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

1 Introduction

  1.1 Project statement

  1.2 purpose

  1.3 Goals

2 Deliverables

3 Design

  3.1 Previous work/literature

  3.2 Proposed System Block diagram

  3.3 Assessment of Proposed methods

  3.4 Validation

4 Project Requirements/Specifications

  4.1 functional

  4.2 Non-functional

5 Challenges

6 Timeline

  6.1 First Semester

  6.2 Second Semester

7 Conclusions

8 References

9 Appendices

1 Introduction

1.1 Project statement

To create an intuitive and effective UI to allow the user to control farming equipment around their farm. This includes monitoring the equipment, taking in and serving up data in real time, and being able to have complete control over their equipment at all times.

1.2 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.

1.3 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

2 Deliverables

To meet the goals outlined in the introduction we will need the following 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

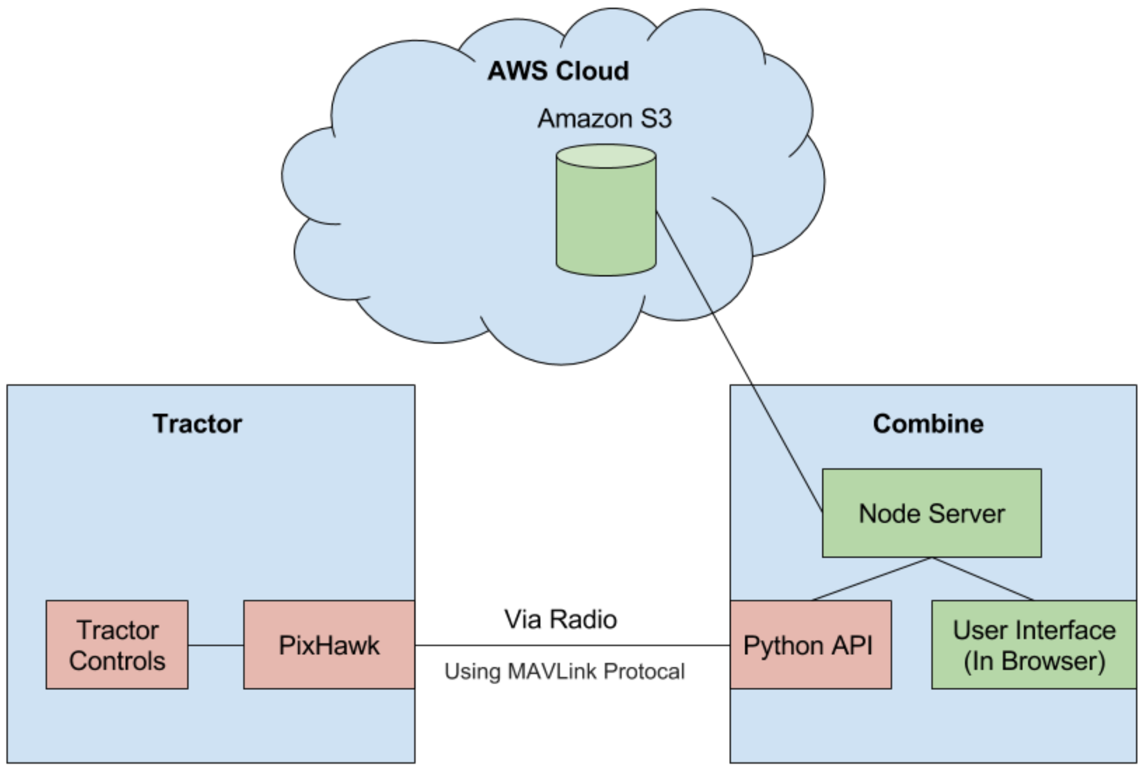
3 Design

3.1 Previous work/literature

Research was conducted about all forms of software architect. Advantages and disadvantages from various software designs were weighed. Some research included research about client server networking (Client Server Networking), and research about Multi Tier architecture (Multi Tier Architecture). We also did a lot of research into various databases and available cloud technologies. Conversed with professor Dr. Lie Tang about path planning.

Ultimately what dictated our design in the end though was our client's requirements. Our client wanted an application that could work on board the combine and communicate to the tractor in field without internet (Although internet would be guaranteed when starting the user interface). The client also wanted to be able to push and pull information from a secure database in the cloud, if internet was available. Another requirement was to have the option, if desired in the future, to move the application to the cloud in it's entirety (which would require the user to have internet at all times).

3.2 Proposed System Block diagram



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.

3.3 Assessment of Proposed methods (I updated this, but could use a read through/edit)

With this design we met all the client's requirements. Because the node server is being locally hosted on the combine, it can communicate with the python API without internet. It 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 real time communication with their tractor reliably. This also allows our UI to achieve that 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.

3.4 Validation

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.

