **Chapter 2** states the literature survey i.e. the background details of our system including the software engineering paradigm and explaining about the technologies (Software and Hardware requirement) which we have used building the system.

- **Chapter 3** states about the Analysis of the whole system i.e. identification of system requirement about the feasibility study-Technical Feasibility, Financial Feasibility, operational Feasibility.

- **Chapter 4** states about the Design of the whole system including all the UML diagrams, all the tools used with ER Diagram and Data Flow Diagram and Data Dictionary.

- **Chapter 5** states the whole code of the system and its integration and adaptability.

- References: The books, websites, journals, blogs which we have referred.

CHAPTER 2

Chapter 2: Literature Survey

2.1 Study

We explored the existing IoT-based smart irrigation systems, particularly those designed for large-scale agricultural use. These systems typically incorporate advanced features such as real-time soil, moisture monitoring, weather-based irrigation scheduling, and water conservation techniques. However, we found that such systems are often expensive, complex, and impractical for small-scale applications like home gardens or potted plants. Through this study, we identified a significant gap in the availability of affordable, user-friendly, and compact solutions tailored for smaller setups, which served as the foundation for this project.

2.2 Problem Methodology

The problem of water wastage and inconsistent irrigation in small-scale plant care can lead to poor plant health and resource inefficiency. To address this, our IoT-based solution uses soil moisture sensors, a NodeMCU ESP8266 microcontroller, and an automated water pump system. This setup monitors soil moisture in real-time and activates irrigation only when needed, based on peredefined thresholds. This methodology conserves water, ensures optimal plant growth, and reduces the need for manual intervention.

2.3 Software Engineering Paradigm

For this project, we followed the **Agile Software Development Paradigm**, which allowed us to build the system iteratively and improve it based on testing and feedback. The iterative approach helped in refining features, such as real-time monitoring and remote control, while maintaining flexibility to accommodate changes. Agile methodology was ideal for this project due to its focus on quick prototyping and continuous improvement.

- **Gap Identification**: We identified that existing IoT-based irrigation systems are primarily designed for large-scale agriculture, making them expensive and impractical for small-scale setups. This highlighted the need for a compact, affordable solution tailored for smaller applications.
- **Technology Integration**: We utilized IoT technology by incorporating components such as the NodeMCU ESP8266, soil moisture sensors, and a water pump to automate irrigation. The system is further enhanced with remote monitoring and control through a mobile or web interface.
- **Sustainability and Accessibility**: Our system optimizes water usage, reduces wastage, and ensures sustainable plant care. It is designed to be user-friendly and accessible, making it ideal for small-scale applications.

2.4 Software Development Life Cycle:

The software development process for the project followed the Waterfall Model, starting with a detailed requirements analysis. We first defined the system's functionality, such as monitoring soil moisture and automating irrigation. The design phase included setting up the microcontroller, sensors, and IoT integration. Implementation involved coding and integrating all components, while testing ensured the system operated reliably. Finally, the deployment phase focused on making the system user-ready with a mobile/web interface for remote monitoring.

2.4 Technology Methodology

We selected IoT as the core technology for this project due to its ability to enable real-time monitoring and control. The system uses the NodeMCU ESP8266 microcontroller, which is compact, affordable, and supports Wi-Fi connectivity. Soil moisture sensors detect the water level in the soil, and the data is processed by the microcontroller to trigger a water pump if needed. IoT integration allows the system to communicate with a mobile or web application, providing users with remote access and control.

2.4.1 Hardware Requirements

The hardware requirements for this project include the following:

- **NodeMCU ESP8266 Microcontroller**: To serve as the central processing unit and enable IoT connectivity.
- **Soil Moisture Sensors**: To monitor the water content in the soil.
- **Relay Module**: To control the water pump.
- **Water Pump**: To automate the irrigation process.
- **Power Supply**: To provide energy for the system.
- **Water Reservoir**: To store water for irrigation.

These components were chosen for their affordability, availability, and suitability for small-scale applications.

2.4.2 Software Requirements

The required software for development environment:

1. **Arduino IDE**

- Used to program the NodeMCU ESP8266 microcontroller with the required code for the system's operation.

- Allows integration of sensor libraries and communication protocols like MQTT or HTTP.

2. IoT Platform

- Blynk App, ThingSpeak, or Adafruit IO can be used to enable real-time monitoring and control of the system through a mobile or web interface.
- These platforms facilitate visualization of sensor data, device control, and notification alerts.

3. Embedded Libraries

- Libraries such as ESP8266WiFi.h, Adafruit_Sensor.h, or similar, depending on the sensors and IoT platform, are required to support hardware-software communication.

4. Database/Cloud Storage (Optional)

- If needed, services like Firebase or AWS IoT can be used to store historical data related to soil moisture levels, irrigation events, or water usage patterns.

5. Mobile or Web Application

- An application for user interaction, either pre-built (like the Blynk app) or custom-developed using frameworks such as Flutter or ReactJS, to allow remote control and real-time updates.

6. Operating System

- A compatible OS such as Windows, macOS, or Linux is required to run the Arduino IDE and manage system configurations.

CHAPTER 3

Chapter 3: Analysis

3.1 Identification of System Requirements

Hardware Requirements

The system must include essential hardware components such as:

- **Soil Moisture Sensors**: These sensors measure the water content in the soil, providing critical data to determine when irrigation is needed. Accurate readings prevent overwatering and ensure optimal plant growth.
- **Temperature and Humidity Sensors**: These sensors monitor environmental factors that influence water requirements, enabling dynamic adjustments to irrigation schedules.
- **Microcontroller**: Devices like Arduino or Raspberry Pi act as the system's brain, processing sensor data and executing automated irrigation decisions. Their flexibility and programmability are crucial for system efficiency.

Connectivity Requirements

Reliable connectivity modules, such as Wi-Fi or GSM, are essential for transmitting sensor data to the cloud platform and receiving user commands. This ensures seamless remote monitoring and control, critical for real-time operations.

Software Requirements

A cloud-based solution is mandatory for storing and analyzing data in real time. This supports data visualization and actionable insights. Additionally, a mobile or web application is required to enable users to monitor system performance and control irrigation remotely.

Power Supply

A stable and uninterrupted power supply is vital for the system's continuous operation. Incorporating backup options, such as a battery or solar panel, ensures reliability during outages and enhances the system's practicality.

Scalability and Compatibility

The system must be designed to accommodate additional sensors and expand to larger coverage areas. Compatibility with a variety of sensors and irrigation systems ensures the solution can adapt to different use cases, making it versatile for agricultural and landscaping applications.

3.2 Functional Requirements

Real-Time Monitoring

The system must continuously monitor soil moisture levels to ensure plants receive the correct amount of water. Real-time data helps avoid over-irrigation or water shortages, directly impacting crop health and resource conservation.

