Irrigation Balance Estimation and Evapotranspiration Forecast

Team 7 Final Report

ECE 455/555 Embedded System Design
Fall 2016

Instructor
Dr. Wei Gao

Ryan Daniels
Paul Proctor
Adam Rosenbaum

December 1, 2016

1 Introduction

Precision agriculture has been a growing field of interest for research and development in the Internet of Things (IoT), cyber-physical systems, and wireless sensor networks. Many of the challenges farmer's face are akin to urban gardeners and residential consumers, such as the history, present, and predicted future condition of plants, crops, soil, and root environment. This new market is accessible due to reduced sensor and associated component costs and wide use of smartphones. However, most precision agricultural systems require large initial investment and extensive training to setup, operate, maintain, and understand the data.

Most urban gardeners and residential consumers do not have the extensive background knowledge that most farmers do. Therefore, for any system to be successful it must provide an easy to use interface and understandable graphics and results. Additionally, there needs to be justification for predictions that instill confidence, therefore transparency in calculations and methods need to be provided. As such, our goal has been to design a small-scale precision agricultural system that targets the most typical issue experienced by plant owners. This issue is if the plant currently needs to be watered and when to water it next. In this paper, we describe a full stack solution to alleviate the usability burdens typical of current systems specific to irrigation management along with preliminary experimental results.

2 Project Update

In our project, we initially planned to use data collected by MOIST (see appendix A) to predict evapotranspiration. However, we determined that the goals of MOIST and that of the term project did not fit, thus we modified our project scope to include a sensor platform to measure and report on a household plant's soil moisture content and environmental data. After the mid-term presentation, there was not enough time to run another experiment with the additional sensors or to configure the backend to incorporate the sensor data for display or evapotranspiration prediction. However, we have performed significant literature review and understand the phenomena and how to incorporate sensor data into the standard calculations.

3 Background

Agrohydrology is regarded as the study of hydrological processes in soil environments that are influenced by soil mechanics, biophysical processes, and the microclimates surrounding plants [1]. Water balance includes the total water cycle and can be generalized by equation 1 that consists of surface water and groundwater. In the water balance equation, there are five components; precipitation (P) that equates to evaporation (E), evapotranspiration (T), surface water runoff (Q), and ground water flow (G) [2].

(1)

3.1 Surface Water Balance

Surface water balance (SWB) is a subset of water balance described in Agrohydrology in which only the top soil layer and soil-air interface are accounted for (Fig. 1). Farmers typically use either a full water balance measurement system or SWB to determine irrigation scheduling. When calculating water balance with indoor houseplants, runoff and ground water flow can be assumed to be zero, thus leaving evaporation and evapotranspiration. This calculation provides an absolute measurement for surface water balance, yet complex relationships still exist. The Penman Equation and derivative works explain these relationships.

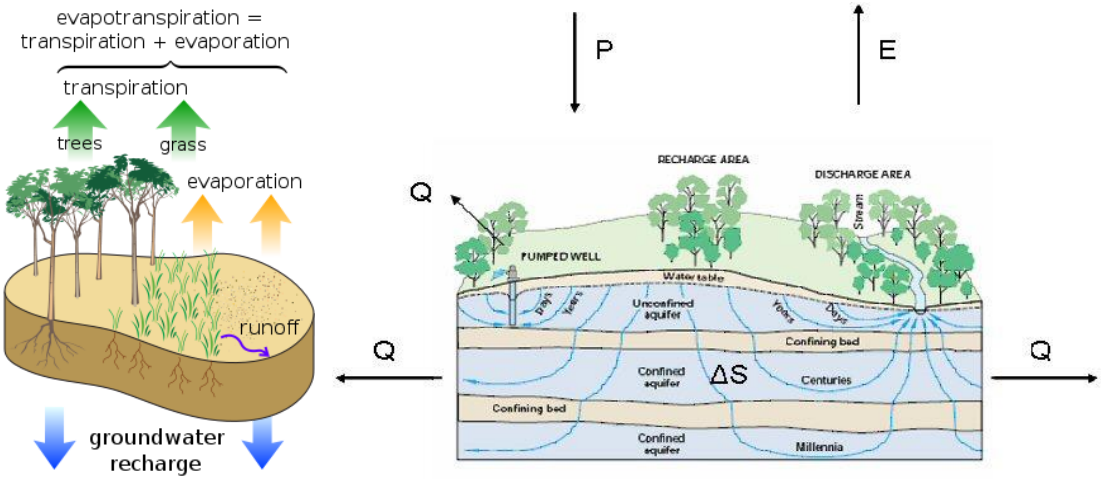


Figure 1. Water balance illustrations [3 & 4].

3.2 Evaporation and Evapotranspiration

Evaporation is process of water molecules transitioning from the liquid state to the gaseous state. The rate of evaporation from leaves or soil is dependent upon many factors, but vapor pressure is the primary parameter. About 10% of all the moisture in the atmosphere originates from evaporation from plants [5]. The process of water moving through a plant and evaporating from leaves and other plant sections is called transpiration. Transpiration is directly proportional to plant stress and air temperature which causes the stomata or the pores plants to open or close to release or retain moisture, respectively [6]. The combined process of land or ocean evaporation and transpiration is called evapotranspiration (ET) [7] and reference 1 from midterm]. This is the primary parameter used to predict a plant's healthy watering range and specific irrigation balance and scheduling for a specific soil type.

Forecasting ET's real-time value, rate of change, and effect on the climate provides key insights for climatologists, hydrologists, and irrigators across the United States and the world. In 1948, Howard Penman developed what is now called the Penman Equation (1948) that describes ET from open water and homogeneous land surfaces [8].

The actual parameters in the Penman Equation are impractical to know usefully on a large scale, such as farms, but applicable to small scale gardens or individually plotted plants. These parameters include the use of daily mean temperature, wind speed, air pressure, and solar irradiance to predict ET. However, the original Penman

Equation does not use SI units, so the Shuttleworth modified Penman Equation or Penman Equation (1993) is commonly used.

