MOIST
(Mesh-Operated Irrigation Sensor Technology)

Department of Electrical Engineering and Computer Science

University of Central Florida

Group G

Fall 2018

Ahmed Hamdy	ahamdy1995@gmail.com	EE, CS minor
Pierre Elange	elange.pierre@knights.ucf.edu	EE
Gabriel Santos	gabetsantos@knights.ucf.edu	CpE

Documentation Contents:

Figure List:

Table List:

Code List:

1.0 Executive Summary

At the forefront of today's standard irrigation systems for open fields lies the sprinkler system: a both labor and time-efficient group of devices that waters grass and plants on its own without the need of human supervision, for the most part. As technology progresses, new ways to accomplish just about any task are constantly discovered. MOIST (Mesh-Operated Irrigation Sensor Technology) focuses on innovating the modern irrigation system. The format in which most sprinklers currently operate has its advantages in terms of autonomy, but has major repercussions in terms of efficiency and power consumption.

According to the EPA (Environmental Protection Agency), around half of the water currently used to water plants and grass outdoors is wasted because of evaporation, wind, or overwatering. MOIST aims to be vastly more efficient than current sprinkler systems due to its ability to respond to changes in moisture in the grass it is overseeing. Sensors will be placed throughout the plane of grass and will be distributed evenly enough so that moisture in all sections of the grass is accounted for. Instead of the grass being watered for the same amount of time every week like they currently are with sprinklers, the sensors will give out a moisture reading on an hourly basis stating whether each area of the grass has consumed enough water for the day or not. If the moisture readings are too low, the network of sensors works together to send out a signal detailing what section of the field needs to be watered and how much watering the section needs based on the moisture level.

We previously worked with a company that has a sensor built with this functionality in mind. MOIST focuses on implementing this sensor into a mesh network that will tell an autonomous watering vehicle what areas of a grass field needs watering. Our previous sponsor intends to, one day, have their technology efficiently watering large areas of grass, such as golf courses and commercial real estate. The area of study for this project, however, will be on a 20 by 30 feet backyard area comprised of grass native to the average backyard of a residential home in Florida for the sake of time and cost efficiency. With that being said our research and development of this technology will take it beyond everyday irrigation process and impact the environment in a positive manner.

With the evolution of the Internet of Things and Smart Automation, current irrigation systems is one more application most landscapes use today that require human supervision unnecessarily, but with implementation of MOIST irrigation will result in less energy consumption, less waste of resources, and less costs overall, making it a solution that will surely become more popular as such systems become more commercialized and prominent in the irrigation market.

Although MOIST is considered a stand-alone project, it is also part of a three-group interdisciplinary project consisting of two teams of computer engineering and electrical engineering students, and one mechanical engineering group. The other groups will oversee creating and commanding a watering robot that will water different areas of grass at a time based on the readings that MOIST sensors relay out. This system was designed to be a solution for our previous sponsor’s Interdisciplinary project, and double up as an add-on to common sprinkler systems. With considerations taken, MOIST aims to be the best solution for the next generation of irrigation systems.

2.0 Project Description

As a soon to be graduate engineers, there are many motives to doing projects. Our collective interest in bettering our community brought us together to work on this project and resolve the given problem beyond the academic scope. With many arising issues in the twenty-first century, there can never be enough engineers to solve every problem. Yet when the situation arises we must work together, work hard, and work in a clever manner to collaborate and compromise for the best solution the team finds to be fit.

2.1 Problem Statement

This project began with a specific problem regarding the conventional irrigation systems. Ones based on underground piping, fixed sprinkler heads, and rudimentary controls. The issue lies with over spraying water, damages that can spew wasted water and, costs of such installation and maintenance. We needed a new-found solution to revoke these concerns, particularly a solution that avoids any of these issues and adds new features for further irrigation improvement.

2.2 Project Goal

Understanding our problem was the first step to our solution. We analyzed why water was being wasted, and our conclusion is that our yards are not being monitored appropriately. A form of a wireless self-powered sensor was needed, and that it was durable to withstand weather and other forms of damages. Many designs and variations were thought but our deciding factor relied on cost, maintenance, and easy installation.

This design of monitoring the yard had two parts, the multiple sensor nodes spread across the yard and a base station the acted as controller and a gateway for the user interface. This provided a sense of control, real-time status updates of the yard and a bridge to the Autonomous Irrigation Vehicle. This will allow the necessary watering as a next immediate step if need be. We plan to coordinate with Autonomous Irrigation Vehicle designed by another group to accomplish this. This design rids of unnecessary piping, and prevents damages of the non-existent sprinkler heads.

2.3 Motivation & Major Objectives

What sparked the interest of the students involved in this project are the career preparation skills that will be gained from solving problems that were faced in the realms of electrical and computer engineering using critical thinking. One of the students working on this project already works with plumbing at his current internship, and plans on implementing the technology produced in this project into his future endeavors to continue making the watering of plants a more efficient process. This project also provides plenty of opportunities for us to improve on concepts related to our majors, but that are barely touched upon or spoken of in our classes. These concepts include creating our powered devices and using a web application to track the status of the sensor network we will create.

The main goals that this project was set out to accomplish are enhancing an irrigation system to be more efficient than sprinklers, receiving real-life career experience by working as a group for two semesters, delegating different tasks between the group, and compromising on conflicting ideas, and getting hands-on experience creating our own PCBs and implementing hardware and software communication between devices in a network.

2.4 Requirement Specifications

Quantitative measurements for the preliminary requirements for the hardware in this project are estimates based on research that are likely to change as this project is further delved into. Sensor nodes should require no more than 10W to operate. System should be able to be remotely operated via web app. The field of grass that we will base our research and models on will be 30 X 20 ft. There should be at least ten different sections of the field that are separately accounted for, and at most 20. None of the node areas should overlap with each other.

These specifications remained the same as we entered and finished EEL 4915.

The minimum range between two communicating nodes should be no less than 15 ft. Max soil moisture that sensors can detect before the irrigation system is engaged is 20%. The irrigation system should be able to relay control for the duration of the watering. The minimum span of ground that the irrigation system should cover is 600 square feet. The irrigation nodes should be self-powered with minimum maintenance. The web-app should be able to map all nodes and display them in a GUI. The gateway should support the database of nodes and send signals for override control. The GUI should be able to display the status of a selected node and give an option for a manual control for the user.

Plenty of project constraints were discovered as more research and hands-on implementations were conducted. Refer to Table 1 for more details as to what exactly the constraints are and the requirements we need to achieve. The constraints for the irrigation system will vary depending on what hardware and software is ultimately used in the end. This is where prototyping comes into play. For now, the project constraints include the signal strength and speed of the hardware used to allow the mesh nodes to communicate with the central gateway. Also, what measurements can be taken by the sensors used to analyze the networks' actions. The range and accuracy of the sensors. As well as performance when more nodes are introduced.

Other constraints and requirements depended on the other two groups involved in this project. This includes the potential of the watering robot, as well as the range and duration that the robot can handle. Our biggest concern involving this multidisciplinary project is compatibility of information and how to appropriately manipulate the data for the other subsystems to make use with. By sketching diagrams and tables, we can convey useful information.

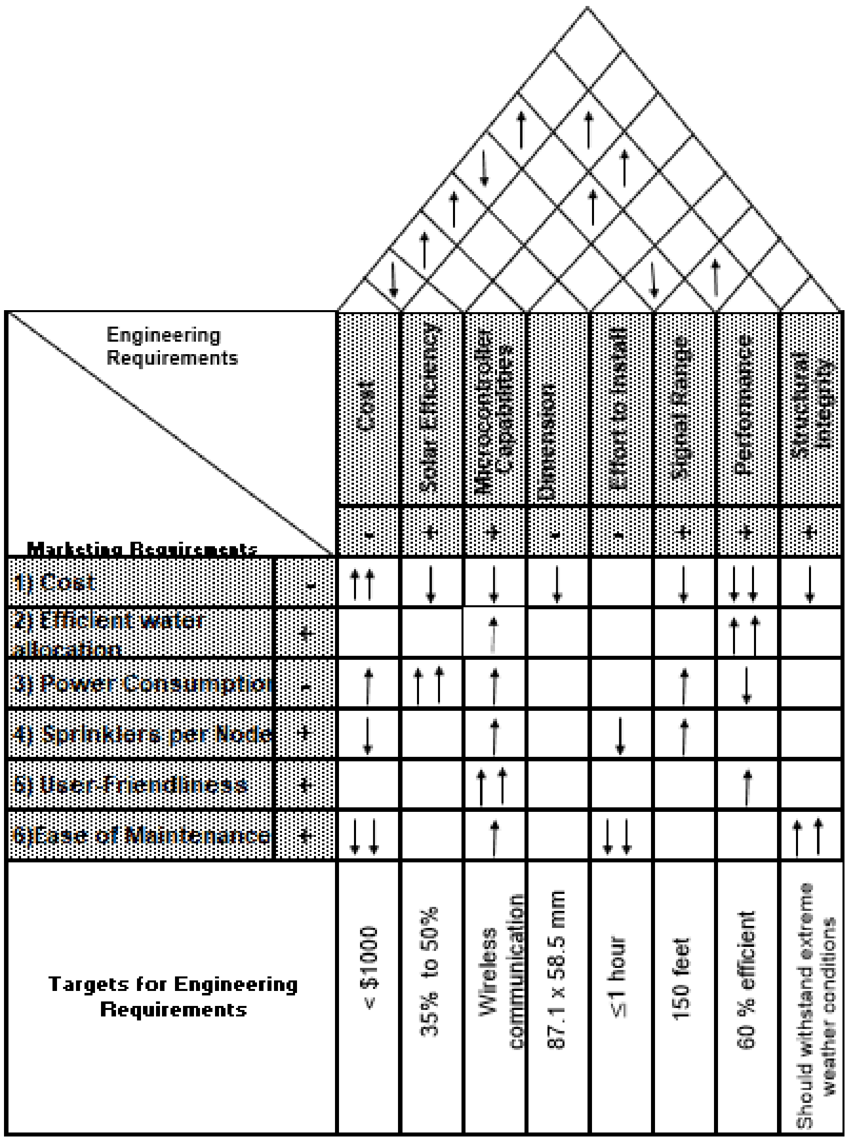
2.4.1 House of Quality

It was very important for us to understand what our client needs and how we can provide a satisfactory solution. Determining the problem is the first layer of problem solving, next is to break down the general problem into a series of smaller more specific sub-problems to analyze the components and how it contributes to the original problem. A Quality Function Deployment is a table designed for this scenario. It compares given solution requirements, usually stated by the client, and possible solution specifications, determined by the engineers at work. In our situation the sponsor gave our project requirements and as a team we came up with specification our design is projected to implement, all listed on Table 1.

After working on the House of Quality it is easier for us to understand the areas of concern and not get distracted by ideas that will not contribute to the solution. Quality Factor Deployment is an ideal form of planning and progressing our state in project design.

Referring to Table 1, there are six major criteria our client wants to accomplish and those are relating to initial cost, resource consumption, and ease of use. We then used the table to state our target of a system under $1000.00, solar-powered with efficiency of 35-50%. A system deploying a wireless communication with nodes of a small profile, 87.1 x 58.5 mm enclosure withstanding severe weather conditions. A signal strength of 150 ft., with %60 more effectiveness then a traditional sprinkler system. After initial planning and first phase designing, researching the right components is mandatory to achieve our target.

The House of Quality design was inspired by the design that was shown to us in a Senior Design I lecture. The roof was implemented along with the base of the House of Quality to display the relationships between the different components mentioned in the House of Quality. Two arrows up in one box means that there is a strong correlation between the two components that match up in that box. Two arrows down in one box means that there is a strong correlation between the two components that match up in the box. The roof allows the components in the top box to be compared to each other, rather than solely with the components in the lower left side.



Engineering Requirements / Marketing Requirements		Cost	Solar Efficiency	Microcontroller Capabilities	Dimension	Effort to Install	Signal Range	Performance	Structural Integrity
		-	+	+	-	-	+	+	+
1) Cost	-	↑↑	↓	↓	↓		↓	↓↓	↓
2) Efficient water allocation	+			↑				↑↑	
3) Power Consumptio	-	↑	↑↑	↑			↑	↓	
4) Sprinklers per Node	+	↓		↑		↓	↑		
5) User-Friendliness	+			↑↑				↑	
6)Ease of Maintenance	+	↓↓		↑		↓↓			↑↑
Targets for Engineering Requirements		< $1000	35% to 50%	Wireless communication	87.1 x 58.5 mm	≤1 hour	150 feet	60 % efficient	Should withstand extreme weather conditions

Table 1- House of Quality

3.0 Research

To solve a problem, one must first understand a problem. Design starts with research, hence the acronym R&D. Our steps in research, which were simple but repetitive, consisted of analyzing potential sub-systems and understanding it's implementation. For our system we browsed available systems from local off the shelf product and high-tech exclusive system to past accomplished senior designs

3.1 Existing Designs

There have been several technologies created that perform similar functions to the products that will be used in our senior design project. This section will discuss the different technologies that have been researched that provided insight and inspiration for the mesh network. Another aspect of existing designs that was explored are UCF senior design projects that have been completed in the past. There are two in particular, the FARM project and the Triton sensors that the last group sponsored by Guard Dog Valves worked on. These projects involve products and methods that have guided the designs of our mesh network.

3.1.1 Commercial Solutions

Comparative study of commercial solutions has been carried out for small scale farms. The solutions that incorporate sensors do not include systems based on fuzzy logic which allow to establish the watering quantities in a precise way. Aifro WaterEco [1] considers climatology in order to lower or increase irrigation but it is focused on the definition of threshold values and does not include fuzzy logic or sensors, such as soil and land humidity. Blossom [2], encompasses crop irrigation and generation of calendars, depending on the climate these calendars can be edited manually, it has common functionalities but allows for remote management, it also does not include fuzzy logic in its behavior. BlueSpray [3] includes seasonal information to adjust irrigation as in the previous example, it does not include fuzzy logic based behavior. GreenIQ [4] and IrrigationCaddy [5] are conventional programs that can be managed remotely from mobile applications and include the feature of creating irrigation calendars. Lono [6] incorporates threshold values and seasonal information and reduces crop watering according to the thresholds, as in the previous cases it does not include fuzzy logic and does not have weather sensors. On the other hand, the Orbit B-Hyve [7] system incorporates a control through smartphones that is able to change some parameters in order to edit the system schedule. The parameters that device takes into account when configuring the irrigation timer are: the slope of the site, the soil type, if it is in the sun or shade, history of rainfall in the area and the current weather. The Rachio Smart Sprinkler Controller [8] system also has a Wi-Fi connection and is able to send the data from the sensor to the user's smartphone. This device requires an initial configuration which is established by indicating the type of crop and the type of soil. In this way, the system can estimate the irrigation time required by the crop. The fuzzy system is not applied, nor are the flexible rules. Rainmachine [9] is another commercial system which incorporates an automatic irrigation program. It is capable of calculating the percentage of evaporation and transpiration of the soil, according to the weather conditions obtained from the data of the meteorological service. This system, like the others, does not include fuzzy knowledge. The Spruce irrigation [10] system combines the information obtained from all the temperature and humidity sensors and rainfall forecasts. Lastly, we list the Rain Commander [11] system for its ease of use and its integration with mobile devices for remote irrigation control. However, this system lacks an intelligent configuration, it has no fuzzy logic rules, and only considers the schedule and the irrigation time that has been configured manually by the user.

3.1.2 Senior Design Projects

Past senior design projects like F.A.R.M. (Fundamental Agriculture Resource Monitor) [12] offered a low powered, wireless, plant monitoring system. Although it achieves its goal, its design concentrated more on studying the plant for gardens and pots to print out reports for its users. Our system needed to achieve that key-point in addition to automated irrigation aimed for scalable landscape while withstanding weather climates and prone accidents. It is the project that we based the concept of MOIST on, with a slight variation of focus on the sensors that have already been developed by Guard Dog Valves, rather than create our own or buy other existing ones. Also, our project focuses less on providing a farmer or gardener with various conditions that their plants are experiencing, and hones down specifically on the moisture level of plants. However, with the loss of generality this project undertakes compared to F.A.R.M., it makes up for in the improvement of water efficiency for grass. MOIST only focuses on water moisture in grass, but the involvement with a mesh network to send the reading levels of each node in a field’s moisture to a central hub and have only these specific areas watered is what makes this project the next step toward extreme water efficiency in the future. In addition, this project allows us to only water different areas of grass for as long as the node has a low moisture level. This feature would allow gardeners using the F.A.R.M. system who don’t have too much time on their hands to have their plants watered for them for the right amount of time and with the right amount of water while they are away from their gardens.

Irrigated Water Monitoring System [13], a project previously sponsored by Guard Dog Valves was aimed for this purpose. Initially presented by the UCF Department of Mechanical and Aerospace Engineering focused on the material of the sensor probe. Their design research involved extensive testing of possible material used for an anode and a cathode of the device. The study and design included a correlating resistive value and moisture level. Our intent is to take this designed sensor and improve on the electronics and network. Thus, helping our previous sponsor obtain a state-of-the-art irrigation system.

3.2 Strategic Components for Hardware

Considering previous designs, we came up with list of major components of a node we needed to further investigate. This breaks down the design into smaller sections of design that all must be analyzed to integrate appropriately. These sub-parts include the following: A sustainable power system, that does not need constant maintenance; a central controller dedicated to keep the operations and program running; a communication system, to link with the network; a sensor for reading measurements; and an enclosure to house all of this. This is a single PCB called a node is a design produced in scale. Figure 1 details the individual parts and their links

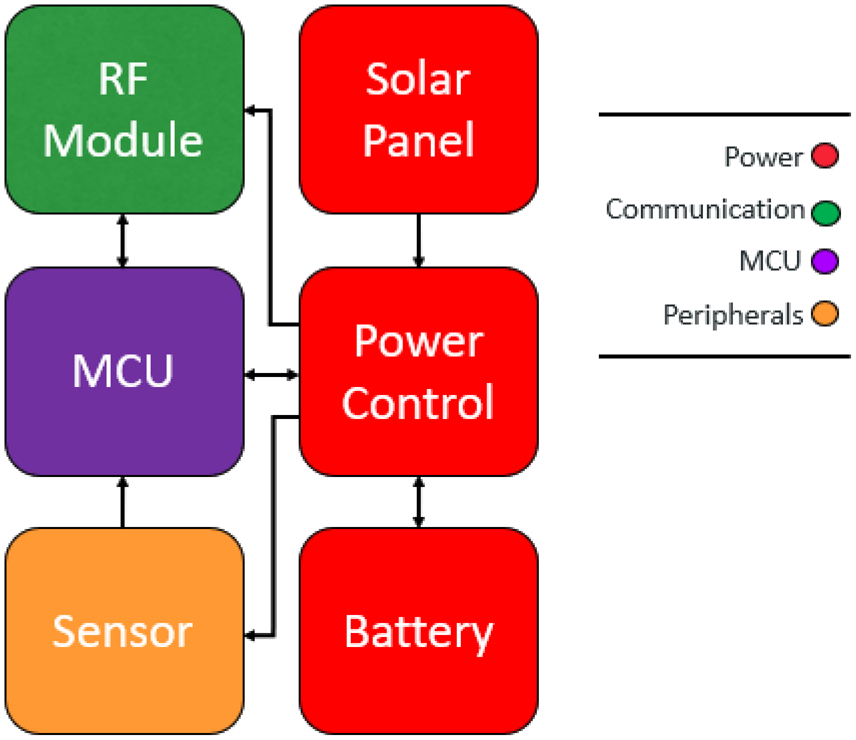


Figure 1- Node Block Diagram

The second set of components include an embedded computer used as a server, that hosts a web-application, the database, and all of software needed to start the network. This gateway called a hub has a designed PCB stacked onto it which involves a communication system, and possible solenoid control system for the hose of the Autonomous Irrigation Vehicle. This Gateway is the head link between the nodes so only one per network is needed. Figure 2 represents the hardware and software needed for a Gateway.

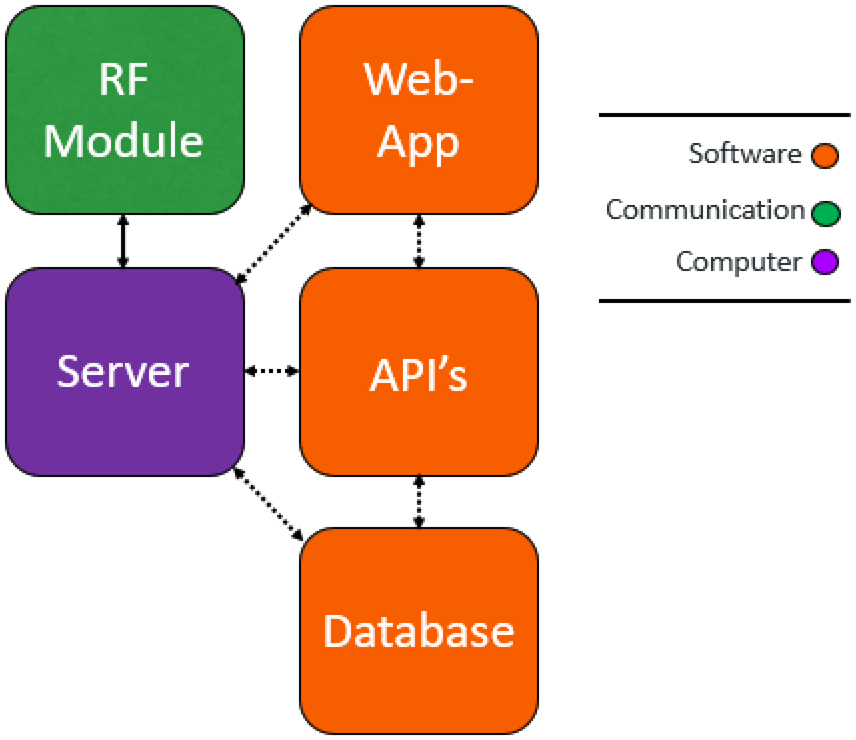


Figure 2- Gateway Block Diagram

This is a potential way of designing our solution, understanding the necessary components and its function lays out the design for us, our client, and others in a detailed manner. This our final design and abstraction for the project. The potential for solenoids was removed since there was no need for it in our project.

3.2.1 Solar Panel & Battery

In order to maintain the functionality of our project, an independent power system is required to supply the main key components of our system such as: solar panel which is comprised of solar cell that convert light energy to electrical energy; charge controller that threshold the output voltage from the solar panel going into the battery; the voltage regulator that regulate to a non-changing voltage; the battery that provide power directly to the components.

Solar panel is available in three types (monocrystalline, polycrystalline, and amorphous). Between the three-solar panel technology, amorphous is the cheapest but also the least efficient; the polycrystalline combine efficiency and cost which is the best choice for our project because of the high cost of monocrystalline. Because of the portability of the project the parameters for choice of the battery are crucial such as size, voltage, spillage free, and environment friendly.

We ultimately decided to go with a polycrystalline solar panel.

3.2.1.1 NUZAMAS 2W

NUZAMAS provides 2W at a 12V output. It is a poly-crystalline solar panel. It is 17% efficient in power retention, and resistant to corrosion and humidity, which is ideal for our sensors out in the field.

3.2.1.2 Customized 18650

This is a Li-Ion rechargeable battery meant to power small devices. Has a nominal capacity of 7.4V with a charge capacity of 1800mAh with a cut-off of 6.0 volt. The 18650 will provide the power to the system which require no more than 3.3 volt for the microcontroller. The 46.5 gram weight, the 18 mm diameter, and 65 mm height dimension of the battery make it very suitable for the portability of the project. The discharge capacity of the cell with 1.3 amperes down to 3.0 volt is 1 hour which in our case might be more than 3 hours because of our low power system design. In order to mitigate any power conundrum we have decided to pair. the battery with a solar panel which constitutes as a charger for the battery.

3.2.2 MCU

When designing any embedded device, some form of control is required to operate the hardware, this is where the MCU (Microcontroller Unit) comes to play. Its sole purpose is to be programmed with the right algorithm to coordinate the process of its peripherals, it is the connection between the sensor and RF module, as well as monitoring the battery level. Most MCUs have few dedicated pins used for programming the chip through a programmer and an associated IDE for writing the code.

3.2.2.1 Microchip PIC16F15354

PIC16 family is one of the simplest existing microcontroller available with 8-bit data word and 14-bit instruction word of. Only costing under a dollar it is fundamentally a straight forward mid-range component. This chip offers 28 pins. This device stores up to 14 KB of program memory and 1 KB of data. At 32 MHz clock and a voltage range from 2.3-5.5V, this chip dissipates a max power of 800mW.

PIC16 is similar to the RISC architecture and based on Harvard architecture. This device is a candidate for our design

3.2.2.2 TI MSP430

The MSP430 was introduced to us here at UCF in a programming course. The benefits of using this device is the familiarity working with the device in C-Programming or Assembly. MSP430 comes in a development board that has integrated peripherals and a programmer.

3.2.2.3 Parallax Basic Stamp 2

The Basic Stamp 2 was introduced in an intro to engineering course here at UCF. The main difference between this device and the rest is that instead of it being a single chip, it is a small module with few components. The BASIC Stamp 2 uses a variant of BASIC programming called PBASIC, it is a simple language used but lacks the power of C-programming.

3.2.2.4 Atmel ATmega328P-PU

This chip is most commonly used microcontroller whether it’s users realize this or not. It is found on the Arduino Uno hardware. With easy to use interface and pin headers for accessing the MCU pins. This is a candidate to consider. The supported IDE runs a variation of C++ meant for embedded systems. With low cost and adequate power consumption it evenly compares to other MCUs.