Automated Control

The system must use predefined thresholds to automate water flow. This eliminates manual intervention, optimizes water usage, and ensures timely irrigation based on the soil's needs.

Alerts and Notifications

The system must send timely alerts to users about critical conditions such as low moisture levels, sensor failures, or irregularities in water flow. These notifications ensure prompt corrective actions and reduce the risk of system failure.

Data Logging and Analysis

By storing data on irrigation patterns and water consumption, the system enables users to track performance and identify trends. This data helps in refining irrigation strategies and supports decision-making for sustainable water management.

Integration with Weather Services

Incorporating weather forecasts ensures that the system can adjust irrigation schedules based on expected rainfall, preventing water wastage and enhancing efficiency.

Remote Operation

Remote control via a mobile or web application provides users with flexibility to monitor and adjust the system from any location, improving user convenience and accessibility.

3.3 Non-Functional Requirements

Reliability

The system must operate consistently across different environmental conditions. Dependable performance is critical to maintaining user trust and ensuring uninterrupted irrigation.

Low Latency

Quick response times to sensor inputs and user commands are essential to avoid delays that could adversely affect plant health or water efficiency.

Scalability

The system should allow easy integration of additional sensors and support expansion to larger areas, ensuring adaptability to growing needs**.**

Security

The system must implement robust security protocols to protect sensitive data and prevent unauthorized access, ensuring system integrity and user privacy.

Energy Efficiency

The system should be designed to minimize energy consumption. Using low-power components and integrating renewable energy sources like solar panels can further reduce operational costs.

Usability

The user interface must be intuitive and simple, catering to users with varying levels of technical expertise. Clear visualizations and straightforward controls enhance usability.

Maintainability

Durable components and modular design simplify repairs andreplacements. Minimal maintenance requirements ensure the system remains operational over long periods without significant user effort.

3.4 Feasibility Study

3.4.1 Technical Feasibility

• **Component Availability:**The project leverages readily available technologies like IoT-enabled sensors and microcontrollers (e.g., Arduino or Raspberry Pi), ensuring ease of procurement.

• **Connectivity:**Connectivity modules such as Wi-Fi, GSM, or LoRaWAN enable reliable communication between the sensors, controller, and cloud platform.

• **Software Tools:** The use of programming languages like Python and open-source libraries accelerates development and supports efficient data processing.

• **Cloud Platforms:** Scalable platforms like AWS or Google Firebase provide robust infrastructure for data storage, analysis, and remote accessibility, making the system technically viable.

3.4.2 Financial Feasibility

• **Affordable Components:**Cost-effective sensors and microcontrollers reduce the initial investment required for system deployment.

• **Open-Source Software:** Free software tools minimize licensing expenses, lowering overall project costs.

• **Long-Term Benefits:** Significant savings in water usage and reduced labour costs ensure a high return on investment. Improved crop yields further enhance financial viability.

• **Scalability:** The modular design minimizes costs for future expansions or upgrades, making the system economically sustainable.

3.4.3 Operational Feasibility:

• **Ease of Integration:** The system's design allows for straightforward integration into existing agricultural setups or residential gardens, reducing setup complexity.
• **User-Friendly Interface:** The intuitive application simplifies interaction, enabling even non-technical users to operate the system with ease.
• Automation: Automated processes eliminate the need for constant human intervention, enhancing efficiency and reducing the time required for irrigation management.
• **Adaptability:** The system's flexibility ensures reliable operation across diverse environmental conditions, from arid farmland to urban gardens**.**
• **Low Maintenance:** Robust components and simplified maintenance protocols ensure long-term operational reliability, reducing downtime and associated costs.

CHAPTER 4

Chapter 4: Project Planning

Introduction to Project Planning

Project planning is a critical phase in the development of any system, ensuring the systematic execution of tasks to achieve the project's goals. For our IoT-based Water Conservation and Smart Irrigation System, proper planning ensures the successful integration of hardware, software, and sustainable practices. This chapter outlines the timeline, resources, risks, and strategies involved in the project.

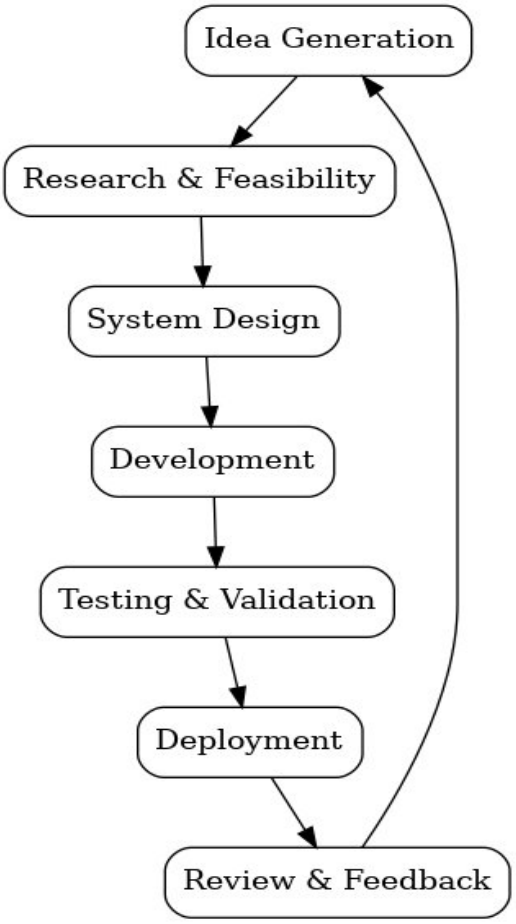


Figure 4.1: Project Planning

1. Objective of Project Planning

The primary objectives of project planning for this project include:

1. Defining the scope and deliverables of the system.
2. Ensuring efficient utilization of resources for hardware and software integration.
3. Establishing milestones and timelines for development, testing, and deployment.
4. Identifying potential risks and mitigation strategies.
5. Aligning the project execution with sustainability and environmental goals.

2. Key Phases of the Project

1. **Requirement Analysis**:

- Understand the problem of inconsistent plant watering in institutional setups.
- Identify sources of wastewater, e.g., wash basins, and assess its usability.
- Determine soil moisture needs using IoT sensors.

2. **System Design and Architecture**:

- Create a blueprint of the system, including:
    a. Sensors for soil moisture detection.
    b. Water storage tank design.
    c. Controller integration for decision-making and automated irrigation.

3. **Component Procurement and Assembly**:

- Identify and procure essential components:
    a. Soil moisture sensors, water pumps, microcontrollers (e.g., Arduino/Raspberry Pi), storage tanks, and piping systems.

