Web controlled Smart Drainage and Sub Irrigation Control System

Design Document

Team Dec1602

Team Leader:	Anne Ore
Webmaster:	Griffen Clark
Key Concept Holder:	Michael Parker
Key Concept Holder:	Adam Wolter
Communications Liaison:	Rodney Barto
Client:	Agri Drain
Advisor:	Nicola Bowler

Dec1602@iastate.edu
Dec1602.sd.ece.iastate.edu

Revised: 3/2/2016 - Version 1.0

Table of Contents

1 Introduction

1.1 Project Statement

The Agri Drain Web Controlled Smart Drainage and Sub Irrigation Control System is simply a M2M (machine to machine) control system for water management structures. What this means is that a user will have the ability to control and monitor control structures in the field from a variety of devices, such as a smartphone, personal computer, or tablet. The end product will be a web application that fits into Agri Drain's website and is integrated with their current database. The web application will be able to report flow rates, water levels, and rainfall totals collected by the structures in the field. It will also be able to provide to the user an easy to use control that manages the flow rate of each structure while also being able to set up schedules of certain drainage rates.

1.2 Purpose

Agri Drain, the leader in industry for manufacturing these water control structures, wants to provide this easy to use functionality to farmers. This would allow farmers to increase crop yields while also conserving water and soil. The end product will be able to provide data, such as flow rates, water levels, nutrient levels, rainfall totals, and other data in a graphical, easy to read, format. End users will also have the ability to export this data to a spreadsheet file.

1.3 Goals

The obvious goal our team wishes to accomplish with this project is to have a fully functional web application that provides the user with the ability to monitor and control elements of the water control structure remotely. This web application needs to have the functionality of being able to use Agri Drain's existing user database and be able to be placed on Agri Drain's server along with their existing web site. Our team aims to provide an intuitive and beneficial tool to farmers that could not only be used as a cost and time saving device but also an effective way to reduce harm done to valuable soil and the environment due to over irrigation or nutrient runoff.

2 Deliverables

The main deliverable for this project is the working web application and source code for this project. Other large deliverables include a fully configured database integrated into the web application, and a comprehensive test suite that covers most of the functionality and serves as documentation for the source code. However, other deliverables will include the intellectual content within our product. This could include our development environment, development process, as well as how the various objects or components of the product connect. These components can be found in the block diagram of **Figure 3** in Appendix 7.2.

3 Design

Our client specifically desires that the product be in the form of a web application that is able be placed alongside the client's existing website and follow that site's existing theme. The design of the web application is going to have many iterations based on input from the client as well as potential end users; however, the beginning design will be heavily based off of the example dashboard provided by the client, as seen in **Figure 1** in Appendix 7.2.

3.1 System Specifications

We have not received any specific system specifications from our client as of yet. We plan on developing a codebase that is cross-platform, so it can function on any system that the client provides for us.

3.1.1 Non-functional Requirements

We have not yet received any non-functional requirements from our client. However, our web application will have to have a reasonable response time both client side navigation as well as communicating over a cellular network to hardware located in the field. It must also be able to scale for many users and many hardware units in the field ("many" has yet to be defined). It is essential that our team takes into account disaster recovery, such as what is going to happen to the hardware units if communication is lost. The product must also be easy to navigate which can be enhanced by using large text and contrasting colors.

3.1.2 Functional Requirements

There are many functional requirements of the web application, some of them are listed below. These functional requirements are highlighted in both **Figure 1** and **Figure 2** in Appendix 7.2.

- The web application must show locations of hardware units and allow the user to select individual units from a map in order to see details about that unit.
- The web application must provide alerts to users in multiple ways when an event occurs that could require attention, such as quickly rising water levels.
- The web application must be able to keep a log of events and data while also being able to graph and export this data in a variety of ways that are easy to read.
- The web application must provide automation of events to control the hardware units.
- The web application must provide several pieces of data including water and nutrient levels.

