

Smart Irrigation system for precision farming integrating weather API

A PROJECT REPORT

Submitted by

S GOKUL [RA2211003011996]

R LOKESHWARAN [RA2211003012018]

Under the Guidance of

Dr. Kavisankar L

(Associate Professor, CTECH)

in partial fulfillment of the requirements for the degree of

**BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING**



**DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
COLLEGE OF ENGINEERING AND TECHNOLOGY**

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR- 603 203

MAY 2025



Department of Computational Intelligence
SRM Institute of Science & Technology Own Work*
Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course : B.TECH CSE

Student Name : S GOKUL / R LOKESHWARAN

Registration Number : RA2211003011996 / RA2211003012018

Title of Work : Smart irrigation system for precision farming integrating weather API

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 21CSP302L - Project report titled “**Smart irrigation system for precision farming integrating weather API**” is the bonafide work of “**S GOKUL [RA2211003011996], R LOKESHWARAN [RA2211003012018]**” who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. Kavisankar

ASSOCIATE PROFESSOR
DEPARTMENT OF
COMPUTER
TECHNOLOGY

Examiner 1

SIGNATURE

DR. G NIRANJANA

PROFESSOR & HEAD
DEPARTMENT OF
COMPUTATIONAL
TECHNOLOGIES

Examiner 2

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. Leenus Jesu Martin M**, Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We encompass our sincere thanks to, **Dr. M. Pushpalatha**, Professor and Associate Chairperson - CS, School of Computing and **Dr. Lakshmi**, Professor and Associate Chairperson -AI, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our Head of the Department, **Dr.G.Niranjana** , Professor and Head Department of Computer Technologies SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinators, Panel Head, and Panel Members Department of Computational Intelligence, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. Madhumitha K**, Department of Computational Intelligence, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. Kavisankar L**, Department of Computational Technologies, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his / her mentorship. He / She provided us with the freedom and support to explore the research topics of our interest. His / Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff members of CTECH , School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement

ABSTRACT

This project is about building a Smart Irrigation System that can water plants automatically by checking how dry the soil is and using weather data. The main goal is to save water and reduce the need for manual work by making the watering process automatic.

The system works by using a soil moisture sensor to check if the soil is dry. If it is too dry, the system turns on a small water pump to water the plants. We use an Arduino UNO to control the sensor and the pump. To make the system smarter, we also added a NodeMCU (ESP8266), which connects to the internet and checks if rain is expected by using the OpenWeatherMap API. If rain is predicted, the system skips watering, which helps save water.

An LCD display is used to show the current moisture level of the soil, so users can see what's happening in real time. The system is simple to set up, low-cost, and can be used in home gardens, greenhouses, small farms, and other places where water management is important.

This Smart Irrigation System is a small but effective step towards using technology in farming. It helps improve plant care, saves water, and supports sustainable and modern agricultural practices.

TABLE OF CONTENTS

ABSTRACT		v
TABLE OF CONTENTS		vi
LIST OF FIGURES		vii
ABBREVIATIONS		ix
CHAPTER NO.	TITLE	PAGE NO.
1	INTRODUCTION	1
	1.1 Introduction to Project	1
	1.2 Motivation	2
	1.3 Sustainable Development Goal of the Project	3
	1.4 Product Vision Statement	4
	1.5 Product Goal	5
	1.6 Product Backlog (Key User Stories with Desired Outcomes)	6
	1.7 Product Release Plan	7
2	LITERATURE SURVEY	9
	2.1 Overview of the Research Area	9
	2.2 Existing Models and Frameworks	9
	2.3 Limitations Identified from Literature Survey	10
	2.4 Research Objectives	11
3	SPRINT PLANNING AND EXECUTION	12
	3.1 Sprint 1	
	3.1.1 Sprint Goal with User Stories of Sprint 1	12
	3.1.2 Functional Document	13
	3.1.3 Architecture Document	16
	3.1.4 UI Design	19

3.1.5 Functional Test Cases	20
2.1.6 Daily Call Progress	21
2.1.7 Sprint Retrospective	21
2.1.8 Circuit Diagram	22
 3.2 Sprint 2	
3.2.1 Sprint Goal with User Stories of Sprint 2	23
3.2.2 Functional Document	24
3.2.3 Architecture Document	25
3.2.4 Functional Test Cases	27
3.2.5 Sprint Retrospective	28
3.2.6 Circuit Image	28
 4 RESULTS AND DISCUSSIONS	29
4.1 Project Outcomes	30
4.2 Committed vs Completed User Stories	31
 5 CONCLUSIONS & FUTURE ENHANCEMENT	31
APPENDIX	32
PLAGIARISM REPORT	33

LIST OF FIGURES

CHAPTER NO	TITLE	PAGE NO.
3.1	System Architecture Diagram	17
3.2	UI Design For Login Page	19
3.3	Detailed Functional Test Case	20
3.4	Standup Meetings	21
3.5	Circuit Diagram	22
3.6	Circuit Image	28
4.1	Committed Vs Completed User stories	31

ABBREVIATIONS

1. **Arduino** UNO - Microcontroller used to control the system.
2. **LCD** - Liquid Crystal Display; used for displaying real-time status.
3. **DC** - Direct Current; refers to the type of power used for the water pump.
4. **ESP8266** - A Wi-Fi Module used for fetching weather data.
5. **NodeMCU** - A development board based on the ESP8266 used for IoT applications.
6. **DHT11** - A temperature and humidity sensor.
7. **PWM** - Pulse Width Modulation; used to control the water pump's speed.
8. **GPIO** - General Purpose Input/Output; used to interface with sensors and components on Arduino.
9. **Wi-Fi** - Wireless Fidelity; used for internet connectivity via ESP8266.
10. **API** - Application Programming Interface; used to fetch weather data or control external systems.

CHAPTER 1

INTRODUCTION

1.1 Introduction to Weather-Integrated System

Taking care of plants can be time-consuming, especially when it comes to watering them regularly. Sometimes we water too much, and sometimes we forget. That's where our Smart Irrigation System comes in. This project is designed to help automate the watering process based on how dry the soil is. Instead of guessing, the system uses a soil moisture sensor to check the actual condition of the soil.

When the soil becomes dry beyond a certain level, the system automatically turns on a DC water pump to water the plants. As soon as the soil becomes wet enough, the pump turns off. The heart of this system is the Arduino UNO, which takes the input from the sensor and decides when to activate the pump. It also displays the soil moisture level on a 16x2 LCD screen, so users can always see the current condition.

