Portable Watering Device

Chris Havekost, Joan Henriquez, Peter Nachtigal, Ronak Patel

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida

***Abstract* – The Portable Watering Device is a self-sustained system composed of a solar panel, a microcontroller with a few sensors behind it, a water tank and a pump. The solar panel is used to charge a battery on the device which we use to power the system. The microcontroller behind the solar panel is connected to a few sensors (temperature, humidity, pressure, etc.) which are used to predict when the best time to water the plants will be throughout the day. Users will be able to see and control all this information from a smartphone application. The three main requirements we are after with this device are efficiency, portability and connectivity.**

I. Introduction

Planting is a very popular hobby, many people like to have plants at their residence for growing food, landscaping, or making their home look beautiful. The people who own a house have easy access to all the resources required to maintain the plants, such as a water sprinkler system. However, the people who rent an apartment tend to grow plants on a balcony and have no access to an automatic water sprinkler system that waters the plants, and due to rental agreement restrictions, the residents are sometimes not allowed to plant outside. House owners could set the timer on their sprinkler system as desired, but the apartment owners don't have this luxury and would have to water the plants themselves.

One of our teammates grew mint plants in his apartment balcony and they were very healthy plants until summer arrived. For a busy college student, he or she can water their plants at most once/twice a day due to school or work constraints. When it is too hot, watering plants once a day isn't enough for the plants to survive. This is a problem we brainstormed to find a solution for, and we thought what if there was something that could water the plants for us when we are out at work or school. We had this idea of a solar powered and portable water sprinkler system that is smart enough to understand the weather conditions and waters the plants independently. We thought portability was important because people who rent are more likely to move to another apartment frequently and it will be best for them if they were able to move the system with them easily rather than getting a new one. So, the difficulty of growing plants in an apartment balcony motivated us to build a solar powered and portable water sprinkler system.

Some of the main high-level requirements we set for our device are efficiency, portability and connectivity. The system needs to be efficient enough to provide high performance, while maintaining low power consumption. It needs to be small in size and lightweight to help with portability since apartment renters tend to move more often. Lastly, it needs to show sensor data and let the user operate the system from a smartphone application, helping them connect with the device.

II. System Components

When starting out, we had to set some expectations for the project. Any water pump we selected needed to be low voltage and draw a low amount of current to save battery power while simultaneously providing enough pressure to propel a stream of water at least five feet. We had to find a solar panel that could output enough energy to charge the device's battery and be able to power the device on its own. We set expectations higher when we asked ourselves to find a microcontroller capable of being able to handle all sorts of input from sensors, accept power from the battery and solar panel, drive the water pump, and still be small enough to fit into a portable container. We also wanted to design a device that would be easy for anyone to assemble and be user friendly. With these expectations in mind, we set out to find the right components for our device while making the least amount of compromises.

For our microcontroller, the Atmega328 is the one that stood out among the rest because of its performance and price. It features a clock rate of 20 MHz, a low-power mode of 75 microAmps/MHz and only costs $2.18.

For our solar panel and battery combo, of all the combinations we researched during Senior Design 1, we decided to use the 9W solar panel and the V44 battery. This is because the 6V solar panel will generate enough volts to power all the components of the sprinkler system such as the PCB, microcontroller unit, water pump, and all the sensors. And the V44 battery will provide enough USB outputs and capacity of 12,000 mAh for a long-lasting charge and to power the system in case if the weather conditions are not ideal for the system to generate power via the solar panel. The price for the solar panel and battery combo is $159, making it our most expensive component.

For the Wi-Fi Module, we have decided to go with the ESP8266 since it has a lower cost ($6.95), takes up less space and still maintains all the same functionality from the Arduino Wi-Fi Shield, which is one of the best modules available for Arduino embedded projects.

For the water pump, we saw that with a greater range of operation voltage and a difference of only $3.00 compared

to the Lightobject it was the obvious choice for us to use the 3M Water Circulation Micro Brushless Water Pump to drive our plant showering aspect of the design. The price of this pump is $12.

For the water level sensor, originally, we decided to go with a small ultrasonic sensor due to its cost, size, and ease of implementation. The ultrasonic sensor we had decided to go with was the ELEC Freaks Ultrasonic Sensor HC-SR04 which cost $3.95. However, after finding out about water level strip sensors, we thought it may be easier for our end users to set up so we decided to give that a try.

