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: September 28, 2018 / Version 1.0

Table of Contents

List of Figures

List of Tables

List of Definitions

AWS - Amazon Web Services
ETG - Electronics and Technology Group
MVC - Model-View-Controller
MVP - Minimum Viable Product

1 Introductory Material

1.1 Acknowledgment

The Inventory Automation Team would like to thank the Iowa State University College of Electrical and Computer Engineering for providing the student team a professional experience, resources, and consultation with experts. The team appreciates the University's Electronics and Technology Group's (ETG) availability for providing hardware and server components for the project. Special thanks also go to Iowa State's Goce Trajcevski for weekly consultation when dealing with technical issues and moving forward throughout the development progress. Appreciation also goes towards our client from Crafty Inc. 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 Statement

Crafty Inc. is a warehouse company that delivers food to company pantries. There is currently infrastructure is based on a single crafty employee at each of their client offices to handle shipment orders and physically monitor when the company pantry is low on items. This is an unnecessary expense with human error. It also makes the routing orders of truck severely inefficient and expensive since the company sends trucks out to any office individually to restock on items when an office employee submits an order request.

Our solution is the development a microcontroller device that can automatically monitor the amount of items in an office pantry and then place orders to Crafty when the office is running low on an item. We will then develop software that processes the current orders from all office orders to create optimize shipment routes. This will save a significant amount of money and time for orders.

1.3 Operating Environment

The physical microcontroller device to monitor office shipment levels is expected to be stored in a dry pantry area with a WiFi connection. The current device is a microcontroller-connected "Smart-Bin" that is expected to fit on shelves. Any setup with the device aside from turning the device on will be done through an web application component. The web application component will be used by Crafty employees at office locations for just setting up the physical device along with Crafty employees in the Warehouse who see and handle shipment orders.

1.4 Intended Users and Intended Uses

There are three main users of this solution. Each user will do most interaction through the web application component.

1. *Pantry Employee*: Actor based in a company office with minimal technical experience 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 with moderate technical experience. Warehouse employees are in charge of overseeing shipment orders and filling trucks with the proper items. This user will go into the application to view orders the routes.

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

1.5 Assumptions and Limitations

Assumptions:

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

Limitations:

- Accuracy of automatic orders is completely based on the accuracy of the device to measure weight
- Solution monitors product availability in in bulk, not individually
- Solution can only make correct order if the Crafty user properly establishes what products are being monitored

1.6 Expected End Product and Other Deliverables

The final product delivery will be split into a proof-of-concept, minimum viable product, and a final product deliverable. 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 products 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 be 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

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

2.3.1 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 or not a product needs to be restocked.

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

2.3.2 Analytics Software

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

2.3 Constraints Considerations

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 employ a level of redundancy to audit the values of the other sensor. It will also be accomplished through testing of the sensors 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 amount 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 easily be able to select which sensor is monitoring a specific product.

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 our project, Impinj clients have a stock managed by them. Their solution incorporates RFID tags to track what is moving in and out of the room.

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. These routing algorithms were mainly built for travelling from one destination and possibly entering stopping points along the trip. 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, and use information known about weather and traffic to create one master route that is the most efficient with regards to time and gas used. Our client believes that the existing tools for finding efficient travelling routes do not apply directly to a trip with a lot of separate sub-routes such as would be seen when delivering products to several clients in a day.

2.5 Proposed Design

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 in order to employ redundancy to minimize error. The sensors being considered are as follows: weight sensors and barcode scanners.

Weight sensors will be placed in specific locations of the clients' storage locations to monitor the weight of product left 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.

A barcode scanner will be incorporated into our design as well. This will serve a purpose of redundancy. It is unclear at this point of what specific role the barcode scanner will serve. It may be used to scan items as they enter and exit the room, or used at closing time to account for all items.

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 to 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. This was chosen over alternatives such as MongoDB because the data necessary to be stored is found to well-defined, so data manipulation and querying will be optimal for a relational database architecture.

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 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 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. This will allow the sensor network to become flexible and scalable for a variety of stockrooms. Another benefit is the on-board Linux OS, which helps the modifiability of the product. Any changes to the hardware orientation can easily for fixed or changed in the code, which can be access remotely if need be. This also allows for remote deployments of software updates. Arduinos were ruled out early on due to their lack of some crucial features that assist in the realization of the architecture.

There are two sensors being considered for the sensor network: barcode scanners and weight sensors. Although RFID tags were originally considered, they were replaced with a barcode scanner to reduce on the unnecessary overhead 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 the provide the readings with redundancy to minimize the accumulation of error and ensure the inventory readings remain accurate.

