Automating Inventory Management

Routing through Sensor Networks

Project Plan

Team Number: sdmay19-29
Client: Jimmy Paul
Adviser: Dr. Goce Trajcevski

Team Members:
David Bis — Meeting Facilitator, Backend Developer
Hanna Moser — Meeting Scribe, Frontend Developer
Adam Hauge — Report Manager, Computer Network Architect
Ben Gruman — Resource Acquisition, Hardware Architect
Sam Guenette — Public Relations, Backend Developer
Noah Bix — Documentation Manager, Hardware Architect

Team email: sdmay19-29@iastate.edu
Team Website: http://sdmay19-29.sd.ece.iastate.edu/

Revised: December 2 2018 / Version 3.0

Table of Contents

List of Figures

List of Tables

List of Definitions

- AWS - Amazon Web Services
- Crafty - Crafty, LLC.
- ETG - Electronics and Technology Group
- MVC - Model-View-Controller
- MVP - Minimum Viable Product
- RFID - Radio Frequency Identification

1 Introductory Material

The objective of this project is to implement a system that can effectively monitor goods in office pantries and create efficient orders based off their current status. Using these orders, an optimal delivery route will be calculated to each office location.

1.1 Acknowledgment

The Inventory Automation Team would like to thank the Iowa State University Department of Electrical and Computer Engineering for providing this team with a professional experience, quality resources, and consultation with experts. The team appreciates the willingness of the Electronics and Technology Group's (ETG) to help provide hardware and server components for the project. Special thanks also go to Iowa State's Dr. Goce Trajcevski for weekly consultation with regards to dealing with technical issues and moving forward throughout the development process. Appreciation also goes towards our client, Crafty LLC., for the given project. Special thanks goes to CTO and co-founder Jimmy Paul for making time to meet with the development team to gain more information and feedback throughout the project's development.

1.2 Problem and Project Statement

Crafty, LLC is a warehouse company that delivers food to office pantries. Their current infrastructure is based on having an employee at each of their client offices handling shipment orders and physically monitoring when the company's pantry needs to reorder certain products, which is prone to human error. Furthermore, warehouse truck routing becomes severely inefficient and expensive from restocking at individual offices based on separate orders.

Our objective is to provide an integrated solution that will enable more effective management of office pantries. Specifically, we aim at balancing the local (i.e. at the level of individual offices) and the global (i.e. at the level of a geographic region) management in terms of both customer satisfaction and overall efficiency for Crafty. Towards that, our proposed solution consists of two main collaborative modules that we plan to develop: (1) a microcontroller device that will monitor the quantities of different items in an individual office pantry, coupled with other existing monitoring tools (e.g. barcode scanners); (2) software tools that will be able to combine the multiple restocking notifications from different offices, and plan effective shipment (i.e. routes) and delivery.

1.3 Operational Environment

The microcontroller used to monitor office shipment levels is expected to be stored in a dry pantry area with a reliable power supply and WiFi connection. The current device consists of a microcontroller-connected "Smart-Bin" that is expected to fit on shelves. Any software setup with the device will be done through a web application. The web application will be available for Crafty employees stationed at office locations for setting up the physical device and in the warehouse for overseeing and handling shipment orders.

1.4 Intended Users and Intended Uses

There are three main users of our sensor network. Each user will interact with the sensor network through use of the web application component. The following are brief overviews of the three users:

1. *Pantry Employee*: Actor based in a given company office in charge of handling shipments sent from the warehouse. This user is in charge of setting up the microcontroller device and registering items for the device to monitor.

2. *Warehouse Employee*: Actor based in the company warehouse in charge of overseeing shipment orders and filling trucks with the proper items. This user will go into the application to view orders and routes. This user is expected to possess moderate technical experience.

3. *Warehouse Truck Driver*: Actor in charge of the physical delivery of items between the Crafty warehouse and client office pantries. This user will employ the online application to view their assigned shipping route for the day.

1.5 Assumptions and Limitations

Assumptions:

- Pantry sensor network has access to a reliable power source
- Pantry sensor network can connect to internet
- Pantry sensor network is not at risk of water damage
- Pantry Employee properly sets up sensor network and associates each monitoring device with the corresponding product being stored through the online application

Limitations:

- Accuracy of automatic orders is completely reliant on the accuracy of the sensors used in the sensor network
- Sensor network monitors product availability in bulk, not individually
- Sensor network can only make correct orders if the Crafty user properly establishes what type of products are being monitored

1.6 Expected End Product and Other Deliverables

The product deliverables will be split into (1) a proof-of-concept, (2) minimum viable product, and (3) a final product. The proof-of-concept and minimum viable product (MVP) will be delivered by the end of the fall 2018 semester. The finalized product will be delivered at the end of the Spring 2019 semester. An overall design manual of the product and its architecture will also be provided for any future development.

1. Proof-Of-Concept (October 25th, 2018)
    - The Proof-of-Concept prototype will prove the product's capabilities and potential to both monitor physical storage units and communicate that

information to monitor and analyze. Users will be able to store a product in a single "Smart-Bin" which will send that information to an SQL Database. Users can log into a web application to register the device and monitor storage levels of the product in the Smart-Bin.

2. Minimum Viable Product (March 15th, 2018)
    - The Minimum Viable Product will be deployed for beta-testing. Multiple cheaper Smart-Bin devices will communicate with a single WiFi component that sends information to the database. The web application component will monitor information and build an appropriate and optimized shipment orders.

3. Finalized Product (April 15th)
    - The Finalized Product is ready to integrate for use with the client's existing infrastructure. The smart-bin device is expected accurately monitor multiple products. The user interface will also provide easy setup, handling, monitoring, and registering of smart bin devices. The database is connected to a Cloud Management System to analyze data and learn to better optimize shipping routes and customer orders.

2 Proposed Approach and Statement of Work

This section will cover our technical solution and design to our problem statement.

2.1 Objective of the Task

The objective of this project is to design a sensor network capable of automatically counting and managing the inventory of a pantry stockroom. The data collected should help Crafty determine when they need to send shipments out to their clients to restock their pantry stockrooms. Based on the delivery schedule, the system should calculate an optimal route each day for delivery drivers to travel.