In 2006, [9] derived a simplified version of Penman's Equation (1948) (Eq. 2). Where temperature is T, solar radiation (R_S), extraterrestrial solar radiation is R_A, relative humidity (RH), wind speed (u), a_u is a wind speed coefficient, and E_{pen} is potential – open water – evapotranspiration in mm/day. Many of these parameters can be obtained or derived from local weather station data [10].

$$E_{PEN} \approx 0.051(1-\alpha)R_S\sqrt{T+9.5} - 2.4\left(\frac{R_S}{R_A}\right)^2 + 0.052(T+20)\left(1-\frac{RH}{100}\right)(a_U - 0.38 + 0.54u). \quad (2)$$

E_{pen} determines the amount of water evaporated from an open area per day assuming no water is added to it.

Despite these simplifications to ET, making direct measurements of all these parameters accurately can be expensive and without accurate classification of the soil type large errors are likely to occur when attempting to determine absolute ET.

3.3 Soil Types and Their Effect on Water Holding Capacity and Mobility

Currently, most measurements place emphasis on Volume Water Content (VWC) and soil tension. The former measures how much water is held in the soil mixture as a percentage. The latter measures the work required of a plant to remove a volume of water from the soil mixture [1]. VWC and tension measurements are typically used to optimize irrigation and crop yield and health, respectively [11].

Both measurements depend on the soil type the plant is rooted in. Soil is comprised of three primary materials; sand, silt, and clay, from largest to smallest particle size. Water holding capacity and mobility are determined the soil mixture. Figure 2 illustrates how changes in soil mixture of sand, silt, and clay change the soil's characteristics. At the center of the triangle is a region called "Loam" that is characterized by a balance between water holding capacity and mobility; an ideal soil mixture for plants to thrive.

Water holding capacity is the force balance between surface tension and gravity applied to water molecules held in soil. A smaller mean particle size has a larger surface area, thus the surface tension exerted by water molecules clinging to particles is greater. A greater surface tension increases the force required by gravity to pull water down through the soil, but also that of plants to retrieve the water. Therefore, plants must exert more energy to extract water from soils with a high clay content and less from soils with more sand [11].

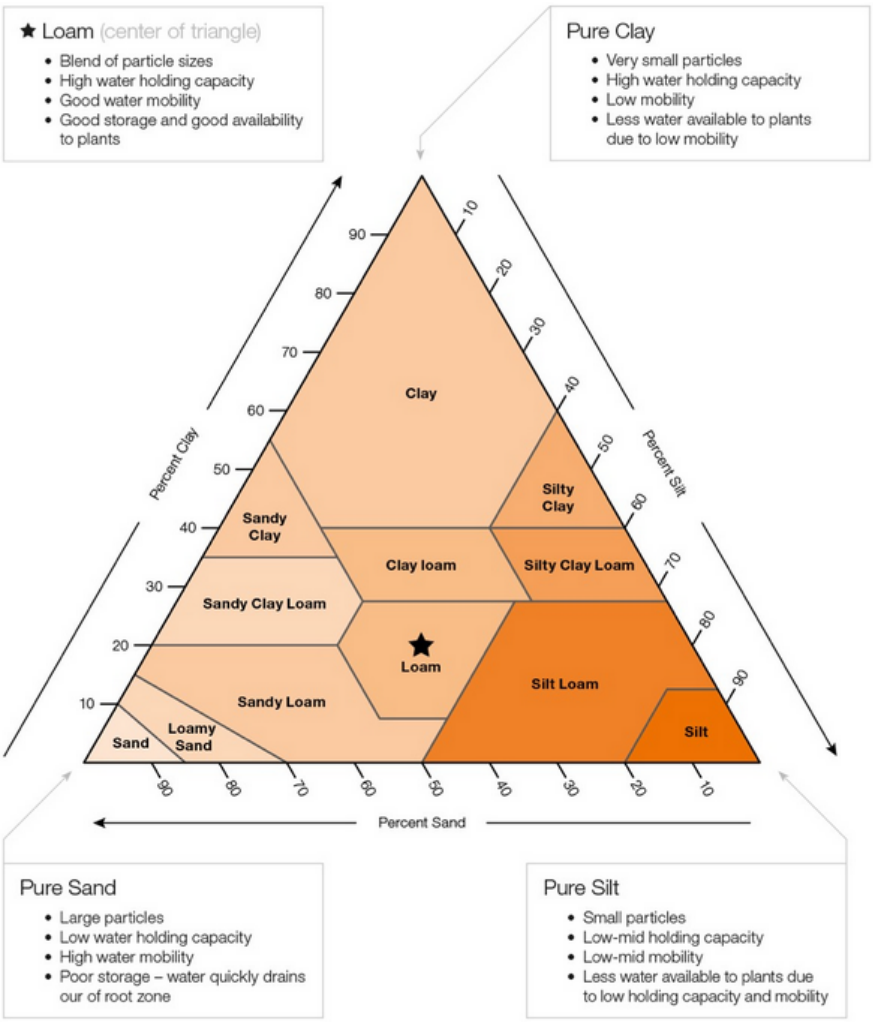


Figure 2. Illustration of the soil texture triangle [11].

3.4 Soil Moisture Measurements

Instead of measuring SWC or tension, soil moisture can be measured as a reference. A variety of technologies and methods can measure soil moisture; resistance or capacitance device, Tensiometers, or the feel method [12]. By measuring soil moisture directly, the soil type, environmental changes, and water uptake are combined into a signal measurement without having to directly measure other parameters.

4 System Design

Instead of measuring the parameters for ET, soil moisture was measured to simplify the characterization of dry, wet, and saturated soil. Additionally, we assumed that most homeowners use all-purpose soil in potted plants. This assumption about the soil type allows for a range of values to be used based on reference soil to determine relative moisture content [13]. Figure 3 shows the overall system architecture for the prototype. The prototype design of the system uses an Arduino Mega and Raspberry Pi 3 connected to a router for internet connectivity

