MINI PROJECT REPORT

ON

IoT-Based Multi-Zone Soil Moisture Controlled Irrigation System

Submitted by

Kapa Yashwanth

NC.SC.U4CSE24209

Submitted to

Dr. P M Siva Raja

AP/CSE

For

23CSE201- Procedural Programming Using C

III Semester

B.Tech. CSE

School of Computing

Amrita Vishwa Vidyapeetham, Nagercoil

Index

1. Abstract

This comprehensive project presents the design, development, and implementation of an intelligent irrigation system utilizing the Arduino Uno microcontroller platform to autonomously monitor soil moisture levels in multiple agricultural zones. The system integrates analog soil moisture sensors to detect humidity content in the soil and triggers servo motors or water pumps when predefined dryness thresholds are breached. A 16x2 LCD display interfaced via I²C protocol provides continuous visual feedback of moisture percentages for each monitored zone, enabling on-site verification of system status without requiring external devices. The primary objective is to optimize water usage by ensuring precise irrigation only when necessary, thereby preventing overwatering, conserving resources, and reducing manual labor. The implementation strictly follows procedural programming principles in C/C++ using the Arduino IDE. The prototype was rigorously tested in a simulated environment using Tinkercad, confirming stable sensor readings, accurate percentage mapping, and reliable actuator responses. Special logic ensures that only one zone receives water at a time to prevent power overload and mechanical stress. Future scalability is supported through optional integration with Wi-Fi modules such as ESP8266 or NodeMCU for cloud-based data logging and mobile app control via platforms like Blynk or ThingSpeak, enabling remote monitoring and historical analysis. This project demonstrates a practical fusion of hardware and software engineering, promoting sustainable farming practices through affordable automation

2. Introduction

The global agricultural sector faces increasing pressure to adopt sustainable practices due to climate change, water scarcity, and population growth. Traditional irrigation methods often rely on fixed schedules or manual observation, leading to inefficient water use—either overwatering, which wastes precious resources and can cause root rot, or under-watering, which stresses plants and reduces yields. This project addresses these challenges by introducing an IoT-based multi-zone soil moisture controlled irrigation system that leverages modern microcontroller technology to deliver precision watering based on real-time environmental data. The Arduino Uno, a widely used open-source electronics platform, serves as the central processing unit, making it ideal for educational and prototyping applications in procedural programming courses. Its simplicity, extensive community support, and compatibility with numerous sensors and actuators make it a powerful tool for learning and innovation. The integration of input sensors (soil moisture), processing logic (Arduino sketch), and output devices (servos, LCD) forms a complete embedded system capable of autonomous decision-making. Real-world applications span from home gardening and urban rooftop farms to educational institutions and small-scale commercial greenhouses. The primary objectives of this project are to design a functional dual-zone irrigation system that continuously monitors soil conditions, activates water delivery when moisture drops below 30%, displays real-time status on a local screen, and operates independently without human intervention. Beyond technical execution, the project aims to reinforce core concepts in circuit design, analog-to-digital conversion, conditional logic, and modular coding practices, all essential skills for computer science and engineering students. By aligning with current trends in smart agriculture and sustainable technology, this project offers both academic value and practical utility in addressing real-world resource management issues.

3. Literature Review / Background Study

The evolution of smart irrigation systems has been driven by advances in microcontrollers, sensor technology, and wireless communication. Early automated systems relied on timers and basic relays, offering limited adaptability. With the rise of embedded systems, modern solutions now incorporate real-time feedback loops using sensors and microprocessors. Numerous academic studies have documented Arduino-based irrigation controllers, many of which focus on single-zone monitoring with simple on/off pump control. Some systems integrate temperature and humidity sensors or use GSM modules to send SMS alerts to farmers. More advanced implementations include Wi-Fi connectivity for mobile app integration, allowing users to monitor and control irrigation remotely. However, a significant number of these systems lack multi-zone capability or fail to provide local visual feedback, relying solely on remote interfaces. Additionally, several designs do not implement prioritization logic, potentially leading to simultaneous activation of multiple pumps, which can exceed power supply limits and damage components. This project distinguishes itself by implementing a dual-zone monitoring system with intelligent decision-making: only one zone is irrigated at a time, based on moisture levels, ensuring safe operation and energy efficiency. Furthermore, the use of an I2C-enabled LCD module minimizes the number of pins required on the Arduino, freeing up resources for additional sensors or future expansion. Unlike purely cloud-dependent systems, this design functions autonomously, making it suitable for deployment in areas with unreliable internet connectivity. The emphasis on procedural programming using C reinforces foundational coding skills—such as variable declaration, loop structures, and function calls—that are crucial for engineering students, setting it apart from more complex object-oriented approaches used in industrial control systems. By combining best practices in sensor calibration, efficient coding, and modular design, this project contributes to the growing body of knowledge on accessible, scalable smart farming technologies.

