Evergreen

SENIOR PROJECT

Ashley Grover | CPE 462 | Winter 2017
Advisor: Dr. DeBruhl

TABLE OF CONTENTS

INTRODUCTION

PROJECT OVERVIEW

The goal of this senior project is to develop an Android application that interacts with hardware to provide a plant health monitoring system. This project allows the user to keep a plant alive and healthy. This is accomplished by notifying the user when plant soil moisture levels are excessive or insufficient, allowing the user to view overall plant data over a week's time and daily averages, and keeping the user informed through strong visual aspects. The inspiration for this project transpired from my strong interest in Android development and embedded systems. The problem that motivated me to choose the topic of plant health originated from my mom, who has had several house plants that always tend to dry up and look unhealthy. This project is intended on benefiting my mom and others, as well, who have the same issue and want to keep their plants alive and healthy. This will be accomplished by monitoring plants and notifying the user when their health is lacking. This project will use several hardware components, which consist of a microcontroller, a Bluetooth module, and a soil moisture sensor, and an Android application.

CLIENT

There is no specific client for this project. However, the main point of contact for specific requirements and functionality needed is my mom, Melanie Grover. She is the individual who inspired the idea of this project and various aspects of useful information regarding what a user would want in an application like this is derived from her. She has planted various house plants and found it difficult to keep up with them, resulting in

dying plants due to lack of water. With a system that notifies a user of individual plant health, plants will be given adequate water and have a longer life span.

PROJECT GOALS AND OBJECTIVES

I define project goals to summarize what I plan to achieve with my plant health monitoring system project, *Evergreen*. They include:

- Develop a system that accurately reads soil moisture data from a sensor and sends this data successfully to an Android application.
- Develop a system that successfully notifies a user when plant health is declining.
- Establish a clean and informal user interface that makes use of material design.

I define project objectives or tasks that need to be accomplished in order to reach my project goals. They include:

- Design hardware that detects and communicates soil moisture levels.
- Research and implement necessary methods for Android's Bluetooth API to allow an interface between the hardware and software components.
- Configure a database to correctly store data based on username, unique plant name, plant type information, and plant statistics.
- Design a user interface that effectively presents plant data and statistics, keeping in mind the Android practice of material design.
- Ensure all requirements can be met, make updates, and inform "client" of all modifications that may occur to ensure successful implementation.
- Ensure successful functionality of the system with software testing.

PROJECT OUTCOMES AND DELIVERABLES

At the end of this project, I intend to have a fully-completed plant health monitoring system that meets all expected functionality, as discussed in the section titled *Engineering Specification*. Its implementation will allow the user to use the application on a daily basis to check plant health at any time of the day.

Success in this project will allow my client and others with an interest in this topic to have a fully functional application that allows them to track how their plants are doing with regards to water levels. It will allow individuals to enjoy planting as a hobby or activity and will serve as a benefit for those who have trouble remembering to water their plants.

BACKGROUND

After researching applications similar to the one my project implements, I found various systems that are relatable. I discuss related projects in two categories—hardware and software.

The hardware system consists of a Bluetooth module that sends data retrieved from a soil moisture sensor to the software system. Similar applications to this are found in various applications. For example, someone can send photos or documents from one smartphone to another via Bluetooth. This technique of data transmission and retrieval can be mimicked in my system, yet with several differences. With further research related more specifically to my project, it was found that there are various applications already in existence that retrieve data from a Bluetooth module, the HC-06, which is the Bluetooth module incorporated into my project (ForceTronics). These existing applications

incorporate an Android app into the project, which consists of a terminal-style communication system that can send data to the HC-06, with the HC-06 having the ability to send data back to the Android app.

The hardware system also requires a battery to keep the Arduino Uno powered. While researching this, I came across several options to power the system. One included a standard 9V battery. Pros of this option include a small size and simple setup. Cons, however, include small mAh, resulting in the battery needing replace every few hours. Other options I found relate to the 9V battery option with the same pros and cons, therefore not making them ideal battery sources. An ideal option in which I came across is a portable solar charger battery. The pros of this battery are it has 10000 mAh per charge, can be charged via solar, and is rain-resistant. The only con is its larger size than a typical battery, yet this does not outweigh the significant pros.