4. **Software Development**:

- Develop a program to:
    a. Read sensor data.
    b. Analyze soil moisture levels.
    c. Trigger the irrigation system using actuators and pumps.

5. **System Integration**:

- Combine hardware and software for a seamless operation.
- Ensure proper communication between IoT sensors, controllers, and actuators.

6. **Testing and Calibration**:

- Test the system for different environmental and soil conditions.

- Optimize the water flow based on soil requirements.

7. **Deployment**:
- Install the system in an institutional environment.
- Train users for basic operations and maintenance.

3. Milestones and Timeline

Milestone	Expected Duration
Requirement Analysis	1 week
System Design	2 weeks
Component Procurement	1 week
Hardware Assembly	2 weeks
Software Development	2 weeks
System Integration and Testing	2 weeks
Deployment and Documentation	1 week

4. Resource Allocation

- **Hardware**:
  - Sensors (e.g., soil moisture sensors): Detect soil conditions.
  - Microcontroller: Acts as the brain of the system.
  - Water storage tank: Stores collected water from wash basins.
  - Actuators: Controls water flow for irrigation.
  - Piping and valves: Deliver water to plants.

- **Software**:
  - Programming tools like Arduino IDE, Python, or MATLAB for sensor integration and control logic.

- **Manpower**:
  - Team members include:
    - A hardware engineer for system assembly.
    - A software developer for programming and IoT integration.
    - A project manager for monitoring progress.

5. Risk Management

Identifying and mitigating risks is crucial for the project's success:

1. **Water Quality Issues**:
- Use basic filtration methods to prevent clogging and damage to sensors.

2. **System Malfunction**:
- Conduct regular testing and calibration to ensure reliable performance.

3. **Power Supply Interruptions**:
- Integrate backup power solutions, such as solar panels or battery packs.

4. **Sensor Calibration Errors**:
- Periodically test sensors to ensure accurate soil moisture readings.

6. Budget Estimation

An approximate budget allocation for the project is as follows:

Item	Estimated Cost (in INR)
Sensors and controllers	5,000
Storage tank and piping	3,000
Pump and actuators	2,500
Microcontroller (Arduino/RPi)	2,000
Software and testing tools	1,500
Miscellaneous	1,000
Total	15,000

7. Sustainability Goals

- Utilize wastewater from institutional wash basins to reduce water wastage.
- Enable smart irrigation to conserve water by providing only the required amount to plants.
- Reduce dependency on human intervention, promoting automation and efficiency.

CHAPTER 5

Chapter 5: Design

5.1 Introduction to UML

Unified Modeling Language (UML) is a standardized modeling language used to visualize, design, and document the structure and behavior of software systems. It provides a set of diagrams to represent the different aspects of a system, making it easier for developers, designers, and stakeholders to understand and communicate ideas.

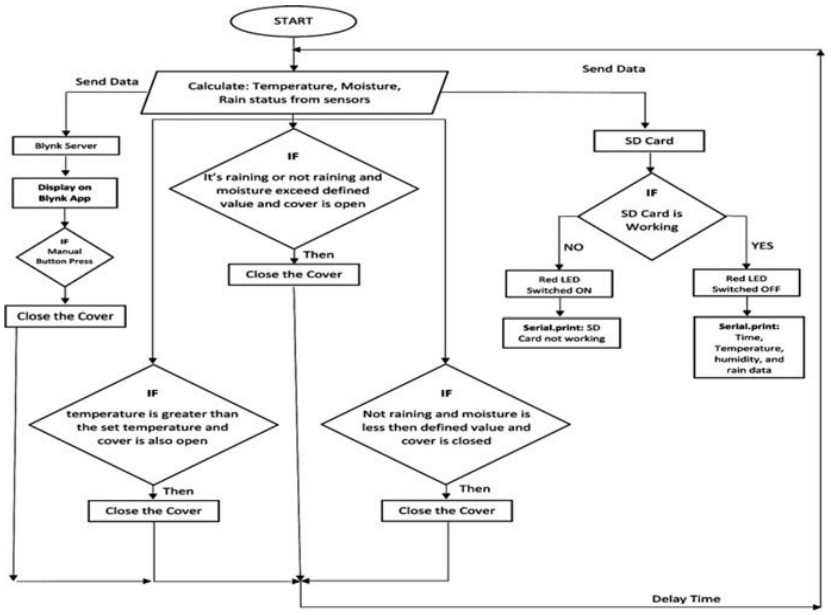
UML plays a crucial role in the design of our IoT-based Water Conservation and Smart Irrigation System by helping us conceptualize and plan the system's architecture and interactions.

Why Use UML?

1. **Visual Representation**: UML offers a clear visual way to design complex systems, making it easier to understand relationships and dependencies.
2. **Standardization**: It is widely accepted and used, ensuring consistency in communication among teams.
3. **Abstraction**: Simplifies complex systems by breaking them down into components.
4. **Improved Collaboration**: Helps both technical and non-technical stakeholders understand the system.
5. **Documentation**: Serves as a blueprint for future reference or system maintenance.

5.2 UML Diagrams

Unified Modeling Language (UML) diagrams are graphical representations of a system's structure and behavior. They help visualize, design, and document software or systems, making it easier to understand and communicate their architecture and processes.



5.3 Use Case Diagram:

- Captures the interactions between the system and its external actors.
- The actors in this project:
  i. Soil Moisture Sensor
  ii. IoT Device (Microcontroller)
  iii. Water Manager
  iv. User (System Monitor)

- Use Cases
  i. Collect soil data
  ii. Trigger irrigation
  iii. Monitor water level in the storage tank
  iv. Automate water collection from wash basins