2.2 Functional Requirements

The project solution can be broken into two components: a sensor network and analytics software. The functional requirements for both components are shown below.

2.2.1 Sensor Network

FR.1: The system can keep track of the levels of various product in a customer's pantry stockroom automatically. This inventory should be sent over to Crafty to determine whether a product needs to be restocked.

FR.2: The user should be able to register/unregister sensor arrays to fit their unique stockroom.

2.2.2 Analytics Software

FR.3: The system can calculate an optimal route for deliveries for restocking client pantry stockrooms daily.

FR.4: The system can display the stockroom inventory data and filter it according to refined searches.

2.3 Constraints Considerations

In order to ensure that the project satisfies the necessary requirements, the team has taken into consideration the following constraints. These constraints include relevant non-functional requirements and standards to follow, such as IEEE standards.

2.3.1 Non-Functional Requirements

Scalability - The sensor network should be able to be replicated and integrated into multiple pantry stockrooms. The sensor network should also be able to easily support a variety of different products that can be found in stockrooms. This requirement will be accomplished by designing the sensor network to be modular and adaptable.

Data Integrity - The sensors should be accurate and consistent throughout their lifetimes in order to ensure the inventory is correctly monitored. This will be accomplished by employing a variety of sensors to audit each other's the readings. The sensors will also be tested to ensure accuracy.

Availability - The system should be available to update the inventory at least once per work day. It should also be available to determine delivery routes once per work day.

Deployment - The sensor network should be easily deployed and installed in pantry stockrooms. This will be accomplished by designing the sensor network architecture to be modular and adaptable to a variety of environments.

Usability - The initial setup of the sensor network by the Crafty employee should be intuitive. Since the number of sensors and the product they are detecting differs from stockroom to stockroom, and can be changed in a stockroom, the assigned Crafty employee should be easily able to select which sensor is monitoring a specific product.

Resilience - The sensor network will be deployed in an environment of constant change due to the movement of boxes in and out of the stock rooms. This can run risk of damaging the system if a box comes into unwanted contact with the sensors. The sensor arrays should be designed to be constantly protected by the expected motions of boxes in the environment.

2.3.2 Standard Protocols

The project will abide to the following standards:

- **IEEE 802.11**: Wi-Fi between ESP8266 and Raspberry Pi. Being that our solution involves communication of information between a local Wi-Fi and our own and database, development has to follow appropriate networking standards. Likewise, it is important that each of sensor devices are abide to the standard Wi-Fi protocol so that they can properly connect to the network and be discovered by the master Raspberry Pi.
- **IEEE 1012-2016 - IEEE Standard for System, Software, and Hardware Verification and Validation:** The project will contain a rigorous testing suite to ensure the solution satisfies all the necessary requirements. This will help verify and validate the practicality of our solution once it is implemented.
- **IEEE 1532 In-System Configuration of Programmable Devices** is essential since since our product is an IoT technology involving a massive network of monitoring devices that record and communicate information to each other as well as a web component. This involves a significant amount of hardware configuration and embedded software that needs to be implemented effectively and securely.

2.4 Previous Work and Literature

Automatic inventory management is not a new concept and has been integrated in the past. Companies such as Impinj have already explored applications of automating inventory management with medical supplies using RFID technology [1]. Similar to Crafty LLC, Impinj manages and stocks inventory for various clients. Their solution incorporates RFID tags to track what is moving in and out of the room. Unfortunately, RFID scanners are expensive and tagging as many items as Crafty does will be very costly. For this reason, using an RFID scanner falls short of meeting our non-functional scalability requirement, so finding less expensive options is important.

Another company in automated inventory management, Barcodes Inc., uses barcode scanners to manage inventory [2]. Barcodes are scanned by employees and the scanners used update the inventory data in the company database. Barcode scanners are relatively cheap and easy to implement. However, this system is still prone to human error as employees may forget to scan items. Therefore, this system does not meet our data integrity non-functional requirement. Though this method would be easy to implement, more automation will be needed to ensure greater data accuracy, thus removing the potential of human error.

With regards to the rerouting portion of the project, it is well known that there are companies that have developed phenomenal routing algorithms, such as Google and Apple. Companies such as Route4Me use a method called Dynamic Route Optimization which creates an efficient delivery route that travels to all of the given delivery locations [3]. These routing algorithms were mainly built for travelling to one destination and possibly through stopping points along the trip. Our client believes that the existing tools for finding efficient travelling routes do not apply directly to a trip with separate sub-routes such as would be seen when delivering products to several clients in a day. The routing algorithms that need to be developed for the project at hand need to take in all of the locations that need to be visited in one day. It is also required that they use information known about weather and traffic to create one master route that is the most efficient with regards to time and gas used.

2.5 Proposed Design

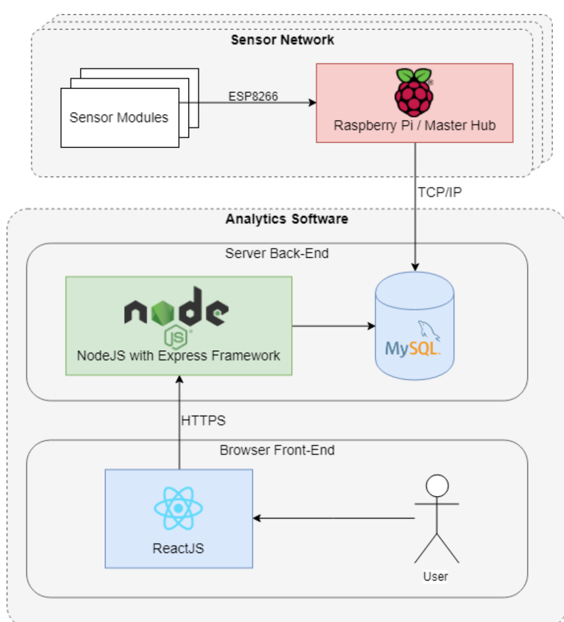

Figure 1: Project Block Diagram

The project will be composed of two systems: the sensor network and the analytics software. The sensor network will be responsible for monitoring stockroom inventory. The analytics software will be used to take client usage data and restocking order metrics in order to determine an optimal route for delivery each day. This two-system solution was selected to minimize coupling between them. The sensor network will only interact with the analytics software through an external API to update the database.