The software system of this project consists of an Android application that shows the moisture levels of the plants' soil, both currently and over time. Its intention is to notify the user when the data retrieved is too low or too high, depending on the plant type. While researching for applications similar to my project that already exist, I came across several Android apps in the Android Play Store that help users manage their gardens with alarms, notifications (Plant Nanny), and special designs (Flower Care), yet none in the store included a hardware counterpart. After researching projects others have done similar to mine, I found some comparable to what my project intends to accomplish, yet not exact.

Plant Nanny. *Fourdesire*. Four Desire, 2013. Web. 12 Feb. 2017. <http://plantnannyapp.com>.

Flower Care. *Google Play*. Beijing HHCC Plant Technology Co., 1 Mar. 2017. Web. 12 Mar. 2017. <https://play.google.com/store/apps/details?id=com.huahuacaocao.flowercare&hl=en>

ENGINEERING SPECIFICATION

The purpose of this section is to present a detailed description of Evergreen. Its intention is to explain the features of the system, specific use cases to be implemented, how the system works and components are integrated together, and how the system reacts to external stimuli (i.e. the user and hardware).

SCOPE

This project consists of two systems that work together to create a productive Android application that informs a user of plant health. The first system incorporates hardware – the ATmega328p microcontroller, a sensor, and a Bluetooth component – to create a technique of gathering plant data and transmitting it to an Android application. The second system, being the Android application, takes the transmitted data (via Bluetooth), stores it in a database for future reference, analyzes it, and presents it to the user in a statistical manner.

This project is intended to incorporate software and hardware together and let the two components interact with one another. Keeping track of plants' hydration is important to many people and having an Android application that records all data occurring within this system can be quite beneficial. This project is intended to be informative and convenient for the user, while also encouraging others to grow organically or simply just take care of a common plant(s).

SYSTEM ENVIRONMENT

The overall system environment of this project consists of both hardware and software components. The hardware encapsulates a microcontroller, a soil moisture sensor, and a

Bluetooth module. Bluetooth is the core factor that connects the two systems together, wirelessly. The Android application consists of the plants statistics retrieved from the transmission of data between the two systems; the data is stored in a database and kept track of over time, while being visually accessible to the user in the form of UI. The user interacts with the system environment by adding, deleting, updating, and selecting options for the plants under inspection. Figure 1 shows a visual of the system environment diagram.

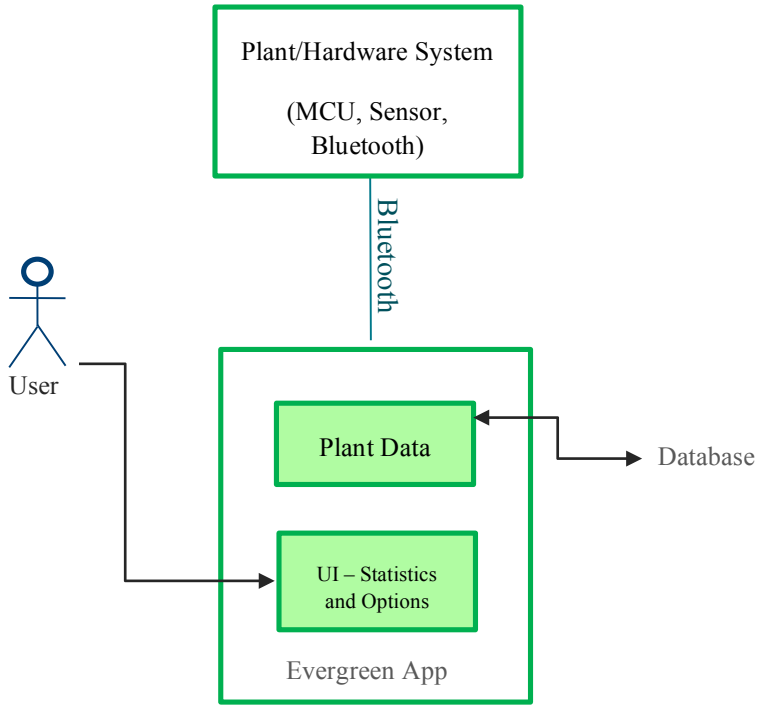


Figure 1: System Environment Diagram

FUNCTIONAL REQUIREMENTS SPECIFICATION

This section outlines hardware requirements and individual use cases between the User, the hardware, and the software.

Requirements involved in the hardware and plant system, while disregarding the Android application, are included in the list below.