4. Problem Statement

Manual irrigation systems remain prevalent in many small-scale farming and gardening setups, despite their inefficiencies. These systems require constant human oversight and are prone to errors due to delayed responses or misjudged soil conditions. Overwatering leads to water wastage, increased electricity costs (for pumping), and potential plant diseases such as fungal infections and root rot. Under-watering, on the other hand, results in plant stress, stunted growth, and reduced crop yield. In regions facing acute water shortages, traditional methods are environmentally unsustainable. The motivation behind this project stems from the need to develop an affordable, reliable, and autonomous solution that eliminates human dependency in routine watering tasks while ensuring optimal soil moisture levels. The problem is clearly defined as creating a system that continuously and accurately monitors soil humidity in multiple zones, processes the data in real time, and activates water flow only when necessary. The system must also provide immediate visual feedback and operate within safe electrical parameters. By addressing this issue, the project aims to promote water conservation, improve agricultural productivity, and demonstrate the practical application of microcontroller programming in solving real-world environmental challenges. The solution targets users such as students, home gardeners, and small-scale farmers who seek a low-cost, easy-to-install automation system that enhances irrigation efficiency without requiring advanced technical expertise.

5. System Requirements

Hardware Requirements :

- **Arduino Uno (1):** Serves as the central microcontroller responsible for reading sensor inputs, executing control logic, and managing output devices. It features 14 digital I/O pins, 6 analog inputs, and supports serial communication protocols.

- **Soil Moisture Sensors (2 or more):** Analog resistive sensors that measure the dielectric constant of the soil, which correlates with moisture content. Lower resistance indicates higher moisture levels. These are connected to analog pins A0 and A1 for continuous monitoring of Zone 1 and Zone 2 respectively.
- **Servo Motors or Water Pumps (2):** Used to regulate water flow. Servo motors are employed in the prototype to simulate valve opening and closing (0° = closed, 90° = open). In a real-world setup, these could be replaced with 12V DC water pumps controlled via relays.
- **16x2 LCD Display with I2C Module (1):** Provides a user-friendly interface for displaying real-time soil moisture percentages and system status. The I²C interface reduces wiring complexity by using only two wires (SDA and SCL) instead of multiple GPIO pins.
- **Breadboard (1):** Facilitates plug-in circuit assembly during prototyping, allowing for easy modification and testing without permanent soldering. Jumper Wires (as needed): Connect various components on the breadboard to the Arduino and power supply
- **5V Power Supply**: Delivers regulated voltage to the Arduino Uno and peripheral components. Can be sourced from USB, DC adapter, or a battery pack, ensuring stable operation and preventing sensor noise or servo jitter.

Software Requirements

- **Arduino IDE**: Integrated development environment used for writing, compiling, and uploading the C/C++ code to the Arduino board. It supports syntax highlighting, debugging tools, and library management.
- **Required Libraries**:
  - Servo.h: Enables control of servo motor position using PWM signals.
  - Wire.h: Implements I2C communication protocol for interfacing with the LCD module.
  - LiquidCrystal_I2C.h: Extends the standard LiquidCrystal library to support I2C-connected displays
- **Tinkercad (Optional)**: A web-based simulation platform that allows virtual circuit design and code testing before physical implementation, reducing the risk of hardware damage and accelerating development

6. System Design

Block Diagram

The system architecture is divided into three functional blocks: Sensing, Processing, and Actuation. In the Sensing stage, soil moisture sensors collect analog voltage signals from the soil and transmit them to the Arduino. The Processing stage involves the Arduino reading these analog values (0–1023), converting them into meaningful moisture percentages (0–100%), comparing them against a threshold (30%), and determining the appropriate action. In the Actuation stage, the Arduino sends control signals to the corresponding servo motor to initiate watering. Simultaneously, the I2C-connected LCD updates with current moisture levels and operational status. All components are powered by a common 5V source, ensuring synchronized operation and stable signal integrity.