To make the system even smarter, we added a NodeMCU (ESP8266) Wi-Fi module that connects to the internet and checks the weather forecast using data from the OpenWeatherMap API. If it's going to rain, the system won't turn on the pump, even if the soil is dry. This helps save water and makes the system more intelligent.

In simple terms, this project is a step toward modern farming and gardening. It's low-cost, easy to use, and very helpful for saving water and reducing the need to manually water plants.

1.2 Motivation

The main reason we started this project is that we saw how much water is wasted in everyday plant care. People often water plants out of routine, not based on whether the soil really needs it. Also, checking soil moisture manually and remembering to water every day can be tiring, especially for farmers or busy individuals. We wanted to solve this problem by building something smart that could do it all on its own.

Another motivation is the growing need for sustainable farming practices. With water becoming a limited resource in many areas, it's important to use it wisely. By using a system that waters plants only when they actually need it, we can save a lot of water and avoid overwatering.

We also wanted this system to be simple and affordable so that anyone—whether a home gardener, a farmer, or a student—can use it without needing deep technical knowledge. The project is designed to help not just individuals, but also contribute to larger environmental goals by promoting eco-friendly and smart farming.

1.3 Sustainable Development Goal of the Project

Our Smart Irrigation System supports multiple Sustainable Development Goals (SDGs) introduced by the United Nations, especially those focused on water conservation, innovation, responsible usage of resources, and action against climate change.

Goal6:CleanWaterandSanitation

This goal focuses on ensuring the availability and sustainable management of water for all. Our system helps achieve this by preventing water wastage. It only turns on the water pump when the soil is dry and skips watering if rain is expected. This way, plants get only the amount of water they need, and not a drop more.

Goal9:Industry,Innovation,andInfrastructure

Our project uses affordable technology like Arduino and sensors to bring innovation into the field of agriculture. By combining hardware with weather APIs and automation, we create a small-scale, smart system that represents how technology can improve traditional practices in a low-cost and effective way.

Goal12:ResponsibleConsumptionandProduction

The system ensures that natural resources like water are used efficiently. Instead of watering plants on a schedule, it makes decisions based on real-time soil moisture and weather data. This responsible use of water contributes directly to sustainable agriculture and gardening practices.

Goal13:ClimateAction

Climate change is affecting water availability around the world. Our system helps users adapt by managing water better, reducing dependency on manual watering, and cutting unnecessary usage.

In summary, this project may be small, but it directly supports these global goals by promoting smart technology, saving water, and encouraging sustainable habits in

farming and gardening.

1.4 Product Vision Statement

1.4.1 Audience:

- **Primary Audience:**
Home gardeners, small-scale farmers, agricultural students, and researchers looking for an easy and efficient way to automate plant watering.
- **Secondary Audience:**
Schools, eco clubs, urban gardening enthusiasts, and tech hobbyists interested in applying IoT to real-world environmental projects.

1.4.2 Needs:

- **Primary Needs:**
 - An automated system that waters plants only when the soil is dry.
 - Real-time moisture monitoring for better decision-making.
 - Simple, hands-free operation that reduces manual work.
- **Secondary Needs:**
 - Weather-based watering logic that avoids irrigation before rain.
 - Clear display of soil moisture data through an LCD screen.
 - Affordable and easy-to-use setup using Arduino and basic sensors.

1.4.3 Products:

- **Core Product:**
A Smart Irrigation System that uses soil moisture sensors, Arduino, and weather APIs to water plants automatically and efficiently.
- **Additional Features:**
 - Weather forecast integration using NodeMCU and OpenWeatherMap API.
 - LCD display showing live moisture levels.

- Relay-based control of a DC water pump for smart irrigation.
- Optional manual override and adjustment settings for advanced users.

1.4.4 Values:

- Core Values:
 - Simplicity: Designed to be easy to use, install, and maintain.
 - Sustainability: Focused on saving water and supporting eco-friendly farming.
 - Accessibility: Affordable for students, farmers, and home users.
- Differentiators:
 - Weather-Aware Intelligence: Uses real-time forecasts to make smarter watering decisions.
 - IoT Integration: Combines local sensing with online data for better efficiency.

1.4 Product Goal

The main goal of our Smart Irrigation System is to reduce water waste and make irrigation automatic. Instead of watering plants every day without knowing if they actually need it, our system checks the soil and makes the decision for the user. It saves time, water, and effort.

We also want this system to work well in different situations. Whether it's a small garden at home, a school project, or a small-scale farm, the system should work easily and without problems. That's why we've kept the design simple and the components affordable.

Another goal is to support sustainable and smart farming. Many places around the world face water shortages, and farmers struggle with managing water properly. Our system can help them water their crops efficiently and without wasting water.

In the future, we want to improve the system by adding more features, but the core goal will remain the same: to make watering plants easy, smart, and environmentally friendly. We believe that by achieving this goal, our project can help both people and the planet.

1.6 Product Backlog

S.No	User Stories
#US 1	As a user, I want the system to check the moisture level of the soil so that I can know whether the plants need watering.
#US 2	As a user, I want the system to automatically turn on the water pump when the soil is too dry, so that the plants get water without me doing it manually.
#US 3	As a user, I want the system to stop the water pump when the soil is wet enough, so that water is not wasted.
#US 4	As a user, I want the system to display the soil moisture percentage on an LCD screen, so I can easily monitor the condition of the soil in real-time.
#US 5	As a user, I want the system to check the weather forecast using Wi-Fi, so that it doesn't water the plants if it's going to rain.
#US 6	As a user, I want the system to work even if I am not nearby, so that the plants are always taken care of.
#US 7	As a user, I want the system to be low-cost and easy to set up, so that anyone can use it without special tools or knowledge.

These user stories helped us break the project into small, manageable tasks and build features that are actually useful. We used them to plan each stage of the project and make sure the system works as expected.

1.7 Product Release Plan

The development and testing of the Smart Irrigation System were planned and carried out in simple, manageable stages. Each stage focused on building and testing key features of the system based on the product backlog. The goal was to complete the core functionalities first and then gradually add improvements like weather forecasting and user feedback display.

Sprint 1: Core Functionality

Duration: 1 week

Focus:

- Setting up the Arduino UNO with the soil moisture sensor
- Reading sensor data and printing it on the serial monitor
- Connecting the LCD display to show moisture levels
- Calibrating dry and wet soil values
- Writing basic logic to detect soil dryness

Outcome:

By the end of Sprint 1, we had built a working system that could measure moisture levels and show the results on an LCD screen. We also tested the soil sensor across different types of soil to check if the readings were accurate.

