EECS 1021
Final Project Report

Name: **Mian Muneeb Mahmood**

Student Id: **219140417**

Table of Contents:

Table of Contents: **2**
Introduction: **3**
Project Overview: **3**
Technical Requirements/Specifications: **4**
Flow Chart: 4
Components: 4
Procedure: **6**
Testing and Troubleshooting: **6**
Learning Outcomes: **7**
Conclusion: **7**

Introduction:

In this project, we aim to develop an automated plant watering system by integrating Java IntelliJ and Arduino IDE. The system will monitor the moisture level of a plant’s soil in real-time and automatically water the plant whenever the moisture level falls below a certain threshold. By implementing this project, we are bridging software and hardware concepts to create a practical solution that can be universally adapted to various gardening and agricultural needs.

This project provides students with valuable insights into how a software interface can communicate with hardware components to perform specific tasks. It also introduces some fundamental hardware concepts that mechanical, electrical, and software engineers need to understand, making it a great entry-level project for those interested in embedded systems and IoT (Internet of Things).

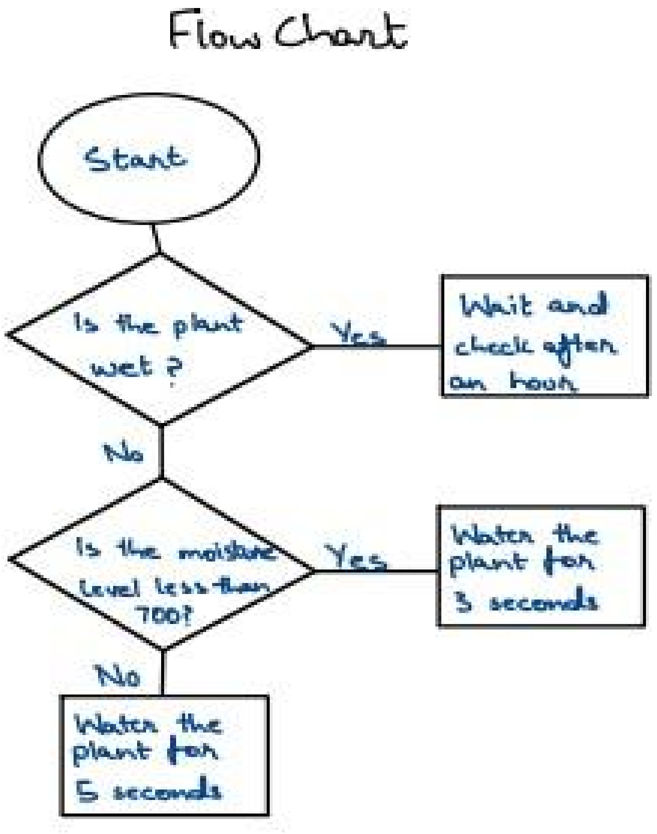
Project Overview:

The system's core functionality revolves around the continuous monitoring of the soil moisture level. The sensor periodically checks the moisture level and compares it against a predefined threshold. If the moisture level drops below the specified value, the system activates a water pump to supply water to the plant for a specific amount of time, ensuring the plant remains hydrated.

The project was developed using the Arduino platform for hardware control and Java for managing the logic of the system. This hybrid system helps students understand how to develop solutions that require both coding and electronic interfacing, providing them with hands-on experience in problem-solving through both software and hardware means.

Technical Requirements/Specifications:

Flow Chart:


Flow Chart
Start
Is the plant wet ?
Yes
Wait and check after an hour
No
Is the moisture level less than 700?
Yes
Water the plant for 3 seconds
No
Water the plant for 5 seconds

Components:

The following components were used to bring the project to life:

Arduino Kit:

- The core of the system that controls the water pump, monitors the moisture sensor, and executes commands from the Java program.

Java IntelliJ IDE:

- Used to write the program that communicates with Arduino. The Java code contains logic to handle the moisture level data, make decisions, and send commands to the hardware.

Plant:

- The actual subject of the experiment that requires consistent watering based on its moisture needs.

MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor):

- Used to control the water pump's power supply. The MOSFET acts as a switch that is turned on and off by signals from the Arduino, allowing the pump to operate efficiently.

