Green Tech: Automated Plant Watering System

Adam Carranza
University of St Andrews
St Andrews, Scotland
Amc47@st-andrews.ac.uk

ABSTRACT

This paper covers the process and design of an automated plant watering system which includes a 5V DC motor, raspberry pi, graphics, moisture sensor, temperature sensor, and some electrical engineering components. The user can schedule when the plant should get water and also have the ability to add more water with the tap of a button.

Author Keywords

Plant, android, java, python, Raspberry Pi, electrical components, graphics, firebase

INTRODUCTION

I was inspired to create this system when a family member was having trouble watering her orchids. As you may know, orchids are notoriously difficult to keep happy; any inconsistencies in watering could mean the death of the plant. That was when I decided to create a consistent system that measured the moisture level and watered the plant based on those measurements. In addition, I created some graphics (smiley face and sad face) to personify the plant.

WATERING

Initially I programmed the system to water every time soil moisture was low. For example, if the sensor inside the soil noticed dry conditions, the motor would turn on to water the plant. This; however, would be dangerous for some plants because consistent moisture is not necessarily a good thing for all plants. For one, it may cause root rot in bonsai plants. That is why I allowed the user to decided how much the system waters the plant. The user would first be asked how often the system should water the plant then how much water to provide. The system would then follow the orders and maintain measurements by the minute. Furthermore, I experimented with my own plants to determine what soil moisture level was low and what was too high. I found a good amount of moisture was between 450 and 830, which was shown as "happy" on the pi and the app.

THE MOTOR

The 5V DC motor sucks up water and disperses it to the plant using another tube. It is connected with a power source of its own (the battery pack) to prevent short circuits in the pi.

THE SENSOR

The Stemma sensor has a moisture reader and a temperature gauge which the pi records.

THE OTHER ELECTRICAL COMPONENTS

In addition to the battery pack, DC motor, and stemma; I used jumper cables, two breadboards, a motor driver chip, GPIO extension ribbon, a raspberry pi, and a display

INTERFACE

There are two interfaces. The first is on the raspberry pi. It simply asks the user "How many times per week would you like to water your plant?" and "How big is your plant?" This information is then collected to calculate the next watering time and the amount due. The second interface is the android app which can be seen in the photo on page 2.

THE GRAPHICS

The graphics are an additional feature to the interface and changes according to the plant measurements just like the app.

THE ANDROID APP

The app reads data it gets from firebase. The data on firebase is constantly changing eg the moisture level changes in the soil and the raspberry pi sends the new data to the app. The app changes according to the plant's measurements. The user can then determine if the plant needs more water on top of the amount already scheduled.

ISSUES

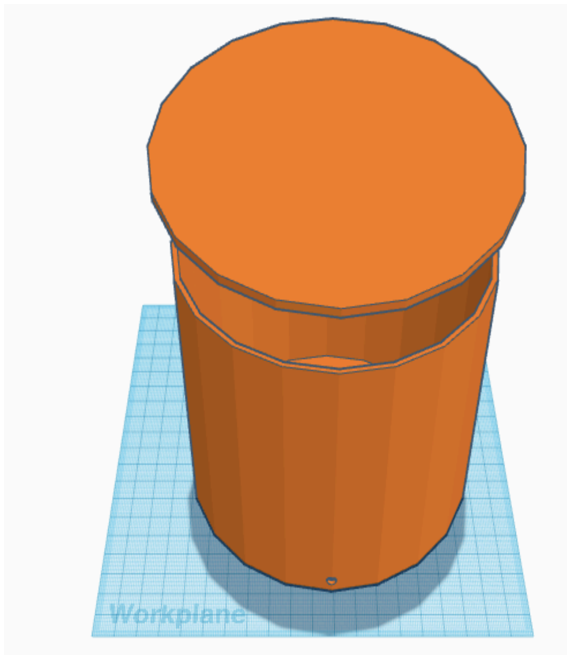
The most difficult issues, by far, was the eduroam wifi. Honestly, I was so tempted to just not use android firebase because of how heinous it is to connect the raspberry pi to eduroam. I ended up breaking the wifi on the pi altogether and had to spend 2 hours in the IT department to try to fix the issue. Thankfully, in the end, I was able to connect to the wifi. Another issue I ran into was going from java to python. It took a bit to switch my way of thinking into python format. This was especially difficult when trying to make the countdown and checking the plant measurements run concurringly.

