

Need Based Sprinkler System Design

MSCS CE CSULB Spring 2015

Author: Joshua Baird 007700067

Advisor: John Tramel

**Abstract:**

This is a report documenting a sprinkler/watering system based on the need of the surrounding soil. The requirements for the system are described. This is hardware and software design of the system and the interconnected workings behind each component. A diagram of the entire system, details about the specific hardware, software diagrams, as well as a software listing are provided. Testing strategy and test results are also provided. A manual of how to use the system is included, as well as screen shots of the user interface. A conclusion with future enhancements and references can be found at the end.

Contents

[Introduction: 3](#_Toc418488836)

[Requirements: 3](#_Toc418488837)

[Background: 4](#_Toc418488838)

[Method: 4](#_Toc418488839)

[System Design 5](#_Toc418488840)

[Raspberry Pi, Model B+ 5](#_Toc418488841)

[Capacitive Moisture Sensor by Catnip electronics 6](#_Toc418488842)

[Output Signal 6](#_Toc418488843)

[Wifi enabled router- or Network 6](#_Toc418488844)

[A Desktop PC, Laptop or Cell phone 6](#_Toc418488845)

[System Diagram Sprinkler System: 7](#_Toc418488846)

[System Diagram in house plants System: 8](#_Toc418488847)

[Software Design: 9](#_Toc418488848)

[Software Diagram: 9](#_Toc418488849)

[RESTFUL API 10](#_Toc418488850)

[PYTHON CODE: 11](#_Toc418488851)

[Testing Strategy: 12](#_Toc418488852)

[Testing Results: 12](#_Toc418488853)

[User Manual: 13](#_Toc418488854)

[Initial Setup: 13](#_Toc418488855)

[Modify Setup: 15](#_Toc418488856)

[Daily Use: 16](#_Toc418488857)

[Expand Use/Script Writing: 16](#_Toc418488858)

[User Interface: 17](#_Toc418488859)

[Conclusion: 26](#_Toc418488860)

[Accomplishments: 26](#_Toc418488861)

[Future Development: 27](#_Toc418488862)

[References –Code Listings: 28](#_Toc418488863)

# Introduction:

Over the years the field of water conservation and water usage have become a greater and greater concern, as the populations grows and with recent droughts this has never been so evident. It had occurred to me that many people water their lawns with simple timers. Or even with standard mechanical systems without really knowing if there soil is in fact in need of water. There are solutions for determining at a greater level of “intelligence” if the sprinklers should turn on and off, some based even basing this on weather, moisture, and other factors. However all of these products can be very expensive and even require a homeowner or user to completely replace their current system. I wanted to take this further and explore expanding this to not just sprinkler systems but all types of water systems, with all types of sensors. This project is my solution to a flexible watering system that can be used with multiple sensors with multiple watering methods.

## Requirements:

Below is a list of initial requirements put on the design.

* The system must be responsive enough to changes in ground moisture that once the moisture level has been reach the sprinkler system is turned off within 5 seconds.
* The system must be running in conjunction with currently installed systems.
  + That is it must be easy to implement and adapt to any already operating sprinkler system.
  + Must be adaptable to be used in place where no system currently exists.
* The system must provide the user the ability to modify parameters for moisture level at each node.
* The system must be scalable for more than a single sensor. A good range is one to ten sensors. Two will be used for demonstration.
* It should not require any special training to setup and use.

# Background:

A user wishes to add moisture based sensing to his already installed sprinkler system. However he does not wish to go out and buy an entirely new sprinkler system.

With this system he could enjoy the benefits of a moisture sensing sprinkler system without the hassle of tearing out his sprinkler system and installing a whole new system in some cases.

The system can also be used for in house watering where a sprinkler system is not installed. For example a simple light or speaker can be used to notify the user that the plant requires water. This is exactly what the demo shows.

# Method:

The design allows for many methods of sprinkler systems and math methods of measurement.

This is achieved by creating a single REST API that allows different sensors and hardware to be controlled by it. The different sensors and hardware may require separate scripts, but these scripts can don’t require a recompile of the system.

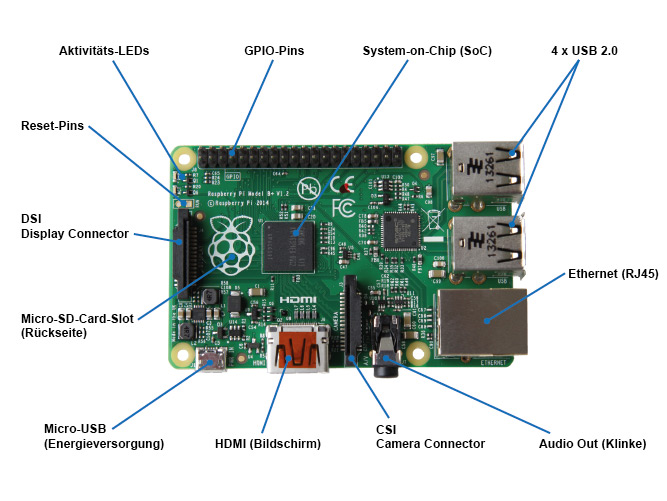
For a standard sprinkler system we can control it by simply redirecting the common grounds of the sprinkler valves to the raspberry pi and connecting moisture sensors to an I2C bus, with some setup on a web based setup page (see diagram on last page). The user can tie the moisture node to the common ground input and when a threshold is reached the common ground is bridged back to the common ground of the sprinkler system. So in an essence the raspberry pi is working as a switch with logic for when it is on and off built in.

For you indoor plans a simple LED light or other notification type device can be used that a simple low active signal can be used to drive. In fact most systems can be modified to work with a low active signal to determine when they are allowed to run.

# System Design

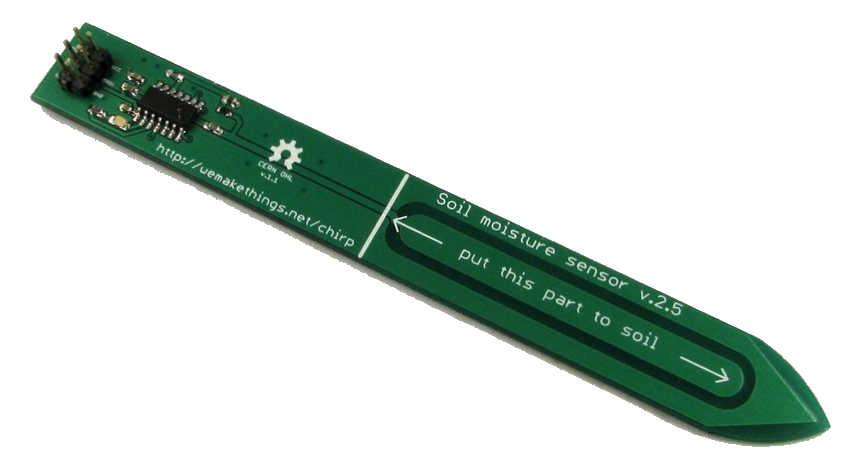
The system will contain the following main components:

## Raspberry Pi, Model B+



* + See <https://www.raspberrypi.org/products/model-b-plus/> for detailed specs.
  + To act as a receiver for the moisture sensors on the I2C BUS
  + To act as a low side driver.
    - Switch like controller for the common wires between the sprinkler valve and a standard sprinkler controller.
    - LED driver for in home plants.
  + To act as a server to display a web page setup for the “need” of each moisture sensor.

