Smart Backpack Sprayer for Small-Scale Agriculture Applications

Final Report

Team Number:
sdmay20-53

Client:
IntelliSpray

Advisor:
Dr. Daji Qiao

Team Members/Roles:
Kevin Davis - Hardware Research Engineer
Sean Doran - Logistics Engineer
David Hayes - Communication Engineer
Madison Kriege - Communication Engineer
Donald Laracuente - API Research Engineer
Shuangquan Li - iOS Engineer

Team Email:
sdmay20-53@iastate.edu

Team Website:
http://sdmay20-53.sd.ece.iastate.edu/

Table of Contents

1.0 Revised Project Design

1.1 Acknowledgement

We would like to thank our client, IntelliSpray, for proposing the project and helping throughout the entirety of the project. They provided us with hardware, financial support to purchase necessary components, and with technical guidance and support as needed for the project. We would also like to thank our mentor, Dr. Qiao, for providing support with our documentation and for providing technical support. Dr. Qiao also ensured our group was focused on the project and that the team was working efficiently and in an organized way.

1.2 Problem and Project Statement

This project is modifying a backpack sprayer used for weed killers, bug spray, or similar chemicals on small farms and acreages. Commercial backpack sprayers currently do not offer a "smart" option, making it difficult for users to keep track of product placement information. We will develop an iOS application, paired with an Arduino device inside the backpack, to meet project requirements and provide users with spray information. This application will store location, flow rate, time of coverage, and other relevant data. The purpose of our project is to simplify the lives of our client base and reduce waste caused by excess chemical coverage.

1.3 Operational Environment

The operational environment for this project is an agricultural environment. This means that the solution will be exposed to rain, wind, dirt, dust, chemicals, and other elements. The solution must be robust such that it will function in a wide variety of conditions.

1.3.1 Requirements

1.3.1.1 Functional Requirements

- Read sensor data
    - Includes GPS location, rate of application
- Backpack sprayer must communicate with mobile app
- The app should analyze data
- Create a map of spray amount and time of application

1.3.1.2 Economical Requirements

Initial Budget of $750 from the clients. The clients will provide us with what is needed for the completion of the project. This includes hardware such as the compass, gps module, Arduino Mega 2560, flow sensor, and other components as needed. The clients will also supply accounts for software/backend as needed.

1.3.1.3 Non-Functional Requirements

- Portable
  - Easily worn and carried on one's back
- Scalable
- Reliable
- Intuitive and easy to learn/understand/use by a wide variety of users.

1.3.1.4 Environmental Requirements

- Waterproof: IP67
- Signal Range: 10m
- Temperature: -25 ~ 100 °C
- Altitude: 0 ~ 2500m
- Wind Speed: below 25 km/h

1.4 Intended Users and Uses

The intended user is one that works in a small scale agricultural environment. The user has a need to spray the land somewhat frequently, where learning about each spray application would be beneficial in order to maximize the efficiency of spraying and to learn from previous applications. The use of this product is for a small scale operation so that the battery life and amount of spray is sufficient to cover the desired area. The user will typically wear the product while using it.

1.5 Assumptions and Limitations

1.5.1 Assumptions

- Limited Number of Users
- The end product will be used in spring and summer
- The end product will not be used in the rain
- The end product will be self-contained inside of the backpack
- Then end product will measure in imperial and metric units

1.5.2 Limitations

- GPS Measurement Accuracy within a meter

- Flow Rate Measurements based off of what the sensors allow
- Battery-life must meet Realistic usage times
- Expenses are within the $750 initial budget

1.6 Expected End Product and Deliverables

The deliverables are a backpack sprayer solution and a mobile application. These two need to be able to communicate together and create the “smart” backpack sprayer.

