DESIGN OF A SMART IRRIGATION SYSTEM FOR AGRICULTURAL OPERATIONS

By

GROUP 11 MEMBERS (ROBOTICS PROJECT):

OLUEKE TESTIMONY (2024/13425)

EGBERINDE ENOCH (2024/13482)

ABDULLAHI AL-AMEEN (2024/13528)

ESAN AYODEJI (2024/13554)

OLEKA EBUBE (2024/13561)

ICT215 PROJECT

FINAL PROJECT REPORT

BELLS UNIVERSITY OF TECHNOLOGY, OTA

BY GROUP 11, 2025

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

_____

Contents

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to everyone who contributed, directly or indirectly, to the completion of this project. Special thanks go to my instructors and supervisors for their guidance, feedback, and support throughout the course of this work. Their insights and recommendations were invaluable in shaping both the technical and practical aspects of the project.

I also acknowledge the availability of open-source tools and documentation that made the design, simulation, and implementation processes possible. Finally, I am grateful to my peers for their encouragement and constructive discussions, which helped improve the overall quality of this project.

SMART IRRIGATION SYSTEM

CHAPTER ONE: INTRODUCTION

1.1 Background of the Study

With the advancement of embedded systems and low-cost microcontrollers, automated monitoring and control systems have become increasingly accessible for small-scale and educational applications. One important area of application is environmental and agricultural monitoring, where parameters such as soil moisture directly influence plant health and water efficiency.

Traditional irrigation methods often rely on manual judgment, which can lead to overwatering or underwatering. Microcontroller-based systems provide a more reliable approach by continuously monitoring environmental conditions and making control decisions automatically. Platforms such as the Arduino Uno have gained popularity due to their simplicity, affordability, and strong community support.

This project focuses on the design and implementation of an Arduino-based soil moisture monitoring system that uses an analog sensor input to control a relay and indicator LED. The system integrates analog sensing, embedded programming, and printed circuit board (PCB) design, making it suitable for learning and practical automation applications.

1.2 Problem Statement

Manual monitoring of soil moisture is inefficient, inconsistent, and prone to human error. In many cases, there is no immediate indication of when soil conditions fall outside acceptable limits. Additionally, beginners in embedded systems often struggle to integrate sensors, actuators, microcontroller code, and hardware design into a single functional system.

There is therefore a need for a simple, reliable, and well-documented system that demonstrates how soil moisture can be monitored automatically and used to trigger control actions using a microcontroller-based approach.

1.3 Objectives

General Objective

To design, simulate, prototype, and implement a functional microcontroller-based system that integrates analog sensing, control logic, and actuator output using an Arduino platform.

Specific Objectives

1. To design and simulate the complete system using Proteus.
2. To interface an analog soil moisture sensor with the Arduino Uno.
3. To write and implement microcontroller code for real-time monitoring and decision-making.
4. To control a relay and indicator LED based on sensor threshold values.
5. To design a professional PCB layout for the complete system.
6. To document the design, implementation, and results of the project.

1.4 Research Questions

1. How can soil moisture be measured using an analog sensor and an Arduino Uno?
2. How can threshold-based logic be implemented to control a relay output?
3. What are the challenges involved in transitioning from schematic simulation to PCB design?
4. How effective is a microcontroller-based system in automating basic monitoring tasks?

1.5 Significance of the Study

This project demonstrates the practical application of embedded systems in environmental monitoring and automation. It serves as an educational reference for students learning microcontroller programming, sensor interfacing, and PCB design. The system can also be extended for real-world applications such as automated irrigation, smart agriculture, and Internet of Things (IoT) systems.

Additionally, the project provides hands-on experience in integrating hardware and software into a single functional design.

1.6 Scope and Limitations

Scope

- The project is limited to monitoring soil moisture using a single analog sensor.
- Control action is implemented through a relay module and an LED indicator.
- The system is based on the Arduino Uno microcontroller.
- PCB design is included for demonstration and prototyping purposes.

Limitations

- The system does not include wireless communication or remote monitoring.
- Environmental factors such as temperature and humidity are not considered.
- The relay controls only a simulated or low-power external load in this implementation.

1.7 Organization of the Study

This report is organized into the following chapters:

- **Chapter One** presents the introduction, objectives, and scope of the study.
- **Chapter Two** reviews relevant concepts and related works.
- **Chapter Three** describes the system design and methodology.
- **Chapter Four** covers implementation, testing, and results.
- **Chapter Five** presents the conclusion and recommendations for future work.

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction

This chapter reviews related works and theories relevant to microcontroller-based automated systems, especially those that use soil moisture sensing and actuation. It covers existing systems similar to the current project, analogue circuit techniques in robotics and embedded control, the rationale behind microcontroller choice, the role of design tools like Proteus in engineering practice, and PCB design trends for embedded systems.

2.2 Review of Existing Similar Systems

Researchers and hobbyists have developed numerous automated irrigation and environmental monitoring systems using microcontrollers. For example, Ariwibowo detailed a system using an Arduino Uno to sense soil moisture and automatically switch the irrigation state based on predefined humidity thresholds, demonstrating the feasibility of low-cost soil moisture monitoring and control.

Hamoodi et al. presented a similar approach in which a photovoltaic-powered irrigation system relies on soil moisture readings to control a water pump via Arduino, ensuring water is supplied only when soil conditions require it.

A research project documented in the Automatic Irrigation System journal used Arduino to monitor moisture content and trigger pump control, integrating web and app interfaces for monitoring purposes, illustrating extended system architectures beyond basic control logic.

Prasojo et al. discussed the design of an automated watering system based on Arduino with application to agricultural environments, reinforcing the practical significance of automating irrigation using embedded sensors and controllers.

Sreesudha's work on a smart irrigation system using Arduino automation further emphasizes the advantages of using microcontrollers and sensor feedback to improve irrigation convenience and efficiency.

These examples show that soil moisture-based irrigation automation is a relevant and active research area, with implementations ranging from simple threshold control to IoT-augmented systems.

2.3 Analog Circuit Techniques in Embedded Systems

Analog circuit design remains critical in interfacing sensors with digital microcontrollers. Sensors such as soil moisture probes output analog voltages that must be read and interpreted by an ADC (Analog-to-Digital Converter). Embedded systems use conditioning circuits — resistive dividers, filters, and buffering — to prepare these signals for stable digital sampling. Since embedded controllers like the Arduino have limited ADC resolution (0–1023 for a 10-bit ADC), designers carefully scale and filter the signals to maximize sensitivity and reduce noise.

The project makes use of basic analog interfacing to read soil moisture levels, relying on the inherent ADC of the ATmega328P to convert the varying resistance (or voltage) into a digital value that drives relay actuation.

Furthermore, analog techniques such as thresholding and debouncing are widely used in robotics and control systems to interpret sensor signals and reduce erroneous triggers from spurious circuit noise. Studies of analog responsive robotics, such as BEAM robotics, highlight how simple analog circuits can be used for stimulus-response behaviors without significant computational overhead, offering lessons in efficient design.

2.4 Microcontroller Selection and Justification

Microcontrollers are at the heart of embedded systems and robotics. Among them, the Arduino platform — based on the ATmega328P — is widely chosen for its ease of programming, large community support, and sufficient I/O peripherals for typical sensor and actuator tasks. The ATmega328P provides analog inputs, digital outputs, timers, and good development tool support, making it ideal for prototyping and implementing simple autonomous systems such as soil moisture controllers.

While other microcontrollers (e.g., STM32 in advanced environmental monitoring systems) offer higher performance and networking capabilities, Arduino's simplicity and resources make it appropriate for the current system's objectives.

2.5 Use of Proteus in Industry and Education

Proteus remains a popular tool for schematic capture, simulation, and PCB design within both educational settings and small-to-medium engineering workflows. It allows designers to validate logic and component interaction before committing to hardware prototyping. Many institutions use Proteus to teach embedded systems and electronics because it supports microcontroller simulation with code, providing a safe, low-cost environment for students to test configurations.

While not a substitute for industry CAD tools like Altium or Eagle in large professional teams, Proteus's ability to integrate schematic design, simulation, and PCB layout makes it valuable for early design phases and teaching labs.

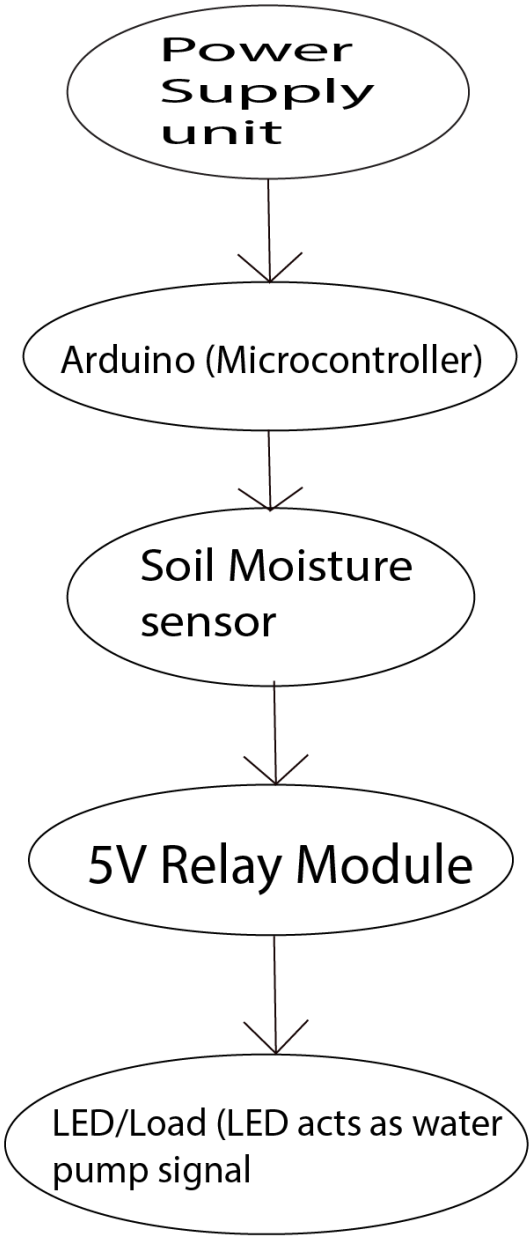
2.6 PCB Design Trends in Embedded Systems

PCB design has become an essential skill in embedded system engineering. Trends in PCB design emphasize compact layouts, signal integrity, modularity, and manufacturability. The shift toward "hardware-software co-design" encourages designers to consider PCB constraints, like trace routing and component placement, earlier in the system lifecycle. Creating professional PCBs — even for academic projects — reflects current expectations in industry and can improve a system's reliability and scalability.