## Capacitive Moisture Sensor by Catnip electronics



* + See <https://www.tindie.com/products/miceuz/i2c-soil-moisture-sensor/> for detailed specs.
  + Senses moisture and provides the localized moisture level on a I2C BUS

## Output Signal

* + Standard Sprinkler system Or LED’s, something to display or turn on when water is needed.
  + To interface with using the raspberry pi.

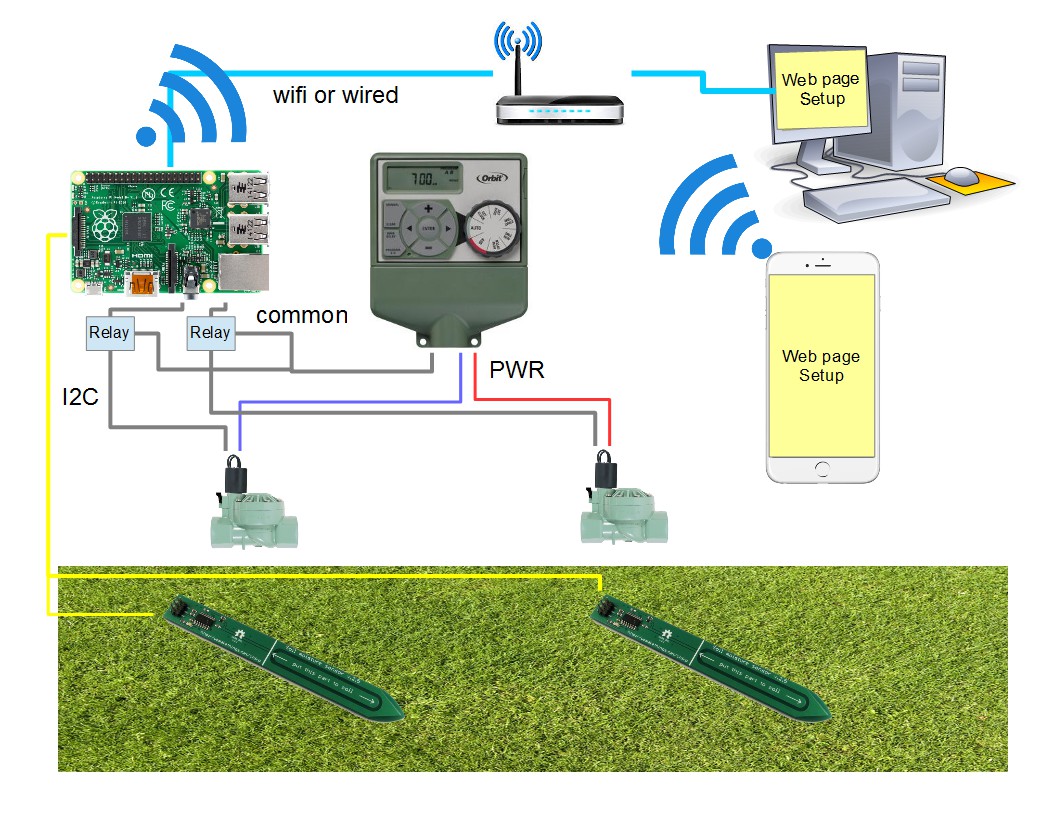
## Wifi enabled router- or Network

* + Used to connect devices to the raspberry pi’s web page to setup the levels for turning on and off an individual sprinkler

## A Desktop PC, Laptop or Cell phone

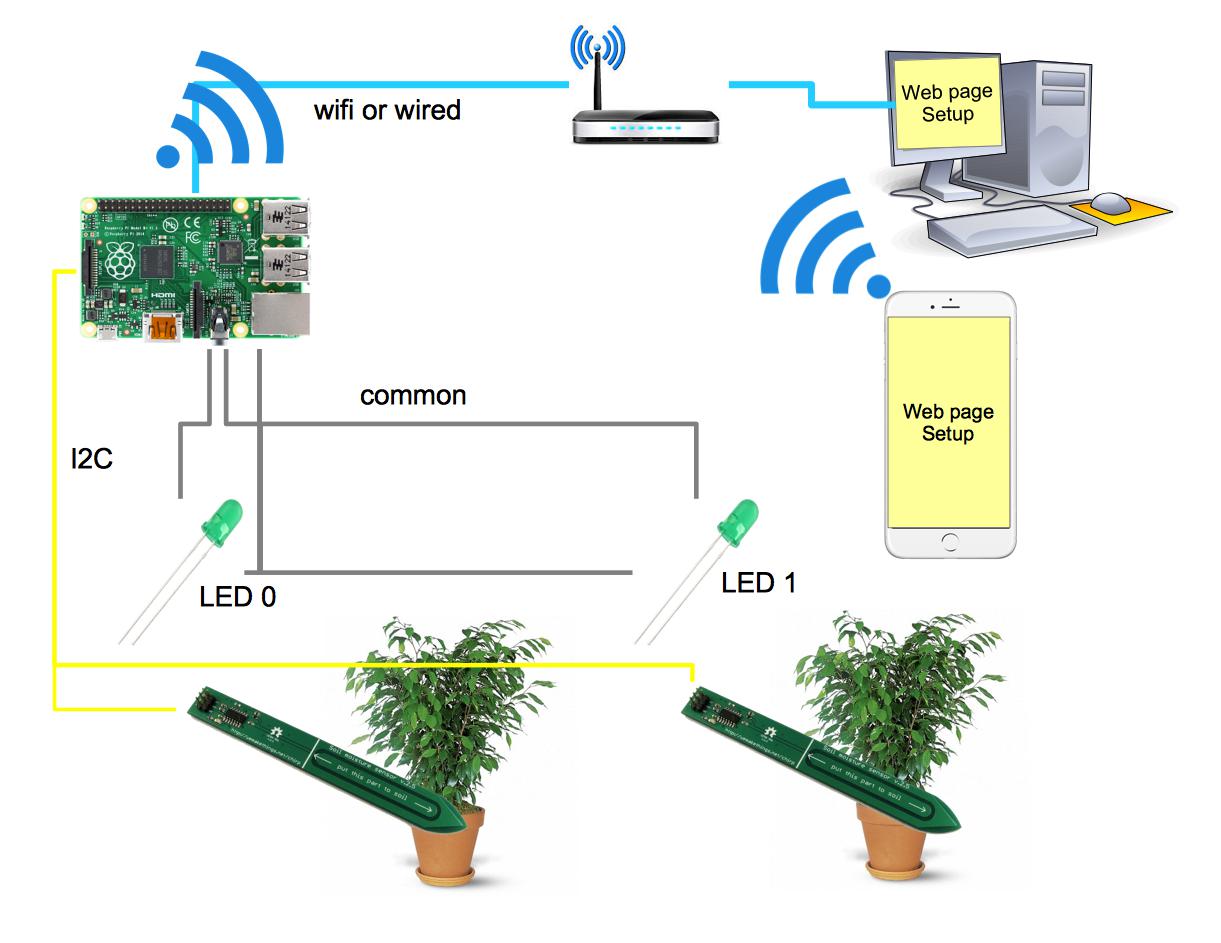
* + Capable of connecting to the network and opening a web page.