- The hardware system shall detect soil moisture levels with the use of a sensor.
    - Other sensors, including temperature and humidity, may be included.
- The ATmega328p shall be implemented to read the soil moisture levels.
    - Coded in embedded C.
- The data shall be transmitted to the Android application via Bluetooth.
- The data in the app shall be updated when the user is in close proximity.
    - Depending on Bluetooth range.
- The battery life shall last a reasonable amount of time – non-frequent replacement.
- The hardware system shall accommodate at least one plant.

Requirements involved in the Android application, while disregarding the hardware system, are included in the use cases below.

Use Case – Add a Plant

Brief Description:
The user adds a plant to keep track of.

Initial Step-by-Step Description:
This use case is one of the first actions available to the User after startup of the mobile app.

1. The User selects to *Add Plant*.
2. The app presents a UI layout that will be used for further use case details.
   2a. Select Bluetooth.
   2b. Search for plant.

Use Case – Select Bluetooth

Brief Description:
The User selects the Bluetooth device in which he or she wants to correspond to the newly added plant.

Initial Step-by-Step Description:
This use case can take action after the User selects to add a new plant.

1. From start layout, the user clicks on the floating action button that adds a new plant.
2. The user then selects the correct Bluetooth device from a list of paired devices in which the phone is already paired to.

Use Case – Search for Plant Type

Brief Description:
The user searches for a plant type, depending on what he or she intends on growing.

Initial Step-by-Step Description:
This use case can be initiated after the User adds a new plant to keep track of.

1. The User selects the search button to pick which plant he or she is interested in.
2. The User types his or her search keyword.
3. The system displays the various choices of plants related to the search keyword to the User.
4. The User selects which plant fits his or her specification.
5. The system indicates the chosen plant and saves it into correct field.

Use Case – Upload Photo of Plant

Brief Description:
The User has the option to capture a personal photo of the plant under inspection.

Initial Step-by-Step Description:
This use case can be initiated only after a new plant is added.

1. The User taps on the “update photo” floating action button.
2. The system presents the option to change current image.
3. The User can select to change the current image by clicking “OK.”
4. If the User selects to change the current image, access to the phone’s camera is required.
5. After accepting camera access, the User’s camera application will display.

6. The User can then take a photo of the plant that he or she wishes to display as the updated photo.
7. After capturing, the photo is positioned in the correct photo area.

Use Case – View Plant Statistics

Brief Description:
The User can view daily and weekly average data retrieved from the plant specified by current plant name.

Initial Step-by-Step Description:
This use case is used by the User once a plant has been added to inspection and data has been transmitted via Bluetooth.

1. The User can view plant statistics by scrolling down on the initial layout displayed.
2. All details, including average soil moisture and temperature levels, daily levels, and user productivity, are displayed.

Use Case – Delete a Plant

Brief Description:
The User has the option to delete a plant no longer in use or if a mistake occurred while adding a plant.

Initial Step-by-Step Description:
This use case can take action after a plant has been added. If the User mistakenly added a plant or has a plant he or she no longer wants to track data for, then this use case is acknowledged.

3. From start layout, the user clicks on the plant to be deleted.
4. The user clicks on the options menu in upper right corner.
5. The user clicks on the option “delete plant.”
6. A pop-up option appears to delete the plant or not.
7. The user can then confirm the plant deletion.

Use Case – Store Plant Info in Database

Brief Description:
All plant data is sent to an external database for storage. In this use-case our “Actor” is the database and interacts with the Android application to retrieve necessary data.

Initial Step-by-Step Description:

1. Data is retrieved and sent via Bluetooth to the Android application.
2. The data is stored in an external database, Android's Firebase, in appropriate categories for easy access.
3. This data will be easily updated and accessible.

Other use cases include login authentication, registering, deleting account, signing out, and forgetting password - send reset instructions. These cases are related to Firebase and the server side of the application, in which Android mostly takes care of.

This completes the software requirements specification for the project.

DESIGN DEVELOPMENT

This section shows the complete view of the entire system, breaking it down into smaller parts that can be understood easily when necessary. This will build off the current project description in order to represent a suitable model for coding and implementing. The design depicted in this document briefly describes all platforms, systems, services, and processes that it depends on and includes important changes that need to occur, if any.

MAIN DESIGN FEATURES

The main design features include three major parts. These are the architecture, the technology, and the user interface. These features are described below, where diagrams are used represent their functionality, if necessary.

APPLICATION ARCHITECTURE

The high level overview of the application architecture is shown below. This diagram shows the relationship between the hardware system's Bluetooth with the Android app

and the Android app's relationship with Firebase. Evergreen uses several Android APIs, with three of the main ones being shown in the architecture diagram.