In robotics and automation, PCBs are designed to handle both logic and power, requiring attention to grounding, power planes, and routing of analog vs digital signals, ensuring stable operation of sensors and actuators.

CHAPTER 3: METHODOLOGY

3.1 System block diagram



The system block diagram illustrates the functional interaction between the major components of the project. The soil moisture sensor provides an analog signal proportional to soil moisture level. This signal is read by the Arduino Uno microcontroller, which processes the data using a predefined threshold algorithm. Based on the decision logic, the Arduino controls a relay module that switches the connected load. Power is supplied to all system components via the Arduino's power input.

3.2 HARDWARE DESIGN

3.2.1 Analog Circuit Design

Introduction

Analog circuits play a critical role in this project because they provide a method for the Arduino microcontroller to receive and interpret continuous signals from a sensor representing soil moisture levels. Unlike digital signals, which have discrete levels (0 or 1), analog signals vary continuously over a range of voltages. Proper analog circuit design ensures accurate and reliable readings that can be processed by the microcontroller to make real-time control decisions.

In this project, the analog circuit interfaces the soil moisture sensing element (simulated by a potentiometer) with the Arduino Uno. The design converts changes in resistance, due to soil moisture variations, into measurable voltage changes that can be read by the Arduino's analog input pin.

Sensor Interface

The sensor used in this project is represented by a **potentiometer**, which acts as a variable resistor. The potentiometer has three terminals:

1. **VCC terminal** – connected to a stable 5V supply from the Arduino
2. **GND terminal** – connected to the common ground of the system
3. **Wiper terminal** – provides the variable voltage output based on the resistance ratio

The wiper terminal produces a voltage that is directly proportional to the simulated soil moisture level. Adjusting the potentiometer changes the resistance between the wiper and the two fixed terminals, producing a continuous range of voltages from 0V to 5V. This simulates the varying output voltage that would come from a real soil moisture sensor.

Voltage Divider Principle

The potentiometer is configured as a **voltage divider**, a common analog circuit technique used to produce a variable voltage from a fixed supply. The voltage at the wiper terminal can be described mathematically as:

$$V_{out} = V_{cc} \times \frac{R_{wiper}}{R_{total}}$$

Where:

- V_{out} is the voltage at the wiper (analog signal to Arduino)
- V_{cc} is the supply voltage (5V)
- R_{wiper} is the resistance from the wiper to ground
- R_{total} is the total resistance of the potentiometer

As the soil becomes drier (or the potentiometer is adjusted to simulate dryness), R_{wiper} increases, resulting in a lower V_{out}. Conversely, a wetter condition decreases R_{wiper}, increasing V_{out}. This provides a **direct analog representation** of soil moisture in voltage form.

Analog-to-Digital Conversion

The Arduino Uno reads the analog voltage via its **10-bit Analog-to-Digital Converter (ADC)**. The ADC converts the continuous voltage (0–5V) into a digital value ranging from 0 to 1023, according to the formula:

$$\text{ADC\_Value} = \frac{V_{out}}{V_{ref}} \times 1023$$

Where V_{ref} is the reference voltage voltage (5v in this case)

- **0V → ADC = 0**
- **5V → ADC = 1023**

This conversion allows the microcontroller to process a digital representation of the analog input and compare it against a predefined threshold to make control decisions.

Signal Conditioning

In this design, **no additional analog signal conditioning** (such as amplification, filtering, or buffering) is required because:

1. The potentiometer output is already within the acceptable voltage range of the Arduino's ADC (0–5V).
2. The 10-bit resolution of the ADC provides sufficient granularity for the application.
3. The low sampling frequency and stable supply voltage minimize the effect of noise on the analog signal.

In a real-world soil moisture sensor, capacitors or operational amplifiers might be added to smooth the signal and reduce noise, but for simulation and prototyping, this simple configuration is sufficient.

Threshold Detection and Control Logic

The analog signal from the sensor is continuously monitored by the Arduino. A **threshold value** is defined in the Arduino code to distinguish between "wet" and "dry" soil. The microcontroller performs the following logic:

- **If ADC value < threshold** → Soil is dry → Relay module and LED indicator activated
- **If ADC value ≥ threshold** → Soil is wet → Relay module and LED indicator deactivated

This threshold is adjustable in the code and can be calibrated based on experimental observation or desired moisture levels.

Component Integration

The analog circuit includes:

1. **Power Supply:** Provides stable 5V to the potentiometer and Arduino.
2. **Potentiometer (sensor simulation):** Produces variable voltage based on resistance.
3. **Voltage Divider:** Converts resistive changes into a proportional analog voltage.
4. **Arduino ADC Input:** Converts analog voltage to digital data for processing.

Proper grounding and power distribution ensure consistent operation and prevent measurement errors due to voltage drops or electrical noise.

Summary

The analog circuit design provides a reliable interface between the soil moisture sensor and the microcontroller. By using a potentiometer in a voltage divider configuration, the system produces a continuous voltage signal that accurately represents soil moisture levels. The Arduino's ADC converts this analog signal into digital values, which are then processed to control the relay and LED indicator.

This simple yet robust analog design forms the foundation of the system, enabling precise and automated decision-making, and demonstrates fundamental principles of analog circuit design in embedded control applications.

3.2.2 Microcontroller & Pin Mapping

Microcontroller Selection

The Arduino Uno microcontroller was selected for this project due to its availability, ease of use, and compatibility with both analog and digital devices. Its built-in analog-to-digital converter (ADC) allows direct interfacing with the soil moisture sensor, while its digital pins can drive the relay module and LED indicators. The Arduino Uno's 5V logic level matches the supply voltage of the connected components, ensuring safe and reliable operation.

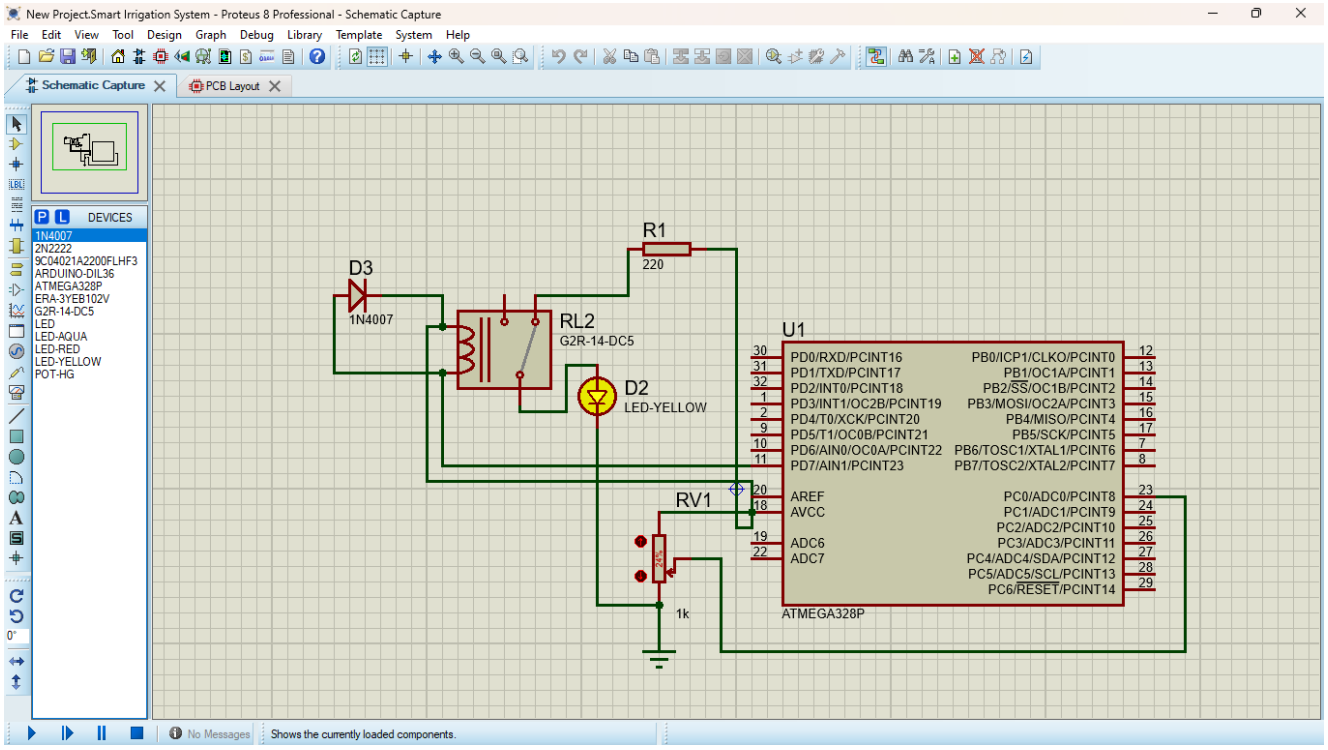
Pin Mapping

Component	Arduino Pin	Purpose/Reason
Soil moisture sensor (potentiometer wiper)	A0(Analog input)	Reads variable voltage representing soil moisture
Relay module control input	D7(Digital output)	Activates relay when soil is dry
LED indicator	D6(Digital output)	Provides visual indication of relay status
Power supply	5V	Powers Arduino and sensor circuitry
Ground	GND	Common reference for all components

Notes / Justification:

1. **A0 (PC0)** is used because the sensor produces a continuous voltage, requiring an analog input.
2. **D7** is chosen for relay control because it provides a digital HIGH/LOW signal to switch the module.
3. **D6** is used to drive an LED, giving immediate visual feedback about system status.
4. **5V and GND** provide consistent power and reference for all components, ensuring proper operation.