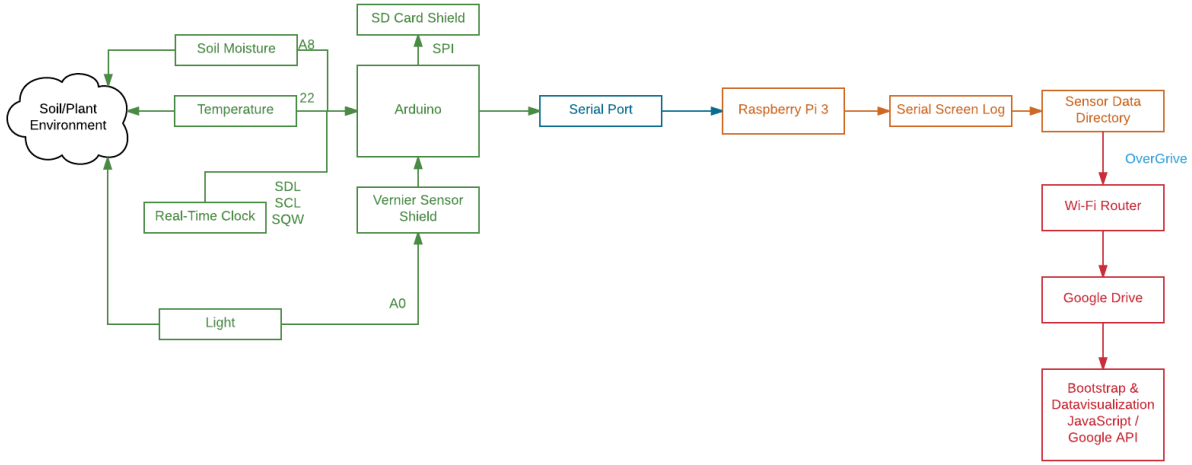


Figure 3. System architecture

4.1 Sensors

The sensors used in this project include a corrosion resistant analog capacitive soil moisture sensor, a digital humidity and temperature sensor (DHT11), and an analog silicon based light sensor from Vernier. Additionally, a real-time clock (DS1307) is connected to the Arduino as well as all the sensors. The Arduino is used because it has an onboard 10-bit ADC and a microcontroller to control the DAQ process. After each sensor reading the measurement is stored as a string and float for storage in the onboard microSD card shield and transmitting the data via serial communication, respectively.

4.1.1 Corrosion Resistant Soil Moisture Sensor

A capacitive soil moisture sensor embedded in a PCB provides corrosion resistance when left in the ground for long periods of time. Other common off-the-shelf soil moisture sensors are based on resistance change measurements. These sensors work well for relative SM measurements as well, however, they are prone to failure by corrosion when left in the ground longer than a week.

To calibrate the capacitive soil moisture sensor, a measurement needs to be taken in air and while submerged in water no further than the warning line. These two measurements are then used as the new dry soil and moist soil boundary values, respectively. With the upper and lower limit defined, the region is divided into three, the highest, middle, and lowest being dry, wet, and saturated, respectively. The manufacturer referenced moisture regions are: dry (430 – 520), wet (430 – 350), and saturated (260 – 350). However, the validity of this data is dependent upon the correct positioning of the SM sensor (Fig. 4).

The specifications for the sensor include supporting a supply voltage at 3.3 to 5.5 VDC and operating at < 1 mA. These power specifications show that capacitance soil moisture sensors are ideal for low power applications.

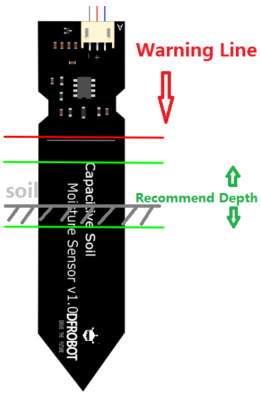
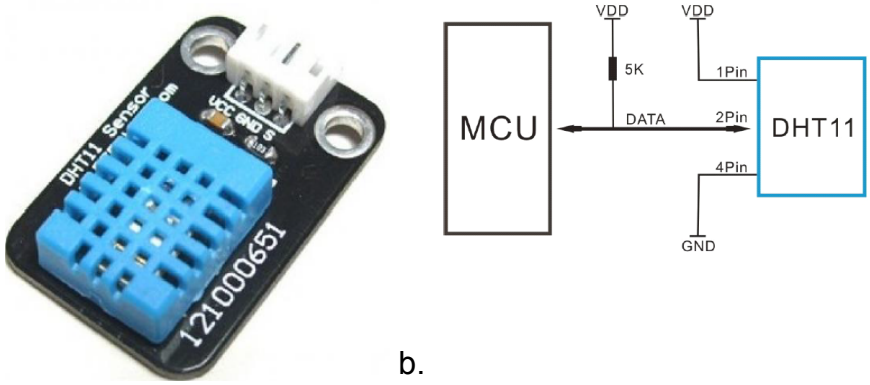


Figure 4. Recommended soil moisture soil insertion depth guide [14].

4.1.2 DHT11

A DHT11 was used to measure relative humidity (RH) and temperature in the local plant environment (Fig. 5). An NTC temperature device and a resistive-type device to measure RH connected to an 8-bit microcontroller are embedded in the DHT11. Calibration coefficients are stored on EEPROM. The accuracy of the DHT11 relative humidity and temperature measurements are +- 5% RH and +- 2 C. Relative humidity also has a range limit of 20 – 90 % RH between 0 – 50 C. Response times for measuring RH range from 6 – 15 seconds [15]. Similarly, the temperature sensor has a response time of 6 – 30 seconds. The resolution for this device is better than the accuracy, as is the case for most sensors. For the DHT11 sensor, RH and temperature resolution is 1 %RH and +- 1 C, respectively.

The power requirements for the DHT11 sensor includes 3 – 5.5 VDC at 0.2 – 1.0 mA average active current draw and 100 - 150 in standby mode. The output from pin 2 on the DHT11 sensor is connected to digital pin 22 on the Arduino Mega.



a. b.

c.

Figure 5. DHT11 temperature and humidity sensor a) image and b) schematic wiring diagram [15].

Temperature and relative humidity can be used directly and indirectly in the Penman Equation or to evaluate correlations in rate changes in soil moisture to environmental parameters.

4.1.3 Solar Irradiance

A Vernier analog Lux sensor is used to measure the ambient lighting conditions the plant is exposed to (Fig. 6). This sensor provides three ranges of lux measurements; 0 - 600 lux, 0 - 6000 lux, and 0 - 150000 lux. With a 10-bit ADC, the resolution for the sensor is 0.3, 8, and 200 lux, respectively.