Application Architecture Overview Diagram

TECHNOLOGY ARCHITECTURE

This section describes the Android application architecture, the data access layer, and the tools used to develop this project.

Android Application Architecture

The front end of this project uses an Android application user interface, which will be discussed in more detail under the User Interface section. Functionality regarding use cases is also described in the Functional Requirements Specification section.

Data Access Layer

The database used for this project will be accessed by only the Android application, itself. No user will be given permission to access this database, unless requested by the

developer. In the case of external storage usage on the Android mobile device, the user must grant the application the right to use this storage.

Tools Used

The tools used to implement this project include:

1. Arduino IDE, used to implement code in the ATmega328p microcontroller.
2. Atmel Studio, as an alternative to Arduino Studio.
3. Android Studio, a Java based IDE for Android development, used to compile and build the Android application.
4. All hardware found in Bill of Materials section of this report.
5. Android Firebase database, used for storing plant data and statistics for the user's plants and backend development.
6. USDA Plants Database.
7. Microsoft Word, for all other related documentation.

USER INTERFACE

The user interface makes use of material design and UX. It consists of design components that relate directly to the theme of this project – botany. It is implemented with various layouts needed to make the user's experience easy and pleasant. It is intended to have a simple overall layout to avoid confusion and frustration, yet complex in matters of material design to create a positive and strong theme. Animation is used in appropriate areas to make an emphasis on certain sections of the application that

represent importance. This allows for an overall better user experience – an application that is informative but also interesting to use.

DATABASE DESIGN

Saving individual users' plant information requires the use of a database. Evergreen uses Android's Firebase database. Firebase is a realtime database that allows developers to build a better app by saving crucial development time. It features cloud messaging, authentication, cloud storage, cloud functions, hosting, a test lab, and crash reporting. I chose to use Firebase for these reasons and in order to create the most efficient Android application possible.

Evergreen's Firebase consists of a user entity, with every user having several plant entities. Each plant that is added to the database has several different data values added as each plant updates. These data keys include the plant's common name, scientific name, symbol, family, current moisture level, the average daily moisture level values for the past week, and other statistical readings of plant health. All data stored in the database is sent and received using Firebase's cloud functions feature, while each new user is created in the database using Firebase's authentication feature.

ERROR HANDLING

If any error is encountered, this is something that should be taken care of (caught) in the code. By the end of the project, all testing will be completed, so this error handling should not be an issue. If, however, this is an issue, an error message shall be stated in a pop up box in the Android application to inform the user of it's occurrence.

MAINTAINABILITY

The main prospect of maintainability includes replacing the battery on the hardware system, when applicable. This also may include replacement of the sensor (if residue accumulates on it). Besides this, very little maintenance should be required for this project. The Android application will function without any maintenance, yet updates may want to be made in the future.

FINAL DESIGN CONCEPT

BACKGROUND

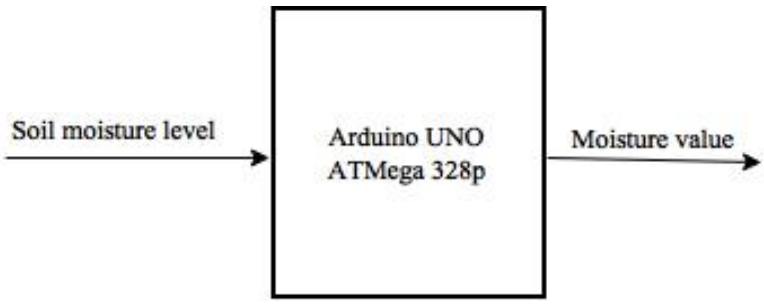
For the final design of this project, I made a point to design around the requirements and use cases present. I sought to develop a design that users could easily interact with and have a positive user experience with. As stated previously in this report, my goals were to develop a system that accurately reads soil moisture data from a sensor and sends this data successfully to an Android application, to create a system that successfully notifies a user when plant health is declining, and to establish a clean and informal user interface that makes use of material design. These goals are achieved to my best ability, with significant components of the project being described in more detail in the sections below.

HARDWARE

There are three main hardware components in this project:

- Arduino UNO, consisting of the ATmega328p microcontroller
- High sensitivity moisture sensor
- HC-06 Bluetooth Module, communicating in slave mode

The high level block diagram of hardware is shown in the diagram below.