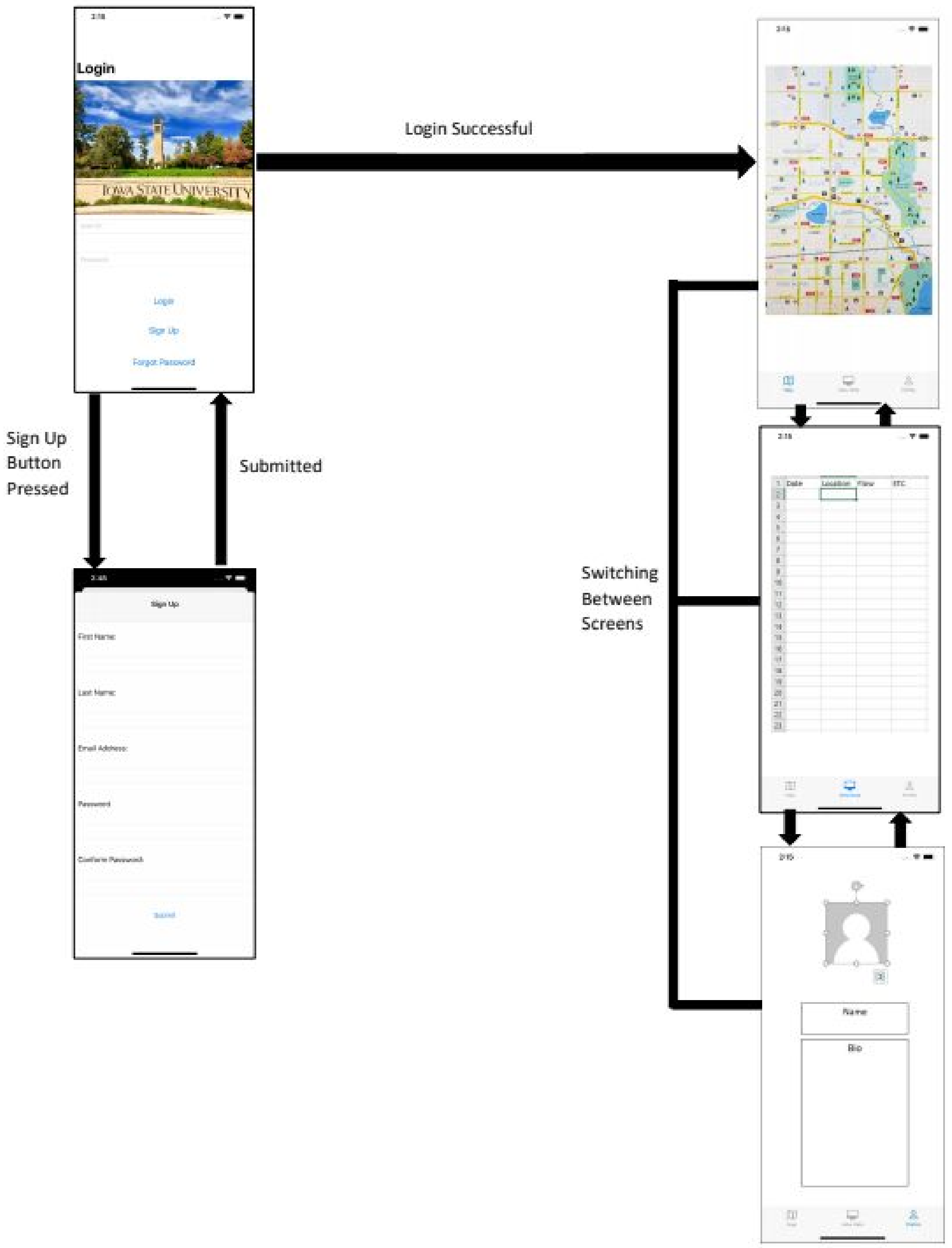
The backpack sprayer shall be a portable/wearable solution that will allow the user to spray substances on the small-scale agricultural operation. This will also include sensors that are able to collect GPS coordinates, directional data, and flow data from the backpack sprayer. An Arduino housed within the backpack will be a component of this solution. The Arduino will use Bluetooth to send data to the user’s mobile device.

The mobile application will be developed for iOS. It shall be able to receive data from the smart backpack using Bluetooth and then parse this data and represent it in a clear and effective way to the user. This will include a map of the locations sprayed as well as the flow rate at each location, the time of the spraying, direction of where the spray was directed, and what was being sprayed from the backpack. The application will be able to record historical data from previous sprays.

2.0 Specifications and Analysis

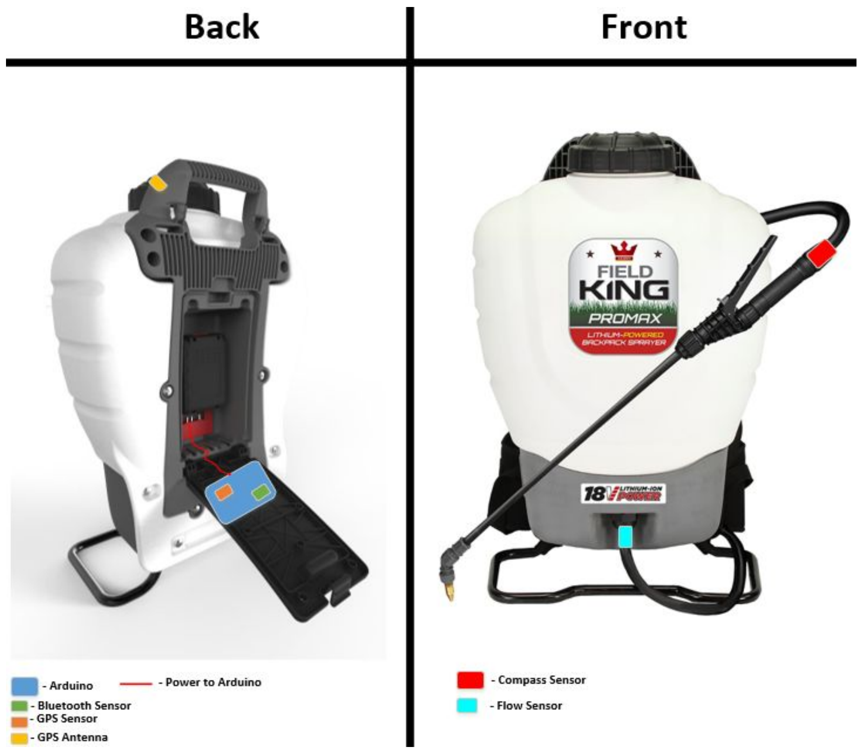
2.1 Proposed Design

So far, the proposed design can be summarized by the following diagrams/figures. We will use an iOS application, and the screens that will be in this app can be shown in the diagram as well.

