The Smart Garden Controller – Automating and Analyzing a Garden

Alexander Burns, Geovanny Chirino, Temple A. Corson IV, & Renan Coelho Silva

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — Cultivating a home garden is common pastime among families and individuals in the United States and around the world. The objective of the design for this project is to assist home gardeners and small farmers in maintaining and understanding their garden. The system autonomously waters a garden's contents with an automated schedule and on-demand watering features, records sensor data, and interfaces with the user through a web based application as well as buttons and an LCD placed on the device. The data to be collected by this system includes, but is not limited to, Temperature, Humidity, Pressure, and Soil Moisture.**

***Index Terms* — Autonomous Systems, Environmental Factors, Humidity Measurement, Soil Moisture, Temperature Measurement, User Interfaces.**

I. Introduction to the Smart Garden Controller

Gardening is time consuming. Having to go to out of the house to turn on watering systems and trying to determine for how long to water the plants can be difficult. The smart garden controller is a solution to some of the work involved in making sure the home garden or a small farm is taken care of. The smart garden controller also collects data regarding soil moisture, temperature, and light. The metrics collected by the smart garden controller sensors are very useful in determining how often and how long to water the garden.

The primary goal for the smart garden controller is to take away the manual labor involved in watering plants and automate the watering maintenance. The smart garden controller will allow the gardener to set a schedule for each zone of the garden independent of one another. The gardener will be able to control how often to trigger the irrigation as well as for how long to water the plants. The Smart Garden controller will also allow for on demand watering session if more time is deemed necessary. Both on demand and scheduled watering sessions will be controlled via a web interface. An LCD and buttons will also allow for on demand watering sessions to be triggered directly via the device as a fall back to the web application.

The second goal of the smart garden controller is to provide metrics about the garden. Moisture sensors in the soil will detect the capacitance of the soil to determine the moisture. The soil sensor will also gather temperature and light. Another sensor will collect air temperature, humidity, and pressure. All the data collected will be uploaded back to a database. Based on the metrics collected, the user will be able to adjust how often the different zones should be watered while taking into account the sensor data collected. All the metrics should be available for review via a web application.

Once the system is setup the user should be able to fully control the watering system and review metrics via the web application. The smart garden controller lets the user manage and analyzer his garden without ever leaving his computer or home. Therefore, the following are the main detailed requirements for the system.

1) The controller should be able to control and regulate watering service for the garden. Each watering section will be independent of the others, meaning that each zone could be watered according to a different schedule, and any number of zones could be watered on-demand.
2) The system should collect sensor data from at least three of the following six physical phenomena: temperature, relative humidity, atmospheric pressure, soil moisture, ambient light, and wind speed.
3) The system must have a user-interface for the user to set a watering schedule for at least one zone, initiate an on-demand watering order for at least one zone, and to view the data collected by the system.

II. Existing Solutions

Most technologies build upon existing ideas and technologies, the smart garden controller does as well. Comparing our requirements and design to that of existing technologies is time well spent. Review of previous products can help a project stay on track by keeping requirements realistic. Also, evaluation similar technology and previous projects can help avoid pitfalls seen in other projects. Finally, knowledge of existing device similar to the smart garden controller will show if this solution is well developed.

The following are some projects which are similar to the Smart Garden Controller. The main difference to note is that some of the following projects are for hydroponics rather than soil based growing. Another difference is that most of the mentioned projects were geared towards growing plants inside the home, while the Smart Garden Controller focuses on growing plants outside the home. Also, most of the following projects were fully contained systems, while the Smart Garden Controller's goal is to integrate with an existing garden.

The first being the Autobot. Initially, it started out as a smart garden with soil and regular sprinklers. However, eventually the project moved to hydroponics. The project had PH, light, and a water sensors. Based on sensor

readings the water level and PH could be adjusted. Also, the project had wireless communication. The project aimed at scaling down an outdoor garden to fit into a house. This project was similar to the smart controller presented here in which it aimed at growing plants. However, the AutoBott aimed at growing plants inside the house while the smart garden controller aims at the outside garden

The second being the Energy Sustainable Hydroponics with Automated Reporting and Monitoring. The system aimed at automating some of the tasks necessary for maintaining a hydroponics system. The system was similar to the AutoBott project previously mentioned, and in addition uploaded data collected from the system to a web server.

