RFID Luggage Tracking

Michigan State University

ECE 480 Design Team 1

Fall 2015

Project Sponsor:
Dr. Satish Udpa

Project Facilitator:
Dr. Tongtong Li

Design Team 1:
Marwan Baraya
Henry Nguyen
Brian Prange
Emmanuel Wadieh
Ziye Xing
Acknowledgements:

Dr. Satish Udpa: The sponsor for design team 1 this year, Dr. Udpa provided the team with a unique project allowed the team to learn more about an emerging field of technology. The experience with RFID technology can be applied to so many fields and will be invaluable to the team's professional future after MSU. Not only did Dr. Udpa push the in regards to the senior design course, but he has also pushed the idea of developing the project even further and starting a business venture. We truly appreciate the support and opportunities we received from Dr. Udpa and hope he never loses his luggage again.

Dr. Tongtong Li: As the faculty facilitator Dr. Li was a very helpful force in guiding the team with the various brainstorming, reports and projects throughout the course. Her positive outlook made meetings enjoyable and memorable. While her knowledge of the engineering process provided a quality outside perspective on various issues that arose in the group.

Dr. Timothy Grotjohn and Dr. Lalita Udpa: We would like to thank Dr. Grotjohn and Dr. Udpa for the knowledge they were able to impart on us throughout the semester. This information will be utilized and carried with us for the rest of our professional careers.

Jonathan Trites: A big thank you from the team to Jonathan Trites for his help in the production of our Design Day presentation video. Jonathan helped us capture and visualize the success of the team and the full features of the final product in a concise, professional video.
Executive Summary

Design Team 1 was tasked with improving airport luggage tracking and retrieval through the use of Radio Frequency Identification (RFID) technologies. The team was given fourteen weeks to create a product that could easily integrate RFID technology into the airline luggage systems in addition to the creation of a mobile application. The current luggage system, involving barcodes, is prone to failure as evidenced by the 15% failed scan percentage. By replacing the barcode system with an RFID tagging and tracking procedure, successful identifying and routing of luggage would increase to a rate near 100%. The project was centered around the ability to differentiate RFID tags attached to luggage with a mounted scanner. After being scanned, the information can be used by the airport for routing purposes and by the passenger through the application that gives real time tracking updates.
# Table of Contents

## Chapter 1: Introduction & Background
1.1 Problem Description: 4  
1.2 Project Description: 4

## Chapter 2: Finding a Solution
2.1 Design Parameters 5  
2.2 Proposed Solutions 5  
2.3 Final Solution 7  
2.4 Final Solutions: Hardware 8  
2.5 FAST Diagram 10  
2.6 GANTT Chart 10  
2.7 Budget 11

## Chapter 3: Technical Design
3.1 Hardware 12  
3.2 Hardware: Airport Scanner 12  
3.3 Hardware: Mobile Scanner 13  
3.4 Hardware: Problems 15  
3.5 Software and Interface Design Requirements 17  
3.6 Software Implementation 17  
3.7 Software Problems 22

## Chapter 4: Testing and Verification of Design
4.1 System Testing 23  
4.2 Testing Result Analysis 25

## Chapter 5: Summary and Conclusion
5.1 Final Cost 26  
5.2 Schedule 26  
5.3 Conclusion 26

## Appendix 1: Technical Roles
Appendix 1.1 - Brian Prange 28  
Appendix 1.2 - Ziye Xing 29  
Appendix 1.3 - Emmanuel Wadieh 30  
Appendix 1.4 - Marwan Baraya 31  
Appendix 1.5 - Henry Nguyen 32

## Appendix 2: References
Appendix 2.1 - Works Cited 34  
Appendix 2.2 - Datasheets 34

## Appendix 3: Technical Attachments
Appendix 3.1 - Software 34  
Appendix 3.2 - Parts List 34
Chapter 1: Introduction & Background

1.1 Problem Description

The current luggage tracking system depends entirely on barcodes and automatic 360 degree barcode readers mounted above conveyor belts. The issue with the current system centers around the readability of the barcode tags. Wrinkled tags are difficult to read and dirty barcode readers cannot consistently identify the labels, which leads to mishandled luggage. According to the United States Department of Transportation, the total number of mishandled bags in 2014 was 2.1 million, which equates to nearly $7 billion in reimbursement fees for the airline companies. (Bureau of Transportation) At the request of the sponsor, Design Team 1 was tasked with developing a more efficient tracking method along with an iOS mobile application to aid in the luggage return process.

