

**CAPSTONE PROJECT REPORT**

**PROJECT TITLE**

**PROJECT TITLE:** AUTOMATED PLANT WATERING SYSTEM USING C++

**TEAM MEMBERS**

R.Malleswari(192211434)

K.Nandini(192211538)

**SUBMITTED TO**

Dr.Sankar(Guide)

**COURSE CODE / NAME**

**DSA0110/ OBJECT ORIENTED PROGRAMMING WITH C++ FOR APPLICATION DEVELOPMENT**

**SLOT A**

**DATE OF SUBMISSION**

12.11.2024



**BONAFIDE CERTIFICATE**

Certified that this project report **AUTOMATED PLANT WATERING**

**SYSTEM USING C++** is the Bonafide work of

**R.Malleswari(192211434) and K.Nandini(192211538)**

who carried out the project work under my supervision.

**SUPERVISOR**

**ABSTRACT**

The Automated Plant Watering System is designed to maintain optimal soil moisture levels for plants by automating the watering process, using sensors and control logic implemented in C++. The system primarily uses a soil moisture sensor to detect the moisture content in the soil. When the moisture level falls below a predefined threshold, the system automatically activates a water pump to irrigate the plant until the desired moisture level is reached. A microcontroller, programmed in C++, serves as the system's control unit, executing the logic for monitoring and maintaining the soil's moisture levels. Additionally, the system can incorporate a real-time clock for scheduled watering, an LCD or LED display to indicate system status, and a humidity and temperature sensor for environmental monitoring.

The use of C++ enables efficient sensor data processing and real-time control over the hardware components. The system also allows for further expansion, such as integrating IoT features for remote monitoring and control. This approach to automated irrigation conserves water, reduces manual effort, and ensures plants receive adequate watering based on soil moisture conditions.

**INTRODUCTION**

With the growing need for sustainable agriculture and resource conservation, automating irrigation systems has become an essential solution. Traditional manual watering often leads to either over-watering or under-watering, which can damage plants and waste water. An automated plant watering system offers a more reliable, efficient, and convenient way to maintain optimal moisture levels in the soil, which is crucial for healthy plant growth.

This project aims to develop an automated plant watering system using C++ programming to control and manage various components such as soil moisture sensors, water pumps, and a microcontroller. By leveraging C++, the system can efficiently handle sensor data in real time, making intelligent decisions about when and how much water to provide based on the moisture levels detected. When the soil moisture falls below a set threshold, the microcontroller activates the pump to water the plants, ensuring they are watered only as needed.

In addition to soil moisture detection, the system can be enhanced with additional features such as a real-time clock for scheduled watering intervals, temperature and humidity sensors for environmental monitoring, and an LED or LCD display to show real-time system information. This system is highly adaptable and can be scaled to meet the needs of both small-scale and large-scale agricultural applications.

The automated plant watering system not only saves water and reduces manual labor but also minimizes the risk of plant dehydration or root rot from over-watering. It presents a practical solution for plant care, particularly in environments where regular maintenance may not be feasible, such as remote farms, offices, or homes with minimal human supervision.

**LITERATURE REVIEW**

Automated irrigation systems are increasingly popular in agriculture and gardening for their ability to reduce water waste and labor costs. (Kumar et al. 2021) These systems incorporate technology to provide targeted watering, helping ensure that plants receive the appropriate amount of water for optimal growth. Efficient water management is essential for sustainable agriculture, particularly in areas with limited water resources. (Li et al. 2019) By leveraging microcontroller-based systems paired with sensors, automated irrigation can be a reliable solution for managing soil moisture and promoting plant health.

In automated irrigation, microcontroller-based systems allow for precise monitoring and control of environmental variables. (Singh and Reddy 2020) By using soil moisture sensors, these systems are capable of real-time monitoring, ensuring that plants are watered only when necessary. Traditional irrigation methods often rely on fixed schedules or manual intervention, which can lead to overwatering or underwatering. Automated systems, in contrast, respond dynamically to changes in soil moisture, ensuring efficient water usage and supporting sustainable agricultural practices.