The design found which most closely matched the smart controller was the PASS, Plant Automated Sustainable System. The PASS did not have a web interface and was controlled via Bluetooth. The PASS used regular soil, compared to most of the other projects found which focused on hydroponics. The user needed either a tablet or a phone to interact with the device. The system used a network of sensors to determine when to water the plants. However, the system did not provide metrics for the user for review. Also, the main focus of the system seemed to be portability and for an audience which needed to plant indoors.

Thorough research and examination was expended in the effort to properly design and implement a suitable and successful project. The system design in any project is a crucial piece of design work that takes the projects end goals and makes them realizable using the technology that is available, and in a way that is in line with the financial and functional goals of the end product or user. The system design must account for the financial, economic, technological, and ethical requirements and constraints throughout the life of the project to ensure that all goals are met within the limits of the constraints. A good system design requires research on relevant technologies and is often accompanied by a number of trade studies that offer a tradeoff analysis of certain aspects and characteristics that pertain to the overall system performance. These tradeoffs are often compared between competing technologies so that the designer is easily able to discern what options seem best suited for an application, and what the tradeoff with that selection would be.

The following subsections detail our research in each specific area and provide clarification on the final part selections with supporting evidence for each item. For each subsection there will be a trade analysis that demonstrates the different options that were available and the pros and cons of each option that ultimately led to the final decision

III. Strategic Components & Design Decisions

A microcontroller is an embedded system that contains a processor, memory, and peripheral interfaces all contained within a single chip or module. Microcontrollers are used in almost all electronics and embedded devices. Microcontrollers are cost effective. Most microcontrollers use very little power and have sleep options that greatly improve battery performance. Some of today's microcontrollers even provide wireless connectivity. In a way, microcontrollers are platforms which abstract the circuitry necessary to control hardware. Therefore, microcontrollers are great for improving reliability and efficiency of products.

For the purposes of this project, the microcontroller will serve as the brains of the entire system. The microcontroller will be responsible for running the main firmware of the system which executes the decision algorithm based on the sensor and user inputs. The microcontroller will also be responsible for all timer based and calendar functions, including setting up and properly running and monitoring the necessary timers, and executing the necessary functions based on the timer interrupts. The microcontroller will be responsible for handling the wireless interface and communicating with the web server used for the user interface. This user interface will allow the user to access the device via WiFi and update settings as well as monitor sensor analytics. Finally, the microcontroller is responsible for running all peripheral devices including; LCD display, user interface buttons, weather sensors, soil sensors, water valve control, etc. Our microcontroller needs for this project drove us to evaluate the following three microcontrollers as possible solutions. The main requirements that were evaluated in our microcontroller selection were the microcontrollers WiFi capabilities and ease of use and integration for WiFi. The microcontroller needs to be low power so that battery life could be optimized. The microcontroller needs to have sufficient interface options to easily optimize the system design.

The Texas Instruments CC3220MODA is a low power IoT microcontroller. The wireless microcontroller module that contains the built in WiFi connectivity is FCC, IC, CE, and Wi-Fi Certified. The CC3220MODA uses an 27 ARM Cortex-M4 processor running at 80 MHz for faster more reliable processing. The microcontroller contains a wide array of peripheral interfaces, including I2S, SD/MMC, UART, SPI, I2C, and a 4 channel ADC. The Wi-Fi network subsystem contains an additional ARM MCU that off loads Wi-Fi tasks from the application MCU. The network subsystem contains the 802.11 radio, a baseband layer, MAC layer, Driver layer, Supplicant layer, TCP/IP layer, SSL layer, and IP layer. The network subsystem on this module would allow for 256-bit encryption and supports WPA2 personal and enterprise, as well as WPS 2.0. All of these features combined lead Texas Instruments to claim