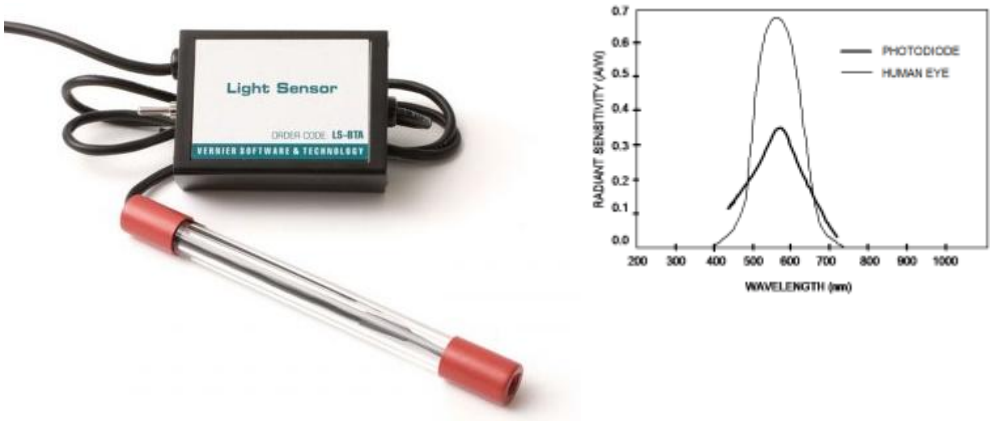


Figure 6. Vernier analog Lux sensor a) image and b) wavelength sensitivity with respect to the human eye [16].

This sensor was selected because it overlaps with the photosynthetically active radiation (PAR) region (Fig. 7). Measuring this parameter provides additional information about the environment the plant is in and correlates environmental phenomena with observed changes in plant health or other sensory data. Correlating to other sensor data provides context awareness, for instance when the temperature increases in the locale of the plant inside a home, is it from the heating system or solar irradiation? By measuring solar irradiance, data could be correlated to sun light being present or not on the plant.

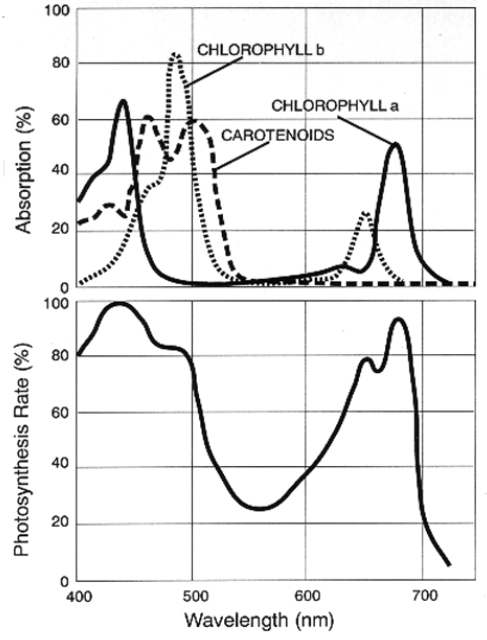


Figure 7. Photosynthetically active radiation absorption with respect to plant organelles and photosynthesis rate for isolated chloroplast [17].

4.2 Real-Time Clock Module

One of the most important components is in any sensor system is time. Knowing the time allows for external events and systems to compare events on a common domain. To measure time, a DS1307 real-time clock (RTC) from Sparkfun was used (Fig. 8). The RTC communicates on pins SDA and SCL using I2C protocol. Digital pin 45 is connected to SQW that provides a 1 Hz square wave to increment time and has a 10 k resistor used as a pull-up resistor across the 5 V supply to enable to the output driver [18].

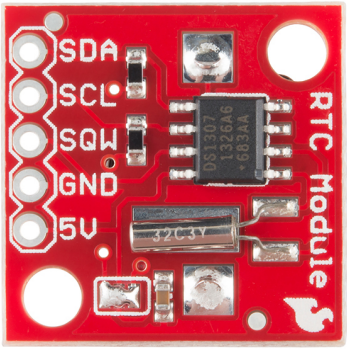
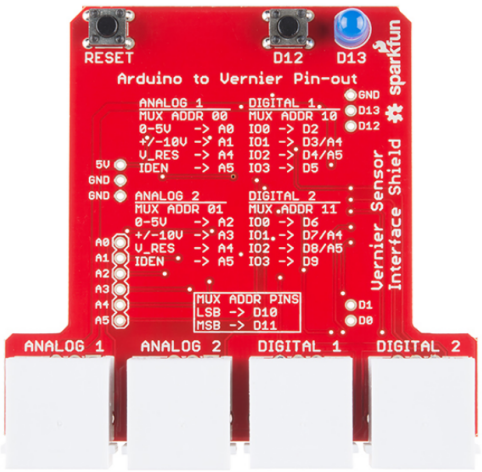


Figure 8. Image of a DS11307 RTC [18].

4.3 Arduino Shields

Two shields were used in the prototype; the Arduino to Vernier shield [19] and an Arduino compatible Ethernet and microSD card shield [20 & 21] (Fig. 9 a & b). Both shields are supplied with 5 V from the Arduino Mega 2560. The analog channels 1 and 2 on the Vernier shield interface with British Telecom Analog (BTA) and corresponds to pins A0 - A5 between where A0 provides the measured voltage output from the connected device. Digital channels 1 and 2, which are not utilized, interface with British Telecom Digital (BTD) and corresponds to pins 2 - 9, A4, and A5. Solar irradiance connected to analog channel 1 was the only sensor interfaced with Vernier shield. The Ethernet and microSD card shield use digital pins 4, 10, and 50 - 53 on the Mega to interface to the serial peripheral interface (SPI) bus.

a. 

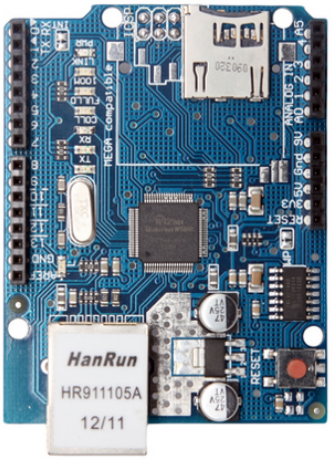
b. 

Figure 9. Images of the a) Arduino to Vernier shield and b) an Arduino compatible Ethernet and microSD card shield.

4. 4 Raspberry Pi 3

The Raspberri Pi 3 (RPi 3) was selected because of the extensive functionality it provides. There are numerous connectivity options to choose from; Wi-Fi, Bluetooth, Ethernet, and serial USB. Additionally, the RPi 3 hosts a 1.2 GHz 64-bit quad core ARM CPU with 1Gb of RAM and a version of Mathematica 10 that provides more than adequate computing power to process data and generate sophisticated visualizations not available on the Arduinos [22]. After receiving data from the serial port, the RPi 3 and supporting software synchronize the data and visualizations on the cloud using an Ethernet cable connected to a NETGEAR modem.