3.2.3 Complete Schematic in Proteus



The complete system schematic was designed and implemented using Proteus Design Suite. The schematic integrates all major hardware components, including the Arduino Uno microcontroller, soil moisture sensing element (represented by a potentiometer), relay module, LED indicator, and power supply connections.

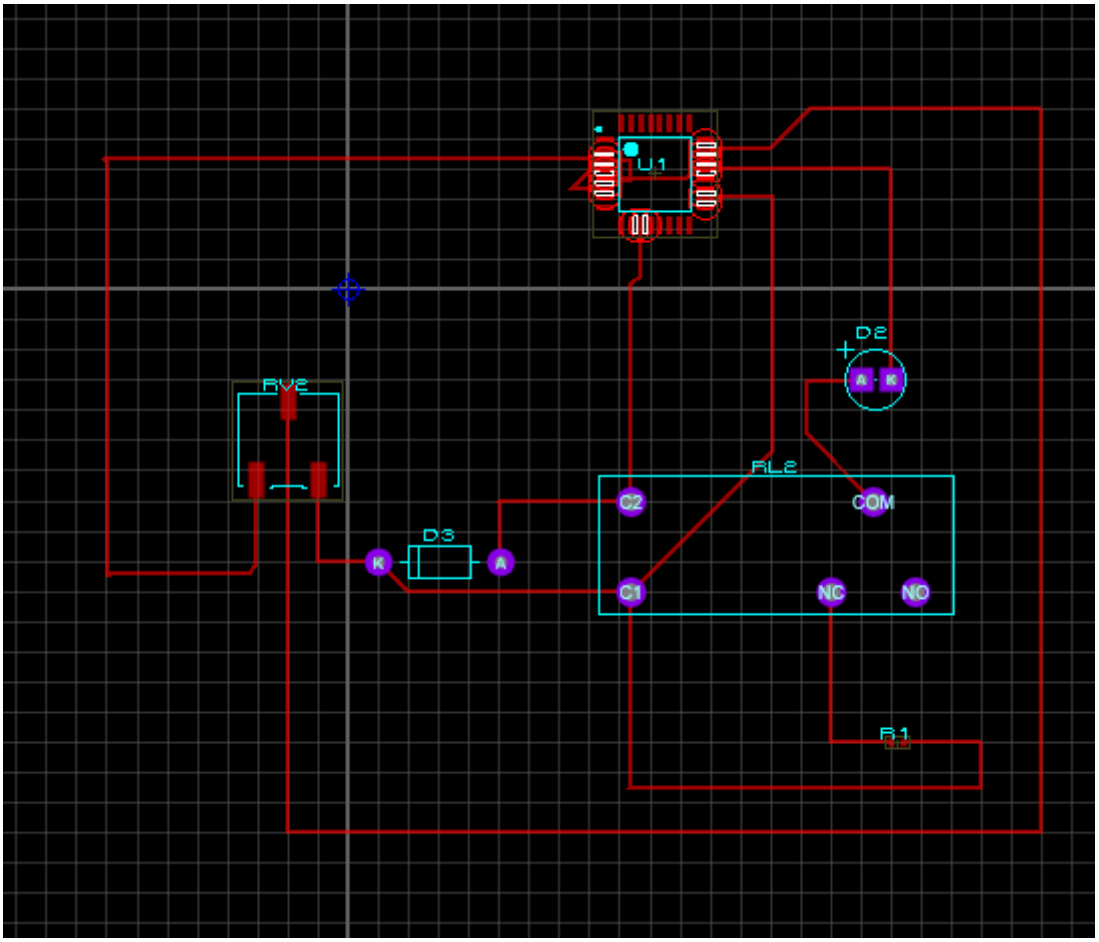
The analog output from the soil moisture sensor is connected to the Arduino's analog input pin, while digital output pins are used to control the relay module and LED indicator. Proper power and ground connections were established to ensure reliable operation. The schematic was designed to reflect the actual hardware implementation and served as the basis for both simulation and PCB layout development.

3.3 PCB Design

To enhance reliability and ensure a compact and professional hardware implementation, a printed circuit board (PCB) was designed for the project. The PCB design was carried out using Proteus Design Suite, following successful schematic design and simulation. The PCB integrates all major components of the system, including the microcontroller, sensor interface, relay module, and indicator LED.

3.3.1 PCB Layout Design

Component placement was carried out in the PCB layout environment of Proteus to minimize signal interference and ensure efficient routing. The Arduino microcontroller, relay module, sensor connections, and passive components were arranged to allow short and direct signal paths. Adequate spacing between components was maintained to reduce pad-to-pad violations and simplify routing.



3.3.2 PCB Routing

Following component placement, copper track routing was performed to establish electrical connections between components. The routing process was guided by the schematic netlist to ensure correct connectivity. Design Rule Check (DRC) was used to identify potential layout violations, and necessary adjustments were made to complete the routing process successfully.

PCB routing is the process of creating conductive paths (tracks) on the printed circuit board to electrically connect all components according to the schematic design. In this project, PCB routing was carried out after all components had been correctly placed on the board layout, ensuring that routing decisions were guided by both electrical and physical considerations.

The routing process began with careful observation of the net connection guides generated from the schematic. These guides indicated which pins were required to be electrically connected, serving as a reference for manual routing. Particular attention was given to ensuring that all power, ground, signal, and control lines followed their respective net assignments.

Manual routing was adopted instead of automatic routing to maintain greater control over track placement and to minimize unnecessary overlaps and sharp angles. Tracks were routed with adequate spacing to satisfy design rule constraints and to reduce the risk of short circuits or signal interference. Where possible, shorter and more direct routing paths were preferred to reduce resistance and improve signal integrity.

Special consideration was given to routing between closely spaced microcontroller pins. Due to limited spacing, multiple routing attempts were required to achieve clean connections without violating pad-to-pad clearance rules. In some cases, component repositioning was necessary to create sufficient routing space and improve overall layout clarity.

Power and ground connections were routed in a structured manner to ensure reliable voltage distribution across the board. Shared ground references were maintained to reduce potential noise issues and ensure stable operation of both analog and digital sections of the circuit.

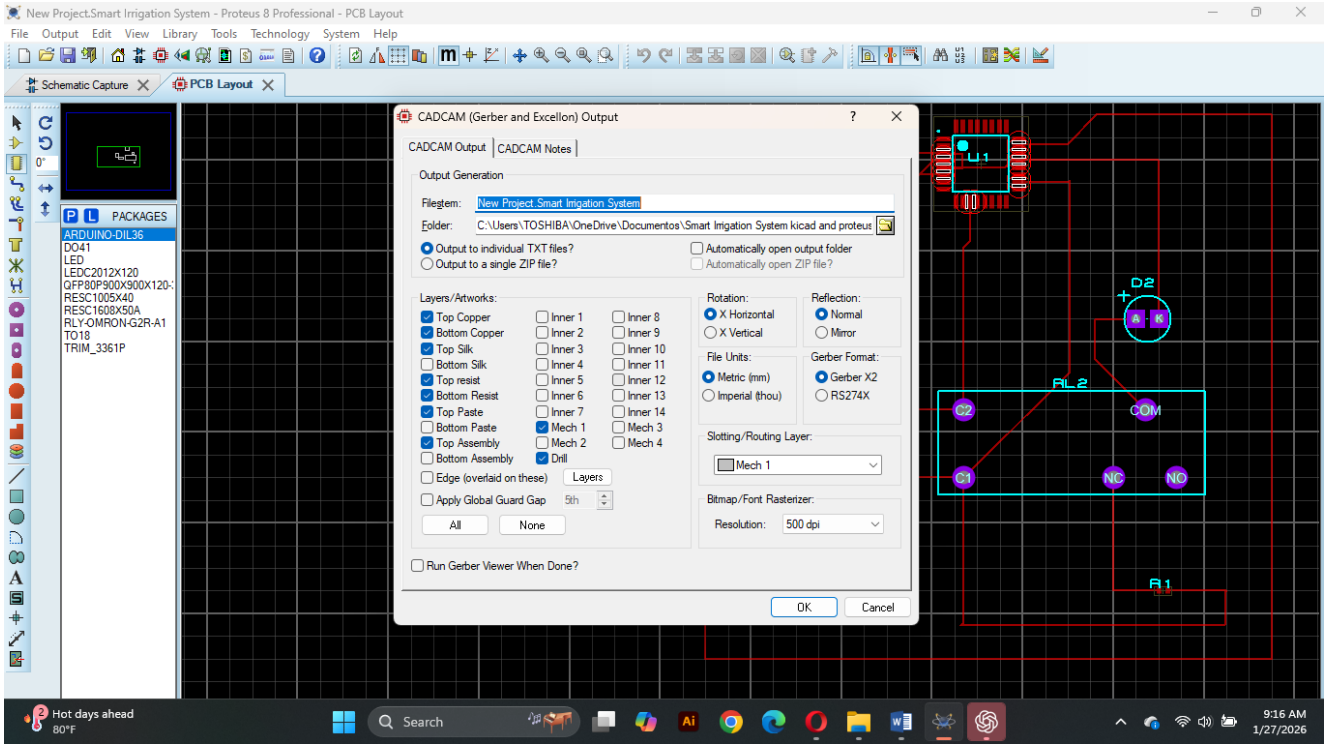
Throughout the routing process, frequent design rule checks (DRC) were performed to identify clearance violations, overlapping tracks, or incomplete connections. Identified issues were addressed by rerouting affected tracks and adjusting routing paths until all essential connections were successfully completed.

Although aesthetic optimization was not the primary objective, reasonable effort was made to keep routing organized and readable. This improves ease of debugging, future modifications, and manufacturability.

Upon completion, the PCB routing accurately reflected the intended electrical connections of the schematic and served as a reliable foundation for PCB fabrication.

3.3.3 Gerber File Generation

After completing the PCB layout and routing, Gerber files were generated from Proteus. These files contain all the necessary information required for PCB fabrication, including copper layers, drill files, and board outline. The generated Gerber files can be submitted directly to a PCB manufacturer for production.