Highest Level Block Diagram

The high level block diagram consists of one input, *soil moisture level*, which is retrieved from the soil moisture sensor. The soil moisture sensor uses capacitance to measure dielectric permittivity of the surrounding soil, with the dielectric permittivity being a function of the water content in the soil. The sensor creates a voltage proportional to this dielectric permittivity, which is essentially the water content of the soil. The ATmega328p then takes this retrieved data and sends it to the Android application via Bluetooth (the output of the diagram). The data is sent approximately every twenty seconds to the application to ensure a consistent update of the plant, yet not too excessive (i.e. sending continuously).

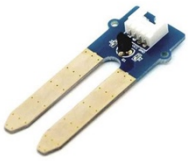
The Arduino Uno is a microcontroller based on the ATmega328p and was chosen for this project due to its robustness and my strong familiarity with it. Its ability to allow me to build the circuit easily made the hardware of this project simpler and successful.

The HC-06 Bluetooth module (shown to the right) used in the project supports slave mode, modified by AT commander. I decided to implement this specific Bluetooth module into the project due to its small size, its compatibility with the Arduino Uno, and

its UART interface with baud rate setup function, which I am familiar with. Incorporating Bluetooth into the project was essential to allow for wireless data transfer. This allows the user to be a farther distance away from the plant under inspection, yet within Bluetooth range, which is a great advantage for the project. If the system was hardwired, it would be a nuisance for the user and the user experience would disintegrate.

HC-06 Bluetooth Module

The soil moisture sensor (shown to the left) used in this project has a high sensitivity measurement and was tested to ensure reliable data readings across multiple different types of soil. The readings of the sensor are based on values between 0 and 1024, i.e. 0V to 5V. As stated earlier, the sensor creates a voltage proportional to the dielectric permittivity, which is essentially the water content of the soil. The input voltage, V_{in}, of the Arduino is 5V, which explains why the sensor values range between 0 and 1024. The higher the value read by the sensor, the more water contained in the soil, and vice versa.


Soil Moisture Sensor

SOFTWARE

The software for this project is written in Java and developed in Android Studio. Its development process follows an iterative approach, rather than waterfall. Continuous integration and testing of the application, as well as having something "runnable" produced at every iteration allowed for a more productive process.

Databases

The searchable plants are stored in an Android SQLite database, which is stored in the application itself, rather than having to be accessed via internet. This type of database is used for this purpose due to its reliability, accessibility, and reduced complexity; SQLite only loads the data it needs, rather than reading the entire file and holding a complete parse in memory. Although this database choice seemed like the most appropriate choice before implementation, I did come across one problem after testing my project. The unfortunate problem in existence occurs during the user's first time use of adding a plant. Evergreen creates the SQLite database by uploading an already existing file that contains all the different plant types and information into the database. This creation occurs the first time a user decides to add a new plant, and it takes approximately eight minutes to create. Due to lack of time, I have not fixed this problem but hope to in the near future to allow for a more positive user experience.

All plant data is stored in Android's Firebase database. This database was the most convenient to use and worked successfully in the project. It stores all necessary information needed to show plant information and statistics, as previously mentioned in this report. Refer to the section *Database Design*.

Application Program Interfaces

Evergreen uses various Android APIs. These include both Bluetooth and Camera. Bluetooth adapters and sockets are used to create and establish a connection with each necessary Bluetooth device in which the plant corresponds to. Initially, the Bluetooth API was rather tricky to implement, yet with more research and implementation it worked successfully. The Camera API allows the user to take a photo of the plant in

view. The user can update each plants photo, where the Bitmap of the photo is stored in Firebase.

User Interface

The user interface makes its best use of material design with appropriate colors, drawables, and animation. The theme of the application follows the main focus of the application—plant health and botany. The color scheme also relates more directly to the female population, due to the statistical fact that women tend to do more gardening rather than men. The layout design is written in xml, while the functionality of the layout, such as scrolling lists, graph and wheel animation, and object listeners, is written in Java.

SYSTEM INTEGRATION AND TESTING

TESTING

Testing this project falls into two categories: manual testing and software testing. Manual testing requires manually testing the software for defects, with a tester using most of all features of the application to ensure correct behavior and acting as an end user. Software testing involves writing unit tests where individual units of software are tested, validating that each component of the software performs successfully.

