Dynamic Plant Watering System

Howard Li
American High School
Fremont, United States
howardl94555@gmail.com

***Abstract*—The purpose of this paper and project was to create a dynamic plant watering system that adjusts based on weather and sunlight data. The system utilizes data from the National Weather Service, as well as the plant's orientation and location to adjust the watering frequency. A working proof-of-concept was built and tested on an indoor potted plant. Testing showed that the system was able to successfully adjust based on local weather conditions and changing sunlight hours. This project serves as an example of how open-source data and resources can be used to make day-to-day tasks much more effortless.**

I. Introduction

Watering a plant is usually considered a mundane task that must be done out of pure necessity. It requires people to keep track of watering schedules manually. Sometimes people use assistants, such as apps, to help them keep track, but those apps aren't fully comprehensive and only suggest a fixed interval between watering sessions. Other projects use local temperature and humidity sensors to determine when plants should be watered, but they require many components and don't give the user the full picture.

This project aims to create an automatic watering system that uses an adaptive algorithm to dynamically update its watering schedule, based on weather data collected from external sources. No local sensors are required, and all parts, except for the servo and Raspberry Pi and servo, can be 3D printed.

Inspiration for this project was drawn from existing projects, such as MudPi, which will be described in further detail later on. Additionally, this project utilizes several open-source libraries in order to retrieve data and control the servo. This paper will explain the technical aspects of the project, such as the adjustment algorithm, hardware components, and the results.

II. Related Works

A. MudPi

MudPi is a similar project that serves as partial inspiration for this one. It uses local temperature and humidity sensors to determine its watering schedule. Because of this, it requires the use of many electronic components, adding layers of complexity that the everyday person would struggle with.

B. NOAA API

The National Oceanographic and Atmospheric Administration's National Weather Service API played a major part in making this project possible. It provides weather forecasts for specific locations, used to gather data for calculating my watering schedules.

III. Software

All of the programming for this project was done with Python 3.12.4 and all of the code was run on a Raspberry Pi 4 Model B with 8 GB of RAM. The full project files can be found at my Github repository.

A. Packages Used

This project uses the following Python Packages, which can all be found on PyPI.

- Datetime: Used to track date and time and determine the length of intervals between watering.
- Pgeocode: Used to convert a user-provided zipcode into geological coordinates (latitude, longitude).
- Time: Used to pause the program for intervals determined using Datetime.
- Gpiozero: Used to interface with RPi4B GPIO pins and control servo actuator to dispense water.
- Numpy: Used to calculate the rolling average of sunlight data across 7 days
- Noaa-SDK: A Python package that serves as a wrapper for the NOAA National Weather Service API.
- Suntime: Used to find sunrise and sunset times, which are used to calculate the total amount of sunlight in a day.

B. Algorithm

The adjustment of the watering schedule is based on a calculation of how much light the plant will receive in a day. This number is based on several factors, such as weather conditions, plant orientation, and time of year.

Weather conditions are determined by processing the short forecast that is queried from the NOAA NWS API. A possible example of such a forecast could be, "Partly Cloudy." This forecast is split into individual words and parsed to search for key phrases such as "rain," "cloud," "fog," and, "sun." If a key phrase is found, the short forecast is assigned a numerical value W, in foot-candles, that corresponds with the amount of sunlight present during those conditions. For this project, "Rain" was assigned a value of 100 foot-candles, "Fog" a value of 400 foot-candles, "Cloud" a value of 700 foot-candles, and "Sun" a value of 1,000 foot-candles based on reference sources. [2]

Plant orientation is user-inputted, and it can be one of four possible values: "N," for North, "E," for East, "S," for