The sensor network will consist of a variety of sensors for redundancy, which will minimize error. The sensors being considered are as follows: weight sensors, barcode scanners, and sonar sensors.

Weight sensors will be placed in specific locations of Crafty's clients' storage locations to monitor the weight of product remaining in inventory. When the weight of a certain product gets below a certain threshold, Crafty will be notified that the client needs a new order of the product.
The weight sensors will be connected to an ESP8266 chip using the wiring shown in Figure 2.

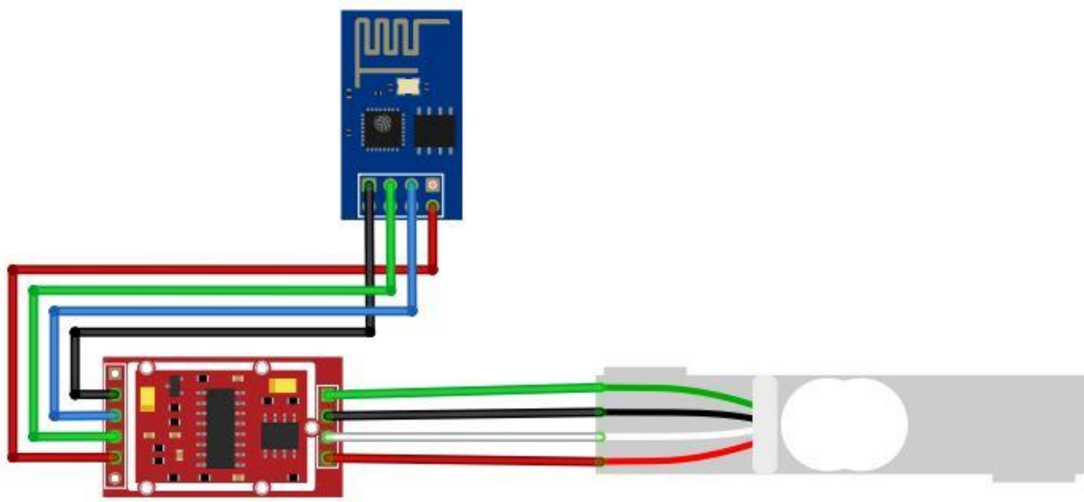

Figure 2: Load Cell Wiring Diagram

A barcode scanner will be incorporated into our design as well. This will serve a purpose of redundancy. Items will be scanned as they enter and exit the company's pantry, subtracting or adding to the current amount of inventory in that client's database. Similarly to the weight sensor, when the number of items in the client's stockroom gets below a certain threshold, Crafty will be notified to place an order for that item. The barcode scanner may also be used by on-site Crafty employes to register new items when their clients decide to make changes to their inventory.

To achieve maximum accuracy, sonar sensors will be the third and final sensor incorporated into our design. These sensors will be placed above each item to monitor the height of its pile. The sonar sensors will compliment the other sensors nicely. All three sensors together will provide an extremely high accuracy rate that one single sensor couldn't achieve on its own.

Similar to the weight sensors, the sonar sensors will be connected to an ESP8266 chip using the wiring diagram in Figure 3.

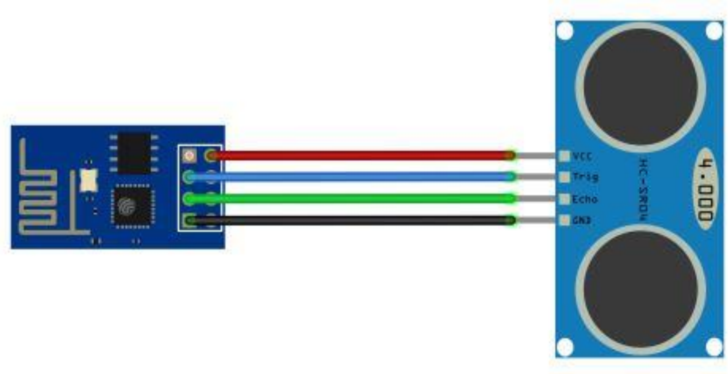

Figure 3: Sonar Sensor Wiring Diagram

The analytics software will composed of two parts: a web front-end and back-end. The web front-end will be made using the ReactJS framework, which was selected due to the team's familiarity with the framework. Likewise, since the framework is maintained by Facebook, the team has confidence about its continued support by them, as well as additional support from other developers who continue to contribute to it with additional libraries. The front-end will communicate with the back-end via HTTPS calls.

The back-end will be made in a NodeJS environment with the ExpressJS framework. The ExpressJS framework doesn't have a well-defined architecture required by the framework, unlike those found in alternatives such as Java Spring. However, the team recognizes the importance of such an architecture model, so the MVC model has been identified for use in the architecture. ExpressJS will allow for the the creation of an API to control the flow of data between the relevant components of the project. Data will be stored using a MySQL relational database. MySQL was chosen over alternatives such as MongoDB because the data necessary to be stored must be well-defined, so data manipulation and querying will be optimal for a relational database architecture. The planned database schema is shown below in Figure 4. Within the database, primary and foreign keys will be used to relate separate data tables and their information. An Entity-API called Sequalizer will be used to convert database information into a proper corresponding MVC model. This will make it easier and more scalable for accessing information and other information that relates to it. However, these relations rely on the databases being properly configured with the correct primary and foreign keys to connect different tables of data.