1.2 Project Description

While considering possible solutions, Dr. Udpa requested that the team incorporate RFID technology in its proposed designs. The RFID tags would replace the barcode tags used in the current system. The tags are scanned by passing within range of an antenna attached to an RFID reader. Since RFID technology uses radio waves, the scanner can permeate through the luggage and read the tag regardless of its orientation. This improved scanning ability would reduce the scanning failure rate to nearly zero. Additionally, by developing a mobile application, the passengers would be able to verify which bag is theirs at baggage claim. The successful completion of this improved luggage tracking system would save airline companies billions of dollars suffered from mishandled luggage compensation while also increasing ease of travel and customer satisfaction.
Chapter 2: Finding a Solution

2.1 Design Parameters

The team decided upon four key criteria to rate the feasibility of each team member's unique design approach. Each criteria is weighted according to its overall importance to the final design and displayed in Table 2.1.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration</td>
<td>Due to the low cost of RFID tags, they can be used with the existing regular barcode tag. The data on the RFID will be the same as the data that can be read from barcode tag. This will allow us to use existing handling systems without having to create a separate baggage handling system. This smooth transition will convince more airports and airlines to adapt our technology.</td>
<td>30</td>
</tr>
<tr>
<td>Accessibility</td>
<td>The ease of use for the customer when considering the mobile app. The app should aid in the process of tracking the customer's luggage without creating any further hassles in the departure and return process.</td>
<td>30</td>
</tr>
<tr>
<td>Cost</td>
<td>Due to budget constraints, the overall cost of hardware/software and labor must be taken into account.</td>
<td>30</td>
</tr>
<tr>
<td>Time</td>
<td>How long it will take for the hardware/software to be delivered. Also, an estimated time for completion of the proposed solution.</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 2.1: Design Parameters

2.2 Proposed Solutions

Each team member made a basic conceptual design that they believed could improve the airline tracking system. A breakdown of each conceptual design and the overall rating of each design based on the previously mentioned design parameters can be seen in Table 2.2 on the next page.
<table>
<thead>
<tr>
<th>Concept Designer</th>
<th>Design Summary</th>
<th>Integration</th>
<th>Accessibility</th>
<th>Cost</th>
<th>Time</th>
<th>Total</th>
</tr>
</thead>
</table>
| Brian Prange     | Low Frequency Readers  
- Cheap  
- Must be fairly close to scan  
- lower frequency can penetrates thicker steel frame luggage  
Passive RFID  
- Can be picked up by LF readers | 3 | 1 | 5 | 4 | 13 |
| Marwan Baraya    | Passive RFID  
- Cheap  
UHF Reader  
- Small, mountable  
- Radial scanning  
- Can scan multiple tags at once  
Mobile updating  
- Instead of passenger scanning they will receive updates on their luggage from the airport database | 4 | 5 | 4 | 3 | 17 |
| Ziye Xing        | Active RFID  
- More costly  
- Larger radius and reliable than passive tags  
- Can be reused  
Mounted radial RFID scanners  
- Similar to 360 barcode scanners  
- not affected by dust and small debris | 4 | 4 | 2 | 2 | 11 |
| Henry Nguyen     | Twin pillar checkpoint towers  
- Similar to those in retail stores  
- Picks up on radio frequencies for RFID tags as they are passed through Cellphone scanning app  
- Utilize the wireless capabilities of the iPhone and iOS app developer to allow customers to scan for their bags with their phones | 4 | 2 | 2 | 4 | 12 |
| Emmanuel Wadieh  | Permanent RFID  
- Embedded in lining of luggage Reading Arch  
- Reader would be built in arch shape  
- Anything passing through the arch is scanned  
- Costly | 4 | 4 | 1 | 2 | 11 |

Table 2.2: Conceptual Design Matrix

*Scale is from 1-5, with 5 being very good and 1 being very poor*
2.3 Final Solution

Marwan’s design utilizing the Ultra-high frequency (UHF) tags along with a mobile app that reflects when a passenger’s luggage passes a reader was chosen as the final design solution. As seen in the decision matrix in Table 2.2, the design rated highly in accessibility, cost, and integrability into airlines. The International Air Transport Association (IATA) has different permitted radio frequencies per country, except in the 860-960 MHz range. The IATA regulation led to the team's decision to use UHF tags which operate in this range. The UHF tag was also ideal due to its ability to permeate non-metal materials. The tag would be placed into a traveler’s luggage and passed under an arch with an RFID scanner mounted at the top. Also, a portable version of the mounted scanner in conjunction with a mobile app would be developed to aid passengers during luggage return.