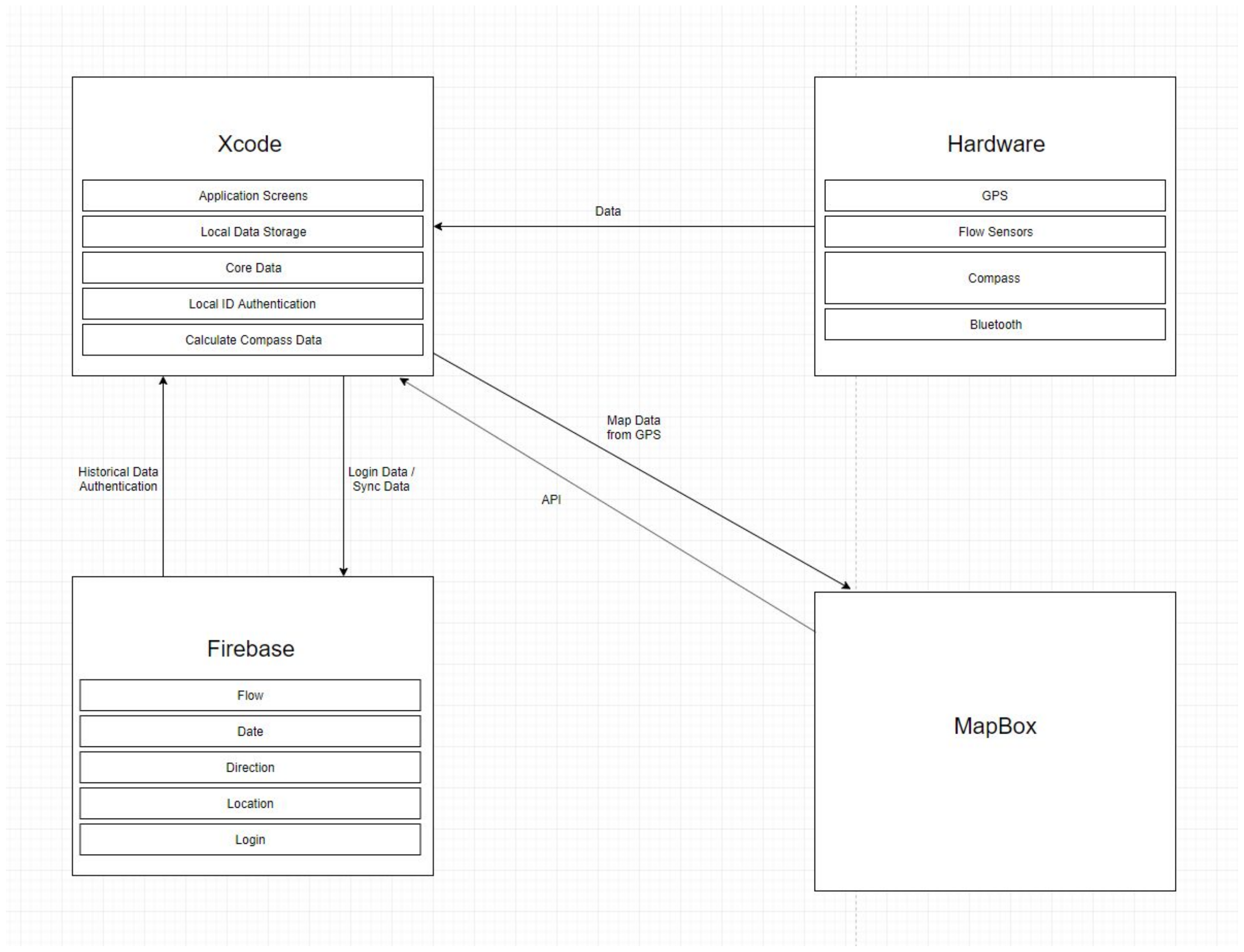


2.1.1 Proposed Application Views/Layout

In order to gather data about the spraying application, several sensors will need to be used during the time of the spray. An Arduino Mega 2560 has been purchased and will be used to control all of the sensors, gather data, and send data to the mobile application. The sensors that have been purchased are a GPS module(Neo 6m), a Bluetooth module, a flow meter, and a compass. The GPS module will be used to gather the coordinates of the sprayer, which will then be used to map which area has been sprayed. The compass will show direction or spray and give a more accurate description of what was sprayed. The flow sensor will be attached to the hose of the sprayer and record how much spray has been used and where the spray was used. The Bluetooth module will be used to send the aggregated data to the mobile application. This will cover the functional requirements of the hardware. Accuracy testing is necessary to ensure that all of the requirements are being fulfilled. If it is discovered that a requirement is not being satisfied, different hardware components will be evaluated as part of finding a solution. After evaluation, a PCB that will allow for easy connection of the hardware to the Arduino Mega 2560 will be designed and developed. This will make it simple to hold the necessary hardware on the backpack sprayer when in use.



2.1.2 Proposed Hardware/Backpack Design



2.1.3 Proposed Architecture Design

2.2 Design Analysis

Researching hardware components and purchasing those that best fit the needs of the project was the initial step in analyzing the project's design. Finding and verifying that hardware exists for the needs of the project proves that the overall design is feasible. After purchasing appropriate hardware, completing testing of each component to ensure the accuracy and quality of the component will satisfy both the functional and non-functional requirements is necessary. The GPS module has been tested outdoors and shown to have accuracy within 2-3 meters. This will be adequate for the project's needs. The flow meter has also been tested and shown to be accurate when running water through it. Running about 3L of water resulted in a reading of 3.03L, which is accurate enough for the requirements captured. One strength we have learned in the design analysis is how precise our equipment is. It has been higher than expectations and should help provide accurate data to the user. A few weaknesses we have are not getting different types of hardware to test the difference. We did get a couple of GPS sensors, but only one type of each of the other sensors.