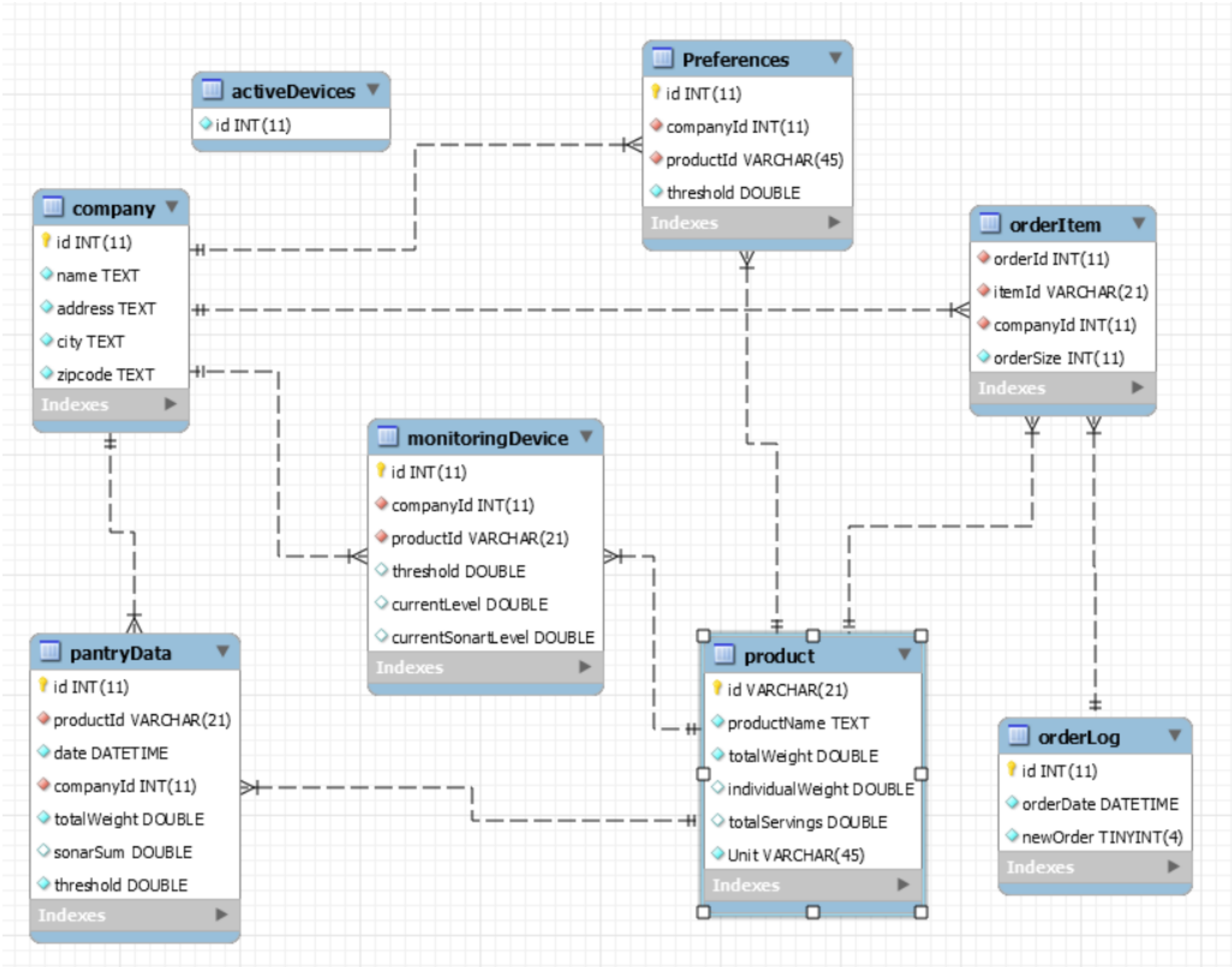

Figure 4: Database Schema

2.6 Technology Considerations

The project can be broken into two different systems: sensor network and the analytics software. This decision to keep these two system distinct is to ensure that they can be independently tested and are not reliant on one another. The sensor network block diagram is shown on Figure 5.

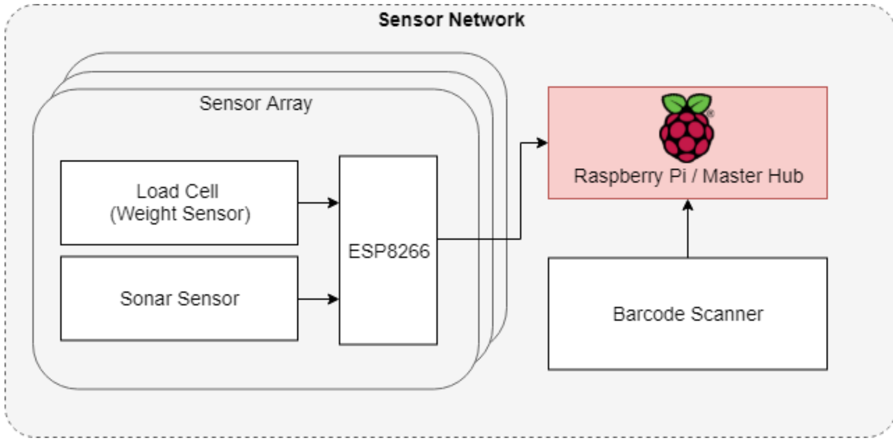

Figure 5: Sensor Network Block Diagram

The sensor network needs to have a central controller to gather all of the sensor data, so the team decided to use either an Arduino or a Raspberry Pi. Ultimately, a Raspberry Pi 3 was selected because of its built-in wireless module as well as an on-board Linux OS. The wireless module is particularly important because it will handle communication from the sensors to the database, which will be hosted remotely. The sensors will be independently powered and will communicate with the Raspberry Pi with an ESP8266 chip. Allowing the sensors to communicate wirelessly with the Raspberry Pi will grant flexibility when implementing the sensor network at any location. Another benefit is the on-board Linux OS, which helps the modifiability of the product. Any changes to the hardware orientation can be easily applied in the code, which can be accessed remotely if necessary. This also allows for remote deployments of software updates. Arduinos were ruled out early on, due to their lack of crucial features such as low-level configuration and an underlying operating system which assisted in refinement of our monitoring device architecture.