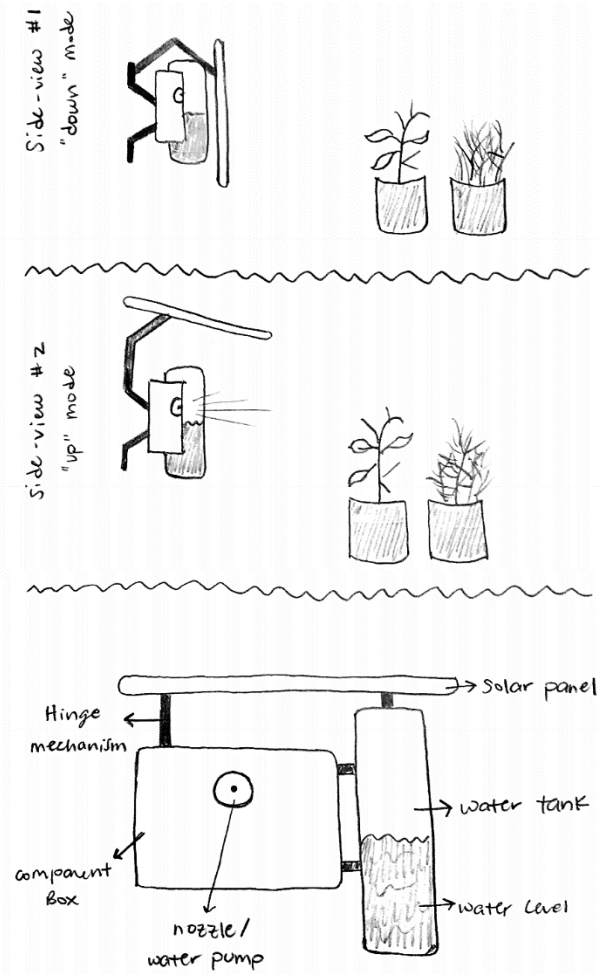
We found that eTape manufactures different versions of these continuous fluid level sensors, and we decided on one of their 12-inch versions. We felt that this may be large for some tanks, but the sensor has no problem sticking out the top of a tank and its size makes it usable across a wider variety of tanks. It cost a bit more money, coming in at $39.95 for one sensor plus a resistor and required pin connector but we felt that, despite the cost, using this type of level sensor would be beneficial to our project. It is more accurate than an ultrasonic sensor, with none of the drawbacks. As long as the strip sensor fits into the tank, it can measure any amount of liquid in it, whereas the ultrasonic sensor needs a minimum distance between it and the liquid. End users won't have to mess around getting the ultrasonic sensor mounted correctly to the top of the tank, the strip sensor can be placed on the side of the tank and attached with tape and will work as long as the pins are not submerged.

For the temperature, pressure, and humidity sensors we originally decided to go with the BME280 from Bosch Sensortec which combines them all into a discrete little package for only $7.58. Unfortunately, we realized we would not be able to use the BME280 once we received them in the mail since they were extremely small in size and nearly impossible to solder into our PCB with our current soldering tools and experience. Instead, we will be using the DHT22 from Aosong for temperature and relative humidity sensing. For barometric pressure sensing we will be using the KP235 analog sensor from Infineon which cost $8.43. While each of these cost more money and take up more space than one of the Bosch sensors, we will actually be able to use them effectively for this project and we will take advantage of their increased accuracy.

Lastly, we have a few miscellaneous items that are needed to make this project possible. Those items include resistors, relays, capacitors, diodes, etc.

III. System Concept

As seen in the sketch below, our portable watering device will consist of three main physical components. The first component is the rectangular box that will house the water pump and the PCB with all the supporting sensors. This box will make sure that all the electrical components are safe and protected from all the outside elements (i.e. water, dust, excessive wind, etc.).