that this module requires no prior Wi-Fi experience for development. The CC3220MODA utilizes Texas Instruments' Integrated Development Environment (IDE), Code Composer Studio. This IDE is familiar with all team members as all members of the team have used this IDE for development with the MSP430 microcontroller in previous projects. The level of comfort the team has with Code Composer Studio did play a part in the decision between these three microcontroller modules. The CC3220MODA also comes equipped with an on-board chip antenna, which would offer better wireless performance than the trace antennas of the alternative modules. Some wireless microcontrollers did not have an antenna. For such cases, and antenna would need to be built into the PCB or connected to the device. Therefore, the built in antenna contained in the CC3220MODA MCU removes the risk and effort involved in either creating our own antenna or attaching it. Knowing that the CC3220MODA is intended to be interfaced with using Code Composer Studio and taking into account the fact that the module is designed for Internet of Things low power applications with a focus on easy Wi-Fi integration, it was not a difficult choice to select the CC3220MODA. Along with the obvious advantages of using the CC3220MODA by analyzing the features out of the box, our group is also able to leverage historical knowledge of development with this part. At least one team member has been intimate in the electrical design and system application of the CC3220MODA in a real-world project that was the result of an internship opportunity. Documentation can often make or break the efficiency of a team when developing software for an MCU. Texas Instruments has multiple online resources which will be helpful in developing, troubleshooting, and understanding the MCU used. Some of the resources include online tutorials, pdf, and an active user community. While in the battle of price per module the CC3220MODA came in last place, there were many other considerations that influenced the decision to move forward with the CC3220MODA as our microcontroller/WiFi module for the Smart Home Garden project. The CC3220MODA uses the Cortex-M4 processor, which is a well-known high performance, and reliable processor. The current consumption was comparable to the ATWINC1500 during receive and transmit conditions, but neither compared to the low current consumption of the WiFi activity of the ESP8266. While this low power usage for WiFi activity would help the unit conserve battery life, therefore creating a longer run time between charges. But the determination was made that, in the trade between low current consumption and complexity of design and implementation, a less complex design and easier to use integration environment were more favorable than the lower current consumption. The memory allocation between the three different modules was another major factor that was taken into consideration, and in this category, the CC3220MODA reigned supreme with 256KB of RAM, which is double that of the ATWINC1500, and more than quadruple that of the ESP8266. This memory would allow for larger code and execution space and reduces the risk of having to optimize code that is too large to run on a given microcontroller. The flash memory is less than that of the ATWINC1500, but with the SD/MMC interface of the CC3220MODA we maintain the ability to upgrade the unit with an SD Card or SD memory interface to expand the flash memory if it is deemed necessary. The final characteristic that led us to settle on the CC3220MODA is the fact that it is able to operate down to 2.3V rather than 2.6V as in the ESP8266 and ATWINC1500. This extra 0.3V of operating voltage will allow the unit to maintain its normal operational state for a longer period of time as the battery degrades and the voltage drops lower and lower. The ability to leverage the experience of the previous implementation of this module in an electrical design, along with the functionality provided by the module itself and the deterministic factors that were derived from the trade study, led us to choose this part as the microcontroller and wireless interface for the Smart Garden Controller.

Fig. 1 Microcontroller Trade Analysis

Microcontroller Trade Analysis			
Module	ATWINC1500	ESP8266	CC3200
Price	$ 8.08	$ 5.00	$ 9.35
Voltage range	2.7V-3.6V	2.7V-3.6V	2.3V-3.6V
Processor	Cortus APS3	Tensilica L106	Cortex-M4
Bits	32-bit	32-bit	32-bit
Receive Current	61mA	56mA	59mA
Transmit Current	265mA	170mA	278mA
RAM	128kB	50kB	256KB
Flash	4MB	n/a	64KB
Wifi Protocol	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n
Data Rate	72.2Mbps	72.2Mbps	54Mbps