## System Diagram Sprinkler System:



As you can see in the diagram above the Raspberry Pi will act as a web server backend, supplying the user with an interface to both setup and monitor the system. The sensors will be connected to the raspberry pi via i2c bus. The allows the system to grow or shrink based on the limitations of i2c. At this point the raspberry pi is capable of addressing over 127 different devices. The Raspberry pi will then write those sensor values to a database, which is connected to the configuration for that sensor. Another threaded script will access that data and determine if a grounded output should be active or not. When it is active that output will drive a relay that completes the common ground circuit for the sprinkler system there by allowing it to turn on and water the area. All the while the sensor reading thread is still sensing, so once the moisture in the ground reaches a threshold the water will shut off until it is needed again.

## System Diagram in house plants System:

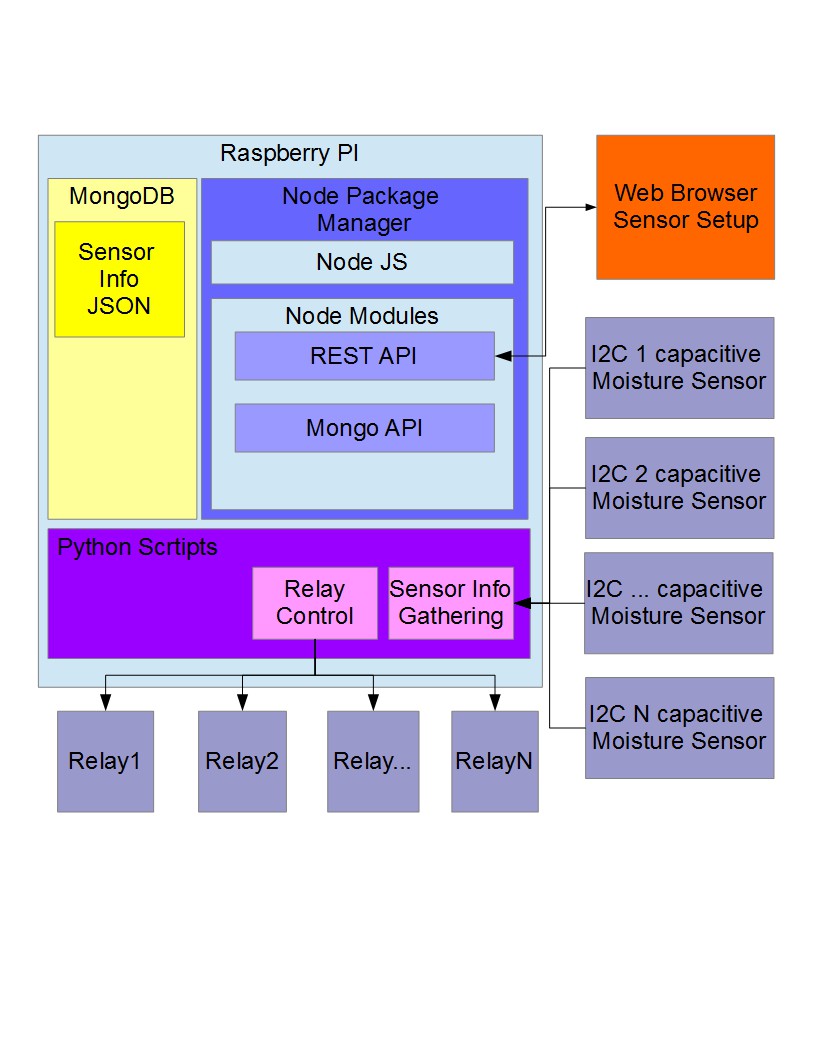


Similar To the above design this design operates in much the same way the only difference is the sprinkler system has been replaced with simple LEDs to turn on when water is needed and off when it is not. The LEDs are connected to a common power supply of 5v and each connected to the ground activated pin. When water is needed the pin is grounded which turns on the LED (in the instance above this was a relay). The LED will remain on until the sensor reads the max threshold at which point it will turn off.

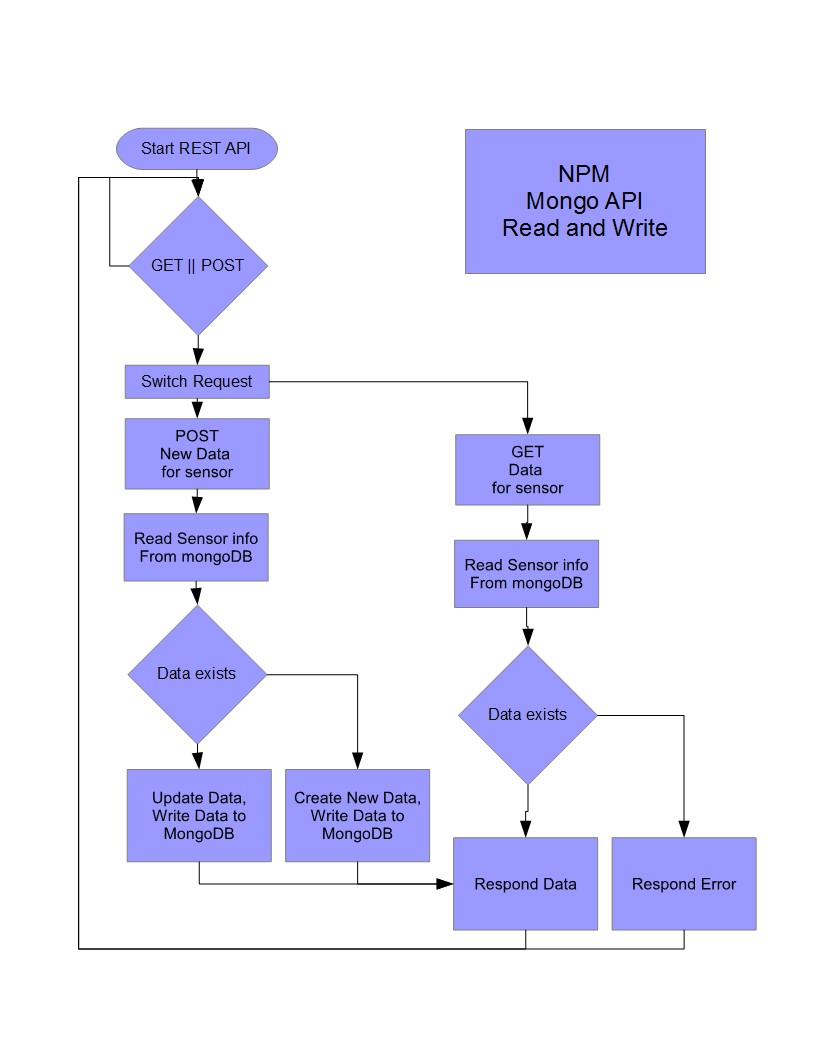
# Software Design:

* All Software and latest code listings can be found on github:
  + https://github.com/joshbaird/MoistureSensingSprinkler

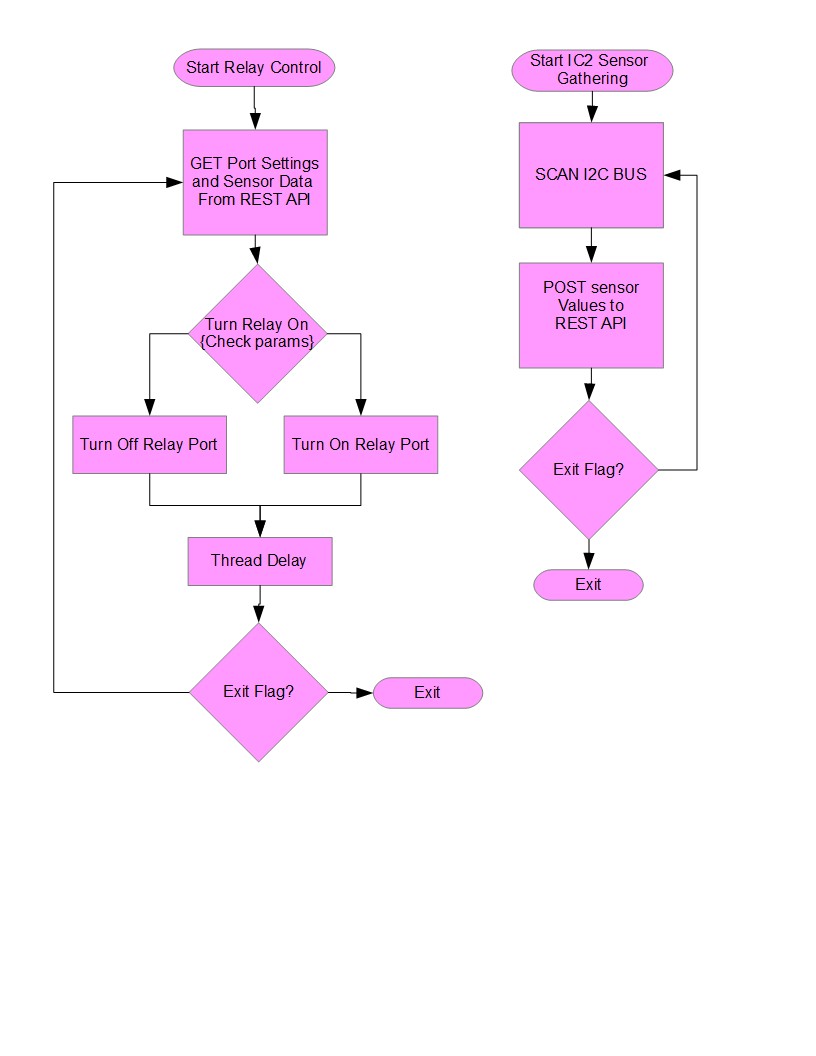
## Software Diagram:



## RESTFUL API



## PYTHON CODE:



# Testing Strategy:

The test for this product will be conducted using the system as a whole minus the sprinkler heads as they are difficult to remove and move. Instead Lights will simulate the operation of the sprinkler and a more reliable hand delivery of water will be used. The test will involve replacing the sprinkler heads with Lights that turn on when water is demanded and off when not. Simple LED lights will be used in conjunction with grounding them to the Raspberry PI. Two simple planters pot, with initially dry soil will be used to test. Two moisture sensors will be used for the demo. They will be placed in different buckets to show different readings. The test sequence is as follows:

1. Setup System.
   1. Follow system diagram below for details on connections.
   2. Actual pin numbers and connections will follow
2. Connect to system using laptop and configure sensors.
   1. Set the initial state of the moisture sensors to just above turning off the lights.
3. Pour water onto the sensor area until the light turns off.
   1. Allow the soil to soak up the water. And give it some time before pouring more.
4. Demonstrate that you can adjust the sensors on and off level in real time on the laptop.
5. Repeat with different levels of moisture for each sensor.

# Testing Results:

The results of the test showed that the sensors were not as responsive as I would have thought. There is a good couple of seconds before the values begin to shift. This could be due to allowing the moisture to settle in the dirt around the sensor.

I also noticed that the Light sensor takes quite a bit longer to read, and may consider removing it from the readings in the future as there may not be much benefit to having it.

In all the test results show that this system is capable of doing what it was intended to do.

# User Manual:

The most up to date instructions and manuals can all be found here: https://github.com/joshbaird/MoistureSensingSprinkler

## Initial Setup:

The first step as in most systems is the most crucial, setting up the system. In this system that means setting up the raspberry pi, setting up and connecting your sensors, and then configuring and starting the system.

Setting up the pi requires the following steps:

1. Setup NPM and web server + Mongodb.
   1. Install node *sudo apt-get install nodejs*
   2. Install express *sudo npm install -g express*
   3. Install global generator *sudo npm install -g express-generator*
   4. Install requests for python *sudo apt-get install python-requests*
   5. Clone repo *git clone https://github.com/joshbaird/MoistureSensingSprinkler.git*
   6. Change directory to clone repo *cd MoistureSensingSprinkler*
   7. Change directory to nodemoistureserver project folder *cd nodemoistureserver*
   8. Install and start MongoDB.
      1. Many ways to do this. I would follow this guide: https://emersonveenstra.net/mongodb-raspberry-pi/
   9. Install dependencies *npm install*
   10. Create data directory *mkdir data*
   11. Start web server to test *npm start*
   12. Browse to: [http://localhost:3000](http://localhost:3000/)
2. Enable I2C on the pi.
   1. remove i2c from blacklist *sudo nano /etc/modprobe.d/raspi-blacklist.conf*
   2. comment out the *blacklist i2c-bcm2708* line to *#blacklist i2c-bcm2708*
   3. Enable kernel i2C module *sudo nano /etc/modules* and add *i2c-dev* to the end
   4. install additional packages, *sudo apt-get update sudo apt-get install i2c-tools*, *sudo apt-get install python-smbus*
   5. add pi user to i2c group *sudo adduser pi i2c*
   6. change baud to 1000-3000, start low and work up. *sudo modprobe i2c\_bcm2708 baudrate=1000*
   7. reboot pi...
   8. test with command i2cdetect -y 0, for raspberry pi b, if that does not work try i2cdetect -y 1 for raspberry pi b+

Your pi is now setup. You can browse to [http://localhost:3000](http://localhost:3000/) and you will see the setup and configuration page [http://localhost:3000](http://localhost:3000/)/config.