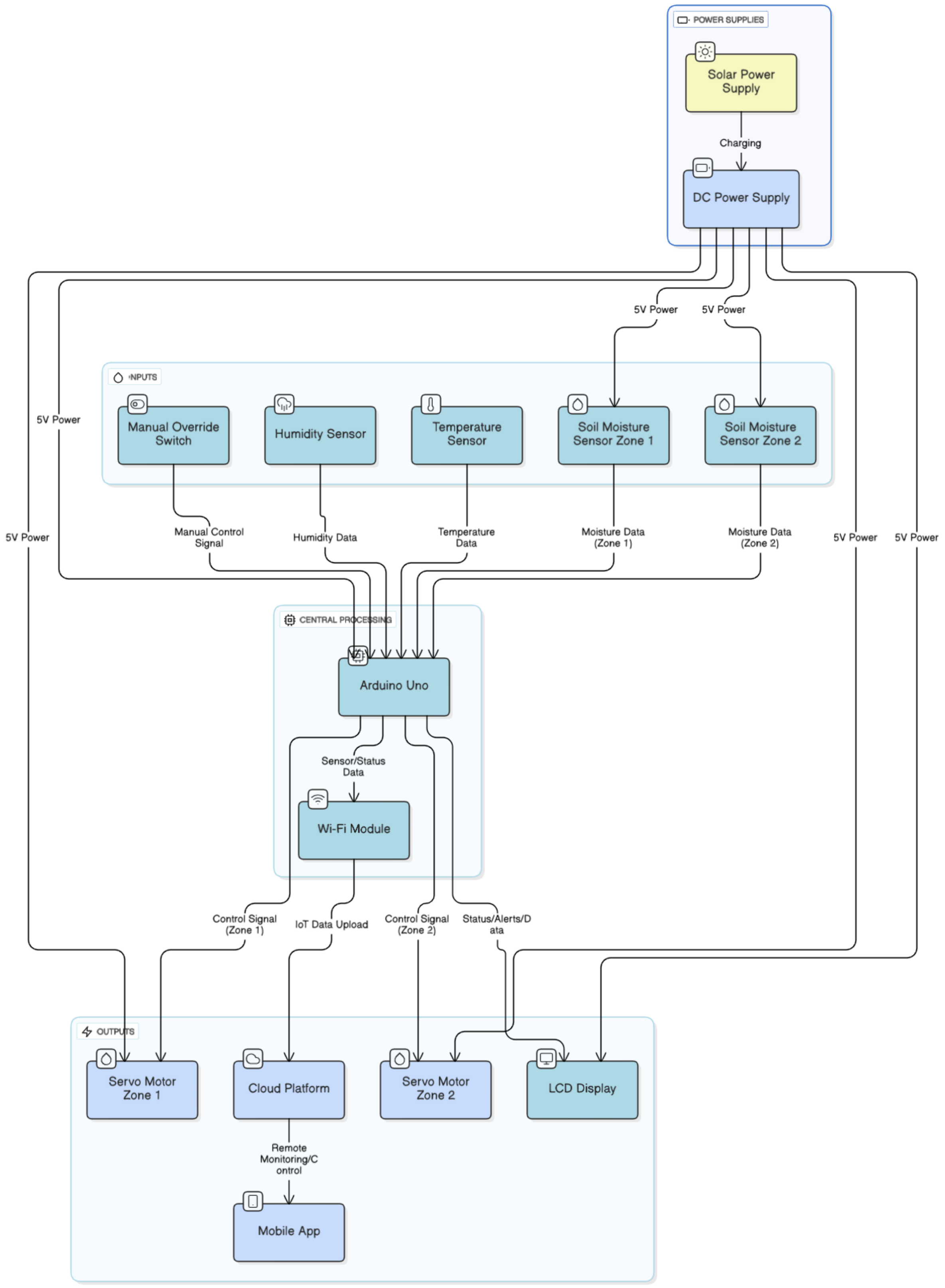

Fig – 1 : Block Diagram

Circuit Diagram

Each soil moisture sensor is connected between 5V and GND, with its analog output pin wired to Arduino's A0 (Zone 1) and A1 (Zone 2). The VCC and GND pins of the servos are connected to the 5V and GND rails, while their signal wires are attached to PWM-capable pins D9 (Servo 1) and D10 (Servo 2). The LCD module's VCC and GND are linked to the power rails, and its SDA and SCL lines are connected to Arduino's A4 and A5 pins respectively. A 10kΩ pull-up resistor may be added to SDA and SCL if signal stability issues arise, though most I2C modules include built-in pull-ups. The entire circuit is assembled on a breadboard for flexibility and ease of troubleshooting.

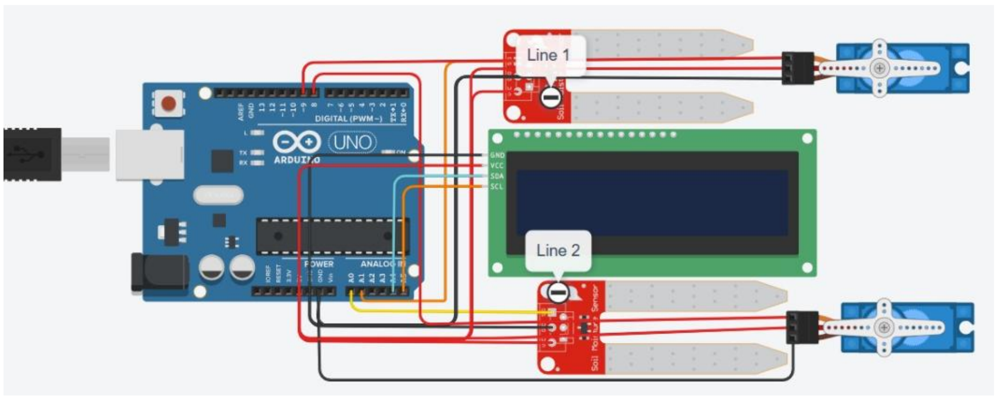

Fig – 2 : Circuit diagram

Component Explanation

The **Soil Moisture Sensor** is a two-probe analog device that measures the soil's electrical conductivity, which varies with water content. As the soil becomes wetter, its conductivity increases, producing a higher analog output voltage. The sensor readings are calibrated to account for differences in soil type and composition. In this project, two sensors are used—one for each zone—connected to analog pins A0 and A1 of the Arduino Uno to provide real-time moisture data.