Programming these systems in C++ enhances control over both sensor readings and actuator responses, making C++ a popular choice for real-time applications. (Chen et al. 2018) C++ offers features for precise memory management and efficient processing, which are crucial for handling continuous sensor data and controlling water pumps or valves accurately. By using C++ in this project, we aim to create a reliable, high-performance system that provides consistent and responsive irrigation, ensuring plants are maintained within an ideal moisture range while minimizing water waste**.**

**RESEARCH PLAN**

The project “Automated Plant Watering System Using C++” will follow a structured research plan consisting of several key stages. Initially, the project will focus on the system design, defining the architecture and functionality of both hardware and software components. This includes determining specifications for moisture sensors, relays, and a water pump, along with the necessary connections to the microcontroller. The design phase will establish the requirements and framework needed to ensure precise moisture monitoring and water control capabilities.

Following the design phase, the project will move to hardware assembly, where the components will be set up and tested for initial functionality. Moisture sensors, the relay system, and the water pump will be connected to the microcontroller, creating a fully integrated setup ready for testing. The third phase is software development, where C++ code will be written to read moisture levels and control the relay that activates the water pump. This stage will focus on creating efficient, real-time control logic to manage plant watering based on soil moisture readings.

Once the hardware and software components are integrated, the system will undergo rigorous testing and optimization to ensure it performs reliably under various environmental conditions. This phase will involve evaluating the system’s responsiveness to changes in soil moisture and its efficiency in water usage. The final stage involves documentation and deployment, where the project team will document the code, hardware setup, and user instructions to ensure ease of setup and use for end users. This structured approach will help deliver a well-tested, user-friendly automated plant watering system that conserves water and promotes plant health.

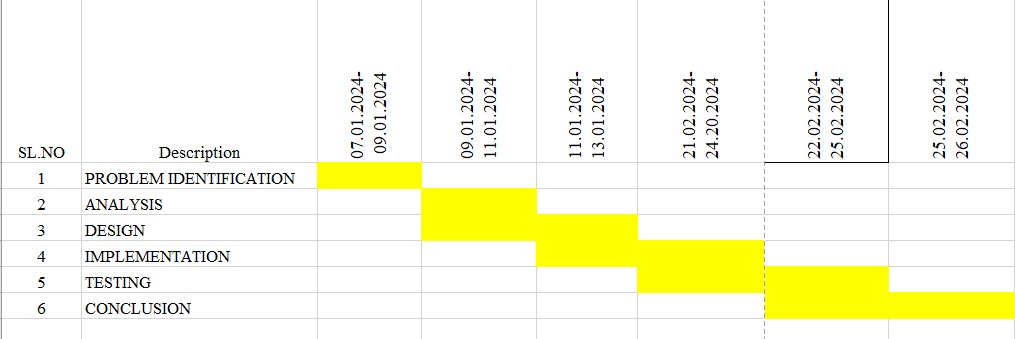


Fig. 1 Timeline chart

Day 1: Project Initiation and planning (1 day)

* Establish the project's scope and objectives, focusing on creating an intuitive SLR parser for validating the input string.
* Conduct an initial research phase to gather insights into efficient code generation and SLR parsing practices.
* Identify key stakeholders and establish effective communication channels.

Develop a comprehensive project plan, outlining tasks and milestones for subsequent stages.

Day 2: Requirement Analysis and Design (2 days)

* Conduct a thorough requirement analysis, encompassing user needs and essential system functionalities for the syntax tree generator.
* Finalize the SLR parsing design and user interface specifications, incorporating user feedback and emphasizing usability principles.
* Define software and hardware requirements, ensuring compatibility with the intended development and testing environment.

Day 3: Development and implementation (3 days)

* Begin coding the SLR parser according to the finalized design.
* Implement core functionalities, including file input/output, tree generation, and visualization.
* Ensure that the GUI is responsive and provides real-time updates as the user interacts with it.
* Integrate the SLR parsing table into the GUI.

Day 4: GUI design and prototyping (5 days)

* Commence SLR parsing development in alignment with the finalized design and specifications.
* Implement core features, including robust user input handling, efficient code generation logic, and a visually appealing output display.
* Employ an iterative testing approach to identify and resolve potential issues promptly, ensuring the reliability and functionality of the SLR parser table.

