Anthony Ma ID# 219380047

EECS 1021 Object-Oriented Programming from Sensors to Actuators

2023/03/29

Minor Project

INTRODUCTION

Getting tired of watering your plant all the time? Well, this is where the Automated Plant Watering System comes in.

CONTEXT

This Automated Plant Watering System as stated before, utilizes object-oriented programming to water your plant for as long as you like without you having to interact with it. Why is this important? Many people tend to forget to water their plants due to having a busy life or just being forgetful which leads to improper care of their plants. Like any other life on earth, plants require proper care and sustenance and to accommodate those that are not always able to give that to their plants, this Automated Plant Watering System is the perfect solution for them.

TECHNICAL REQUIREMENTS / SPECIFICATIONS

(CLO 2)

- The system should use firmata to incorporate the grove Arduino kit.
- The system should be able to make constant readings for the moisture level over multiple days.
- The system should be able to pump water into the plant when in a state of being dry or almost moist.
- The system should display moisture sensor and water pump information on the OLED screen.
- The system should include a graph that displays the moisture readings as voltage over time.
- The system should use an ArrayList to store moisture values to be used in the graph.

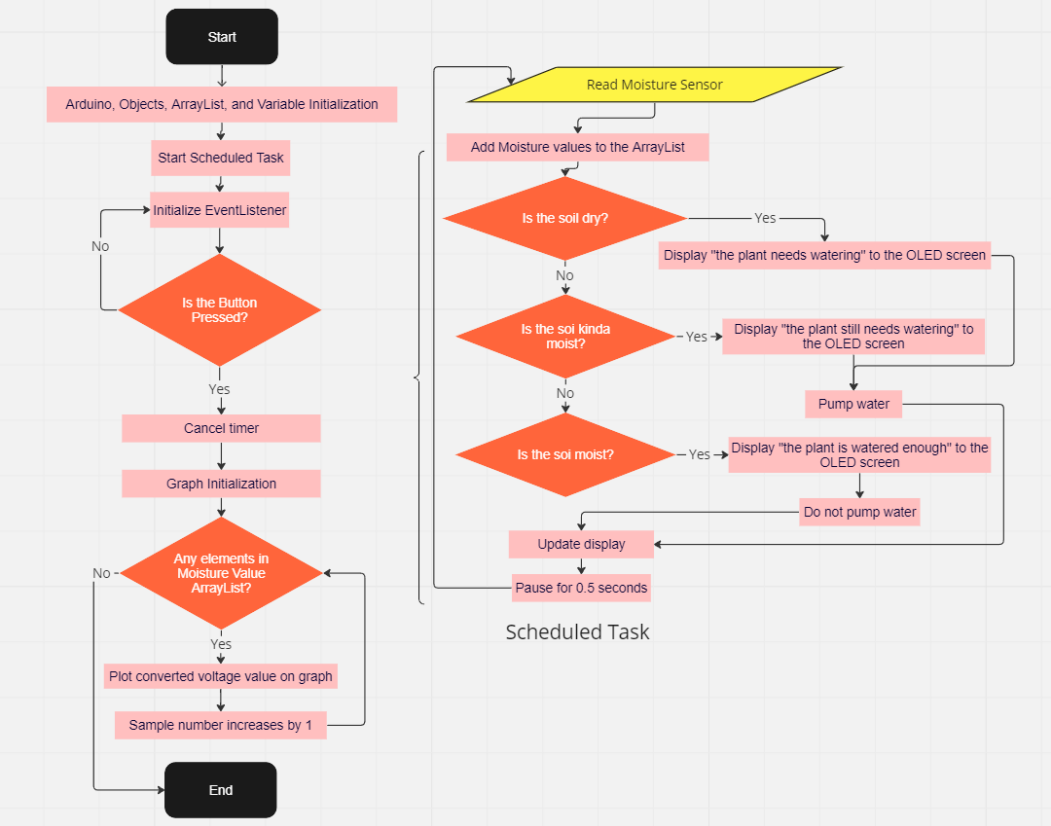

Figure 1 Flowchart
https://miro.com/app/board/uXjVMV67LHY=/?share_link_id=896113089276

(CLO 4)

I planned on dealing with events in this program by pressing the button on the arduino to stop the program and display the graph.