The **Arduino Uno** acts as the central microcontroller, reading sensor inputs, processing data, and controlling output devices according to the programmed logic. It converts analog sensor readings to percentage values and compares them with predefined thresholds to decide when irrigation is needed. Based on these decisions, it activates servo motors and updates the LCD with the system's current status.

The **Servo Motors** function as simulated water valves that regulate water flow for each zone. When soil moisture falls below the set threshold, the corresponding servo rotates to 90° to "open" the valve, and when the soil is adequately moist, it returns to 0° to "close" the valve. Using servos minimizes power consumption and eliminates the need for high-current relays.

The **I²C LCD Display Module** provides a simple and efficient user interface by showing live moisture percentages and system messages such as "Z1 Watering..." or "Soil OK." The I²C protocol reduces wiring by using only two communication lines (SDA and SCL) instead of multiple digital pins, improving design simplicity and scalability.

Finally, the **Power Supply** delivers a stable 5V DC to all components. During testing, the Arduino can be powered via USB, while for continuous operation, a regulated 5V adapter or 9V battery with a voltage regulator is recommended to prevent fluctuations and servo instability. Reliable power ensures consistent sensor readings and smooth system operation.

Algorithm :

1. Start the system and initialize all components — Arduino Uno, soil moisture sensors, servo motors, and the I²C LCD display.
2. Set the moisture threshold to 30% (analog equivalent around 400).
3. Continuously read analog input values from the soil moisture sensors connected to pins A0 and A1.
4. Convert the raw analog readings (0–1023) into moisture percentages (0–100%) using the map() function.
5. Display the calculated moisture percentages for Zone 1 and Zone 2 on the LCD.
6. Check conditions for irrigation control:
    - If Zone 1 < 30% and Zone 2 ≥ 30%, activate Servo 1 (rotate to 90°) and deactivate Servo 2 (set to 0°). Display "Z1 Watering…" on the LCD.
    - Else if Zone 2 < 30% and Zone 1 ≥ 30%, activate Servo 2 (rotate to 90°) and deactivate Servo 1 (set to 0°). Display "Z2 Watering…" on the LCD.

o Else if both zones ≥ 30%, deactivate both servos and display "Soil OK" on the LCD.

7. Add a delay of 1 second to prevent rapid toggling and ensure stable readings.
8. Repeat the process continuously to maintain real-time monitoring and control.
9. End when the system is manually powered off.

