DEC1609
Colin Hurd
Swarmy Ponpandi
Alex Clark - Team Leader
Alex Peck - Technical Lead
Chandler Heisler - Communications
Kurry Watson - Webmaster
DEC1609@iastate.edu
http://dec1609.sd.ece.iastate.edu

Revised: 10-23-16/3.0

SAVI Software Platform

PROJECT PLAN

Contents

Revised Project Design

Purpose

As the need for more and more food is constantly on the rise. Farmers need to be able to produce more food, which means to either increase yields or increase the amount of land being farmed. More farm land means more time needing to be spent in the field. But with the new robot and software we are developing farmers will be able to remotely control their tractors. This will open up the possibility for farmers to have multiple tractors in multiple feeds reducing the time the farmer needs to spend in each field.

Goals

Establish a server and database to collect and store data.

-store and serve data from fields

-store and serve data on tractors

-store and serve data from UI to Tractors

Create a web based UI to interface with the farm equipment

-Show real time updates on equipment progress in fields

-Allow user to change and update information on fields and machines

-Allow user to command equipment to move/stop/sync/nudge

Continuous development on UI as client adds requirements/changes

Deliverables

- A functioning UI
    - monitor the status of their tractor
    - send the tractor to staging and unloading
    - sync tractor and nudge it when synced
    - Draw, save, and load fields
    - Adjust speed of tractor
    - Must be easy to use, intuitive, and look appealing
- Create access to cloud database that allows UI to store and load data
- Add functionality to UI, server, or database as needed at clients request
    - This project has been a very iterative so concrete deliverables were often growing and changing throughout the year.
    - This was our most important deliverable