We decided to go with the Atmel ATmega328P-PU

3.2.3 RF Module

It was difficult to foresee all potential IoT applications having in mind the development of technology and the diverse needs of potential users. The IoT applications are addressing the societal needs and the advancements to enabling technologies such as electronics and cyber-physical systems continue to be challenged by a variety of technical (i.e., scientific and engineering), institutional, and economical issues. Present communication technologies span the globe in wireless and wired networks and support global communication by globally-accepted communication standards. The Internet of Things Strategic Research and Innovation Agenda (SRIA) intends to lay the foundations for the Internet of Things to be developed by research through to the end of this decade and for subsequent innovations to be realized even after this research period. Within this time frame the number of connected devices, their features, their distribution and implied communication requirements will develop; as will the communication infrastructure and the networks being used. Everything will change significantly. Internet of Things devices will be contributing to and strongly driving this development. Changes will first be embedded in given communication standards and networks and subsequently in the communication and network structures defined by these standards. Data management is a crucial aspect in the Internet of Things. When considering a world of objects interconnected and constantly exchanging all types of information, the volume of the generated data and the processes involved in the handling of those data become critical. A long-term opportunity for wireless communications chip makers is the rise of Machine-to-Machine (M2M) computing, which one of the enabling technologies for Internet of Things. This technology spans a broad range of applications. While there is consensus that M2M is a promising pocket of growth, analyst estimates on the size of the opportunity diverge by a factor of four. Conservative estimates assume roughly 80 million to 90 million M2M units would be sold in 2014, whereas more optimistic projections forecast sales of 300 million units. According to strategy Analytic the global M2M industry size is forecast to grow to around 200 billion us dollar in revenue by 2020[16]. Based on historical analyses of adoption curves for similar disruptive technologies, such as portable MP3 players and anti-lock braking systems for cars, it is believed that unit sales in M2M could rise by as much as a factor of ten over the next five years. There are many technologies and factors involved in the “data management” within the IoT context. Some of the most relevant concepts which enable us to understand the challenges and opportunities of data management are:

• Data Collection and Analysis

• Big Data

• Semantic Sensor Networking

• Virtual Sensors

• Complex Event Processing.

Based on our research we are going to present different wireless protocols and standard for possible used in our project in their prior application in other project. This strategic component is the sole piece that allows the singular device to connect to the rest of the network giving it the capability of exchanging data wirelessly. With that being said there are two main standards to consider that dictates the protocol: IEEE 802.11, and IEEE 802.15.4.

3.2.3.1 Wi-Fi

Wi-Fi is the most common wireless communication used. Primarily used to connect a device to the internet giving it access to the web. Although Wi-Fi is an option there are many underlying issues with the using this protocol. It can quickly crowd an existing network, constricting the bandwidth and slowing internet speed. Neither does it not support a mesh network setup, forcing limited range without scalability, and processing on the gateway. Out all possible RF protocol this is the only one dictated by the IEEE 802.11 standard. Wi-Fi has a suitable communication distance of about 20 m and 100 m in indoor and outdoor environments, respectively. In PA applications, Wi-Fi extends diverse architectures by connecting several types of devices via an ad hoc network. Wi-Fi and 3G wireless technologies were employed in to estimate the agricultural applications of mobile phones. Remote access and short message services have Given that Wi-Fi requires much power, long communication time, and huge data payload. Wi-Fi technology is not preferable for our project in spite of the fact that a Wi-Fi server prevents data losses by adopting data redundancy techniques. In addition, Wi-Fi cannot be employed for multi-hope applications and influenced by number of users and the signal intensity; thereby, it is inappropriate for WSNs. Moreover, the W-Fi nodes listen all the time, so power consumption will increase.

3.2.3.2 Bluetooth 5.0

This protocol also runs on the 2.4 GHz bandwidth, but is very low power and limited range. With the latest version 5.0 Bluetooth now supports a mesh setup for local devices. Bluetooth 5.0 is the newest technology and therefore not a lot of support and hardware are readily available, plus with no previous experience between our group members with this technology makes this choice a questionable candidate. Bluetooth (BT) Wireless Protocol The BT standard has been utilized to establish a communication link between movable and portable devices, such as laptops, over a short distance of up to 10 m. Given its pervasive nature and availability in most mobile devices, BT has been employed to satisfy multilevel agricultural requirements. Weather information, soil moisture, sprinkler position, and temperature are monitored remotely using Global Positioning System (GPS) and BT technologies. Bluetooth (BT) Wireless Protocol The BT standard has been utilized to establish a communication link between movable and portable devices, such as laptops, over a short distance of up to 10 m. Given its pervasive nature

3.2.3.3 ZigBee

ZigBee also runs on the 2.4 GHz bandwidth but is dictated by the IEEE 802.15.4 standard. ZigBee is viable candidate with long ranges, low power consumption and mesh capability. The is a lot support for this hardware, the only downside is the cost of the device and since ZigBee is a proprietary protocol, a paid license must be acquired to use commercially. ZigBee wireless protocol is considered one of the best candidate technologies for the agriculture and farming domains. Given its low duty cycle, ZigBee is considered appropriate for PA applications, such as irrigation supervision, water quality management, and fertilizer and pesticide control, all of which require a cyclic information update. Based on this technology, the sensor nodes in the agricultural field can communicate with the router or coordinator node over a long range (i.e., 100 m) when XBee Series 2 is used. ZigBee can also reduce the communication distance by up to 30 m for indoor conditions.

3.2.3.4 Nordic nRF24L01+

This leaves us with our last candidate, the nRF24L01. Similar to ZigBee, nRF24L01 supports long range mesh network, with low cost and many variations from different vendors makes this choice optimal. Abiding by the IEEE 802.15.4 standards makes this protocol widely available with existing libraries from supporting communities. The ease of integration of this device is very beneficial for what we aim to accomplish with our expertise.

3.2.3.5 NB-IoT

Narrowband-IoT (NB-IoT) is a new IoT system constructed from current Long Term Evolution (LTE) functionalities. Subsequently, NB-IoT is possible to share the spectrum of LTE without coexistence problems and to utilize the same pieces of equipment, as well as it is possible to connect seamlessly into the LTE main network. This permits all network facilities such as security, tracking, policy, charging, and authentication to be totally supported. The design goals of NB-IoT cover high coverage area, extended battery life (i.e., 10 years), high network size (52,000 devices/channel/cell), and low-cost devices.

3.2.3.6 LoRa

Long Range Radio (LoRa) Protocol is introduced by the LoRa Alliance as a protocol stack for low power and wide area Internet of Things (IoT) communication technologies that are associated with indoor transmission. The basic network architecture of a LoRaWAN consists of LoRa end devices, LoRa gateways, and a LoRa network server. The LoRa end device communicates with gateways that employ LoRa with LoRaWAN. LoRa gateways pass raw LoRaWAN packets from the end devices to a LoRa network server with a high throughput based on the backhaul interface, which is typically 3G or Ethernet. Accordingly, LoRa gateways act as a bidirectional communication or protocol adapter with the LoRa network server. In this case, the LoRa network server takes charge of decoding the data packets transmitted by the LoRa devices and creating the frames that would be directed back to the devices. LoRa offers a bidirectional solution that matches machine-to-machine (M2M) Wi-Fi or cellular technologies. LoRa presents a cost-effective method for linking batteries or mobile devices to the network or to end devices.

We decided to go with the nRF24L01+ for our communication between the nodes and the gateway.

3.2.4 Geolocation

Our system is based on real world positioning and identification of these Nodes, to map our network and to relay accurate information to the Autonomous Irrigation Vehicle. The Nodes must have a way to retrieve that data without too much hassle during installation and maintenance. Geolocation serves this purpose, for our project. There are hundreds of satellites orbiting Earth at any given time, and few are dedicated to geolocating and are accessible globally.

3.2.4.1 Bearing Measurement System

This system uses a directed radio antenna to locate a radio fix, a transmitted signal from a fixed position. This requires a minimum of two transmission to find the intersection of the radio waves. The disadvantage of this system is the size of the required antenna, although accurate this is more implemented for buildings and stations of that size. The antenna must constantly move to stay connected, but for our system we need something lightweight and stationary.

3.2.4.2 Beam System

Beam system as also an old technology that requires a dedicated aircraft to broadcast a narrow beam. The aircraft created the airway and uplifted the load on the receiver, and as long as the receiver was in sync the airway would hop stations to keep the airway stable. The disadvantage of this system is the required aircraft needed for this, and if the receiver was outside the range of the beams transmitted, the navigation was no longer maintained.

3.2.4.3 Transponder System

With any two measurements of angle or distance this system determines the position with high accuracy. Radar has been introduced since the World War era and used this type of technology for geolocation. Since then aviation are the primary users. With dedicated towers for radars that sweep a wide region, and Time of Flight technique, this system can detect any object resonating in the radars frequency band. The downside of this system is the cost and the need for a radar.

3.2.4.4 Hyperbolic System

A modification of the Transponder System, this system is based on fixed hyperbolic lines, hence the name, charted down. Two measurements produce a fix on location. These systems eliminate the need of an airborne transponder and therefore eliminating manual triangulation that the Transponder system uses. By using Time Difference of Flight from known stations, position is found along the charted lines. More stations can receive this signal, improving the accuracy of the lines and eventually fixing in on a point.

3.2.4.5 Satellite System

This system also known as GPS (Global Positioning System) is also a form of a Transponder System, that supersedes it by replacing the airborne transponder with a space satellite, this what most of the world uses as their geolocating technology. The satellite works in conjunction with few others and each transmit an ephemeris data, an accurate calculation of its location as well as a clock signal dictated by an onboard atomic clock all to the receiver. The receiver then takes this information to triangulate its position. This system is the most accurate there form of geolocating with the push of technology development and marketing. This is the cheapest solution for geolocating.

We ultimately settled on the satellite system due to our network being mainly used outdoors.

3.2.5 Sensor

There is a wide variation of sensors available for many purposes. The kind of sensor we need is one that is capable of measuring the soil moisture. For this strategic component allows us to give feedback to our users for irrigation purpose, potentially saving water, thus saving the agriculture, and finally saving cost.

3.2.5.1 Triton

This is the sensor developed by the Integrated Water Monitoring System project. It entails what we need for our system and is a viable candidate since it comes from our sponsor. The measured data is in the form of resistance making the value universal compared to other sensors. Since it is not a predetermined scale testing the accuracy is in our favor.

3.2.5.2 VH400

The VH400 is a moisture sensor from Vegetronics that we decided to go with for our nodes. It is a little more expensive than your average moisture sensor at over forty dollars per sensor, but in return it offered us a precise measurement between areas that are wet and dry.

3.2.6 Central Hub

The final strategic component is the central hub, this is the origin of our network and thus a vital component for relaying data back to our user. The central hub consists of a server with a RF module stacked. Selecting the right CPU is crucial for our network speed and internet connectivity

3.2.6.1 Pine64

A 64-bit CPU capable of up to 512 MB of RAM and expandable MicroSD Card slot up to 256 GB makes this near-unbeatable candidate. Supporting a USB 2.0 ports and a Gigabit Ethernet port makes certain that the internet speed is in par with our system. OS supported includes various Linux distribution at a cost of only $15.00. With this new of a technology, widespread support is readily available.

3.2.6.2 Raspberry Pi

At a price of $40.00, Raspberry Pi is the most known single board credit card sized computer today. It supports 512 MB of RAM, USB ports, and a 10/100 Ethernet controller. Raspberry Pi is aimed for hobbyists and supports a resourceful community. With our group having access to one and experience on the Linux OS makes the Raspberry Pi a suitable candidate.

3.2.6.2 Raspberry Pi

We also considered using the Banana Pi M3 in EEL4915, which offers similar components as the other two choices. We decided to go with the Raspberry Pi because of our familiarity with it already and the amount of open source documentation available online to help us use it to control our mesh network.

3.3 Strategic Designs for Software

Just as important as the technology used to power the mesh network system is the software integrated inside all of the hardware parts that governs the behavior that the mesh network will take on and how it will respond to user interactions. Various software options exist for the number of operations that the web application is capable of carrying out. With these various options comes different programming languages, IDE’s, and website developers that can be used to produce the most functional software results possible. In order to have made our final decisions on the software that was used in this project, several factors were considered heavily and equally, such as ease of use by group members, the functionality that it brings into the project, the cost worthiness, and whether the software used assists our group with potential jobs and schoolwork assignments in the future.

3.3.1 Programming Languages

Throughout the undergraduate experience of the members of this group, several programming languages have been used to code different types of functions, games, programs, and applications. The choice made for what programming language will be used in this project largely depends on what languages we can master and implement into our project the most compatibility.

3.3.1.1 Java

Java is a language that only the sole computer engineering student of the group behind MOIST has proficient experience in. Java is the main code behind Android Studio and would be beneficial should our group decide to extend our web application functionality into a mobile app.

3.3.1.2 C Code/C++

All members of the group working on this project have worked with C in at least a couple classes throughout our undergraduate experiences. C can be used to code the TI MSP430 and other hardware devices that we are interested in using. This programming language was used in this project due to the hardware devices that can be programmed via C code. Working with C to make MOIST functional prepared the students working on the software side of this project for possible projects requiring the use and knowledge of C in their future careers and endeavors.

3.3.1.3 Assembly

Assembly is a rather complicated coding language, compared to the other coding languages featured in this report, that all members of this group have prior experience with. Assembly was used in a couple of our core engineering classes program a TI MSP430 alongside the use of C code. Although a more complex code than C in terms of not being a high-level language, Assembly offers plenty of power and support. This language is used heavily alongside MSP430's and can be used to have this specific hardware device perform a myriad of tasks, such as temperature sensor data collection and displaying different characters and patterns on an LCD screen connected to an MSP430.

3.3.1.4 HTML Languages

The coding for the web application that was used to monitor and send commands to the different nodes within the mesh network proved to be some of the most complicated coding that the students of this group have been involved with. This is due to the lack of experience our group had in web development and in having data be constantly accessed and altered in a database that can have data stored in it via the commands of the web application and changes in moisture level readings of the sensors.

The main web development programming languages that we considered using are JavaScript and PHP incorporated with HTML coding. Both languages offered opportunities to make a presentable web application that can be monitored by all members of this group with some learning and practice put into place. JavaScript ended up being used as an assistant in styling the web application and running commands on through it as well.

3.3.1.5 Python

Python is considered a more modern programming language than most of the other languages being considered for use in this project. Python was beneficial to the results of this project in that it possesses the capability to perform powerful functions and data interpretations based on a relatively small amount of code and coding experience. For instance, in one of the students working in this group's classes based on robotics, this student was able to scan multiple pictures of a single object taken from various angles and find perfect matches between these pictures, despite these pictures being scattered within pictures of several different types of objects. This scan match was performed using only a maximum of twenty lines of code which is minimal considering the multiple projects that the computer engineering student in this group has had to write hundreds of lines of code for when programming in Java. This technology could potentially be used in conjunction with the other EE/CpE group working on the interdisciplinary portion of our project to detect sensors from different angles when the AIV approaches a node. Figure 3 shows different angles of one object that the AIV would be able to recognize as the same object from using Python coding.

Python ended up being used to generate a graph of the node coordinates on the web application below the sensor data table. Looking at this graph, MOIST users could get a more direct interpretation of what nodes in their system were picking up specific data.

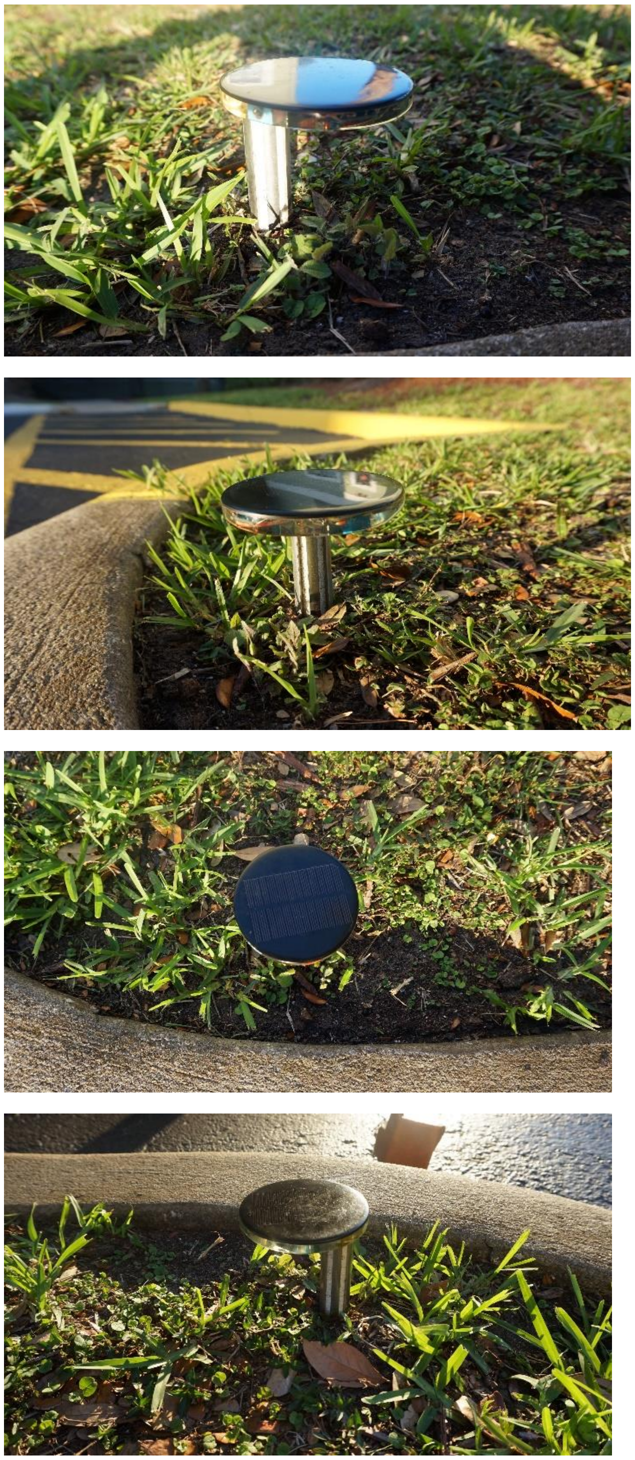


Figure 3- Python Image Angle Recognition

3.3.2 Integrated Development Environments

A vital part of the software that was used to govern our mesh network includes the IDE’s that were used to program the hardware involved in our project. There are several options for IDE’s in terms of what type of programming languages we wanted to code in, and what capabilities each coding environment contributed to the functionality of this project.

3.3.2.1 Android Studio

Android Studio provides a user-friendly interface for coding a mobile application on an Android device. Using the layout template, users can drag and drop UI features, such as text boxes and buttons, into a screen and toggle the UI settings to have the application function as the user sees fit. An emulator projecting the features of an actual Android mobile device of the user’s choosing can then be displayed in conjunction with the IDE to display how the designs constructed inside Android Studio would actually appear and function when accessed on a mobile device. The benefits that come along with using Android Studio include the familiarity that the computer engineering student working on this project already has with this IDE in one of his programming classes this semester, and also the amount of community support that already exists for Android Studio due to the popularity of Android mobile devices throughout the world. The Java code below written in Android Studio returns a screen on an Android app with the ability to send information to a database written in a textbox with the tap of a button.

```
public class MainActivity extends AppCompatActivity {
  public static final String EXTRA_MESSAGE =
"com.example.MOIST.MESSAGE";
  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
  }

  /** Called when the user taps the Send button */
  public void sendMessage(View view) {
 Intent intent = new Intent(this, DisplayMessageActivity.class);
     EditText editText = (EditText) findViewById(R.id.editText);
     String message = editText.getText().toString();
     intent.putExtra(EXTRA_MESSAGE, message);
```

```
        startActivity(intent);
    }
}
```

Code 1- Java Send Function for Mobile Application

3.3.2.2 Microsoft Visual Studio

Microsoft Visual Studio is the IDE that the members of the group working on this project are the least familiar with. This IDE can be used in all portions of the software side of our project, including web services and web applications. The main reason that this IDE may be implemented into our project is its compatibility with NativeScript. With a download of all the features that NativeScript offers comes along the ability to write mobile application codes in Visual Studio as well. There are commands preinstalled into Visual Studio that can open, run, and edit NativeScript applications without the need for opening a command prompt, switching to the correct directory that contains the application, and running manual lines of code to edit the application.

Microsoft Visual Studio was not used in our project to code in.

3.3.2.3 Code Composer Studio

Code Composer Studio is an IDE used in some electrical engineering classes that focuses on computer abilities as well. This IDE is what the C and Assembly language codes that the members of our group have written to program the TI MSP430. Although it is an uncommon program for our group in terms of usage outside of a classroom setting, there is enough support both online and from the documents accessible to UCF students to write plausible code in this cleanly-styled editor to program the TI hardware we have previously used in a couple of our classes.

CCS is the least preferred IDE to use that every student in the group working on this project has experience with. Although coding in C and Assembly allows powerful commands to be programmed into a TI MSP430, the IDE’s use is limited to only this specific type of TI hardware as far as our expertise in this field is concerned. There are various hardware devices that Code Composer Studio can program as well, but most likely this IDE would only be used if we were to have ended up implementing the TI MSP430 into our final design plans.

3.3.2.4 NativeScript

NativeScript is another IDE that may have been used to create a mobile application to further enhance user capabilities of this project. NativeScript takes its name from the ability given to those who use the program to create mobile apps that have completely native code, or code that is executed straight from the CPU of a computer, running them. The benefit that NativeScript brings to the table is its compatibility with JavaScript coding and with Microsoft Visual Studio. If a web application that performs the same functions as a mobile application has all the code needed to run it written already, making the mobile application using Android Studio requires restructuring of all the code that has already been written for the web application. JavaScript code written for the web application would essentially have to be converted into Java code. However, the use of NativeScript allows for JavaScript implementation, meaning that a majority of the code used to program the web application could also be used for the mobile application with minor tweaks made to the JavaScript code. This would save our group a significant portion of time creating the mobile application, allowing us to test the functionality of our web and mobile applications at a sooner time than using Android Studio would allow.

However, we did not have to use it since we only made a web-application and not a mobile on.

3.3.2.5 Atom

Atom is a highly-functional IDE that the computer engineering student in the group working on this project has used to code and debug programs written in Java. Atom can be used in conjunction with Android Studio to write code for a possible mobile application extension to this project. Atom also has downloadable extensions that make it compatible with most of the other programming languages that may be used in this project and offers its own compiling services.

This was not used either.

3.3.2.6 Notepad++

Notepad++ is the simplest IDE out of the ones being considered for use in this project. This editor will be used to access and program the HTML languages used in the web application development that is integral to the user interactions that will occur with the mesh network. PHP pages that control the structure of the web pages in the web application are easily editable and readable in Notepad++.

This was the main IDE used to code in this project since it supports every type of coding language that was necessary for MOIST to function properly.

3.4 Online File Management

The two web development stacks that may have been implemented in this project, LAMP (Linux, Apache, MySQL, PHP) and MEAN (MongoDB, Express, Angular, Node.js), both require a file directory for the pages that the web application will contain and display. To upload changes to the web application to the actual server that it will be hosted on, rather than just making changes to it on a local host such as a student’s laptop, the edited files will be placed in the file directory of the web application’s server using one of a few file hosting programs. Figure 007 shows how the LAMP stack will be structured in terms of the different components that are involved in it. The code along with the server will pair up with and contact the database to deliver and retrieve new information picked up by the sensors.

For MOIST, both stacks have their own advantages. The MEAN stack would have been useful for providing this project with simpler access to more modern action commands and UI capabilities. For instance, using the NativeScript IDE to create a mobile application to govern the mesh network is simpler writing in the code that responds well with the MEAN stack, TypeScript, than using JavaScript, the code that corresponds with the LAMP stack. Using MEAN would have allowed us to create a nicer looking design for the mobile application through NativeScript than using the other option.

Given the advantages of the MEAN stack, the LAMP stack will still most likely be used to create the web and/or mobile application that governs the mesh network due to the familiarity that this group of students has with web development using the components of the LAMP stack. Figure 4 below shows the functionality of the different components of the web stack. The operating system used to code the web application for this project will be Linux based, or Windows if too much difficulty is encountered trying to use Linux to code the application. The Apache web server will be connected with and the PHP pages will be coded using the Linux OS. These components will incorporate JavaScript on the front-end to create the user-interface of the web application. The Apache, Linux, and PHP pages are considered the back-end of the application, since they all center around the aspects of a web page that is invisible to the common user. These back-end components will connect the web application to the database, MySQL, whenever a change is authorized or requested by the user through the front-end.

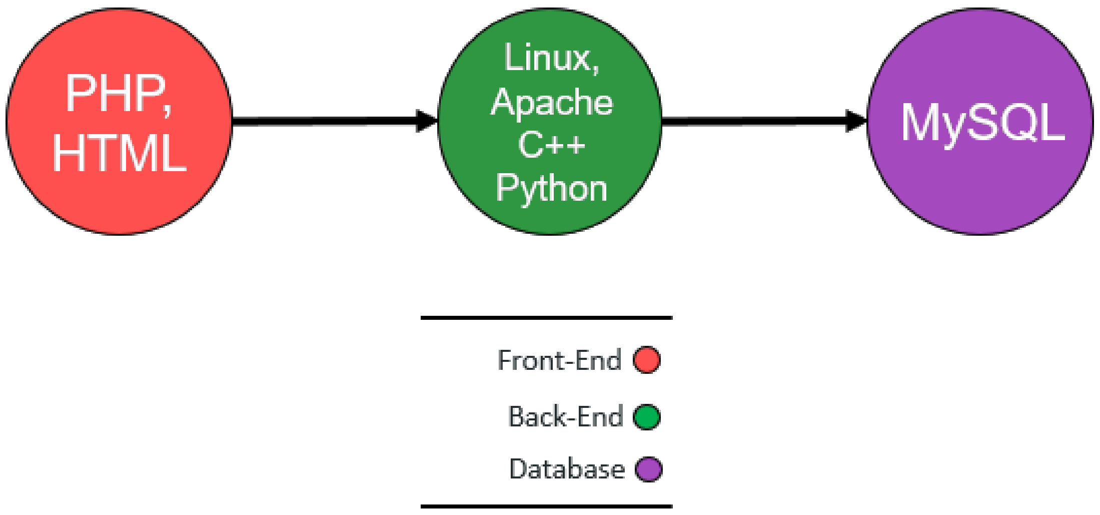


Figure 4- LAMP Stack Functionality

3.4.1 FileZilla

FileZilla is a free FTP (File Transfer Protocol) program that transfers files between a local host and a directory that the local user connects to via a secure login procedure. To access a certain server, the correct host location and user and password credentials must be entered into the site manager option in

In order for a connection to be made between a local host and a server on a different computer using the internet FileZilla has to be used. This program allows for secure connections to be made between a user and a private server using the different options for connectivity that FileZilla provides such as SFTP (Secure File Transfer Protocol). This program will likely be used manage the web application due to the experience that the computer engineering student in the group working on this project already has with creating a website using FileZilla.

3.5 Abstraction

With all of the strategic components listed, abstract diagrams showing how the components work together and what the flow of operation in this project is helps solidify our design. From the hardware algorithm to the user's front-end web application, these components are all concurrently working together to achieve our objective. Each diagram gives insight into a different aspect of MOIST that, when paired together with each other, display the inner workings of this project and how it ultimately functions.