2.3 Development Process

As the project becomes more mature and the necessary tasks are known, the development process that will be followed is Agile. This process will be utilized in all components of the project, from the mobile application to the APIs used, to the hardware development.

2.4 Design Plan

Use Case	***Requirement***	***Dependency***
Spray the fields	Portable/Backpack Sprayer	Purchase and modify sprayer/hardware.
Collect the data	Hardware sensors for GPS, flow, direction, Bluetooth, Arduino	Purchase appropriate hardware and connect to Arduino. Be able to spray fields.
Upload the data to the phone	Bluetooth connection between hardware and mobile application	Use Bluetooth module, ability to connect to the mobile application, be able to spray fields.
Upload the data from phone to database	Store historical data about spraying application	Have spray data, have a database setup, enable communication between mobile application and database.
View the data on Mapbox on phone	Display spray data on a map for the user.	Ability to use Mapbox/create a map, ability to get data from the backpack in a readable format on iOS.

2.4.1 Use case table

2.4.1 iOS

The initial design required Bluetooth communication between hardware devices and iPhones, data presentation and data persistence, and user interface construction. We chose to start implementing the user interface first, we have 3 main views for the UI which are Map View, Data View, and Profile View. For each view, they have their own sub-views. For the Map View, there are pins representing where did the user spare, and those pins can be tappen on after you tap on it, it will give you a detail of the time, GPS data, and direction at that pin. Also, the user's current location will be displayed as well. Moving on to the Data View, it contains all historical data for the past period of time, those data are grouped by activities, each activity represents each user usage, from start to end. Under those activities section, is a detailed data point, with complete data information, like, compass data, GPS data, temperature, direction, date, and time. Each of those individual data sections also corresponds to a pin in the Map View. Lastly, for Profile View, it contains user information, and a button to give the user a customised setting.

The next cycle of development, we start to do the integration between apps and hardware devices, we choose to use CoreBluetooh SDK for searching BLE devices, establishing a connection, and transmitting data. The main logic is, we first find the target device, then asking for its UUID, after confirming the connection, we start to transmit data. Upon data received by the iPhone, the corresponding method starts to analyze the JSON data, and use them to construct objects, and at the same time use the GPS from it to plot pins on the map. At this stage, the object we created using JSON is also stored in the local database. Later, we start to integrate local and cloud, we use firebase as our backend, during the time we store data to local we also send it to the cloud.

Last, we start the testing process by doing unit testing for each function, then we make a beta version of the app to distribute it to different iPhones via TestFlight, then gather feedback from it to improve our application.

2.4.2 API

The initial design of the API includes the communication between the iPhone application and Firebase/Mapbox.

Inside of Firebase, we created a project that included access to Firebase Auth and Firebase Firestore. This will allow users to securely login to the application and access their personal data. The Firestore instance allows us to store the data received from the hardware for later use. The main goal of the project was to allow users to view historical data in the application interface.

The next step of development was importing the needed CocaPods into the XCode project. This allows us to connect to various Firebase SDK from the iPhone application. More details about the iPhone application and its development can be found in “2.4.1 iOS”.

Our next development step of the project was the implementation of the Mapbox SDK. This SDK allows the iOS application to send and receive various mapping data from the Mapbox platform. Once the data is received from the hardware, it is processed on iOS and sent to Mapbox to create the various MapViews.

2.4.3 Hardware

The initial design required an analysis of hardware component options. Due to the availability and ease of development, an Arduino Mega 2560 was chosen. It includes plenty of interfaces to connect various sensors. The project also requires a flow meter sensor, a compass sensor, a GPS module, and a Bluetooth module to satisfy the client’s requirements. These components are also readily available, and able to be integrated with the Arduino Mega 2560. After consulting with the client and advisor, the components were selected.

The next step of development required testing each component individually. A small program was written to run on the Arduino that will capture the appropriate data for each sensor. The goal of this task is to ensure that the purchased sensors function as expected as well as have an accuracy that satisfies the hardware functional requirements. Most of the sensors have guides available online that aid in the development of his test programs.

Once each sensor has been tested for functionality on the Arduino Mega 2560, integrating all sensors together in one program was the next task. The goal of this task was to have one program that could collect data from all the sensors and send it via Bluetooth to the phone. To accomplish this task, we used snippets of code from our individual testing and integrated them together. The program collects data at a 1-second frequency.

One of the components that proved most challenging is the GPS module. The reliability of the module has proved troublesome and often takes a prolonged period of time to begin gathering data and reporting the coordinates to the Arduino. Because of this, the mapping feature and data packaging are delayed, or rendered infeasible/useful. In the future, other GPS sensors would be considered, with the goal to enhance the functionality of the hardware as a hole on the backpack sprayer.

With support, guidance, and approval, a backpack sprayer Field King 190515 was chosen and purchased. The main criteria for the backpack are size, battery life, and the ability to mount various hardware components.