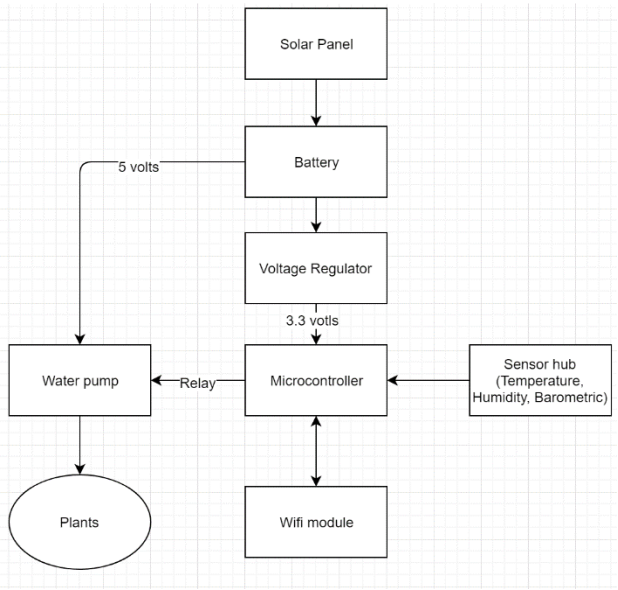


The second major component is the water tank. This tank will be attached to both the wall mount and the PCB box. The wall mount will give it structural support to stay in place and the PCB box connection will allow the water tank to talk to the PCB with its integrated water level sensor. For the first few prototypes, the water tank will be a fixed height and width determined by us. However, in the future we hope to replace that with a modular system that can adapt to a variety of bottles so that is it up to the user what kind of size tank they want to have.

The third component is the solar panel. The solar panel is the biggest physical component, and also the most expensive, but we'll get to that later. This component will be attached to a hinge mechanism that will allow it to move up and down. This mechanism will allow the entire device to have a smaller footprint when in its "down" position. This smaller footprint will allow the device to be easier to

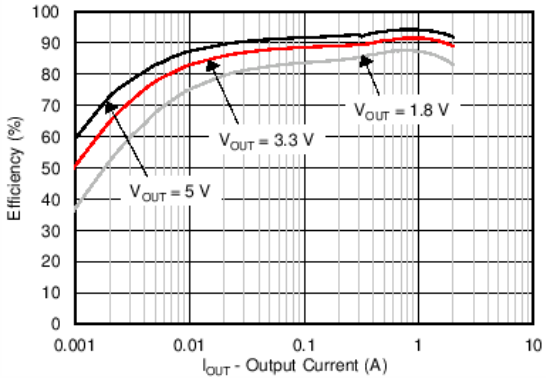
carry and move around and also to package if we want to think that far ahead. In the “down” position, the solar panel will sit right in front of the “component” box and the water tank. In the “up” position, the solar panel will sit on top of the component box and the water tank at an angle. This will no longer block the water pump and will allow the solar panel to receive the highest amount of solar energy.

IV. Hardware Detail



Powering the board will be a solar panel and a battery. These two devices are intended to supply a steady 5V to the voltage regulator, that powers the sensors and the microcontroller, and all other components. The battery is included because it can hold a charge when there is no access to direct sunlight, keeping the integrity of our system intact. The battery will be connected to a relay that will help operate the water pump, and a voltage regulator will be connected between it and the microcontroller and sensor hub. The regulator will keep voltages at a constant level, preventing damage and unwanted brown or blackouts.

The voltage regulator will be used to step down our voltage for our sensors below five volts. Doing this requires a buck converter acquired from T.I. The TPS562200 takes voltage between four and a half to seventeen volts and steps it down according to the figure below. The optimum operating value is between 3.3 and 4.4 volts for the sensors and microcontroller. This range would give a strong current of close to one amp which is plenty to supply the four devices that will be powered from this regulator. By using a basic voltage divider circuit at the output of the buck converter, it is easy to control the output voltage to the necessary level. For this project 4.2 volts with around a .75 amp current will work and have some extra power in case a part in the design draws more power than expected. The figure below shows output current and efficiency depending on output voltage [1].