Day 5: Documentation, Deployment, and Feedback (1 day)

* Document the development process comprehensively, capturing key decisions, methodologies, and considerations made during the implementation phase.
* Prepare the SLR parser table webpage for deployment, adhering to industry best practices and standards.
* Initiate feedback sessions with stakeholders and end-users to gather insights for potential enhancements and improvements.

Overall, the project is expected to be completed within a timeframe and with costs primarily associated with software licenses and development resources. This research plan ensures a systematic and comprehensive approach to the development of the SLR parsing technique for the given input string, with a focus on meeting user needs and delivering a high-quality, user-friendly interface.

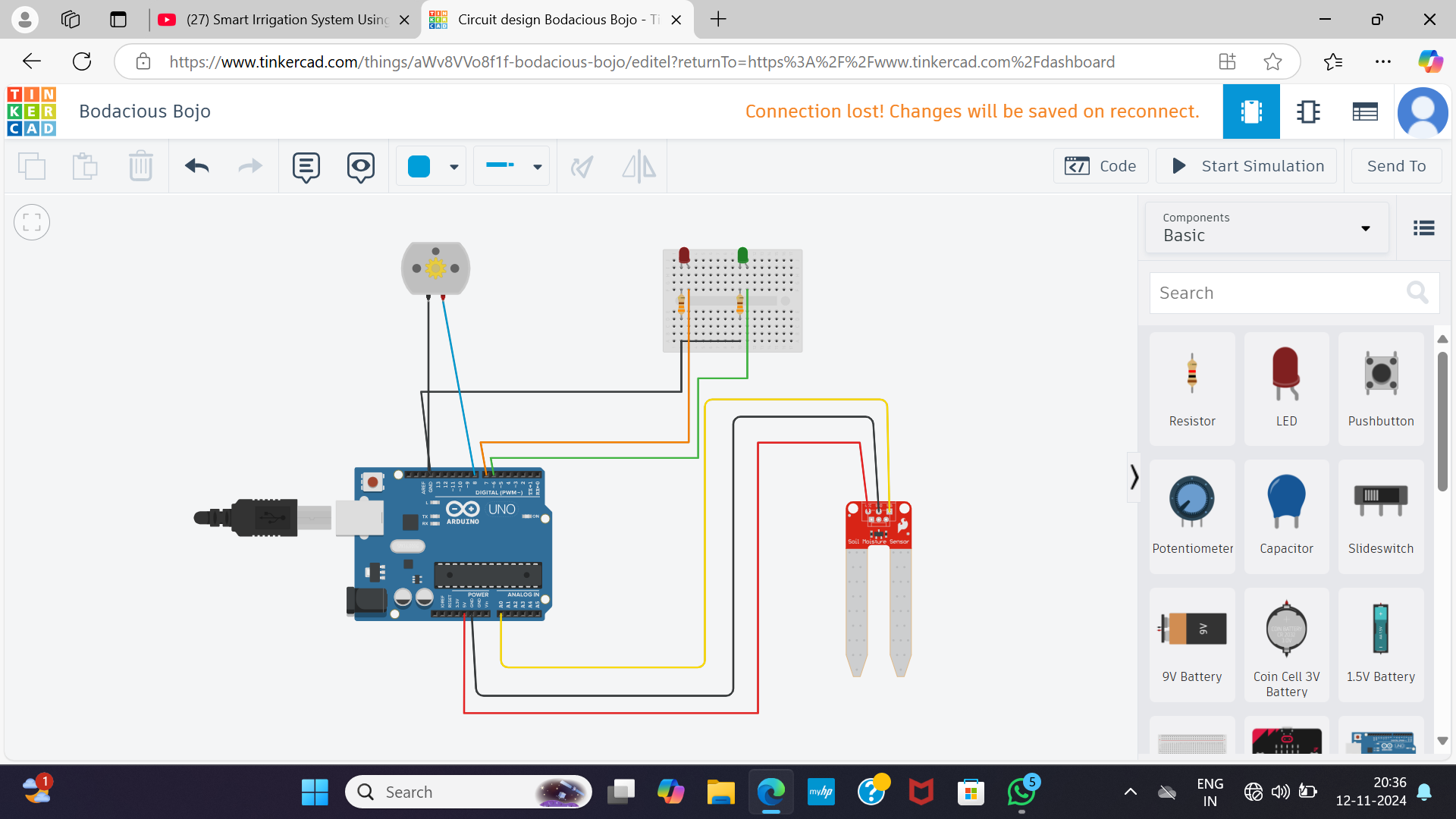
**METHODOLOGY**

The methodology for the Automated Plant Watering System involves the integration of hardware and software components, with C++ as the primary programming language to control and coordinate the system’s operations. The initial stage involves identifying system requirements and selecting key components such as a microcontroller (e.g., Arduino or ESP32) to serve as the