The sensor chosen for our application is the I2C Soil Moisture Sensor version 2.7.5 by Catnip Electronics. It delivers readings for relative moisture level, Temperature, and Ambient light. This sensor was chosen considering its communication protocol (I2C), its price, and its measurement methods and capabilities. Compared to other sensors which can only read a single metric, this choice was able to cover three of our six major data points, and therefore was a clear superior. The moisture level measurements are taken by this sensor using a capacitive moisture sensor. This technology is widespread and reliable, as it is used in several fields on consumer electronics including but not limited to touchscreen devices, position & acceleration sensors, and humidity sensors. When the sensor is taking measurements in soil, the capacitive surface will form a circuit with conductive substances, water for our purposes, and cause a specific

voltage drop corresponding to a moisture level datum that is sent to the microcontroller. The capacitance of this circuit will increase or decrease depending on the amount of water in the soil touching the sensor (respectively).

This sensor operates reliably within the temperature range 0°C to 85°C (32°F to 185°F). Temperature readings below this range within reason are not expected to damage the sensor, but the temperature measurement will never read a negative value. Temperature readings above this range are not possible in the operational environment. Moisture readings taken in an environment above 30°C (86°F) are expected to show some drift from the expected value, but this variation is well within reasonable error. The figure above shows that the moisture reading reaches a maximum of less than 4% error in environments above 50°C (122°F). Any measurement error above these temperatures is negligible due to the impracticality of the systems use in environments with temperatures consistently above that threshold.

This sensor also has the ability to continuously sample data at a maximum rate of 16 MHz. When the sensor is in the state, it uses an average of 4.5mA with a supply voltage of 5.0V (2.8mA at 3.3V), which is significantly less current than the single measurement consumption rate. Considering this, if several measurements are necessary in a short period of time, the constant polling mode is superior in terms of current consumption. There is variation seen caused by voltage drop over time due to loss in the capacitive circuit. This drop is corrected consistently with a change in applied voltage to the capacitive surface. The peaks of the data indicate the current relative moisture level, approximately 530, which is based on the capacitance of the sensor. The temperature data recorded by this sensor is saved to the I2C bus in terms of tenths of degrees Celsius, giving our application three significant figures of accuracy for temperature. Ambient light and moisture levels are recorded and saved in terms of a single scaler integer of three significant figures and five, respectively. Both of these measurements are approximately linear.

The sensor chosen to measure Humidity and Atmospheric Pressure is the Bosch BME280 Humidity and pressure sensor. This sensor was chosen considering its communication protocol (I2C), its price, and its measurement capabilities. Several versions of this sensor are available, with different target data points including temperature and ambient light, however this version was chosen in conjunction with the other sensors to reduce the number of sensors needed for the system. The atmospheric pressure is recorded by this sensor and saved to the I2C bus in terms of hectopascals (hPa) with a range of 300 hPa to 1100 hPa, and the Humidity is recorded in terms of Percentage of relative humidity with a standard range of 0% to 100%.

The LCD screen selected for use in this system is the LCD-013-420 Display Module. This component can display four 20-character lines of white text on a blue surface with a backlight. The backlight will improve readability in low light conditions, for example in case the user has to configure the device at night and outside. The LCD turned out to be a bit bigger than expected, however we believe this turned out to be a good point for readability.

This module uses I2C serial interface protocol to communicate with the CC3200MODA microcontroller. This screen was ideal for use in our application due to its price, size, interface protocol, and its ability to meet the functional requirements of the design. This screen can display all necessary characters essential to user understanding and manipulation.

The system design originally included Smart Home Speaker integration using devices such as Amazon Alexa and/or Google Home. After careful consideration and development of the system our team decided to omit this functionality in favor of successful implementation of the basic system features.