4 Project Requirements/Specifications

4.1 functional

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

4.2 Non-functional

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

5 Challenges

5.1 Project setup and maintenance

The setup to this project is fairly elaborate. From scratch, it probably takes about 15 minutes to set up when you know exactly what you're doing and much much longer when you don't. There are tons of different modules and frameworks that need to set up and everyone needs to be on the same version. If someone adds a module, everyone has to download it on their setup as well. There was a lot of time spent talking between each other about issues like these.

5.2 Python Integration Without Tractor

Probably the hardest challenge we had was integration with the python API and lack of access to the tractor. Our code makes calls to the python API, which we did not design but did have access to. The python API needs to be connected to the tractor to work properly. Because we cannot connect to the tractor when we are coding, we needed to make a mock python API with the same API. Our mock server just returns static (or randomly generated) data so we can test the UI is calling the python API properly. The problem that we encountered was when we needed to test functionality in the UI that deals with tractor moving around and updating the path accordingly. All of this functionality is super easy to test in the field but much much harder to be verify without it.

5.3 Constantly Changing Expectations

This was a very iterative project, and what we decided to work on next was mostly dictated by what our client needed from us next. Sometimes we would structure our code in a way that made sense at the time it was written, but made less sense a few weeks later as the project developed. This means we either had to leave it in a less sensible structure causing more confusion later, or take the time to restructure it to make it a slightly better fit for the current version, but temporarily forgo new work. For example, when we first created pins that the tractor could drive to, there were just two. We added a third intermediate pin, and had to rewrite similar logic for this pin that spanned many many files. If we had been aware of the possibilities of more pins in the future, we would have generalized pin logic from the start. There were lots of incidents like this and that could not be accounted for at the project start.

6 Timeline(read through/edit if needed)

We were put on this project during its time of planning and research. Our specifications for the project have changed gradually over the course of this project, and our timeline has not stayed static during the process. We

6.1 First Semester

Our plan for the first semester was to fully research and get a full scope of our project, work on most of the front end software as we wait for the project specs to be complete.

We received initial project specs. on January 21st. For the month of January, we accomplished all of the administrative work. As a team we set up meetings times with our client, our advisor, as well as team meeting times.

For the months of February and March, we focused on planning, research, and basic development of code. Our client (Colin Hurd) wished us to use a language we have not had much exposure to, so we spent most of these months learning and framing our project.

During this time, the entire team we did research into React, a javascript framework that helps in creating HTML/UI pages for the application. Chandler and Alex Peck will be working on getting the backend and database setup, while Alex Clark and Kurry Watson start setting up the front end of the application.

April was spent putting what we had together in a working function, and getting ready for the summer development team to continue work on the project.

Our goals for the first semester are as follows,

- Understanding of Javascript and React
- Basic UI
- Database setup
- Basic functionality with robot including have the robot move, getting sensor information, and tracking location

6.2 Summer Semester

Over the summer, a lot of the specs changed, including:

- Get a working UI with basic functionality to start real world tests in clients fields
- Change over from prebuilt machinery, to a module that hooks up to existing farming machinery and hooks into its controls, monitoring devices, and gps
- (if time) split database from a single database, to an sql database for real time read/write, and an s3 database for file storage and dumps

Our team took part in expediting the task of a fully functional UI, getting all of the small things working to allow for tests in the field.

6.3 Second Semester

Our second semester timeline is volatile and open to reevaluation as the semester moves on based on our clients testing sessions, and how the farmers view the product. Because all basic functionality is taken care of, our second semester timeline looks has turned into more of a priority queue with the following specs.

- Fix bugs as they come up from validation testing
- Take client requests for adding functionality to website/databases as they come up
- Set up and move functionality to use SQL and S3 database
- Implement all data writing and storage for future farming analysis
- Continuous work on the backlog of visual/functionality updates

7 Conclusions

The goal of Smart-Ag is to create a system of Autonomously controlled farming equipment to help with the day to day tasks around the farm. We had the task of creating the user interface and cloud storage. The UI has complete control over the tractor and is pulling tractor dianostic data to display to the user. Our motivation is to give a farmer more time to do tasks a human needs to do, by helping him do tasks a robot can do. We believe our solution is an efficient and intuitive way to help farmers.

8 References

1. Project proposal: https://drive.google.com/open?id=0B5SA0PDouSRRTWo1U3BRUUtuQ00
2. Client specifications document: https://docs.google.com/document/d/1bHd2PPzdyn1devf1dfDf61sgbQtK4nlkZIRZU2UioSc/edit#
3. Old Client minimum viable product + precision ag document: https://docs.google.com/document/d/1xUuwwhmpM4v9Q-5Mc_mBep0YnUH2rYn2KQPdsRt9FG4/edit
4. Client Server Network: Advantages and Disadvantages: http://www.ianswer4u.com/2011/05/client-server-network-advantages-and.html#axzz40OsiBTgL
5. Multitier Architecture: https://en.wikipedia.org/wiki/Multitier_architecture