Manual Testing

Testing the software for defects manually includes ensuring each use case has correct behavior. Manual testing occurred for the following use cases:

- Adding a plant
- Selecting Bluetooth

- Searching for plant type
- Upload photo of plant
- View plant statistics
- Delete a plant
- All authentication/database related functionality

Extensive manual testing for each of these took place, where code updates were made when failure occurred. Currently, the only failure that occurs seldom is uploading photo of plant. Uploading a photo of the plant sometimes result in an OutOfMemory exception due to the large file size, in which code is still being updated to fix. The proposed solution is to convert the Bitmap to a smaller size. All other use cases are successful with no exceptions thrown. The use case of storing a plant in the database was tested by viewing the data updated in the database in realtime, in which all values were stored successfully.

Software Testing

Android testing for this project is based on JUnit and tests are run on the local development machine. Building local unit tests are ideal due to its efficiency in avoiding overhead of loading the target app and unit test code onto the physical device every time the test is run. The execution time for running the tests is reduced, as well. Setting up the testing environment including specifying the JUnit and Mockito libraries as dependencies in the build.gradle file of the app. Test methods were implemented to verify a single functionality in the component being tested. To perform validation checks, junit.Assert methods were used. The procedure for running local unit tests was

trivial in that right-clicking a test and clicking *Run* ran a single test and running all tests in a directory simply required selecting *Run tests*.

Software testing was more difficult than manual testing due to the hardware component integrated into this project. Hence, manual testing became a simpler approach to testing the project.

SATISFYING PROJECT REQUIREMENTS

Software requirements are fulfilled by successfully passing the use cases as stated in the section above, *Testing*.

Hardware requirements are satisfied by ensuring all statements below were met:

- The hardware system detects soil moisture levels with the use of a sensor.
    - Pass; moisture levels are accurately read ranging from values of 0 to 1024 (0V – 5V)
- The ATmega328p shall be implemented to read the soil moisture levels.
    - Pass; ATmega328p adequately reads the soil moisture level.
- The data shall be transmitted to the Android application via Bluetooth.
    - Pass; data is successfully transmitted.
- The data in the app shall be updated when the user is in close proximity.
    - Both pass and fail; passes when connection is established and application verifies this; fails when the application doesn't recognize the connection is there; this is a known error with the HC-06 and is due to it's nature of slave mode only.

- The battery life shall last a reasonable amount of time – non-frequent replacement.
    - Pass; the battery source is rechargeable, both solar and electrically powered.
- The hardware system shall accommodate at least one plant.
    - Pass; one plant is tested and all functionality works.

TIMELINE

Below is the timeline followed for the course of this project.

Quarter	Week	Description
1 – Fall 2016	1	Background Research
1 – Fall 2016	2	Background Research/Requirements Document
1 – Fall 2016	3	Requirements (use-cases) Document
1 – Fall 2016	4	Architecture/High-Level Design and Vertical Prototype
1 – Fall 2016	5	Hardware setup
1 – Fall 2016	6	Hardware setup
1 – Fall 2016	7	C code implementation for mcu 1
1 – Fall 2016	8	C code implementation for mcu 1 Complete
1 – Fall 2016	9	C code implementation for mcu 2
1 – Fall 2016	10	C code implementation for mcu 2 Complete
2 – Winter 2017	1	Android dev Iteration 1
2 – Winter 2017	2	Android Iteration 1 Complete (design/code/test/deploy)
2 – Winter 2017	3	Android dev Iteration 2
2 – Winter 2017	4	Android Iteration 2 Complete (design/code/test/deploy)
2 – Winter 2017	5	Android dev Iteration 3
2 – Winter 2017	6	Android Iteration 3 Complete (design/code/test/deploy)
2 – Winter 2017	7	Android dev Iteration 4
2 – Winter 2017	8	Android Iteration 4 Complete (design/code/test/deploy)
2 – Winter 2017	9	Final Report Write-up/Final Report complete
2 – Winter 2017	10	Submission of Final Report to Library/Department

APPENDIX

BILL OF MATERIALS

Listed in the table below are the materials and corresponding costs needed to develop this project.

Material/ Product	Price
Arduino Uno	$9.95 (≈)
High sensitivity moisture sensor	$5.98
HC-06 Bluetooth module	$7.29
Jumper wires M/M	$7.59
Android Device	$59.99

Bill of Materials

The total cost for this project is $90.80. Disregarding the Android device, the total cost is approximately $30.81. With additional plants added to the system, additional Bluetooth devices and moisture sensors are needed, thereby increasing the total cost.