Fig. 2 Part Selection Summary

Item	Name/ID	Manufacturer / Distributor	Cost
Smart Speaker	Echo Dot	Amazon	$49.99
Anemometer	1733	Adafruit	$44.95
Humidity & pressure sensor	BME280	Digi-Key Electronics	$16.62
Moisture, temperature, and light sensor	I2C Soil Moisture Sensor	Catnip Electronics	$13.00
Microcontroller	CC3200MODA	Texas Instruments	$10.00
Liquid Crystal Display (LCD)	LCD-013-420	SuperDroid Robots	$24.90

IV. Hardware Design Details

The initial architecture for this design is intended to assist the user in the proper maintenance and care for a home garden. The architecture is intended to allow the user to set up and operate the system with ease. The architecture for this design was based around the fact that wireless connectivity via WiFi would play an essential role in this system. The idea was that with the ability for the system to access the internet and collect data, as well as giving the user wireless control, would set this device apart in the marketplace. With this in mind the initial step in the system architecture design was to determine the method of gaining internet connectivity. It was determined by the team that WiFi connectivity would be the most feasible as well as offering the broadest customer base possible. If we were to have chosen a different wireless interface that is not as commonplace as WiFi the product would likely have

suffered due to that fact that most users do have easy access to WiFi but are less likely to have access to another form of standalone wireless technology. This also means that without WiFi there may have been additional hardware that would need to be supplied to the user for proper operation of the device. For these reasons and more, the decision was made to use WiFi 802.11 as our wireless protocol for this proof of concept prototype design.

After selecting the wireless interface, it was necessary to evaluate the market for the appropriate wireless chipset that would allow for our desired connectivity. Much research was put into the decision to move forward with the CC3220MODA as you can see in previous sections of this document, but the decision was based on the familiarity with the part and the use of modules from the same chipset family in other successful designs. The CC3200MODA is a robust module with a highly integrated WiFi stack and standalone WiFi processor so that the wireless interface does not burden the processor. While the option of using a chip down version of this part was considered, it was determined that using the module involved less risk as the amount of integrated circuitry on the module, including the antenna, could be quite burdensome to design from scratch.

Once the microcontroller and wireless interface portions of the architecture were solidified, the next step was to determine the proper process for moving, querying, and storing data from the internet. For this it was determined that there should be a web client for the user interface. This web client would be hosted by the web server that is created to handle the wireless interface portion of our system. The web server would access a database that stores all relevant data for the system as well as the user. This database will be the powerhouse of the wireless infrastructure and will allow the device to operate the appropriate algorithms based on collected data and watering profiles that add a deeper level of intelligence to the Smart Home Garden. These aspects of the design collectively make up the wireless portion of the system that can be thought of as a small cloud for the device.

Consideration was made for users who may not have access to internet as well as a situation where a user may not be able to access their WiFi network temporarily but still need to interface with the Smart Home Garden. For these cases it was necessary to give the user some form of manual control of the device. This would require both inputs and outputs as the user would need some verification that whatever operations are being entered manually into the device are being properly acknowledged. This led us to incorporate an LCD display as well as pushbuttons for the user. This manual user interface will give the user the ability to trigger on demand watering without having access to WiFi. Aside from the user interface, the device has other system components that are contained in the main chassis housing. At the top of this list is very obviously the CC3320MODA. While this wireless module handles the wireless portion of the system, it is also the master microcontroller for the entire system. All analytics and algorithms will be run on this module which will be able to determine complex metrics based on a variety of inputs. The CC3200MODA will be the master for all communications within the system, and will collect, store, analyze, distribute, and report the appropriate data based on the configuration. The microcontroller will handle all memory allocations and data transmissions. This is really the heart and brain of the entire system.