Careful thought was put in to the design of the diagrams in this section. The group of students working on this project did a large amount of discussion and collaboration before deciding how the components of these diagrams, such as the nodes and the firmware, were going to be structured. After a large amount of deliberation, the connection between all of the components in this project was decided upon. The diagrams are a reflection of this connection, and shows how each key area in the project, which each of us focused primarily on separately, comes together to make the project flow and run smoothly.

As insightful as these diagrams are, they are only the preliminary plans of the operations that this project will carry out. Although the actions taken to complete this project still mirror the designs in this abstraction, the actual in-detail designs of the different components in MOIST are more complex than what is shown in this section. The main layouts that are found in this section were studied and then expanded upon to allow adjustment for the products that would ultimately end up being used in this project. This implementation of the actual products used only made the designs in this section slightly more in-depth than what the abstraction diagrams contain. Figure 5 details the process of the Node.

These firmware flowchart diagrams did not change since EEL 4914. The Node and Gateway Block Diagrams were altered to display the updated connection between the different parts of our project. Color was also added to these diagrams to further show the distinction between these sections.

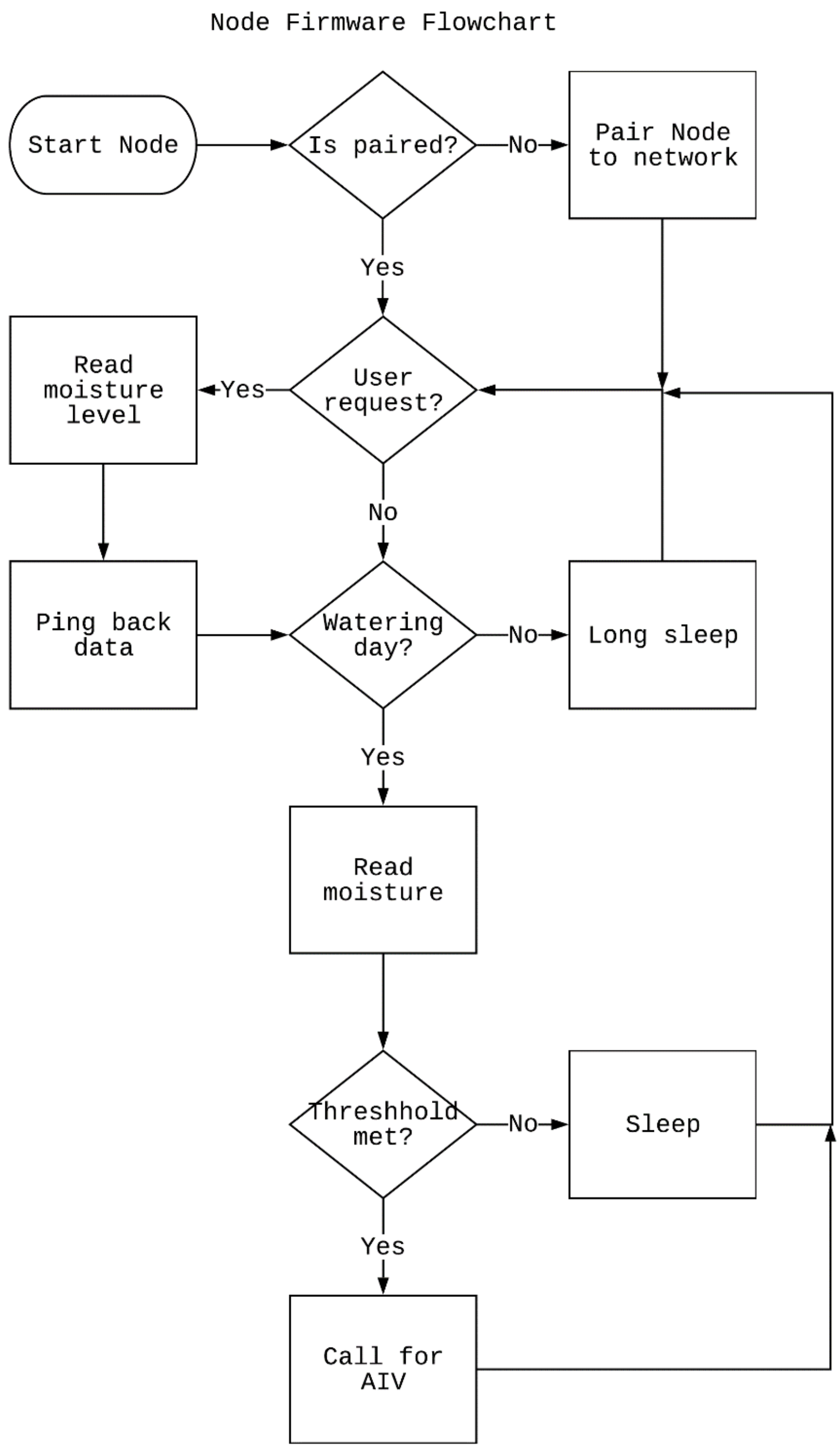


Figure 5- Node Flowchart

To start the network, individual Nodes need to transmit and receive at different addresses paired to a single parent Node and few child Nodes forming a tree topology. The Gateway will be the head of the network and doesn't have a parent Node. After a Node is connected it saves the setup so when it reboots its already connected. The normal procedure initiates in an endless loop. It waits for any signal received from the user, if it times out, the Node checks for watering day, if it is then proceeds to read sensor level. If not, it goes back to a timed sleep. Figure 6 details the operations of the Gateway and Uses its APIs to connect its own hardware to the internet and web-app as well as having APIs for accessing the database.

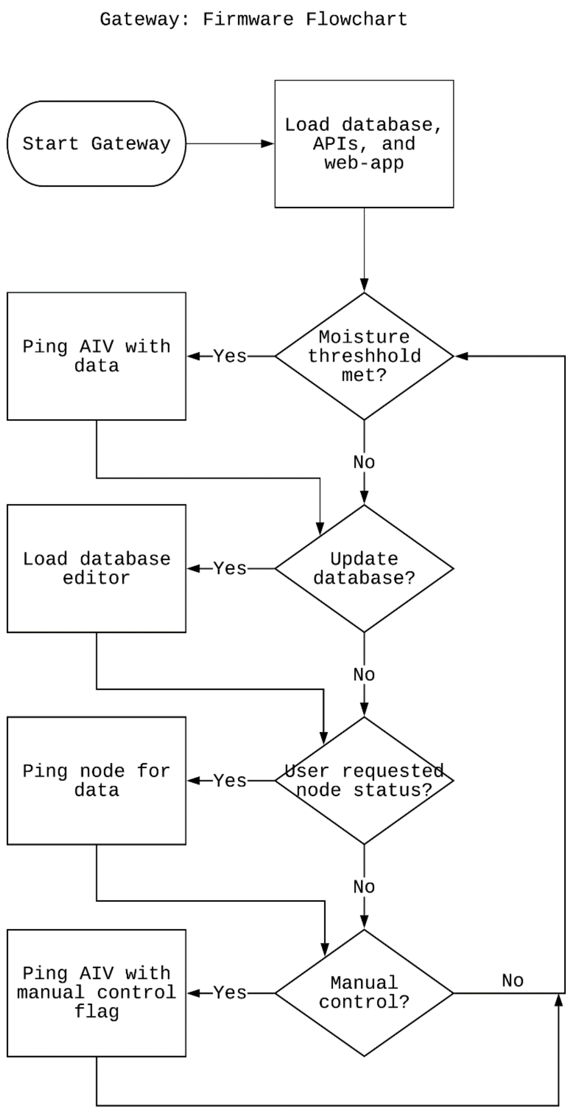


Figure 6- Gateway Flowchart

The Gateway starts by connecting to a LAN and hosts the web-app, and database for the Nodes. The root code will make these calls as it transmits to the nearest Node.

The next set of figures explains the procedures of the APIs: primarily the web-app and the database of the system. They detail how the connection is made and maintained so network is stable as the users accesses the interface. Figure 7 describes how the menu operations would precede.

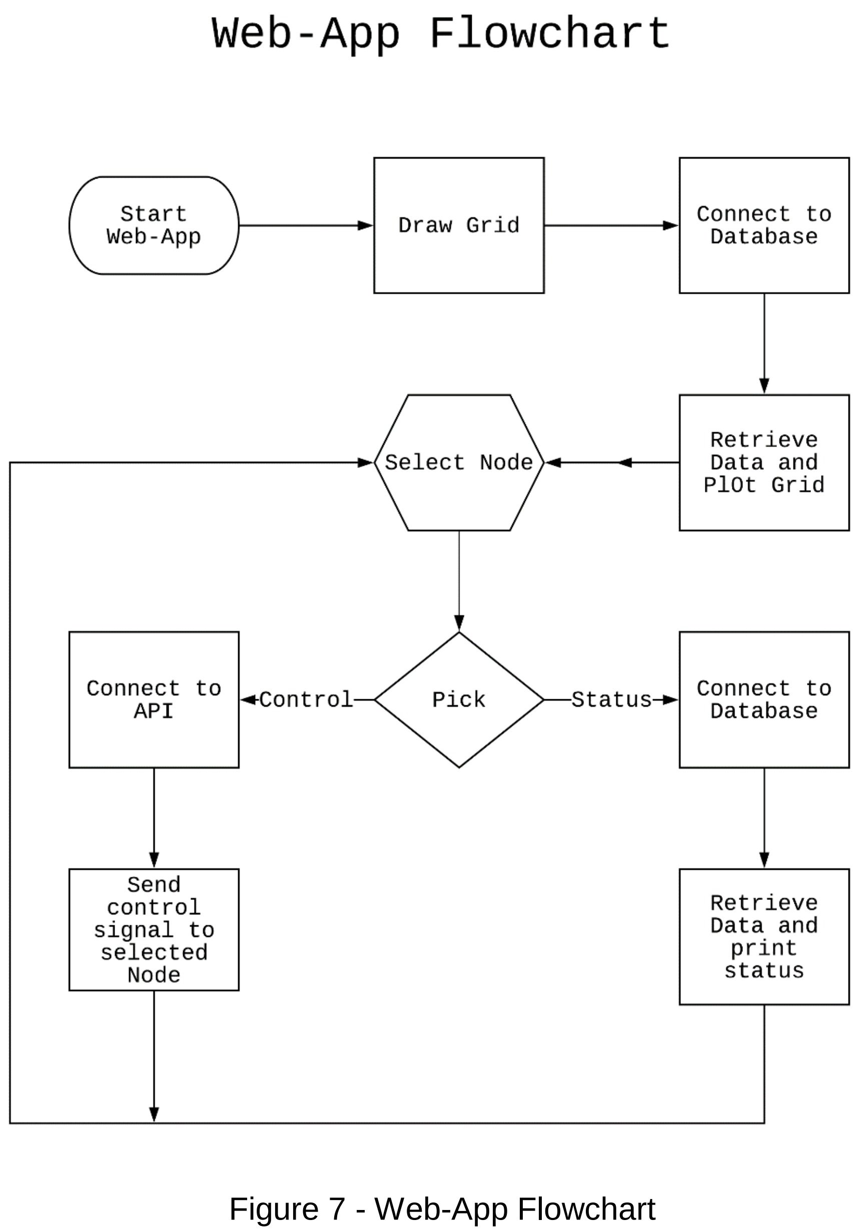


Figure 7 - Web-App Flowchart

The web application is more of the front end of the user interface. It is what our client or any user would see when accessing the system. It starts with the script accessing the database via an internal API, and plots the points on a graphic grid. This would be relative it the position of the Nodes on the field. After the Nodes are updated it runs in a loop and waits until a user selects a Node on the web-app, make a call for a status or command and use an API to accomplish the request. Figure 8 outlines what is happening internally in the database as it waits for a request.

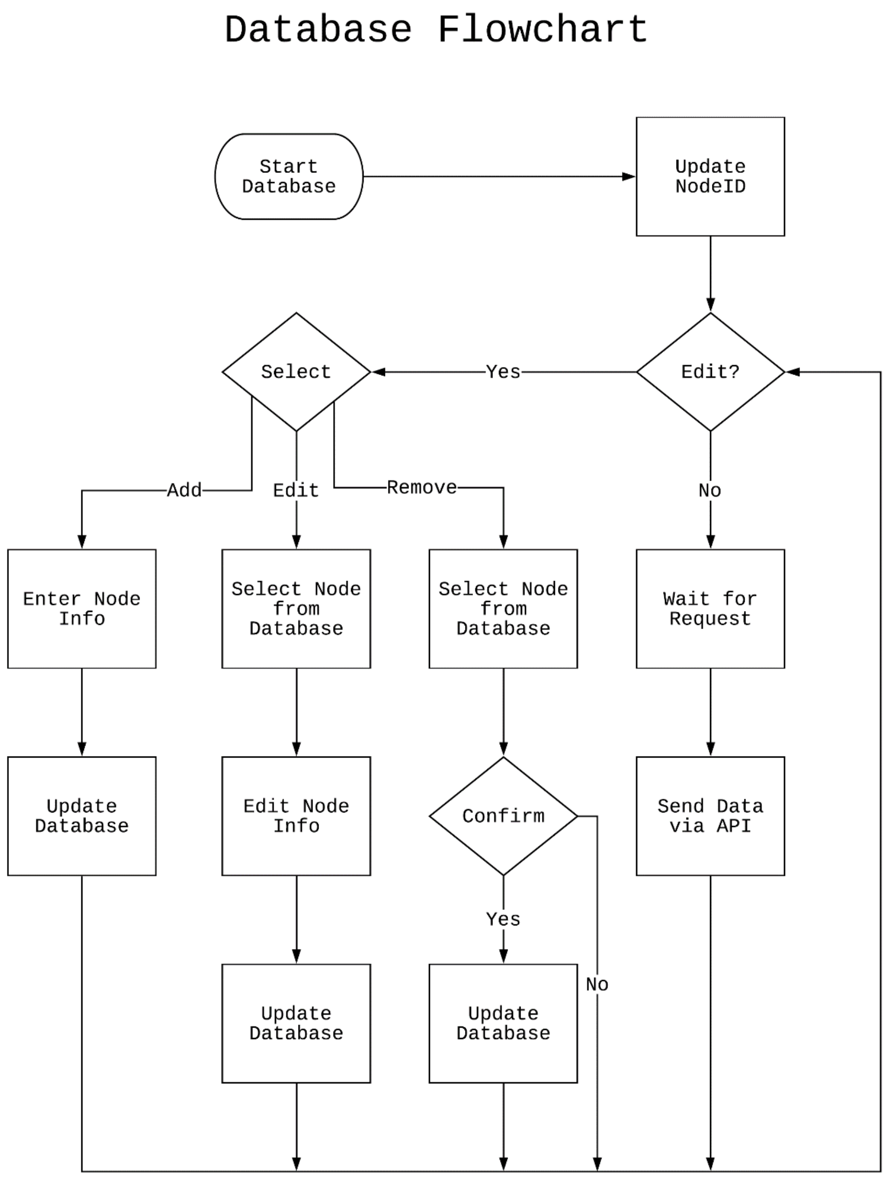


Figure 8- Database Flowchart

The database is a huge component for the server. It holds, and organizes all the data for each Node entered, it is accessed directly or through API calls in functions of the main code. When that happens the database search for the correct entry and returns the data to the firmware and through the mesh, web-app or even to the Autonomous Irrigation Vehicle. Then it waits again for another request. Figure 9 maps one for the login screen, three more to connect to the database, and an additional one for the firmware script.

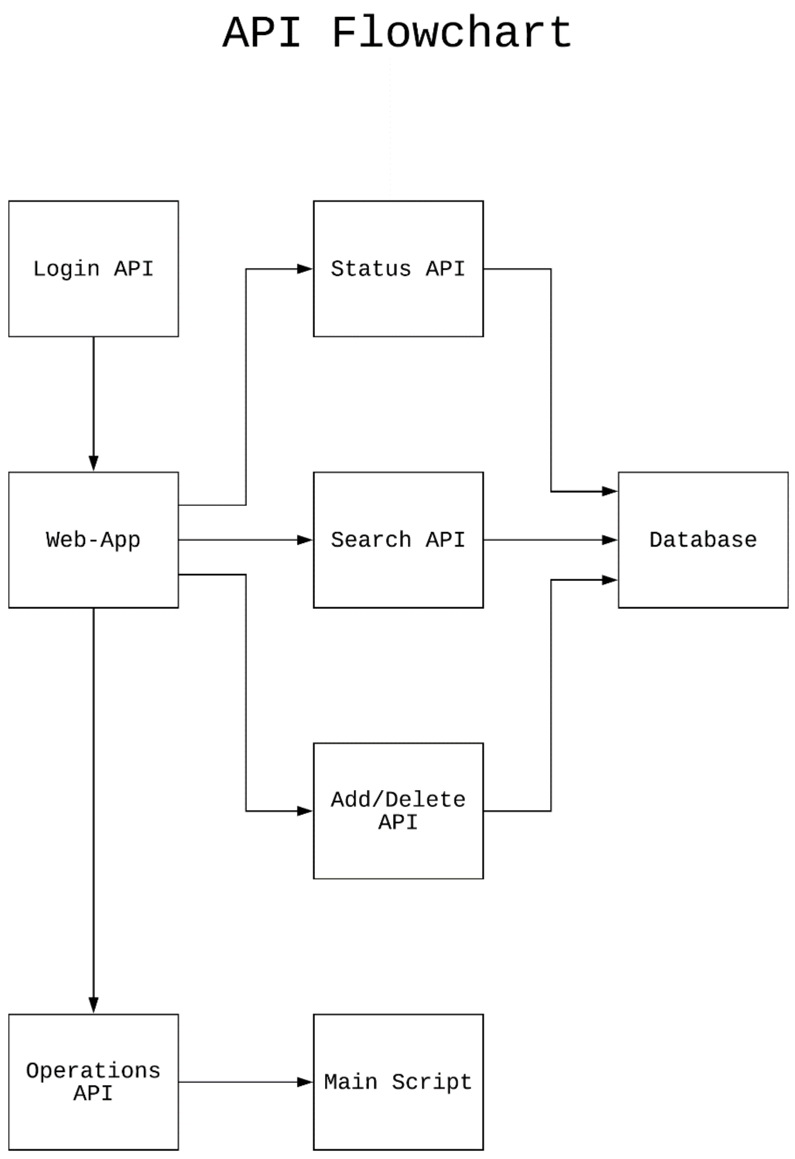


Figure 9- API Flowchart

With all the sub programs in the server running, some internal connectivity is required to make the operations run smoothly. Hence API (Application Programming Interface), snippets of code or executed apps, that are used when calls from one service is requested for another. There are five main APIs that help the Web-App make its connection. These APIs are not a main concertation of our design but more so of supplements and therefore are necessary.

4.0 Project Design Detail

In this section exists a detailed layout of how our system is implemented for our project. As with any project, brainstorming and then finalizing the components is a must. Here our system is broken down into two main hardware subsystems and few software programs to achieve our objective.

4.1 Node Power System

A sensor node is usually equipped with rechargeable batteries, which have limited capacity and pose a challenge to long-term application. These batteries supply power to the sensor nodes by providing the necessary current to maintain each part of the sensor nodes working properly. The total power consumption of a sensor node is the sum of each element in the node (i.e. sensor, microcontroller unit, and RF module) each component may operate at different energy states. Therefore, the lifespan of a sensor node is the time consumed to exhaust its batteries under a sustainable operation threshold. We need to provide sensor nodes with an infinite lifespan.

- Solar panel for charging the batteries; the choice for solar panel
- Charger controller to prevent over charging
- Voltage regulator for regulating a constant 5V DC
- Battery indicator which indicates the remaining voltage left inside the battery
- Cover box to contain all the electronics circuit
- Rechargeable battery capable of holding adequate charge and output enough power

4.1.1 Solar Panel

Solar energy has become an alternative source of energy because of its efficiency and affordability, and also because of the arising cost of other sources of energy. The power needed for our project requires independent power source to aliment the sensors and wireless communication module in order to maintain scalable mesh network. The main factors for the choice of power source are cost, efficiency, and portability. The renewable energy such as solar energy which is mainly available was used as a power source for each node in that project. The poly-crystalline solar panel is chosen among the three types of solar panel technology, because it balances cost and efficiency. The size of the solar panel as well as the wattage matter. In this project we selected a 2W solar panel manufactured by NUZAMAS shown in Figure 10. Since we need a solar panel capable of charging a 7.4 Volt rechargeable battery then the open circuit voltage of the solar panel must be greater than 7.4 volt which is the reason that we chose a 2 watts solar panel with a Voc(Open Circuit Voltage) of 14.9 volt.

Figure 10- NUZAMAS 2W 12V 160mA

Designing the holder for our solar panel will be crucial, as the position of the solar panel determines how effective its light to electricity conversion in any weather in comparison to a better position in that same weather.

We ultimately decided to use a NUZAMAS 2W solar panel because it met all of our requirements for a solar panel while consuming less power than the one we originally intended on using.

4.1.2 Charge Controller

The charge controller prevents over charging of the battery by automatically reducing the current to a low level when the battery is fully charged. The input is connected to the solar panel and the output is connected to the battery. The state of the switch is determined by two voltage thresholds.

When the battery is bulk charging, the load switch is open and the full solar panel output is applied to the battery. When the upper threshold is reached, the load switch closes and the battery discharges through the load until the lower threshold is reached and the switch opens allowing the battery to charge once again. In a series controller design, a relay or solid-state switch either opens the circuit between the array and the battery to discontinue charging or limits the current in a series-linear manner to hold the battery voltage at a high value. As these on-off charge cycles continue, the ‘on’ time becomes shorter and shorter as the battery becomes fully charged. In our project, we are choosing LTC4079 from linear Technologies.

The LTC4079 is a low quiescent current, high voltage linear charger for most battery chemistry types including Li-Ion/Polymer, Lead-Acid or NiMH battery stacks up to 60V. The maximum charge current is adjustable from 10mA to 250mA with an external resistor. The battery charge voltage is set using an external resistor divider. With an integrated power device, current sensing and reverse current protection, a complete charging solution using the LTC4079 requires very few external components. Thermal regulation ensures maximum charge current up to the specified limit without the risk of overheating. Charging can be terminated by either C/10 or adjustable timer. Input voltage regulation reduces charge current when the input voltage falls to an adjustable level or the battery voltage, making it well suited for energy harvesting applications. Other features include temperature qualified charging, bad battery detection, automatic recharge with sampled feedback in standby for negligible battery drain, and an open-drain CHRG status output.

Figure 11 depicts the pinouts of the chip. This will inidcate how we wire the chip to the PCB.

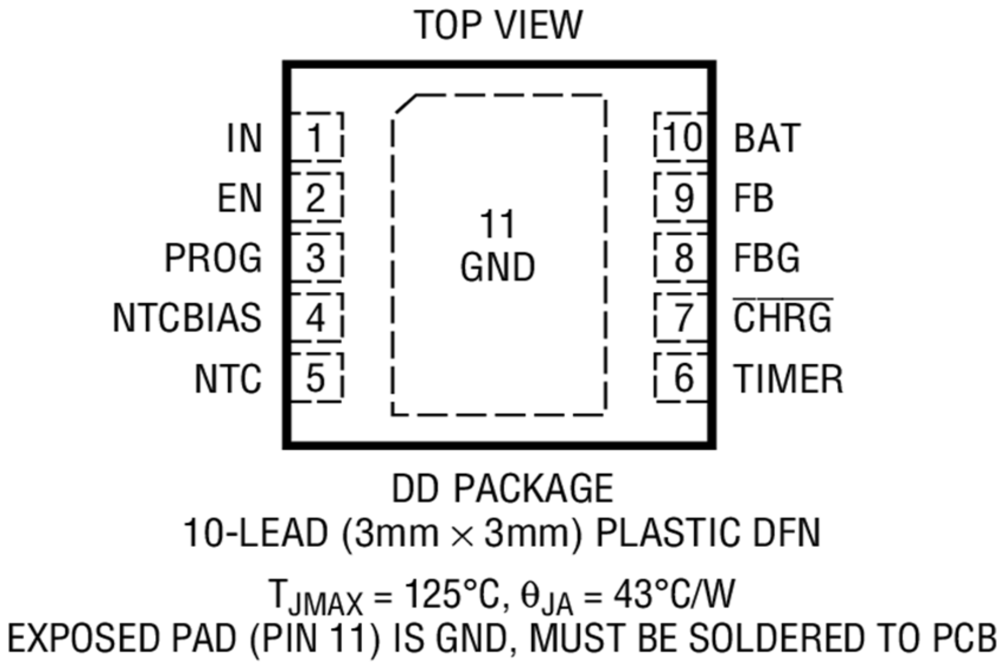

TOP VIEW
IN
EN
PROG
NTCBIAS
NTC
1
2
3
4
5
11
GND
10
9
8
7
6
BAT
FB
FBG
$\overline{\text{CHRG}}$
TIMER
DD PACKAGE
10-LEAD (3mm × 3mm) PLASTIC DFN
$T_{JMAX} = 125°C, \theta_{JA} = 43°C/W$
EXPOSED PAD (PIN 11) IS GND, MUST BE SOLDERED TO PCB

Figure 11- LTC4079 Pinout (Permission was requested to Analog Devices)

IN (Pin 1): Input Supply Pin. This input provides power to the battery charger. Bypass this pin with a ceramic capacitor of at least 1µF [14].

EN (Pin 2): Enable Input. Charge current starts flowing when this input rises above 1.190V, its regulation threshold. When using a current limited power source, connect this input to an external resistor divider from IN to GND to avoid UVLO oscillations. This configuration can also be used to maintain the source voltage (IN pin) at the maximum power threshold (e.g., for solar panel). Pulling this pin below 0.750V shuts down the device. This pin should not be left floating [14].

PROG (Pin 3): Charge Current Program Pin. The current out of this pin is 1/250th of the current out of the BAT pin. A resistor connected from PROG to ground sets the charge current in constant-current mode. This pin servos to 1.190V during constant-current charging. Do not leave this pin open. Limit parasitic capacitance on this node to less than 50pF[14].

NTCBIAS (Pin 4): NTC Thermistor Bias Output. Connect a low drift bias resistor from NTCBIAS to NTC pin, and a thermistor from NTC pin to GND. The value of the bias resistor is typically equal to the nominal resistance of the thermistor at 25°C. Minimize parasitic capacitance on this pin[14].