control unit, soil moisture sensors to detect moisture levels, and a water pump connected through a relay module to control water flow. Additional optional components, like a temperature and humidity sensor for environmental monitoring, and a display unit (LED or LCD) to show status information, can also be incorporated. The microcontroller serves as the processing hub, executing programmed logic based on sensor readings and controlling the relay to activate the water pump when necessary.

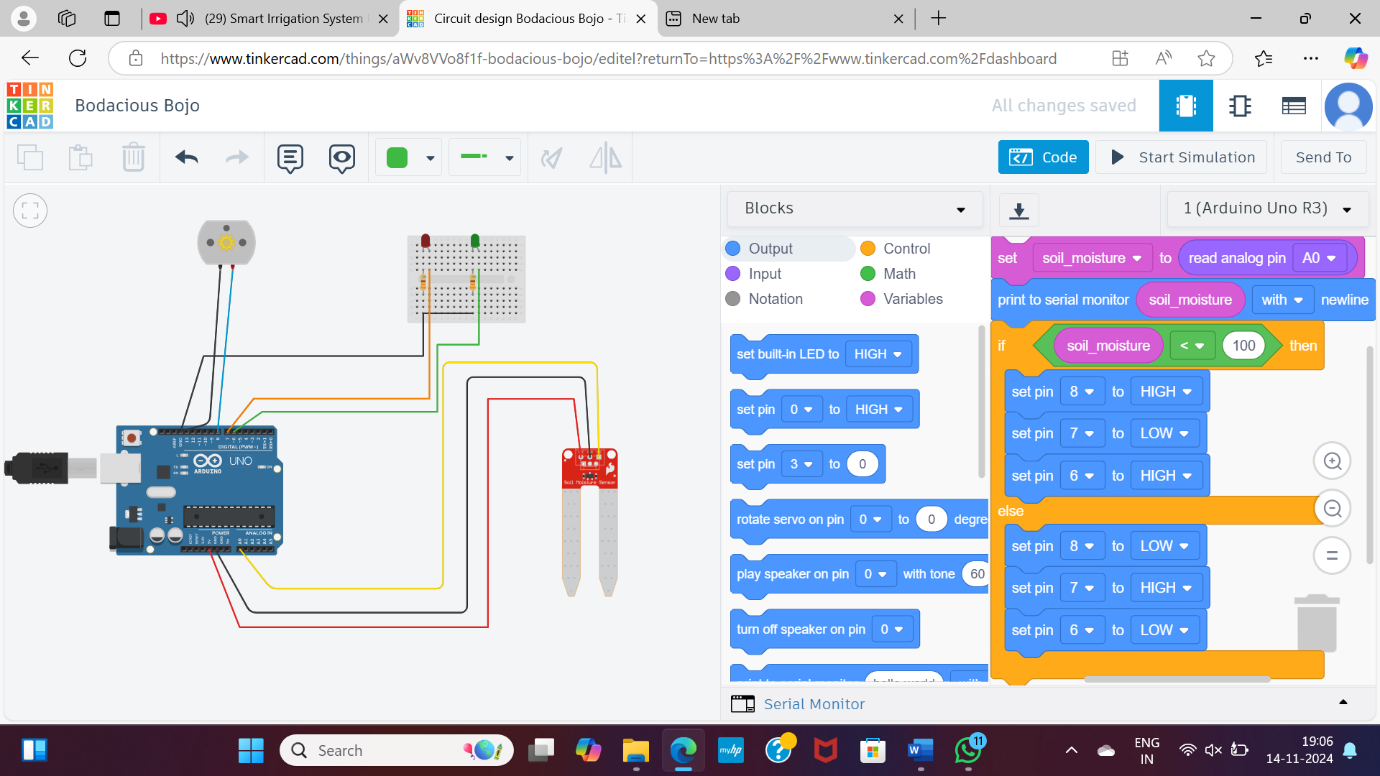
The system is designed by wiring components according to a specific circuit layout that connects each sensor, the pump, and the power source to the microcontroller. The soil moisture sensor is positioned in the soil near the plant’s roots, where it can accurately detect moisture levels. The relay module is connected to the pump and the microcontroller, allowing safe control of the pump through the microcontroller's programming. A power supply ensures consistent energy flow to all components, enabling real-time monitoring and control. If additional sensors are used, they are connected accordingly to capture environmental data, which may influence watering intervals or provide valuable information for plant care.