Sprint 2: Automation and Weather Forecast Integration

Duration: 1–1.5 weeks

Focus:

- Integrating the relay module and water pump
- Automating pump control based on soil dryness
- Setting up the ESP8266 (NodeMCU) module

- Fetching live weather forecast data from OpenWeatherMap API
- Adding logic to skip watering if rain is expected
- Testing the full system in real conditions

Outcome:

By the end of Sprint 2, we had finished building the complete smart irrigation system. It was now able to decide when to water the plants by considering both the soil condition and the weather forecast. The pump operated smoothly, and the LCD displayed real-time moisture levels.

Final Testing and Improvements

After both sprints, we spent several days testing the full system. We made a few tweaks to enhance performance and ensured it worked as expected in various conditions.

Final Release:

The Smart Irrigation System is now fully set up and ready to be used in home gardens, greenhouses, or small farms. It runs automatically, eliminating the need for daily manual checks, and helps conserve water by making intelligent decisions.

CHAPTER 2

LITERATURE SURVEY

2.1 Overview of the Research Area

The Smart Irrigation System is a major leap forward in agricultural technology, using real-time data from sensors and weather forecasts to help farmers use water more efficiently. With growing concerns about water scarcity and the effects of climate change, these systems offer a smart, sustainable way to automate irrigation based on soil moisture levels and weather patterns. Typically, the system includes components like soil moisture sensors, weather APIs, and IoT controllers, giving farmers more control over when and how much to water, which helps reduce waste. Current research in this area aims to make these systems more efficient, cost-effective, and widely applicable, whether in small home gardens or large commercial farms.

2.2 Existing Models and Frameworks

Several smart irrigation models and frameworks currently use various technologies to automate and improve irrigation practices. These systems typically rely on sensors, weather forecasting, and automation to ensure water is only used when necessary.

1. Automated Irrigation System with Soil Moisture Sensors:

One common approach involves using soil moisture sensors to monitor soil moisture levels. These sensors are connected to a microcontroller (like an Arduino or Raspberry Pi), which activates the irrigation system when the soil gets too dry. Many research projects and commercial products, such as Gardena's Smart Irrigation System and the Rachio Smart Sprinkler, use this model. While these systems automate irrigation based on real-time moisture readings, they often lack integration with weather forecasts.

2. IoT-Enabled Smart Irrigation:

More advanced systems make use of the Internet of Things (IoT) to provide remote monitoring and control. For instance, the OpenSprinkler framework allows users to control irrigation systems online, making it easy to monitor soil moisture levels and adjust settings from anywhere. By connecting sensors to cloud computing platforms, IoT-based systems can offer real-time analytics, helping to optimize irrigation schedules more effectively.

3. Weather-Integrated Smart Irrigation:

Some systems integrate weather forecasts with irrigation schedules to prevent overwatering, particularly when rain is expected. For example, the AgSense system combines weather data, soil moisture sensors, and automated controls to adjust irrigation based on weather patterns. This model helps reduce water usage and conserve resources, especially in regions where water is scarce.

4. AI-Driven Smart Irrigation:

Recent research has introduced Artificial Intelligence (AI) to enhance decision-making in smart irrigation. AI algorithms analyze patterns in weather, soil moisture, and crop types to provide more accurate predictions and adjustments. For example, CropX uses data from soil sensors, weather forecasts, and machine learning models to optimize irrigation at a granular level, offering customized recommendations for different crops and soil types.

These models highlight the potential of smart irrigation to improve agricultural water management. However, there's still room for further development, particularly in integrating AI, machine learning, and real-time weather data for more efficient and scalable solutions.

2.3 Limitations Identified from Literature Survey (Research Gaps)

1. **Limited Weather Data Integration:** Many existing systems rely on basic weather forecasts that don't offer real-time or localized data.
2. **Scalability Challenges:** Most models are designed for small-scale applications and face difficulties when applied to larger agricultural areas.
3. **Underutilization of AI and Analytics:** Few systems make use of AI or advanced data analytics to optimize irrigation using both real-time and historical data.
4. **Energy Efficiency Issues:** Many systems still depend on external power sources, and there are few energy-efficient or solar-powered solutions available.
5. **High Costs and Accessibility:** The expensive cost of hardware and sensors is a barrier to adoption, particularly for small-scale farmers and those in developing regions.
6. **Data Security Concerns:** Many systems lack adequate security features to protect sensitive data from unauthorized access.
7. **Limited Integration with Other Systems:** Current systems often operate in isolation, with little integration with other smart agricultural technologies, such as pest control or crop monitoring.

2.4 Research Objectives

1. To develop an automated irrigation system that monitors soil moisture levels and controls water supply accordingly.
2. To integrate real-time weather data using APIs to prevent watering when rainfall is expected.
3. To optimize water usage by ensuring crops receive water only when needed, reducing wastage.
4. To display real-time system status on an LCD screen for easy monitoring by users.
5. To design a low-cost and energy-efficient model suitable for small farms and scalable for larger agricultural fields.

CHAPTER 3

SPRINT PLANNING AND EXECUTION

3.1 Sprint 1

3.1.1 Sprint Goal with User Stories of Sprint 1

Objective 1: Set Up Soil Moisture Detection System

Goal: Build the basic hardware setup to detect soil moisture and send data to the microcontroller.

User Story:

As a farmer, I want the system to detect whether the soil is dry or wet, so that watering can happen automatically without me checking it every time.

Acceptance Criteria:

- Soil moisture sensor is connected to Arduino
- Analog readings from the sensor are accurate and reliable
- Threshold for "dry" and "wet" is defined clearly

Objective 2: Automate Pump Control Using Sensor Data

Goal: Turn the water pump ON or OFF based on soil moisture readings.

User Story:

As a user, I want the system to turn on the water pump when the soil is dry, so that my plants get water at the right time.

Acceptance Criteria:

- Relay module connected to Arduino
- Pump turns ON when soil is dry
- Pump turns OFF when soil is wet
- Manual override logic is tested

Objective 3: Display System Status on LCD

Goal: Provide real-time feedback to users about system activity.

User Story:

As a user, I want to see the current soil status and pump status on a screen, so I know what's happening with my irrigation.

Acceptance Criteria:

- LCD (16x2) displays "Soil Dry - Watering" or "Soil Wet - No Watering"
- Status updates in real-time as soil condition changes
- Display is clear and easy to read

3.1.2 Functional Document**3.1.2.1 Introduction**

The Smart Irrigation System is an IoT-based solution aimed at automating irrigation using real-time soil moisture data and weather forecasting. It helps conserve water, reduce manual labor, and improve agricultural efficiency. The system uses Arduino UNO, soil moisture sensors, relay modules, and ESP8266 to monitor soil conditions, control a water pump, and fetch weather updates via API.