NTC (Pin 5): Input to the Battery Temperature Sense Circuit. Connect the NTC pin to a negative temperature coefficient (NTC) thermistor, which is typically co-packaged with the battery, to signal the charger if the battery is too hot or too cold to charge. The room temperature value of the thermistor should be at least 2kΩ. If the battery's temperature is out of range, charging is paused until the

battery temperature re-enters the valid range. Connect a 1%, low drift bias resistor from NTCBIAS to NTC and a thermistor from NTC to ground. Minimize parasitic capacitance on this pin. Tie the NTC pin to GND to disable battery temperature sensing [14].

TIMER (Pin 6): Timer Capacitor Input. A capacitor on this pin sets the maximum duration for battery charging from charger enable or from the beginning of a recharge cycle. For maximum charge duration of tTIMER (in Hours), the

required capacitance value can be determined as follows: CTIMER = (tTIMER • 18.2nF/Hr) A typical value of CTIMER is 100nF which terminates the

charge cycle after 5. hours. Minimize leakage on this pin to maintain timer accuracy. The timer is disabled when this pin is tied to GND. In this case charging terminates when the charge current falls below 1/10th of the programmed charge current ICHG [14].

CHRG (Pin 7): Open-Drain Charge Status Output. Typically pulled up to a voltage source through a resistor or a low power LED and a resistor. This pin is pulled low by an internal NMOS when LTC4079 is charging the battery. The pin goes to high impedance when the charge current drops below 1/10th of the programmed current, or the charge cycle is timer terminated. [14]

FBG (Pin 8): Ground Reference for Battery Voltage Divider. This pin is connected to ground internally through an NMOS switch when the battery is charging and disconnects the battery voltage divider from GND when it is not needed. When sensing the battery voltage the NMOS switch presents a low resistance (RFBG =160Ω) to GND[14].

FB (Pin 9): Sense Pin for Divided Battery Voltage. This pin servos to 1.170V (VFB(CHG)) during the constant-voltage phase of the battery charge algorithm. The battery charge voltage is set by using an appropriate resistor divider from BAT to FB to FBG. Minimize leakage and parasitic capacitance on this pin.

BAT (Pin 10): Battery Charger Output. This pin provides charge current to the battery.

GND (Exposed Pad Pin 11): Ground. The exposed pad must be soldered to a continuous ground plane of the printed circuit board for electrical connection and the rated thermal performance.

Setting The Battery Charge Voltage

The battery charge voltage is set by connecting a resistor divider from the battery to the FB and FBG pins as shown . The charge voltage is determined as follows:

VCHG = 1.170V • (1+RFB1)/(RFB2 +RFBG)

where RFB1 is the resistor from BAT to FB, RFB2 is the resistor from FB to FBG and RFBG is the resistance of the internal switch of the FBG pin (160Ω typical. Figure 12 is an image of how an actual package looks like.

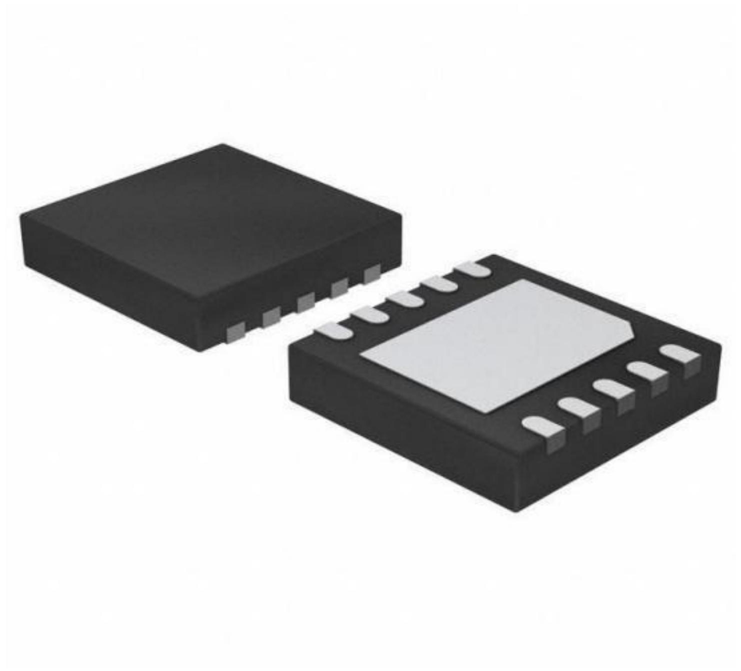


Figure 12- LTC4079 chip (Permission was requested to Analog Devices)

The package is of a surface mount type with unique flat pinouts. A design of the Eagle library was required since it is not a common shape, size, and number of pins.

4.1.3 Voltage Regulator

The voltage regulation (VR) set point is one of the key specifications for charge controllers. The voltage regulation set point is defined as the maximum voltage that the charge controller allows the battery to reach, limiting the overcharge of the battery. Once the controller senses that the battery reaches the voltage regulation set point the input voltage supplied by the battery is suitable for the microcontroller and other peripheral devices of the nodes such as wireless chips and sensors which need a voltage regulator to regulate a constant 3.3 volt to aliment the board.

Figure 13 is the schematic for our voltage regulator.

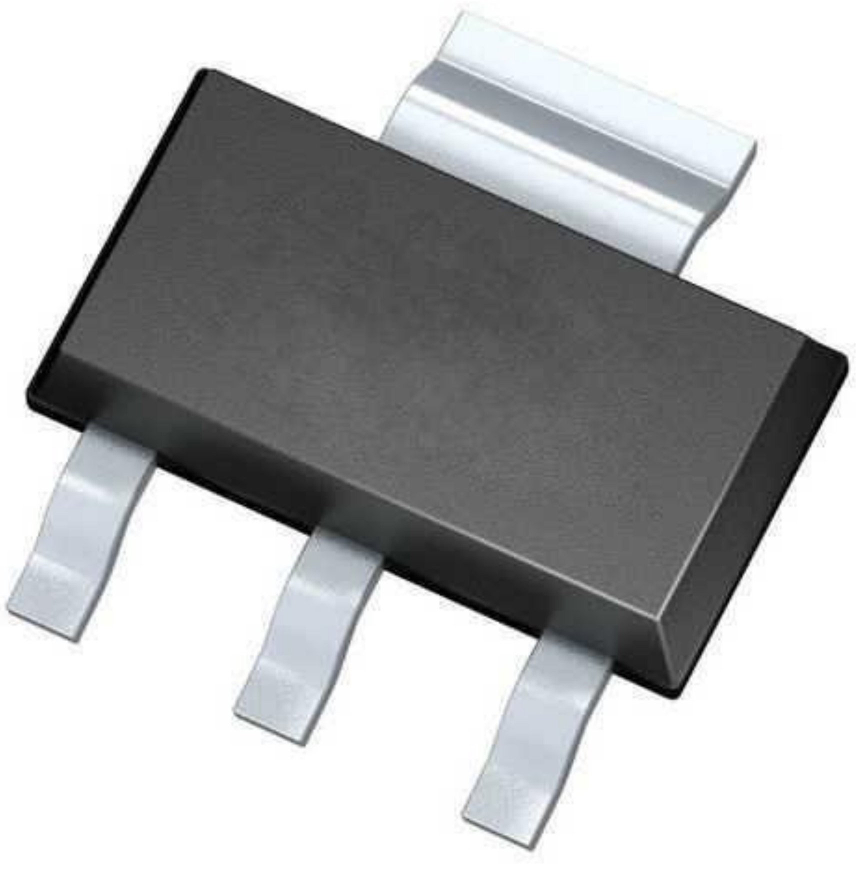


Figure 13- Voltage Regulator Package (Permission was granted by TI)

After careful consideration we have chosen a switching regulator which is constitute of LM1117 integrated circuit. The tab is connected to the second pin and is intended for a heatsink.

4.1.4 Battery

Solar panels are non-ideal power sources; batteries are often paired with solar panels to provide energy storage. The battery used for this project is the Customized 18650 5 V output with a 2200 mAh. Figure 14 shows how the battery looks. The battery was paired with the solar panel so that when there is not a proficient amount of energy provided by sunlight to power the Node's functions, then the battery assists the node with the rest of the power it needs to maintain the node's functionality.

Figure 14- Customized 7.4 18650

4.2 Node: Processing System

As with any form of system, there had to be a main controller to direct all the micro-operations of the node. Our choice of a micro-controller unit came down to many factors primarily including power consumption, adequate IO and ADC pins, and ease of programmability. Our candidate is the ATmega328P-PU, this choice was finalized based on our familiarity with this chip. The ATmega328P-PU is an Atmel, now owned by Microchip, IC commonly found on the known development board called Arduino.

4.2.1 Micro Processing Unit

This device has a power consumption of only 26 mW when active and drops to 6 mW when idling. With this low of a power consumption rate, it's a fit for our wireless, solar powered design. The biggest advantage of this chip is the stated familiarity of the device. Arduino provides an IDE that is user-friendly to program in and debug. We used an existing Arduino board as a programmer for this chip. This enables us to easily program the chip without adding excess hardware to our project design. And if in the case of a MCU failure, the PDIP socket allows for easy swap-out of chips.

ATmega328P-PU is based on the 8-bit AVR family of microcontrollers with a RISC architecture meant for low power consumption and IO peripherals in an embedded environment. Figure 15 shows the package for the MCU. The advantage of this MCU is that it features multiple types wired communication which is needed to interface with the rest of the hardware.

With 28 available pins, UART, I2C, and SPI can all be used simultaneously with additional IO pins allocated. In our design we only need a certain number of pins. Pin 7 is the VCC for powering the MCU ranging from 3.3-5.0V, Pin 8 and 22 are the ground pins for sinking current. Pin 24 is PC1 which can be configured as and ADC input for our sensor. Pins 15, 17, 18, 19 will be configured as IO for selecting a slave device, specifically the RF module, MOSI (Master Out Slave In), MISO (Master In Slave Out), and SCK (Serial Clock) respectively. This is the SPI interface which is required for communication with the RF module.

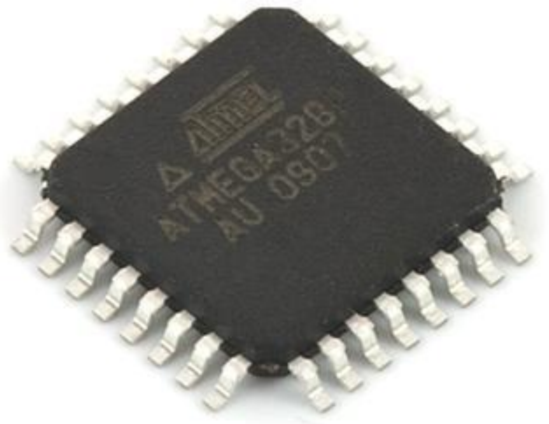


Figure 15- ATmega328P-TQFP (Permission was requested to Microchip)

This variant of the MCU is a surface mount device, therefore making it one-sided and avoids excess drilling. With modern electronics being mostly surface mount, most of our design was aimed to be kept up with this standard.

The Arduino community and Atmel as a whole provides a lot of support to troubleshoot any arising problems. Since Arduino is an open-source hardware, integration of the device is readily available, and marketable under the correct licenses.

4.2.2 MCU Programmer

The advantage of using the ATmega328P-PU is the Arduino hardware available for this device. The Arduino was initially used as a breadboard prototyping for testing components as well as doubling as a chip programmer. First upload the Arduino with ArduinoISP file, this allows it to be used a programmer. Then attach the new chip via SPI and burn the bootloader from the Arduino IDE to chip. Last write the code for the node, then upload using the Arduino. Once completed the chip is then disconnected and soldered into our PCB while maintaining the written code.

4.3 Node: Communication System

Our design is based on a mesh network of sensor to accurately measure soil across a wide range of field. For this to happen a form of RF communication is needed. A mesh is a type of wireless multipoint network where any one device, a sensor node in our project, can relay a message to any other device. The advantage of this is the possible scalability across a field. A gateway is responsible for collecting data and issuing commands to any node. If the requested node is set at a distance past the wireless range limit, an arbitrary node located in some mid-point between the gateway and the requested node acts as a repeater of message. The gateway broadcasts the message, the arbitrary node receives that message and rebroadcast it, and finally the requested node listens in on the rebroadcast. The larger the distance, the increased number of bounces happen.

Our decision on a viable component came down to these main requirements listed. First and foremost, the compatibility between the RF module and our ATmega328P-PU, with the support for an existing library to simplify our code and ease of programmability. Second requirement was the choice of frequency and protocol, which was decided upon the 2.4 GHz ISM radio band defined by the IEEE 802.15.4 standard. This type network is proven to be effective for personal multi-device set up. The last main requirement was an access for an external antenna via U.FL (IPEX) connection. This allows for a burrow of the device without the sacrifice of RF range.

4.3.1 RF Module

After extensive research our choice for this component came down to nRF24L01+ mini. It is a surface mount module with nRF24L01 chip embedded, this device is wired to the MCU via SPI, a four-wire interface which out performs the I2C in terms of data speed up to 10 Mbps. This module has a wireless transmission rate of up to 2 Mbps. While active the module consumes 42.9 mW during transmission of data and 37.95 mW during reception of data. And only consumes as low as 3.3 uW during sleep.

nRF24L01 consists of eight surface mount pads used to wire to the rest of our PCB depicted in Figure 16. Pin 1 is the VCC at 3.3V, pin 2 being the ground. Pin 3 is CE(Chip Enable) for powering down the device. Pins 4, 5, 6, 7 are CSN(Chip Select), SCK, MOSI, and MISO respectively, wired to the MCU for interfacing. Pin 8 is IRQ, an interrupt request pin, not currently implemented in our design.

The device is readily available with a low cost and power consumption made this a desired device in our design.

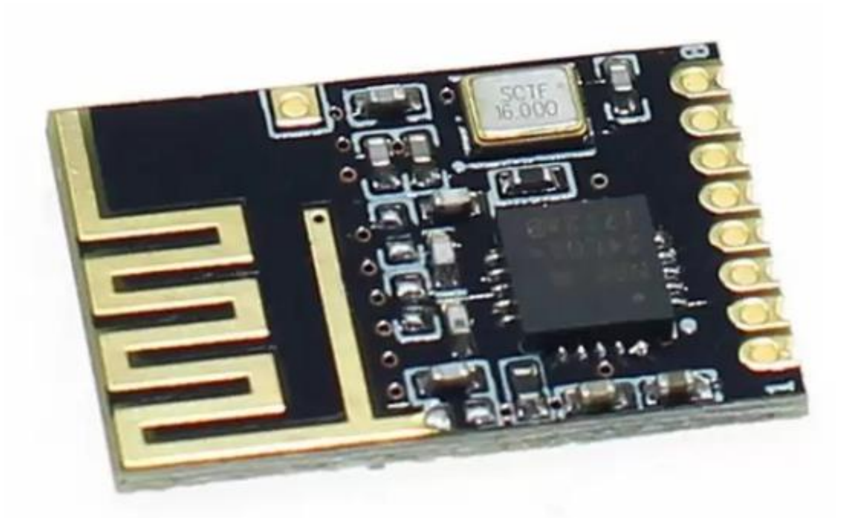


Figure 16- nRF24L01+ Mini

Since this module uses the nRF24l01+ as its transceiver, it’s compatible with open-source libraries and know uses mesh networking. The module made communication easy and integration applicable.

4.4 Node: Geolocation System

The need to setup a coordinate for each node is a need for our database, and for our web-app to accurately map the Nodes with potential location. As an initial install a GPS is only needed when setting or editing a location. This location also provides information to the Autonomous Irrigation Vehicle for accurate watering, thus achieving our objective.

4.4.1 GP-20U7

Based on research and availability, the Satellite System, also known as the GPS, is the best fit for our application. In particular the GP-20U7 is the optimal choice, with UART interface making it easy to communicate and available libraries to command this device. This very GPS module only draws 40 mA at 3.3V.

Figure 18- GP-20U7

Figure 17 depicts an image of a GPS module. With a small size and a shielding for its internal circuit. This module was easily integrated into our design. Making our Nodes capable of Geolocation.

4.5 Node: Environmental Sensor System

The system has taken into consideration power consumption which was aimed be minimized as much possible at every level of the system design. Each node is constituting of a probe with multiple sensors to monitor the moisture and temperature in the soil. Each sensor in the probe was calibrated to measure the data from the soil in order to properly convert the output of sensor to a measured unit, and the output of the sensor is interfaced with a microcontroller. To optimize the quantity, diversity and accuracy of information extracted from a precision irrigation deployment, a variety of reliable, high-performance and cost-effective sensor technologies were required. Many parameters were controlled such as temperature, humidity, soil moisture. These Nodes allowed for the installation of a large number of sensors with wireless data transfer to a base station. Several soil parameters had to be controlled to have an efficient irrigation system. Measuring soil moisture is important to calculate the exact quantity of water needed for each region in a field. Measuring temperature indicates when to open the water container or send a message to an AIV to start irrigating an area. There are different principles of moisture sensors such as capacitive sensors where soil moisture content is determined via its effect on dielectric constant by measuring the capacitance between two electrodes implanted in the soil. In our project the chosen system is the resistive method which consists of two electrodes enclosed in a block of porous material. Resistance blocks work on the principle that water conducts electricity. The electrical resistance between the two electrodes increases as the water content of the porous block decreases. The principle key to implement this method is the voltage divider circuits which reads out the voltage drop across the soil samples and converts the voltage drop reading to the moisture level of the soil. As the moisture level increases, resistance of the metal (usually metal pins) decreases, that is, conductance of the metal increases and similarly when the moisture level decreases the resistance of the metal increases, that is, conductance then decreases. Therefore, this sensor was used instead of a conductivity sensor because it is insensitive to salinity and it is not corrosion over time. Indeed, the sensor used is small, rugged and of lower power. The soil moisture sensor was provided by UCF for soil monitoring, and with sensor (DHT22) we can control ambient temperature and humidity. The two-wire serial interface and internal voltage regulation allows for easy and fast system integration. The tiny size and low-power consumption makes DHT22 the ultimate choice for our project. When urgent conditions are detected on other nodes, a signal will be given to an AIV or the actuator to open or not the water valve for irrigation. To control the irrigation process, we would have used solenoid valves provided by Orbit Irrigation for testing purpose. The electric valve is a low-power consumption device and it is powered with 24 V for 50/60 Hz. In other words, sensor nodes went to sleep when idle and woke up to transmit data when required. The irrigation valves opened or closed depending on the values stored in the coordinator node.

4.5.1 VH400

We decided to use the VH400 sensor primarily for its accuracy and relevant data format. This sensor is made by Vegetronix, making this device reliable. Its output as an analog voltage reading correlated to moisture content. Integrating this device was easy since it only requires an ADC, and manipulation of the reading to convert it to VMC. Figure 19 is an image of the sensor.

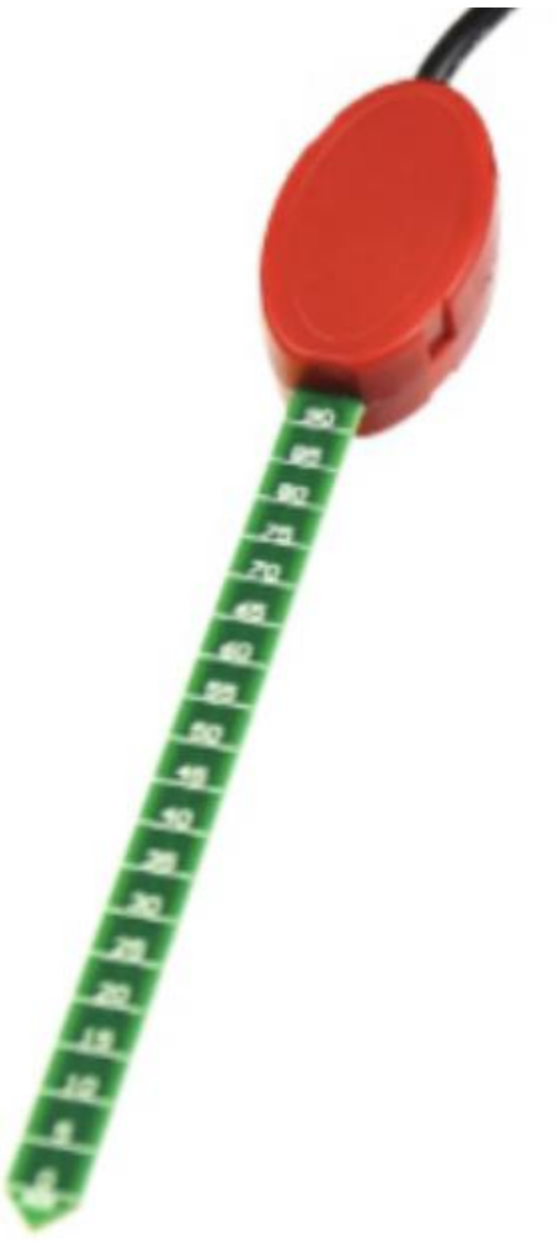


Figure 18- VH400 Sensor

This picture depicts the probe, which is powered via 5V and reads the measurement based on capacitance.

4.5 Node: Enclosure

We are not creating the sensors that will make our network system more efficient, we are instead seeking methods to build on to the beneficial impact of these sensors to society that the group who built these sensors were not able to implement themselves due to a lack of time or due to a degree of difficulty. The main way we attempted to improve the already existing design of these sensors was by enclosing the sensors along with the other hardware components we would have been included in each node of the network with a cover that keeps all the materials inside of this enclosure safe from the harm that outside conditions can impose on electrical equipment. The enclosure was not made to be fully impenetrable, however, to allow the hardware inside each node to receive solar energy through their solar panels.

4.5.1 Material Composition

The material of the enclosure had to be an acceptable balance between durable and lightweight, so that the enclosure could be easily movable and so that it would not self-implode and break inward, harming the contents of the Node, but also could not have been too light to the point that an insignificant force, such as a light wind, could relocate or deface enclosure material used.

4.5.2 Shape

Along with being lightweight and durable, the enclosure had to embrace a shape that covers the whole area of the node without being obtrusive to the field of grass that these Nodes were placed on. The shape had to also be as small as it can possibly be made while also covering the whole node with enough breathing room to spare for all the hardware located within each node to run at its standard level without suffering from heat damage in a small enclosure. This minimal design helped our group save money that was properly allocated toward other getting a hold of other necessary products.

4.6 Node: Optional Watering System

When the concept of our project was initially thought of, our solution to efficient irrigation involved using our own solenoids to connect our own mesh network. After we talked to our professor and previous sponsor, we decided that we would not be focusing solely on the networking aspect of our initial concept rather than incorporating sprinklers into it. As a result, we decided to leave pins on our nodes that can connect to a solenoid and be tested with a sprinkler network should our group ever decide to explore the capability that our network will have with different instruments of watering plant life.

4.7 Gateway: Central Hub

Although personal computers are relatively inexpensive, there are distinct disadvantages to using them in a sensor networks project, especially as sensor nodes. If the sensors are located in areas where main power is unreliable or unavailable, or where there is a risk of overheating, or where there is simply no room to install a personal computer, we must either transmit the data to another node for processing or store it locally and process it later. However, there is another limitation to using a personal computer as a sensor node: a personal computer has no general input/output (I/O) ports. We could have purchased expansion cards for collecting data, but these are often built for use in server or desktop computers. The cost of the computer and the data-collection card, the cost of the sensor node becomes uneconomical. So what do you do in these cases? If only there were a low-cost computer with sufficient processing power and memory, that used standard peripherals, supported programmable I/O ports, and had a small form factor. That's exactly what the Raspberry Pi can do, and Figure 20 depicts an image. For the base station, we used a low power credit-card-sized single-board computer Raspberry Pi Model B. The CPU on the board is an ARM processor with 700 MHz clock speed. CPU performance can be compared to a Pentium II 300 MHz processor. It has a variety of interfacing peripherals, including USB port, HDMI port, 512MB RAM, SD Card storage and interestingly 8 GPIO port for expansion. Monitor, keyboard, and mouse can be connected to Raspberry Pi through HDMI and USB connectors and it can be used like a desktop computer. It supports a number of operating systems including a Debian-based Linux distro, Raspbian, which is used in our design. Raspberry Pi can be connected to a local area network through Ethernet cable or USB Wi-Fi adapter, and then it can be accessed through SSH remote login. Functional building blocks of the base station, including gateway application, database, and web application, are shown in the Abstraction section. Raspberry Pi is connected to the network with an attached shield containing nRF24L01 module. The gateway application is the intermediate layer between the sensor network and the database. It sends out configuration and data collection commands to sensor nodes and inserts the data received from sensor nodes into the MySQL database. The gateway application is programmed in Python that comes built-in with Raspbian. Some of the required Python packages include PySerial 2.7, MySQL-python 1.2.5, Advanced Python Schedule (APScheduler 2.1.2)

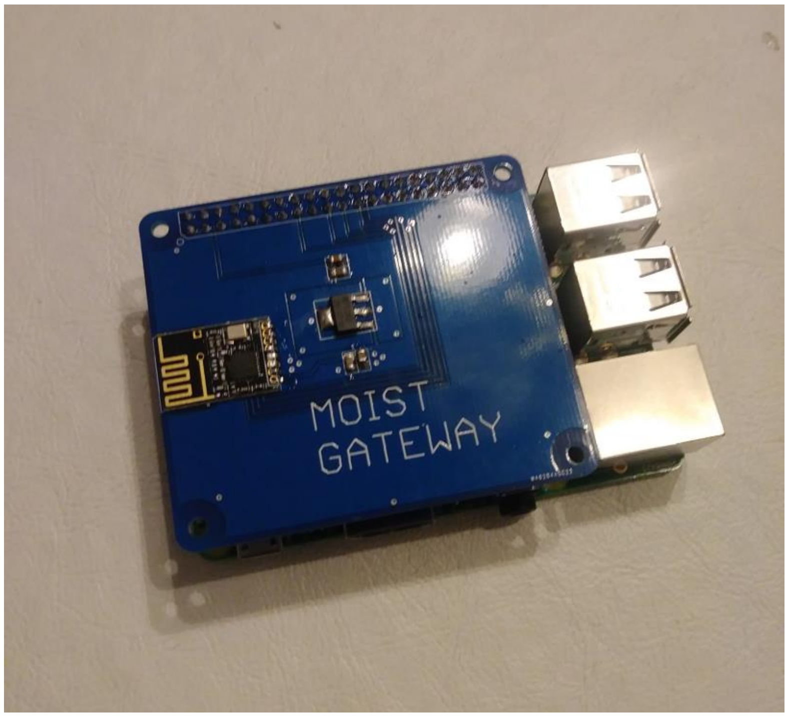


Figure 19- Raspberry Pi Model B

With all taken into consideration for our application, the Raspberry Pi is the best fit for our design and was used in our project accordingly.