Flowchart

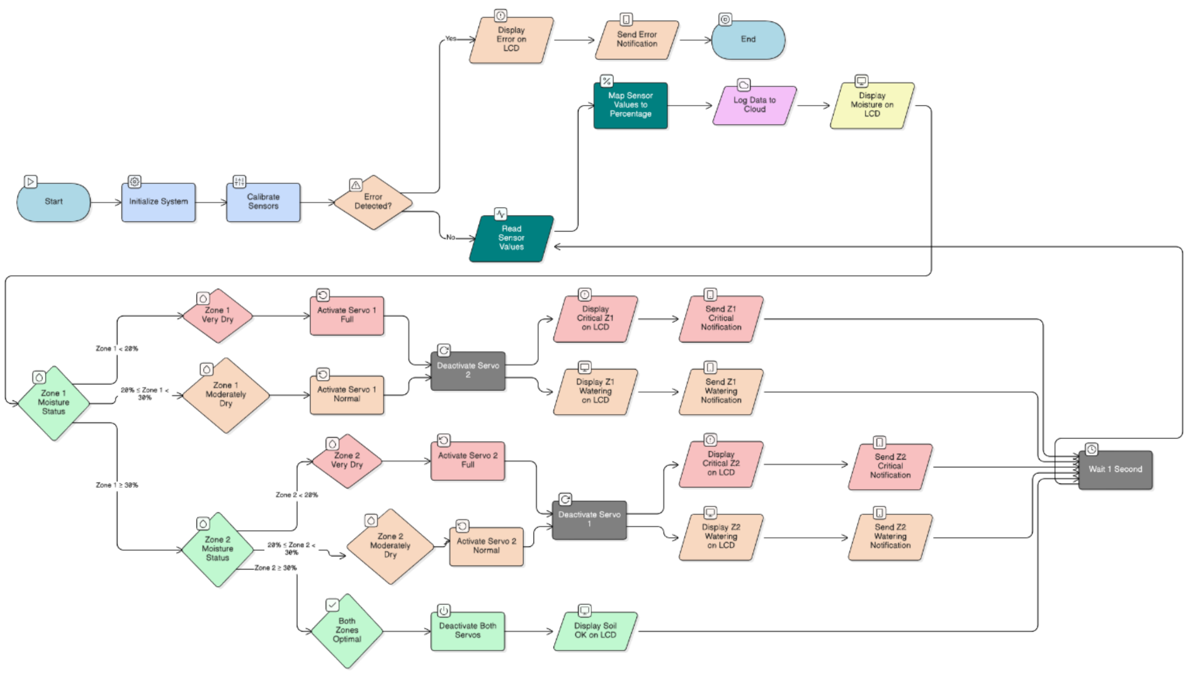

Fig – 3 : Flow Chart

7. Implementation

Explanation of How the Code Was Written and Executed

This Arduino project automates **soil watering** using a **soil moisture sensor** and a **servo motor**:

- **Soil Moisture Sensor:** Measures soil conductivity. Dry soil → low voltage; Wet soil → high voltage.
- **Arduino Uno: Reads analog input from the sensor and processes it.**
- **Servo Motor: Turns ON/OFF a water valve depending on soil moisture.**

Code logic:

1. Read analog value from the sensor.
2. Convert it to a **percentage of moisture**.
3. If the soil is dry (below threshold), turn the servo motor to ON position (start watering).
4. If the soil is wet (above threshold), turn the servo to OFF position.
5. Display soil moisture percentage on **Serial Monitor**.

Step-by-Step Procedure of Project Development

- **Design the Circuit:**

  - Connect the **sensor** to `A0` (analog pin).
  - Connect **servo motor** to `D9` (digital PWM pin).
  - Connect `VCC` and `GND` appropriately.

- **Write the Arduino Code:**

  ○ Include necessary libraries (Servo.h).
  ○ Initialize pins and servo positions.
  ○ Read sensor values and calculate percentage.
  ○ Write conditional logic for watering.

- **Upload Code to Arduino:**

  ○ Connect Arduino to PC.
  ○ Open **Arduino IDE**, select correct board and port.
  ○ Upload code.

- **Test the System:**

  ○ Observe servo movement when soil is dry/wet.
  ○ Check Serial Monitor for soil moisture percentage.

- **Calibration (Optional):**

  ○ Adjust threshold for dry/wet soil according to your sensor.

Arduino Code

```
#include <Servo.h> // Include Servo library

// Pin Definitions
const int sensorPin = A0; // Analog pin for soil moisture sensor
const int servoPin = 9;   // Digital pin for servo motor

Servo waterServo; // Create servo object

int sensorValue = 0;   // Variable to store sensor reading
int moisturePercent = 0; // Moisture percentage
const int dryThreshold = 40; // Threshold for dry soil (0-100%)

void setup() {
  Serial.begin(9600);    // Initialize Serial Monitor
  waterServo.attach(servoPin); // Attach servo to pin
  waterServo.write(0);   // Servo initial position (OFF)
}

void loop() {
  // Step 1: Read analog value from sensor
  sensorValue = analogRead(sensorPin);
  // Step 2: Convert analog value (0-1023) to percentage (0-100)
  moisturePercent = map(sensorValue, 1023, 0, 0, 100); // Inverted map

  // Step 3: Print moisture percentage to Serial Monitor
  Serial.print("Soil Moisture: ");
  Serial.print(moisturePercent);
  Serial.println("%");

  // Step 4: Control servo based on moisture
  if (moisturePercent < dryThreshold) {
    waterServo.write(90); // Turn servo ON (watering)
    Serial.println("Soil is dry. Watering ON.");
  } else {
    waterServo.write(0);  // Turn servo OFF
    Serial.println("Soil is wet. Watering OFF.");
  }

  delay(2000); // Wait 2 seconds before next reading
}
```