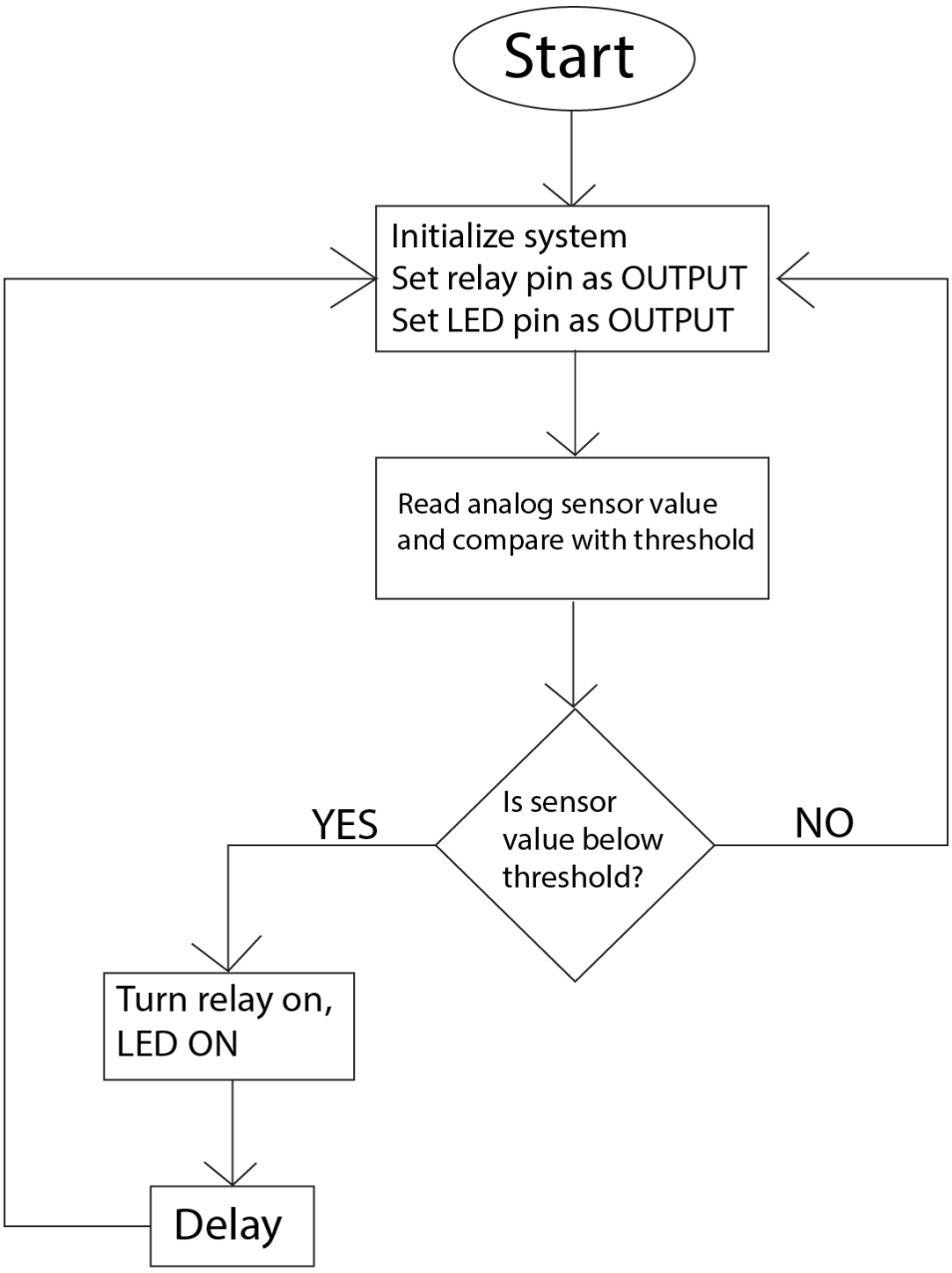


The image above shows the process of gerber file generation after PCB routing is completed. Gerber files prove manufacturability. The following gerber files are generated: top copper, bottom copper, top silk, top resist, bottom resist, top paste, top assembly, Mech 1, and drill.

3.4 Software Development

3.4.1 Program flowchart

The flowchart illustrates the logical sequence of operations performed by the Arduino microcontroller. The program begins by initializing the required input and output pins. The system then continuously reads the analog value from the soil moisture sensor and compares it with a predefined threshold. If the sensor value indicates dry soil, the relay and LED are activated. Otherwise, the relay remains deactivated. This process is repeated continuously to ensure real-time monitoring.


Start
Initialize system
Set relay pin as OUTPUT
Set LED pin as OUTPUT
Read analog sensor value
and compare with threshold
Is sensor
value below
threshold?
YES
NO
Turn relay on,
LED ON
Delay

3.4.2 Arduino Program Code

The complete Arduino program used to control the system is presented below. The code was written to read analog sensor data, process the input, and control the relay module accordingly.

This is the written code:

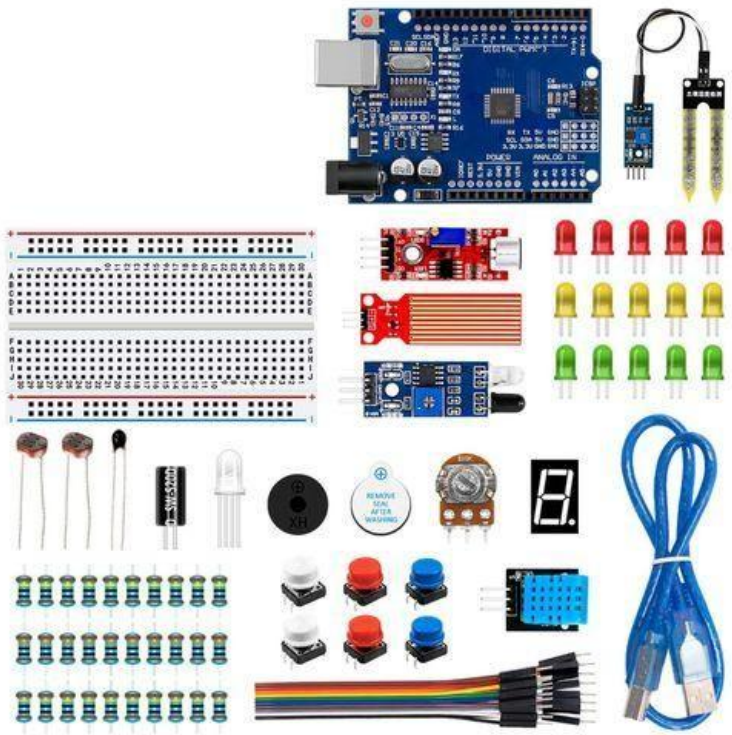
```
// Smart Irrigation System
// Written for Arduino Uno (ATmega328P)
// Components wired exactly as per schematic:
// Relay coil to PD7 and AVCC, LED via NO pin, Potentiometer to A0

// Pin definitions
const int relayPin = 7;       // PD7 on Arduino controls relay
const int ledPin = 6;         // PD6 connected to LED
const int sensorPin = A0;     // Analog input from potentiometer (soil moisture)

int sensorValue = 0;          // Variable to store analog reading
int threshold = 500;          // Threshold value for watering (adjustable)

// Setup runs once
void setup() {
  pinMode(relayPin, OUTPUT);  // Relay control pin as output
  pinMode(ledPin, OUTPUT);    // LED pin as output
  pinMode(sensorPin, INPUT);  // Potentiometer/soil sensor as input
}

// Main loop runs repeatedly
void loop() {
  sensorValue = analogRead(sensorPin); // Read soil moisture value

  if (sensorValue < threshold) {       // Soil is dry
    digitalWrite(relayPin, HIGH);      // Activate relay (water on)
    digitalWrite(ledPin, HIGH);        // Turn LED on
  } else {                             // Soil is wet
    digitalWrite(relayPin, LOW);       // Deactivate relay (water off)
    digitalWrite(ledPin, LOW);         // Turn LED off
  }

  delay(1000); // Wait 1 second before next reading
}
```

3.5 Prototype Development

The prototype development phase involves the physical implementation of the designed system based on the validated schematic, simulation results, and PCB layout. Due to project scheduling constraints, the physical prototype construction is planned for the subsequent phase of the project. However, the methodology for prototype development has been clearly defined.

The planned prototype development process consists of the following steps:

1. **Component Procurement:**
   All required electronic components, including the Arduino microcontroller, relay module, potentiometer (soil moisture sensor), LED, and jumper wires, will be sourced.
2. **Component Assembly:**
   The electronic components will be soldered onto the fabricated PCB according to the finalized PCB layout.
3. **Firmware Upload:**
   The developed Arduino program will be uploaded to the microcontroller using the Arduino IDE.
4. **System Testing:**
   The prototype will be tested by varying the soil moisture input and observing the relay and LED response.
5. **System Validation:**
   The physical behavior of the prototype will be compared with the simulation results to validate correct operation.



The image above shows an image of the Arduino Uno R3 DIY Kit to be used in prototype development. It features a breadboard, resistors, LEDs, jumper wires, the Arduino board itself (ATMEGA328P), et cetera.

The image on the left shows a 5V relay module, 1 channel. It is also used in the prototype development. It takes analog signals from the microcontroller and electrically controls the other device it's connected to in real time.

3.6 Testing and Validation Procedure

Testing and validation of the system were carried out primarily through simulation using Proteus Design Suite. The testing process focused on verifying the correct interaction between the sensor input, microcontroller logic, and output devices.

During simulation, the soil moisture sensor value was varied by adjusting the potentiometer to represent different soil conditions. The Arduino microcontroller's response was observed to ensure that the relay module and LED indicator were activated or deactivated correctly based on the predefined threshold.

Validation was achieved by confirming that the system behavior in simulation matched the expected control logic. Future validation will involve physical testing of the fabricated PCB and assembled prototype, where real sensor inputs and external loads will be used to confirm consistency between simulation and real-world operation.

3.7 Ethical and Safety Considerations

Ethical and safety considerations were taken into account during the design and planning of this project. The system was designed to operate at low voltage levels, reducing the risk of electrical hazards during testing and operation. Electrical isolation provided by the relay module ensures safe separation between the control circuit and external loads.