4.8: Gateway: OS

All the data that is gathered and used during the usage of the sensor network in this project travels from the sensors into the Raspberry Pi being used through the Linux server that is used to control the functions of the network. Although the students in this group all have Windows computers and perform most of their daily tasks involving a computer on Windows, a Linux server has been chosen to operate the functions of the mesh network due to the familiarity that a couple of our students already have with running commands on Linux command prompt to control a Raspberry Pi. This technology is designed to be better equipped for code ran in a Linux environment. In addition to the ease of use between Linux and Raspberry Pi, coding the Pi's functions in Linux format allows for a deeper learning and understanding of the Linux OS, which is not commonly used in most electrical or computer engineering classes offered at UCF. Using Linux as the gateway provides us with a beneficial change from the Windows programs that are usually used by this project's group members. There are several engineering companies that are currently involved and will continue to be involved with projects that are centered around a Linux programming environment. Working with a Linux OS has provided us with a broader scope of technical range in terms of software and hardware, widening the scope of jobs that we will be able to obtain upon graduation. The students in this group currently have experience with running Linux commands in a virtual machine running a less modern version of Ubuntu. Since Linux is open-source under the GNU General Public License, support is available via forums or other guides, and the support for a LAMP stack is easily implemented. This support has lessened our learning curve for the OS.

4.9 Network Topology

There are many types of potential network setup each with pros and cons. By carefully understanding these networks and how each can be implemented is crucial for our design. Setting up the network is vital in terms of speed and power consumption. If not networked correctly the overuse of a single Node or a fail in data transfer can be a potential issue. Collaborating with our client to figure out the land space and how to set up is necessary. Although many topologies exist, only a few need to be considered for our application, based on number of devices, distance, and frequency of communication.

4.9.1 Mesh Topology

Generally, a WSN consists of a number of sensor network nodes and a gateway for the connection to the internet. The general deployment process of a Wireless Sensor Network is as follows: firstly, the sensor network nodes broadcast their status to the surroundings and receive status from other nodes to detect each other. Secondly, the sensor network nodes are organized into a connected network according to a certain topology (linear, star, tree, mesh, etc.). Finally, suitable paths are computed on the constructed network for transmitting the sensing data. The power of sensor network nodes is usually provided by batteries, so the transmission distance of WSN nodes is short. The transmission distance can be up to 800 to 1000 meters in the open outdoor environment with line of sight. It can sharply decline in the case of a sheltered indoor environment to an estimated few meters. In order to expand the coverage of a network, the sensor network uses multi-hop transmission mode. That is to say the sensor network nodes are both transmitter and receiver. The first sensor network node, the source node, sends data to a nearby node for data transmission to the gateway. The nearby node forwards the data to one of its nearby nodes that are on the path towards the gateway. The forwarding is repeated until the data arrives at the gateway.

The positions of WSN nodes are random, and the nodes can be moved, sheltered and interfered with. The topology of mesh networks have great advantages in flexibility and reliability compared with other network topologies. The self-organizing management approach of network nodes can greatly improve the robustness of the network, resulting in a smart mesh networking technology, as shown in Figure 18. In smart mesh ad hoc networking technology, the node first monitors the neighbor nodes and measures the signal strength, and then it selects the appropriate neighbor node for time synchronization and sends a joining request. Then the neighbor node delivers the request to the gateway. The gateway receives the request and assigns network resources for the node. Figure 21 displays an example of the mesh network.

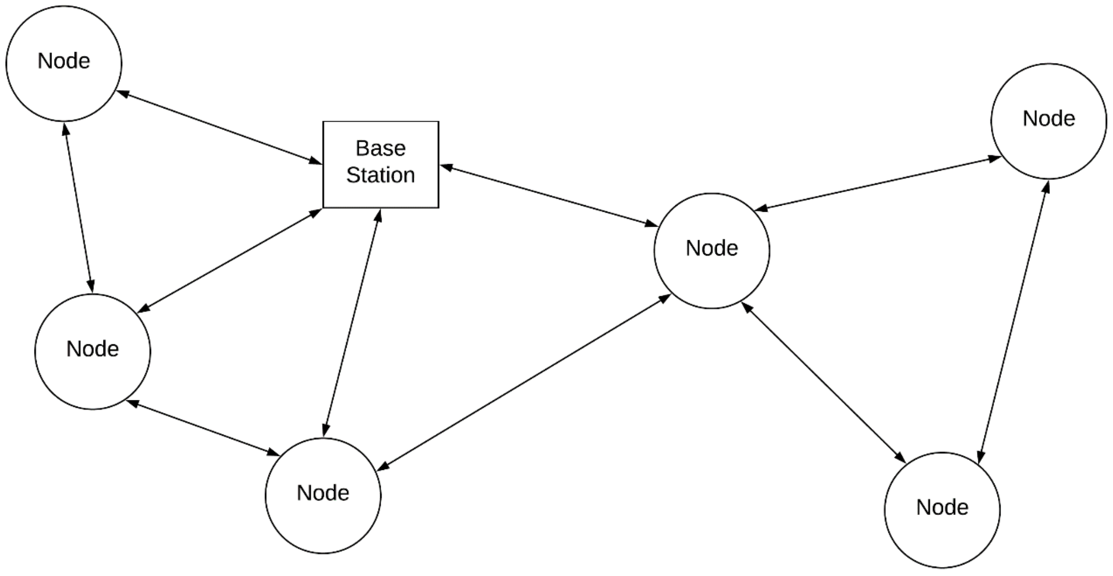


Figure 20- Mesh Network

This network will ultimately vary between land spaces as more or less Nodes were required in different instances. The position of the base station is also be determined by the exterior faucet for hoses. This needs to be in close proximity as required by the Autonomous Irrigation Vehicle. The implementation of the network is based on RF24Mesh libraries from NRF24L01+.

RF24 supports a variety of Linux based devices via various drivers. Some boards like Raspberry Pi can utilize multiple methods to drive the GPIO and SPI functionality [17].

PIN	NRF24L01	RPI	RPi -P1 Connector
1	VCC	rpi-3v3	(17)
2	GND	rpi-gnd	(25)
3	CE	rpi-gpio22	(15)
4	CSN	rpi-gpio8	(24)
5	SCK	rpi-sckl	(23)
6	MOSI	rpi-mosi	(19)
7	MISO	rpi-miso	(21)
8	IRQ	-	-

Table 2- nrf24l01 and Raspberry Pi Connection

Code 2 manually configures a node via RF24Mesh, and sends data to the master node. The nodes then refresh their network address as soon as a single write fails. This allows the nodes to change position in relation to each other and the master node[17].

```
#include "RF24Mesh/RF24Mesh.h"

#include <RF24/RF24.h>

#include <RF24Network/RF24Network.h>

RF24 radio(RPI_V2_GPIO_P1_15, BCM2835_SPI_CS0, BCM2835_SPI_SPEED_8MHZ);

RF24Network network(radio);

RF24Mesh mesh(radio,network);

uint32_t displayTimer=0;

int main(int argc, char** argv) {

// Set the nodeID to 0 for the master node
mesh.setNodeID(4);
// Connect to the mesh
printf("start nodeID %d\n",mesh.getNodeID());
mesh.begin();
radio.printDetails();
while(1)
{
// Call mesh.update to keep the network updated
mesh.update();
// Send the current millis() to the master node every second
if(millis() - displayTimer >= 1000){
displayTimer = millis();
if(!mesh.write(&displayTimer,'M',sizeof(displayTimer))){
// If a write fails, check connectivity to the mesh network
if( ! mesh.checkConnection() ){
// The address could be refreshed per a specified timeframe or only when
sequential writes fail, etc.
printf("Renewing Address\n");
mesh.renewAddress();
}else{
printf("Send fail, Test OK\n");
}
}else{
printf("Send OK: %u\n",displayTimer);
}
}
delay(1);
}
return 0;
}
```

```
#include "RF24Mesh/RF24Mesh.h"
#include <RF24/RF24.h>
#include <RF24Network/RF24Network.h>
// Set the nodeID to 0 for the master node
mesh.setNodeID(4);
// Connect to the mesh
printf("start nodeID %d\n",mesh.getNodeID());
mesh.begin();
radio.printDetails();
while(1)
{
// Call mesh.update to keep the network updated
mesh.update();
// Send the current millis() to the master node every second
if(millis() - displayTimer >= 1000){
displayTimer = millis();
if(!mesh.write(&displayTimer,'M',sizeof(displayTimer))){
// If a write fails, check connectivity to the mesh network
if( ! mesh.checkConnection() ){
// The address could be refreshed per a specified timeframe or only when
sequential writes fail, etc.
printf("Renewing Address\n");
mesh.renewAddress();
}else{
printf("Send fail, Test OK\n");
}
}else{
printf("Send OK: %u\n",displayTimer);
}
}
delay(1);

}

return 0;

}
```

Code 2- Raspberry Pi Mesh Config

As the master node is configured as the gateway node where almost every communication occurs between Nodes and the master Node, we also need some Node to Node communications which is provided by this code. This code defines the node to node communication using four principals libraries: RF24, RF24Network, SPI, and RF24Mesh. The code to make this node communication operable, Code 3, is shown below.

```
#include "RF24.h"

#include "RF24Network.h"

#include "RF24Mesh.h"

#include <SPI.h>

//#include <printf.h>

//########### USER CONFIG ###########

/**** Configure the nrf24l01 CE and CS pins ****/

RF24 radio(7, 8);

RF24Network network(radio);

RF24Mesh mesh(radio, network);

/**

* User Configuration:

* nodeID - A unique identifier for each radio. Allows addressing to change
dynamically

* with physical changes to the mesh. (numbers 1-255 allowed)

*

* otherNodeID - A unique identifier for the 'other' radio

*

**/

#define nodeID 3
```

```
#define otherNodeID 2

//####################################

uint32_t millisTimer = 0;

uint32_t stringTimer = 0;

char dataStr[] =
{"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ12345678
90"};

char tmpStr[sizeof(dataStr) + 1];

uint8_t strCtr = 1;

uint32_t delayTime = 120;

void setup() {

Serial.begin(115200);

//printf_begin();

// Set the nodeID manually

mesh.setNodeID(nodeID);

// Connect to the mesh

Serial.println(F("Connecting to the mesh..."));

mesh.begin();

}

unsigned int sizeCtr = 2;

uint32_t errorCount = 0;

uint32_t duplicates = 0;

uint32_t totalData = 0;

void loop() {

mesh.update();

while (network.available()) {

RF24NetworkHeader hdr;

size_t dataSize = network.peek(hdr);

totalData += dataSize;

if (hdr.type == 'S') {
```

```
if (dataSize != sizeCtr) {
if (dataSize == sizeCtr + 1) {
duplicates++;
}
sizeCtr = dataSize + 1;
errorCount++;
} else {
sizeCtr++;
if (sizeCtr > sizeof(dataStr)) {
sizeCtr = 2;
}
//if(sizeCtr > 12){ sizeCtr = 2; }
}
network.read(hdr, &tmpStr, dataSize);
//Serial.println(tmpStr);
} else if (hdr.type == 'M') {
uint32_t mills;
network.read(hdr, &mills, sizeof(mills));
Serial.print(F("Rcv "));
Serial.print(mills);
Serial.print(F(" from nodeID "));
int _ID = 0;
_ID = mesh.getNodeID(hdr.from_node);
if ( _ID > 0) {
Serial.println(_ID);
} else {
Serial.println("Mesh ID Lookup Failed");
}
Serial.print(F("Total Data Received: "));
Serial.print(totalData);
Serial.println(" bytes");
Serial.print(F("Detected Errors in data received (Including Duplicates): "));
Serial.println(errorCount);
Serial.print(F("Duplicates: "));
Serial.println(duplicates);
Serial.println(F("------------------------------------"));
}
}
// Send to the master node every second
if (millis() - millisTimer >= 1000 ) {
millisTimer = millis();
// Send an 'M' type to other Node containing the current millis()
if (!mesh.write(&millisTimer, 'M', sizeof(millisTimer), otherNodeID)) {
Serial.println(F("Send fail"));
if ( ! mesh.checkConnection() ) {
Serial.println(F("Renewing Address"));
mesh.renewAddress();
} else {
Serial.println(F("Send fail, Test OK"));
}
} else {
Serial.print(F("Send OK: ")); Serial.println(millisTimer);
}
}
if (millis() - stringTimer >= delayTime ) {
stringTimer = millis();
//Copy the current number of characters to the temporary array
memcpy(tmpStr, dataStr, strCtr);
//Set the last character to NULL
tmpStr[strCtr] = '\0';
```

```
// Send the temp string as an 'S' type message
// Send it to otherNodeID (An RF24Mesh address lookup will be performed)
//bool ok = mesh.write(tmpStr,'S',strCtr+1,otherNodeID);
if (mesh.write(tmpStr, 'S', strCtr + 1, otherNodeID)) {
strCtr++;
delayTime = 333;
//Set the sending length back to 1 once max size is reached
if (strCtr == sizeof(dataStr)) {
strCtr = 1;
}
//if(strCtr == 12){ strCtr=1; }
}
}
}
```

Code 3- Mesh Operation

4.9.2 Data Aggregation Network

In the energy-constrained sensor network environments, it is unsuitable in numerous aspects of battery power, processing ability, storage capacity and communication bandwidth, for each node to transmit data to the sink node. This is because in sensor networks with high coverage, the information reported by the neighboring nodes has some degree of redundancy, thus transmitting data separately in each node while consuming bandwidth and energy of the whole sensor network, which shortens lifetime of the network. Data aggregation technology saves energy and improves information accuracy, while sacrificing performance in other areas. On one hand, in the data transfer process, looking for aggregating nodes, data aggregation operations and waiting for the arrival of other data are likely to increase in the average latency of the network. On the other hand, compared to conventional networks, sensor networks have higher data loss rates. Data aggregation significantly reduces data redundancy but lose more information inadvertently, which reduces the robustness of the sensor network. Figure 22 depicts an example.

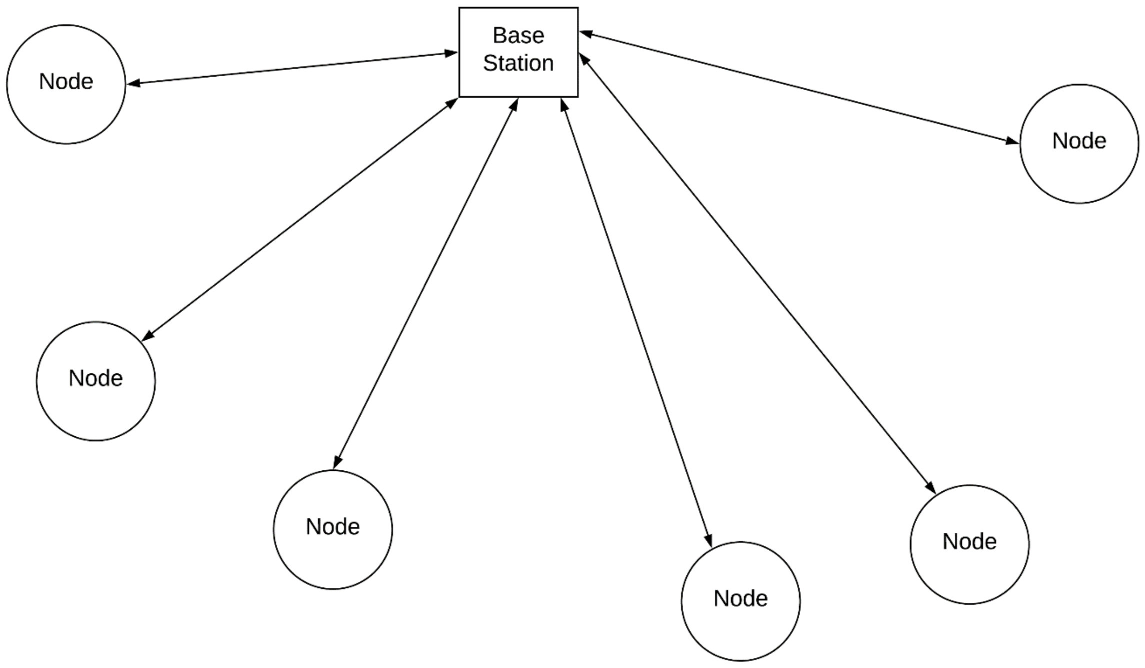


Figure 21 - Data Aggregation Network Topology

As you can see in the figure above, the location of the base station is vital for our network. Since the base station is more likely than not to be near the side of a house, the range of connection is severely limited, thus putting a lot of network strain on the base station and potentially causing a bottleneck of data.

4.10 Web Application

A major area of focus for MOIST, in order to enhance its efficiency and user-friendliness, is the web application we incorporated into our design that can assist users of the mesh network with checking the status of each node in the network, maintaining an operable watering system, and deploying commands through any of the nodes using a computer or smartphone with privileged access to the network.

4.10.1 Web Application Setup

The web application from the user's perspective initially was going to consist of a login screen so that the network remains secure and only allows those authorized to control the water usage of this patch of grass to have access to the system. The login screen then takes the user to a home page displaying the status of every sensor currently engaged within the mesh network. This page will indicate to the user what sensors are active and their most recent readings. From this page, users will be able to toggle the functionality of each sensor, such as what sensors should be turned on or off. There will be five API's (Application Programming Interface) running on the web application that will make the user functions described possible: login, search, status, and operations. There is only four in order to keep the web application simple and allow users to log in and immediately check their desired information.

The login screen was later deemed unnecessary since the network is already secure because everything is running on a localhost and not on the internet.

The client-side web interface is implemented with HTML, CSS, JavaScript, Ajax, jQuery, and Flot. HTML and CSS is used in combination to markup and style the web page. JavaScript is used for client-side scripting to enable dynamic display and interactive user interface. jQuery is a widely-used JavaScript library that greatly simplifies JavaScript programming. Ajax, an acronym for Asynchronous JavaScript and XML, is a group of interrelated web development techniques used on the client side to create asynchronous web applications. With Ajax, client-side web applications can exchange data with a server asynchronously in the background without interfering with the display and behavior of the existing page. Flot is a JavaScript plotting library for jQuery, with a focus on simple usage, attractive looks and interactive features. Generally, it supports all browsers that support the HTML5 canvas tag. In our design, we thought about using Flot which can be used to visualize sensor data in both static and dynamic real-time graphical displays. In the real-time display mode, instead of refreshing and redrawing entire web page, Flot provides the capability to only update the chart with new data that are fetched periodically from the server. Ajax and jQuery are used in our design to feed the Flot charting functions with continuous flow of data from the MySQL database on the server via the web services written in PHP. The data is serialized into the JSON (JavaScript Object Notation) format to be communicated between server and client. With the data access utility, users can download and view sensor data stored in the database conveniently. also monitor the sensor data updates in real-time using the real-time display utility. With the system configuration utility, authorized users can configure the sensor network system with a number of global and node-level settings, including measurement period, which sensor nodes to measure, and which sensor to measure. The configuration requests generated by users on the web interface are sent asynchronously to web service on the server and the requests are inserted into a command table in the MySQL database. The gateway application monitors the network command table periodically for any configuration updates to be sent to sensor nodes. The command table is used to share data between gateway application and web application. Such a design decouples gateway application and web application and greatly simplifies the inter-process data sharing problem. In general, environmental monitoring applications are tolerant to delays in the order of seconds or even minutes, so that the amount of delay introduced by such a method is acceptable in our targeted application scenarios. The sensor nodes and base station can work independent of web application. Authorized users can directly access the data and command tables in the database and remotely configure sensor nodes from Raspberry Pi by logging into its Linux system. On the other hand, the web application can be updated without interfering or interrupting sensor nodes and the gateway application residing inside the Raspberry Pi. Such a design makes it very convenient to perform application-specific customization and revision of the system.

4.12 Web Application API's

The web application used in this project functions through the use of several API's (Application Programming Interfaces). Each API is in charge of carrying out a process that can assist users of the app in monitoring the mesh system and implementing commands when manual interaction with the node system is required. The API is also responsible for interconnecting the different parts of the server to each other.

4.12.1 Login API

The login API will consist of a login screen that takes in the user's username and password before allowing entry into the program. The login screen will have a sign in and sign up button. For signing in, the API will ensure that only usernames and passwords that have already been registered can access the application's functionalities. When signing up a new user, the registration screen will ensure that emails have to be entered in the correct format (example@example.com) and that the password corresponding to a user is at least 8 characters long. Since the watering conditions of a property with grassland on it should only be alterable by the persons in charge of the property, the web application will only store one username and password. Users will have the option to change their username and password and submit a new password, should they forget their old one, using email confirmation. Code 4 is the PHP login code for the web application, containing valuable information such as the URL of the server we will be using to host our web application when it is online and functional.

```
<?php

        $con = mysqli_connect("groupGdb.co9dcxo307nm.us-east-
2.rds.amazonaws.com", "groupGDB" , "groupG" , "groupGDB");

        $username = $_POST["username"];
        $password = $_POST["password"];

        $statement = mysqli_prepare($con, "SELECT * FROM Testing WHERE
        username = ? AND password = ?");
        mysqli_stmt_bind_param($statement, "ss", $username, $password);
        mysqli_stmt_execute($statement);

        mysqli_stmt_store_result($statement);
        mysqli_stmt_bind_result($statement, $userID, $username, $password);
        $response = array();
        $response["success"] = false;

        while(mysqli_stmt_fetch($statement))
        {
                $response["success"]= true;
                $response["username"]= $username;
                $response["password"]= $password;
        }

        echo json_encode($response);

?>
```

Code 4- Login PHP Code

Depicted above is the code that not only connects the web application to the URL, but also makes few calls to the database for accessing login credentials. Thus, making the network secure from hacking and misuse.

The login API was ultimately left out of the web-app since hosting our software locally is secure enough for the operation that our system undergoes.

4.12.2 Status API

The status API retrieves the status of each moisture sensor upon logging into the web application. If a sensor is active, the data it is currently possessing is retrieved and displayed on the web application screen. If the sensor is off, a “not available” reading is displayed. This API functions through the RF device attached to each sensor. When the status screen is being displayed on a user’s device, the most recent moisture readings are then available. Users also have the option of refreshing the status page to ensure that the most up-to-date readings are being displayed to them.

4.12.3 Search API

The search API will allow a user to sort the sensors in current use by number, activity status, and location and will allow a user to search for a certain sensor. This command will simply consist of a search implementation into the web application. The search bar will have autofill capability, meaning the name of each sensor may come up in the search bar before the user is done typing in the full name of the sensor.

The search API was left out of the web application since the number of nodes used in testing our project (three) did not require us to have to search for any of them.

4.12.4 Add/Delete API

The Add/Delete API allows a user to add a new sensor to be used in the current system of sensors in the initial setup of a network and after a network of sensors already goes into use. When a user first uses the web application, the option to add sensors into the current system is available until the user decides to start using the system. After the system is already in use or has already been used, the user has the option to add more sensors into the current network, given that the network is not currently in use when the user decides to add a new sensor. Deleting a sensor from the system functions similarly to the add API. Rather than creating a new row for the new sensor when adding one, the row of the existing sensor in the database is deleted.

4.12.5 Operations API

The operations API controls the sensors via the web application, placing the current location of each sensor into the location database, and sending the location of each sensor that reads a moisture level lower than the desired level to the watering robot. Users have the option to turn sensors on and off, if they are connected to a power source, and collect a reading from the sensors through the press of a button. This functions by a signal being sent to the sensors when a user presses a command in the web application. The location of each sensor is placed into the location database through the user. Before the network is initially put into use, the user goes up to the desired location of each sensor and presses a button at these locations. A GPS system built into the web application tracks the current latitude and longitude of the desired location and enters it into the location database. If a previous location for the sensor already exists, the new location then overwrites it. Once the locations for all the sensors that are being used are loaded into the database, the web application tue emits the location of the sensors with low readings as an X, Y coordinate on the plane which is the field of grass.

4.13 Database

At the base of the communication between the sensors and the central hub lies the location database. This database keeps track of the positions of each sensor in the field plane. The positions of these nodes are entered manually through a web application once the sensors are placed in the ground. The choice to enter these locations manually, rather than have GPS functionality on each sensor, was chosen to decrease the cost of each sensor and the decrease the complexity of each sensor’s setup. Any data that the sensors report corresponds to their location, meaning the central hub considers the sensor readings to reflect the status of moisture in that area. When changes in the sensors occur, the sensors’ current status is then retrieved from the database and then altered to reflect its current standing.

4.13.1 Database Format

A MySQL database is used to store the locations, along with the level of moisture, temperature, and humidity at each location. When new readings are discovered, the old information in the database is swapped out with the new readings. The database is organized into four columns: the sensor number, its location, its level of activity (on/off), and its moisture reading. There is a row for each sensor. The information in the database is read to a user when accessing the web application to check the status of the sensors. Table 2 is an example of the organized database.

Table 3 is an example of how the data is stored, as you can see each record is identified by a unique Node ID, that is associated to each Node out on the field. The location is only updated at the initial setup and any other time the user wants to edit this field of data. The on/off state and moisture readings are updated periodically as new readings are registered from the Nodes. This database organizes the data, and looks up the ones in need through an associated API making the call. To the user this is all running in the background without much visual indication of what is happening.