This brings us to the setup of the sensors. To do this you must individually setup the sensors. First by connecting one sensor then adjusting or changing its address. A helper application has been provided that enables you to tell it the current address and the desired new address.

Follow these steps to setup your sensors:

1. Connect a single new sensor(address is default 0x20) to the i2c bus on the PI.
2. Run the changeAddress.py script under the python\_scripts directory as follows:
   1. Python changeAddress.py 20 30
   2. This will change the address from 0x20 to 0x30.
3. Confirm the change by running i2cdetect -y 0, or i2cdetect -y 1 depending on system.
4. Disconnect newly configure sensor in preparation for next sensor.
5. Repeat steps 1-4 until all sensors are configured. Carefully using different addresses for each one.
6. Once all are setup connect them all back to the bus and scan the bus again to make sure all are visible.
7. You now need to create an entry in the system for each of these. Navigate your browser to [http://localhost:3000](http://localhost:3000/) and create an entry for each of these.
8. You will also need to create a config for the sensor type ‘ic2’ on this page:
   1. http://localhost:3000/config
   2. These values represent the actual min and max values from the sensor reading in raw form. I have found the following values work well:
      1. Light range, 0-65000
      2. Temp, 0 – 700
      3. Moisture 100 - 800
9. And that’s it. All your sensors are now configured.

Now that all sensors are configured you need to completely start the system. To do this you must start two scripts:

1. Start the sensor gather script under the python\_scripts folder:
   1. python gatherSensorInfo.py &
   2. The ‘&’ allows it not to hog the terminal and allows you to run other scripts this can be left off if you wish to run multiple terminal windows, as this is what I do.
2. Start the relay control script under the python\_scripts folder:
   1. sudo python relayControl.py &
   2. this must be ran as sudo in order to control the output pins.
   3. Again the ‘&’ is optional.

The system is now running. It is monitoring/sensing the moisture, light, and temp. It will activate the output whenever the values drop below the min, and it will remain on until the values reach the max.

## Modify Setup:

Modifying a setup is simple. You can add new sensors at any time as well as change the values of sensors at any time. Here is a list of operations and steps to each:

1. Add a sensor:
   1. First you must stop the gather, and relay control scripts to add sensors.
   2. Add a sensor in the same manner you did the initial setup. First by removing all sensors from the bus and adding the new one.
   3. Test the new sensor after address change.
   4. Reconnect all sensors.
   5. Add entry on setup page for new sensor.
   6. Restart python scripts.
2. Remove a sensor:
   1. Similar to adding a sensor, however you do not need to stop and restart the scripts.
   2. To remove a sensor, simply remove the entry from the setup page by deleting it.
   3. Give the system 5 minutes to purge out the old data and then you may disconnect the sensor.
3. Change a min and max value for a sensor:
   1. This falls into daily use, but to do this simply adjust the value on the setup page, by retyping all the values back in. (in the future I would like to populate this table when you click a sensor)

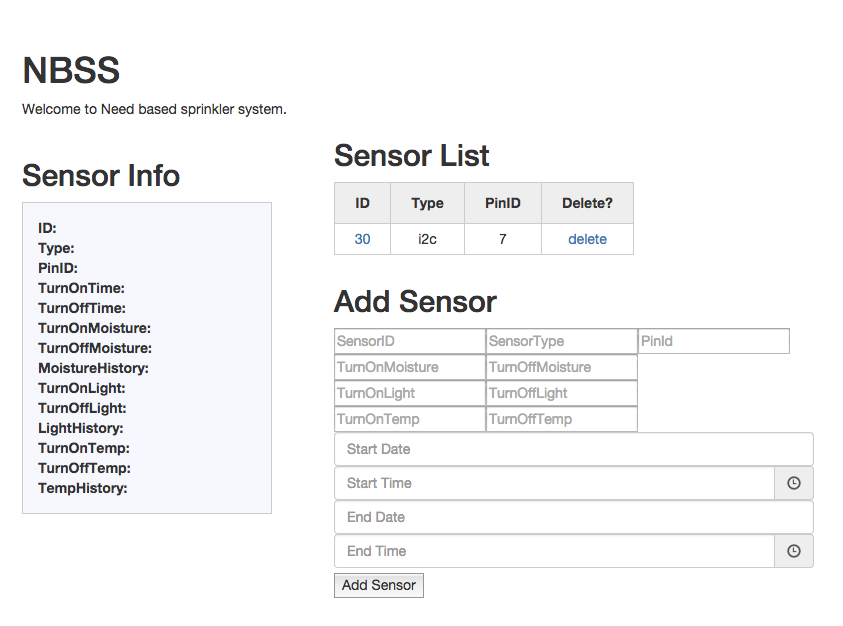
## Daily Use:

Daily use of this system is simple and easy. The system will handle itself after the initial setup. There is no real need for daily use. However; some may feel inclined to tweak values or monitor the system. This can all be done from the main setup page. When you click a sensors ID you will see the senor info, along with the historic values read. You can tweak the values for that sensor by entering all the values again and adding it. This will update the sensor based on the ID. If the ID is the same it will overwrite any previous values stored, giving you new minimums and maximums.

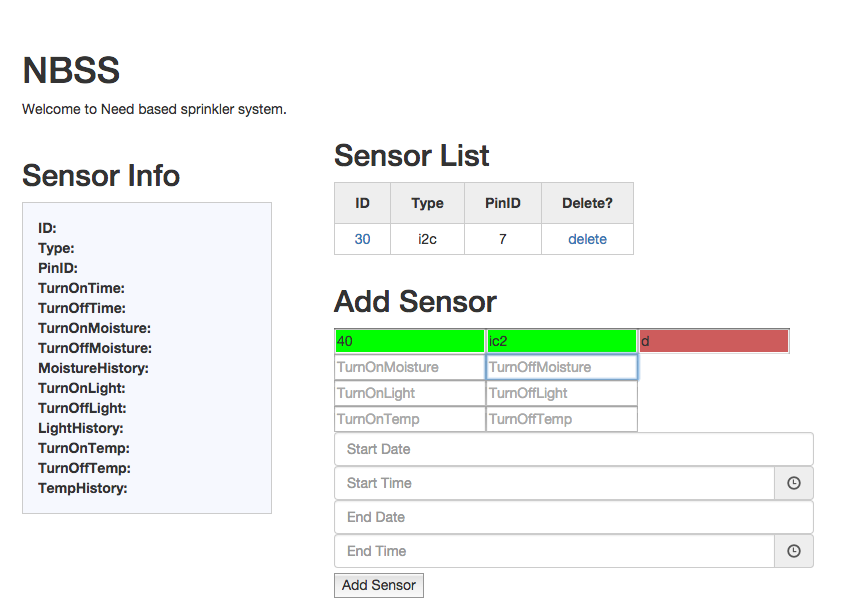
## Expand Use/Script Writing:

This is for advanced use. If you wish to use your own scripts to gather, and or control the outputs you may do so. Since the architecture of the system is RESTFUL you may use any language that is capable of doing GET, and POST. Please use the gatherSensorInfo.py as an example for python script to gather system info, and use the relayControl.py script as an example of controlling the output.