After initial testing, it was determined that a PCB designed to allow all sensors to easily connect to the Arduino will be beneficial to the project. Development of a PCB that is of identical size to the Arduino Mega 2560, has headers that each sensor can connect to, and removes the use of many wires on the Arduino began as a first version. (add more about it)

Once creating the first version of the PCB, soldering the components and headers on allowed for members of the team to learn new skills, while also benefiting the project. 3 PCBs were ordered through ETG, so the team was able to build upon a first attempt to produce a higher quality PCB, using the labs on campus. However, after fitting the PCB and Arduino Mega 2560 into the backpack, the fit is too small to comfortably power the Arduino while the backpack's battery is also in the compartment.

To solve this, an alternative to the Arduino Mega 2560 was considered. This alternative is the Arduino UNO, due to its smaller footprint. It contains the same functionality that is required but will fit better in the backpack. A second PCB, developed for the Arduino UNO, was completed. Due to constraints of this semester, the team was unable to purchase and order this 2nd PCB.

3.0 Statement of Work

3.1 Previous Work and Literature

While we are not directly following any other products, similar solutions do exist for large scale agricultural operations. These solutions allow one to collect smart data about spraying applications, such as flow, time, location, etc. - Similar to the goals of this project. The purpose of this project is to use similar technology, but for a small scale operation. The usage of these technologies is similar, but the solution developed for this project is unique and not based on any particular existing work.

Due to events of the semester that are outside the control of this project, testing, integration, and work on the project was hindered. The team was unable to meet in person for testing, development, and work in general in the ladder part of the fall semester. This made producing a comprehensive deliverable as outlined originally unrealistic. After talking with the client and advisor, adjustments were made to complete as much work as possible. The hardware was integrated as much as possible, but testing with the application was not realistic due to the aforementioned constraints. Much of the project's individual components are complete, and there are no foreseen issues in the integration of these components.

3.2 Technology Considerations

There is a lot of technology available for this project. There are many microcontrollers out there that we could use. One of the strengths because we have so many options is the controllers are very cheap. One weakness is that you cannot test out to see what microcontroller would work the best. Some of the trade-offs we make are we just got a microcontroller that was pretty cheap and did not do too much research into it. There are a lot of considerations that need to be made with the hardware. We are still working out what all of these are and will expand this section. For the mobile application, both Android and iOS were considered. Given group experience and opportunity for growth, iOS was decided upon. As for the mapping ability, the main considerations were Google Maps and Mapbox. Ultimately Mapbox was chosen for cost and feature reasons.

3.3 Task Decomposition

To solve the problems the team faces and accomplish the outlined goals, the team has split into 3 teams. One team focuses on the mobile application, one focuses on the API and the backend, and the third team focuses on hardware. While all teams must communicate and work together, each team is able to focus on specific challenges and collaboration amongst the team to progress the project.

3.4 Possible Risks and Risk Management

The main risks that the project faces are related to the technology being used. The hardware sensors, mapping tools/API, and communication techniques. If these end up not working after considerable time is devoted towards a particular solution, the quality of the final product may be hindered. This is why it is important to continuously test and consider possible impacts on the project throughout the process. For example, the GPS module proved to be inaccurate during initial testing. Because of this, the team decided to purchase a more accurate antenna to help satisfy the GPS functional requirements.

3.5 Project Proposed Milestones and Evaluation Criteria

Key milestones can be for each team - the iOS team, the API team, and the hardware team - as well as looking at the project as a whole. Creating a running application that outlines the necessary functionality is a milestone for the iOS team. Talking with the backend and establishing proper communication with APIs is another milestone. Creating a backpack sprayer that can hold the necessary hardware and collect data is a hardware milestone. Sending real data from the backpack to the application is a milestone too. Putting everything together and displaying smart spraying data to the user is the ultimate milestone for the project.

Integration and final testing will lead to a functional prototype, so this is one of the last large milestones of the project. The deliverable that is expected is a working prototype, so being able to create this prototype and do formal testing is near the completion of the project.

For each milestone, test cases that align with the requirements and ensure that the solution is properly made, by looking at efficiency, user experience, accuracy, etc. are some of the ways that these milestones will be evaluated.

3.6 Project Tracking Procedures

We will use GitLab Issues to track the project’s tasks and progress throughout the course of the project, including both this semester and next semester. We will be able to centralize all of the work with this tool and effectively track progress at all times, as well as have the ability to go back and look at issues and work that has been completed. It will allow for the planning of the project too.

3.7 Expected Results and Validation

Along with meeting the functional and non-functional requirements, the desired outcome of this project is to be able to report spraying information back to a mobile device. The sprayer should be able to wirelessly communicate with an app to share data such as chemical, flow rate, time, GPS coordinates of spray, and direction. This information should be able to be displayed in an easy to understand map for the user. Users should be able to store historical data related to spraying applications.

Due to the changes of the semester, and the fact that most of the team was unable to meet in person for formal testing and integration. This changes our expected results. The team is only able to complete each component individually, and do “as much as possible” integration. So, the result of this is that the prototype will need more testing and finalization to satisfy the original expected results. The validation has changed to be less strict and more in line with the updated expected results.