The project does not involve the collection of personal data, thereby raising no privacy concerns. Additionally, care was taken to ensure that all design decisions follow standard engineering practices, including proper grounding and component selection.

During future physical implementation, appropriate safety measures such as careful soldering practices and safe handling of power sources will be observed to prevent damage to components and ensure user safety

CHAPTER 4: RESULTS AND DISCUSSION

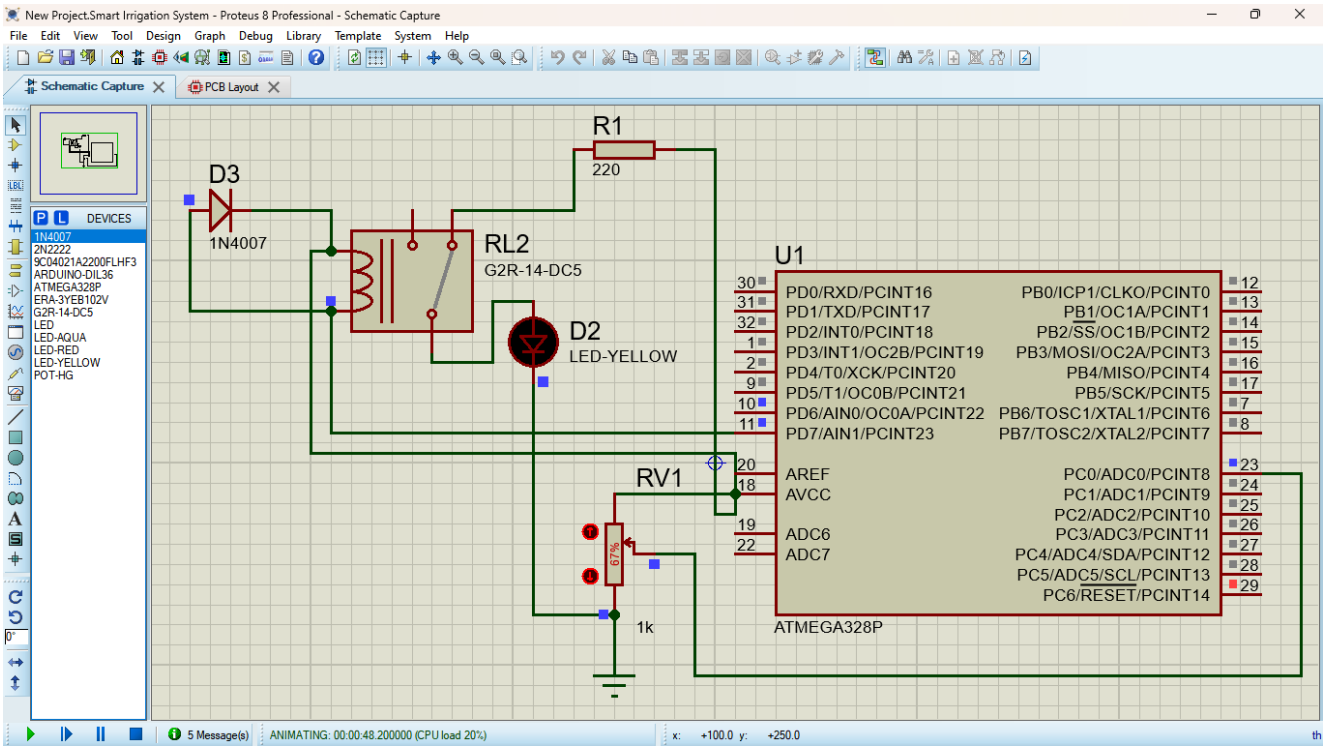
4.1 Introduction

This chapter presents and discusses the results obtained from the simulation of the designed system using Proteus Design Suite. The results demonstrate the functional behavior of the system based on varying soil moisture conditions and validate the effectiveness of the hardware and software design.

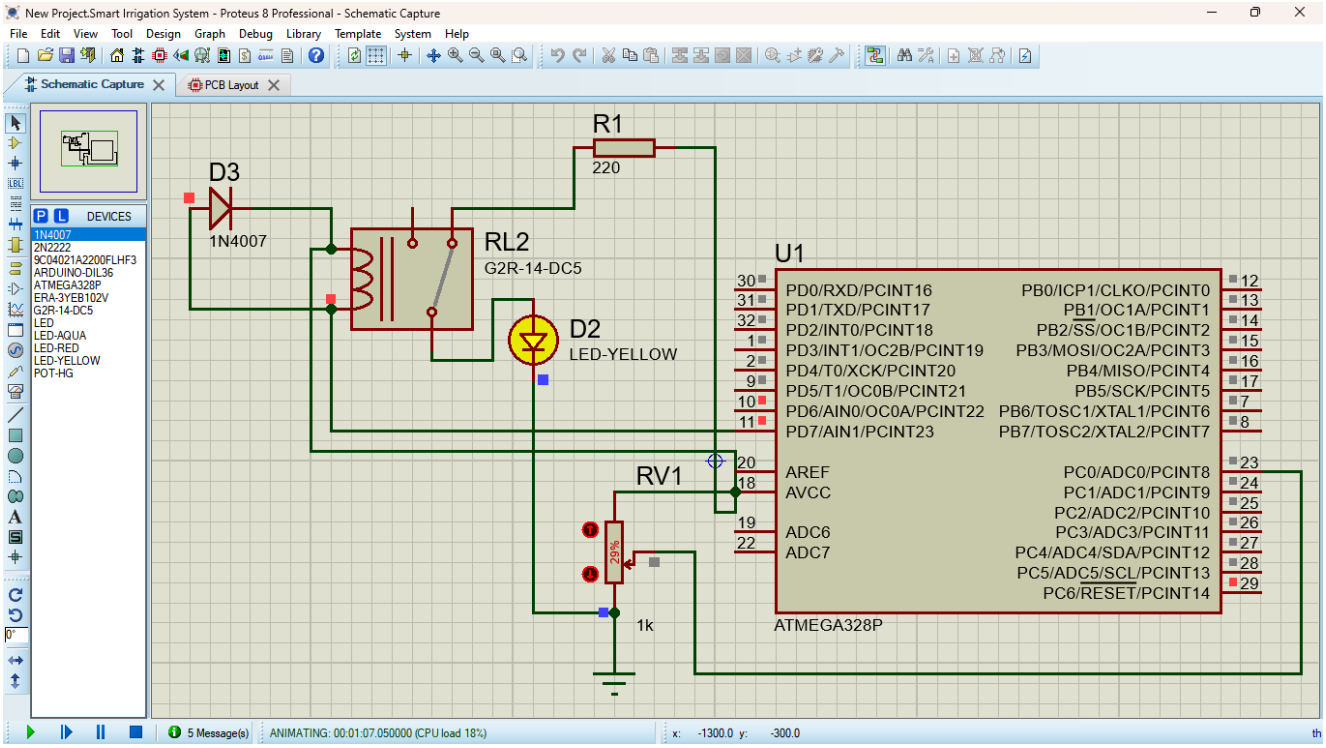
4.2 Simulation Results

The system was tested in the Proteus simulation environment by varying the soil moisture sensor input, which was represented using a potentiometer. Different resistance values were applied to simulate dry and wet soil conditions. The Arduino microcontroller successfully read the analog input and responded according to the programmed control logic.

When the simulated soil moisture level fell below the predefined threshold, the relay module was activated, and the LED indicator turned on. Conversely, when the moisture level exceeded the threshold, the relay was deactivated, and the LED turned off. These results confirm that the system operates as intended under simulated conditions.

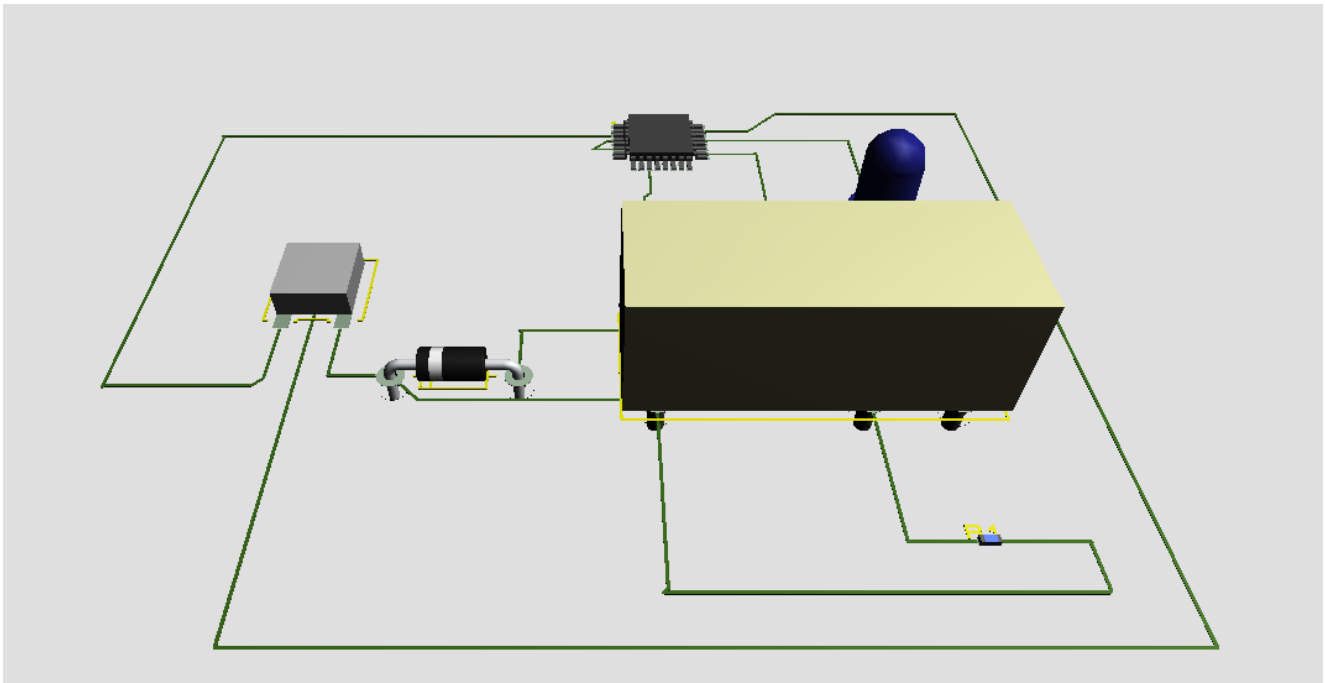


The image above is a screenshot of the schematic design being simulated in proteus. It shows that the potentiometer reading is above threshold, indicating that the soil moisture is within the optimum/desired range. Hence, the microcontroller doesn't electrically switch the relay on and the LED, indicating PUMP, is off.