Sensor #	Location	On/Off	Moisture Reading
1	28.5383° N, 81.3792° W	On	10%
2	28.5383° N, 81.3792° W	On	15%
3	28.5383° N, 81.3792° W	On	20%
4	28.5383° N, 81.3792° W	Off	N/A
5	28.5383° N, 81.3792° W	On	30%
6	28.5383° N, 81.3792° W	Off	N/A
7	28.5383° N, 81.3792° W	On	20%
8	28.5383° N, 81.3792° W	Off	N/A
…	…	…	…

Table 3 - MySQL Location Database Example

4.13.2 Database Server Configuration

MySQL uses the InnoDB storage engine by default. InnoDB is a fully transactional, ACID storage engine. A transaction is a batch of statements that must all succeed before any changes are written to disk. Code 5 is necessary for the Raspberry Pi to communicate with the database is shown below:

```
pi@raspberrypi $ sudo apt-get update

Hit http://archive.raspberrypi.orgwheezy InRelease

Get:1 http://mirrordirector.raspbian.orgwheezy InRelease [12.5 kB]

Get:2 http://mirrordirector.raspbian.orgwheezy/main armhf Packages

Hit http://archive.raspberrypi.orgwheezy/main armhf Packages

Ign http://archive.raspberrypi.orgwheezy/main Translation-en_GB

Ign http://archive.raspberrypi.orgwheezy/main Translation-en

[...]

pi@raspberrypi $ sudo apt-get install mysql-server

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following extra packages will be installed:

heirloom-mailx libaio1 libdbd-mysql-perl libdbi-perl libhtmltemplate-

perl

libmysqlclient16 libnet-daemon-perl libplrpc-perl mysql-client-5.5

mysql-common mysql-server-5.5 mysql-server-core-5.5

Suggested packages:

exim4 mail-transport-agent libipc-sharedcache-perl libterm-readkeyperl

Tinyca

pi@raspberrypi $ mysql -uroot -psecret

Welcome to the MySQL monitor. Commands end with ; or \g.
```

```
Your MySQL connection id is 42

Server version: 5.5.28-1 (Debian)
```

Code 5- Database Install Commands

4.13.3 MySQL Database Client

When your sensor network is connected to the Internet—or, more precisely, your slave is connected to the Internet—you can use your mobile phone to read the data from anywhere. Several applications that you can download for your smartphone and tablet allow you to access MySQL servers. For example, the MySQL Lite app for iPhone. This feature is more of a developer use, as the web-app was developing, we needed a way to make sure the data was being stored and organized correctly. For the users, the web-app ultimately serves this purpose in a more graphic manner. That is the only way that the necessary data is viewed and all the data for the internal process is stripped and preventing a potential misuse.

4.14 Useful Data Representation

Throughout the course of this project and throughout the testing of the nodes, there is a vast amount of data that the mesh network takes in and spreads within itself. One extra step we chose to take towards a more efficient solution to irrigation problems that current irrigation systems in place right now have is to represent the data collected by each node in a manner useful to promoting the efficiency of our project. Over the course of a month of the irrigation method that this project uses' performance, enough data that the nodes collect is sent to the computer in place for this system and helpful charts and graphs are to be produced for users of this specific irrigation, displaying statistics such as the average water that the area surrounding each node received over the course of a month. This is useful for showing potential customers the impact that switching from standard sprinkler irrigation to using the node system has on their water and power conservation. Depicted above is an example of multiple line graphs each representing a unique Node and is plotted against a moisture level vs. week. Over the course of the system lifetime, more and more data will be collected and plotted, giving a better feedback, and a possible machine learning feature to predict when to water based on the trend.

We ultimately chose to not incorporate this feature into our web application because the testing of this would have required several lengthy testing periods, such as several months, to accurately portray substantial data collection.

4.15 System Architecture

Understanding how the components are integrated together is just as important as the individual components. Like the abstraction, a system architecture details the interconnection of device and the protocols and platforms used for the system. Figure 23 details the links and how they were all achieved.

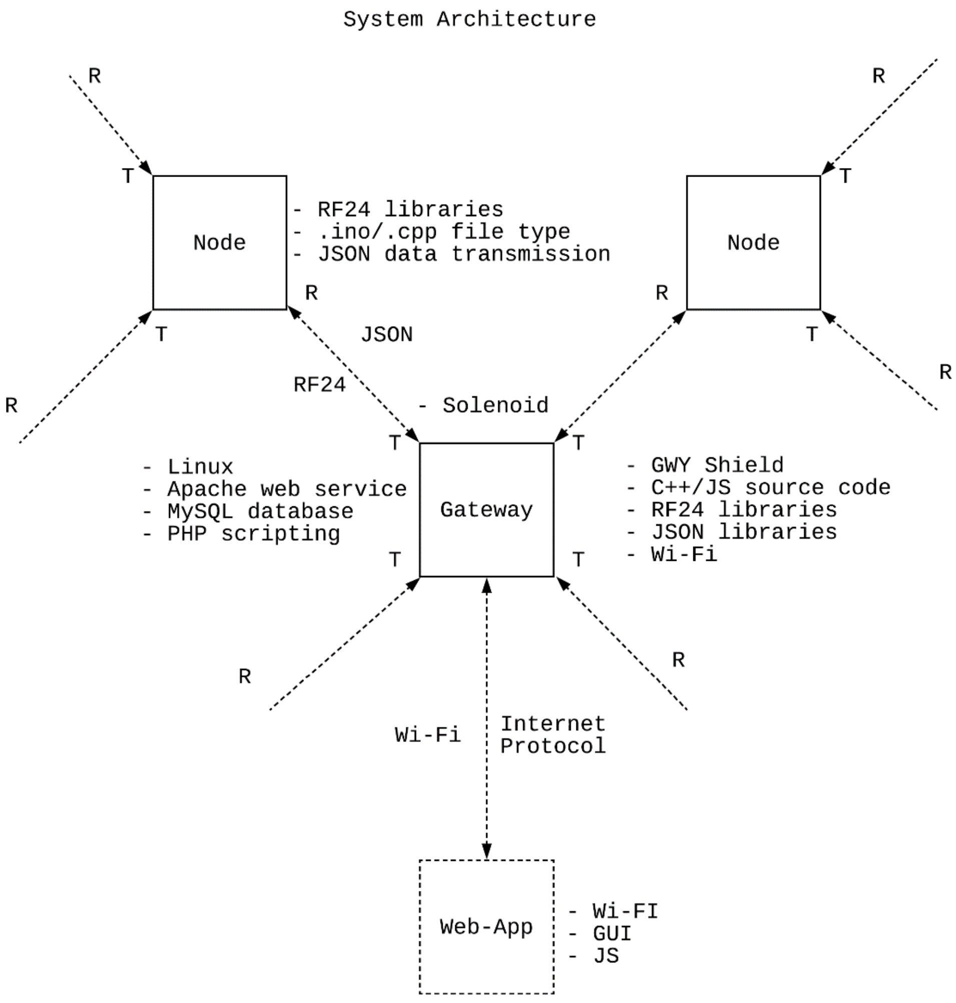


Figure 22- System Architecture

5.0 Standards & Design Constraints

With any design and application, constraints are always existent and therefore must be considered. Whether it be a client's specification, or safety hazard, engineers must work around these constraints and design a solution that still meets the requirements. When faced with design constraints, we have treated them the same given any field that they belong too. We have devised our preferred level of performance for every part that has been used in our project, from software to human resources available to us. We then did research to conclude what levels of performance we can actually obtain from resources given to us. These levels of performance that are realistically attainable were then sought after to the best of our ability. These constraints were all met due to our planning and testing throughout the second semester of senior design.

5.1 Standards

Standards are needed for interoperability both within and between domains. Within a domain, standards can provide cost efficient realizations of solutions, and a domain here can mean even a specific organization or enterprise realizing an IoT. Between domains, the interoperability ensures cooperation between the engaged domains, and is more oriented towards Internet of Things applications. There is a need to consider the life-cycle process in which standardization is one activity. The standardization bodies are addressing the issue of inter-operable protocol stacks and open standards for the IoT. This includes as well expending the HTTP, TCP, IP stack to the IoT-specific protocol stack. This is quite challenging considering the different wireless protocols like ZigBee, RFID, Bluetooth, BACnet 802.15.4e, 6LoWPAN, RPL, and CoAP. One difference between HTTP and CoAP is the transport layer. HTTP relies on the Transmission Control Protocol (TCP). TCP's flow control mechanism is not appropriate for LLNs and its overhead is considered too high for short-lived transactions. In addition, TCP does not have multicast support and is rather sensitive to mobility. CoAP is built on top of the User Datagram Protocol (UDP)

For IoT, this is of particular importance. A complexity with IoT comes from the fact that IoT intends to support a number of different applications covering a wide array of disciplines that are not part of the ICT domain. Requirements in these different disciplines can often come from legislation or regulatory activities. As a result, such policy making can have a direct requirement for supporting IoT standards to be developed. It would therefore be beneficial to develop a wider approach.

Standardization is a voluntary cooperation among industry, consumers, public authorities and other interested parties for the development of technical specifications based on consensus. The component layer basically reflects the devices like sensors and actuators, but also gateways and servers which run the applications. The communication layer is responsible for the data exchange between the components while the information layer represents the actual data. Standardization complements market-based competition, typically in order to achieve objectives such as the interoperability of complementary products/services, to agree on test methods and on requirements for safety, health and environmental performance. In order for key components of our system to work safely and properly, the power system which is a photovoltaic system was built and operated according to certain standards such as IEC62253, IEC61194, and IEC62124 [16].

5.2 Design Constraints

The initial idea of what the Node looked concrete at first, but it was quickly changed as more and more restrictions were stated. These constraints are determined by many factors ranging from basic economic impact to the potential ecological hazards that can be involved. In the project we focused mainly on how to build a small hand-sized object that can tolerate the dirt and rain. This need for durability pushed our design in this direction.

5.2.1 Supply Constraints

Ideally, the system used to check the water moisture of a 20 X 30 feet yard has to consist of about 20 sensors scattered throughout the yard in a perpendicular grid format. Since the sensors only pick up the moisture of grass spanning a couple of inches from the entry point of the sensors into the grass, each section of grass surrounded by sensors at each of its four corners, or a node, is watered based on the readings of its surrounding sensors. The project constraint facing this goal is that Guard Dog Valves can currently only provide us with a maximum of three sensors. This has altered our research so that only a 4 X 5 feet section of the original yard size was tested. Based on the result of this smaller test, an estimate of the results for a larger area of grass then came into fruition but may not accurately represent the same levels of watering that would occur in a larger field. Using a smaller portion of the size given to us by the sponsor can irregulate our test results. For instance, if the watering robot that waters the areas of grass that our sensor network deems in need of more water shoots out water at loads that are proficient for a 20 X 30 feet yard, but are excessive in a smaller field of grass, the sensor readings result in overwatering of the grass.

5.2.2 Power Constraints

A perfect irrigation system consists of sensors that report the exact levels of moisture of grass at any given second, and these sensors instantly disperse a signal when the moisture reading for a certain node is considered too low. These perfect conditions are unattainable by the network of sensors in this project due to the unrealistic power demands that such a system required. Because of the limited power that each sensor can consume in order to take up a minimal amount of space in a field of grass, the sensors only turn on in an hourly basis when the watering system is engaged, and only make readings at a faster time of every minute when water readings are lower than the desired moisture level. The small amount of time that the sensors are actually engaged is also determined by how large of a battery pack that can power the sensors. Too large of a power source results in the sensors taking up too much space on a field, interrupting the motion of the watering robot that is watering the grass, and providing a hindrance to moving vehicles that come in to close proximity with any of the sensors.

5.2.3 Economic Constraints

In addition to the ideal performance criteria that this irrigation monitoring system has been designated to meet lies the capabilities that weren't be reached due to the lack of enough financial assistance to have certain hardware and software acquired for this project. The hardware that is used in this project has been capable of accomplishing the goals set out by the group members of this project and the sponsor initially backing the project. However, the network of sensors could perform their duties even better than required, and to a higher standard, with more expensive technology, used and attained only by prominent communication companies. For instance, one type of hardware used in this project has been RF communication chips because of the relatively short distance that the sensors have had to communicate through in a typical residential backyard. The scope that we planned on spreading our project out to involves large acres of commercial land such as golf courses and farms. The tremendous size of these fields compared to the one that is being tested in this project results in sensor nodes communicating with each other at distances of over 1000 meters. This required range for larger fields easily bypasses the range that the hardware used in this project can handle, which is only 10 to 100 meters at a time.

5.2.4 Environmental Constraints

The environmental constraints that MOIST has faced are the most minimal out of all the other constraints considered for this project due to one of the main goals being set out to be accomplished is requiring new irrigation systems to use substantially less water than sprinklers already use. That being said, even with the sensor system in the project replacing a sprinkler system, there was still more water used than necessary, even if this smaller amount has been significantly less than what sprinklers use up. This is due to the sensors not being able to be on and function at every given second. The sensors only turn on once every hour, so the moment that a patch of grass goes from having enough water inside of it to lacking water is in between the hourly check-ins that the sensors produce. The sensors also are not able to sense changes in water moisture levels immediately once a node that requires water has begun being watered. This may cause a patch of grass to be overwatered slightly until the sensor is up-to-date on the moisture level readings it gives out. Given these environmental constraints, there is less water consumption with the sensor system that has been put in place.

5.2.5 Social/Interdisciplinary Constraints

Throughout the completion of this project we have worked with two other engineering groups, a mechanical engineering advisor, our previous sponsor, and our professor for the Senior Design course. There are joint goals and missions that were set at the beginning of the course by all parties involved. There is only so much strict adherence that had to be followed when considering the requests of the other two engineering groups to build or design our project a certain way that meets their preferred criteria when our professor or sponsor asked us to carry out certain tasks in a manner that may not fully function with the other groups. Part of the challenge of this project was deciding what priorities we should focus on in terms of the satisfaction of the other groups and in terms of our professor and sponsor. The most important aspect of completing this assignment to the members of this group is receiving an above satisfactory grade on the report, presentation, and demo, and complying with the specifications of our initial design so that the parts we needed to get to accomplish our tasks were obtainable in an affordable manner. Satisfying the needs of the other groups we worked with is also vital to our success in this course due the increased potential that working with more people provides in terms of labor and efficiency. Positive collaboration between the groups was established so that our group doesn't receive any lower marks on the work we turn in for Senior Design due to poor ratings that our group members receive from students in the other interdisciplinary groups as a result of poor collaboration efforts amongst each party involved.

5.2.6 Political Constraints

In a time where key members of our government are in disagreement over the importance of water conservation and energy consumption, the political constraints of this project are centered around the necessity of an irrigation system that is more water-efficient than irrigation methods of the past. Although water is a resource that is far from being scarce, a lot of energy is consumed when water is pumped out to keep plants hydrated. This energy usage adds up in the long run. The irrigation system that this project introduces has the ability to greatly decrease the amount of water and energy used by not letting water run off as often as current irrigation methods do. Should the methods used in this project ever be carried out on a large scale in the future (ex. whole neighborhoods or golf courses), the political constraints that ensue involve members of our nation's government approving of the use of such an irrigation system that is more efficient than current ones, but also bring additional costs to taxpayers' pockets for the installation of these new systems. Trust and privacy are the cornerstone of our society. Users' privacy concerns about the accessibility and use of information captured by IoT devices and sensors is an important challenge, and users need to be assured that the data collected does not provide any information that might violate their privacy. They must be enabled to understand and manage and control the exposure of their private and sensitive data.

5.2.7 Ethical Constraints

The goal of this project is to create a more efficient form of irrigation than the standard ones that exist in our country today. A more efficient irrigation system deals with some constraints that previous irrigation systems that exist today don't adhere to. Current sprinkler systems that water endless fields of grass in our country are left turned on for a lot more time than necessary and cover a much larger area with the water they let out than they need to. Keeping this information in mind, one ethical constraint of this project, which is really the main goal that has been set out to accomplish, is wasting as little water and energy as possible when the mesh network is being used. This means ensuring that the data sent between the nodes and the central hub is as accurate as possible. This also means picking materials, hardware, and software to use for MOIST that are not too expensive and that don't require too much wasting of resources to create.

5.2.8 Health & Safety Constraints

The constraints that come along with this project in terms of the well-being of plants, animals, humans, and structures that the node system may come into contact with during its use are highly minimal due to the gentle nature of the contents of each node. There are no moving parts involved in the sensors taking in data and sending the data to other nodes or to the central hub. This results in the node operations being safe. Some constraints in this subject regarding the interdisciplinary project that MOIST is involved in as a whole include the safety and well-being of the field of grass that the AIV will be navigating through as our nodes give them the signal on whether the water near a node should be watered or not. To ensure that the grass under the AIV isn't damaged, the mechanical engineering group had to have designed the rover so that it isn't too heavy or intrusive. To ensure that the nodes aren't damaged when the area around each of them is being watered, the node encapsulation had to be designed so that water is unable to penetrate it and damage the hardware located inside each node. People and animals have to avoid the area of grass that is being watered whenever the node network is engaged to prevent accidents from slipping and falling. Plant life may also be in danger if the wrong signals are transmitted via the mesh network, resulting in a certain area receiving too much or not enough water and grass dying.

5.2.9 Manufacturability Constraints

Since this project involves a lot of water usage, a priority in the development of the nodes of our mesh network had to be a sturdy case for the node that prevents water from damaging the hardware inside of it. In addition, a method for placing these nodes in the ground together and have all the hardware work together well had to be developed as well. With these propositions in mind, the constraints that were faced in terms of creating the necessary parts to house all the hardware used in this project involve the price of the materials we wanted to use, the amount of time we had to build the most efficient and sturdy node case possible, and the students in this group's limited knowledge and experience creating encapsulation that can house multiple devices together. In terms of cost, there is only a limited amount of money that can be obtained from our own funds to purchase the necessary items for the project. This senior design project is only limited to two semesters worth of time, so our allotted time to design the most beneficial node encapsulation possible will have an impact on how well the encapsulation performs. Although one of the students working on this project has his own 3D printer and has experience with his own projects, the encapsulation that is covering the nodes requires more expertise regarding its nature and had to be larger than any other objects this student has previously produced on his own time.

5.2.10 Sustainability Constraints

The major areas of concern for our project in terms of sustainability are the continuous access to Wi-Fi that our mesh network needs in order to communicate and send data between the nodes, and the supply of power that is needed to power the hardware used to have the nodes be turned on during the network's usage. In order for each node to communicate with each other, there has to be wireless communication device engaged across every node that is being actively used to monitor a particular field of grass's moisture levels. This means that RF communication has to be maintained whenever the mesh network is in use, and that the communication between each node has to be constantly checked on to ensure that the correct data is being transmitted between each one. For power, a minimum of five watts has to always be supplied to the PCB's used in this project when the mesh network is running so that data is being accurately sent and retrieved throughout the system. The nodes also have solar panels incorporated into them, so the nodes are placed in areas where they receive enough sunlight to produce the supplementary power that accompanies the battery power already powering the hardware in each node and the central hub.

6.0 Project Prototype Build

To test our system in its full potential, a few iterations of prototyping had to be developed. Whether it is to test the individual components, or the final design, every iteration is crucial in its entirety. This phase of the project is where the ball gets rolling. Actual development of operations occurred. Here we built a prototype of our finalized design and tested it against its limitations. If it failed, we went back and troubleshooted the issue to fix it up. Sometimes when the unexpected outcome was severe, a whole redesign was required. To prevent excess redesign, with time and cost involved, prototyping was broken down into smaller phases. We started by testing the individual components on a development board and a breadboard, this avoids soldering and board design. Once code works on the development board then we design the PCB. We then tested it again and if any issues occurred we troubleshooted back on the development board to implement on the next board design.

6.1 Hardware Prototype

Before the project can be tested and put into use, a prototype of all the different parts we had to use must be built so that we can test any conditions we desire before moving on to producing our final products next semester. These prototype materials are not of the best quality that we were capable of producing a product due to time, financial restrictions, and, more importantly, the purpose of the prototypes being solely a testing product not intended to be placed on market shelves in their current formats. Prototyping is an optimal way to save money and resources when having a finalized product in mind. If we were to skip the prototyping process and skip straight to the final versions of our network system, we would have wasted money and time with a product that is not optimally functional, and going back in the design process to do testing on what failed would then result in more money and resources being wasted than creating well-tested prototypes to begin with.

6.1.1 Development Board

Our first executed prototype was just for testing the individual strategic components primarily the power, RF, and sensor systems. Programming the logic was also taken into account. We used an Arduino UNO R3 shown with a solderless nRF24l01+ attached by jumper wires, and a 9V battery for testing the range in Figure 25 we tested these components without wasting efforts on the PCB design.

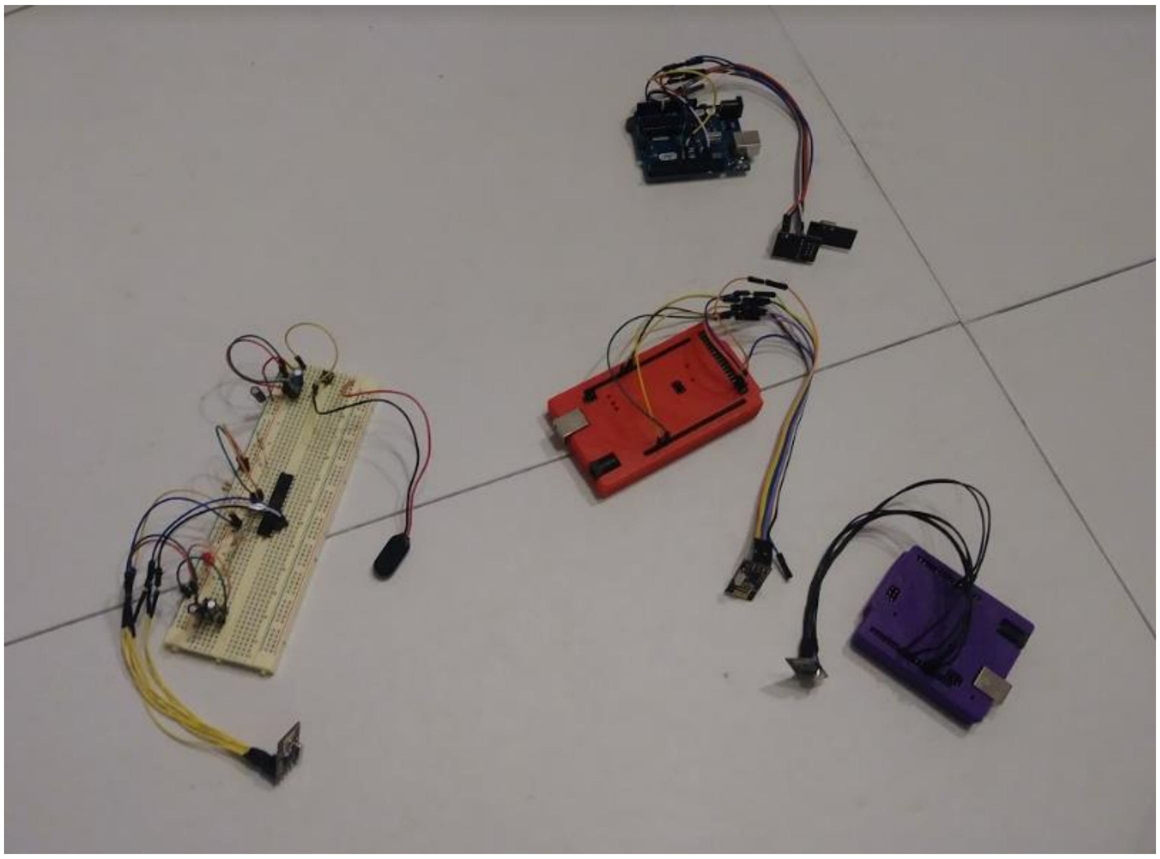


Figure 23- Arduino Development Board

As we successfully used the RF module with another development board, we moved on to focus on other strategic components like power setup and sensors, adding more and more pieces without soldering a board. The firmware was then constantly updated to include features as we added to the development board. The Arduino has an easy programmable interface for this reasoning, and we used that to our advantage to discover what the potential of our machine was capable of.

6.1.2 Node PCB

After finalizing our test results and calibrating our components, a singular PCB design was then made for actual use. The advantage of creating a PCB was to prevent loose wires and components as well as rid unnecessary material for simplicity, size, and cost. Figure 26 is how the PCB was designeded for production release and how it was laid out by this schematic, all the wiring correlating to actual copper traces joined by common net name.

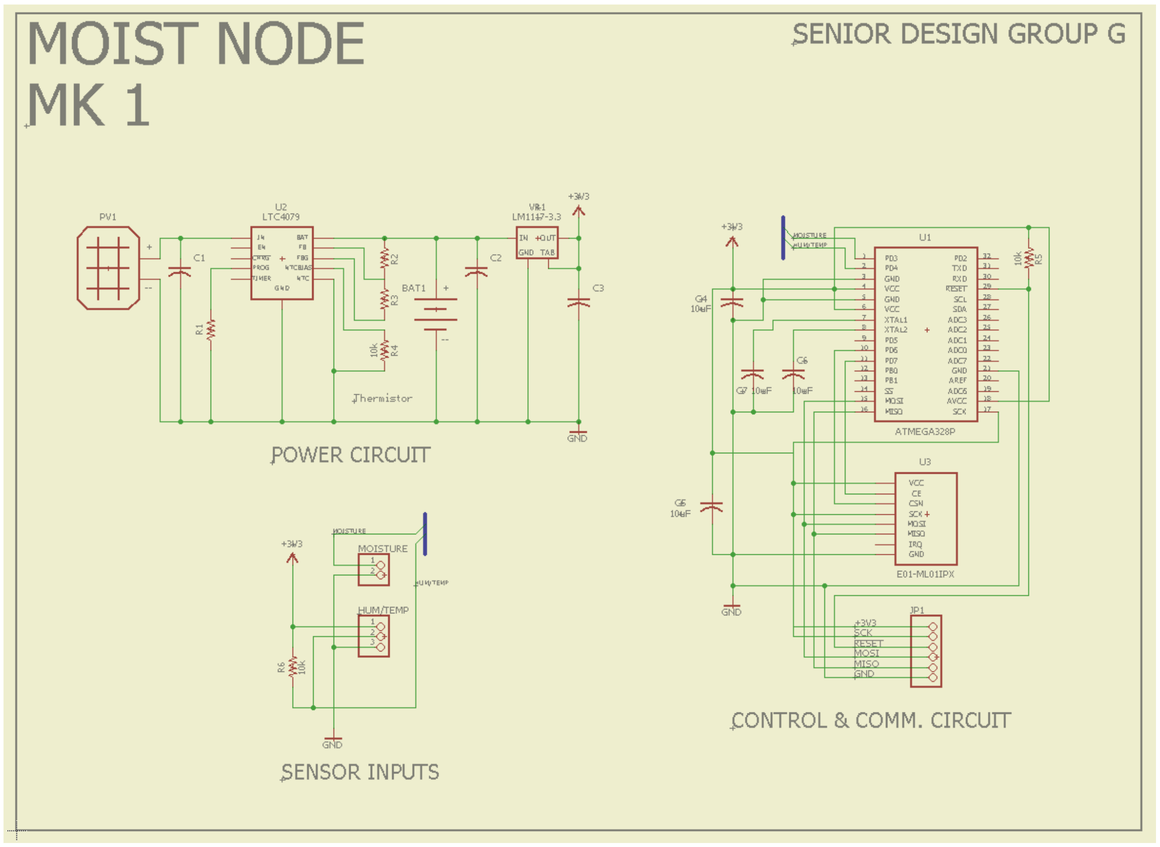


Figure 24- Node PCB Schematic

The Schematic details three sections of sub circuit, that are grouped based on functionality and common junctions. The power circuit includes a port for a pair of solar panel wires, the ECO-WORTHY 10W, a port for a pair of battery wires, the LTC4079, and LM1117 surface mount components with their auxiliary capacitors and resistors. The second circuit is the control and communications circuit, this consists of the ATmea328P, surface mount variant, the E01-ML01IPX with their auxiliary capacitors and resistor, and a port for programming the microcontroller. This circuit was grouped together based on similarity of the shared SPI communication lines.

