EECS 1021

Minor Project: Auto Plant Watering with Arduino & Java

Abstract

This lab consists of creating what's called a 'state' machine. This refers to the core of a program that takes actions based on things that are sensed or measured. A state machine is created to monitor and water a plant.

Ali Saheb

alisaheb@my.yorku.ca

218791376

April 14, 2022

Abstract

INTRODUCTION: This project consists of creating a Java script (through StandardFirmata) using different Arduino hardware to code a functioning self-watering plant system, based on if the plant requires water or not. **CONTEXT:** The objective of this lab is to have a fully self-dependant plant-watering system that determines if the plant needs water based on a moisture sensor that gives different voltage readings depending on the amount of water it is touching (resistance). We would then like to take said readings and display them on the OLED screen included on the Grove board beginner kit. **REQUIREMENTS:** To have a functioning system, the plant must be entirely self-sustainable (so long as it has a water source). The system must be able to read the current moisture of the soil and display it onto the OLED screen. Based on these readings, the system must also have the ability to enable and disable the pump; watering the plant itself without any interference. Based on the data collected by the system, a live graph/line must also be displayed on the OLED screen allowing for a visual representation of the "wetness" of the plant. The system should be able to run endlessly/continuously without any issues. **COMPONENT LIST:** The self-watering plant system is made up of the following devices/items:

- Arduino Grove Board Beginner Kit
    - Master HUB containing all connections to sensors/devices (e.g., OLED, moisture sensor, MOSFET Board, etc.)
- OLED Display
    - Built into the grove board
- Moisture Sensor
    - Gives values based off resistance from meeting water
- MOSFET Board
    - Controls voltage/current flow from the 9V battery to the pump
- Water Pump
    - Submerged in water supply allowing for water to be pulled through a clear tube and into the plant
- Plant Pot & Plate
    - The pot and plate reduce the number of spills and dirt from falling
- Spearmint Plant
    - Neglected wilting Spearmint plant named "Fred"
- Laptop
    - Connection to the Arduino Grove Board through IntelliJ IDEA using Arduino IDE StandardFirmata to connect.

PROCEDURE: IntelliJ IDEA is used to create a code that reads the moisture sensor, and different voltage values for wet and dry soil were assigned based on previous testing. The code then reacts depending on if the soil is wet or dry and pumps a sufficient amount of water to the plant. The system is then repeated on a loop; reading, and displaying the data (e.g., moisture reading, pump state, etc.) on the OLED screen, along with a live moving bar that grows longer depending on how dry the plant is. Ideally the line should be around halfway filled for the plant to be in preferable conditions. **TEST:** In order to test that we have a working system, a dry plant can be simulated by holding the moisture sensor in the air. This allows us to view the pump enabling and updating the OLED screen, as well as send a signal to and enable the pump. To test the opposite, we can simply dunk the moisture sensor into our water source

afterwards to clearly see that the pump then stops watering after reading a sufficient moisture value. This was demonstrated near the end of the video demonstration. **RESULTS:** The plant successfully reads the moisture sensor voltage value and reacts to it depending on what is required, and the system is on a loop allowing it to be self-sustainable and essentially take care of itself so long as a water source is present. The system is also very vocal in the sense that it displays what is being executed on the OLED screen, reading "Pump: ON" when the plant is being watered and "Pump: OFF" when it no longer requires any more water. The moisture sensor values along with the live graph/line on the OLED screen provide an excellent visual representation of the plant's current state and conditions. **LEARNING OUTCOMES:** Based on the learning outcomes outlined in the project document, all objectives have been crossed off throughout the creation of this system:

- A timely amount of debugging and troubleshooting took place to display anything on the OLED screen, as well as pump water using the pump. This was easily counteracted with the use of the "try" and "catch" methods; the errors in the code are effortlessly highlighted (rather than being sent on a "wild goose chase" looking for what's wrong with your code).
- The given requirements were easily met, as the plant does what is expected of it without the need for third party interference by implementing different suitable APIs such as Firmata4j, JavaFX and Open CSV.
- We also accomplished the creation of an event-driven application, using sensors and actuators such as the moisture sensor and MOSFET board to connect events to physical actions (e.g., based on the readings of the moisture sensor, a physical action of the pump watering the plant occurs.
- Using different methods taught throughout the course, obstacles were overcome using "an object-oriented language" such as java and solved on the computer. These teachings will come in very handy in a software engineering position.