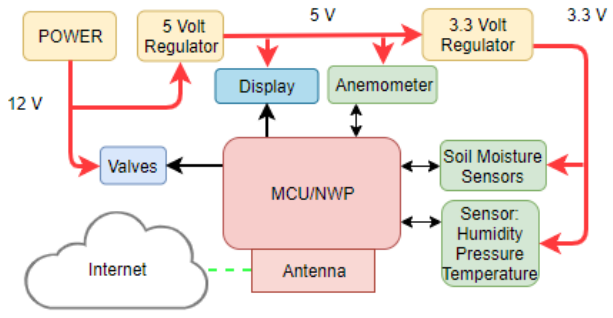
Outside of the microcontroller, but still on our custom PCB, there are a number of other components that allow the Smart Home Garden system to operate properly and efficiently. There is a JTAG interface for device programming. This JTAG interface is a standard programming interface that allows the designer to easily program the device. There is a USB to UART interface that allows terminal access to the microcontroller for debug. This interface is controlled using an FTDI chipset that converts USB terminal data into UART data automatically so that you can use terminal functions on the microcontroller for debug and testing purposes. Also, on the main board within the chassis housing is the humidity/barometric sensor that is used to measure the atmospheric pressure in the air as well as the humidity level of surrounding air. This combination sensor will aid in the weather data collection.

To power the microcontroller and all of its peripherals and architecture for a power system needed to be designed and incorporated into our custom circuit card assembly. The initial plan for the power architecture was to use battery power to operate the device in most cases and send an alert to the user when the battery needed to be charged. This initial architecture was determined to be unfeasible after investigation into the power consumption of solenoid valves for water flow control deemed to be far too power hungry for a purely battery powered implementation. Once the determination to move away from battery power was made, it was decided to power the device off of the grid using the user's home AC power. With this determination, a power architecture was created that implemented the use of an AC to DC power adapter that converts the higher voltage AC into a lower voltage DC. This power adapter will be located outside of the Smart Home Garden chassis and will require access to a power outlet. This AC to DC power adapter feeds our custom circuit board with DC voltage so that the electronics are able to operate. Once the DC voltage hits the board, it is run through multiple power components that translate the DC voltage to the appropriate levels needed to operate the different components on the board. Special care was taken in choosing these power components to ensure that the current draw of downstream

components would be easily handled by the power design with enough overhead to account for worst case situations.

Outside of the main chassis housing is where the real purpose of the Smart Home Garden becomes clear. Cabled into the main chassis housing is a series of sensors that are placed in the user's different gardening zones. These sensors are able to collect data about the moisture levels of the soil, as well as collecting temperature and ambient light data all individually. These sensor cables will be coupled with a hose that will provide each zone the ability to water separately when it is appropriate. The intent is that the user runs the coupled cable and hose to the desired location, which would likely be in the center of what is being considered by the system a zone of the garden. The sensor is able to collect data that is specific to that zone, and this will allow the user to adjust the watering cycle for each specific zone based on what that zone may require dependent on its crop. The hoses will have sprinkler heads attached to the end so that each zone can be watered effectively using the proper sprinkler head. The watering is controlled by solenoid valves that are actuated by the microcontroller. The valves are normally closed unless the proper voltage is introduced across the terminals. This means that the water to the system would have to be on and ready so that when the system determined watering was necessary, there would be water available. It is the group's intention to instruct users to dedicate a certain spigot to the Smart Home Garden so that this spigot can be left on to supply water to the system at all times. This overall system architecture is what has led this project down the path to create the design we have today. While there are items that may change based on unknowns that currently exist in the development process, we are confident that we have a solid architecture that will be successful as a prototype.

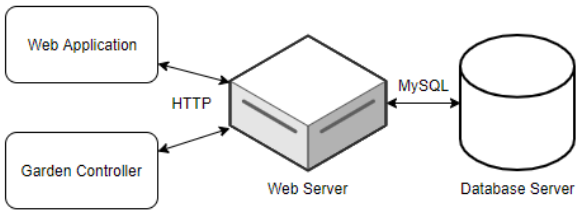
Fig. 3 System Hardware Architecture



V. Software Design Details

The MCU and the web application will communicate to the web server via representational state transfer (REST). REST "is an architectural style that defines a set of constraints and properties based on HTTP. Web Services that conform to the REST architectural style, or RESTful web services, provide interoperability between computer systems on the Internet. REST-compliant web services allow the requesting systems to access and manipulate textual representations of web resources by using a uniform and predefined set of stateless operations." In other words, both the controller and the web interface will retrieve data from the database via HTTP requests. The results of such requests will be in plain text formatted either JSON or XML, which can then be processed by either the controller or the web user interface. The web server will communicate with the database directly using a database connector. The connector is the code called to perform queries against the database and return the data back to the web server. The following image is a high graphical representation of the communication between the controller and web user interface to the web service. Because each of these sections are decoupled, not dependent on each other, development should be simplified and independent.