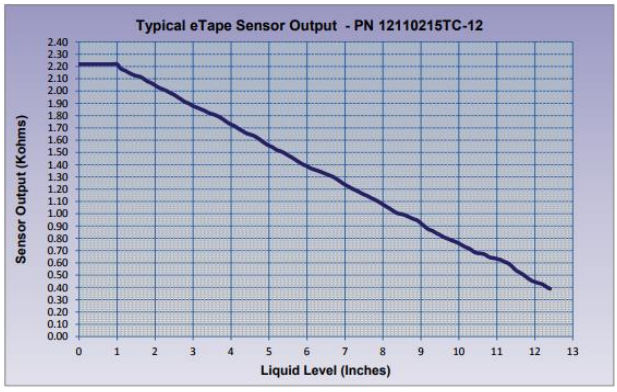


After researching each of the microcontrollers, we noticed they each showed a benefit whether it was the highest clock rate, the lowest power draw, most on chip memory or lowest power consumption. After doing initial research it was noted that memory would also have to be considered in this process to help reduce size and cost of the printed circuit board. While architectures were considered in our choice, the microcontrollers all had similar reduced instruction set computer architectures. While GPIO pins are also very important it was found that all of the controllers had enough pins to support all of the peripherals used by the project. The design of this project will not be executing too many crucial instructions. It will receive data from three sensors, transmit that data to an application, and drive a relay if necessary. Since the data isn’t too heavy then the controller doesn’t need to be too extreme. After careful consideration and much research, we decided to go with the ATmega328P-DIP package as our microcontroller of choice.

For our water pump selection, it was imperative to have a pump that was able to move enough water at a good rate to have a great projection of water which could be controlled to cover the patio or balcony of an apartment. We also took into account the possibility that the voltage being produced for the pump may not be able to reach the highest max it can handle. Therefore, we had to find a pump that could produce a great flow with a wide range of turn on voltage. This creates a good flow and will allow us to direct the spray direction and distance with ease by purchasing or 3D printing a simple nozzle. With flow rate considered to be a non-issue with our choice in pumps, the comparison of turn on and operational voltages began. Most water pumps are able to operate around 12 Volts input which may not be able to be completed at all times. With the potential of battery failure or a part of the PCB drawing too much power it is important to take into account a range at which pumps can operate. The two best pumps at doing

this were the Lightobject EWP-7L9 and the 3M Water Circulation Micro Brushless Water Pump. The Lightobject could operate at small voltages which is perfect for our low-power design but would essentially still need a boost converter to operate it. With a greater range of operation voltage and a difference of only $3.00 compared to the Lightobject it was the obvious choice for us to use the 3M Water Circulation Micro Brushless Water Pump to drive our plant showering aspect of the design.

Sensors will be gathering data and sending it to the microcontroller. Three major types of data will be gathered in order to make the decision to water the plants or not, they are: temperature, humidity and barometric pressure. For all three of these types of information the BME280 sensor managed to capture all of the environmental information easily. Unfortunately, this device was also found to be too small to hand solder onto the Printed Circuit Board. With this downfall, it was discovered the project would require a few sensors in order to operate at its full potential and have the ability to be assembled by the designers by hand. After searching with the new criteria, the DHT22 temperature and humidity sensor was found. This device works perfectly with the Arduino bootloader that is on the Microcontroller and has through hole based soldering to make it easier on the designers. For barometric pressure the KP235 matched the criteria of an operating range between 30 and 120 kPa set for the sensor and also having a surface mounted design large enough to be soldered by hand. For the water level sensing it was originally planned to be an ultrasonic sensor but was found to be affected by the water and humidity too much. Therefore, it was changed to a water level sensor that uses the pressure applied to the length of the device and registers it as a resistance, that resistance is then translated into a level between 1 and 0 for the level of water in the container, this can be seen represented in the figure below [2].



In order to decide which solar panel type is best for the sprinkler system, we studied various different kinds of solar panels available that have different manufacturing processes and use different materials. Because they use different materials, efficiency results vary as some elements gather heat and convert heat to electricity faster than other elements, and also some save more power than others.

After comparing the solar panels, we can say that it will be best to use a single-crystal/monocrystalline solar panel to power the sprinkler system. This is because we know that the water sprinkler system is a small-scale system that will only water a few plants sitting in an apartment balcony. For a small-scale system, we will need a panel that is small in size and that can generate enough power efficiently otherwise if we have a polycrystalline panel, it will take up quite a bit of space in the balcony. The monocrystalline solar panel meets all of these requirements. Voltaic Systems claims that these monocrystalline solar panels have an efficiency of 19% and that they are waterproof. Because of this, Voltaic Systems V44 panel was found to be the best panel we could use for this sprinkler system.