CONCLUSION: The plant reads its voltage values consistently, waiting for the need of water and pumping in an adequate amount as required, automatically stopping when enough water has been supplied. The live updating bar and moisture readings provide an excellent understanding of the system while it's running. In conclusion, the system is efficient, and fully functional allowing the plant to water itself as intended.

Moisture Sensor Readings

Not immersed in anything (air): ≈ **780**

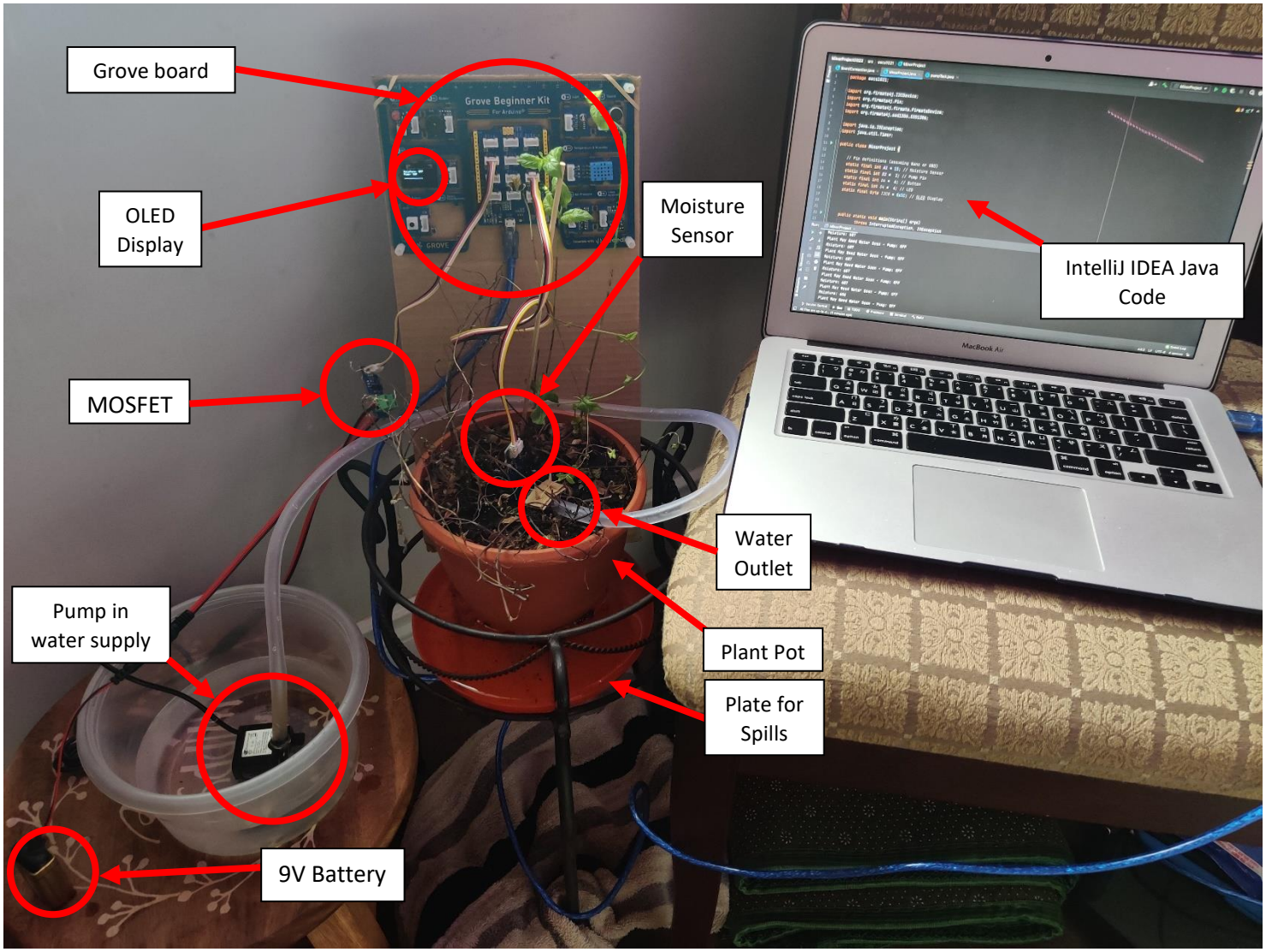
Immersed in dry soil: ≈ **700**

Immersed in water-saturated soil ≈ **620**.