The analytics software will be made using an MVC pattern. This is to allow for an easily understood environment that the team already has previous experience with, but also to also 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 through AWS.

The database will be implemented with a MySQL relational database. A relational database was selected for its ability to easily manage data, and provide the user with significant control to do analytical processing on the data later.

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 side 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 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 would have to do with 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 have any chance of being shocked.

2.8 Task Approach

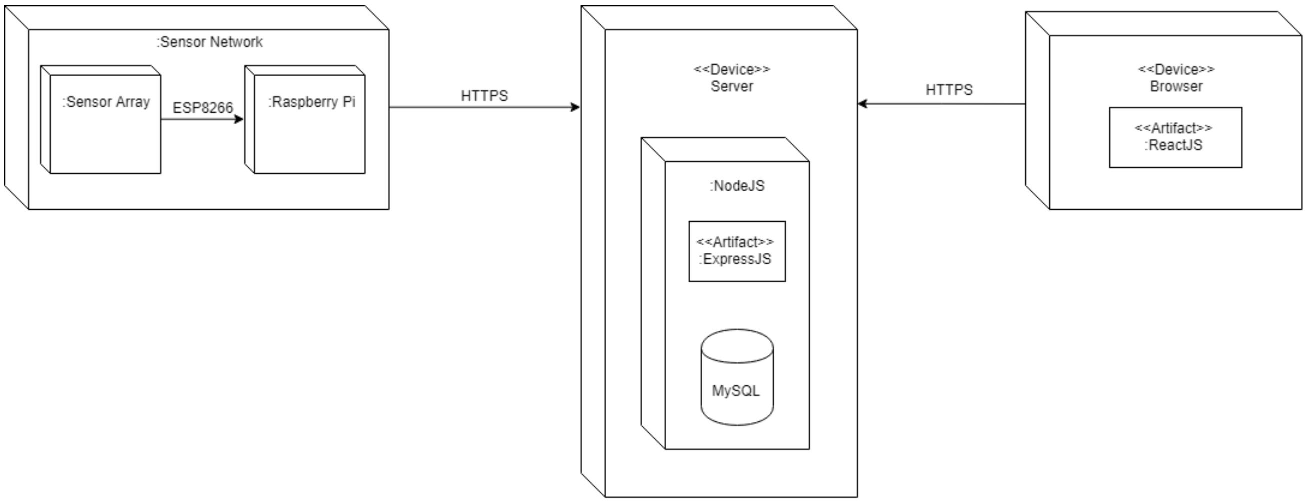

Figure 1. Project UML Deployment Diagram

The project will be broken into two major components: the sensor network and the analytics software. The sensor network will consist of a Raspberry Pi communicated with a variety of sensors with ESP8266 chips. It will communicate via HTTPS to the back-end, which will host the MySQL database. It will also host the front-end, which will be interacting with via a web browser.

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 are unfamiliar with
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 also. 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 is trying 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 delivery data versus expected delivery data along with what the time gaps along specific legs of the routes were with respect to expected versus actual data. If there are major 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 base 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.
- Integration
  - The web application is modified so it is ready ready with the client's existing software infrastructure
- Finalize Product
  - The finalize product will have all bugs fixed while the web application's monitoring and routing system is optimized. The entire project 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 keep track of their inventory at their clients' sight 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 their clients sight. The sensor will then send a signal to our software through raspberry pi. Our software will then collect the data and let Crafty know what items need to be sent 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 to take. This route optimization network will give Crafty's drivers the most efficient by taking into account the location of each sight and amount of products needed at each sight.

We will validate our product through the test plan below

2.13 Test Plan

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 or not 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.

Expected Results: 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 sensors 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 be detecting a specific product.
3. Ensure that it is calibrated to the specific product metrics.

Expected Results: The sensor array is able to monitor the inventory of the 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

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

3 Project Timeline, Estimated Resources, and Challenges

3.1 Project Timeline

This project will entail a continuous cycle of brainstorming, research, prototyping, testing, refining, and demonstrating. We plan to have the first prototype finished by the end of week 7, after which new prototypes will be developed in 2-to 3-week intervals. With each successive prototype, we will make steps from experimenting with possibilities to refining a finished product, so that the final prototypes assembled in the spring will be a fully-functional system that meets all requirements and constraints.

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 operation 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.