4.0 Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

1. Design Documentation V1- Due Oct 6, 2019
2. Weekly Reports Interspersed throughout Semester
3. Hardware Delivery - Oct 9, 2019
4. Assembling Hardware - Oct 23, 2019
5. Design Documentation V2 - Due Oct 29-31, 2019
6. Software Completion - Nov 16, 2019
7. Integration - Dec 2, 2019
8. Design Documentation V3 (Final For Semester 1) - Due Dec 3-5, 2019
9. Prototype for Software and Hardware - Due Dec 09, 2019
10. Receive Physical Backpack - Due Jan 31, 2020
11. Testing of Prototype v1 - January 13-February 29 2020
12. Prototype v1- Due Feb 28, 2020
13. Testing of Prototype v2 - March 2020
14. Prototype v2 - Due March 31, 2020
15. Testing for final Prototype - March-May 7, 2020
16. Final Prototype - Due May 7, 2020

4.1.1 Project Gantt Chart

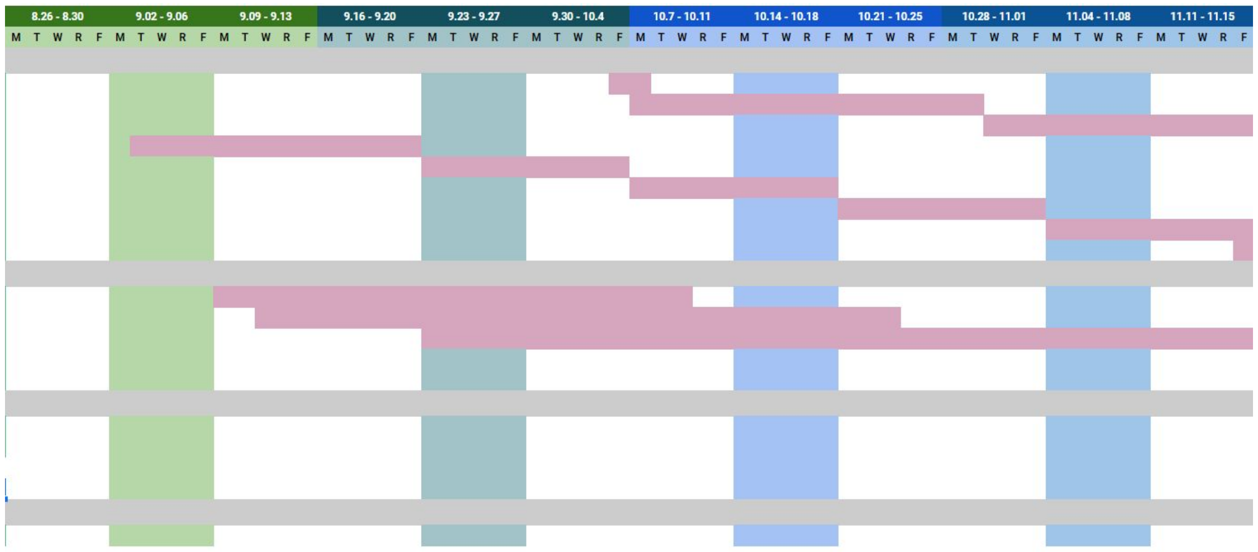
4.2 Feasibility Assessment

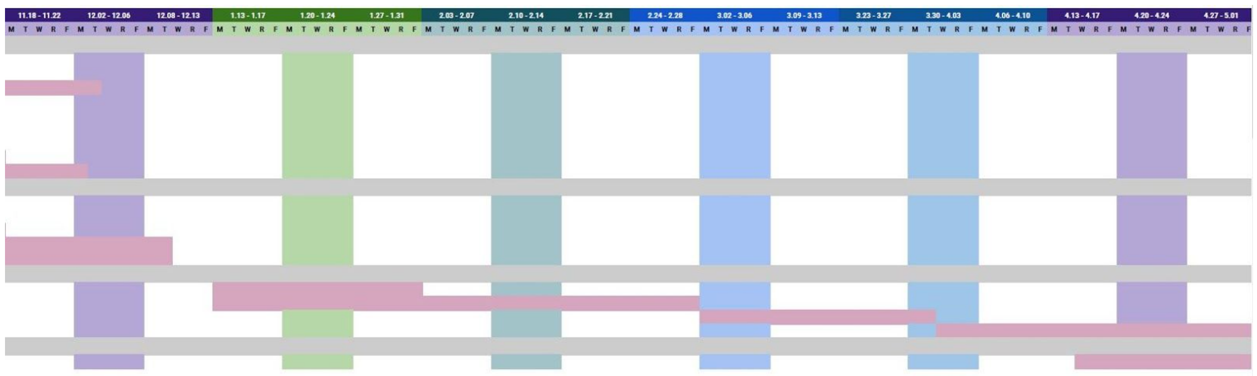
The end product of this project will be a backpack sprayer that has been modified to collect data, such as GPS location, the direction of spray, and the flow of spray and can communicate this data, along with other data, to a mobile application. Given existing hardware components, it is realistic to accomplish this goal, and the team has experience with iOS application development which will help in this project. The main challenges will be using new tools and APIs and getting these technologies to work together and result in a successful project.