4.5 Hardware Prototype

An Arduino Mega 2560 was selected was for the ease of programming and versatility of the ATmega2560. Additionally, the Arduino Mega has many analog and digital I/O pins desirable for prototyping (Fig. 10). However, the ATmega2560 is limited in its capabilities as a data acquisition and real-time reporting platform. Initially we designed the system to use Ethernet to transmit data between the Arduino Mega and Raspberry Pi. However, we encountered challenges with the modem identifying and authenticating the Ethernet shield's MAC address. Thus, we connected the Mega and Raspberry Pi using a serial USB cable to transfer data. Data was collected once every 15 minutes and would wait for 15 minutes to transmit. Before transmitting data, all measurements were stored on the microSD card and in the serial buffer.

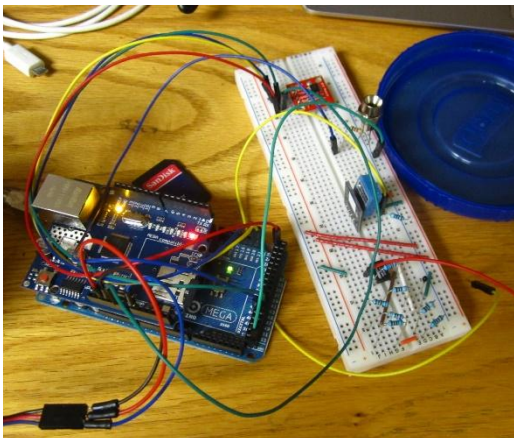


Figure 10. Hardware prototype

4.6 Software Architecture

Figure 11 illustrates the relationship between the different hardware and software platforms used in the prototype.

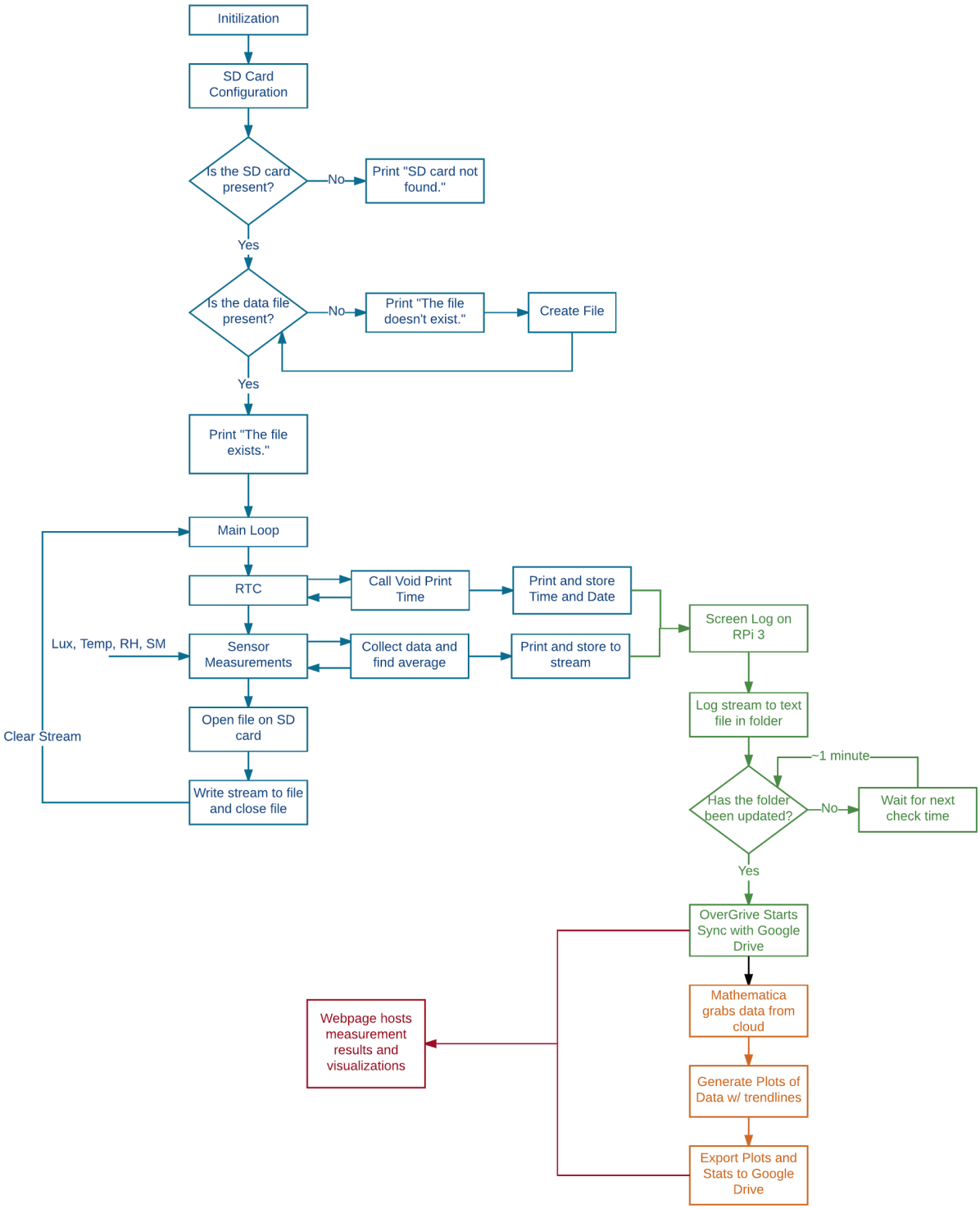


Figure 11. Overall software architecture implemented. The color code represents the platforms and programs used; blue: Arduino and Arduino IDE, green: RPi 3 and Unix, orange: RPi 3 and Mathematica, and red: Internet and JavaScript.

4.6.1 Arduino

An Arduino Mega running Arduino IDE 1.6.12 was used as a data acquisition system to collect sensor data from previously mentioned analog and digital sensors. While initializing the program, the several libraries are called (see Appendix D) and the SD card configuration is started. To complete the SD card configuration the program checks for the presence of the SD card and file to store measurements in. If the SD card is not found the program reports the condition and rechecks until it is found or the system is turned off. After the SD card is detected it checks if the file is present. If the file is absent the program creates the file and checks for its existence. When the file is detected the program starts the main function.