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	20
Implement User Setup Interface	Create online form to register physical monitoring device and note	80

	what product the device is monitoring	
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

SQL Database
Linux Server for Hosting
AWS Cloud Service Connection

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
ADC	Analog-to-digital converter to interface between analog load cell and digital microcontroller	10
Microcontroller	In-house processor to gather sensor data into a single information package to be sent to the server	100
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

4.1 Conclusion

At this point in our project, our team has completed gathering requirements by learning the needs and expectations of our client through various team meetings and communicating with Crafty. Furthermore, we have begun and have been making great progress working as a team to design a solution to our given problem and are in the beginning stages of prototyping our solution. We hope that once all of our required materials are acquired that we will be able to demonstrate our solution to our client as quickly as possible in order to begin improving on our solution. Our team has split into two groups to work on software and hardware for this project. The software group will focus on creating and organizing the database and placing online orders from the Crafty warehouse. The hardware team will work with sensors and microcontrollers to create a sensor network to gather all inventory data.

4.2 References

Intellectual Reference

1. https://www.impinj.com/solutions/healthcare/inventory-management/
2. http://wind.cs.purdue.edu/doc/adhoc.html

Pricing Reference Websites:

1. https://www.digikey.com/product-detail/en/SEN-13329/1568-1852-ND/7393715/?itemSeq=272691527
2. https://www.digikey.com/product-detail/en/SEN-13879/1568-1436-ND/6202732/?itemSeq=272691544
3. https://www.adafruit.com/product/1083

4.3 Appendices

Raspberry Pi Model 3b Datasheet

https://www.raspberrypi.org/documentation/hardware/computemodule/datasheets/rpi_DATA_CM_2p0.pdf

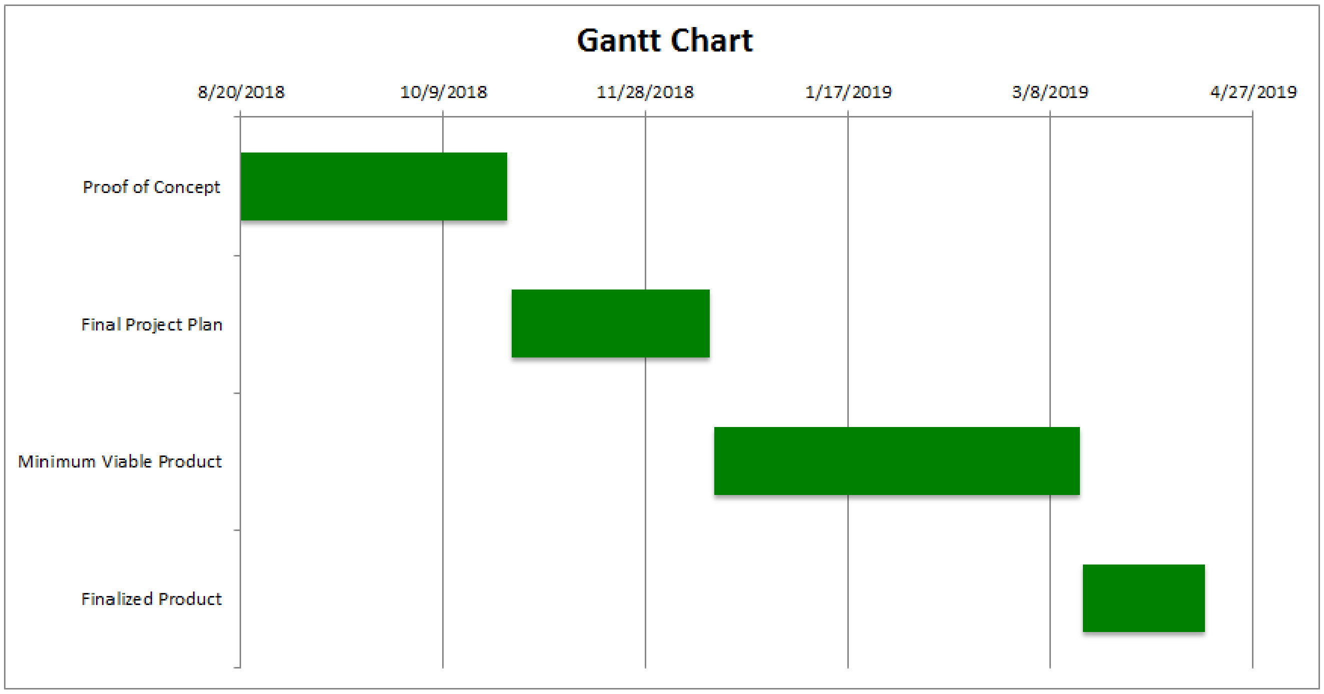

Figure 2. Gantt Chart