THE SHELL

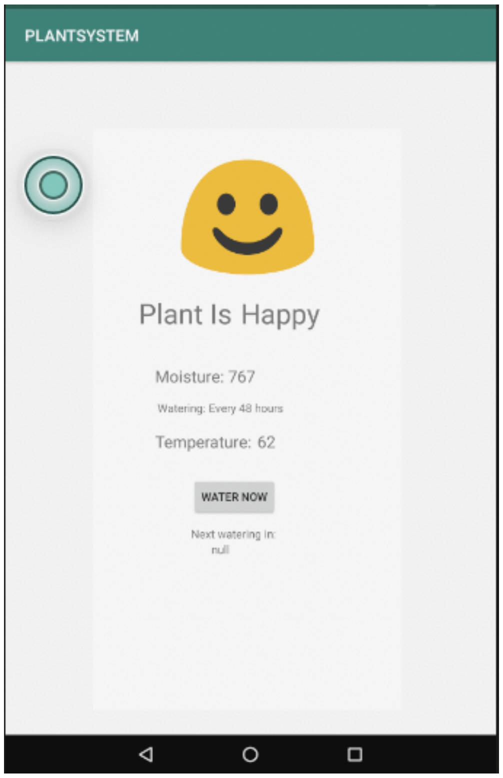
The encasement holds the many components needed to make the system run. Moreover, I used heavy-grade Velcro to hold all the components inside. This included the display in the front, the battery pack, the breadboard, and the DC motor. Additionally, I covered much of the box with waterproof tape which should protect the structure from water damage. I wanted to use the 3D printer to create a water container that had a perfect fit for the in-tube but it would've taken too long. Below is a rendering of the 3D design.

THE FUTURE

The main reason I took this class was to find ways to merge computing and environmental protection. This system is the beginning of that journey. Although this project is small in scale, I can see its usage in home gardens or even on farms to reduce water usage.



Example of the now scraped 3D design for the water container. Notice the 5mm hole in the bottom which fit the tube perfectly.



PLANTSYSTEM

Plant Is Happy

Moisture: 767

Watering: Every 48 hours

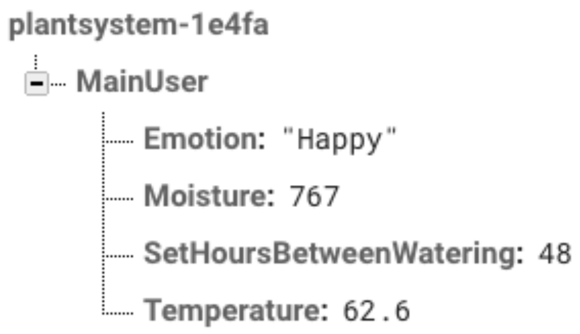
Temperature: 62

WATER NOW

Next watering in:
null

Example of the android app interface that updates data based on the sensors in the plant using firebase.

This graphic shows on the raspberry pi display when the plant is moist. Similarly, a red background and frowny face shows when the soil is dry.



```
plantsystem-1e4fa
 ⊟ MainUser
     Emotion: "Happy"
     Moisture: 767
     SetHoursBetweenWatering: 48
     Temperature: 62.6
```

Here is the online firebase data that allow the pi to communicate to the app.

ADITIONAL NOTES:

1) To see program that runs in the pi open the “Smile.py”
2) To see the program that runs in the android app see “MainActivityCopy.java” or “PLANTSYSTEM” for the whole folder.

3) Shared OneDrive link: https://universityofstandrews907-my.sharepoint.com/:v:/g/personal/amc47_st-andrews_ac_uk/EXzXySWtg6pPqbJLUh0aACABp1ObmJDr8cUYxMheNb12cw?e=8Ix0kL

{link may only work when opened in the regular dotx word document (provided) }