The first process in the main function initiates the RTC function and stores the time and date in the variable and data stream. After determining the time and date, the program commences to collect data. Before collecting data, an independent function calls each sensor. In each sensor function, 50 points are collected and averaged to reduce the environmental noise, particularly for the soil moisture sensor that is sensitive to water and air movement in soil. For the analog sensor values, the average reading represents the average ADC voltage value, which is converted to an actual physical value when returned to the main function and stored in the variable and serial buffer. However, the soil moisture data is not converted because a relative scale is used to classify the soil moisture content of the soil. At the end of the main function, variable values are stored on the SD card.

Throughout the measurement process, the Arduino transmits data through a USB serial connection natively implemented on the Arduino to a RPi 3 listening for the transmission. However, as mentioned before were not able to implement the additional sensors in the final experiments.

4.6.2 Raspberry Pi

While the Arduino collects sensor data a serial connection is made between the Arduino and Raspberry Pi 3. We utilized software for the Raspberry Pi to process the incoming serial data from the Arduino that communicates at 9600 baud rate. The Unix screen command was used to acquire the serial data and place it into a log file. This file served as a backup in case of internet connection failure and as the file OverGrive (described in section 4.6.3) placed on the google cloud server. This solution was relatively simple to implement and provides scalability. For additional sensor nodes, a separate file would be generated from the streamed data using addressed streams.

4.6.3 The Cloud: Google Drive and OverGrive

When the Raspberry Pi is connected to the internet and a program called OverGrive is used to facilitate cloud storage synchronization. This method is required because Linux does not have a native way to connect to Google's cloud storage. A directory is predetermined for OverGrive to synchronize with Google Drive. Synchronization is bidirectional, so any changes on the cloud or directory updates the other. Once the system is initialized data could be appended to the existing file and OverGrive checks for changes automatically and updates the cloud version if a change

is detected. Therefore, Google's API can be used to pull data from google drive to use in subsequent applications, such as data analytic programs or web based tools.

4.6.4 Mathematica

Mathematica 10 is pre-installed on the Raspbian OS and was used to process data on the RPi 3. The program consists of several steps to import and separate the data from the comma delimited format. Once separated a nonlinear model fit function based on a log series was used to determine the observed trend while in transience. However, once the soil moisture content reached an equilibrium, approximately 2 hours after watering, a linear trend could be used to model the change in soil moisture content. After calculating the trendline, the processed data and trend were plotted together and exported to the synchronized folder to upload onto the cloud.

4.6.5 Web Application

Ideally, the user would have access to their soil moisture readings regardless of their geographical location, e.g. while away from their sensor. The previously specified user interface design can be fulfilled through a content delivery system such as mobile phone, webpage, smartwatch, etc. A web application designed for these needs provides the largest scalability for each design implementation choice, thereby making it a powerful design consideration. It also offers expandability as aggregating data from a database does not become limited (nor reliant) to a particular Application Programming Interface (API). This flexibility allowed the display of the data across all viewing platforms (e.g. phone, desktop PC, laptop), regardless of data format or viewing platform. For instance, the collected samples can be stored in either a MYSQL database, or a Comma Separated Values (CSV) file with relatively minimal change.

5 Experimental Methods

Two experiments were performed to check the functionality of the sensors and system. For both experiments the DAQ system (Arduino and breadboards) were placed next to the soil being observed, but away from direct sunlight. While measuring soil moisture, the soil moisture sensor was placed directly in the soil being observed while within the reference zone. Because both experiments were performed in environmentally controlled rooms we assumed the temperature of the room, soil, and water being poured into the soil were in equilibrium. Therefore, the temperature sensor did not need to be placed in the soil.

5.1 Experiment I

In the first experiment used reference soil the Agricultural Institute that had been dried at 80 C for over a week (Fig. 12). The cup contained approximately 250 ml of soil and the soil moisture value in air and dry soil was 560, which was used as the new upper air boundary. With a lower boundary of 260, the new soil moisture content regions were 260 - 360 (saturated), 360 - 460 (wet), and 460 - 560 (dry).

The cup contained about 250 ml of soil and added about 125 ml of water. The soil moisture sensor was inserted into the soil after waiting for the pool of water on top of the soil to soak into the bulk soil.

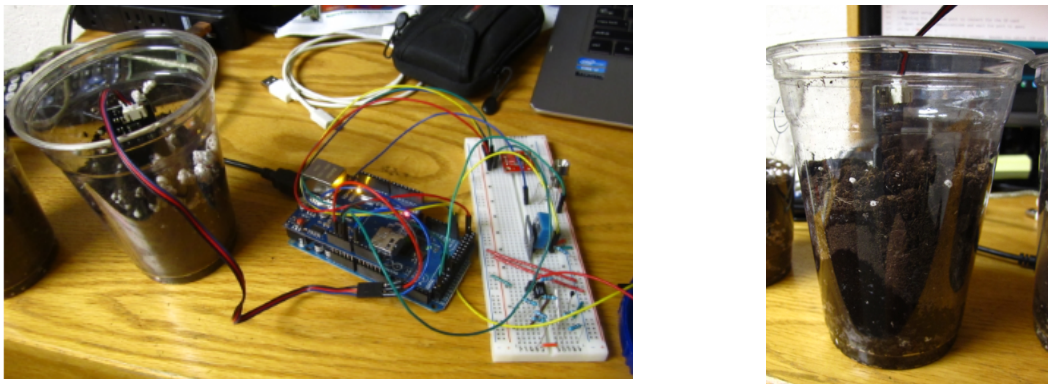


Figure 12. Reference soil in cup being measured immediately after being watered.

5. 2 Experiment II

In the second experiment the soil moisture sensor was inserted into a pot with a plant growing from it (Fig. 13). In this experiment the same amount of water was added to the soil, about 125 ml. Therefore, and difference in the observation would be from the plant and soil, since the local environment was held constant. Additionally, the program was modified to collected 50 samples per measurement instead of 10 compared to experiment one.