Design specs

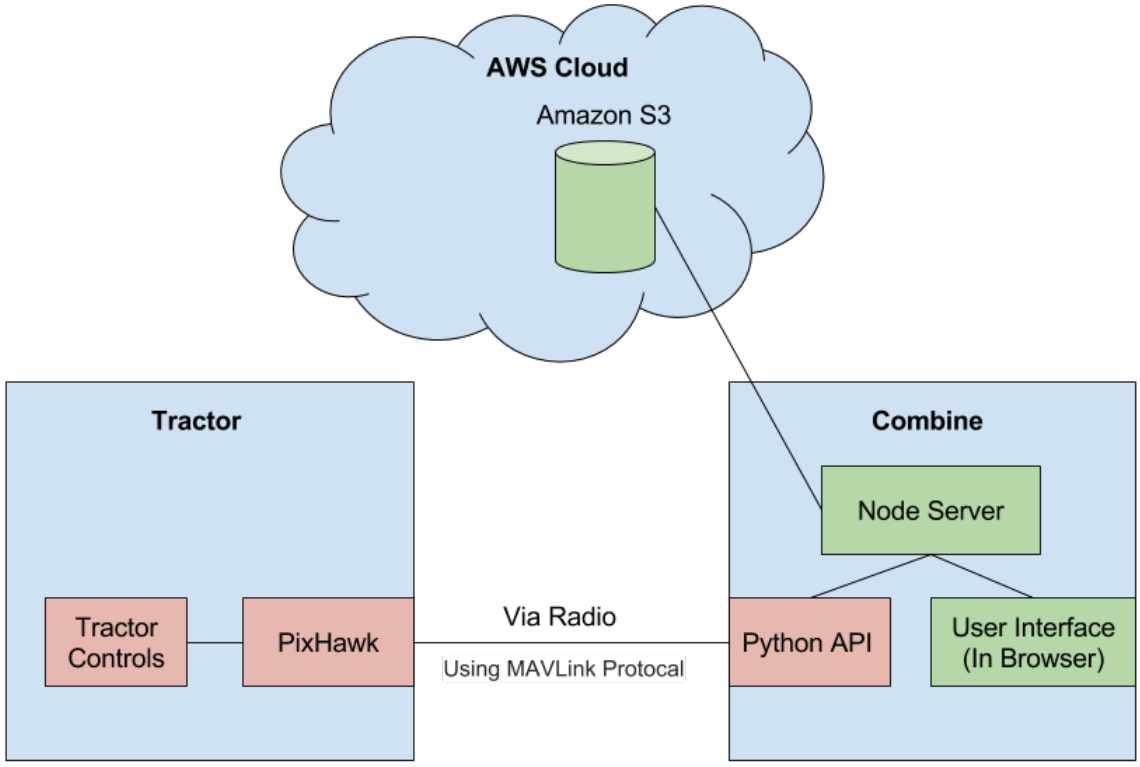
Non-functioning specs

1. Simple, easy to use GUI
2. Everything should be secure and encrypted
3. Cross Platform compatible (Windows, OS, Android)
4. Data portability for easy integration with precision ag data
5. UI should be easy to add new features to
6. UI should be easy to maintain when we are gone

Functioning specs

1. Create, load and save existing field maps
2. Should be able to control tractor without internet connection
3. UI needs to provide clear way to control tractor (speed, movement, etc.)
4. Possibility to be hosted on cloud
5. Store and present all tractor information in real time as needed
6. Allow multiple views for different farms/sets of equipment

Implementation Details



Description

The green components are part of our project whereas the red components are just simply components we're integrating with. The node server is being hosted on the combine and speaks to the Python API and the user interface with REST calls. The Amazon S3 cloud storage is also accessible from our node server.

Analysis

With this design we met all the client's requirements. With node server is being locally hosted on the combine, it can communicate with the python API without internet. The UI can also be moved to the Amazon Web Services Cloud as a web app that is entirely dependent on internet (if this is desired in the future). This design also allows us to store information in the Amazon S3 cloud storage whenever there is a connection to do so.

This design also give us modularity, control, and efficiency when it comes to communicating with the equipment. There is one point of communication between our side and the rest of the project (Node Server -> Python API) and it is done with simple REST calls. Our plan takes into account all relevant needs of a farmer, and makes sure they will have reliable real time communication with their tractor. This also allows our UI to achieve a highly customizable front end status because we can modify the front end however we want as long as it sends the right calls to the python API.

Testing process and testing results

Manual testing

As we add features to the project each one is tested locally to ensure it works as expected. For this we simulated the python server as it was impractical to do all testing with the tractor and the real Python server. For most of the local testing it was fairly simple to see if something worked or didn't work and sense the code we wrote was extremely modular there was little worry about changes having negative effects on other aspects. Nonetheless before changes were pushed to master we retested all the functionality to insure it was working as intended.

Client testing/feedback

We get constant updates on what is working/broken and desirable/undesirable weekly by our client. He gives us feedback in real time, with weekly meetings, status updates, and project reviews. We also receive feedback from farmers who have used this product.

So far the UI has worked as expected and has been stable during in-field tests. All new features we have added work as expected and our client is happy with how the UI turned out.

Appendix I: “Operation Manual”

Setup

The client will never have to set up his own environment. This will always be handled by SmartAg. The only people who will need to know how to setup the project is the developers. In order to setup the project, pull the project from Github and walk through the detailed README of how to set it up.

How To Use

I. Fields Control
   A. The SmartAg logo is a button that contains all the settings options
   B. Creating a field
      1. Select “Create Field” button from the settings dropdown menu
      2. Create the field point by point.
      3. Name the field
   C. “Load Field” button loads in a static predefined field
   D. “View Fields” loads fields from the database and displays them in new menu
      1. Check what fields you want displayed and press the green arrow button to load those fields to the map
   E. Field Selection
      1. Clicking on a loaded field brings up a menu for that field
      2. You can modify, save, delete, and add inner boundaries this field from this menu
      3. Inner boundaries are places where the tractor should not go and are added similar to fields.