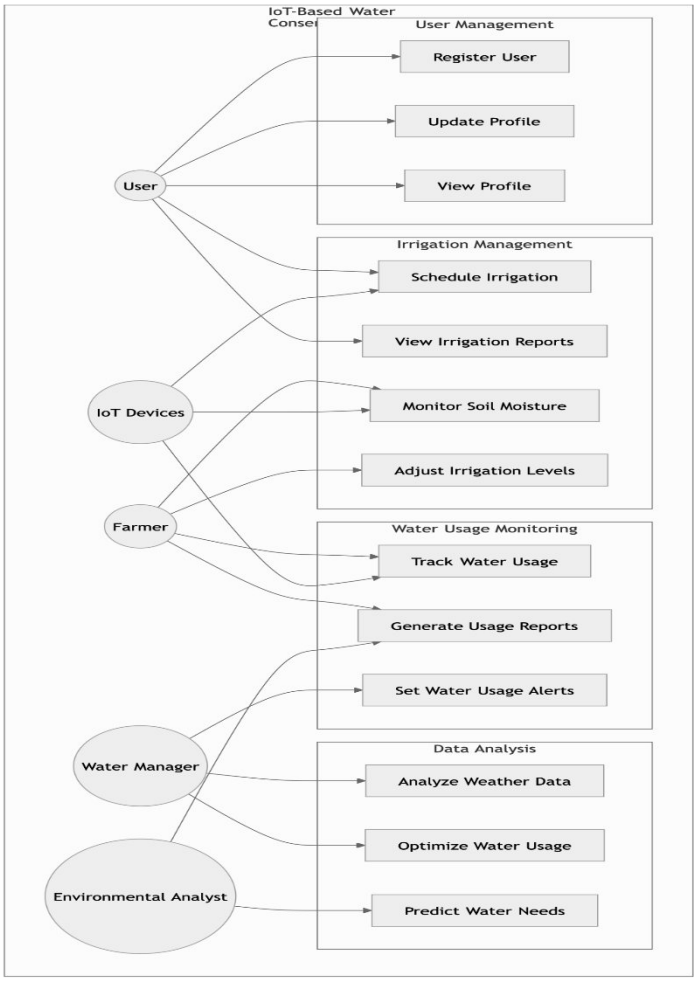


Figure 5.1: Use Case Diagram

5.4 Class Diagram:

- Represents the classes in the system and their relationships.
- Example Classes:
  i. Sensor (attributes: sensorID, moistureLevel; methods: collectData)
  ii. Controller (attributes: controllerID, status; methods: analyzeData, controlPump)
  iii. Tank (attributes: tankCapacity, currentWaterLevel; methods: checkLevel, storeWater)

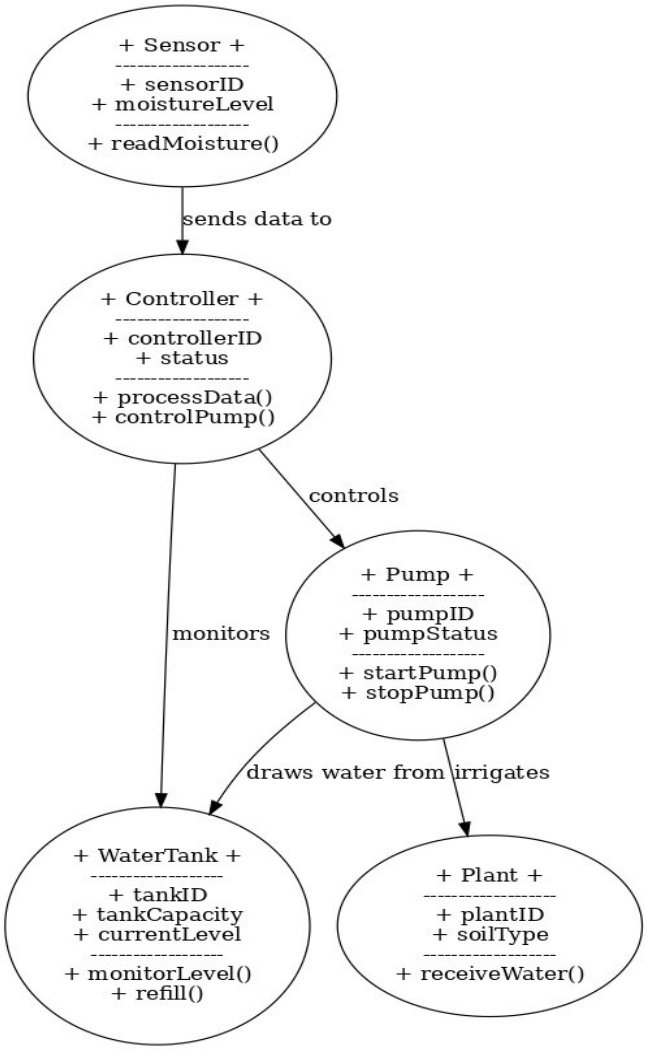


Figure 5.2: Class Diagram

5.4 Activity Diagram:

Illustrates the flow of activities in the system.

5.5Steps for this project:

5.5.1 Wastewater collected from the wash basin.

5.5.2 The storage tank checks the water level.

5.5.3 Soil moisture sensor collects data.

5.5.4 The controller decides whether irrigation is needed.

5.5.5 Water pump irrigates the soil if required.

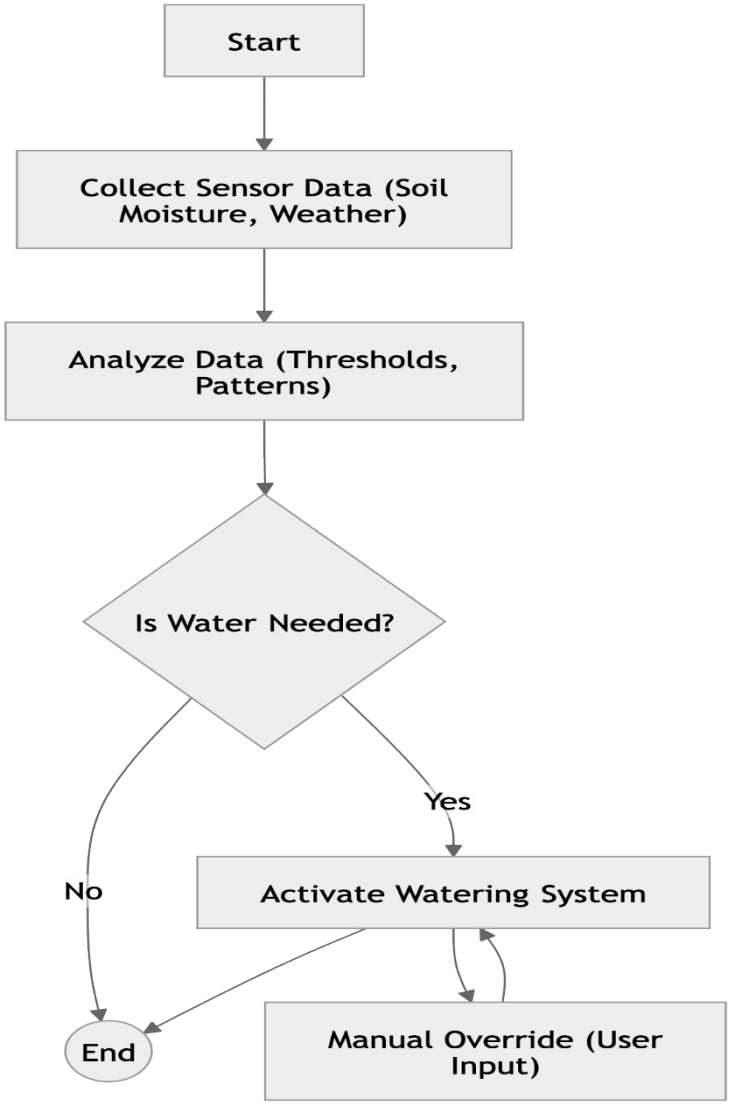


Figure 5.3: Activity Diagram

5.6 Entity-Relationship (ER) Diagram

An Entity-Relationship (ER) Diagram is a visual representation of the data model for a system. It shows the system's key entities, their attributes, and the relationships between them. For the IoT-based Water Conservation and Smart Irrigation System, the ER diagram helps in understanding the data flow and interaction between various components of the system.