Fig. 4 Web Interface Block Diagram



The database will hold all the data collected by the sensors, user data, etc. The controller will periodically send data to the web server. The web server will process the data and insert it into the database. The data in the database will be used by the web application and by the microcontroller. The user via the web application will connect to the database to review the data collected by the controller. The user, based on the data collected, can then make decisions for future watering schedules, whether to increase or decrease watering. The microcontroller will connect to the database and use it mostly for scheduling. The database will be the "memory" of the system.

Once the database is setup, SQL queries will be run to create the database and the tables. Such queries can be run directly against the command line. However, another software will be used to connect to the database for the purposes of setting up the data. The software, MySQL Workbench, will allow for configuration of the database via a GUI, a graphical user interface. Using this open source software will help in development and implementation time. The workbench will simplify how the database and the tables are created. More time would be needed for creating queries, tables, and relationships without the workbench. The work bench will also be used for creating

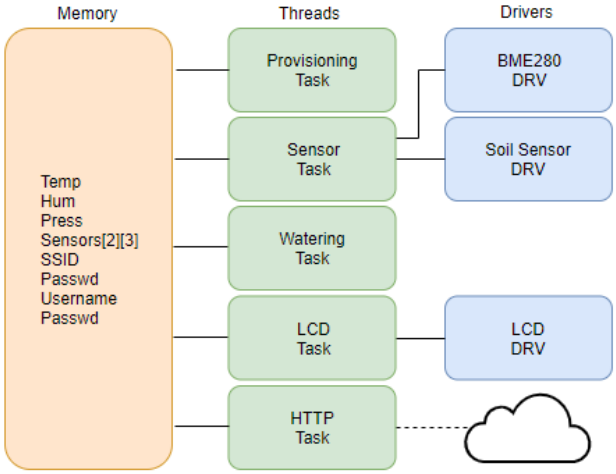
table diagrams, which are helpful in understanding the logical layout of the database.

The web server software will control the flow of communication from the controller to the database, and from the web application to the database. Very often a web server will provide data which is used in building a web page. However, to simplify development we decided that the web server would mostly be processing REST requests. This way it will be easier to develop each integration independently. Because the web server will not provide any HTML, we will be able to use the same web application interface for both the SGC and the web application. The web server will also host the web application which will contain HTML, JavaScript, and CSS code. However, the way the setup works will allow for almost complete independence from the java code and the rest of the code used by the single page application.

Bootstrap will be used to simplify the styling of the webpage. Bootstrap is a free and open-source front-end framework. Bootstrap provides templates of how UI elements should look. This way not a lot of time will be needed for setting up the layout and looks of the web application. Angular JS, a framework for dynamic web applications, will be used to program the logic of the web application, also known as the front-end. This framework takes away a lot of the steps which would be necessary to link the logic to the UI. This way, focus can be kept on the programming rather than on setting up and linking the different parts of the front-end.

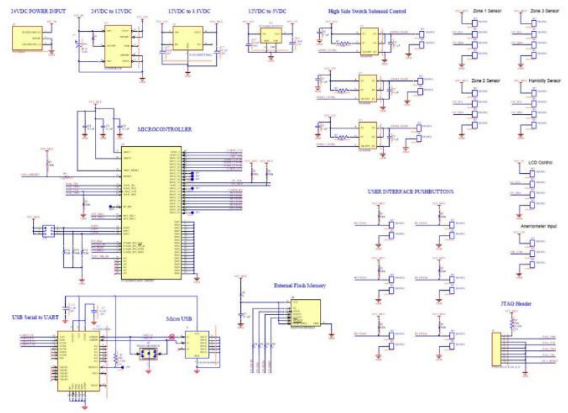
The MCU software will control all the physical hardware involved in the smart garden controller. This software will communicate with the sensors to get metrics to upload to the web server. It will control the valves which will let water through to water the plants. It will also interact with the user via push buttons and an LCD. Therefore, in a way this software will be the brain of the device. This software alone with the hardware will be enough to have the device working and watering the. However, it will still need the web server for the scheduling.