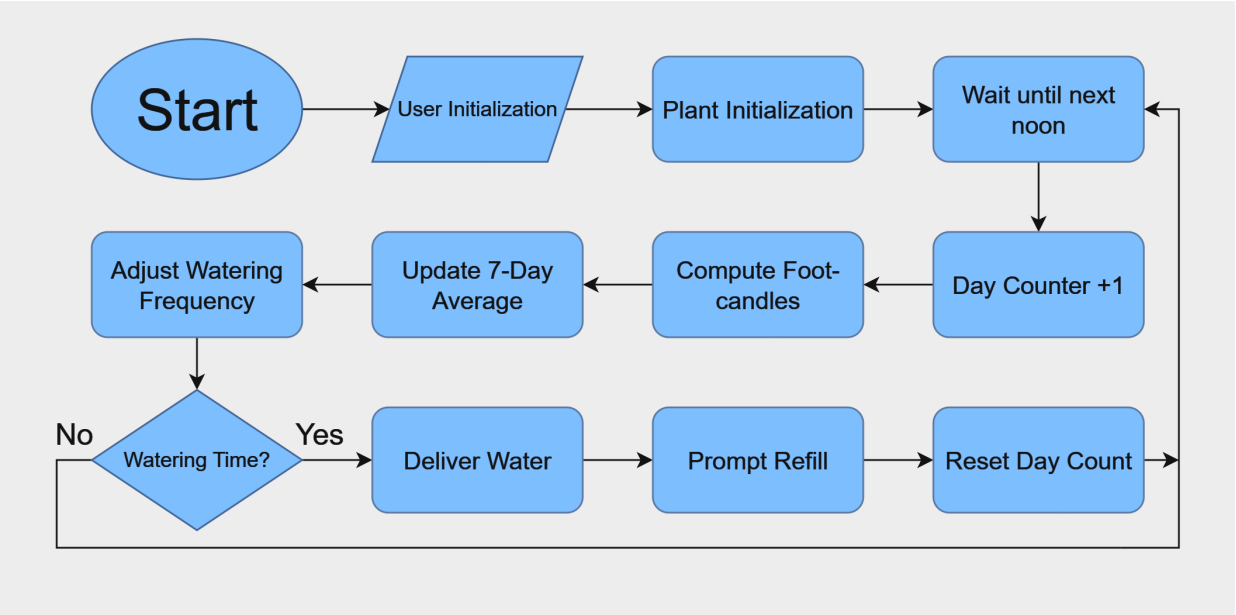

Fig. 1. Software Chart

South, or, "W," for West. A directional foot-candle value, D, is assigned based on this value. North is assigned a value of 100 foot-candles, East and West are assigned a value of 500 foot-candles, and South is assigned a value of 1,000 foot-candles. [1]

The time of year affects sunlight levels because it changes how long the sun is out every day. This value, T, is calculated by finding the difference in time, in hours, between sunrise and sunset on the current day.

These three variables are used in the following equation to calculate the total amount of sunlight that the plant will receive in one day, S, in foot-candle * hours:

$$S = (W + D) * T \tag{1}$$

Starting from noon on the day after the program is first run, it keeps track of the number of days since the plant was last watered. Every following day at noon, the total sunlight for that day is calculated.

Once the value for each day is calculated, it is stored in a NumPy array, and a 7-day rolling average is computed and used to adjust the watering schedule. The purpose of this is to smooth out unusual weather events and highlight repeating weather patterns, which have a much more significant effect on plant growth.

The watering frequency is adjusted by mapping the 7-day average sunlight value to a number between -1 and 1 based on calculated theoretical minimum and maximum sunlight amounts. Then, this value is multiplied by a base watering frequency, dependent on the type of plant specified by the user, and added to the original base watering frequency. Currently, the code contains a dictionary of 5 different plants and their base watering data. The equation for such a calculation is:

$$f_1 = (f_0 * scale) + f_0 \tag{2}$$

where f_1 represents adjusted frequency, f_0 represents base frequency, and $scale$ represents the mapped multiplier value.

Once the count of days since the last watering equals the adjusted frequency, the servo-controlled actuator is opened, dispensing all of the water inside of the reservoir into the pot. Then, the user is prompted through the terminal to refill the reservoir for the next watering session and the day count is reset.

IV. Hardware

All major components of this project's proof-of-concept are 3D printed, except for a TowerPro SG90 Servo and a Raspberry Pi 4 Model B.

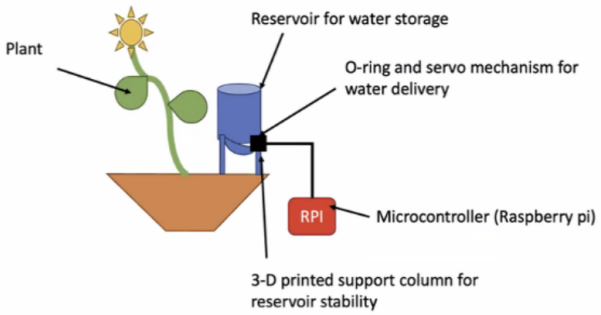

Fig. 2. Hardware Diagram

A. Parts List

The main physical parts of the working concept are as follows (Fig. 2):