The schematic was designed on a software called Eagle, it is a simple ECAD interface used for creating PCB outlines and generate the manufacturing files. The reason this choice was made because of ease of use, the online support, and proficient experiment with the software an electrical engineering team member has.

6.1.3 Gateway PCB

The second PCB that was designed is a shield that stacks on to the Raspberry Pi and offers the RF capability to connect with mesh network. Figure 27 depicts the schematic for the PCB design

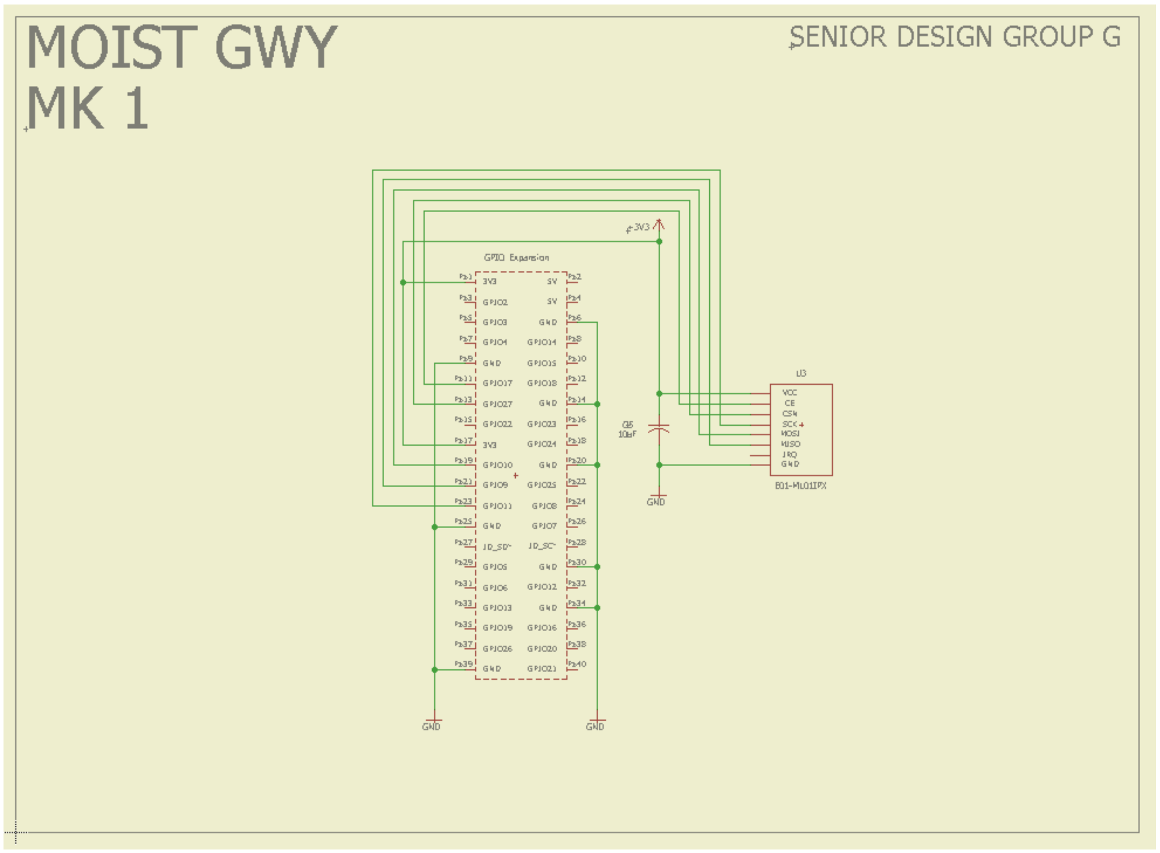


Figure 25- Gateway PCB Schematic

Although much simpler than the Node PCB design, it has constraints of its own. The PCB had to fit the dimension of the Raspberry Pi in order for it to properly connect without any obtrusion involved. This creates the Gateway and then implements the firmware and APIs with the rest of the mesh network. This schematic was designed using Eagle as well.

6.2 Software Prototype (Coding Plan)

The web application was be coded in JavaScript, CSS, and HTML languages, similar to most websites on the internet today. After researching different stacks to use for the development of the web application, the LAMP stack was chosen due to the familiarity that the computer engineering student working on this project already possesses creating websites using LAMP. The nature of the web application is fairly simple, as most of the complexity in this project can be found on the hardware side of making the communication between the sensors and the central hub possible. The web application allows a user to create a username and log in through a secure server, add sensors to the network that is then placed on a field, check the status of each sensor, search for specific sensors, delete sensors, and turn the sensors on or off. The web application takes the format of a website loaded on an Amazon server. To communicate with the sensors, a Raspberry Pi that is connected to the central hub has the database of the web application loaded into it. This information is then constantly updated whenever the system is in use.

6.3 Enclosure Prototype

The need for an enclosure was necessary for housing our electronics in the field. The design process for the enclosure had to be dictated by the PCB size. An initial design was then started and revised as this project is progressed until reaching a final option.

6.3.1 3D Print

The sensors we have used have already been built to withstand typical inconveniences and damages that keeping something outside may inflict upon an object. In order to improve the durability of these sensors, a 3D model was designed that offers more protection to these sensors than they already contain themselves. This model is designed using AutoDesk Inventor to produce a lightweight, but durable, enclosure made out of PLA, a biodegradable substance that further insulates the sensors from damage.

Figure 28 is a model generated on the Autodesk Inventor software application of the general shape envisioned as the enclosure. This design had to accommodate the need for a Node that was originally designed to be buried yet exposing the solar panel, antenna and the sensors attached to it. As we modified this model we had to take those factors and the shape of the PCB into thought. Constraints of design was emphasized heavily as this was more of a mechanical issue than an electrical one.

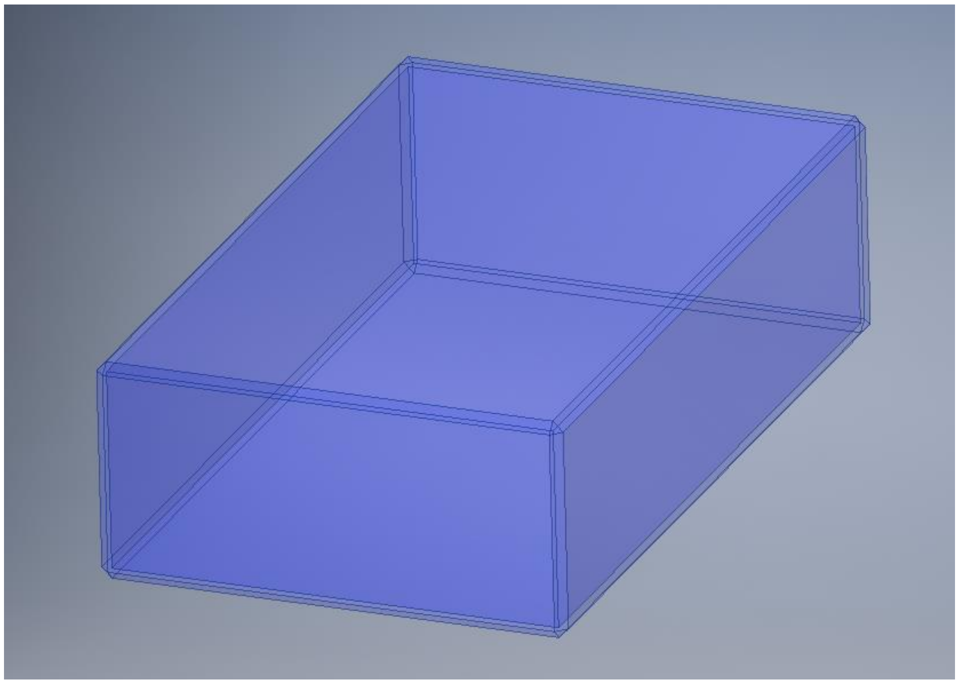


Figure 26- Enclosure Model

The 3D print serves for demonstrating our idea of MOIST because of the material it used may not have been sufficient to withstand weather extremities.

6.3.2 Plastic Mold

Although the 3D printed cover that was implemented on the sensors would have been practical for our testing purposes and useful for protecting them from harm, 3D printing for any material was meant only to be a prototype. If our network system were to ever advance to a commercialization level and become a professionally used product, the PLA material used in the 3D printing process would have to be replaced with a professionally produced plastic cover that provides longer-lasting durability than the environmentally-friendly material used in the 3D process.

The file used for modeling the enclosure on the 3D printer is the same file used to generate the G-code required to machine a mold. This mold is then injected with molten hot plastic that quickly solidifies and takes shape of the model. This is an expensive procedure and requires heavy machinery to produce. For a senior design project and in the realm of knowledge, resources, and deadlines a 3D print of this model has sufficed in our demonstration of the project.

6.3.2 Final Encapsulation Design

After testing out different methods for covering the Nodes, we decided that using a plastic box a little larger than the Nodes' PCB was the right fit for our project. An encapsulation that is too large would be problematic for a user of our system because these Nodes ideally can't take up to much space on a field of grass or else they interfere with activities that are performed on said field. Figure 29 contains the final encapsulation that was chosen when it's open and closed.

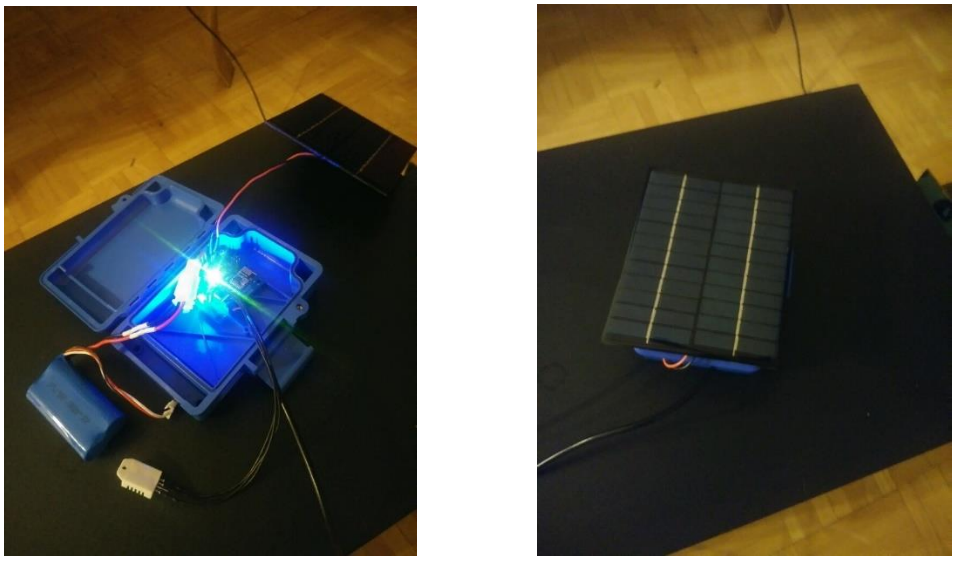


Figure 27 - Final Encapsulation

7.0 Prototype Testing

The research portion of this project relied heavily on comparing current hardware and software on the market and deciding which ones best suit the tasks our network system had to accomplish. However, testing the products we decided to incorporate into our project was just as important as the products themselves. Sometimes the products may have seemed to be the right fit on paper, but then behaved vastly different than what their expected outcome was. Here the testing is broken down into a few procedures to carefully analyze the result with minimum outlying variables. We started by testing the hardware, and its components like the power and the communication and the sensor, then the integration of the components, and then between a Node and a Gateway. Each step ensured that the previous setup was working.

7.1 Hardware Testing

Hardware testing was simply comprised of testing out the different hardware components used throughout this project to ensure that they all were in working order. We used minimum code here to eliminate unnecessary variables that can lead to varying results. We then built this open as we started testing for software.

7.1.1 Power Testing

This testing part functions as confirmation of the research and final decisions on what hardware we were to use from the early stages of this project based on how smoothly the hardware chosen runs and whether it actually ended up using less power-consuming than their runner-up choices. The operations for this setup were very straightforward. Wire the components and measure voltage against the duration of sunshine and then observe the trend over time. This ensures that the voltage level is stable and power is being generated and stored. Measuring power consumption here also gives us feedback as to how the PCB performed in the field.

7.1.2 Sensor Testing

Similar to the soil temperature sensors, the prospective commercial user would want to know the amount of moisture within every 15 inches of the soil. With the droughts that commonly go on, the efficiency of the roots absorbing water is key to prevent over usage. The sensor we used in order to determine was the ITEAD Electronic Brick Moisture Sensor. These sensors were resistive based so essentially, they determine the resistance between the two metal prongs that associate with a certain moisture level. This sensor was unique because an analog or digital output could be read. The digital output would just indicate if the soil was wet or not while the analog would give a particular reading based on the amount of water. A total of 5 sensors were then used in order to calculate the reading. In terms of calibration, after a series of trials, the calibration was very tedious because the analog reading varied. The soil temperature sensor adopts DS18B20 provided by Maxim integrated which is a lower-power consumption digital temperature sensor. The temperature measuring of DS18B20 is −55 to +125°C, and the precision is 0.0625° C, 9–12 bits A/D. Owing to its small size, it has been concluded that it can save many lead wires and much logic circuitry. It requires only one pin for communication and can be powered with 3V. It consists of three wires. The black wire is connected to the ground, the red is connected to the voltage common collector (VCC) and finally the white is connected to digital-analog converter (DAC) channel pin.

The DHT22 is a simple digital sensor that produces digital signals. It requires a single resistor to pull up from the data pin to voltage. *Pull up* in this case makes sure the data value is “pulled up” to the voltage level to ensure a valid logic level on the wire. We then jump right in and connect the hardware The hardware required for this project includes an Arduino, a DHT22 humidity and temperature sensor, a breadboard, a 4.7K Ohm resistor (colors: yellow, purple, red, gold), and breadboard jumper wires. Next, the power is connected from the Arduino to the breadboard. Use one jumper wire to connect the 5V pin on the Arduino to the breadboard power rail and another for the ground (GND) pin on the Arduino to the ground rail on the breadboard. With these wires in place, we were then ready to wire the sensor. The library for this code can be found on GitHub [69].

DHT22 sensor had its own protocol. only a small part of that protocol: the part you must include in your sketch. However, there is much more going on in the DHT library.

Error Code Cause or Recommended Action

DHT_ERROR_CHECKSUM Checksum failed. Could indicate read/send errors. Values read may not be accurate.

DHT_BUS_HUNG Error reading from sensor. Try again at the next cycle.

DHT_ERROR_NOT_PRESENT Cannot communicate with the sensor. Check the wiring.

DHT_ERROR_ACK_TOO_LONG Too long between data sent and acknowledgement. Try again at the next cycle.

DHT_ERROR_SYNC_TIMEOUT Synchronization failure during read. Try again at the next cycle.

DHT_ERROR_DATA_TIMEOUT Timeout sending data. Try again at the next cycle.

DHT_ERROR_TOOQUICK Wait at least 2 seconds between reads.

DHT_ERROR_NONE No error. Read values

Code 6 is used for DHT22 readings:

```
#include <SPI.h>
```

```
#include <DHT22.h>
#define DHT22_PIN 7 // DHT2 data is on pin 7
#define read_delay 5000 // 5 seconds
DHT22 myDHT22(DHT22_PIN); // DHT22 instance
void read_data() {
DHT22_ERROR_t errorCode;
errorCode = myDHT22.readData();
switch(errorCode)
{
case DHT_ERROR_NONE:
char buf[128];
sprintf(buf, "%hi.%01hi, %i.%01i\n",
myDHT22.getTemperatureCInt()/10,
abs(myDHT22.getTemperatureCInt()%10),
myDHT22.getHumidityInt()/10,
myDHT22.getHumidityInt()%10);
Serial.print("Data read:");
Serial.print(buf);
break;
case DHT_ERROR_CHECKSUM:
Serial.print("check sum error ");
Serial.print(myDHT22.getTemperatureC());
Serial.print("C ");
Serial.print(myDHT22.getHumidity());
Serial.println("%");
break;
case DHT_BUS_HUNG:
Serial.println("BUS Hung ");
break;
case DHT_ERROR_NOT_PRESENT:
```

```
Serial.println("Not Present ");
break;
case DHT_ERROR_ACK_TOO_LONG:
Serial.println("ACK time out ");
break;
case DHT_ERROR_SYNC_TIMEOUT:
Serial.println("Sync Timeout ");
break;
case DHT_ERROR_DATA_TIMEOUT:
Serial.println("Data Timeout ");
break;
case DHT_ERROR_TOOQUICK:
Serial.println("Polled too quick ");
break;
}
}
void setup() {
Serial.begin(115200); // Set the serial port speed
}
void loop() {
delay(read_delay);
read_data();
}
```

Code 6- DHT22 Readings

We used the same mocking test as the Integrated Water Monitoring System project group who built the Triton system moisture sensor. The purpose of this test was to correlate a quantity of water to resistivity readings from the moisture probe over time, as well as compare it to the readings from a moisture sensor currently in market. The soils tested were Miracle Gro Potting Mix and Miracle Gro Garden Soil; the moisture probe used for comparison was the Digital PLUS Moisture Meter. The experiment performed was designed to formulate a relationship between water saturation in soil to moisture levels read by the prototype. This was done by inserting the moisture probes in a controlled amount of soil and pouring varying amounts of water onto it. In order to carry out this test, 549 grams of Miracle Gro Potting Mix were placed into a plastic plant pot, and the moisture probes were inserted. Then, 21 grams of water were then poured into the pot. After waiting about five seconds for the water to permeate through the top layers of soil, moisture measurements from the probe were collected. Since the probe code is designed to take measurements every five minutes, a magnet was then used to activate the Halifax switch on the probe in order to allow measurements to be taken more often; measurements on the comparison sensor were also recorded. Measurements were taken intermittently for about 5 minutes, and then 41 grams of additional water was added to the pot; measurements were collected again for about 5 minutes. Finally, 88 grams of water was added and measurements were taken. This procedure was then repeated with the Miracle Gro Garden Soil. The moisture probe was measuring the moisture of the Miracle Gro Potting Mix as a controlled volume of water was added [13].

Another testing that was completed was weighing out dry soil, including adding sufficient water to determine if it was moist. We measured the difference of weight, and then let it dry out over a span of few days, and finally measured for the change in weight once more. This, with the addition of resistance, painted a trend of moisture levels and the threshold we had to set up.

7.1.3 RF Testing

To further understand our network implication, and how the RF behaves, an extensive testing was conducted with the RF modules. WE had multiple devices wired to the RF module and had them constantly relay messages. We did this by by dynamically moving the devices out of range and into obstruction. With the test method we studied the behavior of the communication to adapt our network on the field.

We then setup a debugger window that connects to a Node and observes it as it transmits and receives from a Gateway, then introduced a second Node to see the outcome. We then disconnected to Gateway and observed if the network failed. After that we tested for range between devices.

Our result was that the range between our Nodes and Gateway were only capable of reaching two-three feet due to the lack of an external antenna being connected to the Gateway. This would've allowed for a larger range within the whole system.

7.2 Software Testing

To ensure that the web application was correctly synced with the sensors when they were turned on and when they are in their assigned locations, and to ensure that the buttons and options in the web application performed their duties with no bugs, a series of tests were done on the functionality of the web application including node location testing, operations control testing, and status testing.

7.2.1 Node Location Testing

Before the watering maintenance of a field takes place, the web application controlling the sensors ensures that each sensor is in the location that is loaded for them in the location database.

7.2.2 Operations Control Testing

To test out the functionality of the controls that our web application is able to carry out, we sent the commands through the gateway to the sensors using the web application and observed if the right information got reported back. For example, if a sensor is currently on and connected to the mesh network but is not taking any readings in of water moisture levels at the moment, the command of retrieving the current readings of the sensor is initiated through the web application. If nothing is returned or if the data that is returned is an inaccurate representation of the moisture levels being requested, than the testing of the operations was considered a failure. In addition, if the current reading of a sensor is requested of a sensor that is currently turned off, than the results retrieved should be that the sensor reading is unavailable. A return of an actual percentage resulted in a failure as well.

Our software testing proved beneficial and resulted in all the bugs in our code being worked out so that all the desired commands are carried out as we initially intended them to be.

7.2.3 Status Testing

To test out if the correct status of each sensor that is currently connected to the mesh network is being displayed, the status of each sensor that has been added to the network has to be visible when a user logs on to the server and accesses the user interface connected to the network. If the status of activity correctly portrayed the actual status of each sensor, either on or off, then the testing of this sensor was considered passed. If a sensor that is turned on displays a signal of off to the user, or if the sensor is off but displays a status of on to the user, then the status test for this sensor was considered failed. When this test is failed, probable circumstances that may have occurred could be that the sensor’s power button is broken and won’t turn off correctly, or that the sensor is off and is dysfunctional, meaning it can no longer turn on without its hardware components being repaired.

The status testing resulted in accurate data always being collected from the Node sensors and in the data being relayed to the Gateway.

7.3 Durability Testing

While sensors are placed in the field and are in operational mode, the length of time they can withstand the weather conditions that a typical outdoor Summer time Floridian climate inflicts on the network of sensors will be the ultimate test of durability for the network of sensors. How long, given a full charge of battery in all the hardware components used in the system, that the sensor network can run before the battery for one of the components dies, was the most valid indication of the whole system's durability in terms of time. In the terms of actual physical durability of the components of the sensor network, the conditions that the hardware components were all able to withstand during the testing and actual use of our sensor network determined the overall strength of the network's hardware, and whether or not the materials chosen to be used in the project should have been swapped out with materials that can last for longer periods of time than the ones initially chosen. The amount of time and uses that the software side of the network withstood before previously unforeseen bugs in the code were discovered also contributed to the overall durability of our project

8.0 Administrative Content

The execution of MOIST flowed smoothly throughout the whole process due to the level-mindedness of all the parties involved in completing tasks for this project. This style of execution could've gone several different ways and other groups of students doing the same project as us could have done it in a completely different fashion based on their experiences with their professor and their own personalities. The methods, products, and purchases to get this project done were completed in the way they were because of the milestones and budgets that our group set and re-established for ourselves throughout the course of our research.

8.1 Milestones

Our first milestone for the end of Senior Design I was to have documentation ready that explains our objective, research, and design process for the sensor network we created, and costs associated with the steps that were carried out during the whole process. Our second milestone was to have a prototype built and tested that confirms the research we have completed all semester. If the testing conducted after our research has been done were to have resulted in faults being found that were not considered during the initial research, the prototypes created were then redesigned to fit the needs of the actual system we implemented in a real-life environment. Ideally, we were to have a full-on working prototype ready before the processes of Senior Design II were even commenced. This actually didn't occur due to hiccups we encountered while working on the prototypes. The prototypes we created were built but were not extensively tested on due to further research made as the first semester started to come to an end and further research was done throughout Senior Design II.

Our milestone for Senior Design II was to build a final project that meets the requirements set for our project during Senior Design I, ensure that the final project meets or surpasses the objectives set during the planning stage, and fix or discuss in detail the issues that were encountered during the actual implementations of our design. At this stage of the project, there were still several factors we needed to consider as we delved deeper into the designs of our hardware setup, software applications, and node placement in a network. Our objectives for Senior Design II were to have expanded if it was agreed upon by the professor and the students in this group that not enough work was being done in our project to constitute the receipt of a satisfactory passing grade for the students involved.

Our interdisciplinary milestone for both semesters was to work in unison with the electrical/computer and mechanical engineering groups so that the work we accomplished in each of our separate research and reports were linked to each other and could be referenced when discussing the overall intentions of the project we were working on ourselves. With this milestone we also aimed to focus primarily on our own research and work, and view the compatibility of our designs with those of the other groups we worked with as a second priority in the long run.

Table 5 & 6 lists all of the milestones we set up and persevered through throughout the two semesters. Every week the group of students met up at least once between the students solely involved in the mesh network and then at least once with all of the students in the three groups involved in the interdisciplinary project. Every week, the milestones and completions that our group had made involving MOIST were recorded and assessed. Judging on how much progress was made every week, and whether or not more progress or less progress was made every week than planned, decisions on how much work should be added or removed from the following week's workload were made.

Following this detrimental guideline, the amount of weeks where more than enough work was completed and the amount of weeks where not enough work was completed balanced out, keeping the progress of our project on a constant increase throughout the whole Spring semester. At the beginning of the semester, a great deal of uncertainty clouded the group of student's working on this project's judgements regarding whether the proper amount of research to carry out our project was doable within one semester's time frame. Individually, this skepticism may have been proven correct. However, working together as a team proved to be much more impactful to completing the tasks that MOIST brought up to the table than previously imagined. Splitting up the tasks that needed to be carried out to discover the right plan for carrying out the design and testing of this project allowed us all to spend less time focusing on all aspects of the project and more time focusing on individual areas. After conducting research separately, combining our thoughts on what the best way to implement the design of this project would be was as simple as all of the students involved in the creation of MOIST to review each other's' research and use the methods that we each found individually that function the best with the methods that the other students discovered on their own.

Week	Milestone Accomplishments
Jan. 19	Formed group.
Jan. 26	Finished Divide & Conquer
Feb. 2	Researched initial design
Feb. 9	Discussed design of AIV with other groups
Feb. 16	Contacted Sponsor
Feb. 23	Finalized design parameters
Mar. 2	Picked out strategic components
Mar. 9	Finished Divide & Conquer 2

Mar. 16	Spring break
Mar. 23	Learned about mesh networking
Mar. 30	Researched Triton sensors
Apr. 6	Finished 45-page draft of Senior Design documentation
Apr. 13	Acquired Triton sensors from sponsor
Apr. 20	Finished 90-page draft of Senior Design documentation

Table 5 - Milestone Chart EEL 4914

Week	Milestone Accomplishments
Aug. 20	Began putting our researched products to the test
Apr. 27	Decided on final software to be used for web application
Sep. 3	Began working on initial CDR
Sep. 10	Integrated MySQL database insertion into web-application
Sep. 17	Present initial CDR to professor and class
Sep. 24	Received all parts needed for project
Oct. 1	Finalized PCB design and put in order
Oct. 8	Received final PCB's
Oct. 15	Combined assembled parts together
Oct. 22	Enabled communication between Gateway and Nodes
Oct. 29	Performed midterm demo
Nov. 5	Finalized conference paper
Nov. 12	Began final testing of our system

Nov. 19	Gathered professors for demo panel
Nov. 26	Performed final presentation/demo
Dec. 3	Turn in final documentation and finish SD2 website

Table 6 - Milestone Chart EEL 4915

By jotting down what we did, we checked our overall progress and determined the pace we progressed at. This way we could make up if we slack or relax if we are ahead of the game.