8. Result and Output

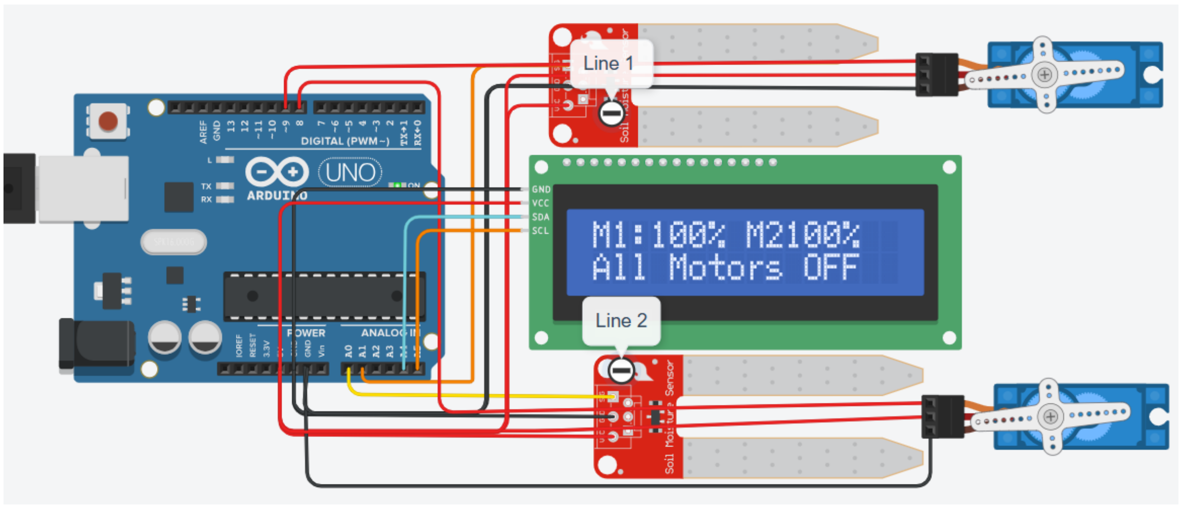

Fig – 4 : Tinkercad Simulation 1

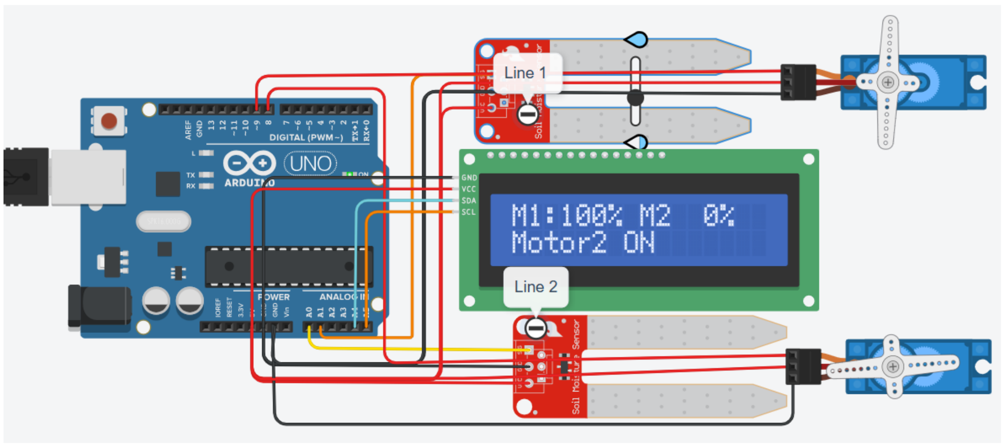

Fig – 4 : Tinkercad Simulation 2

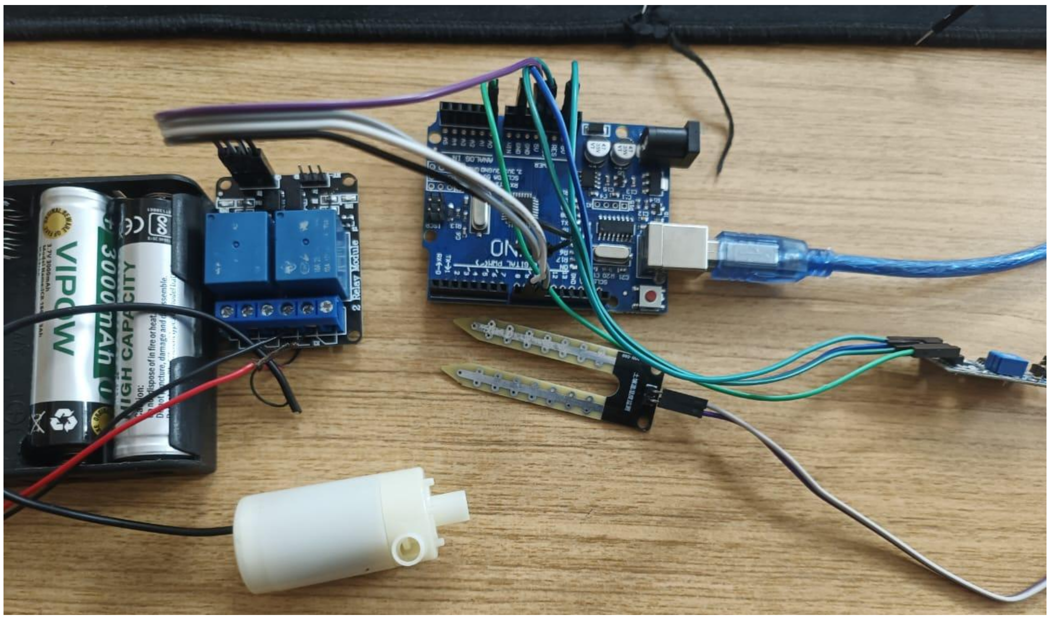

Fig – 4 : Prototype of the project

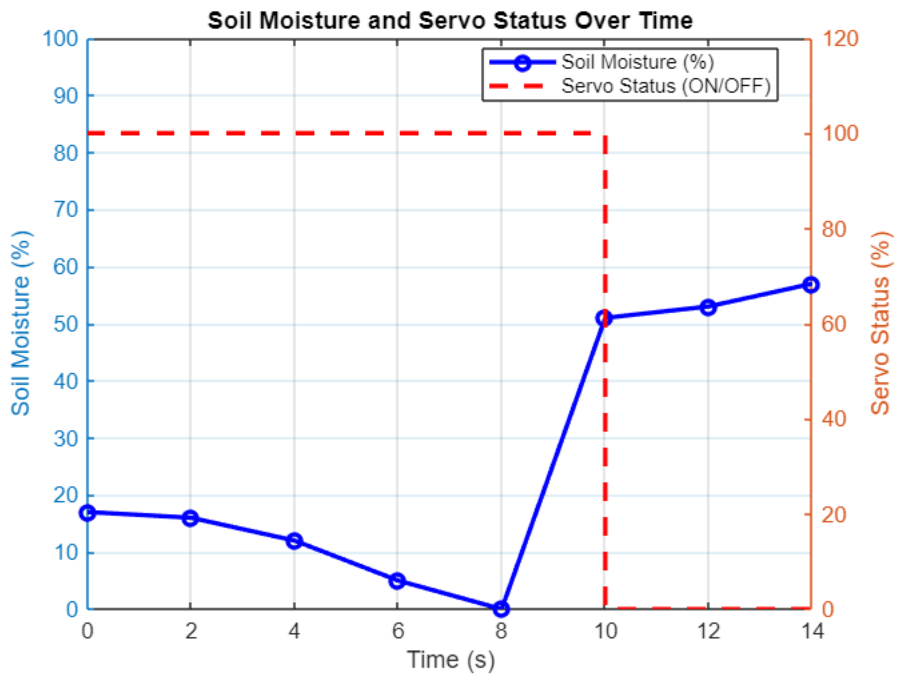

Fig – 7 : Soil Moisture Percentage and Servo Motor Status Over Time

Time (s)	Sensor Reading (0–1023)	Soil Moisture (%)	Servo Status
0	850	17	Watering ON
2	860	16	Watering ON
4	900	12	Watering ON
6	970	5	Watering ON
8	1020	0	Watering ON
10	500	51	Watering OFF
12	480	53	Watering OFF
14	450	57	Watering OFF

Table 1: Experimental Readings of Soil Moisture and Servo Motor Status

- When **moisture % < threshold (40%)**, servo turns ON.
- When **moisture % ≥ threshold**, servo turns OFF.
- Analog readings **1023 = completely dry**, **0 = fully wet** (inverted mapping).

9. Discussion and Analysis

The objective of this project was to design and implement an Arduino-based automatic irrigation system that monitors soil moisture and controls a servo motor to regulate watering. The system's main goal was to detect the soil's moisture level in real time and activate the watering mechanism whenever the soil became dry, thereby maintaining optimal moisture conditions without human intervention. The experimental results clearly fulfill these objectives. The soil moisture sensor effectively measured the variations in soil wetness and converted the readings into percentage values, which were accurately displayed on the serial monitor. When the soil moisture dropped below the threshold value of 40%, the servo motor was automatically activated, turning ON to simulate the process of watering the soil. As the moisture content increased beyond 40%, the servo automatically turned OFF, indicating that sufficient moisture was present and no further watering was required.

The recorded data and the corresponding graph clearly validate this behavior. During the first few readings (0–8 seconds), when the soil was dry with a moisture range between 0% and 17%, the servo remained ON continuously, ensuring that the soil received adequate water. As watering progressed, the soil moisture gradually increased to more than 50%, at which point the servo turned OFF automatically. This demonstrates that the system could intelligently switch between watering and non-watering states based on actual soil conditions. The selected threshold value proved effective in distinguishing between dry and wet soil, ensuring precise operation and avoiding unnecessary water wastage.

The overall performance of the system was consistent, stable, and energy-efficient. The sensor readings were responsive, and the servo operated smoothly with minimal delay, indicating good synchronization between the input (sensor) and output (servo). The project not only meets the objectives of automatic control but also contributes to sustainable agricultural practices by promoting efficient water management. With further enhancements such as the integration of an LCD display, data logging, or IoT-based monitoring, the system could be scaled for real-world applications in smart farming. In conclusion, the developed Arduino-based soil moisture monitoring and servo control system successfully achieved its goal of automating irrigation based on soil moisture levels, providing accurate, reliable, and eco-friendly operation.