- Reservoir - a 3D printed container that holds the water to be dispensed on the plant. It has a small hole at the bottom that is plugged up by the actuator and sealed with an O-ring.
- Actuator - a 3D printed piece that is used to plug up the bottom of the reservoir, and is moved in order to let water flow out of it.
- Servo - a TowerPro SG90 digital servo controlled by a Raspberry Pi that is used to move the actuator and dispense water.
- Raspberry Pi 4 Model B - a compact microcontroller that runs the project code and controls the servo. It requires 5V power and the code needs an internet connection to function.
- Supports - 3D printed columns are used to hold the reservoir, actuator, and servo above the soil and provide stability.

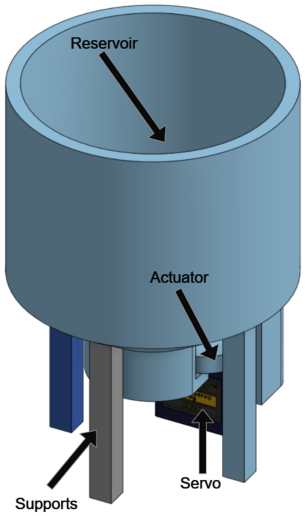

Fig. 3. Labeled picture of CAD

B. Development

A design consisting of a reservoir and actuator was chosen because it would allow for consistent dispensing of larger amounts of water while having few moving parts and a low design cost. The system was modeled and assembled using Onshape, a cloud-based CAD software (Fig. 3). Then, all parts that needed to be 3D-printed were imported into the Bambu software, sliced, and printed on a Bambu A1 3D printer using black PLA filament.

Originally, the servo and program code ran off a Raspberry Pi 5. However, the Pi 5's lack of hardware PWM channels meant that PWM signals were generated using software, causing servo control to be jittery and imprecise. An older model, the Raspberry Pi 4, had hardware-controlled PWM signal channels, and replacing the Pi 5 with a Pi 4 Model B fixed all servo jittering issues while retaining necessary features such as wireless internet connection.

The original actuator flap was too small, making it unable to fully seal against the O-ring and leading to small leaks during testing. It was replaced with a larger flap which was also lightened to decrease horizontal torque acting on the servo shaft. This new actuator flap functioned properly and testing could continue.

V. Results and Conclusions

A. Results

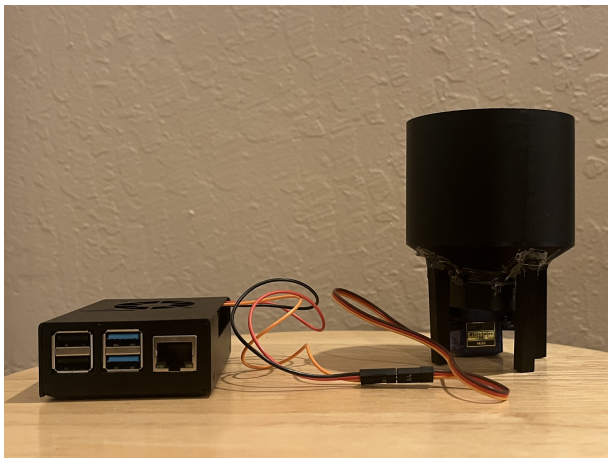

Fig. 4. Picture of complete system

The algorithm designed during this project successfully retrieved information from the user and National Weather Service API and used it to automatically adjust the base watering schedule to account for changing weather and sunlight conditions. The hardware setup (Fig. 4) was able to successfully store the required amounts of water and dispense it at the specified intervals.

If this project were to be repeated or further developed, more resources should be designated toward improving the hardware setup. The current system is a rudimentary proof-of-concept, but spending more time to design and build a better watering apparatus would yield better results.

VI. Acknowledgements

I would like to thank my mentor, Alex Thornton, for providing significant guidance during the completion of the project and the writing of this paper. I would also like to thank my parents for providing me with the funding I needed to purchase the materials used for this project.

References

[1] UmD Extension. Lighting for Indoor Plants — University of Maryland Extension — extension.umd.edu. https://extension.umd.edu/resource/lighting-indoor-plants/. [Accessed 16-10-2024].

[2] Take Three Lighting. How much light is enough? light Levels - Foot Candles — takethreelighting.com. https://www.takethreelighting.com/light-levels.html. [Accessed 16-10-2024].