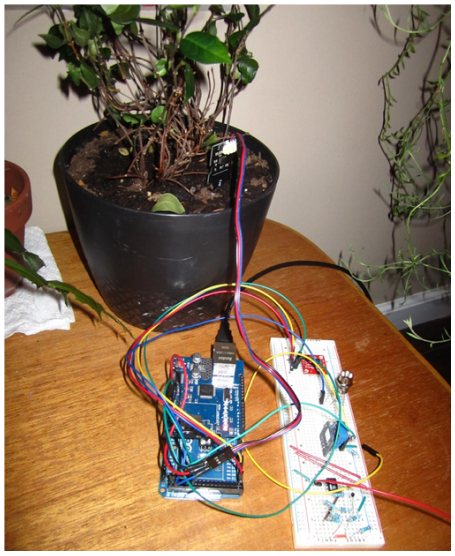


Figure 13. Image of soil moisture sensor collected data from a recently watered potted plant.

6 Results

6.1 Experimental Results

This first experiment collected data every minute for about 20 hours with 10 samples collected for every measurement (Fig. 14).

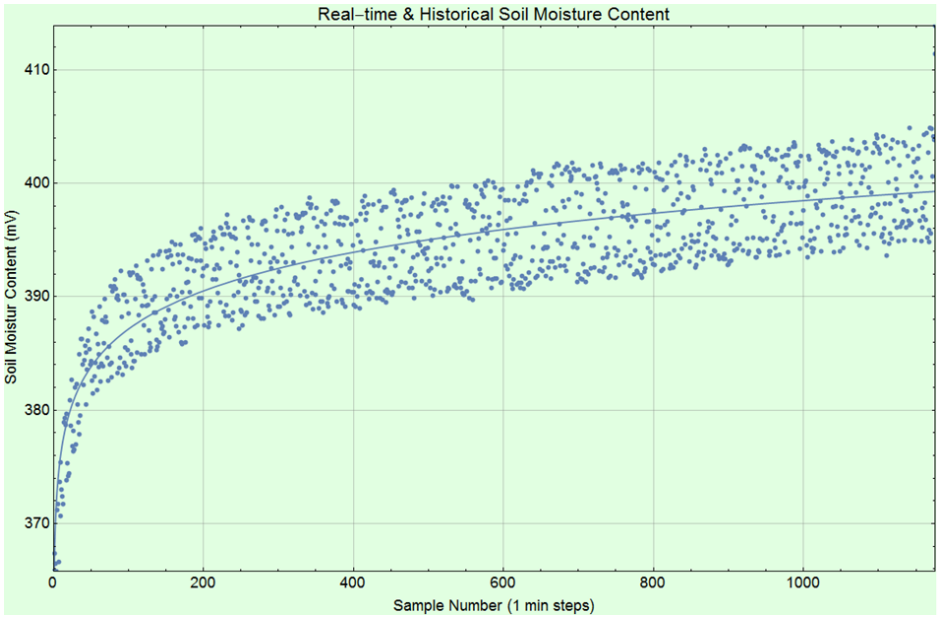


Figure 14. Soil moisture sensor data with log fit (i.e., SM = 68.5 + 297 x^ 0.015) for the first experiment.

The second experiment collected data every minute for about 5 hours with 50 samples collected for every measurement (Fig. 15).

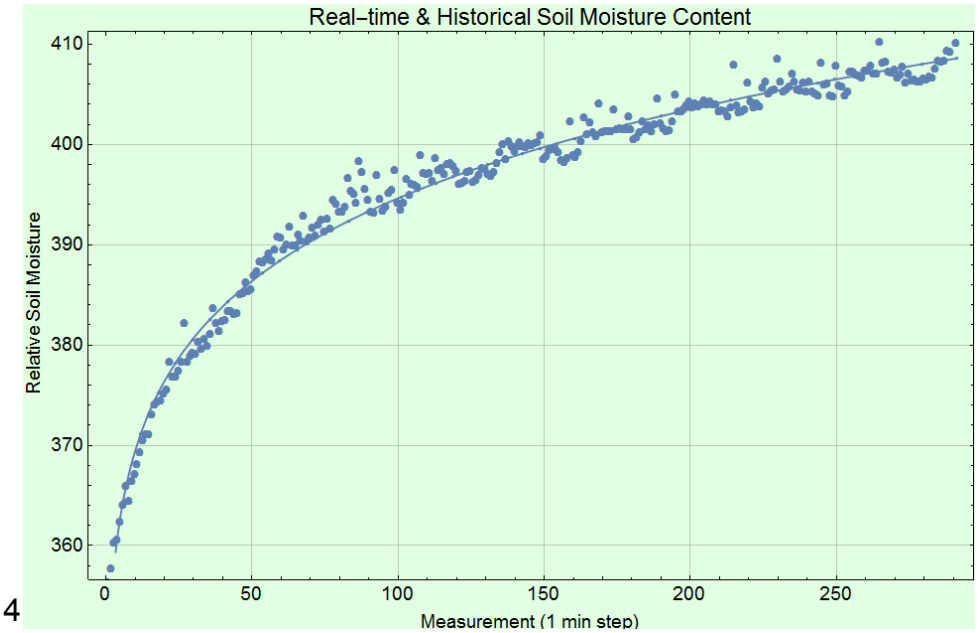


Figure 15. Soil moisture sensor data with log fit (i.e., SM = 277 + 72 x ^0.11) for the second experiment.

6.2 Programming Results

Through a assortment of data acquisitions, the sensor system successfully detected and reported soil moisture readings to a CSV file predetermined interval. Upon

synchronizing the data to the cloud, it was successfully aggregated and qualitatively analyzed. Furthermore, various computer languages were used to automate the analysis process while preventing interruptions to the data acquisition system. Mathematica was used to determine trendlines and provide data visualizations to the local device and network. JavaScript and its derivatives such as the Google Charts API were used to enable user interactivity for the data visualizations.

7 Discussion

7.1 Experimental Results Discussion

The results from the first experiment were extremely noisy throughout the 20 hours of observation. The suspected cause is from the soil and water being mixed together and not sufficiently compacted to remove excessive air pockets from the mixture. The soil moisture sensor, as mentioned on the online wiki, stats a sensitivity to air gaps in soil moving across the face of the sensing element.

A trendline and predication based on the trendline was attempted, but the projected next watering was calculated to be a year from the experiment date. Besides the likelihood of air gaps in the test soil mixture, too short of an experiment is likely the reason the prediction was so extreme.