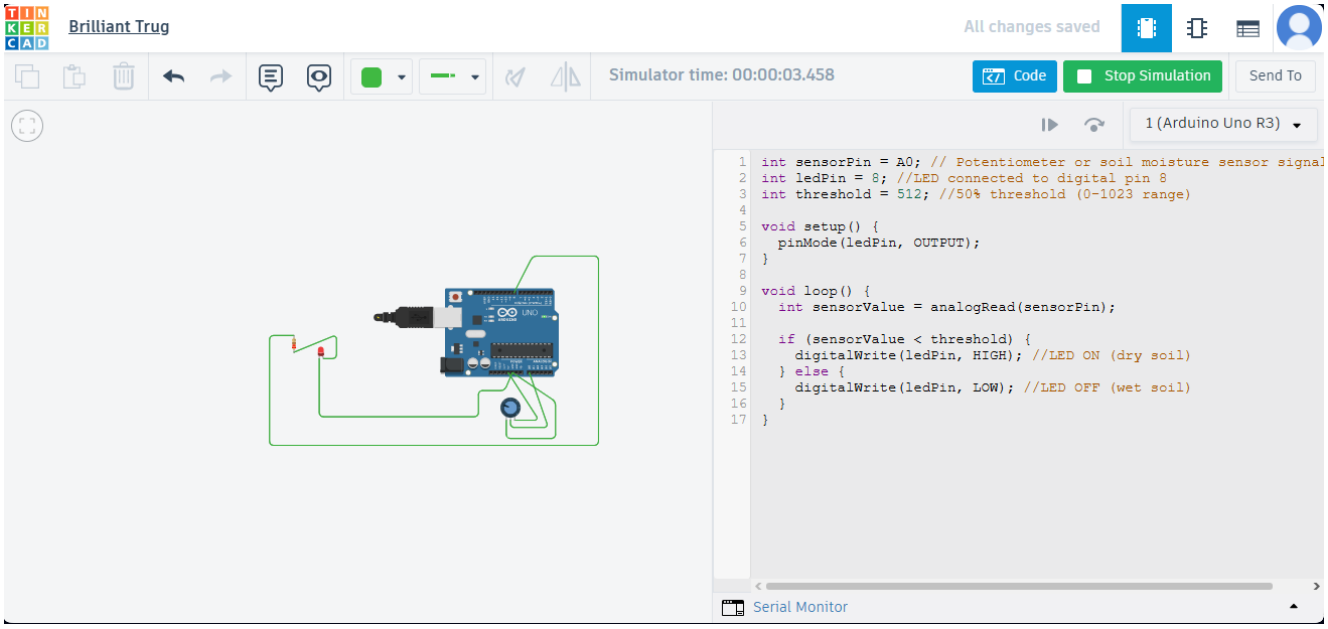


The image above is also screenshot during simulation, but this time, it shows the potentiometer reading below threshold, indicating that it's not in the optimum/desired range and will need some moisture. Hence, the microcontroller electrically switches the relay/pump on, and the LED lights up as shown.

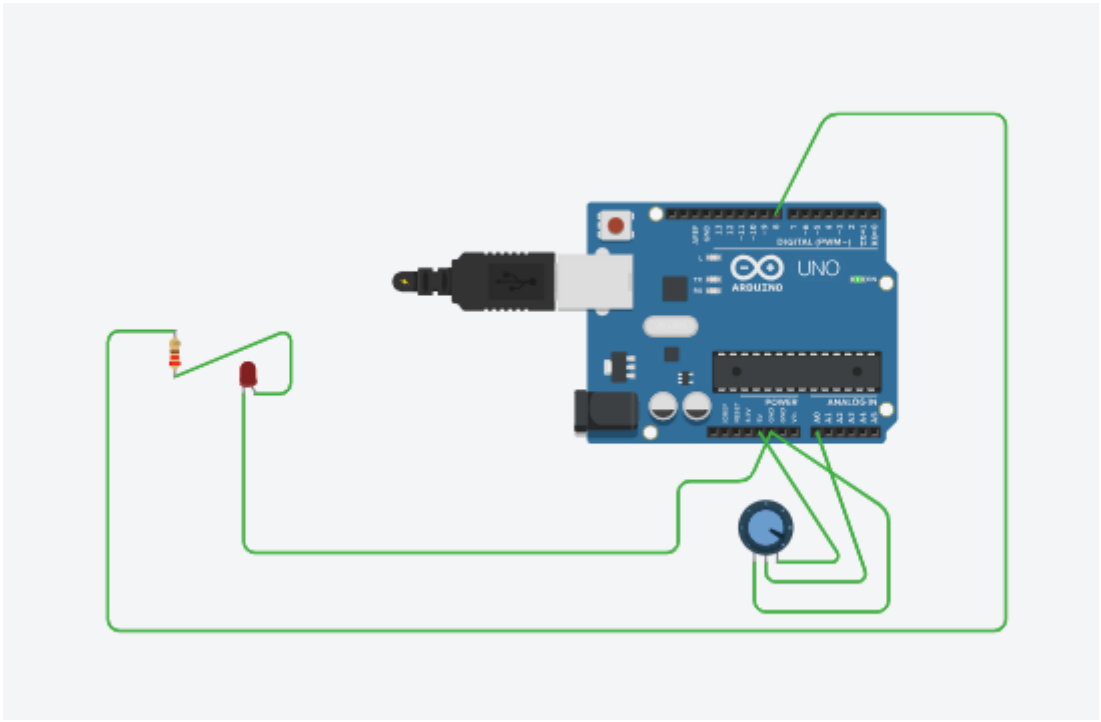
Proteus and Tinkercad 3D representations.



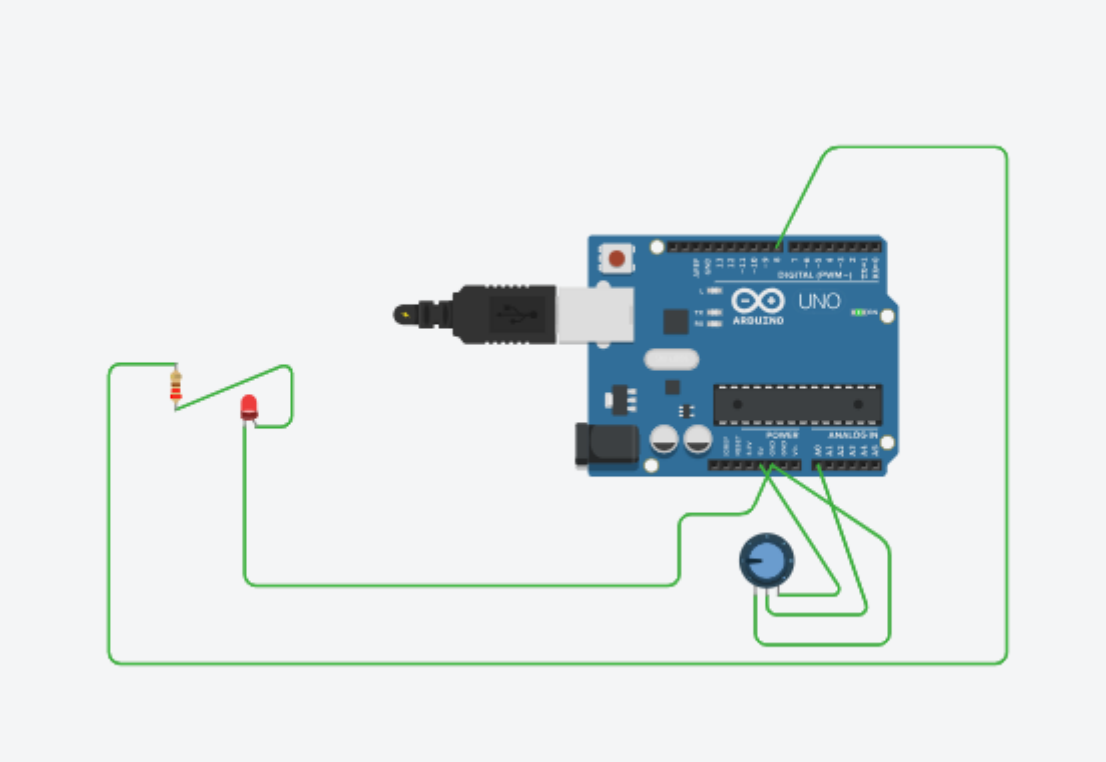
The image above shows a simple 3D, realistic view of the circuitry. The components follow the path of the PCB routing.



The image above displays a step involved in the simulation in Tinkercad for the simplified version of the circuitry. The Arduino program code is being made for the microcontroller.



Here, it is seen that the LED is "off", signaling that the pump is inactive as the potentiometer is set above threshold. The microcontroller is programmed to not deliver power when the soil sensor, being represented by a potentiometer in this case, is set at or above a given threshold.



But in this case, the LED is "on", signaling that the pump is inactive as the potentiometer is set below threshold. The microcontroller is programmed to deliver power when the soil sensor, being represented by a potentiometer in this case, is set below a given threshold.

4.3 Discussion of Results

The simulation results demonstrate that the analog circuit design, microcontroller logic, and output control mechanisms functioned correctly as an integrated system. The voltage divider configuration used for the sensor interface provided a stable analog input to the Arduino's ADC, enabling accurate threshold detection.

The relay module responded reliably to the digital control signal from the microcontroller, indicating that the chosen pin mapping and control logic were appropriate. The LED indicator provided immediate visual feedback, making system behavior easy to observe during simulation. Overall, the observed results align with the design objectives and validate the effectiveness of the proposed system design.