In the current luggage tracking system, barcodes are printed on a tag which is then affixed to a bag. The luggage is placed onto the conveyor belt so that the barcode is facing upward and then the conveyor belt transports the bag under a barcode-scanning arch. There are two major problems with the current system: damaged/misoriented paper tags and dirty scanners. If the tag is not facing one of the scanners, or is dirty or damaged, the scanner will not be able to scan the bag. Also, the scanners require regular maintenance to ensure that they are free of dust and dirt, otherwise the scanner is blocked from scanning tags. The final solution chosen by the team addressed the problem of unreadable/damaged tags since the RFID tags can be read in any orientation. Scanner maintenance would be minimal for the RFID scanner because it relies on permeating radio waves that can bypass debris that would build up. A visual representation of the overall design can be seen in Figure 2.3.
2.4 Final Solution: Hardware

There are two major components needed in RFID tracking: The reader/writer and the antenna. Both the mounted and handheld systems used the AS3992 UHF RFID reader/writer by Austria Microsystems which can be seen in Figure 2.4.1 below.
The handheld system utilized a much smaller antenna than the mounted system resulting in the reading range being shorter than the mounted system. The handheld and mounted antennae can be seen below in Figures 2.4.2 and 2.4.3, respectively.

![Figure 2.4.2: Mounted UHF RFID Antenna](image)

![Figure 2.4.3: Mobile UHF RFID Antenna](image)

The handheld device required a way to wirelessly communicate with the mobile application. The team accomplished this connectivity by using a Blended Micro from RedBearLabs. The Blended Micro consists of an Arduino board integrated with the Nordic Bluetooth smart software development kit. The Blended Micro can be seen in Figure 2.4.4 on the next page.
2.5 *FAST Diagram*

The Function Analysis System Technique Diagram (FAST) shown in Figure 2.5 below displays the overall goals of the project. When reading the diagram from left to right the overall goal is broken down into three major subcategories: airport integration, the mobile app, and RFID integrated tags.

![Figure 2.5: FAST Diagram](image)

2.6 *GANTT Chart*

The GANTT Chart was a scheduling tool introduced in week four of the senior design course. The chart allowed the team to breakdown the entirety of the project in eight phases: project definition, research, design, implementation, testing, production, presentation and
submission. The chart can be seen in Figure 2.6 below. The GANTT Chart provided a clear breakdown of what needed to be accomplished and the dates of completion.

![GANTT Chart](image)

**Figure 2.6: GANTT Chart**

2.7 Budget

After some initial research and with the final design in mind, the team broke down what they would need to fabricate a working prototype. The budget breakdown can be seen in Table 2.7 on the next page.
Chapter 3: Technical Design

3.1 Hardware

The team’s project hardware consists of two major components: airport and mobile. The airport side is responsible for initiating the database and associating a customer ID and itinerary to a unique RFID tag. The mobile side consists of the handheld scanner connected to the traveler's mobile device which will check and confirm the RFID tag attached to his or her baggage.

3.2 Hardware: Airport Scanner

The airport-mounted RFID scanner consists of a UHF RFID scanner that uses the AS3992 Austria MicroSystems model and an antenna that is capable of interrogating passive RFID tags for a distance of approximately five meters. The system operates on a frequency of 915 MHz which meets the industry standard for UHF tags. This frequency is permitted in airports within North America, Europe and Asia which will allow this solution to be implemented in airports worldwide. The scanner is connected to a computer via USB which allows the reader to update the database in the cloud. The antenna was mounted on a bar to simulate how it would be placed above a conveyor belt at the airport. The airport scanner setup can be seen in Figure 3.2 on the next page.