II. Pins
   A. There are 3 types of pins: Unloading, Staging, and Middle pin
   B. Pins are can be placed from the “Place Pins” menu in the settings dropdown
   C. Pin Placer menu
      1. User should click the “Place X Pin” button and then click the map to that place
      2. Once both pins are placed, press Confirm Pins

3. The Middle pin will appear between them but all pins can be moved

III. Tractor Navigation
   A. The UI must be switched to Operations Mode at the top of the screen
   B. The navigation buttons are in the bottom right
   C. The speed slider adjusts the speed of tractor with the top being 100% speed and the bottom being 0% or stopped.
      1. Speed is typically out of 30 mph but can be changed depending on tractor
   D. The Start/Stop button turns the tractor on and off and can be used to emergency stop if necessary
   E. Before following any command to move the tractor, the UI projects a path from the tractor to the destination and asked the user to confirm it.
      1. Upon confirming the path, the projected route stays on the map and keeps updating as the tractor moves
      2. Canceling the path makes the tractor continue as if you never told it a new path.
   F. When the Unload, Stage, or Sync button is pressed, the UI creates the route from tractor to the specified location
      1. If the middle pin is on, the tractor will path through the middle pin before going to it’s specified location
      2. The Middle Pin button above the Start/Stop button toggles the middle pin on and off.
   G. If the tractor is synced with the combine, nudge buttons appear around the sync button
      1. Nudge buttons allow the user to position the tractor in a precise location next to the combine.
   H. The Move button allows the user to create a custom path for the tractor to follow
      1. The Move button ignores the middle pin entirely

IV. Diagnostics and Feedback
   A. Tractor info is displayed in the lower right hand corner
      1. Displays tractor’s distance to destination, MPH, heading and mode
   B. A status bar is located at the bottom and tells the user specifically what the tractor is doing
   C. The status of the connection to the Python API from the Node Server is displayed in top left.
      1. Green = connected, red = disconnected as well as text
      2. If you lose connection with the Python API you can no longer control the tractor

Appendix II: Alternative/other initial versions of the design

1. At first we started the project thinking that we were going to use program the app in Java and Java Swing. We planned out our app in this way. As our client crystallized what he wanted from us in this project, he decided that he would like us to use javascript with particular libraries like React.
2. We were also not really sure on the scope of the project at first. Initially, it sounded like we would be working with a small robot and not only making a user interface for it, but programming the robot and handling communication between robot and UI. At the end of the first semester it was still not clear whether we would be working with the robot directly. By the second semester it was clear that we would not be, just building the the UI. After this realization was made, our scope narrowed and it allowed us to get more detailed on the scope we were working on.
3. At first we just starting building our UI and were not sure exactly how we were going to connect it to the robot or how we were going to connect it to the cloud. The addition of hosting our UI on a local node server solved both of these problems, and as the python API was actually built with the idea that we would be able to call it from our node server.
4. The UI underwent many different designs and different types of functionality. There were a few things we implemented that later dropped entirely. This included keyboard shortcuts for all the selectable buttons and being able to change the map from satellite view to street view. The design also changed a lot. Originally all the nudge buttons were permanent and were on the left side of the screen instead of the right. Also, the styles of the fields and changed often with the most recent change being turning the path projection from yellow to blue. There was also big design changes in the architecture. One of the biggest was implemented webpack into our project which wraps all the javascript and dependendencies into one file.

Appendix III: Other considerations

What We Learned

We learned a lot about Javascript and React and are very comfortable with both now. We also had a ton of practice with REST calls and understand how they work and are utilized. This was the first time any of us had worked with AWS and so learning how cloud storage works and how dynamic servers work was very useful and we will likely use AWS again in our careers. Node was a first for all of us and, while it took us awhile to get the hang of, we all now understand the ins and outs of Node well. We also had a

very agile work environment on this project so we got a ton practice with iterative programming.

Real World Impact

We all really enjoyed working on a project that will have real world implications. One of the points of most pride on this project is that we set the bedrock for an application that will likely actually help thousands of people. This is great experience getting to write code that will actually set to use and not just graded and discarded. Another thing that we had to keep in mind is that this code will have to be maintained when we are gone so the codebase needed to look as organized as possible. We took care to use as many good code practices as we could and refactored when needed to keep in maintable. There is a certain weight with building a UI that controls a huge and potentially dangerous machine. We took great care to make sure everything was working as we intended as we did not want any bugs to happen while in field (Although there were other safety precautions). Overall, we really enjoyed and benefitted from building SmartAg's first UI!