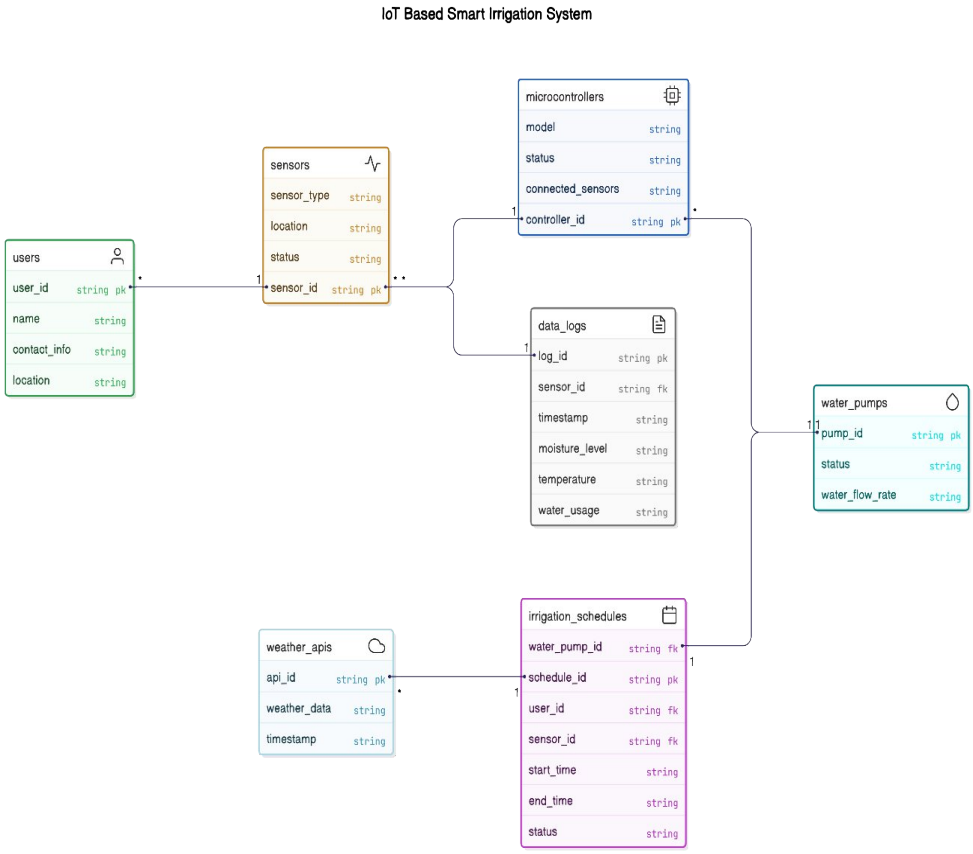


Figure 5.4: ER Diagram

5.7Sequence Diagram:

5.7 Describes the order of interactions between the components.

5.8 Example:

5.8.1 Sensor → Controller: Sends moisture data.

5.8.2 Controller → Pump: Activates pump based on soil needs.

5.8.3 Pump → Plants: Waters the soil.

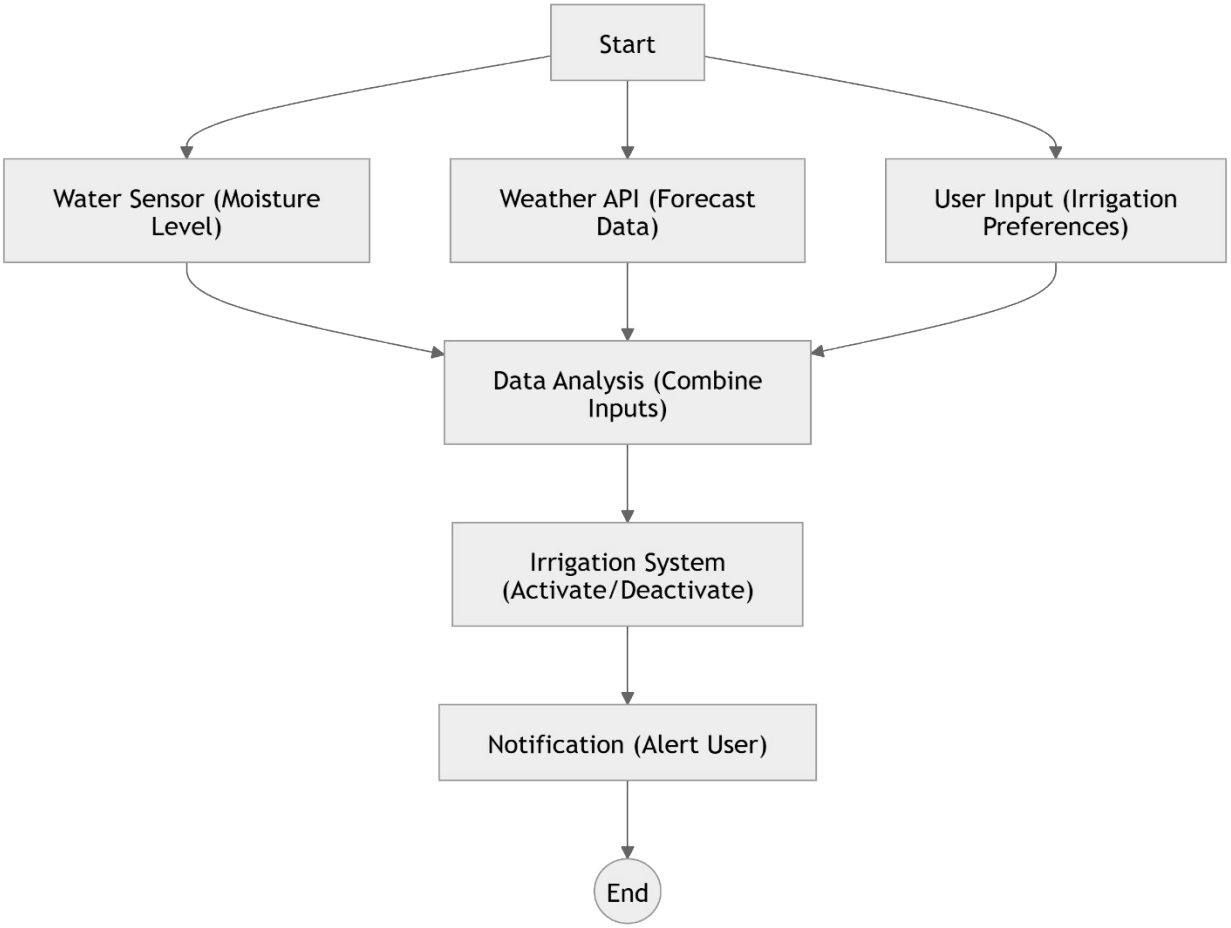


Figure 5.5: Sequence Diagram

5.8 TABLE STRUCTURE:

A.Registration

Column Name	Data Type	Constraints
user_id	SERIAL	Primary Key
username	VARCHAR(50)	Not Null, Unique
password	VARCHAR(100)	Not Null
email	VARCHAR(100)	Not Null, Unique
created_at	TIMESTAMP	Default: CURRENT_TIMESTAMP

B.Field Details

Column Name	Data Type	Constraints
field_id	SERIAL	Primary Key
user_id	INT	Foreign Key (registration.user_id), Not Null
field_name	VARCHAR(100)	Not Null
area_size	DECIMAL(10,2)	Optional
location	TEXT	Optional
crop_type	VARCHAR(50)	Optional

C.Sensor Data

Column Name	Data Type	Constraints
data_id	SERIAL	Primary Key
field_id	INT	Foreign Key (field_details.field_id), Not Null
moisture_level	DECIMAL(5, 2)	Not Null
Temperature	DECIMAL(5, 2)	Optional
Humidity	DECIMAL(5, 2)	Optional
Timestamp	TIMESTAMP	Default: CURRENT_TIMESTAMP

D. Water Usage

Column Name	Data Type	Constraints
usage_id	SERIAL	Primary Key
field_id	INT	Foreign Key (field_details.field_id), Not Null
water_volume_liters	DECIMAL(10,2)	Not Null
irrigation_time	TIMESTAMP	Default: CURRENT_TIMESTAMP
Method	VARCHAR(50)	Optional (e.g., drip, sprinkler)

E.Device Status

Column Name	Data Type	Constraints
device_id	SERIAL	Primary Key
field_id	INT	Foreign Key (field_details.field_id), Not Null
device_type	VARCHAR(50)	Not Null
Status	VARCHAR(20)	Not Null (e.g., active, error)
last_checked	TIMESTAMP	Default: CURRENT_TIMESTAMP

F.User Feedback

Column Name	Data Type	Constraints
feedback_id	SERIAL	Primary Key
user_id	INT	Foreign Key (registration.user_id), Not Null
field_id	INT	Foreign Key (field_details.field_id), Not Null
feedback_text	TEXT	Optional
submitted_at	TIMESTAMP	Default: CURRENT_TIMESTAMP

CHAPTER 6

CHARPTER 6: IMPLIMENTATION

CODE:

```
  digitalWrite(ModeLed, prevMode);
}
 void loop() {

  Blynk.run();
  timer.run(); // Initiates SimpleTimer

  button2.check();
  controlMoist();
}

void button1Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT1");
  switch (eventType) {
    case AceButton::kEventReleased:
      //Serial.println("kEventReleased");
      digitalWrite(RelayPin, !digitalRead(RelayPin));
      toggleRelay = digitalRead(RelayPin);
      Blynk.virtualWrite(VPIN_RELAY, toggleRelay);
      break;
  }
}

void button2Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT2");
  switch (eventType) {
    case AceButton::kEventReleased:
      //Serial.println("kEventReleased");
      if(prevMode && toggleRelay == HIGH){
        digitalWrite(RelayPin, LOW);
        toggleRelay = LOW;
        Blynk.virtualWrite(VPIN_RELAY, toggleRelay);
      }
```

```
 pinMode(ModeLed, OUTPUT);
 pinMode(BuzzerPin, OUTPUT);

 pinMode(RelayButtonPin, INPUT_PULLUP);
 pinMode(ModeSwitchPin, INPUT_PULLUP);

 digitalWrite(wifiLed, LOW);
 digitalWrite(ModeLed, LOW);
 digitalWrite(BuzzerPin, LOW);

 dht.begin();    // Enabling DHT sensor

 config1.setEventHandler(button1Handler);
 config2.setEventHandler(button2Handler);

 button1.init(RelayButtonPin);
 button2.init(ModeSwitchPin);

 if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
   Serial.println(F("SSD1306 allocation failed"));
   for(;;);
 }
 delay(1000);
 display.setTextSize(1);
 display.setTextColor(WHITE);
 display.clearDisplay();

 WiFi.begin(ssid, pass);
 timer.setInterval(2000L, checkBlynkStatus); // check if Blynk server is connected every 2 seconds
 timer.setInterval(3000L, sendSensor); // display and send sensor reading every 3 seconds
 Blynk.config(auth);
 //delay(1000);
 controlBuzzer(1000);
```

```
    if (moisturePercentage < (moistPerLow)){
      if (toggleRelay == LOW){
        controlBuzzer(500);
        digitalWrite(RelayPin, HIGH);
        toggleRelay = HIGH;
        Blynk.virtualWrite(VPIN_RELAY, toggleRelay);
        delay(1000);
      }
    }
    if (moisturePercentage > (moistPerHigh)){
      if (toggleRelay == HIGH){
        controlBuzzer(500);
        digitalWrite(RelayPin, LOW);
        toggleRelay = LOW;
        Blynk.virtualWrite(VPIN_RELAY, toggleRelay);
        delay(1000);
      }
    }
  }
  else{
    button1.check();
  }
}

void setup() {
  // Set up serial monitor
  Serial.begin(115200);

  // Set pinmodes for GPIOs
  pinMode(RelayPin, OUTPUT);
  pinMode(wifiLed, OUTPUT);
  pinMode(ModeLed, OUTPUT);
```

```
void getWeather(){

 float h = dht.readHumidity();
 float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit

 if (isnan(h) || isnan(t)) {
   Serial.println("Failed to read from DHT sensor!");
   return;
 }
 else {
   humidity1 = int(h);
   temperature1 = int(t);
  // Serial.println(temperature1);
  // Serial.println(humidity1);
 }
}

void sendSensor()
{
 getMoisture(); // get Moisture reading
 getWeather(); // get reading from DHT11

 displayData(String(moisturePercentage) + " %", "T:" + String(temperature1) + " C,  H:" + String(humidity1) + " %  " + currMode);
  Blynk.virtualWrite(VPIN_MoistPer, moisturePercentage);
  Blynk.virtualWrite(VPIN_TEMPERATURE, temperature1);
 Blynk.virtualWrite(VPIN_HUMIDITY, humidity1);
}

void controlMoist(){
 if(prevMode){
```

```
5
void controlBuzzer(int duration){
  digitalWrite(BuzzerPin, HIGH);
  delay(duration);
  digitalWrite(BuzzerPin, LOW);
}

void displayData(String line1 , String line2){
  display.clearDisplay();
  display.setTextSize(2);
  display.setCursor(30,2);
  display.print(line1);
  display.setTextSize(1);
  display.setCursor(1,25);
  display.print(line2);
  display.display();
}

void getMoisture(){
  sensorVal = analogRead(SensorPin);

  if (sensorVal > (wetSoilVal - 100) && sensorVal < (drySoilVal + 100) ){
    moisturePercentage = map(sensorVal ,drySoilVal, wetSoilVal, 0, 100);
    // Print result to serial monitor
    Serial.print("Moisture Percentage: ");
    Serial.print(moisturePercentage);
    Serial.println(" %");
  }
  else{
    Serial.println(sensorVal);
  }
  delay(100);
}
```

```
BLYNK_CONNECTED() {
  Blynk.syncVirtual(VPIN_MoistPer);
  Blynk.syncVirtual(VPIN_RELAY);
  Blynk.syncVirtual(VPIN_TEMPERATURE);
  Blynk.syncVirtual(VPIN_HUMIDITY);
  //Blynk.syncVirtual(VPIN_MODE_SWITCH);
  Blynk.virtualWrite(VPIN_MODE_SWITCH, prevMode);
}

BLYNK_WRITE(VPIN_RELAY) {
  if(!prevMode){
    toggleRelay = param.asInt();
    digitalWrite(RelayPin, toggleRelay);
  }
  else{
    Blynk.virtualWrite(VPIN_RELAY, toggleRelay);
  }
}

BLYNK_WRITE(VPIN_MODE_SWITCH) {
  if(prevMode !=  param.asInt()){
    prevMode =  param.asInt();
    currMode = prevMode ? "A" : "M";
    digitalWrite(ModeLed, prevMode);
    controlBuzzer(500);
    if(!prevMode && toggleRelay == HIGH){
      digitalWrite(RelayPin, LOW);
      toggleRelay = LOW;
      Blynk.virtualWrite(VPIN_RELAY, toggleRelay);
    }
   }
}
```