<table>
<thead>
<tr>
<th>Product</th>
<th>Quantity</th>
<th>Cost</th>
<th>Total Per Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFID Kit</td>
<td>1</td>
<td>$250.00</td>
<td>$250.00</td>
</tr>
<tr>
<td>Apple Developer Account</td>
<td>1</td>
<td>$99.00</td>
<td>$99.00</td>
</tr>
<tr>
<td>RFID Tags</td>
<td>5</td>
<td>$1.00</td>
<td>$5.00</td>
</tr>
<tr>
<td>3-D Printed Case</td>
<td>1</td>
<td>$25.00</td>
<td>$25.00</td>
</tr>
<tr>
<td>Luggage</td>
<td>1</td>
<td>$80.00</td>
<td>80.00</td>
</tr>
</tbody>
</table>

Total cost: $459.00

Table 3: Initial Budget Breakdown
3.3 **Hardware: Mobile Scanner**

The mobile scanner is capable of scanning the RFID tags at the UHF frequency and consists of the following parts: The RFID scanner, the antenna, microcontroller, and a battery. The RFID scanner is of a similar model to the scanner used in the airport side. The difference between the two is that the mobile reader is loaded with a firmware that enables UART protocol instead of USB to for data transmission. The reader is configured to operate at a Baud Rate of 9600 Bps so it can be compatible with the utilized microcontroller. Unlike the airport side, the mobile scanner uses a smaller antenna which is only capable of interrogating passive UHF RFID tags at ranges of approximately 20cm. A more expensive antenna could be used to extend the range to about 80 cm. The microcontroller used in this project is a Bluetooth-enabled Arduino by RedBear labs that is capable of operating the RFID scanner and receiving and transmitting data to the mobile phone via BLE (Bluetooth Low Energy) or Bluetooth V4.0. This approach allows the team to connect to iOS devices without requiring any hardware access permissions from Apple. This will also allow the team to integrate the same mobile scanner with other platforms such as Android and Windows phones. The microcontroller uses serial port protocol to
communicate with the RFID scanner via the UART port using two lines (Tx, Rx) and ground. The controller operates at a frequency of 8MHz and 3.3Volts. To power the mobile device we used a 2000mAh 3.7V Lithium ion rechargeable battery that is capable of powering the scanner for nearly four hours. The charger charges the battery at a rate of 500mA/hour, utilizing a USB charger from Sparkfun with a standard Mini-USB port. The connections and placement of the various components inside the case can be seen in Figure 3.3.1 below.

![Figure 3.3.1: Mobile Device Internal Components](image)

The team also attached a switch to allow the device to be turned on and off, along with four LEDs - three external and one internal - used to verify four different functions of the scanner. The LEDs verified the following:

- An internal red LED lights up when the device is charging and turns off when the battery is fully charged.
- An external red LED that indicates power (on when the device is turned on).
- An external green LED that indicates the device is connected to a mobile phone via Bluetooth.
- An external yellow LED that flashes every time the reader successfully interrogates an RFID tag.
The switch and LEDs can be seen in Figure 3.3.2 below.

![Figure 3.3.2: Mobile Scanning Device](image)

### 3.4 Hardware Problems

The biggest issue the team faced when building the handheld scanner was the compatibility of the parts used in the project. After days of debugging and testing we found out that the Arduino used was not compatible with the RFID scanner connected to it. Both use the same protocol (UART), but after further research it was discovered that the two devices operate at different Baud rates. The scanner transmits data at a rate of 115200bps but the Arduino is only capable of receiving 9600bps.

The quick solution to this problem would have been to purchase a different RFID scanner, but after further research, the team discovered that the firmware could be uploaded to the current reader and slow it down to match the Arduino. This was the cheapest solution and the best since other Arduinos are larger in size and do not have Bluetooth capabilities.

To be able to update the new firmware, the team had to use a USB programmer from Silabs, shown in Figure 3.4.1 on the next page, which led to another problem. The USB programmer and the RFID scanner had different interfaces, so the team had to build a
Figure 3.4.1: USB programmer from Silabs

converter circuit, as shown in Figure 3.4.2 below to make the USB programmer compatible with the RFID scanner.