There are three categories of sensors being considered for the sensor network: barcode scanners, weight sensors, and sonar sensors. Although RFID tags were originally considered, they were replaced with a barcode scanner to reduce the unnecessary overhead cost of RFID tags and scanners. A similar procedure can be replicated using a barcode scanner instead. The barcode scanner will exist to scan and monitor entire boxes that are being moved in and out of the stockroom, monitoring macro-scale changes in inventory. Weight sensors will help provide more precise readings, such as detecting partially filled boxes. The reason for using multiple sensors is that they provide the readings with redundancy to minimize the accumulation of error and ensure the inventory readings remain accurate.

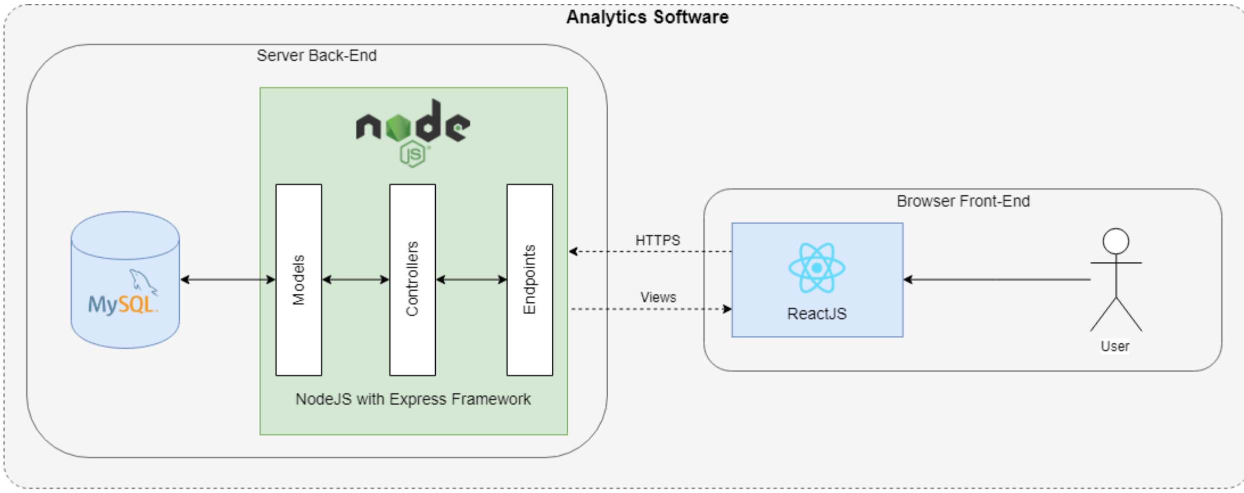

Figure 6: Analytics Software Block Diagram

Figure 6 displays a block diagram of the analytics software. The analytics software will use an MVC pattern. This is to allow for an easily understood environment that the team has previous experience with, but also to allow future developers to expand the project after the current team is complete. Likewise, a NodeJS environment was also selected for the back end due to the team's familiarity with it from previous work. In order to better realize the planned architecture, ExpressJS will be used to construct an API for the product. The back-end and database will be hosted on a remote virtual machine.

The database will be implemented with a MySQL relational database. A relational database was selected for its ability to easily manage data and enable users to balance between elementary querying and analytics, and to port it to a different platform for a more sophisticated analytics service.

The front-end will be implemented with ReactJS web application. This was chosen due to the team's familiarity with the technology and sticking with a JavaScript environment.

2.7 Safety Considerations

All software will be developed using individual personal computers. For an intermediary testing environment, the ETG's virtual machine services will be used. Cyber security attacks will be the most prominent safety issue to consider for the project. The transmission of data with regards to inventory does not pose a very large safety issue, however the truck routing data is considered a high safety risk. Team developers will take the necessary safety precautions to make sure that the software is safe from cyber security breaches.

All of the information gathered on both the inventory and routing side of the project will be stored in a MySQL database. The data stored with regards to keeping track of inventory will only contain information such as the type of the item, the amount of the item left, and the threshold value at which to automatically place a new order. None of this information poses a major safety threat. However, the information stored with respect to the route tracking will include information about the current location of a truck along with the projected route and arrival times. This could pose a security issue as a cyber security attack could lead to an outside source knowing the location of a delivery truck at all times along its route allowing for the ability to intercept the truck anywhere along the route. To address this security concern, all access to the database will be restricted by IP address, meaning that unless the IP address trying to access the database is that of the virtual machine or the Raspberry Pi, then access to the database will be denied.

A safety concern related to the hardware portion of the project is the safety of the sensors being used. For instance, if the sensors have any type of springing mechanism, it needs to be verified that items will not be launched onto clients when they try to remove or add items to the stock. Also, as some of the items being tracked by the sensors may be liquids, the sensors need to be waterproof to ensure that if there is a spill, the client will not be electrocuted.

2.8 Task Approach

The project will be broken into two major components: the sensor network and the analytics software. The two components will be developed simultaneously, having three members assigned to the sensor network and three members assigned to the analytics software. However, design plans and decisions will be worked on as team. This is to ensure all possible designs are considered to maximize the overall quality of our plans. The project will be divided into 4 phases, each advancing the development of a prototype to final product. The goals for each phase are as follows:

Phase 1

- Sensors can collect data
- Sensor data can be sent to remote database
- Front-end can visualize data from the database

Phase 2

- Integrate system for multiple sensors to be registered to a specific product (assemble sensor array)
- Orders can be generated based on measured inventory data

Phase 3

- Integrate multiple sensor arrays for variety of products (integrate master-slave system)
- Route can be determined from multiple order requests

Phase 4

- Integrate predictive analytics for ordering based on historical order habits

Each phase will be treated as an Agile sprint to ensure continuous improvement upon the planned architecture. To see a more in-depth decomposition of the timeline and Agile implementation, see Section 3.1. During each phase, our work will go through testing and validation requirements for further refinement and approval. To see more on our testing and validation, see 2.13.

2.9 Possible Risks and Risk Management

Title: Unfamiliarity with Technology
Information: Many members of the group are working with technology that is completely new to them. For instance, the front-end development will be mainly using JavaScript with the ReactJS framework which the developer for the front-end has not had much experience working with. On the back-end side of the project, the developers are not very familiar with cloud computing and hardware communication. Hardware developers coding skills are not up to standard and they are unfamiliar with Raspberry Pi.
Risk Response Strategy: In order to mitigate this risk we can secure outside sources that are familiar with the technology being used and that are willing to help if any major issues are being faced. New information learned can also be compiled on GitLab as a quick reference for others that may be unfamiliar with the same technology.