3.2 Proposed Design/Method

For our client-sided code, we plan on using React, which is a JavaScript library that was created by Facebook, and is currently used in several large-scale websites in production. React allows us to extract each part of the client sided experience out into individual, reusable components, where each component has its own defined behavior and markup. Alongside React, we plan on using SASS to manage our CSS styles. By giving each component its own associated SASS file, we can leverage this combination of React and SASS to define individual pieces of the product by specifying a component's behavior, markup, and styles. Finally, we plan on using Bootstrap to help do some of the heavy lifting when it comes to responsive design and overall look-and-feel. There are many popular websites that use Bootstrap, so this will allow our user interface to look familiar to other websites that our end-users may have used before.

Our server-sided code will use the Express framework. Express is an extremely easy to use and unopinionated web framework that works with pretty much any configuration that you give it. Express will give us the ability to create different endpoints for our application, which can be used by the drainage controller to send information and receive commands from the user. Our Express code will act as the layer between the user interface and the database / drainage hardware.

Finally, we plan on using a few different tools to assist us with the development of this project. Webpack is one of these tools, and it will allow us to bundle all of our client-sided files (JavaScript, SASS, and HTML) into a single file to be sent to the client. This allows for a drastic reduction in the amount of data that end users need to receive when loading our application. Webpack also gives us the ability to create development servers, which can detect any changes in our code, and automatically re-bundle our code and immediately re-deploy it to the client side. This is extremely helpful for development purposes.

3.3 Design Analysis

Figure 1 in Appendix 7.2 is a highlight of some of the requirements that the product must have. It also serves as a very beneficial guide for our team to meet our client's desires. Our team plans to use this to ensure we cover all of the elements our client is looking for but we also plan to significantly improve on this web dashboard by rearranging items and adjusting the flow of the dashboard to be better suited to how people naturally approach applications. Doing this will provide a better user experience and ease of use. Also, as previously mentioned in the introduction to the Design section, the design of the product will be revised several times based on feedback from potential end users.

4 Testing

4.1 Tools and Software

There are a few different JavaScript libraries that we can leverage to help ensure that our application is adequately tested.

- Mocha will allow us to write unit tests for each of our components to make sure they behave properly on their own. Mocha allows for human-readable test files, which helps improve the simplicity of the codebase.
- Chai will allow us to define assertions in each of our tests to make sure that each component does what it is supposed to do. Chai provides developers with the ability to chain expectations together to form human-readable sentences. This makes it extremely easy for newcomers to get familiar with the codebase quicker.
- Sinon is a framework that we can leverage to help us mock out other components when writing our unit tests, so we can fake . It also allows us to create “spies”, which we can use to determine that certain functions were called with specific arguments during the execution of our tests.
- Cucumber is a testing framework that allows us to write acceptance tests. These tests will exist in separate test files that use the Gherkin syntax, which is a business-friendly human readable syntax that looks almost identical to the English language. This will allow for our client to write acceptance test files that we would provide the code implementations for.

4.2 Testing Process

For the development of this project, our team will be following the methodology of Test Driven Development. This means that we will write our unit tests first, provide the implementation second, and perform any necessary refactoring once we have successfully passed the test. This way, we can write tests based on our given requirements, and only write the code that is absolutely necessary to fulfill those requirements. This will lead to a cleaner codebase with less technical debt to worry about later.

5 Results

So far, our team does not have any results to reflect on. We are still in the planning stages of development which is a very important stage of software development. Developing software without the proper planning significantly increases technical debt which is why our team is thoroughly planning specific details of the project. This will help reduce unmaintainable code and bugs as well as enable easier evolution and maintenance.

6 Conclusions

Summarize the work you have done so far. Briefly re-iterate your goals. Then, re-iterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

In conclusion, we believe that the approach outlined in this document is the best approach that we can take to accomplish our overall goal of creating an interactive web dashboard for remotely controlling and monitoring irrigation/drainage hardware. We believe that our plan will be successful because each team member is experienced with the JavaScript programming language as it pertains to web development. The decision to use JavaScript will greatly improve the quality of our codebase because it will allow us to write our client-sided and server-sided code using the same language. JavaScript also gives us access to tons of different libraries, frameworks, and tools, which we can use to our advantage. All of this will help us speed up the development process and produce quality code. In addition, the continuous feedback that we will be receiving from the client and end users will enable us to provide an easy to use web application that benefits the farmers and the environment as a whole.