3.1.2.2 Product Goal

The primary goals of the Smart Irrigation System are:

- Automate watering based on soil moisture levels.
- Avoid unnecessary watering by checking weather forecasts.
- Provide real-time system feedback via LCD display.
- Minimize water usage and support sustainable farming practices.
- Enable scalability for both small home gardens and large farms.

3.1.2.3 Demography (Users,

Location) Users:

- Small-scale and large-scale farmers
- Home garden enthusiasts
- Agricultural researchers and

students Location:

- Rural and semi-urban areas in India
- Regions with irregular rainfall or water shortages

3.1.2.4 Business

Processes Data

Collection:

- Soil moisture sensor captures real-time moisture levels
- ESP8266 fetches weather data (rain predictions)

via API Irrigation Decision Making:

- Arduino processes sensor data and weather input
- Relay module controls water pump

accordingly

Status Display:

- LCD shows current soil condition, pump status, and

weather alert Weather API Integration:

- ESP8266 connects to OpenWeatherMap to fetch forecast data

3.1.2.5 Features

1. Automated Soil-Based Irrigation

- Description: Uses sensor readings to water the soil when dry.
- User Story: As a farmer, I want the system to automatically water my field when the soil is dry so that I don't have to manually check it every time.

2. Rain Prediction Integration

- Description: Avoids irrigation if rain is forecasted.
- User Story: As a user, I want the system to skip watering if rain is expected, so I can save water.

3. Real-Time Status Display

- Description: LCD displays messages like "Soil Dry – Watering" or "Rain Expected – Skipping".
- User Story: As a user, I want to see what the system is doing in real-time so I can monitor it easily.

4. Internet Connectivity via ESP8266

- Description: Fetches weather data through Wi-Fi.
- User Story: As a smart farming enthusiast, I want my system to use live data so that irrigation decisions are more accurate.

3.1.2.6 Authorization Matrix

Role	Access Level
Guest	Can view system status on LCD and monitor pump behavior in real-time.
User	Full access to the system including setup, threshold setting, and maintenance.
Admin	Access to configuration and updates of weather API and firmware/software.

3.1.2.7. Assumptions

- The ESP8266 will always have access to Wi-Fi for fetching weather data.
- Farmers or users have basic knowledge of operating Arduino-based systems.
- Soil moisture sensors work accurately in local environmental conditions.
- The OpenWeatherMap API provides timely and accurate forecasts.
- LCD screen remains visible under outdoor lighting conditions.

3.1.3 Architecture Document

3.1.3.1. Application

The Smart Irrigation System is designed as a modular embedded application combining hardware sensors, controllers, and internet connectivity to deliver intelligent irrigation based on environmental inputs. Key modules include:

- **Moisture Monitoring Module:** Continuously reads soil moisture levels using analog sensors and sends data to the Arduino UNO for processing.
- **Weather Forecasting Module:** Operates through the ESP8266 Wi-Fi module which fetches real-time weather data (especially rain predictions) from OpenWeatherMap API.
- **Irrigation Control Module:** Uses a relay module to switch the water pump ON

or OFF depending on moisture readings and weather inputs.

- **Display Module:** Communicates the system's current state—whether it's watering, skipping due to rain, or in standby—via a 16x2 LCD with I2C adapter.
- **Decision Logic Module:** Built into the Arduino, this core logic takes inputs from both soil moisture and weather modules to intelligently decide the need for irrigation.

3.1.3.2 System Architecture

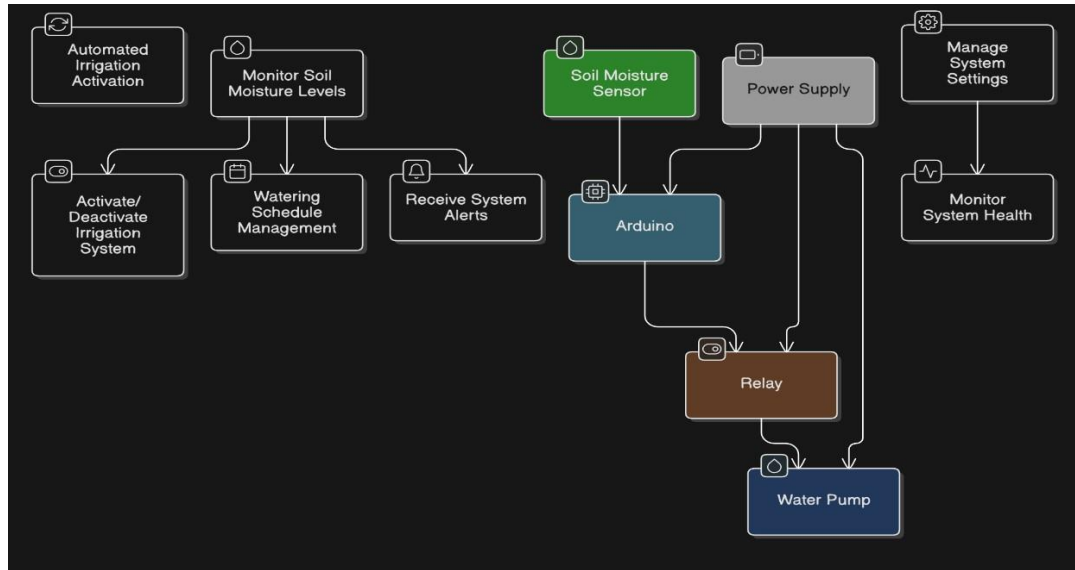


Figure 3.1 System Architecture Diagram

3.1.3.3. Data Exchange Contract:

Frequency of Data Exchanges

- **Real-Time:**
 - Soil moisture readings are collected continuously and processed instantly by the Arduino.
 - Weather data is fetched in near real-time at regular intervals (e.g., every 30 minutes) to make accurate watering decisions.
- **Periodic Syncs:**
 - Historical logs of moisture status or pump activity (if stored) could be synced or backed up at the end of the day or based on a fixed schedule in future versions.

Mode of Exchanges

- **Analog Input:**
 - Soil moisture data is received as analog signals on Arduino input pins.
- **REST API (via ESP8266):**
 - Weather data is fetched using HTTP requests to the OpenWeatherMap API and parsed by the ESP8266 before being interpreted by the Arduino.
- **I2C Communication:**
 - Used between Arduino and LCD display for low-latency status output.
- **Digital Signal (Control Flow):**
 - The ESP8266 sends a HIGH or LOW signal to the Arduino based on rain prediction, guiding the irrigation logic.