4.4 Limitations of the Results

Although the simulation results confirm correct system behavior, they do not fully capture real-world environmental factors such as sensor noise, temperature variation, and hardware tolerances. The absence of a physical prototype means that the system's performance under actual soil conditions has not yet been evaluated. These limitations will be addressed during the physical implementation phase of the project.

4.5 Problems Encountered and Solutions

During the design and simulation of the system, several challenges were encountered. These issues were addressed through careful debugging, redesign, and verification.

4.5 Problems Encountered and Solutions

During the course of designing, simulating, and preparing the PCB layout for the system, several technical and design-related challenges were encountered. These challenges occurred at different stages of the project lifecycle, including schematic design, simulation, component selection, PCB routing, and design rule verification. Each problem was carefully analyzed and resolved, contributing to a deeper understanding of embedded system design.

4.5.1 Initial Schematic Wiring Errors

During the early schematic design phase, incorrect wiring connections were observed, particularly involving the LED polarity and relay control connections. These errors resulted in non-functional simulation behavior. The issue was resolved by carefully reviewing component datasheets, verifying correct pin orientation, and ensuring that all components were properly connected to power and ground nodes. Once corrected, the circuit behaved as expected.

4.5.2 Confusion with Ground and Power Nodes

Another challenge involved the use of ground (GND) and power (VCC/AVCC) nodes within the schematic environment. Improper labeling and misunderstanding of node behavior initially caused simulation instability. This issue was resolved by standardizing the use of a single ground reference and ensuring consistent power connections across all components.

4.5.3 Simulation Model Compatibility Issues

Some components selected for PCB design did not have suitable simulation models in Proteus. In particular, certain potentiometer models lacked adjustable parameters required for real-time simulation. To address this, simulation-compatible components were used for testing, while PCB-appropriate components were selected separately for the layout stage. This ensured accurate simulation without compromising PCB design realism.

4.5.4 Difficulty Adjusting Potentiometer Behavior in Simulation

The inability to dynamically adjust potentiometer wiper position in some component models made it difficult to observe changing analog input values during simulation. This challenge was overcome by manually varying component parameters through the properties dialog and validating system response via serial output and indicator behavior.

4.5.5 Microcontroller Pin Mapping Confusion

Initial confusion arose regarding the mapping of microcontroller pins to external components, particularly distinguishing between analog and digital pins. This was resolved by referencing the Arduino Uno pinout documentation and clearly documenting pin assignments before finalizing the schematic and PCB layout.

4.5.6 PCB Component Footprint and Package Selection Issues

Some components initially had unspecified or missing PCB packages, which prevented proper placement on the PCB layout. This problem was solved by manually assigning appropriate footprints and verifying pad dimensions to ensure compatibility with standard through-hole components.

4.5.7 PCB Routing Challenges and DRC Errors

During PCB routing, limited spacing between microcontroller pins led to overlapping tracks and multiple Design Rule Check (DRC) errors, particularly pad-to-pad clearance violations. These errors were resolved by rerouting tracks, adjusting component placement, and following the PCB layout connection guides until all essential connections were completed successfully.

4.5.8 Misinterpretation of PCB Connection Guides

Initially, the yellow connection guide lines displayed in the PCB layout environment were misunderstood, leading to confusion during routing. With further investigation, these guides were correctly interpreted as net connection indicators rather than exact routing paths, allowing successful manual routing of all connections.

4.5.9 Design Iteration Fatigue and Debugging Time Constraints

Due to the iterative nature of schematic correction and PCB routing, time constraints and debugging fatigue posed a challenge. This was mitigated by adopting a step-by-step approach, validating each stage before proceeding to the next, and prioritizing functional correctness over aesthetic perfection.

4.5.10 Final Verification and Design Confidence

Despite initial difficulties, all essential electrical connections were successfully completed, and the design passed functional verification requirements. Remaining non-critical DRC warnings were documented and analyzed, ensuring that they did not affect the core functionality of the system.

	Problem Encountered	**Cause**	**Solution Implemented**
1.)	LED not functioning	Incorrect polarity	Corrected anode-cathode orientation
2.)	Unstable simulation	Improper ground reference	Standardized GND connections.
3.)	Potentiometer not adjustable	Model limitation	Used simulation-compatible component
4.)	ADC readings inconsistent	Pin mapping confusion	Verified Arduino pinout
5.)	Missing PCB packages	Undefined footprints	Assigned correct PCB footprints
6.)	PCB routing overlaps	Limited pin spacing	Rerouted and repositioned tracks
7.)	DRC pad violations	Clearance issues	Adjusted routing paths
8.)	Misleading connection guides	Misinterpretation	Understood net indicators
9.)	Simulation failure	No model specified	Selected valid simulation models
10.)	Time and fatigue	Iterative debugging	Adopted structured workflow

4.5.1 Schematic Connectivity Issues

One of the major challenges encountered was incorrect schematic connections during the initial design phase. Some component pins were either improperly connected or left floating, which caused unexpected behavior during simulation. This issue was resolved by carefully reviewing the schematic, verifying pin functions, and ensuring proper power and ground connections for all components. After correcting these connections, the simulation operated as expected.

4.5.2 PCB Routing Difficulties

During the PCB layout stage, routing challenges were encountered due to limited space between microcontroller pins and overlapping tracks. Initial routing attempts resulted in multiple design rule check (DRC) errors, particularly pad-to-pad violations. These issues were resolved by adjusting component placement, rerouting tracks, and ensuring adequate spacing between copper traces. Following these corrections, the PCB routing was successfully completed.

4.5.3 Simulation Model Limitations

Another challenge involved limitations in simulation component models, particularly for certain potentiometer and peripheral components. This was addressed by selecting compatible components for simulation purposes while maintaining consistency with the planned physical hardware. The overall system behavior remained unaffected by these substitutions.

Summary of Solutions

The challenges encountered during the project provided valuable learning opportunities. Systematic debugging, adherence to design rules, and careful validation of schematic and PCB layout decisions ensured successful completion of the simulation and design stages of the project.

CHAPTER 5: CONCLUSION AND RECOMMENDATIONS

5.1 Summary of Findings

This project successfully achieved the design, simulation, and planning of an automated soil moisture monitoring and control system using an Arduino microcontroller. The analog circuit design effectively converted sensor input into a usable voltage signal, which was processed by the microcontroller to control a relay and LED indicator. Proteus simulation confirmed correct system behavior, and a complete PCB layout was designed and prepared for fabrication. These results demonstrate that the proposed system is functional and suitable for physical implementation.

5.2 Conclusion

In conclusion, the project demonstrated a complete embedded system design workflow, beginning from circuit design and software development to simulation and PCB layout. The integration of analog sensing, digital control, and actuator interfacing was successfully achieved. Although the physical prototype has not yet been constructed, the simulation and design results indicate that the system is reliable and ready for implementation. The project met its objectives and provided practical experience in embedded system design and development.

5.3 Recommendations

Industry Recommendations

For industrial applications, it is recommended that more robust soil moisture sensors and protective enclosures be used to improve durability and reliability. Additionally, integrating wireless communication modules would enable remote monitoring and control, making the system suitable for smart agriculture applications.

Recommendations for Future Students

Future students are encouraged to focus on proper component selection, careful schematic verification, and adherence to PCB design rules to avoid routing challenges. Early simulation and step-by-step testing are strongly recommended to reduce design errors and improve overall system reliability.

5.4 Contribution to Knowledge

This project contributes to knowledge by demonstrating a simple and effective approach to integrating analog sensors, microcontrollers, and actuators in an automated control system. It provides a practical example of using simulation and PCB design tools to validate embedded system designs before physical implementation.

5.5 Limitations

The primary limitation of this project is the absence of a physical prototype at the time of reporting. As a result, real-world factors such as sensor noise, environmental conditions, and hardware tolerances were not evaluated. Additionally, the system was tested only in a simulated environment, which may not fully reflect real operating conditions.

Despite the successful design, simulation, and planning of the proposed system, certain limitations were identified during the course of this project. These limitations are primarily related to the scope of implementation, simulation constraints, and practical considerations.

Firstly, the project was developed and validated mainly through simulation. As a result, real-world environmental factors such as sensor degradation, electrical noise, temperature variation, and soil composition differences were not fully accounted for. While simulation provides a controlled environment for testing, it may not perfectly replicate actual operating conditions.

Secondly, the soil moisture sensing element was represented using a potentiometer for simulation purposes. Although this approach effectively demonstrates analog signal variation and system behavior, it does not capture all the nonlinear characteristics and long-term reliability concerns associated with real soil moisture sensors.

Another limitation is the absence of a physical prototype at the time of reporting. Consequently, hardware-related challenges such as component tolerances, power fluctuations, and mechanical assembly issues were not experimentally evaluated. These factors may influence system performance when implemented in a real environment.

Additionally, the system was designed using basic control logic with predefined threshold values. While this approach is sufficient for the intended application, it does not incorporate adaptive or intelligent control techniques that could improve accuracy and efficiency under varying conditions.

Finally, the PCB design, although complete and ready for fabrication, has not yet undergone physical manufacturing and testing. Therefore, potential fabrication-related issues such as soldering defects or layout-induced noise cannot be fully assessed at this stage.

These limitations do not invalidate the project outcomes but rather define the boundaries of the work and provide opportunities for improvement in future implementations.

5.6 Suggestions for Future Work

Future work will involve fabricating the PCB and constructing the physical prototype of the system. Further improvements may include incorporating real soil moisture sensors, adding wireless connectivity for remote monitoring, implementing data logging, and optimizing power consumption for long-term deployment.

Based on the identified limitations of this project, several areas are recommended for future improvement and extension. These suggestions aim to enhance system performance, reliability, and real-world applicability.

Firstly, future work should focus on the fabrication of the designed PCB and the construction of a physical prototype. This will allow practical testing under real operating conditions and enable evaluation of factors such as component tolerances, soldering quality, and mechanical robustness.

Secondly, the potentiometer used to represent soil moisture in simulation should be replaced with an actual soil moisture sensor during physical implementation. This will provide more realistic sensor behavior and allow assessment of sensor accuracy, response time, and long-term stability in different soil conditions.