(CLO 5)

I also used an ArrayList in this program to gather data from the moisture sensor during each execution of the task, I then called the method that returned the ArrayList into a for-each loop to be iterated through and graphed.

COMPONENTS LIST

- Grove Arduino Kit
  - Utilizes firmata to operate in intellij.
- MOSFET
  - Connects to the D2 pin to output signals to the water pump.
- Button
  - Serves as the EventListener that creates the graph
- Water Pump
  - Receives signal from the MOSFET and power from the 9V battery.
- Moisture Sensor
  - Connects to the A1 pin by rewiring the yellow and white wires in the connector and records moisture through analog inputs.
- Plant
  - ZygoCactus Plant.

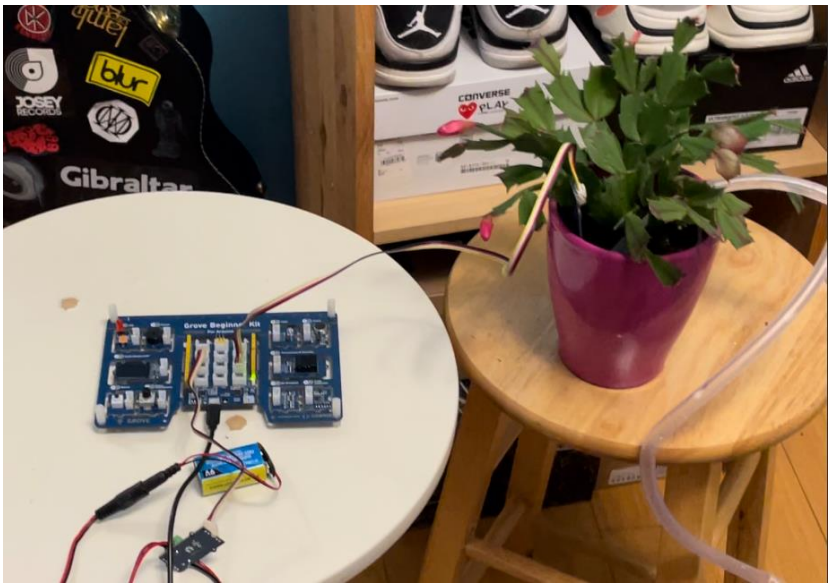

Figure 2 Setup

PROCEDURE

When creating my project, I referred to my experience from the previous minor project in EECS 1011 and what I learned about Java/object-oriented programming in class. At first, I wanted to incorporate two concepts that I learned in 1021 which were Timer/TimerTask and EventListener. My first focus was to translate my work from Matlab into intellij as a starting point which proved to be a bit rocky but successful in the end. I quickly realized that using the strategies I used in the Matlab would be unsuccessful so I decided to change the original usage of for loops and tic/toc into Timer/TimerTask so that I could have constant readings for the moisture with set intervals I could choose and implement a separate class called WateringTask to store the moisture reading method. Next, I implemented the OLED screen, this portion of the procedure was fairly straightforward as once I figured out that I could just use Boolean values and if statements to determine what to write on the OLED screen, I was able to have them display specific information depending on if the pump was running and what state the plant was currently in. After I completed the OLED screen, I encountered a roadblock once I began trying to implement a dynamic graph. At first, I believed that I would be able to use JavaFx to create a graph but it proved to make my program far more complex than I wanted so I was at a loss until I remembered the StdLib. Afterwards, I was able to create a proper graph by using an ArrayList to store the Moisture values over a set duration, and then graph each point using a for-each loop. For the sake of the demonstration as stated by the professor, the program was made specifically to store moisture values for a some time and then stopped when the button is pressed to then graph the data as a static graph but this program may be modified to easily run for multiple days like in the specification. **(CLO 5)** For the fifth learning outcome, my project changed drastically from start to finish as evidently I had many classes I tried to implement but did not find success in JavaFx along with concepts such as a live dynamic graph. In the end, I was able to implement classes that served the specifications better such as Timer/TimerTask and StdLib and proved to be successful even though it was a drastic change from the beginning.