There are many types of batteries available in the market today, however, the one that will be used for this project is a portable battery pack or a power bank. This is because power banks are small size batteries that allow us to charge small devices like phones, or tablets. After carefully analyzing each type of battery technology, it was concluded that the Lithium-Ion battery is the most promising battery for this project. It has the highest charge storing capacity, it provides high current output, and that it has a very slow discharge rate so that it doesn't runout of charge quickly when it is not being used by the sprinkler system to get power. The battery must have at least 2 USB ports with one being a 2-amp port or higher and have a large milliamp per hour storage capability. The V44 battery has two ports containing a 1-amp and 2-amp output as well as a 12,000 mAh battery for long lasting life in case the solar panel is damaged.

V. Software Detail

The smartphone platform of choice for this project will be Android, with the Integrated Development Environment (IDE) of choice being none other than the official IDE for Android App development, Android Studio. The reason for this choice is because Android is a very flexible and powerful platform with an immense number of tools, which have everything we need to create our product. Apple's iOS platform is also very popular, but it requires the use of an Apple computer and XCode in order to write any code for its platform, a device only one out of the four team members owns. This constraint is one of the main reasons why the smartphone application will first be developed and tested on the Android platform, and then may eventually have an iOS version if time permits.

The Android Studio IDE provides one convenient location for us to code, build, and test our application. Although a few members of the development team have

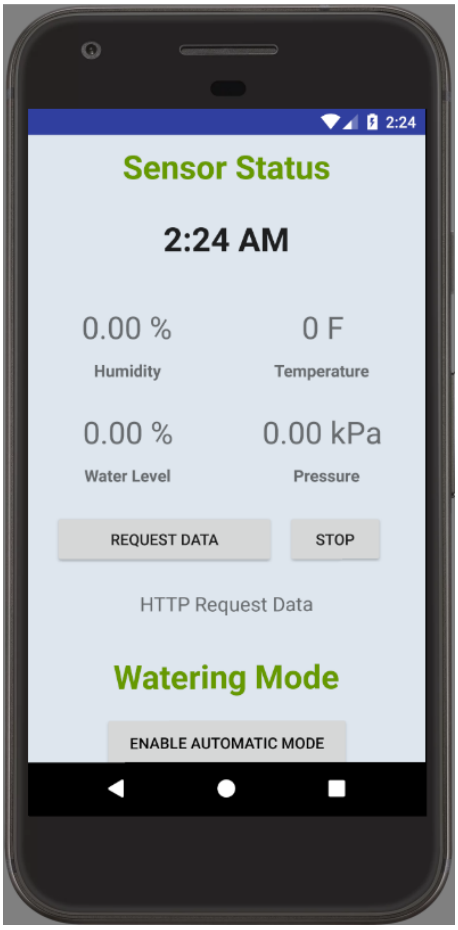
Android phones, Android Studio provides us all with a built-in emulator, allowing us to choose a variety of phones to upload and run our code on, provided we have enough RAM, without having to risk damaging one of our own phones. For our microcontroller and the Wi-Fi Module, we will be using the Arduino IDE to load sketches and flash the module.

The code that will be running within the microcontroller will have various functionalities. The first step is to be able to properly read the incoming data from the various sensors. We will first need to set up a serial connection with the sensors so that they are able to send data to the microcontroller unit for processing. After performing an initial setup, initializing the serial connection and any variables we will need, we can jump into the main loop of the microcontroller, this is where we will request data from the sensors on the board. Luckily, we will be able to use some existing Arduino functions, such as analogRead(), to help us grab this data from the sensors, as opposed to us having to create our own communication protocol from scratch. The analog to digital converter also aids us greatly in helping to grab data from our analog components. We will need to make sure the barometric pressure sensor and water level sensor are both connected to analog pins on the microcontroller and then they will have their analog signal converted to a digital one. From there, we will be able to take that digital data, manipulate it, and send it out to our application to display the results to the end user. The process is largely the same with the rest of our digital components, although there will be less conversion and overall processing to do before the data is ready.

At this point, we are almost ready to let the rest of the code take over and deliver this data to our application, however, we want to make sure we are getting accurate readings. If we were to take just one sample to grab data from one of our sensors, it could be tainted by noise from the rest of the circuit or other uncontrollable factors. Since we want to get accurate readings, we will be taking five samples from our sensors and taking an average of all the readings. We arrived at five samples, with small delays between them, after reading through some of the datasheets for our sensors. We believe this will smooth over any random noise or signals that may be present during any one reading of the sensor, and the cost in terms of time and power is not very high.