Senior Design Project

PROJECT TITLE	Smart Backpack Sprayer

TASK TITLE	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
Planning				
Design Document V1	10/4/2019	10/7/2019	3	100%
Design Document V2	10/7/2019	10/29/2019	22	100%
Design Document V3	10/30/2019	12/3/2019	33	100%
Weekly Report 1	9/2/2019	9/20/2019	18	100%
Weekly Report 2	9/21/2019	10/4/2019	13	100%
Weekly Report 3	10/7/2019	10/18/2019	11	100%
Weekly Report 4	10/21/2019	11/1/2019	10	100%
Weekly Report 5	11/4/2019	11/15/2019	11	100%
Weekly Report 6	11/18/2019	12/2/2019	14	100%
Fall Semester				
Hardware Delivery	9/9/2019	10/9/2019	30	100%
Hardware Assembly	9/11/2019	10/23/2019	42	100%
Software Development	9/23/2019	11/15/2019	52	100%
Software Prototyping	11/18/2019	12/9/2019	21	100%
Hardware Prototyping	11/18/2019	12/9/2019	21	100%
Spring Semester				
Receive Final Hardware	1/13/2020	1/31/2020	18	100%
Prototype Testing 1	1/13/2020	2/28/2020	45	100%
Prototype Testing 2	3/2/2020	3/31/2020	29	0%
Prototype Testing 3	4/1/2020	5/1/2020	30	0%
Finalization				
Prepare Final Presentation	4/15/2020	5/1/2020	16	100%





4.3 Personnel Effort Requirements

Task	***Hours***	***Resource***	***Textual Reference/Explanation***
Research Hardware	8	David and Kevin	Satisfy functional requirements. Find hardware that will give necessary data.
Research APIs	8	Sean and Donald	Determine if Mapbox is suitable. Determine how the project will use APIs - logistics between the app, etc.
Research iOS	8	Shuangquan and Madison	Determine the platform/tools that will be used for the iOS application.
Develop Hardware	64	David and Kevin	Use each hardware component, gather data from Arduino.
Develop APIs	64	Sean and Donald	Begin to create a map - how the map will be used for the project - use mock data until real data is available from the application

Develop iOS	64	Shuangquan and Madison	Create iOS an application - all aspects to fulfill requirements and create a user-friendly experience.
Test Hardware	32	David and Kevin	Ensure that hardware data is accurate and proper. Ensure it is usable by the application.
Test APIs	32	Sean and Donald	Make sure the API part works
Test iOS	32	Shuangquan and Madison	Make sure the iOS Partworks
Integration Testing	16	Everyone	Make sure the integration between the items works
Final Testing	128	Everyone	Make sure it works and fulfills the project requirements

4.3.1 Personal Effort Table

4.4 Other Resource Requirements

Aside from assistance in purchasing the hardware and accounts needed to complete the project, the only resources that will be required is guidance and suggestions from the project's clients and advisor.

4.5 Financial Requirements

Part	**Description**	**Quantity**	**Unit Price**	**Total**
Arduino	Mega 2560	1	$13.99	$13.99
Sprayer	Field King	1	$154.95	$154.95
Flow Meter	1/2" Flow Meter	1	$9.95	$9.95
Compass Sensor	Compass	1	$8.39	$8.39

GPS Sensor	Neo 6M	1	$12.99	$12.99
Bluetooth Sensor	DSD TECH	1	$7.99	$7.99
3/4" to 1/2" Adapter	Hose Adapter	1	$7.99	$7.99
1/2" to 3/4" Adapter	Hose Adapter	1	$3.21	$3.21
GPS Antenna	Gain Antenna	1	$14.99	$14.99
			Total:	$234.45

4.5.1 Bill of Materials/Financial Requirements

5.0 Testing and Implementation

This project will require unit testing for the application, integrity testing for the backend and API components, and user-study testing for the hardware/sensors to ensure that the requirements are fulfilled. Each requirement will require specific testing. The non-functional requirements will need user-study testing. Testing hardware for accuracy is another example of user-study testing. The individual components that will need to be tested are the application, specific pages, functions, and use cases, the backend, map features, database interactions, hardware sensors, hardware accuracy, hardware reliability, and non-functional requirements. These will be outlined in specific tests and traced back to each requirement, to ensure that each is verified by test(s).

5.1 Interface Specifications

5.1.1 For software (iOS)

1. XCTest framework - overall tests
2. XCTestCase - unit tests and performance tests.
3. XCUIElementQuery - user interface tests.

5.1.2 For Hardware

No interfacing needed for hardware testing. This will be completed by testing sensors with known values and comparing the reported values. User-study tests will be used for these requirements. The nature of these tests will be easily verifiable when using known data, such as amounts of liquid sprayed, location, direction sensors are facing, etc.

5.2 Hardware and Software

5.2.1 For software (iOS)

We are using Xcode/XCTest to do all the testing which is iOS related.
Use the XCTest framework to write unit tests for the projects that integrate seamlessly with Xcode's testing workflow. Tests assert that certain conditions are satisfied during code execution, and record test failures (with optional messages) if those conditions are not satisfied. Tests can also measure the performance of blocks of code to check for performance regressions and can interact with an application's UI to validate user interaction flows.

5.2.1 For Hardware

To test hardware, we will use known values, such as known volumes of liquids for the flow sensor, known directions (cardinal directions primarily) for the compass, and GPS coordinates (longitude and latitude) and mapping software to compare GPS reported values. This will allow us to determine the accuracy and functionality of the hardware.