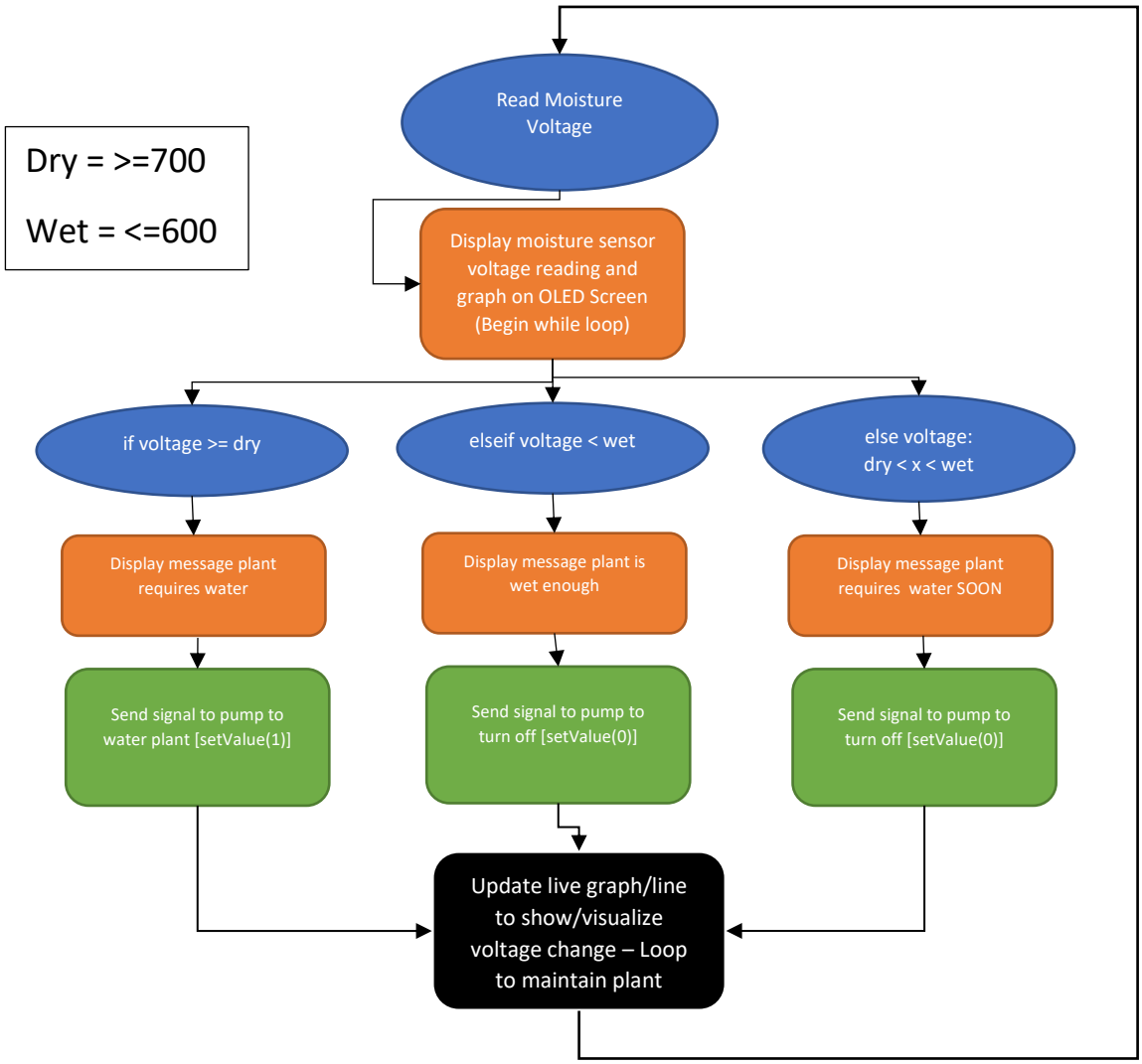
Immersed in a clear glass of water ≈ **530**

These values were measured prior to the creation of the entire self watering system, to get an idea of what values were to be considered wet and dry. With a bit of fine tuning and troubleshooting, the values were essentially perfected, reacting to the soil and determining whether it requires water or not.

Plant Hardware Setup



Flow Chart



Video Demonstration of Minor Project

Click Here for Video Link!

(In case above link doesn't work):

https://www.youtube.com/watch?v=BpeLGeHq-0M

P.S I apologize if the video seems rushed as it was EXTREMELY difficult to fit the entire setup into a video less than 1min. The data is also displayed in the IntelliJ console; however this is not too important as we can see the same data on the built in OLED display.