8.2 Budget & Finance

Since this project was relatively inexpensive to other projects we have come into contact with, budget and finance was not a huge concern. Although we kept the cost of parts in mind as we researched components and overall cost of the bill of materials, our main concern was meeting the satisfied requirements while abiding to all given constraints. For example, when we needed to sacrifice cost for extended range or battery life, by all means we had no objection.

8.2.1 Bill of Materials

The bill of materials is a list of all components that is required to produce a single Node and a single Gateway and all their cost. In a production company this list would help sort what is needed across multiple products and quantity per unit.

Table 7 is our bill of materials. The majority of the costs that purchasing the required products for this project imposed on us were from the Node and were ultimately based on the size of the mesh network. Since this system is scalable the bigger the field a potential client has, the higher the cost of the network. Our aim was to find a low-cost solution by minimizing the number of required Nodes and cost per Node. To minimize these costs, more accurate tests of the equipment that we are in possession had to be taken. The initial plan of the Node layout was to have more than we may need in order to get the most readings possible when examining the grass they are placed on. Further testing the design then reveal that the same accuracy could be reached with less nodes/expensive parts.

Project Budget List			
Item	Quantity	Cost	Description
Node PCB (10)	1	$27.00	Custom Node circuit board
Gateway PCB (5)	1	$27.00	Custom Gateway circuit board
GPS PCB(5)	1	$5.00	Custom GPS sub board
ATmega328P	6	$1.46	Main MCU
LTC4079	6	$5.13	Charge Controller
LM1117-3.3	7	$1.14	5V regulator
LM1117-5.0	6	$1.14	3V3 regulator
nRF24L01+ mini	7	$2.99	RF module
VH400	3	$39.95	Analog moisture sensor
DHT22	3	$9.95	Digital humidity/temperature sensor
Resistors	54	$0.10	Passive element at various value
LEDs	18	$0.22	Light indicator at various colors
Capacitors	51	$0.10	Passive element at various value
Peizoelectric Crystal	6	$1.12	Crystal oscillator
Screw Terminal Block	6	$0.98	Removable wire connections
1"- Headers	19	$0.12	Pin-headers for wire connections
3x2 - Headers	7	$0.12	Pin-headers for wire connections
Female 40-pin header	1	$2.99	Female header for Raspberry Pi
NUZAMAS 2W 12V	3	$12.45	PV Solar Panel
CXAB 18650	6	$9.99	Rechargable Li-Ion Battery
Raspberry Pi 3	1	$35.00	Embedded Computer
Micro SD (32GB)	1	$6.68	Store Linux OS
Total Cost:		**$451.13**	

Table 7 - Budget Listing

9.0 Conclusion & Summary

MOIST has introduced many problems and challenges to the group of students working on this project that were never encountered in other parts of the undergraduate teaching curriculum. In order to persevere through the project and produce a satisfactory result of our diligent research and design, we had to face several issues that a project with such a broad deadline brought upon us, such as time-management, working with a group and with professionals in the engineering field to obtain certain knowledge and materials, and rethinking solutions to the problems we faced that didn’t quite work out to our satisfaction the first time around. Through the hardships that this project made us endure, we learned valuable lessons on professionalism that we can take with us for the rest of our lives and put into use immediately upon completing Senior Design and graduating.

10.0 Appendices

The glossary, permissions, datasheets, and references used for this report are included in this section.

10.1 Glossary

API (Application Programming Interface)

PCB (Printed Circuit Board)

PLA (Polylactic Acid)

RF (Radio Frequency)

MCU (Microcontroller Unit)

OS (Operating System)

IOT (Internet of Thing)

BLE (Bluetooth Low Energy)

NB-IOT (Narrowband Internet of Things)

HTML (Hypertext Markup Language)

CSS (Cascading Style Sheet)

DAC (Digital to Analog Converted)

GPS (Global Positioning System)

WSN (Wireless Sensor Network)

XML (Extended Markup Language)

JSON (JavaScript Object Notation)

EPA (Environmental Protection Agency)

M2M (Machine to Machine)

LoRa (Long Range Radio)

Wi-Fi (wireless fidelity)

TCP (Transport Control Protocol)

UDP (User Datagram Protocol)

HTTP (Hypertext Transfer Protocol)

IP (Internet Protocol)

CoAP (Constraint Application Protocol)

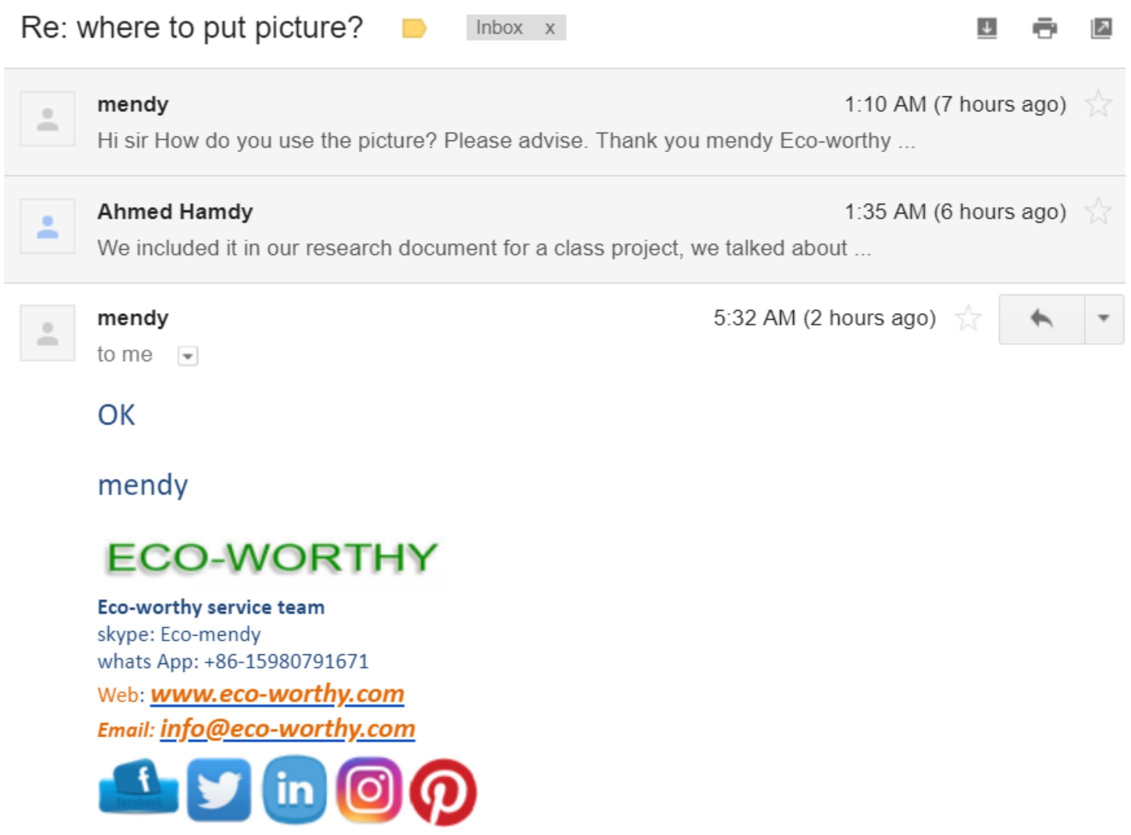
IEC (International Electrotechnical Commission)

ACID (Atomicity,consistency,isolation,durability)

PA (Precision agriculture)

10.2 Permissions

Permissions for images used:



Re: where to put picture? Inbox x

mendy 1:10 AM (7 hours ago)
Hi sir How do you use the picture? Please advise. Thank you mendy Eco-worthy ...

Ahmed Hamdy 1:35 AM (6 hours ago)
We included it in our research document for a class project, we talked about ...

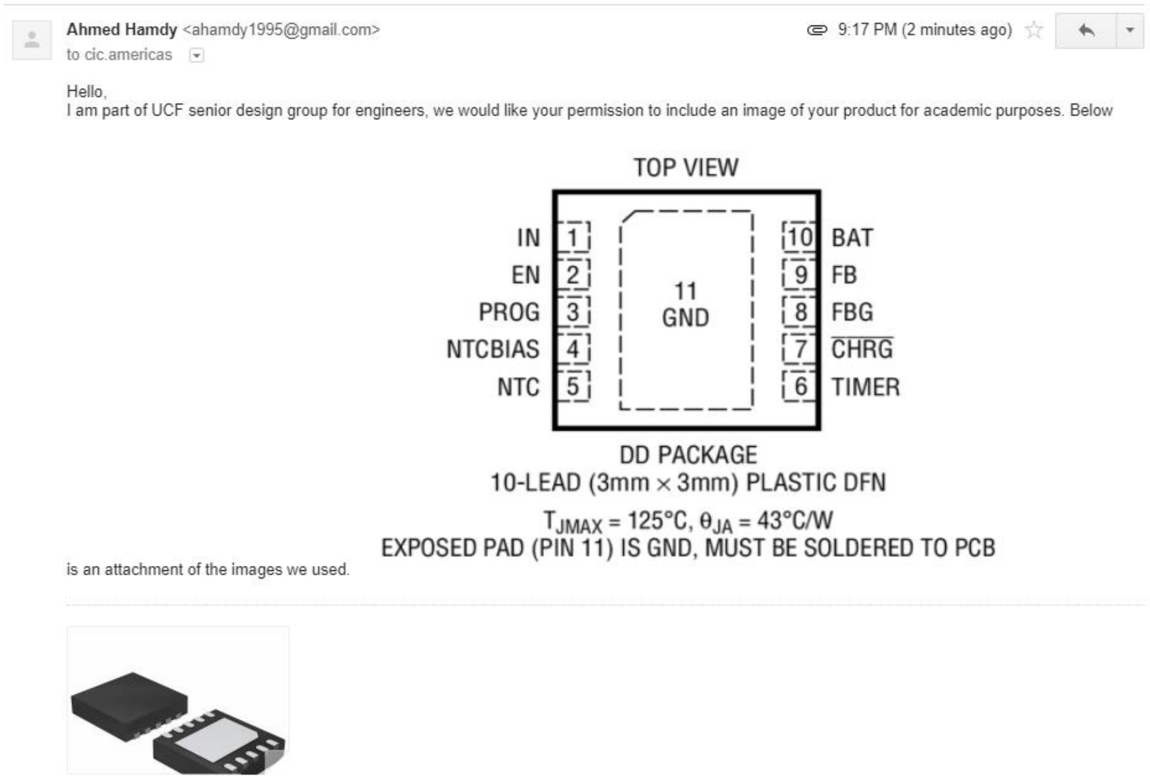
mendy 5:32 AM (2 hours ago)
to me

OK

mendy

ECO-WORTHY

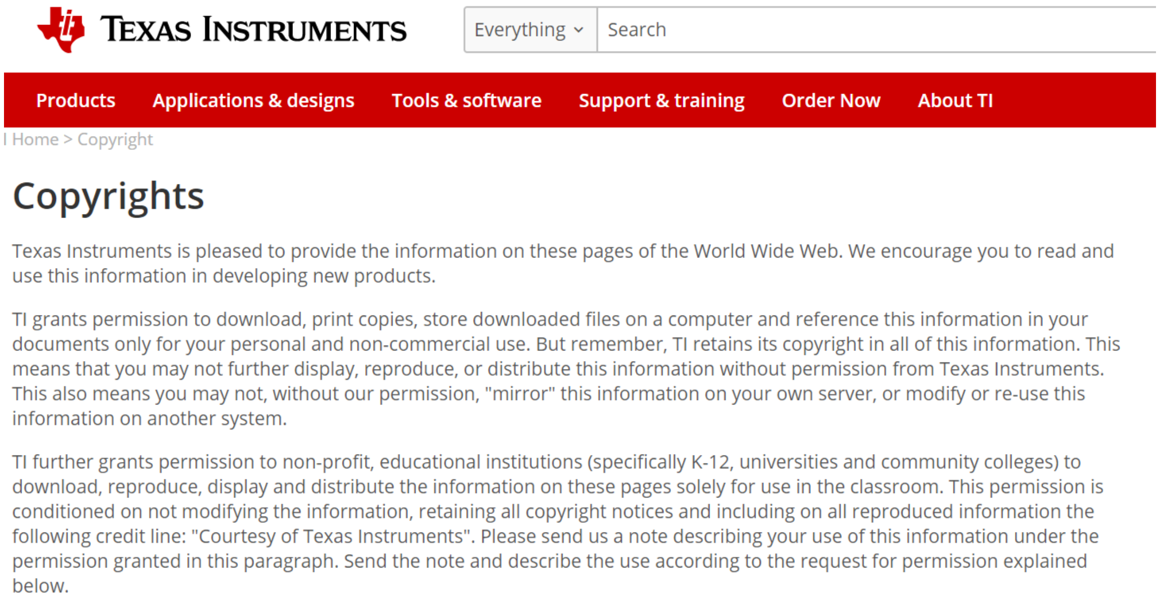
Eco-worthy service team
skype: Eco-mendy
whats App: +86-15980791671
Web: www.eco-worthy.com
Email: info@eco-worthy.com


Ahmed Hamdy <ahamdy1995@gmail.com> 9:17 PM (2 minutes ago)
to cic.americas

Hello,
I am part of UCF senior design group for engineers, we would like your permission to include an image of your product for academic purposes. Below is an attachment of the images we used.

TOP VIEW
IN 1
EN 2
PROG 3
NTCBIAS 4
NTC 5
11 GND
10 BAT
9 FB
8 FBG
7 CHRG
6 TIMER
DD PACKAGE
10-LEAD (3mm × 3mm) PLASTIC DFN
$T_{JMAX} = 125°C, \theta_{JA} = 43°C/W$
EXPOSED PAD (PIN 11) IS GND, MUST BE SOLDERED TO PCB

Figure 13:

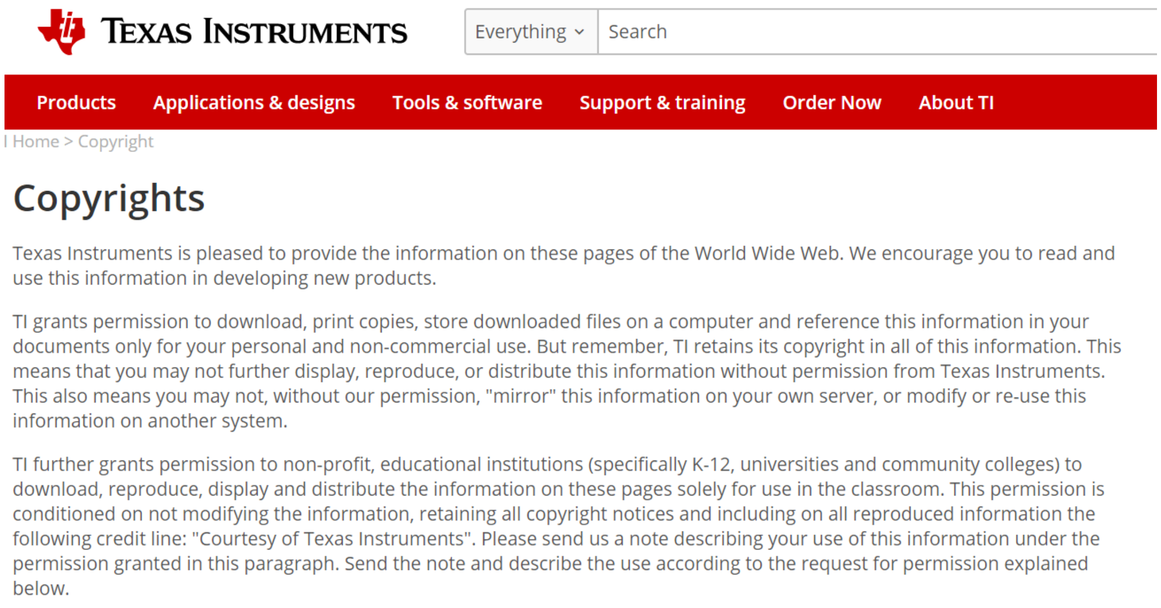


Copyrights

Texas Instruments is pleased to provide the information on these pages of the World Wide Web. We encourage you to read and use this information in developing new products.

TI grants permission to download, print copies, store downloaded files on a computer and reference this information in your documents only for your personal and non-commercial use. But remember, TI retains its copyright in all of this information. This means that you may not further display, reproduce, or distribute this information without permission from Texas Instruments. This also means you may not, without our permission, "mirror" this information on your own server, or modify or re-use this information on another system.

TI further grants permission to non-profit, educational institutions (specifically K-12, universities and community colleges) to download, reproduce, display and distribute the information on these pages solely for use in the classroom. This permission is conditioned on not modifying the information, retaining all copyright notices and including on all reproduced information the following credit line: "Courtesy of Texas Instruments". Please send us a note describing your use of this information under the permission granted in this paragraph. Send the note and describe the use according to the request for permission explained below.

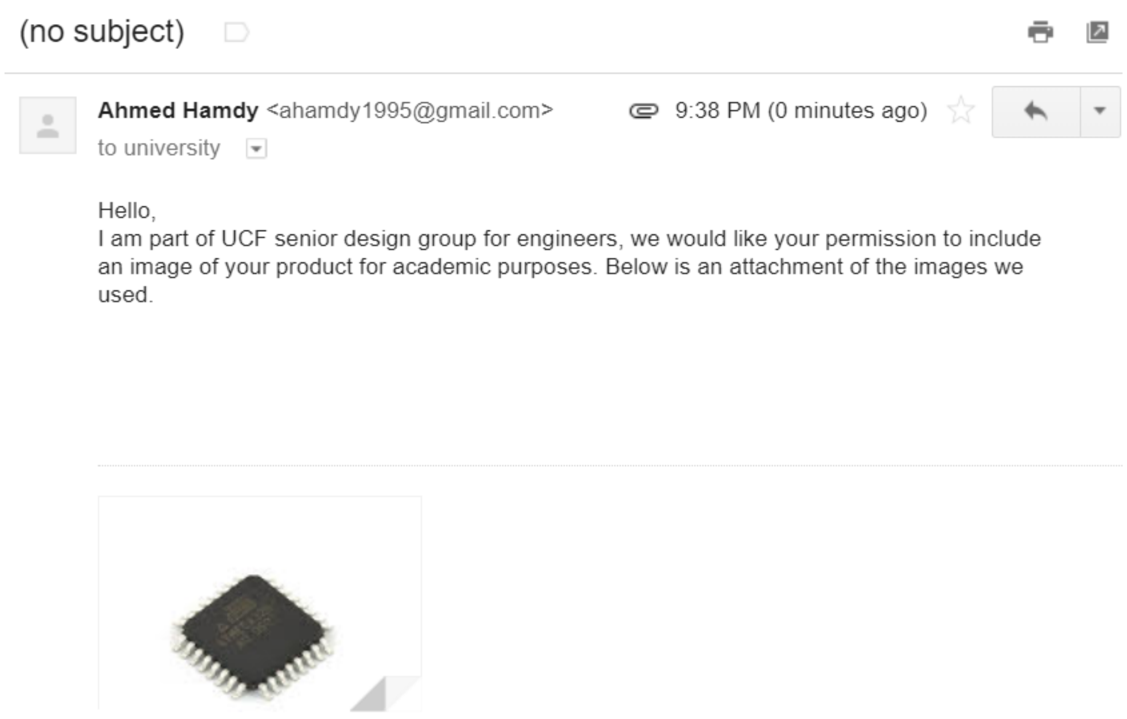


Copyrights

Texas Instruments is pleased to provide the information on these pages of the World Wide Web. We encourage you to read and use this information in developing new products.

TI grants permission to download, print copies, store downloaded files on a computer and reference this information in your documents only for your personal and non-commercial use. But remember, TI retains its copyright in all of this information. This means that you may not further display, reproduce, or distribute this information without permission from Texas Instruments. This also means you may not, without our permission, "mirror" this information on your own server, or modify or re-use this information on another system.

TI further grants permission to non-profit, educational institutions (specifically K-12, universities and community colleges) to download, reproduce, display and distribute the information on these pages solely for use in the classroom. This permission is conditioned on not modifying the information, retaining all copyright notices and including on all reproduced information the following credit line: "Courtesy of Texas Instruments". Please send us a note describing your use of this information under the permission granted in this paragraph. Send the note and describe the use according to the request for permission explained below.



(no subject)

Ahmed Hamdy <ahamdy1995@gmail.com> 9:38 PM (0 minutes ago)
to university

Hello,
I am part of UCF senior design group for engineers, we would like your permission to include an image of your product for academic purposes. Below is an attachment of the images we used.

Permission

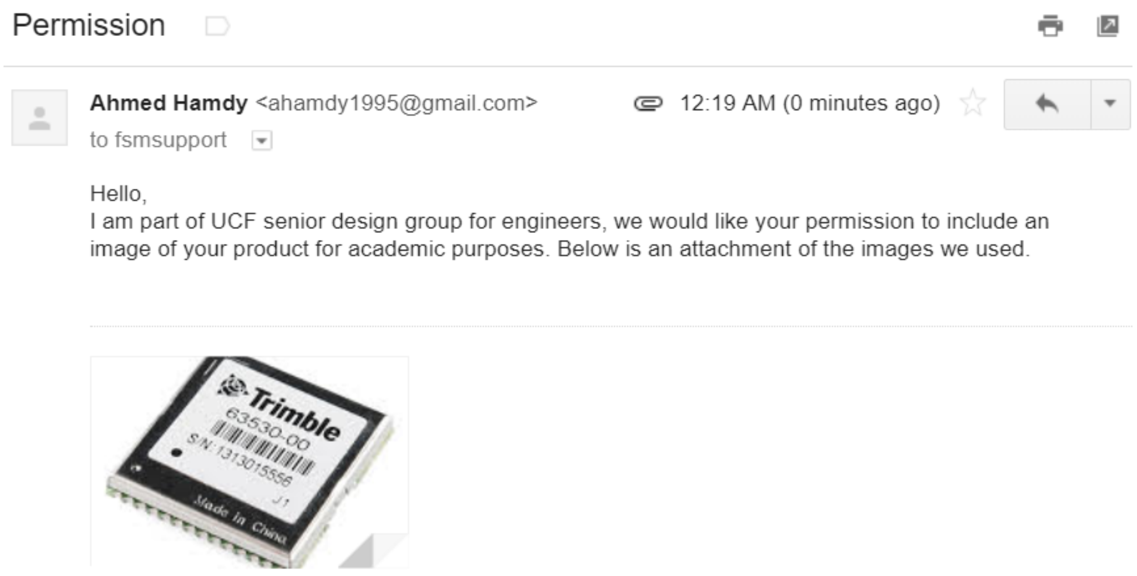
Ahmed Hamdy <ahamdy1995@gmail.com> 10:26 PM (0 minutes ago)
to chenfang

Hello,
I am part of UCF senior design group for engineers, we would like your permission to include an image of your product for academic purposes. Below is an attachment of the images we used.

Permission

Ahmed Hamdy <ahamdy1995@gmail.com> 10:29 PM (0 minutes ago)
to info

Hello,
I am part of UCF senior design group for engineers, we would like your permission to include an image of your product for academic purposes. Below is an attachment of the images we used.



Permission

Ahmed Hamdy <ahamdy1995@gmail.com> 12:19 AM (0 minutes ago)
to fsmsupport

Hello,
I am part of UCF senior design group for engineers, we would like your permission to include an image of your product for academic purposes. Below is an attachment of the images we used.

10.3 Datasheets

ATmega328P-PU

E01-ML01IPX

Trimble Copernicus II

LCT4079

LM1117

10.4 Bibliography

[1] “Aifro WaterEco." *Aifro*, https://www.aifro.com/aifro-watereco.html.

[2] “Blossom." *Blossom*, https://www.myblossom.com/.

[3] "Bluespray." *Bluespray*, 25 Sept. 2016, https://www.bluespray.net/.

[4] "Green IQ." *GreenIQ*, 5 Jan. 2015, https://greeniq.com/.

[5] "Irrigation Caddy." *GreenIQ*, 22 Oct. 2013, http://irrigationcaddy.com/features.

[6] "Lono Quick Start Guide." *Lono*, 4 Apr. 2016, http://lono.io/quick-start.pdf.

[7] "BHyve." *BHyve*, https://bhyve.orbitonline.com/.

[8] "Rachio." *Rachio*, 6 Feb. 2017, https://rachio.com/.

[9] "Rain Machine: How it Works." *Rain Machine*, 13 Jun. 2016, https://www.rainmachine.com/howitworks.html/.

[10] "Spruce." *Spruce Irrigation*, https://spruceirrigation.com/.

[11] "Rain Commander: Water from the Cloud.” *Rain Commander,* 26 Oct. 2014, https://www.rainmachine.com/howitworks.html/.

[12] "F.A.R.M. (Fundamental Agriculture Resource Monitor.” *UCF ECE Senior Design,* http://www.eecs.ucf.edu/seniordesign/fa2015sp2016/g25/.

[13] "Integrated Water Monitoring System.” *UCF ECE Senior Design,* 4 Dec. 2017, http://www.eecs.ucf.edu/seniordesign/sp2017fa2017/g8//.

[14] "LTC 4079.” *Analog Devices,* www.linear.com/LTC4079. https://www.rainmachine.com/howitworks.html/.

[15] "Strategy Analytics.” *Strategy Analytics,* 26 Feb. 2016, https://www.strategyanalytics.com/access-services/enterprise/iot/report.

[16] "IEC 62124:2004.” *IECEE,* 1 Oct. 2007, https://www.iecee.org/dyn/www/f?p=106:49:0::::FSP_STD_ID:6483.

[17] "RF24Mesh.” *GitHub,* 27 Feb. 2007, https://github.com/nRF24/RF24Mesh.

[18]. "Arduino-DHT22.” *GitHub,* 16 Jul. 2012, https://github.com/ringerc/Arduino-DHT22.