Once we have the data from the sensors then it is time to put the data to use and let the code make decisions. One of the main parts of the microcontroller code will be the algorithm that takes care of the "Automatic Watering Mode" of the device. Although we are still working on the fine details, the main idea is that the algorithm will be able to automatically determine when would be the best time to water the plants based on the data it is receiving from the sensors. We are currently doing research to determine what values from each sensor will be the deciding factors when it comes time to water the plants or not.

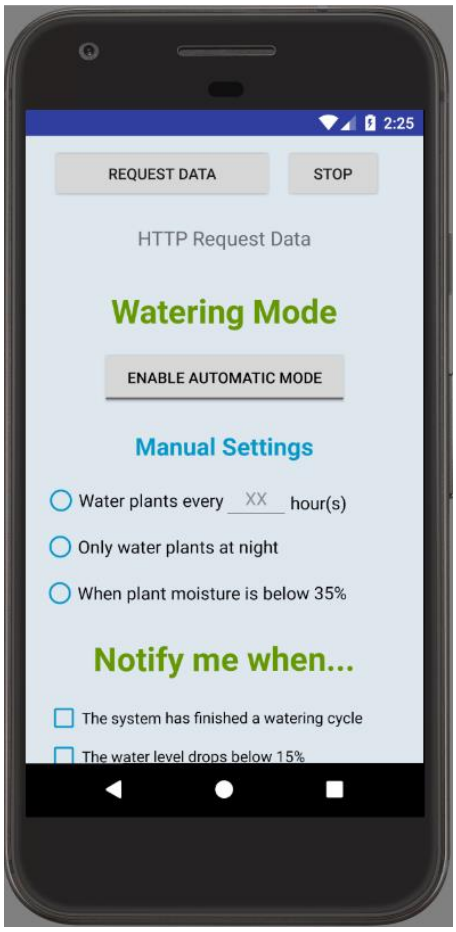
The design of the user-interface (UI) for the Android application will follow a simple one-screen system with different sections. As the user scrolls through the screen, they will be presented with more and more options. This layout is not only simplistic, but also widely used. Having a layout most users have already seen before will make the application very easy to use. The application will revolve around three main sections: Sensor Status (Home), Watering Mode and Notifications. Below we can see a prototype of the application's Home screen.



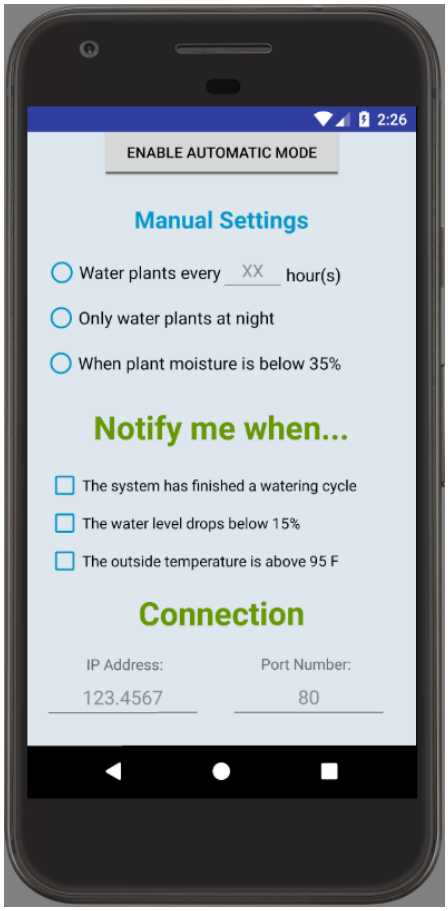
When the application is first opened, it will send the user directly to the "Home" screen, which is the default. From this screen, users will be able to see all the relevant information from the sensors, which includes the temperature of the place in which the device is located, the humidity, the amount of water left in the water tank, the surrounding pressure, among other things. This information will give the user a quick glance at the state of the system and its surroundings. Since our device communicates to the mobile application wirelessly, the user can be anywhere and

still be able to check the state of the system without having to physically walk out to their balcony.

The second section is the "Watering Mode" screen, showcased below. From this section, user will have the ability to choose between the automatic watering mode or a manual watering mode based on their preferences. The automatic watering mode will make use of the sensors in the device to determine when is the best time to water the plants based on our algorithm.