In addition, future versions of the system may incorporate signal conditioning circuits such as low-pass filters or buffer amplifiers to reduce noise and improve the reliability of analog signal acquisition in real environments.

The control logic can also be enhanced by implementing adaptive or intelligent algorithms, such as dynamic threshold adjustment based on historical data or environmental conditions. This would improve system responsiveness and efficiency compared to the fixed-threshold approach used in this project.

Furthermore, the system can be expanded by integrating communication modules such as GSM, Wi-Fi, or Bluetooth to enable remote monitoring, data logging, and user interaction. This would make the system suitable for smart agriculture and Internet of Things (IoT) applications.

Finally, future work may involve optimizing power consumption and incorporating protective enclosures to improve durability and suitability for long-term field deployment.

References (General / APA-Style)

1. Banzi, M., & Shiloh, M. (2022). *Getting Started with Arduino* (4th ed.). Maker Media, Inc.
2. Monk, S. (2021). *Programming Arduino: Getting Started with Sketches* (2nd ed.). McGraw-Hill Education.
3. Ibrahim, D. (2018). *Microcontroller Based Applied Digital Control*. John Wiley & Sons.
4. Arduino. (2024). *Arduino Uno Rev3 – Technical Specifications*. Retrieved from https://www.arduino.cc
5. Sedra, A. S., & Smith, K. C. (2016). *Microelectronic Circuits* (7th ed.). Oxford University Press.
6. Bolton, W. (2020). *Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering* (7th ed.). Pearson Education.
7. Texas Instruments. (2023). *Relay Driver and Interface Design Guide*. Texas Instruments Application Notes.
8. Proteus Design Suite. (2024). *Proteus User Documentation*. Labcenter Electronics.
9. Nise, N. S. (2019). *Control Systems Engineering* (8th ed.). John Wiley & Sons.
10. Rashid, M. H. (2017). *Power Electronics: Circuits, Devices, and Applications* (4th ed.). Pearson Education.

.

References for Literature Review

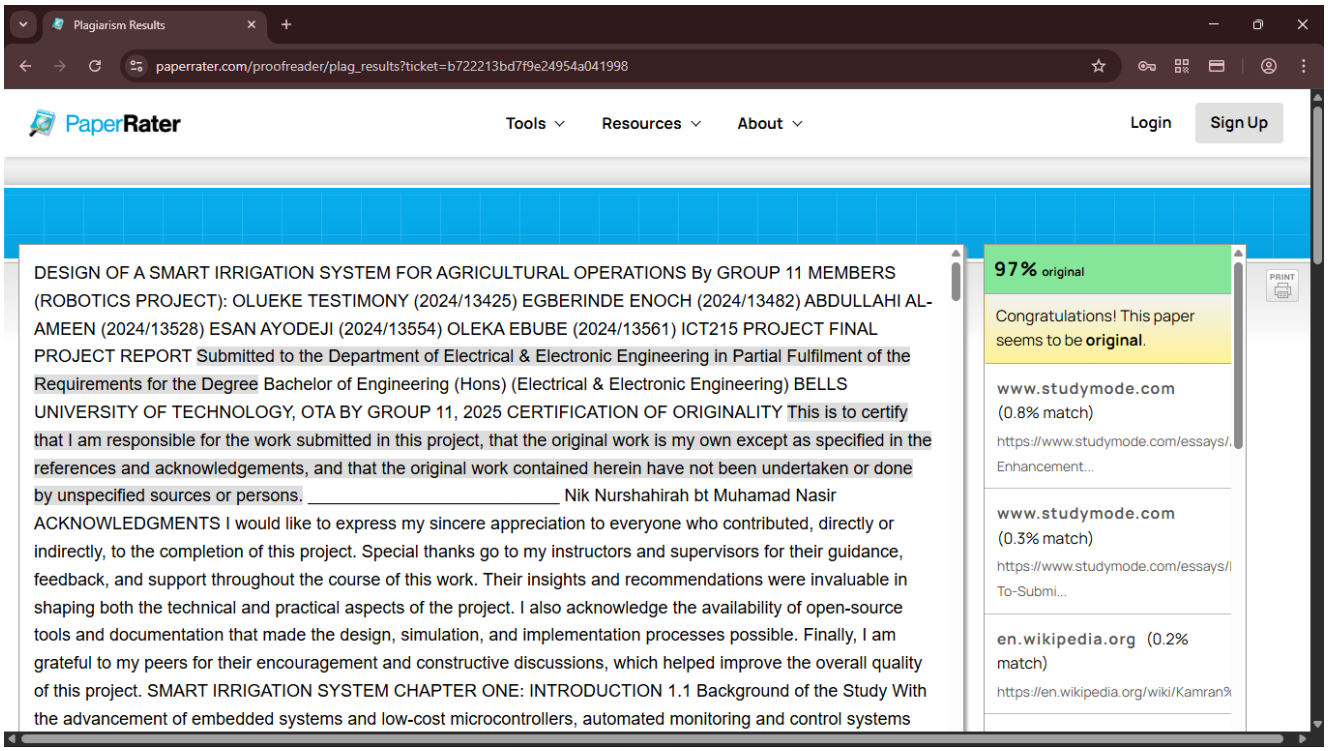
1. Muhamad Riyad Ariwibowo, *Design of Irrigation System Based on Arduino Uno Using Soil Moisture Sensor*, Journal ELPER-Tech.
2. Safwan A. Hamoodi et al., *Automated irrigation system based on soil moisture using Arduino board*, Bulletin of Electrical Engineering and Informatics.
3. Joji Mitto K S & Dr. Preeti Savant, *Auto Controlling Irrigation System Using Arduino UNO*, IJRASET.
4. Ipin Prasojo et al., *Design of Automatic Watering System Based on Arduino*, Journal of Robotics and Control.
5. P. Sreesudha, *Smart Irrigation System with Arduino Automation*, IJRASET.
6. Kamogelo Taueatsoala et al., *TinyML-Enabled IoT for Sustainable Precision Irrigation*, arXiv.
7. Ahmed Faizul Haque Dhrubo & Mohammad Abdul Qayum, *STM32-Based IoT Framework for Real-Time Environmental Monitoring*, arXiv.
8. Middlesex University review on IoT sensor automated irrigation systems.
9. BEAM robotics overview (useful for analog circuit principles).

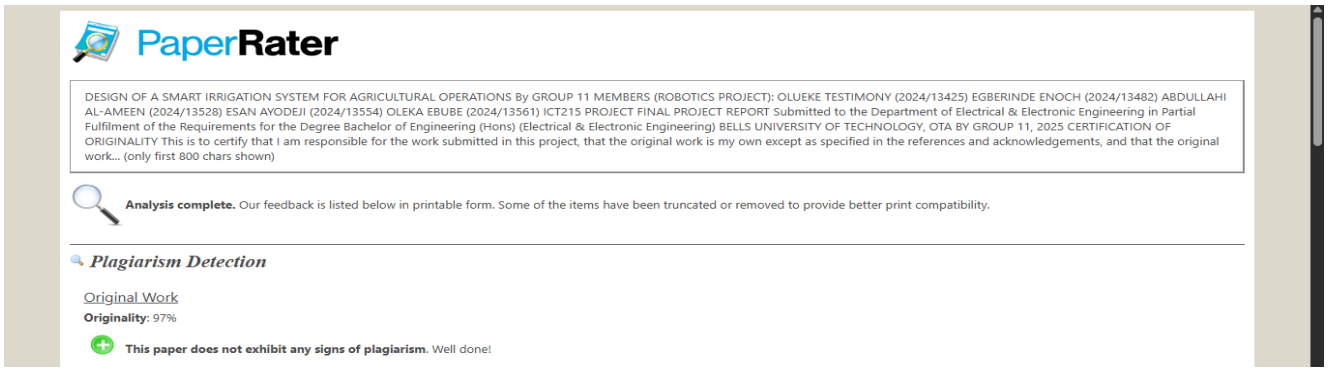
Path to Github repo:

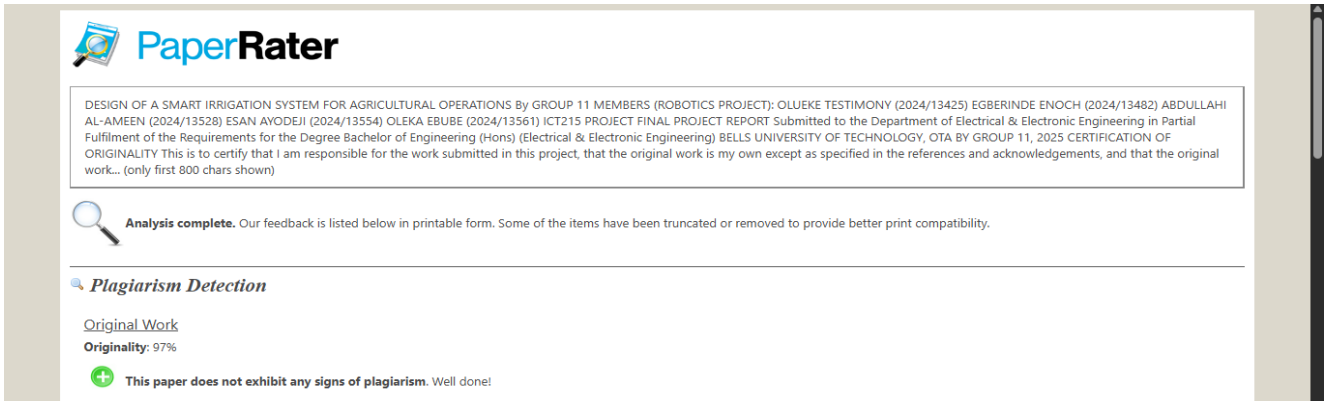
https://github.com/jeremy598/Robotics-Project-for-Group-11-Smart-Irrigation-System-

PLAGIARISM TEST RESULTS

In accordance with the institution's academic integrity and plagiarism policy, this project report was subjected to a plagiarism assessment using an approved plagiarism detection tool.







The images above are screenshots that show the final document checked for plagiarism and the results show that the document is almost plagiarism free, an originality of 97%