Title: Sensor Degradation
Information: Sensors do not stay calibrated forever and so over time they will become less accurate. Sensors may also start to degrade or break down. Both of these issues will lead to inaccurate reading of data and thus inability to automatically reorder products with precision.
Risk Response Strategy: In order to consistently track data regarding the inventory of products, sensors will need to be calibrated on a regular basis, and there will need to be catches to track whether the sensor itself is degrading or is broken.

Title: Routing Inaccuracy
Information: Routing of vehicles for delivery will not always be accurate and so error with routing will need to be accounted for. For instance, the algorithm written for routing may be very inaccurate and cause much longer travel time, and thus more cost for the client. Also, if an accident or bad weather needs to be avoided, then inaccuracy in the routing may instead lead the delivery vehicle into a terrible backup of traffic that may cause them to be unable to make it to all of the delivery locations that they were supposed to reach that day.
Risk Response Strategy: In order to address inaccuracy in routing there will need to be a mechanism that keeps track of actual versus expected delivery data. This mechanism should also account for the time gaps along specific legs of the routes with respect to expected versus actual data. If there is evidence of major time gaps, then developers will need to be notified of the issue.

2.10 Project Proposed Milestones and Evaluation Criteria

- Proof-of-Concept
    - The web application component should be set-up and ready for user to log in and view data being collected by the storage monitoring device. The storage monitoring device should be connected and communication with the web application's database and should be sending information based on information the device's weight sensors are picking up.

- Minimum Viable Product
  - The monitoring devices should be properly calibrated to measure what is being stored. The devices have also been constructed into a "Smart-Bin" where items are placed. Multiple Smart-Bins are able to communicate with single device in a *Master-slave* architecture. The web application component should be able to monitor storage levels of multiple bins in multiple bin in one office pantry and be able to produce a shipping orders and an optimized route based on those storage levels. The web application component should also have a simple setup that users can use to register smart-bin devices and input what product the smart-bin is monitoring.
- Beta-Testing
  - The device and application will be constantly tested with real products over a period to time to find bugs, issues, and areas to improve. People will physically register devices, store products, and use our web application as if they were our client. This will produce feedback and bring awareness to issues that were not previously found.
- Integration
  - The web application is modified so it is ready with the client's existing software infrastructure.
- Finalized Product
  - The finalized product will have all bugs fixed while the web application's monitoring and routing system is optimized. The entire project will be set up to handle multiple smart-bins from multiple office pantries. The application is feeding data into Cloud Services to gain more information and learn trends in the data being collected to increase efficiency in future routes and orders.

2.11 Project Tracking Procedures

We will be posting a weekly status report every Monday to track our progress made during the week. We will also be documenting our meeting notes after every meeting to record our thoughts and concerns that don't make it onto the weekly report. Having separate meeting notes will also serve as a way to validate what is discussed and decided during those meetings.

GitLab will also be heavily used to keep track of weekly progress. This will be an easily accessible way to know where we are on more specific tasks. It will also be a way to look back on more specific tasks that are generalized on the weekly report.

2.12 Expected Results and Validation

The end goal is to have a fully functional, autonomous inventory management system. Crafty should be able to track their inventory at their clients' sites while doing little to no extra work. The product should also meet all requirements discussed in Section 2.2.

Our product will take a sensor, whether it be a weight sensor or barcode scanner, to detect the amount of inventory that Crafty currently has at its client's site. The sensor will then send a signal to our software through a Raspberry Pi. Our software will then collect the data and let Crafty know what items need to be shipped out.

Once all of the orders have been placed in the database, we will then configure what the most efficient routes are for Crafty's delivery service. This route optimization network will give Crafty's drivers the most efficient route by taking into account the location of each site and number of products needed at each site.

2.13 Test Plan

Our test plan will cover both the functional and non-functional requirements. We will validate our product through the test plan below.

2.13.1 Functional Requirement Tests

FR.1: The system can track the levels of various product in a customer's pantry stockroom automatically. This inventory should be sent to Crafty to determine whether a product needs to be restocked.

Test Case: For this requirement, we want to test if the sensor network can accurately detect changes in the inventory of the stockroom and update the database accordingly.

Test Steps:

1. Use the sensor arrays to detect a manually counted product.
2. Send data to the database.
3. Ensure that the saved data accurately represents and matches the manually counted data.

Acceptance Criteria: A database table of accurately counted inventory data.

FR.2: The user should be able to register/unregister sensor arrays to fit their unique stockroom.

Test Case: For this requirement, we want to test if we can add and remove sensor arrays to the sensor network to allow for a modifiable and scalable system.

Test Steps:

1. Add a sensor to the network via the web application.
2. Register it to detect a specific product.
3. Ensure that it is calibrated to the specific product metrics.

Acceptance Criteria: The sensor array is able to monitor the inventory of the newly registered product.

FR.3: The system can calculate an optimal route for deliveries for restocking client pantry stockrooms daily.

Test Case: For this requirement, we want to test the optimal routing algorithm for daily restocking.

Test Steps:

1. Provide the algorithm with test data that will generate an expected route.
2. Repeat for a variety of cases, including edge cases, to ensure maximum coverage of the algorithm.

Acceptance Criteria: The routing algorithm should calculate the most optimal route for restocking multiple client stockrooms.

FR.4: The system can display the stockroom inventory data and filter it accordingly to refine searches.

Test Case: For this requirement, we want to access and display information from the database as it updated.

Test Steps:

1. Search the database for a certain product related to a client's pantry
2. Search for products below threshold

Acceptance Criteria: The web application frontend should display available products in pantry and get more specific results based on filtering down the search.

2.13.2 Non-Functional Tests

The project will be tested for satisfaction of the following non-functional requirements: scalability, data integrity, availability, deployment, usability, resilience. Procedures and success and failure criteria will be provided.