Battery Pack:

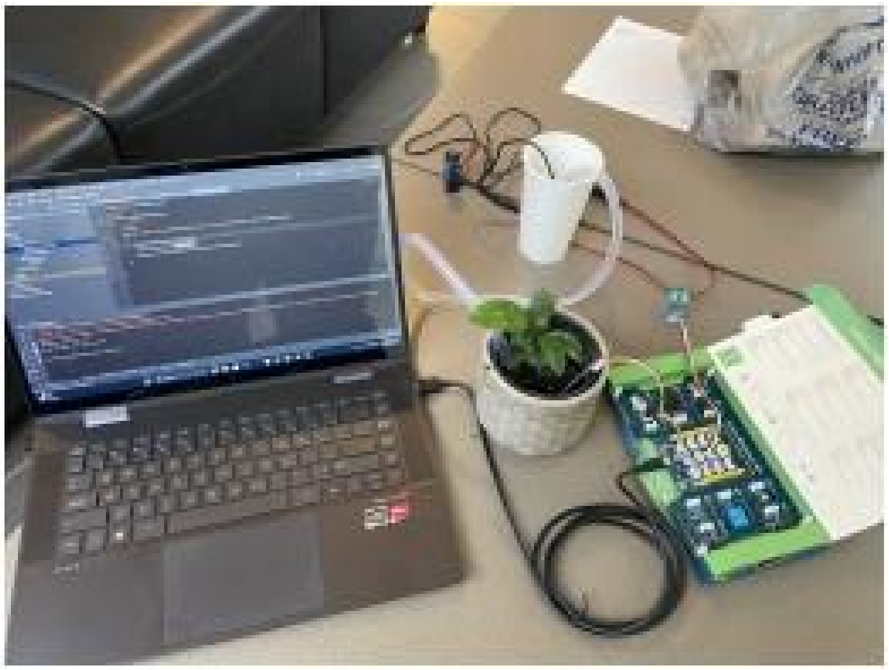
- Powers the entire system, ensuring that the water pump and Arduino function properly.

OLED Display:

- Displays important system messages and provides real-time feedback on the moisture level and the current state of the plant (dry, wet, or damp).

Moisture Sensor:

- The key component that collects real-time data on the soil's moisture level and sends the readings to the Arduino board for processing.



Procedure:

Preparation and Setup:

- Before starting the project, all components were double-checked to ensure proper connections and functionality. The moisture sensor was calibrated, and the code was written and uploaded to the Arduino board using the Arduino IDE.

Initial Challenges:

- In the initial testing phase, the system did not respond to the moisture level readings as expected. The pump was not operating in sync with the sensor readings, and commands were not executed properly. After troubleshooting, I discovered that the issue lay in the moisture threshold detection.

Refining the Code:

- The moisture detection threshold was adjusted, and the program was refined to better handle different moisture conditions. Once this problem was solved, the system began functioning as intended, sending commands to the pump based on the moisture readings.

Running the Program:

- The program was structured into two classes:
  - One class, “OtherClasses,” defined the necessary methods and functions required for sensor reading, moisture level comparison, and communication with the pump.
  - The second class, “MinorProjectCode,” contained the main logic where the methods were called and executed.
- After establishing the connection between Arduino and Java, the system ran two loops:
  - A daily loop that ran for 30 days to simulate a month-long test.
  - An hourly loop that checked moisture levels every hour and watered the plant as necessary.

Testing and Troubleshooting:

The system underwent several rounds of testing to identify and fix logical errors, bugs, and inconsistencies. Initially, the pump did not function for the specified time, and the moisture sensor values were not being processed correctly. By running the program multiple times after each issue was resolved, I was able to refine the logic and ensure that the system behaved as expected.

Testing also highlighted the importance of practicing coding solutions and repeatedly running tests to achieve a fully functional system. By iterating over the program, I learned how to detect and resolve errors efficiently.

Learning Outcomes:

Problem Identification and Solution:

- I gained a solid understanding of identifying technical problems and forming relevant coding solutions to solve them. This project also taught me how to integrate multiple systems (hardware and software) to perform a common task.

Object-Oriented Programming:

- Through this project, I further enhanced my understanding of classes, methods, and constructors. I was able to apply object-oriented programming principles to develop a robust solution that communicated effectively with the Arduino board.

Hardware Knowledge:

- By working with the Arduino Grove board and various sensors, I learned how to handle pins and establish communication between different hardware components and sensors.

Conclusion:

Working on this project was both a rewarding and educational experience. While I had previously worked with Arduino systems in EECS 1011, incorporating Java into the project introduced new challenges and learning opportunities. This project allowed me to expand my knowledge of hardware-software interfacing, troubleshooting, and coding practices.

The project can also be expanded further by adding additional features such as automatic fertilization or integrating the system with a mobile application to provide remote monitoring. The experience gained here has prepared me to work on more complex systems in the future and has provided a solid foundation in embedded systems and IoT technology.