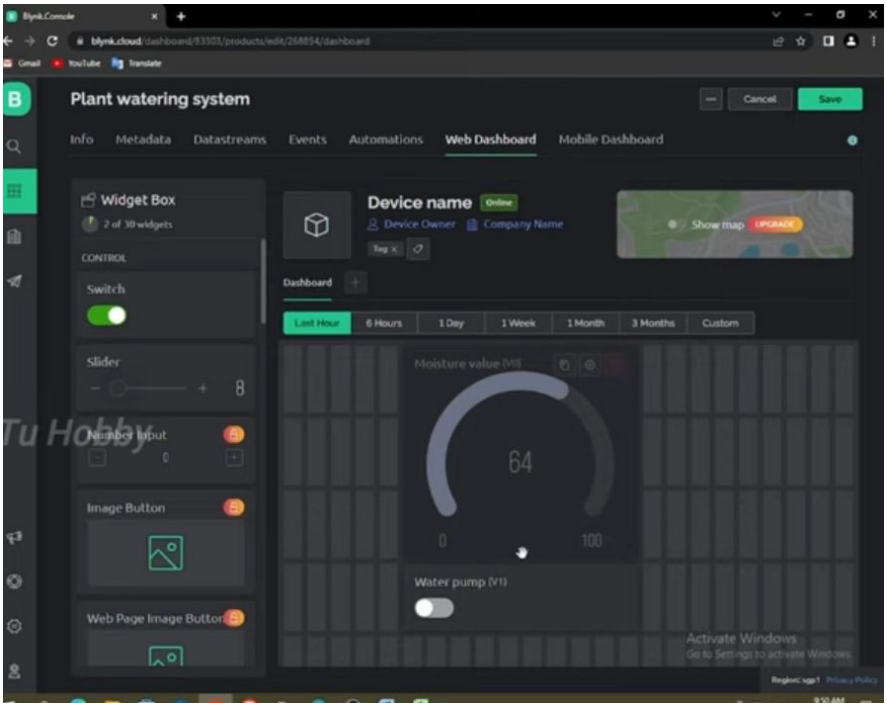
```
int      moisturePercentage;
bool     toggleRelay = LOW; //Define to remember the toggle state
bool     prevMode = true;
int      temperature1 = 0;
int      humidity1    = 0;
String   currMode  = "A";

char auth[] = BLYNK_AUTH_TOKEN;

ButtonConfig config1;
AceButton button1(&config1);
ButtonConfig config2;
AceButton button2(&config2);

void handleEvent1(AceButton*, uint8_t, uint8_t);
void handleEvent2(AceButton*, uint8_t, uint8_t);

BlynkTimer timer;
DHT dht(DHTPin, DHTTYPE);

void checkBlynkStatus() { // called every 3 seconds by SimpleTimer

  bool isconnected = Blynk.connected();
  if (isconnected == false) {
    Serial.print("Blynk Not Connected ");
    digitalWrite(wifiLed, LOW);
  }
  if (isconnected == true) {
    digitalWrite(wifiLed, HIGH);
    //Serial.println("Blynk Connected");
  }
}
 #define RelayPin        25  //D25
 #define wifiLed         2   //D2
 #define RelayButtonPin  32  //D32
 #define ModeSwitchPin   33  //D33
 #define BuzzerPin       26  //D26
 #define ModeLed         15  //D15

 // Uncomment whatever type you're using!
 #define DHTTYPE DHT11     // DHT 11
 //#define DHTTYPE DHT22   // DHT 22, AM2302, AM2321
 //#define DHTTYPE DHT21   // DHT 21, AM2301

 //Change the virtual pins according the rooms
 #define VPIN_MoistPer    V1
 #define VPIN_TEMPERATURE V2
 #define VPIN_HUMIDITY    V3
 #define VPIN_MODE_SWITCH V4
 #define VPIN_RELAY       V5

 #define SCREEN_WIDTH 128 // OLED display width, in pixels
 #define SCREEN_HEIGHT 32 // OLED display height, in pixels

 // Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
 #define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);


 int      sensorVal;
 int      moisturePercentage;
 bool     toggleRelay = LOW; //Define to remember the toggle state
 bool     prevMode = true;
 int      temperature1 = 0;
 int      humidity1    = 0;
```

```
/* Fill-in your Template ID (only if using Blynk.Cloud) */
#define BLYNK_TEMPLATE_ID "TMPL37hrMj6_H"
#define BLYNK_TEMPLATE_NAME "nodeMCUesp32"
#define BLYNK_AUTH_TOKEN "8EQ2X6NOTfVaLs_DGHmzDk3dmZ0pNhtL"

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "mickey"; //WiFi Name
char pass[] = "78907890"; //WiFi Password

//Set the maximum wet and maximum dry value measured by the sensor
int wetSoilVal = 930 ;  //min value when soil is wet
int drySoilVal = 3000 ;  //max value when soil is dry

//Set ideal moisture range percentage(%) in soil
int moistPerLow =   20 ;  //min moisture %
int moistPerHigh =   80 ;  //max moisture %

#include <Adafruit_SSD1306.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <DHT.h>
#include <AceButton.h>
using namespace ace_button;

// Define connections to sensor
#define SensorPin       34  //D34
#define DHTPin          14  //D14
#define RelayPin        25  //D25
#define wifiLed         2   //D2
#define RelayButtonPin  32  //D32
#define ModeSwitchPin   33  //D33
```


```
#define BLYNK_TEMPLATE_ID "TMPL37hrMj6_H"
#define BLYNK_TEMPLATE_NAME "nodeMCUesp32"
#define BLYNK_AUTH_TOKEN "8EQ2X6NOTfVaLs_DGHmzDk3dmZ0pNhtL"

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "mickey"; //WiFi Name
char pass[] = "78907890"; //WiFi Password

//Set the maximum wet and maximum dry value measured by the sensor
int wetSoilVal = 930 ;  //min value when soil is wet
int drySoilVal = 3000 ;  //max value when soil is dry

//Set ideal moisture range percentage(%) in soil
int moistPerLow =   20 ;  //min moisture %
int moistPerHigh =   80 ;  //max moisture %

#include <Adafruit_SSD1306.h>
#include <WiFi.h>
```