The "Watering Mode" section will consist of a button to enable/disable the automatic watering mode, followed by a list of options for the manual watering mode when the automatic one is disabled. Once a choice is made, the other options will be unselected until the current option is turned off. If the automatic watering mode is disabled, the user will be able to choose from a list what option they want to go with for the manual watering mode. For example, they will have the option to either water the plants only at night every day, or once every certain number of hours, or maybe even just water the plants once the moisture drops to a certain level, just to name a few. This way, if the user feels like our automatic setting is not the most efficient, they will have other options. It is important to note that new options can be added later on with software updates based on customer feedback, if this device was to make it into the market.



Lastly, we have the "Notifications" screen, showcased above. From this section, users will have the ability to set up their notification preferences for the device. This screen will consist of a list of events that the user would like to get notified about. Each event will have a toggle next to it which the user can use to enable or disable the notification for that event. The top of the screen will have a title that says: "Notify me when…" and will be followed by the list of events under it. Some of the events will be: "the water level drops below 15%", "the outside temperature is above 95 degrees F", "the system has finished a watering cycle", etc. With this system, if any of those events happen on the device, the user will be notified on their smartphones through the app. For example, if the user has the water level event enabled, they will get a notification as soon as the water level reaches 15%, allowing them to go add more water before it completely runs out. Just like the Watering Mode screen, more options could also be added to this section in the future based on needs or new requirements.

VI. Wireless Communication

In order to communicate with our device from our smartphone application, we need a way to communicate with it wirelessly from long distances. This includes set up, scheduling for watering, power on/off and more. While Bluetooth was considered for this project, it ultimately did not fill all the needs we had for the device, especially since Bluetooth is a short-range communication. The ZigBee technology was also briefly considered but the lack of support online and examples made it a non-appealing option. Therefore, we decided to go with a Wi-Fi-module the board. For an Arduino-based microcontroller, there are two popular options for Wi-Fi modules.

The first one is the Arduino Wi-Fi 101 Shield. This component is a powerful Wi-Fi module with crypto-authentication (developed with ATMEL) that connects an Arduino-based board to the internet wirelessly in a very simple and effective way. It connects to the PCB via SPI port, supports TLS 1.1 (SHA256) and WEP/WPA2 Encryption types and uses the IEEE 802.11b/g/n standard for its connection. It also supports both 3.3V and 5V operating voltages.

The second option is the ESP8266 Wi-Fi Module. The ESP8266 is a self-contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to a Wi-Fi network. With its default firmware, it has the same functionality as the Arduino Wi-Fi Shield, but at a lower cost. It is also designed to occupy minimal PCB area, making it smaller than the Arduino Wi-Fi Shield. The only disadvantage of the ESP8266 is that it is not capable of 5-3V logic shifting and will require an external Logic Level Converter in order to work.

As mentioned previously, we decided to go with an ESP8266 Wi-Fi module to handle communications between our Android application and the microcontroller unit. The ESP8266 has a few modes of communication, but for the purposes of our project and demo, we will be using the module to connect to an access point that our Android phone is already connected to. Once connected to the same access point, the Wi-Fi module will receive commands over HTTP from our phone. It will then take those commands and transmit them over serial to our microcontroller. After the microcontroller parses the string, it will acquire data from the chosen sensor, format a new string to transmit back to the ESP8266 module, then begin transmission. Instead of receiving and immediately reading and manipulating this new string, possibly losing data in the process, the Wi-Fi module feeds the data into a character array that we then manipulate. We package up this array into an HTTP response, and send the data back to our phone, where the Android application takes over processing.

VII. Project Constraints

Time is a very precious item a lot of us take for granted. With every group member taking multiple classes, working and some even having extracurricular activities to attend for an organization, time is definitely something we could use a little more of. With so little free time, time management and planning are two of the main values we try to improve everyday so we do not fall behind with the project. By setting up deadlines and milestones, we have been able to stay on track and focused during this planning stage.

Power is another constraint. A major factor of this design is the fact that it is self sustaining. In order to achieve that the solar panel will provide a certain amount of power to keep the system running. With the solar panel we have picked out a max power output is at 9W. This means we have to keep the power consumption of all the parts of the design to a minimum. Anything exceeding this maximum amount of power draw and the design will fail to work at all. This is also an important factor in keeping the battery charged longer. The more power draw the faster the battery will drain. This limits the amount of peripherals added to the design and the types of components we can use. For instance a water pump limited to a lower voltage.

