ANDREW ATTFIELD
2024-04-04
EECS 2021

EECS 2021 PLANT PROJECT

ANDREW ATTFIELD
2024-04-04
EECS 2021

INTRODUCTION

The following is a technical report regarding my EECS 1021 plant project submission. The project required me to develop a java program using IntelliJ and a Grove Arduino board that could automate the watering of a plant depending on its moisture level. The project itself has specific CLO outcomes that need to be shown and they will be listed in this report and/or in the video.

CONTEXT

This java program automates the watering of a plant. It utilizes a sensor to keep a constant reading of the moisture level of the soil. It uses this information to make a state-based decision and operates the water pump at different intervals based on the level of moisture. This ensures that the soil stays wet enough for the plant to grow and survive.

The reason this is important is that this lets us automate the process of watering our plant. This means that this program can run all the time without human attention. This solves the risk of the plant dying to human forgetfulness and lets us focus our attention on other tasks.

TECHINCAL REQUIREMENTS/SPECIFICATIONS

Firmata4j is a suitable API (Application Programming Interface) for this plant watering project. This is because it is necessary to connect our grove board to IntelliJ so that we can code programs using the device. It lets us control the digital and analog pins, send, and receive data, and handle sensors and actuators.

The StdLib API is suitable for our project because it has the methods to let us graph our data. This graph is important because it lets us track the moisture trend of our plant and can help us recognize why a plant may not be growing properly.

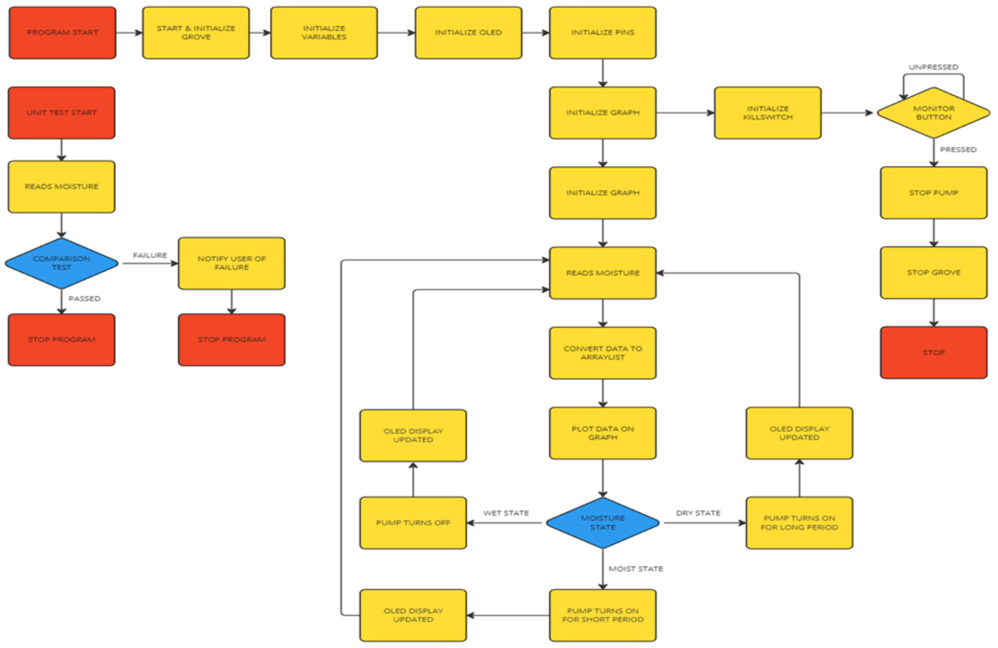
In terms of events, I used a state-machine to drive my while loop for watering the plant. The states of the plant depend on the moisture of the soil, and a state of either dry, moist, or wet soil will be assigned to the plant. These three different states all trigger a different method and controls the water pump in different ways. If it is dry, the water pump turns on for a longer period, if its moist, it turns on only briefly, and if it is wet, the pump turns off.

I also used a java listener to “listen” to the button on my grove board. It constantly monitors the button and if it any time the button is pressed, the event will be seen and the method will disable the pump, notify the user, and then disable the board. This is to be used as a kill switch for when the pump malfunctions or spillage occurs.

Normally the moisture sensor gathers data as an integer value within my code, so I used an ArrayList command to store it in a continuously growing single line array each time the loop runs.