# User Interface:



Main Setup Screen. This is where a user will input, update, and view sensor information.



Main screen, showing visual validation of each input field. Green is acceptable input, Red is incorrect.

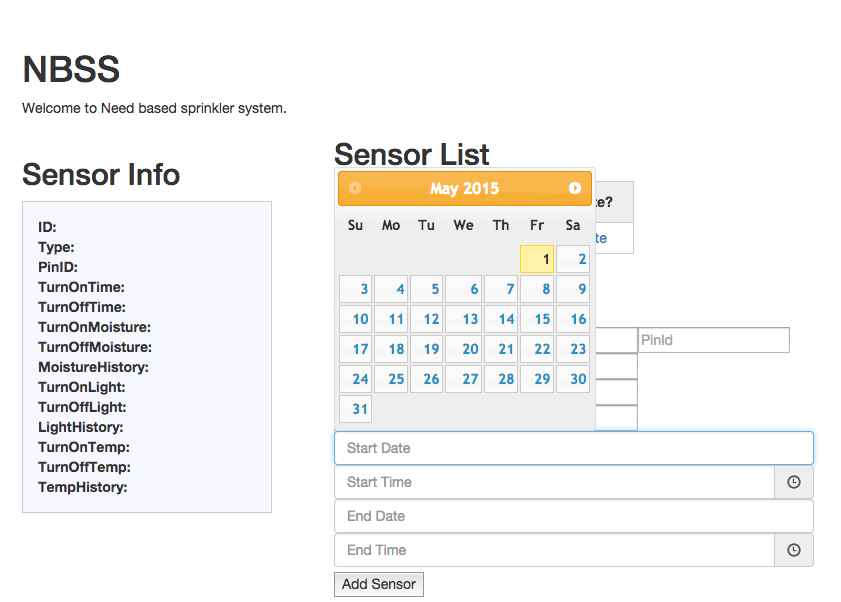


Image shows the date selector. When a user clicks on a date field, this helper will pop up to make for easier date selection.

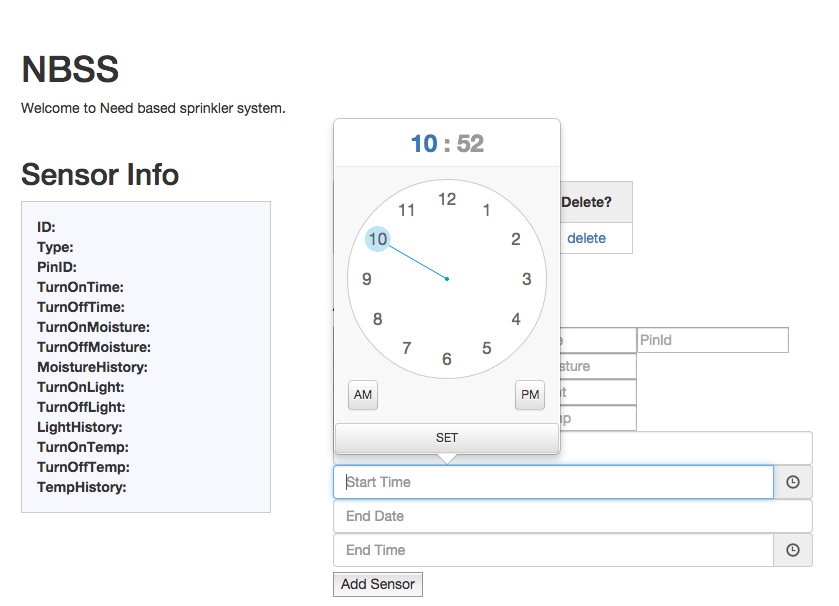


Image shows the time selector. When a user clicks on a time field the time selector helper will appear to allow for easier time inputting the times.