One constraint that was not really thought of at first is the availability of the parts and components we planned to get. While doing research, we were able to narrow down our selections for each component technology and then proceeded to finding the actual components that would fit our needs. To our surprise, a lot of the components we looked at were out of stock temporarily. This would not be a big deal if we had a lot of time to create this project. However, with a deadline of just a little over 4 months, parts need to be ordered as soon as possible so they can start being tested and implemented. We don't have the luxury of waiting for parts to become available again so we can order them. For this reason, a few components had to be chosen based on availability and not necessarily because they were the absolute best option.

Since the parts will be hand soldered onto the printed circuit board it is important to select parts that are able to be handled properly and soldered correctly onto the board. Pins should have a fair amount of distance between them so the risk of soldering the pins together or incorrectly is down to a minimum. Also, the parts themselves should be large enough to be handled by the person soldering. To small of a part and the device might not be able to be soldered to its ports properly. Since the soldering will be done by teammates accuracy of soldering may be considerably lower than a machine soldering job. This requires us to use precision and patience in order to achieve the best form of solder connections. While this saves on cost for the project it leaves room for a lot of errors with the connections between all of the parts on the PCB. One small mistake can

short the component or worse the board and the group would have to restart their PCB with a fresh board.

While there are many more constraints, these are just some of the main ones we faced as a team when developing this device.

VIII. Component Testing

When working with electronics, you never know if things are working properly until you test them. Our approach for hardware testing was pretty simple. We began by testing each component individually. We checked to see if the solar panel was able to charge the battery to full capacity, if the battery was able to supply 5V output at 2 amps for a long period of time without interruption, if the power circuit was able to convert from 5V to 3.3V and keep constant current out and if the relay circuit got switched by input voltage and could drive the water pump through a relay terminal. After individual testing was done, we went ahead and starting connecting all the components on a breadboard to see if we could retrieve all the sensor data and power all the systems simultaneously.

IX. Budget & Funding

Below is a table that illustrates the price of each component and how it compares to the others. The solar panel and battery combo is by far the most expensive.

Component	Price
Water Pump	$12
Solar Panel & Battery	$159
PCB	$50-100
Micro-controller	$2.18
Temperature Sensor	$9.95
Humidity Sensor	
Barometric Sensor	$8.43
Wi-Fi Module	$6.95
Water Level Sensor	$39.99
Water Tank	$1-5
Miscellaneous Parts	$20

This project will be entirely funded by the team. It will be split evenly amongst the four teammates and will not have a sponsor to take over the cost of the project. This means each member is responsible for one-fourth of the cost ranging from $290 to $350. Each member of the group has agreed to this stipulation and believes it is fair. Receipts and other forms of proof of purchase will be saved by each team member and presented at a later date to form a final total for the project, get an idea of who spent how much, and how each member should be reimbursed.

X. Conclusion

The research and development of the self-sustaining plant watering system was at first a large task to handle, with only four group members. The research needed for the solar designs, microcontroller, sensors and pumps were vast seeing as though the market was flooded with numerous designs and we had to obtain the best ones for our project.

Our creation, a self-sustaining plant watering system, was met with many complex and compact designs already marketed today. With such a vast number of designs to compare to, we were able to shape our design into the best possible way for the constraints we obtained ourselves. By combining the power of the sun, with the low-power-mode based microcontroller, we were able to supply enough power to have each sensor function properly as well as obtain a desirable amount of force from the pump to push out water for our plants.

Our goal was to design a simple and easy to take care of system that could operate on its own or manually. The integration of the software application with the hardware allows our device to complete these tasks in an efficient way, leaving little for the user to do. The only interference needed is to set a time or state to water the plants and adding water to the tank when it is low. This design is ideal for those busy workers or students who don't have time to take care of their vegetation. As a product that could be marketed to a broad range of users, our desire of creating a wide used device was complete.

The completion of this project signifies the coming together of thorough research and design concepts to achieve the most efficient and user-friendly device. Overall, the group is proud of this design and the work put in to achieve a device that integrates all of the fields studied as well as providing a product that helps keep our environment green.

References

[1] http://www.ti.com/product/TPS562200/datasheet

[2] https://cdn-shop.adafruit.com/datasheets/eTape+Datasheet+12110215TC-12_040213.pdf