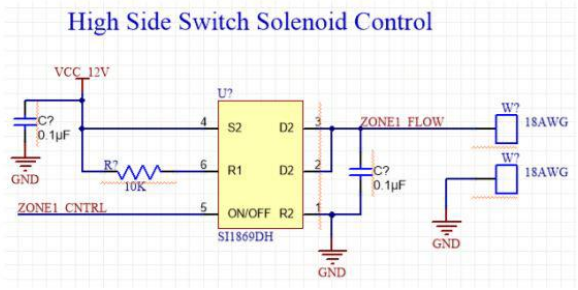
Fig. 5 MCU Software



The MCU software was all written in C. The components chosen for this project did not have drivers available for the real time operating system chosen. However, the same components had drivers available for Arduino in C++. Therefore, the group had to read the code for the drivers in C++ for Arduino and recode them to work with our platform. Re-writing these drivers took more time than expected. Drivers were written for the LCD, BME sensor and the soil moisture sensor as well.

VI. Prototype Schematics





ACKNOWLEDGEMENT

The authors wish to acknowledge the assistance of Professors Zakhia Abichar, Suboh Suboh, & Junjian Qi on the Review Board Committee.

BIOGRAPHY

Temple Corson IV will graduate with a Bachelor of Science in Computer Engineering in December 2018. He aspires to work in the fields of Software Engineering or Information Technology with DropLit.io or Publix Supermarkets, respectively.

Renan Coelho Silva is a senior Computer Engineering student at the University of Central Florida. Renan works as a Staff Support Engineer for a software as a service provider (SaaS). Renan's days consist of reviewing code and analyzing and resolving problems. Renan's goal is to eventually start working with hardware as well, in addition to software.

Geovanny Chirino is a senior Electrical Engineering student at the University of Central Florida. His interest in electronics began during high school, while enrolled in a 4-year avionics program. Current interests are 3D printing and power/renewable energy. He hopes to further work with 3D printing and explore the possibilities the technology has to offer.

Alexander Burns will graduate with a Bachelors in Electrical Engineering. He is currently working for a product development company in Melbourne Florida and plans to continue his career there for the foreseeable future. His interest in engineering was driven by his family and their involvement in the space program.

REFERENCES*

[1]http://batteryuniversity.com/index.php/learn/article/secondary_batteries
[2] http://i2c.info/i2c-bus-specification
[3]http://i2c.info/i2c-bus-specification
[4]http://ww1.microchip.com/downloads/en/DeviceDoc/70005304B.pdf
[5]http://www.aoml.noaa.gov/hrd/tcfaq/A5.html
[6]http://www.cablefree.net/wireless-technology/history-of-wifi-technology/
[7]http://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/
[8]http://www.circuitstoday.com/pcb-manufacturing-process
[9]http://www.csd.uoc.gr/~hy428/reading/i2c_spec.pdf
[10]http://www.electronicdesign.com/4g/talk-multiple-devices-one-uart
[11]http://www.futureelectronics.com/en/microcontrollers/microcontrollers.aspx
[12]http://www.hydraulicspneumatics.com/200/TechZone/HydraulicValv es/Article/False/6409/TechZone-HydraulicValves
[13]http://www.informit.com/articles/article.aspx?p=23760&seqNum=3
[14]http://www.ocfl.net/?tabid=371#.Wz59F9JKiUk
[15]http://www.oracle.com/technetwork/java/codeconventions150003.pdf

*Full list of references and permissions available in Final Project Documentation.