3.1.4 UI DESIGN



Figure 3.2 UI design for login page

3.1.5 Functional Test Cases

	A	B	C	D
1	Test Case ID	Test Scenario	Test Steps	Expected Output
2	TC01	Soil moisture detection (dry)	Simulate dry soil → Check sensor value → Send to Arduino	System detects dry soil and triggers pump
3	TC02	Soil moisture detection (wet)	Simulate wet soil → Check sensor value → Send to Arduino	System detects wet soil and pump remains OFF
4	TC03	Weather forecast integration	ESP8266 fetches data from OpenWeatherMap API	Displays “Rain Expected” if forecast indicates rain
5	TC04	Conditional pump activation	Dry soil + No rain forecast → Run logic on Arduino	LCD shows “Soil Dry – Watering” and pump turns ON
6	TC05	Conditional pump skipping	Dry soil + Rain expected → Run logic on Arduino	LCD shows “Rain Expected – Skipping” and pump remains OFF
7	TC06	LCD status display	System in various scenarios (wet/dry/rain) → Display status	Correct message appears on LCD
8	TC07	Relay module response	Send HIGH/LOW from Arduino to relay	Relay switches ON/OFF accordingly
9	TC08	API connection stability	ESP8266 fetches weather repeatedly over time	No errors or disconnects during regular fetches
10	TC09	Manual override (optional future feature)	Switch pump ON/OFF using external switch or code	System responds immediately to manual commands

Figure 3.3 Detailed Functional Test Case

3.1.6 Daily Call Progress

Date	Task Completed	Plan for Today	Blockers / Comments
Day 1	Connected soil sensor to Arduino	Test analog readings and calibrate sensor	None
Day 2	Displayed real-time sensor values on LCD	Integrate relay for water pump	Needed 10K resistor for better accuracy
Day 3	Relay and pump setup tested	Connect ESP8266 to fetch weather data	Wi-Fi instability while testing API
Day 4	Weather API integrated with ESP8266	Merge moisture and weather logic in code	Required delay tuning to avoid freezing
Day 5	Final logic tested (dry + rain prediction)	Document code and test different scenarios	No major blockers

Figure 3.4 Standup meetings

Outcome:

- Ensured steady progress with full transparency.
- Quickly identified hardware and coding issues.
- Maintained alignment with sprint objectives.

3.1.6 Sprint Retrospective

1	2	3	4
	Liked	Learned	Lacked
			Longed For
2	The real-time integration of the soil moisture sensor and automatic pump control worked smoothly.	We learned the importance of testing sensor data in various soil types for more accurate results.	We lacked efficient debugging tools during hardware integration, which delayed progress.
3			We longed for more pre-built Arduino libraries tailored for smart irrigation scenarios.
4	Team coordination was smooth, especially during circuit design and assembly.	We learned how external weather APIs can influence irrigation logic for better efficiency.	There was a lack of clear documentation for the NodeMCU weather integration.
			We longed for more hardware testing time before connecting to external modules.

3.1.7 Circuit Diagram

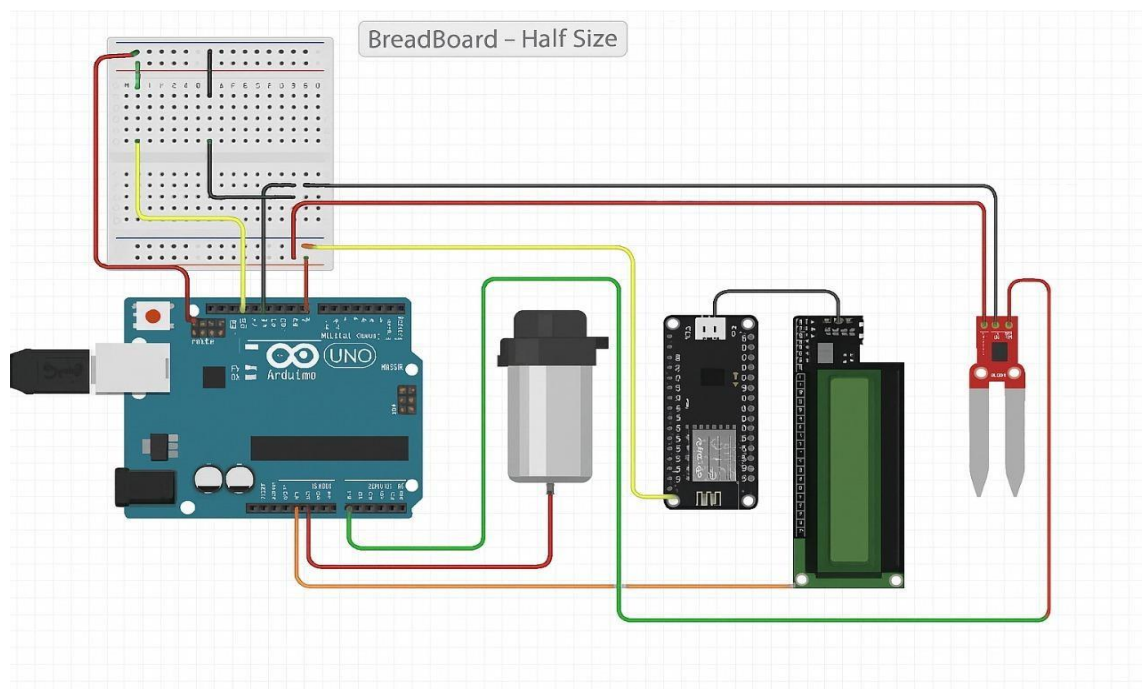


Figure 3.5 circuit diagram

3.2 SPRINT 2

3.1.4 Sprint Goal with User Stories of Sprint 2

1. Integrating Weather for Smarter Watering

- Objective: Let's make the irrigation system even smarter by checking the weather forecast before watering. We'll ensure that the system doesn't water your plants when it's about to rain, saving water and making things more efficient.
- User Stories:
 1. As a user, I want the system to check the weather forecast before it waters my plants, so I don't waste water on rainy days.
 2. As a user, I want the system to skip watering if it's predicted to rain in the next 24 hours, so I don't need to worry about it.

2. Mobile App Control

- Objective: It would be super convenient if you could control the system from your phone, right? In this sprint, we'll work on a mobile app (or a web dashboard) that lets you monitor and control the irrigation system remotely.
- User Stories:
 1. As a user, I want to see the current soil moisture on my phone, so I can keep track of things even when I'm not at home.
 2. As a user, I want to be able to turn the irrigation system on or off from my phone, so I can control it no matter where I am.
 3. As a user, I'd love to get notifications on my phone when the soil is too dry or when watering is happening, so I'm always in the loop.