For the second experiment, a potted plant was selected that had settled soil and one with a living plant. The raw data from the soil moisture sensor showed much less noise, most likely a result of fewer and smaller air pockets and the additionally samples collected and averaged for each measurement. The trend line for both experiments follow a log fit proceeding a recent watering event. However, the coefficients and power value are significantly different and cannot be compared beyond the type of fit.

From the log fit from the data collected in experiment 2, the time until the soil would become dry was 20 hours. Unfortunately, no data was collected during that period to validate the prediction because of attempts to incorporate additional sensors into the system. However, the soil was felt to gauge the dryness of the soil and was similar in condition to when the plant was normally watered.

7.2 Programming Results Discussion

The successful output of sensor results qualifies for a success programming results. Because the delivery system for the content is outside the scope of embedded devices, a simple CSV file generated from the embedded platforms discussed in section 4.6.1-2 is sufficient to meet our programming goals. The final implementation exceeds these expectations; providing an interactive display of the sensor values across a variety of devices including but not limited to: PC, Laptop, and mobile devices (Android or IOS). The content displayed via charting is easily to manipulate via touch or mouse click and reflects the most recent sensor readings currently onboard the Raspberry Pi/Arduino.

7.3 System Limitations

In our experiments, we placed addition sensors on the plant. Resultantly, the embedded system could not handle the increased power requirements whilst maintaining reliable sensor reporting. This reduces the scope of acquisition data from its initial intention, but does not hinder the long-term goal of the project. The embedded

system is not limited by internet connectivity if user facing interactivity is limited to the embedded device itself. However, supposing the user wishes to enable touch interactivity across various devices, internet connectivity is a hard limitation as the design is currently implemented. The device dependent upon a steady power source, as the design does not currently implement any onboard battery power for operation. Additionally, the device is not robust to survive exposure to extreme environmental conditions and would likely not be tolerable to extended outdoor use. These constraints can be summarized with the limitation of “for indoor use only.”

8 Conclusion

Overall, we have developed a system to easily monitor household plants to anticipate future need that can be easily expanded. The current system meets all the goals set out at the beginning, while still planning on how further work can be done. The system accurately shows how soil moisture for the plant changes over time. This provides the user with the ability to anticipate future needs of the plant. It also uses a cloud architecture that can be readily scaled to users’ requirements.

9 Future Work

If we were to continue this project there would be two goals. The add the ability to have multiple plants displayed on the same application. This is a very realistic situation for home use, as most people have more than one plant. We have set the system up so that this would not need to change any major system component. We would just need to setup a different file for each sensor on the Raspberry Pi, which will then have that file in the cloud. Then a way to add additional graphs will need to be made. Each graph should just pull data from one file.

The second goal would be to add additional sensors to the system. This would give the user a more detailed view of plant environment and dynamics. This expansion would most likely require substituting the Arduino or adding processing support to handle the increased processing load. ~~In our experiments we placed more sensors on the plant and the arduino could not handle it.~~ With these expansions, the system would provide a more comprehensive look at the plant health and environmental influences.

References

[1] [http://plen.ku.dk/english/research/env_chem_phys/agrohydrology/

[2] http://plen.ku.dk/english/research/env_chem_phys/agrohydrology/

[3] https://commons.wikimedia.org/wiki/File:Surface_water_cycle.svg

[4] http://iwmi.dhigroup.com/images/used/Watbal_USGS.jpg

[5] http://water.usgs.gov/edu/watercycleevaporation.html

[6] http://water.usgs.gov/edu/watercycletranspiration.html

[7] http://water.usgs.gov/edu/watercycleevapotranspiration.html

[8] http://www.sciencedirect.com.proxy.lib.utk.edu:90/science/article/pii/S00221694060032 6X

[9] http://www.sciencedirect.com.proxy.lib.utk.edu:90/science/article/pii/S00221694060032 6X

[10] Shuttleworth, 1993

[11] https://observant.zendesk.com/hc/en-us/articles/208067926-Monitoring-Soil-Moisture-for-
Optimal-Crop-Growth#types%20of%20soil%20moisture%20sensors

[12] http://pubstorage.sdstate.edu/AgBio_Publications/articles/FS876.pdf

[13] https://www.dfrobot.com/wiki/index.php/Capacitive_Soil_Moisture_Sensor_SKU:SEN01 93

[14] https://www.dfrobot.com/wiki/index.php/Capacitive_Soil_Moisture_Sensor_SKU:SEN01 93

[15] D. Uk, "Temperature Sensor DHT 11 Humidity & Temperature Sensor," *DHT11 Datasheet*, p. 9, 2010.

[16] https://www.vernier.com/manuals/ls-bta/#section6

[17] [http://www.life.uiuc.edu/govindjee/paper/gov.html

[18] https://learn.sparkfun.com/tutorials/real-time-clock-module-hookup-guide

[19] https://learn.sparkfun.com/tutorials/vernier-shield-hookup-guide#vernier-sensor-identification

[20] https://www.arduino.cc/en/Reference/Ethernet

[21] https://www.arduino.cc/en/Reference/SD

[22] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Appendix A - Justification for Project Scope Change

Over the course of this project we have changed the scope severely. We began just adding onto an existing system from the College of Agriculture to developing our own system more focused on monitoring houseplants. This change happened for two reasons, the first, the people who we were working with were hard to get in touch with, and second the second reason is that they would only let us see the data they had collected and not do anything with the sensors. Both of those reasons made that project not work for this class. Once we decided to leave that group we then made the decision to shift from an industrial application to a consumer product. This change meant we needed to change many other things in the design. This included all the hardware and much of the software we intended to use. Instead of using the very expensive sensor provided, we bought much more inexpensive sensors that make more sense for a consumer product. This also changed how we managed the data. In the previous setup there was already a server in place, but then we needed to make our own. We chose to use a cloud based system for both ease of use and expandability. These changes to the scope of work of which we wanted to do allowed us to completely build our own system and have control over every part.

Appendix B - Program Code & Additional Visualizations

Arduino

Final version of Arduino code hosted on GitHub.
DFRobotSM_V2

Raspberry Pi

Final version of the code used on the Raspberry Pi hosted on bitbucket.
https://tnpaul9@bitbucket.org/tnpaul9/embedded-systems.git

Mathematica

Final version of the Mathematica code on GItHub.
DataGrab&Fit

Bootstrap/Javascript/Google API