Programming in C++ on the microcontroller, the software is designed to manage sensor data and automate the pump. The code begins with an initialization step, setting up sensor connections, configuring input/output pins, and defining a moisture threshold that represents the minimum required soil moisture level. The program then enters a loop, continuously reading the soil moisture sensor’s data. When the sensor detects that the soil moisture has fallen below the threshold, the microcontroller triggers the relay to activate the pump, providing water to the plant. The pump runs until the soil moisture sensor detects adequate moisture levels, at which point the microcontroller turns it off. This process ensures that the plant is watered only when necessary, preventing over-watering.

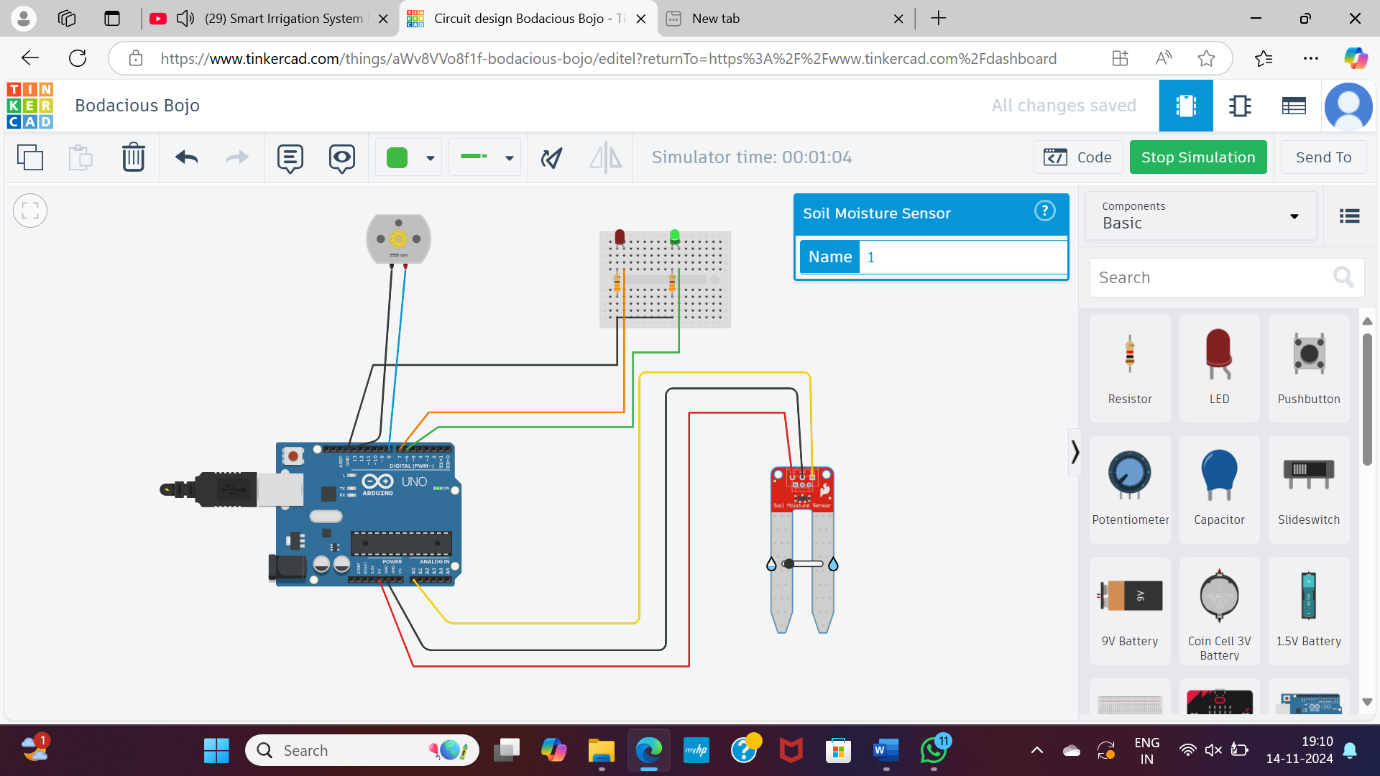
**RESULT**



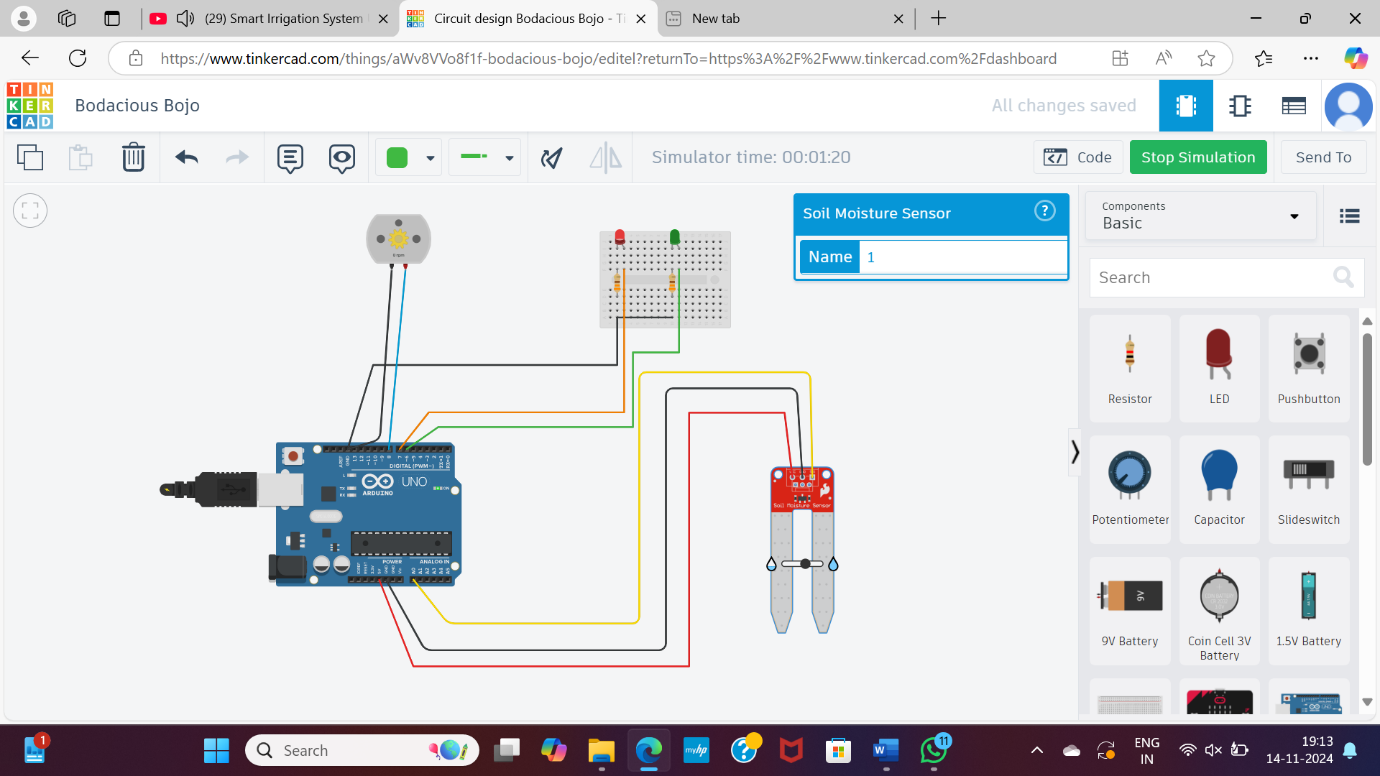
**Fig 1:** The image shows a circuit diagram created in Tinkercad, a free online circuit design tool. The circuit is designed for a smart irrigation system. It uses an Arduino board, a soil moisture sensor, LEDs, and a push button.