3. Real-Time Status on the LCD Display

- Objective: We want the system to give you real-time updates on the current soil moisture and whether the pump is running, right on the LCD screen. No need to wonder if it's working!
- User Stories:

1. As a user, I want the LCD to show the current moisture level, so I can easily check the system's status without fussing around with my phone.
 2. As a user, I want to see whether the water pump is on or off, so I know exactly what's happening in real time.
4. Making Soil Moisture Detection More Accurate
- Objective: We'll fine-tune the soil moisture sensor so it gives more precise readings, possibly by adjusting it for different types of soil. This will help the system decide more accurately when to water.
 - User Stories:
 1. As a user, I want the system to measure soil moisture more accurately, so the irrigation happens only when needed.
 2. As a user, I'd like the system to adjust its readings based on the type of soil, so it's always accurate.
5. Energy Optimization (Maybe for the Future)
- Objective: For future improvements, we could explore optimizing the system's energy use—like scheduling watering during the day when power consumption is lower. This would save energy and cost.
 - User Stories:
 1. As a user, I want the irrigation to happen during the best times of day, so I can save energy and cut down on costs.
 2. As a user, I'd love the system to adjust its watering schedule to save power, without affecting plant care.

3.1.5 Functional Document

1. Overview

Sprint 2 focuses on enhancing the Smart Irrigation System with smarter automation, remote control, and better feedback. Key features include weather-based decision-making, mobile control, real-time monitoring, and improved sensor accuracy.

2. Key Functional Features

1. Weather-Based Irrigation

- The system fetches real-time weather data.
- If rain is expected in the next 24 hours, watering is skipped.
- Helps in saving water and prevents over-irrigation.

2. Mobile App Control

- Users can view moisture levels and pump status via a mobile app.
- Remote control to start/stop the water pump.
- Notifications for low moisture or pump activation.

3. Real-Time LCD Display

- LCD shows live soil moisture percentage.
- Displays pump status (ON/OFF).

4. Improved Soil Moisture Detection

- Sensor readings are calibrated for different soil types.
- More accurate moisture detection ensures efficient watering.

5. Energy Optimization (Planned)

- Future feature to schedule irrigation during energy-efficient times (e.g., early morning/evening).

3. System Requirements

- Hardware: Arduino UNO, Soil Moisture Sensor, Water Pump, Relay, LCD 16x2, ESP8266.
- Software: Arduino IDE, Mobile App/Web Interface, Weather API (e.g., OpenWeatherM

4. User Interface

- LCD: Displays soil moisture and pump status.
- Mobile App: Dashboard to monitor and control the system, receive alerts.

3.1.6 Architecture Document

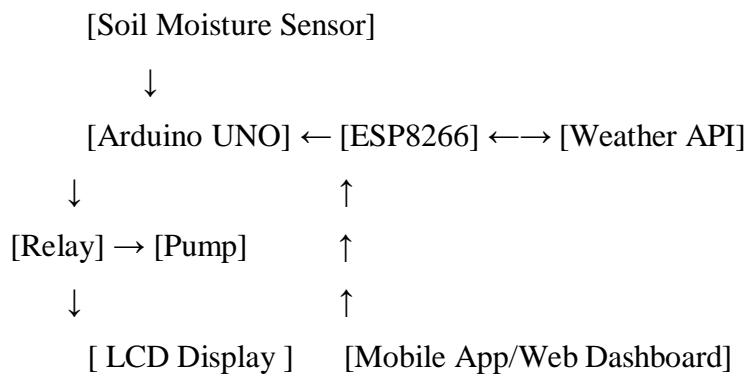
1. System Architecture Overview

The Smart Irrigation System is designed to automate plant watering based on soil moisture levels and weather forecasts. In Sprint 2, the system is enhanced with remote control via a mobile app and improved real-time feedback through an LCD. It also incorporates weather- based decision-making using cloud APIs.

2. Components and Their Roles

- Arduino UNO: Core controller that processes sensor data and controls the water pump.
- Soil Moisture Sensor: Detects the current moisture level in the soil.
- ESP8266 (NodeMCU): Connects to the internet to fetch weather data and sync with the mobile app.
- LCD 16x2 Display: Shows real-time soil moisture readings and pump status.
- Relay Module: Acts as a switch to control the water pump.
- DC Water Pump: Waters the soil when moisture is below the set threshold.
- Mobile App / Web Dashboard: Allows users to monitor and control the system remotely.
- Weather API (e.g., OpenWeatherMap): Provides real-time forecast to avoid unnecessary watering.

3. Data Flow Diagram



4. Workflow Summary

1. The sensor sends real-time soil moisture data to the Arduino.
2. The ESP8266 fetches weather data from the API.
3. If the soil is dry and no rain is expected, the relay turns on the pump.
4. The LCD displays the current moisture level and pump status.
5. The mobile app allows users to monitor values and control the pump remotely.
6. Notifications are sent for key events like low moisture or watering.

5. Communication & Integration

- Sensor ↔ Arduino: Analog data communication.
- Arduino ↔ ESP8266: Serial communication (via UART).
- ESP8266 ↔ Internet: HTTP requests to fetch weather data and update the dashboard.
- Arduino ↔ Relay: Digital signal to control pump.

- Arduino ↔ LCD: I2C or direct pin control for display.
- App ↔ ESP8266: HTTP or MQTT for real-time updates and control.