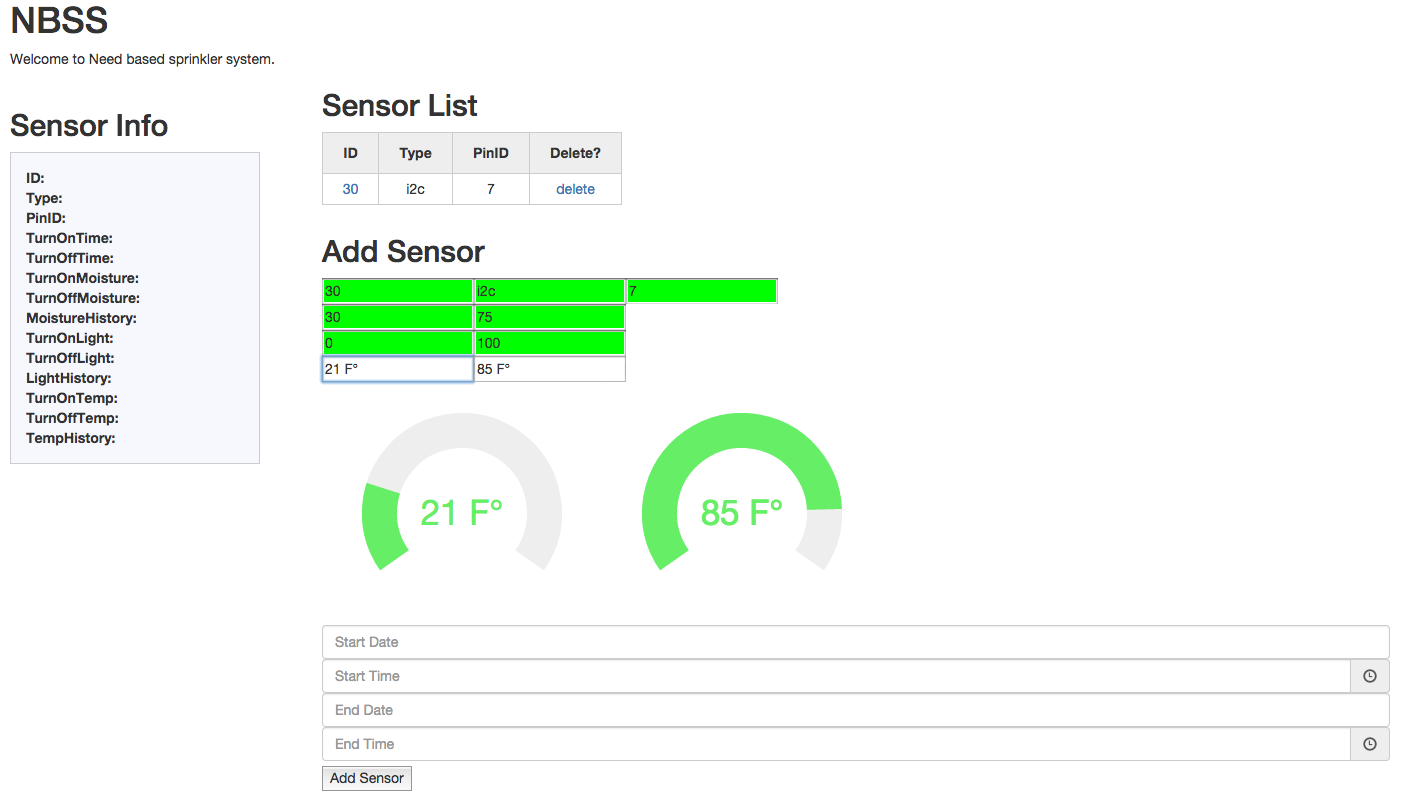
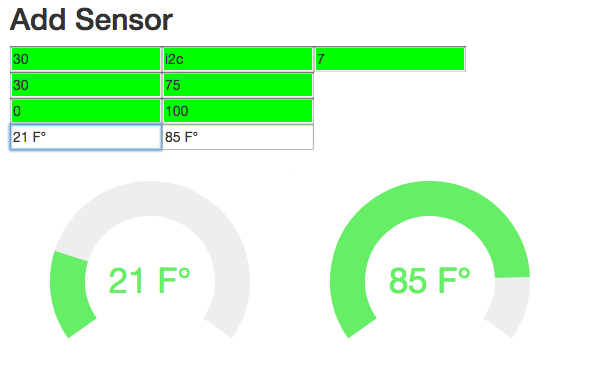
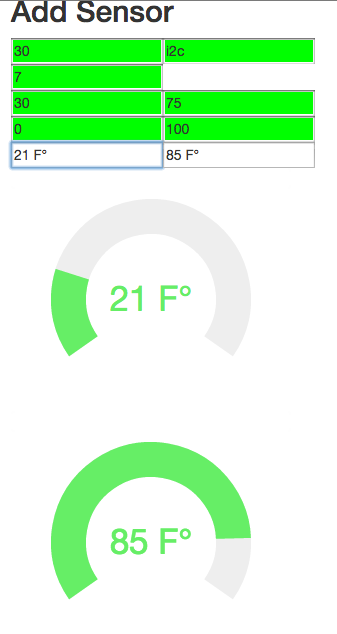


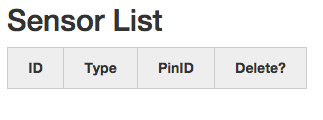
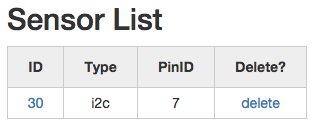
Image shows the temperature selectors. When a user clicks on either of the temperature fields this helper will pop up and allow the user to select the correct temp for on and off settings.



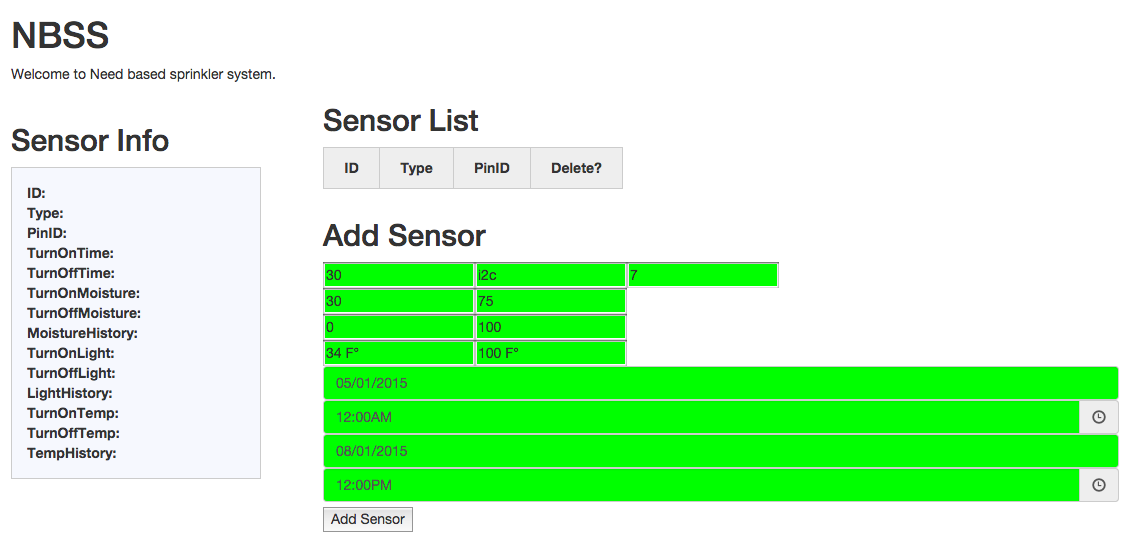
A close up of the temperature selectors. Also you can see valid fields are displayed in green. The temperature helpers render to the screen orientation of size. They will not cut off the edge of the screen.



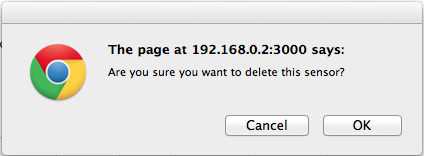
This image shows that when the screen size changes the temperature helpers adjust for it.



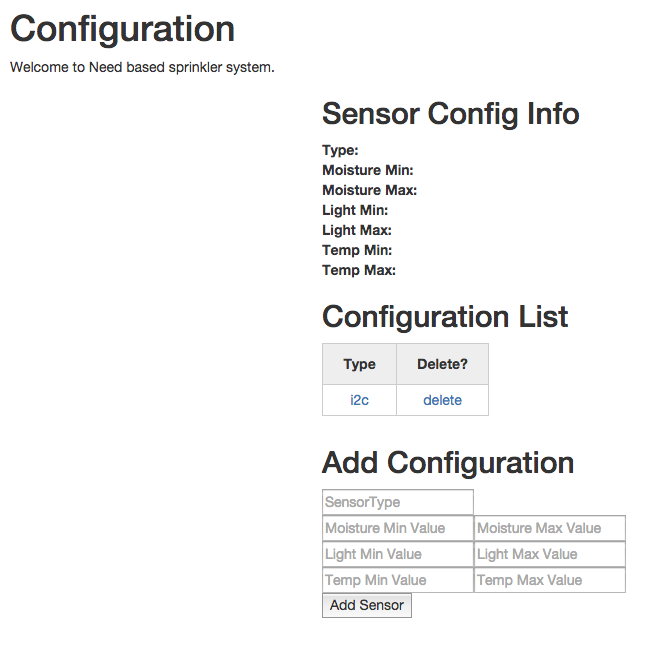
The sensor list is populated with all sensors in the DB (left). If there are not a blank list (right) will be shown.