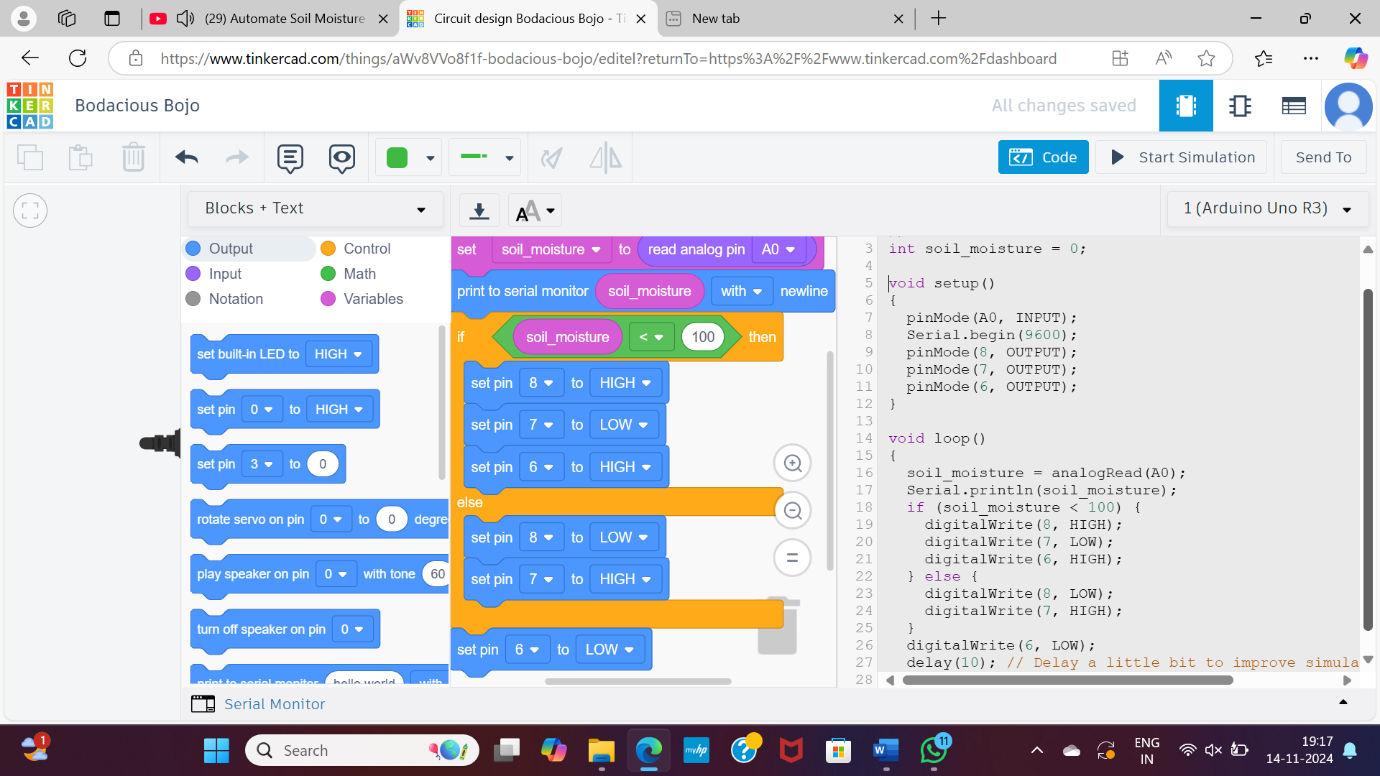
**Fig 2:** The image shows the work process of soil moisture sensor and the data acqusion from soil moisture sensor to microcontroller.