Scalability

Procedure

1. Begin with *k* sensory devices connected to the sensor network
2. Set-up a new sensor devices to connect to the sensor network
3. After completion, open front-end and check if *k+1* sensory devices are connected

Success Criteria: *k+1* sensory devices are identified on the sensor network
Failure Criteria: < *k+1* sensory devices are identified on the sensor network

Data Integrity

Procedure

1. Begin monitoring sensor data from Raspberry Pi. Save sensor data in a *.csv* file.
2. Transmit sensor data to database. Store data into *.csv* on back-end.
3. At the end of the observation period, compare the two *.csv* files to determine equality

Success Criteria: The two files are equivalent
Failure Criteria: The two files are not equivalent

Availability

Procedure

1. Schedule using the *node-scheduler* library an inventory process to happen at the end of every weekday
2. At the start of a new day, check if a new order and delivery route has been generated

Success Criteria: A new order and delivery route has been generated
Failure Criteria: A new order and delivery route has not been generated

Deployment

Procedure

1. Set-up sensory devices in desired location
2. Run Raspberry Pi sensory device discovery protocol
3. Check if three-way handshake is successful for each new sensory device

Success Criteria: Each new sensory device is discovered by the master Raspberry Pi
Failure Criteria: A sensory device is not discovered by the master Raspberry Pi

Usability

Procedure

1. Track the time it takes for one individual to manually count the inventory
2. Based on the inventory counts, determine an order for the day, end timing of process
3. Track the time it takes for the sensor network to log inventory and produce order
4. Compare manual and automatic times

Success Criteria: Automatic time is less than manual time
Failure Criteria: Manual time is less than automatic time

Resilience

Procedure

1. Once per day, add or remove product from the sensory device
2. Repeat process for a month
3. After a month of moving product, check to make sure the sensory device is not damaged

Success Criteria: The sensory device is not damaged
Failure Criteria: The sensory device is damaged

3 Project Timeline, Estimated Resources, and Challenges

This section will discuss our time table and estimated cost for our project in detail.

3.1 Project Timeline

This project will entail a continuous cycle of brainstorming, research, prototyping, testing, refining, and demonstrating. Figure 7 details the team's design process, inspired by the Agile development process.

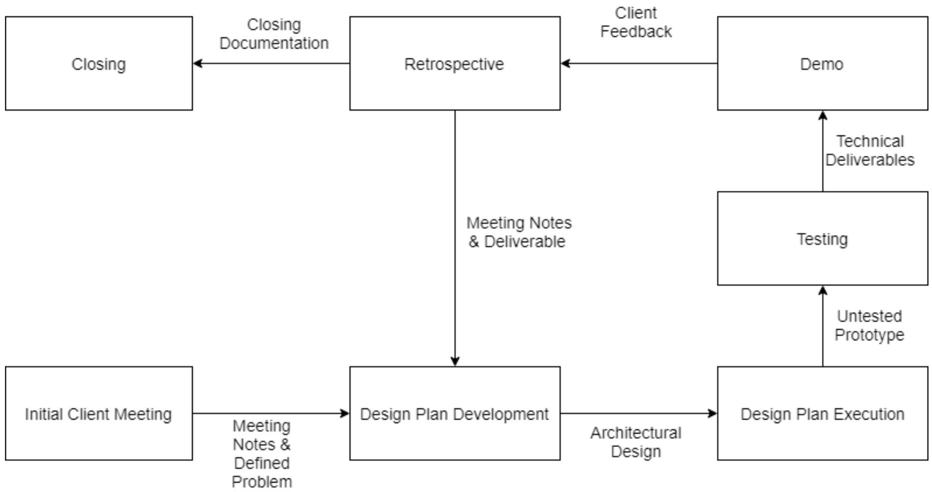

Figure 7: Process Diagram

The process will entail a series of sprints where at the end of each sprint, a prototype or technical deliverable is expected to be completed. The process will begin with an initial meeting with the client to define the project requirements, after which the first sprint will begin. Each sprint will follow a similar process, beginning with creating a design plan for the projected deliverable. Once the design architecture is determined, the plan will be executed and then tested to ensure that it satisfies all of the necessary requirements. Once the deliverable is complete, it will be demonstrated to the client to gather feedback on its progress, which will then guide and focus development. A retrospective meeting will be held by the team to reflect on previous work and to encourage continuous improvement upon the project architecture and design process. Once the project is complete, the team will compile the documentation to ensure that no technical knowledge regarding design decisions is lost.

The project will be broken into 4 different phases, where each phase will operate as an Agile sprint. Timelines for each phase of the project are listed in figures 8a-d.

	Task Title	9/3/18	9/10/18	9/17/18	9/24/18	10/1/18	10/8/18	10/15/18	10/22/18
Prototype 1	Determine Team Values and Norms	■							
	Gather Requirements		■	■					
	Design Initial Solution Architecture		■	■	■				
	Implement Initial Solution Architecture			■	■	■	■		
	Complete Prototype					■	■	■	■
	Testing							■	■
	Documentation							■	■

Figure 8a: Gantt Chart for Prototype 1 Phase

	Task Title	10/8/18	10/15/18	10/22/18	10/29/18	11/5/18	11/12/18	11/19/18	11/26/18	12/3/18
Prototype 2	Provide Database Information	■								
	Register Device		■							
	Implement Sensors		■	■	■					
	Register Items			■	■					
	Database-Model Entity			■	■	■	■	■		
	Create Order					■	■			
	Testing								■	■
	Documentation								■	■

Figure 8b: Gantt Chart for Prototype 2 Phase

	Task Title	1/14/19	1/21/19	1/28/19	2/4/19	2/11/19	2/18/19	2/25/19	3/4/19	3/11/19
Prototype 3 (Minimal Viable Product)	Handle Multiple Sensors (Master-Slave)	■	■	■	■	■				
	Handle Multiple Orders	■	■	■						
	Database Refinement		■	■						
	Route Optimization				■	■	■	■	■	■
	Testing							■	■	■
	Documentation								■	■

Figure 8c: Gantt Chart for Prototype 3 (Minimal Viable Product) Phase

	Task Title	3/18/19	3/25/19	4/1/19	4/8/19	4/15/19	4/22/19	4/29/19
Final Product	Cloud Integration	■	■	■	■	■		
	Predictive Analytics Component		■	■	■	■		
	Final Testing and Troubleshooting			■	■	■		
	Final Documentation					■	■	■

Figure 8d: Gantt Chart for Final Product Phase

3.2 Feasibility Assessment

This project will be more an affirmation of concept than the construction of a finished product. That is, we will not be assembling and operating a full pantry of office kitchen food but assessing the viability of the use of certain sensors and software systems to adequately and scalably achieve the client's goal. We anticipate that prototyping and testing the system outside its intended context will impede the accuracy of our assessments.