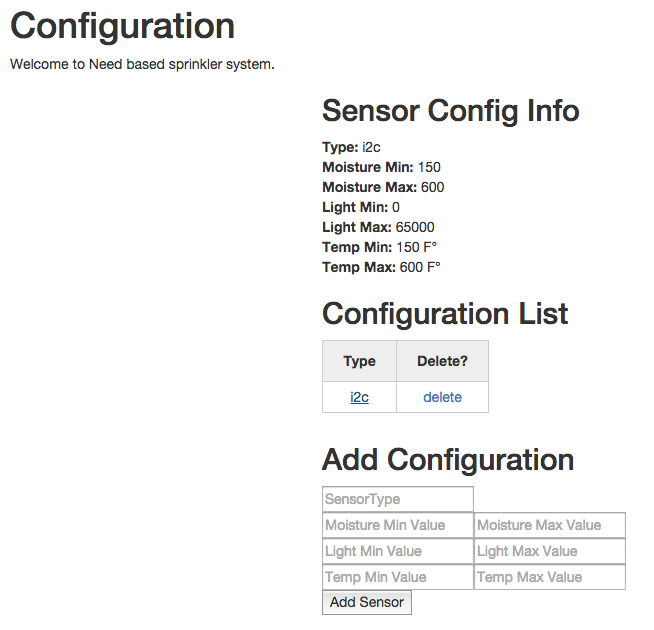
Shows a main page with no sensors added, but all the fields are filled out and validated. The use simply needs to press the ‘Add Sensor’ button, and the information will be sent to the DB.



There are notifications for deletion of entries as well.



This image shows the configuration page. A sensor type requires a configuration in order to determine the min and max values for that sensor type. This is used to abstract the ADC values from sensors and allow the user to use more meaningful values such as percentage.



This image shows that when you click on the type of the sensor the information is display in the information area.

# Conclusion:

In conclusion, I really enjoyed this project, so much so that I plan to continue its development. I can see a real use for technologies like these that allow a user greater control over when and how their watering systems work. I feel this type of technology can also be expanded to many things beyond controlling water systems, electrical power use, and gas usage, and other control systems where the user wants to monitor and control how much and when they can use it.

I feel I have learned a lot doing this project. The learning curve for me was a bit steep as I have never worked with a RESTFUL api, node, or mongo db. I am glad I took a step out of my comfort zone and added these tools to my skill set.

Below I have listed some of the accomplishments I have achieved and some that I may have missed the mark and the reasoning behind each. I have also listed what may be possible in the future with a system like this. Including features I would like to add.

## Accomplishments:

I successfully designed and implemented a full stack api. A complete RESTFUL api backend for a sprinkler system is working. This consisted of configuring and installing a web server on a raspberry pi, as well as a complete mongodb server. This required a great deal of learning on my end. I also had to learn the express framework for web development. I implemented an express generated web server. This required learning the framework for express, node, and jade templates. Again, a giant learning curve for me as most of my experience has been in embedded systems. However; I think these tools are extremely useful and I have shown you can bridge and use them in an embedded situation. As it stands now the industry is continually pushing for develop one and deploy to multiple systems. HTML 5 and web is one solution for this. This is why I felt it would be a good fit as a solution and I am glad I attempted it.

As another set of accomplishments, I successful integrated to the RESTFUL api with python scripts that are capable of accessing the mongodb through the api. This is more of an exercise in testing the api and backbone, but it allows us to control sprinklers and bridge the gap between the software and hardware as the python can control the actual hardware on the pi. But the way the system is design, any device that can connect to the same network can GET and POST to the backend and drive outputs. So it is scalable beyond a single device connecting.

I believe I have reached nearly all my goals. Some of these goals had to be tweaked in order to work. For example, I was not able to obtain a working sprinkler system to test this on. I did not achieve an in place working system with a real world sprinkler system. This is mainly due to finances, as a sprinkler system is not inexpensive and I currently do not own one, or have readily available access to one I could play around with (my friends and neighbors were not too keen on the idea of me hacking there’s apart to test with…). However; I do feel my proof of concept shows this is possible. If we are driving an LED or a relay the logic remains the same, this is why I believe it will work.

## Future Development:

I continue to develop this project, as well as support it. I have made the source code open source as some of the modules and code I am using require this for copyright. The most up to date cost listing and features can be found on github here: <https://github.com/joshbaird/MoistureSensingSprinkler>

This is a list of features I would like to add in no particular order:

* Adding a wireless sensor.
  + Either find a sensor that works over rf or design my own.
  + Thinking a mesh of wireless sensors is really the way to go.
* Adding multiple output drivers.
  + The RESTFUL api allows us to use more than just one device to control outputs. As long as they can connect to the network they can talk directly to the api.
* Adding new scripts for reading different kinds of sensors
* Adding more “helpers” to the setup screen similar to the temperature and date time pickers.
* Adding more scripts and script examples in other programming languages.
* Create a setup script that will do all the steps for the user by hand.
* Stress test the scalability of the system to get a better idea of its expandability.
* Upgrade the Pi to a Raspberry PI 2.
  + This will allow for more features. As it is now the loading of some of the modules takes quite a bit of time.
* I would love to install this system in a real world test environment.
  + For all kinds of sensor types and situations.
* Add a cleaner look and feel to the setup and configuration pages
* Simplify the setup and configuration.
  + Possibly automate all the steps.

These are just a few of the things I would like to do to the system. I think the capability of a system such as this has a lot of potential for growth and expansion.

# References:

Below is a list of references and websites used during the development of this system:

* Github, where all code is stored.
  + <https://github.com/joshbaird/MoistureSensingSprinkler>
* Guide to setup MongoDB on a raspberry pi
  + <https://emersonveenstra.net/mongodb-raspberry-pi/>
* Guide to Express and Restful api.
  + <http://cwbuecheler.com/web/tutorials/2013/node-express-mongo/>
  + <http://cwbuecheler.com/web/tutorials/2014/restful-web-app-node-express-mongodb/>
* Source code for accessing the moisture sensor in python
  + <https://github.com/JasperWallace/chirp-graphite>
* Source code for the moisture sensor
  + <https://github.com/Miceuz/PlantWateringAlarm/tree/release/sensor>
* *Smart Irrigation Controllers: How Do Soil Moisture Sensor (SMS) Irrigation Controllers Work?*
  + <http://edis.ifas.ufl.edu/ae437>

# Code Listings:

Note: all the latest code can be found here: https://github.com/joshbaird/MoistureSensingSprinkler