7 Appendices

7.1 Resources

Below is a list of all of the technologies our team plans on using:

- Node.JS https://nodejs.org/en/
- Express http://expressjs.com/
- React https://facebook.github.io/react/
- Bootstrap http://getbootstrap.com/
- SASS http://sass-lang.com/
- Mocha https://mochajs.org/
- Cucumber https://github.com/cucumber/cucumber-js
- Sinon http://sinonjs.org/
- Chai http://chaijs.com/
- Webpack https://webpack.github.io/
- Gulp http://gulpjs.com/
- MongoDB https://www.mongodb.org/

7.2 Figures

Figure 1 - Example web dashboard provided by Agri Drain

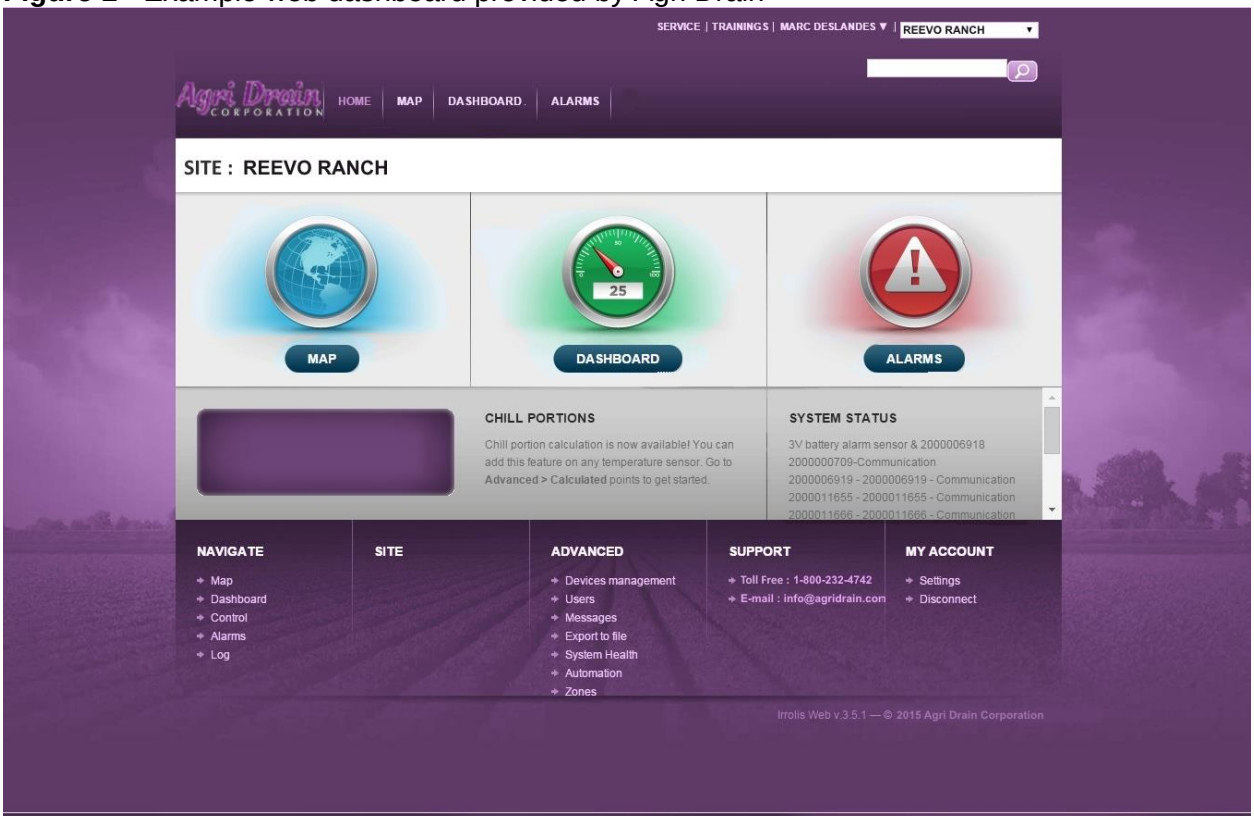

Figure 2 - Example data and control requirements provided by Agri Drain

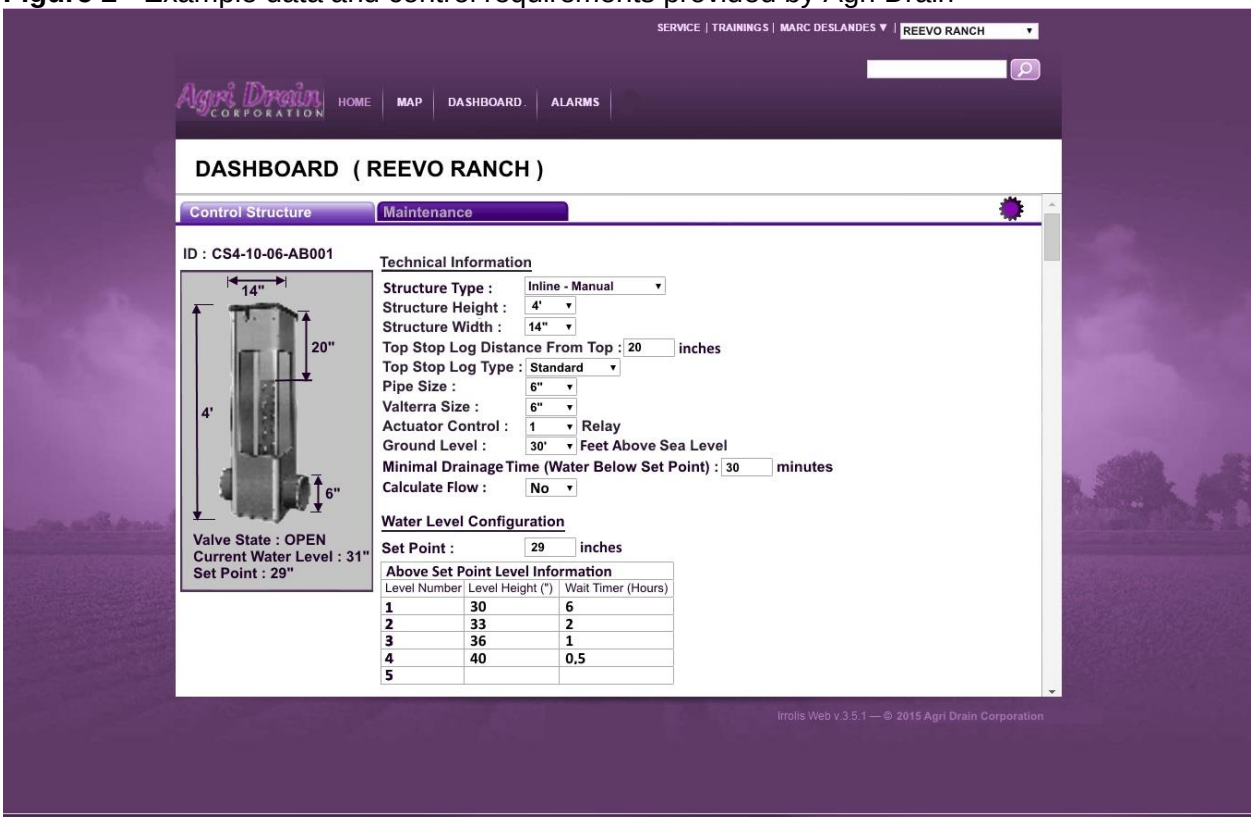

Figure 3 - Block diagram