TEST
(CLO 1)

I tested that the system worked by running the program several times and including System.out.println()s for variables to check that their value was what I was expecting. I also debugged the program by keeping track of errors and paid close attention to the exceptions as they were able to direct me to places where I made an error such as mislabeling a variable or forgetting to initialize objects. For the graph specifically, I first made sure that it worked with the original analog values of the moisture sensor, and then through trial and error, scaled the graph to suit the voltage values.

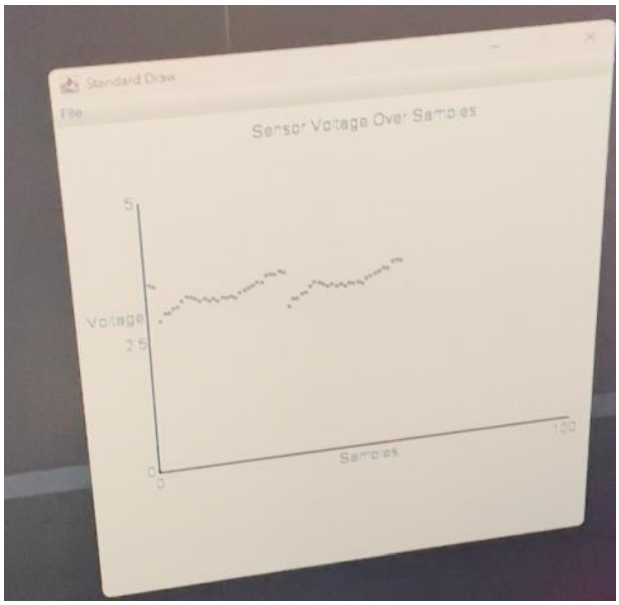

Figure 3 Graph

LEARNING OUTCOMES

Yes, the project addresses all of the learning outcomes.

- **(CLO 1)** I demonstrated the ability to test and debug a given program through countless tests and reiterations of my program through many System.out.println() methods to check the value of my variables. I reasoned about its correctness by constantly checking the value obtained and making sure that the pump behaved consistently and would not react to the wrong value.

- **(CLO 2)** I demonstrated my ability to build an application that meets the given requirements from a given problem specification and API, as I was able to implement a solution that used Firmata4j and StdLib. These classes were appropriate for this project as Firmata4j allowed for the use of Arduino hardware and StdLib allowed for the use of StdDraw to create a graph.

- **(CLO 3)** I demonstrated the ability to use ready-made collections to solve the problem of this project by storing my moisture sensor data in an ArrayList during each execution of the task.

- **(CLO 4)** I demonstrated the ability to build an event-driven application by using the if, else-if and else conditional statements to create a state-machine that would communicate with the Arduino hardware to determine the state of the soil in the plant and act accordingly to the state determined. It also incorporates listeners to stop the timer and graph the ArrayList.

- **(CLO 5)** I demonstrated the ability to use an object-oriented program to solve engineering problems by utilizing classes and creating objects using instances of those classes such as my WateringTask class that extended TimerTask where I created an object from to use its methods in the main class. Over time my implementation changed as I initially used JavaFx but was unsuccessful, so I decided to use TimerTask and StdLib instead.

CONCLUSION

In conclusion, I was able to meet expectations and achieve all of the desired learning outcomes for this project. I may have struggled here and there, but in the end, I was able to complete the project successfully.