3.3 Personnel Effort Requirements

The project will entail a balance of component testing, software development, and system integration tasks. Each task is listed in table 1.

Task	Description	Time (person-hours)
Implement Analytics Software Back-End	Create component to monitor database information	100
Implement Analytics Software Front-End	Create User Interface for users to view information on client product usage	40
Implement User Setup Interface	Create online form to register physical monitoring device and note what product the device is monitoring	80
Design Sensor Apparatus	Plan and construct the physical sensor interface to output data that can be reliably interpreted as a unit quantity	100
Implement Embedded Programming	Program the microcontroller to interpret incoming sensor data as unit quantity and package that information	100
Implement Route Optimization	Integrate Analytics component to produce shipment orders and routes for Warehouse Employees and truck drivers	100

Table 1. Personnel Effort Table

3.4 Other Resource Requirements

MySQL Database
Linux Redhat 7 Server for Hosting

3.5 Financial Requirements

While the bulk of this project occurs in a software space, costs will be incurred in prototyping the sensing and data collection systems.

Item	Description	Cost ($)
Load Cell	Weight-sensing device	10
Load Cell Amplifier	Load cells output a very small signal that can be hard for an ADC to read	10
Barcode Scanner	Reads photographic product IDs to track inbound merchandise	20.99
ADC	Analog-to-digital converter to interface between analog load cell and digital microcontroller	10
Raspberry Pi 3 B+	In-house processor to gather sensor data into a single information package to be sent to the server	39.95
ESP8266	Used for connecting sensor arrays to Raspberry Pi wirelessly	6.95
Ultrasonic Sonar Distance Sensor	Detects distance from sensor, to be used for measuring height of stacked boxes	3.95
Other	Further purchases to complete the system, such as trays and barcodes, will complete the interface between the sensor and the merchandise	50

Table 2. Financial Costs

4 Closure Materials

The conclusion section will reflect upon the current status of the solution and future goals for the team.

4.1 Conclusion

Crafty, LLC's current infrastructure is based on a middleman, creating an unnecessary expense that is prone to human error. This causes warehouse truck routing to become severely inefficient and expensive from restocking at individual offices based on separate orders.
Our solution consists of a microcontroller device that automatically monitors the amount of items in an office pantry and places orders to the Crafty warehouse when the office has reached its minimum threshold value for certain pantry items. We will then develop software that processes the current orders from all office orders to optimize shipment routes.

At this point in the project, our team has implemented the basis for the major components of our project: a Raspberry Pi sensor network and an analytics software application. We have been able to create a prototype so far that can take a weight measurement from the load cell and send it to through the Raspberry Pi to transmit it out to the database. The analytics software is able to interpret the data and determine whether or not an item has to be reordered. It is also capable of displaying the inventory data on a ReactJS front-end through a web browser.

We plan to continue development based on the content of this document. We believe that the design presented will be able to assist Crafty and its clients to a more effective and efficient inventory management system. There will be an emphasis moving forward on testing the implementations to ensure the accuracy of the data and resulting outputs from processing that data. The end product will be capable of tracking the inventory of a pantry and using the inventory data from multiple pantries to determine an efficient route to resupply pantries with products that are at a low quantity.

4.2 References

Intellectual Reference

1. Impinj.com. (2018). Automated Inventory Management with RAIN RFID | Impinj. [online] Available at: https://www.impinj.com/solutions/healthcare/inventory-management/.
2. Route4me.com. (2018). Dynamic Route Optimization™ On-Demand Deliveries & Pickups. [online] Available at: https://www.route4me.com/platform/routing.
3. Barcodesinc.com. (2018). Inventory Control System - Automate Your Tracking. [online] Available at: https://www.barcodesinc.com/articles/inventory-control-system.htm.
4. Wind.cs.purdue.edu. (2018). [online] Available at: http://wind.cs.purdue.edu/doc/adhoc.html.

Pricing Reference Websites:

1. Electronics, S. (2018). SEN-13329 SparkFun Electronics | Sensors, Transducers | DigiKey. [online] Digikey.com. Available at: https://www.digikey.com/product-detail/en/SEN-13329/1568-1852-ND/7393715/?itemSeq=272691527.
2. Electronics, S. (2018). SEN-13879 SparkFun Electronics | Sensors, Transducers | DigiKey. [online] Digikey.com. Available at: https://www.digikey.com/product-detail/en/SEN-13879/1568-1436-ND/6202732/?itemSeq=272691544.
3. Industries, A. (2018). ADS1015 12-Bit ADC - 4 Channel with Programmable Gain Amplifier. [online] Adafruit.com. Available at: https://www.adafruit.com/product/1083.
4. Industries, A. (2018) HC-SR04 Ultrasonic Sonar Distance Sensor. [online] Adafruit.com. Available at: https://www.adafruit.com/product/3942.
5. Sparkfun.com. (2018). Raspberry Pi 3 B+ - DEV-14643 - SparkFun Electronics. [online] Available at: https://www.sparkfun.com/products/14643.
6. Sparkfun.com. (2018). WiFi Module - ESP8266 - SparkFun Electronics. [online] Available at: https://www.sparkfun.com/products/13678.

4.3 Appendices

Raspberry Pi Model 3b Datasheet

1. Raspberrypi.org. (2018). [online] Available at: https://www.raspberrypi.org/documentation/hardware/computemodule/datasheets/rpi_DATA_CM_2p0.pdf.