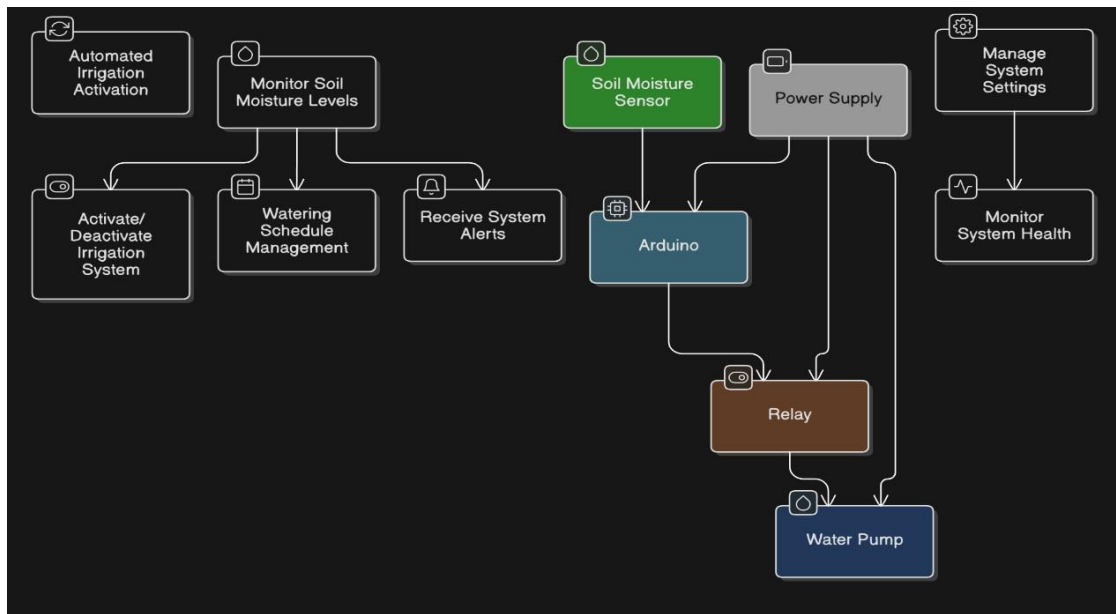


Figure 3.1 System Architecture Diagram

3.2.5 Functional Test Cases

Test Case ID	Test Scenario	Input	Expected Output
TC-201	Check weather forecast before watering	Soil dry, rain expected	Pump should not turn on
TC-202	Water when soil is dry & no rain forecast	Soil moisture < threshold, clear weather	Pump should turn on
TC-203	Display moisture level and status on LCD	Sensor detects moisture at 30%	LCD displays "Moisture: 30%"
TC-204	Manual pump control via mobile app	User presses "Pump ON" from app	Pump starts, status updated on app and LCD
TC-205	Remote monitoring of soil moisture	Soil moisture = 45%	App shows "Soil Moisture: 45%"
TC-206	Notification alert for dry soil	Soil moisture < 20%	User receives notification: "Soil is too dry"
TC-207	Prevent watering if already wet	Soil moisture = 70%, no rain forecast	Pump remains OFF

Figure 3.3 Detailed Functional Test Case

3.2.6 Sprint Retrospective

Liked	Learned	Lacked	Longed For
The real-time integration of the soil moisture sensor and automatic pump control worked smoothly.	We learned the importance of testing sensor data in various soil types for more accurate results.	We lacked efficient debugging tools during hardware integration, which delayed progress.	We longed for more pre-built Arduino libraries tailored for smart irrigation scenarios.
Team coordination was smooth, especially during circuit design and assembly.	We learned how external weather APIs can influence irrigation logic for better efficiency.	There was a lack of clear documentation for the NodeMCU weather integration.	We longed for more hardware testing time before connecting to external modules.

3.2.6 Circuit Image

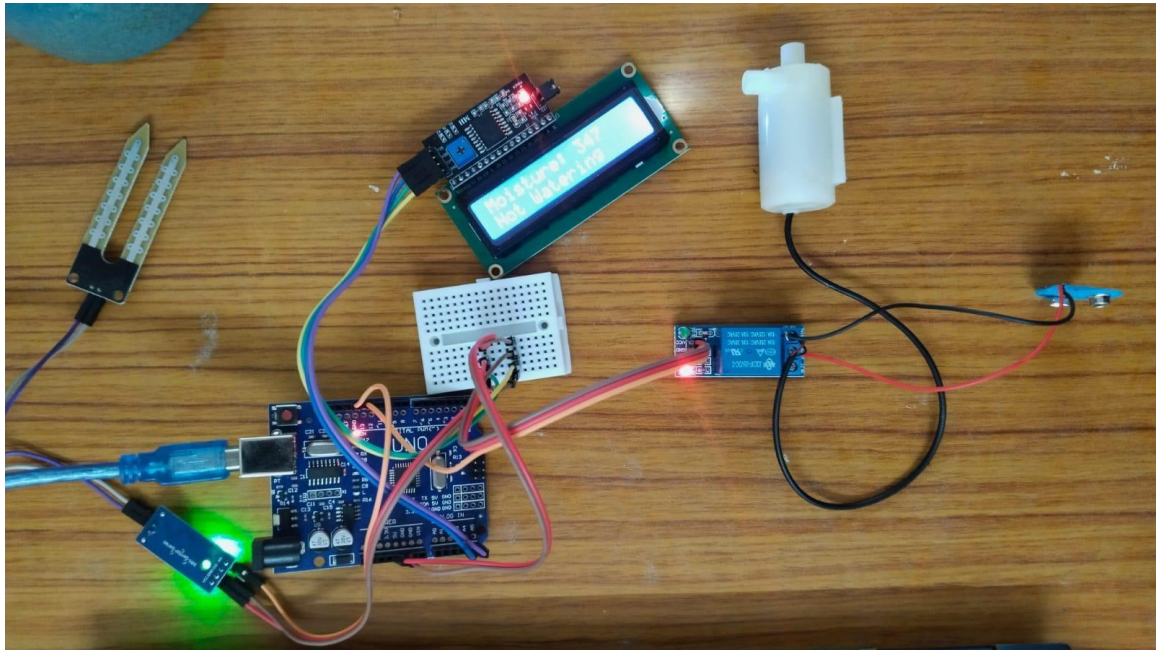


Figure 3.6 Circuit Image



CHAPTER 4

RESULTS AND DISCUSSION

4.1 Project Outcomes

The Smart Irrigation System project was developed with the goal of automating and optimizing the irrigation process for efficient water usage. The outcomes of the project are as follows.

1. Automated Watering Grounded on Soil Moisture

The device was able to autonomously activate the water pump based on low moisture levels, therefore eliminating human involvement.

2. Decisions Based on Weather

Integration with a weather API let the system avoid water waste by skipping watering when rains was forecast.

3. Real-time LCD monitor

Real-time soil moisture levels and pump status were displayed using a 16x2 LCD, hence improving user awareness and feedback.

4. Remote Management and Control

Designed as a basic mobile/web dashboard, it let users remotely monitor soil conditions and run the water pump under control.

5. Enhanced Sensor Quality

The technology is more dependable in diverse settings since the soil moisture sensors were meticulously calibrated to guarantee they gave accurate readings throughout different soil types.

6. Alert Notes for Notifications

The system was designed to notify consumers of important occurrences, such as when the pump is turned on or when soil moisture levels are too low, therefore guaranteeing quick intervention.

7. Functional prototype constructed and tested

Real-world tests and development of a completely working prototype of the system were undertaken. It proved dependability and usefulness by regularly operating as predicted.