5.3 Functional Testing

Examples include unit, integration, system, acceptance testing

ID	Type of Tests	Description
1	Unit Test	Tests login button can log the user in
2	Unit Test	Tests sign up function can register a new user
3	Unit Test	A tests table view can display correct data
4	Unit Test	Tests cached data can be stored in Core Data
5	Unit Test	Tests the data can be displayed properly in map view
6	Integration Test	Tests the data can be received from the hardware device
7	Integration Test	Tests the data can be

		rendered in map view properly
8	Integration Test	Tests data can be sent out to cloud database
9	System Test	Test iOS App can work with all hardware devices
10	System Test	Tests the system complies all regulatory and legal requirements
11	Acceptance Test	Tests iOS App meets all requirements

5.3.1 iOS Tests Table

ID	Type of Tests	Description
1	User Study	After spraying a known amount of liquid, verify accuracy of Flow Sensor is within accuracy requirement
2	User Study	After facing a known direction, verify the compass reports the appropriate direction
3	User Study	After spraying at a known location, verify that the GPS reports a location within the accuracy requirement
4	Acceptance Test	Verify all hardware requirements are met.

5.3.2 Hardware Tests Table

5.4 Non-Functional Testing

Testing for performance, security, usability, compatibility

ID	Type of Tests	Description

1	Performance Test	Tests app launch speed is within 3 seconds
2	Performance / Usability / Compatibility Test	Tests UI is fluent with large amount of data been displayed
3	Usability/Compatibility Test	Tests app runs on all iOS devices
4	Usability/Compatibility Test	Tests the UI of the all runs all iOS model
5	Security Test	Data can only be viewed by the owner

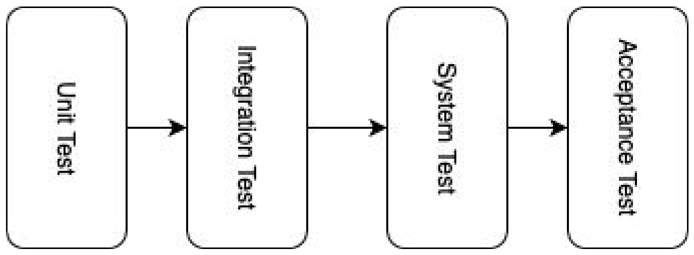
5.4.1 iOS Non-Functional Tests Table

ID	Type of Tests	Description
1	Acceptance Test	Verify that non-functional requirements are met (weight, portability, etc.)
2	Reliability Test	Verify that hardware sensors function as expected to adhere to usability requirements (ease of use).
3	Usability Test	Verify that data is transferable to the application when in use (connection is strong enough)
4	Usability Test	Verify hardware works in required weather conditions

5.4.2 Hardware Non-Functional Tests Table

5.5 Process

– Explain how each method indicated in Section 2 was tested
– Flow diagram of the process if applicable (should be for most projects)



5.5.1 Flow Diagram

5.6 Results

Currently, the initial operations of the various components have been verified and tested informally. The hardware components have been tested for accuracy individually, and an integrated program has been developed. The backend and map service has been developed and tested with mock data. The iOS app provides an easy to use user interface that allows for accounts, map viewing, spraying application data reading, and other features. The communication via Bluetooth is also functional and the data is able to be packaged in JSON format and sent from the hardware to the iOS application. The integration of all components is functional, but needs more thorough testing and formal completion to be considered a truly functional prototype.

5.6.1 iOS

For the iOS application, we successfully implemented the application. It's capable of logging/signing up users, showing the map, plotting pins, storing data both locally and in the cloud, and editing user information/configuration. Data can be transmitted between hardware devices and iPhones, BLE devices can be scanned, found, connected. In addition, the application can be installed and run in all kinds of iOS devices running iOS 13 and above.

5.6.2 API

The Firebase API which lets users securely login to the application and store/receive historical data works as intended. The use of Firebase Firestore allows for us to set up rules to only keep data active for a set period of time. In addition, we have implemented features such as Forgot Password and only allowing one account per email address.

The results for MapBox are within our current given margin of error. The API is currently taking in the data provided by the GPS in the hardware and is displaying them correctly in the iOS application.

5.6.3 Hardware

For the hardware components of the project, the team successfully tested each hardware component. Each component's functionality passed, though the GPS reliability is not as the requirements specified. This is an area for improvement in the future. Due to the changes in the spring semester, some tests were not fully completed, so it is possible that once more integration and formal testing have been completed, more improvements are necessary. The team learned that the hardware components satisfy the requirements outlined in the project, but improvements could be made. Due to sizing, an Arduino UNO PCB has been developed, and the transition from Arduino Mega 2560 to Arduino UNO will yield more testing.

6.0 Closing Material

6.1 Conclusion

Currently, each component of the project is under development. Individually, each area, including the iOS application, hardware/sensors and backend and APIs have made progress towards their goals. The next step will be finalizing where each stands, and begin to integrate each of these together, to create the final product. The goals include creating a functioning application that receives data from the hardware, which then uses the backend and APIs to display the data in a meaningful and efficient way to the end user. This will give the end user smart data about a spraying application. The data will include GPS locations of the spray, direction of the spray at each location, data about the spray (what chemical, when it was sprayed, etc.), and the flow rate, or how much was sprayed, during the application.

6.2 References

This will likely be different than in the project plan, since these will be technical references versus related work / market survey references. Do professional citation style(ex. IEEE).

"XCTest," *XCTest | Apple Developer Documentation*. [Online]. Available: https://developer.apple.com/documentation/xctest. [Accessed: 18-Nov-2019].