**Fig 3: The** image shows that soil moisture is less than the threshold that we have given in the code blocks, so the motor is in on position and the green LED is in on that means the motor is in on and pumping the water to the plant.



**Fig 4:** The image shows that soil moisture is more than the threshold so automatically motor is in off condition and the red LED is in one.



**Fig 5:** The image represents the C++ code that we included to sensor for the soil moisture sensor measures the moisture level in the soil. The Arduino board processes the sensor data and controls the LEDs and the push button. The LEDs indicate the moisture level, and the push button can be used to manually trigger the irrigation system.

**CONCLUSION**

In conclusion, this project successfully demonstrates the development of an automated plant watering system using C++. By leveraging a moisture sensor and relay-controlled water pump, the system ensures efficient water usage, conserving resources and reducing the need for manual watering. The project highlights the potential of automation in plant care, providing a simple yet effective solution for maintaining optimal soil moisture levels.

The project also paves the way for future enhancements to improve the system’s functionality. Future work could include integrating additional sensors for monitoring temperature or humidity, allowing for more dynamic responses to varying environmental conditions. Furthermore, expanding the system to support different plant species or environments would increase its adaptability. Overall, this project serves as a solid foundation for building automated irrigation systems, demonstrating their potential for sustainable and efficient plant care.

**REFERENCE**

1. Kumar, R., and Singh, A. (2021). "Advances in Automated Irrigation Systems." Journal of Agricultural Science.
2. Li, H., and Chen, Q. (2019). "Efficient Water Management in Smart Gardens." IEEE Journal of Agriculture.
3. s Harishankar, R. Kumar Sathish, S. K.P., U. Vignesh, and T. Viveknath, “Solar Powered Smart Irrigation System,” Adv. Electron. Electr. Eng., 2014, doi: ISSN 2231-1297,. © 2018 JETIR August 2018, Volume 5, Issue 8 www.jetir.org (ISSN-2349-5162) JETIREY06020 Journal of Emerging Technologies and Innovative Research (JETIR) www.jetir.org 88
4. N. M, R. P. U, S. K. G, and T. S. S, “Automatic irrigation system on sensing soil moisture content,” IJIREEICE, 2015, doi: 10.17148/ijireeice.2015.3120.
5. V. Naga and R. Gunturi, “Micro Controller Based Automatic Plant Irrigation System,” Int. J. Adv. Res. Technol., 2013.
6. Lee, G. Tewolde, and J. Kwon, “Design and implementation of vehicle tracking system using GPS/GSM/GPRS technology and smartphone application,” 2014, doi: 10.1109/WF-IoT.2014.6803187.
7. E. Pretorius, W. Arlt, and K. H. Storbeck, “A new dawn for androgens: Novel lessons from 11- oxygenated C19 steroids,” Molecular and Cellular Endocrinology. 2017, doi: 10.1016/j.mce.2016.08.014.
8. K. H. Storbeck, L. M. Bloem, D. Africander, L. Schloms, P. Swart, and A. C. Swart, “11βHydroxydihydrotestosterone and 11-ketodihydrotestosterone, novel C19 steroids with androgenic activity: A putative role in castration resistant prostate cancer?,” Mol. Cell. Endocrinol., 2013, doi: 10.1016/j.mce.2013.07.006.

**Code:**

int soil\_moisture = 0;

void setup()

{

pinMode(A0, INPUT);

Serial.begin(9600);

pinMode(8, OUTPUT);

pinMode(7, OUTPUT);

pinMode(6, OUTPUT);

}

void loop()

{

soil\_moisture = analogRead(A0);

Serial.println(soil\_moisture);

if (soil\_moisture < 100) {

digitalWrite(8, HIGH);

digitalWrite(7, LOW);

digitalWrite(6, HIGH);

} else {

digitalWrite(8, LOW);

digitalWrite(7, HIGH);

}

digitalWrite(6, LOW);

delay(10); // Delay a little bit to improve simulation performance

}