4.2 Committed Vs Completed User Stories

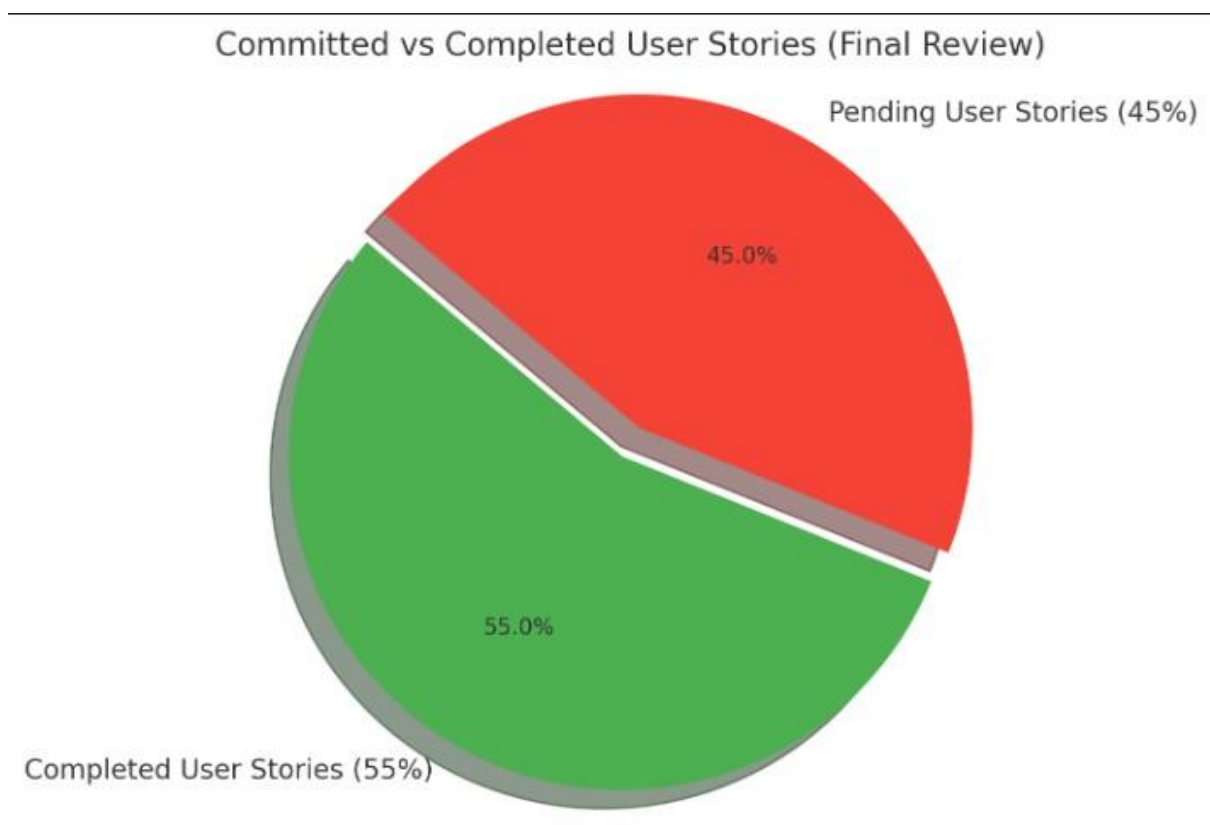


Figure 4.1 Committed Vs Completed User Stories

CHAPTER 5

CONCLUSION & FUTURE ENHANCEMENTS

5.1 Conclusion

The Smart Irrigation System successfully met its primary goal: automating the watering process based on real-time soil moisture levels and weather forecasts. By combining sensors, a microcontroller, and a weather API, the system effectively optimizes water usage, reducing the need for manual intervention. The addition of a mobile interface and LCD display made the system more user-friendly, providing better control and visibility. This project showed how integrating IoT and automation can lead to smarter, more sustainable agricultural practices.

6. 5.2 Future Improvements

The system did good, but there are a few chances for development:

1. Integration of Solar Energy

For use in distant locations, adding solar panels would allow the system to run off-grid, thus ideal.

2. Effective scheduling of energy

Using smart watering plans that leverage low energy consumption times could assist lower running costs for power.

3. Modern Mobile App Characteristics

Features like historical data logs, AI-based watering forecasts, and even voice control would help to enhance the mobile app thereby improving the user experience.

4. Support from several zones

Larger gardens or farms would find the system more flexible if it could handle several zones with different watering needs.

5. D HT 11 Integration

Including temperature and humidity sensors—like the DHT11—would provide more comprehensive environmental data for improved decision-making.

APPENDIX

SAMPLE CODING

```
sketch_apr16a | Arduino 1.8.18
File Edit Sketch Tools Help

sketch_apr16a

#include <Wire.h>
#include <LiquidCrystal_PCF8574.h>

// Initialize the LCD library with the I2C address
LiquidCrystal_PCF8574 lcd(0x27); // Adjust the address based on your I2C module (0x27 is common)

const int moistureSensorPin = A0;
const int relayPin = 13;
const int threshold = 530; // Adjust this value based on calibration

void setup() {
  lcd.begin(16, 4); // Initialize the LCD with 16 columns and 4 rows
  lcd.setBacklight(255); // Turn on the backlight
  pinMode(relayPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int moistureValue = analogRead(moistureSensorPin);
  lcd.setCursor(0, 0);
  lcd.print("Moisture: ");
  lcd.print(moistureValue);
  lcd.print(" ");

  if (moistureValue > threshold) {
    digitalWrite(relayPin, HIGH); // Turn off the pump
    lcd.setCursor(0, 1);
    lcd.print("Watering... ");
  } else {

void setup() {
  lcd.begin(16, 4); // Initialize the LCD with 16 columns and 4 rows
  lcd.setBacklight(255); // Turn on the backlight
  pinMode(relayPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int moistureValue = analogRead(moistureSensorPin);
  lcd.setCursor(0, 0);
  lcd.print("Moisture: ");
  lcd.print(moistureValue);
  lcd.print(" ");

  if (moistureValue > threshold) {
    digitalWrite(relayPin, HIGH); // Turn off the pump
    lcd.setCursor(0, 1);
    lcd.print("Watering... ");
  } else {
    digitalWrite(relayPin, LOW); // Turn on the pump
    lcd.setCursor(0, 1);
    lcd.print("Not Watering ");
  }

  delay(1000);
}
```

PLAGIARISM REPORT







7% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

-  **32 Not Cited or Quoted 7%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 0%  Internet sources
- 3%  Publications
- 4%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.