ANDREW ATTFIELD
2024-04-04
EECS 2021

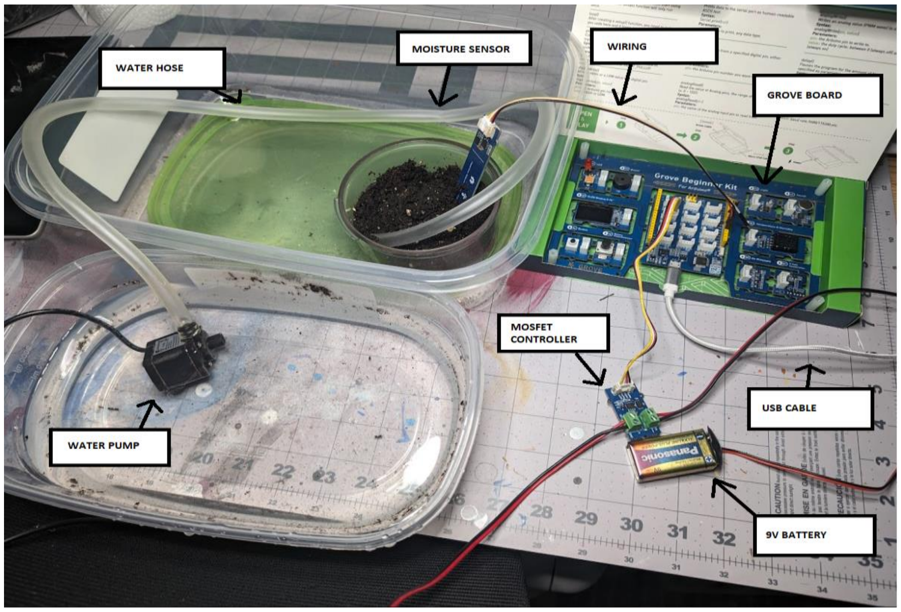
TECHNICAL REQUIREMENTS/SPECIFICATIONS CONTINUED



COMPONENTS LIST

- Grove Beginner Kit for Arduino
- Moisture Sensor
- MOSFET Controller
- Wiring
- Water Pump
- Water Hose
- USB Cable
- 9V Battery

COMPONENTS LIST CONTINUED



ANDREW ATTFIELD
2024-04-04
EECS 2021

PROCEDURE

The object-oriented concepts I used in my code are listed below:

Abstraction (hiding implementation details) – my plantProject class abstracts the entire operation of my watering system like moisture readings, water pump control and the OLED screen display.

Encapsulation (grouping attributes and methods into a class) – my attribute moistureData and method wateringSystem.wateringMode are encapsulated in my plantProject class.

Inheritance (when a class gets the attributes and methods from another class) – my buttonListener and wateringSystem classes are separate from my original plantProject class but use attributes that belong to it.

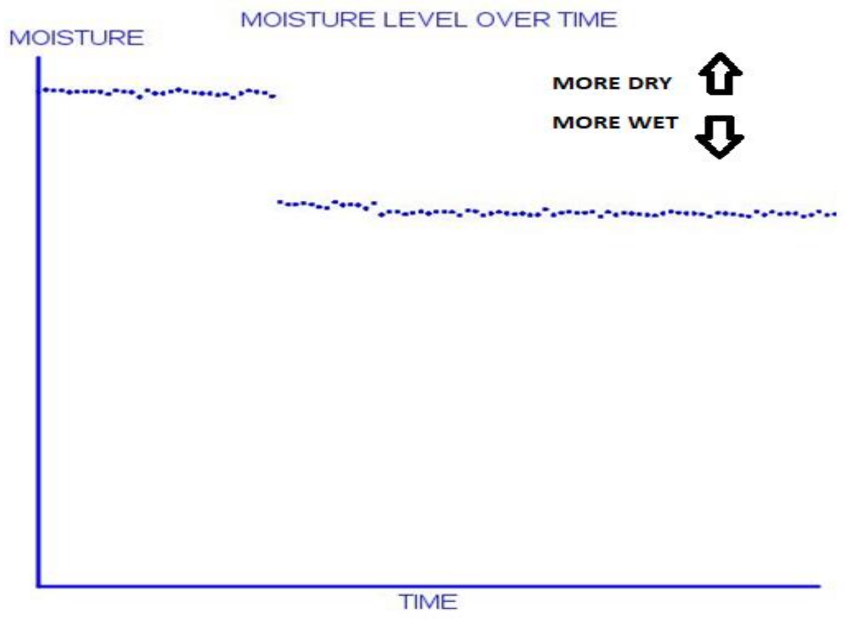
Polymorphism (object is treated as an instance of their parent class) – the use of overrides in my buttonlistner to make changes to existing methods is an example of polymorphism.

PROCEDURE CONTINUED

One way I improved my solution was I originally wrote the program as a very basic, start to finish code that didn't incorporate inheritance. All the code was under one class, and it became very crowded with code. I then created new classes containing methods for controlling my water pump and my event listener for the kill switch to help clear up the original class.

TEST

When I first started creating the program I made a very simple code that just connected one part at a time and would read its values to be sure that the part worked. I first started with my moisture sensor, then OLED screen, and then the water pump. Once I confirmed all were working, I created a unit test class that took a moisture reading with the sensor and tested against an assigned value and deviation.



ANDREW ATTFIELD
2024-04-04
EECS 2021

LEARNING OUTCOMES

I was able to address all the learning outcomes described in the original project document.

- CLO1 – Testing – used a unitTest class that compared my moisture sensor values to a set standard value with deviation and resulted in a pass/failure.

- CLO2 – API – used Std.lib API for graphing and Firmata4j API for utilizing the grove board.

- CLO3 – Ready-made collections – converted my moisture sensor data over to a continuously growing ArrayList object called moistureData. This array list was used to plot the graph and decided what state the plant was in.

- CLO4 – Event-Driven Application – Used a state-based system for my while loop, dry, moist, and wet state. Also used a listener in a separate class to constantly check for button event to occur and acted as a kill switch.

- CLO5 - Object-oriented elements – my code implemented inheritance, where my waterPump and buttonListener classes were separate to my main plantProject class but utilized the attributes from that class.

CONCLUSION

All in all, this plant project proved much more challenging than last semesters plant project due to the complex nature of java. There is a lot more terminology and nuances regarding the code that was more difficult to learn and implement.

My biggest struggles were regarding the physics of the pump, I went through a lot of failed tests because of heigh difference, gravity, and an inflexible water hose.

Fortunately, after some research and modification, I was able to create a successful code that would work as an automated watering plant system.