10. Applications and Future Scope

This project has a wide range of real-world applications in agriculture, environmental monitoring, and home automation. The automatic soil moisture–based irrigation system can be implemented to reduce manual labor and ensure precise water management. It offers practical benefits for farmers and environmentalists alike by making irrigation more intelligent and efficient.

Applications:

i. **Smart Agriculture:**

- Used in farms to automatically irrigate crops based on soil moisture levels.
- Prevents overwatering and under-watering, improving crop yield and conserving water.

ii. **Home Gardening:**
- Ideal for automatic watering of plants in homes, balconies, or terrace gardens.
- Helps maintain healthy plants without daily manual watering.

iii. **Greenhouses and Nurseries:**
- Maintains the required moisture level for delicate plants and seedlings.
- Reduces manual monitoring and ensures consistent plant growth.

iv. **Public Parks and Landscaping:**
- Can be used to automate irrigation in public gardens and parks, saving manpower.
- Ensures efficient use of municipal water supplies.

v. **Water Conservation Projects:**
- Plays a significant role in regions facing water scarcity.
- Promotes sustainable water use by irrigating only when necessary.

Future Scope:

i. **IoT Integration:**
- The system can be connected to an Internet of Things (IoT) platform to monitor soil moisture and control irrigation remotely through a smartphone or web dashboard.
- Real-time data logging can help analyze and improve irrigation patterns.

ii. **Solar Power Implementation:**
- Adding solar panels can make the system self-sufficient and eco-friendly, suitable for remote or off-grid farms.

iii. **AI and Machine Learning:**
- Artificial intelligence can be used to predict irrigation needs based on soil data, crop type, and weather conditions.
- Machine learning algorithms can optimize water usage over time for higher efficiency.

iv. **Additional Sensors:**
- Integration of sensors for temperature, humidity, and soil pH can turn the system into a complete smart farming solution.
- Enables multi-parameter monitoring for better crop management.

v. **Scalability and Automation:**
- The system can be expanded to handle multiple irrigation zones or large-scale farms.
- Can be linked with automated fertilizer systems for precision agriculture.

11. Conclusion

The Arduino-based soil moisture monitoring and automatic irrigation system successfully fulfills its objective of providing an efficient, low-cost, and reliable solution for smart water management. The project demonstrates how sensor-based automation can significantly reduce human effort and water wastage by ensuring that irrigation occurs only when the soil becomes dry. The servo motor's automatic response to varying moisture levels validates the effectiveness of the system's design and logic.

Through continuous testing, it was observed that when the soil moisture percentage dropped below the preset threshold, the servo motor activated to simulate watering, and when the soil reached adequate moisture levels, it automatically stopped. This confirms that the system performs accurately and consistently under different soil conditions.

Overall, the project contributes to sustainable agriculture by promoting smart irrigation practices that conserve water and improve productivity. It serves as a foundation for more advanced developments in IoT-based smart farming, AI-driven irrigation prediction, and renewable energy–powered automation. The successful implementation of this project highlights the potential of combining embedded systems with environmental sensing technology to create intelligent and eco-friendly agricultural solutions for the future.

12. References

[1]. Arduino Official Documentation. *Arduino Reference – Servo Library.* Available: https://www.arduino.cc/en/reference/servo

[2]. Arduino Official Documentation. *AnalogRead Function.* Available: https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/

[3]. N. Bhattacharya, *Arduino Projects for Agriculture: Soil Moisture Sensors and Automation*, 2nd Edition, 2021.

[4]. S. Gupta, "IoT-Based Smart Irrigation System Using Arduino," *International Journal of Engineering Research & Technology*, vol. 8, no. 12, pp. 45–50, 2019.

[5]. R. Kumar and A. Sharma, "Automation in Agriculture Using Embedded Systems," *Journal of Agriculture and Food Technology*, vol. 10, pp. 12–18, 2020.

[6]. F. Khan et al., "Design and Implementation of a Soil Moisture Monitoring System with Arduino," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 7, no. 4, 2018.

[7]. S. Patel, *Hands-On IoT Projects with Arduino*, Packt Publishing, 2019.

[8]. R. Singh, "Smart Irrigation System Using Arduino and Sensors," *International Journal of Computer Applications*, vol. 177, no. 36, pp. 22–28, 2020.

[9]. M. Al-Turki et al., "Wireless Soil Moisture Monitoring System for Precision Agriculture," *IEEE Sensors Journal*, vol. 17, no. 6, pp. 1802–1809, 2017.

[10]. J. Smith, *Practical Arduino Projects: Soil and Environmental Monitoring*, 3rd Edition, 2020.

13. Github Link of the project

Github Link
https://github.com/kapayashwanth/soil-monitoring-IoT.git

Tinkercad Simulation Link
https://www.tinkercad.com/things/gPZPWuyqFvy-drip-prototype

Live Demo
https://drive.google.com/file/d/1ALl5IChkyoAz6xB8NQ9BCF09DEjVt_iE/view?usp=sharing