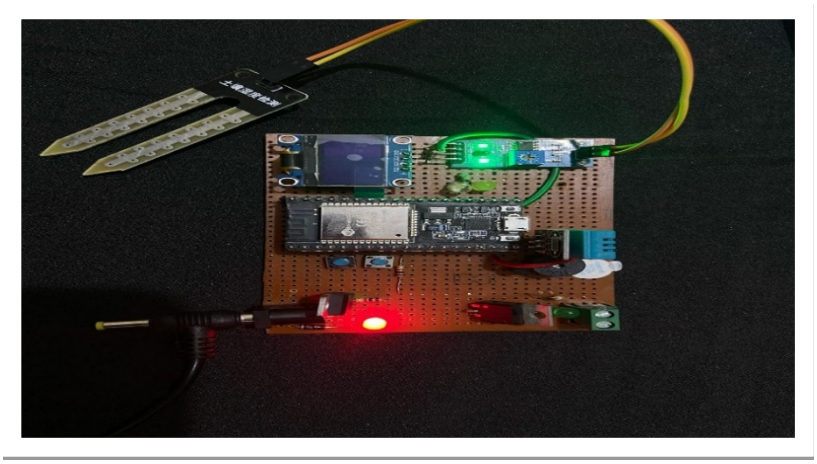

```
    }
    prevMode = !prevMode;
    currMode = prevMode ? "A" : "M";
    digitalWrite(ModeLed, prevMode);
    Blynk.virtualWrite(VPIN_MODE_SWITCH, prevMode);
    controlBuzzer(500);
    break;
}
```

APP for IOT device :Blynk .Cloud

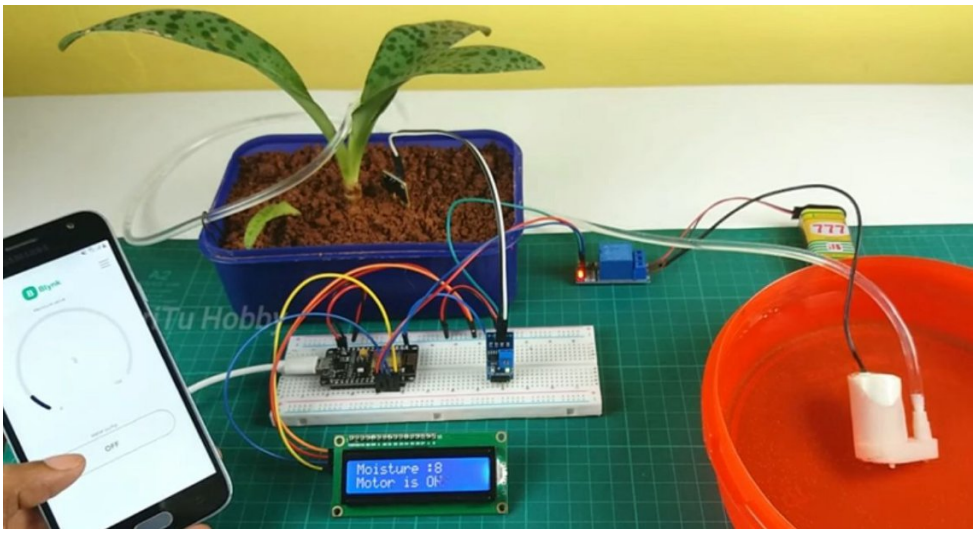


Moblie Application:Blynk APP

TestingImage:



Final Image:



References

Website

[1] How2Electronics, "IoT Smart Agriculture & Automatic Irrigation System with ESP8266,"
Available:https://how2electronics.com/iot-smart-agriculture-automatic-irrigation-system-with-esp8266/. [Accessed: Jan. 29, 2025].

[2]K.G. Liakos, P. Busato, D. Moshou, S. Pearson, D. Bochtis
Machine learning in agriculture: a review
Sensors, 18 (8) (2018), p. 2674

[3]K. Jha, A. Doshi, P. Patel, M. Shah
A comprehensive review on automation in agriculture using artificial intelligence
Artif. Intell. Agric., 2 (2019), pp. 1-12

[4]L. Zhen, A.K. Bashir, K. Yu, Y.D. Al-Otaibi, C.H. Foh, P. Xiao
Energy-efficient random access for LEO satellite-assisted 6G internet of remote things

[5] S. Gupta, M. Kohli, R. Kumar, S. Bandral
IoT based underwater robot for water quality monitoring
IOP Conf. Ser. Mater. Sci. Eng., 1033 (2021), Article 012013, 10.1088/1757-899x/1033/1/012013

[6]B. Anuradha, R. Chaitra, D. Pooja
IoT based low cost system for monitoring of water quality in real time
Int. Res. J. Eng. Technol. (IRJET), Volume: 05 (Issue: 05) (2018)