Figure 3.4.2: Converter Circuit

After constructing the converter circuit, the team was able to update the firmware on the AS3992 RFID scanner and slow it down to communicate with the Arduino to extract data from the RFID scanner.
3.5 Software & Interface Design Requirements

The primary purpose of the RFID baggage tracking is to give customers greater awareness of where their baggage is and to provide a more positive travel experience. A user-friendly software application was developed to improve customer satisfaction. The project required the development of an iOS platform app that would be compatible with the RFID baggage tracking system. To achieve the goal, the app allows the user to:

- search the database by inputting the owner’s information
- save the searched baggage to a list to check status updates
- input unchecked baggage information for future updates
- manage the previous entries
- use accessories such as Bluetooth to connect peripheral scanning devices

In order to satisfy the requirements specified above, there are other auxiliary facilities to assist in the completion of the tasks. All the baggage statuses, along with passengers’ information are stored in a remote server, which would be maintained by an airline. For this project, the team also deployed a remote cloud server to simulate the workflow.

The software solution simulates the entire process of a real flight from check-in to baggage claim. A user’s baggage data is stored in the server’s database after check-in and every time the bag passes an RFID scanner check point, the data should be updated. From the client application, the user is able to pull the latest update from the server. Additionally, the app has the capability to connect a handheld portable scanner via Bluetooth. The app receives scanned tag data and compares the tag with the current information in the database. If the scanned tag’s data matches the tag id associated with the traveler, a sound is emitted from the phone.

3.6 Software Implementation

The software mobile client was developed for the iOS platform using the development environment Xcode 7.0. The application was primarily written in the Swift 2 language to implement the interface and the back-end logical processing. The application was also built on Objective-C to implement the Bluetooth Low Energy module. The software was constructed
under the Model-Controller-View (MCV) pattern which separates the software into the user-interface interactive part and the logical computing controller to implement the displayed image and the data model behind the scenes. The user-interface, called the View in MCV, consists of multiple views and the transition action, which together simulates how the screen would look on a cellular phone. Figure 3.6.1 shows the final design of project storyboard.

The storyboard shows the views contained in the software and how they link to each other. It also displays the interaction between the UI element and the controller. One can insert

**Figure 3.6.1: Application Storyboard**
elements from the storyboard into code by Ctrl-dragging it. The static element was represented as @IBOutlet and the interactive element to implement a function was represented as @IBAction.

Figure 3.6.2: Mobile User Interface

The final component to satisfy the design requirements is shown in Figure 3.6.2 above. The figure shows the interface that allows a user to input information and search the database. A user enters his or her name, airport code of origin and destination, and also the flight number. Then, the travel column will call a text picker for airport code in the range of predefined set of airports. The rest of text fields will call the keyboard instead. A picture could be added if the user preferred to assign an image to the entry to differentiate from others. The photo selection is the only optional entry, while the remaining fields are required in order to enable search actions. As shown above, the “Done” button at the top-right corner is disabled because of the blank text fields.

On the navigation bar at the top of the screen, a user can either choose to add a new entry or edit existing entries. The plus button will take a user to the “Add New” view as depicted above, while the “Edit” button is used to manage and delete old entries. Users can tap the cell in the table, which will trigger the web server and update any changed information. If the bag were previously unchecked, it would return a list of all possible results and exhibit a similar function to “Add New.” Otherwise, the app would guide the user to the “Detail” view. In the “Detail” view, the user can view the summary about the desired baggage information. The “STATUS”
label column will be updated to the latest information from the server. In the “Detail” view, the user can enable the Bluetooth service and search for any suitable devices. An example of an added baggage object can be seen in Figure 3.6.3 below.

![Table View of Archived Baggage](image)

**Figure 3.6.3: Table View of Archived Baggage**

The Bluetooth Low Energy technique was implemented in software using the CoreBluetooth module. The “Detail” view shown in Figure 3.6.4 displays the detailed information and current status of the baggage. A user can connect the handheld scanner by tapping the Connect button, assuming that the Bluetooth reader is in the 46’ range. Once the connection is established, the scanner begins reading tags and sending message to the mobile device. The data transmitted is stored in a buffer, until the buffer size is larger than 64 bytes or a single transmitted packet data is less than 20 bytes at which point the data in buffer will be encoded into a string and cleared. The resulting string is the unique tag id bound to each bag. The app will then compare the reading tag with the tag id stored with the bag. If the stored tag id matches the epc code of the luggage tag, the app will emit a sound to notify the user. This application can be seen in Figure 3.6.4 on the next page.
Another aspect of the project is the Graphical User Interface (GUI). The purpose of the GUI is to help the user interact with the database in order to track luggage during a flight. The GUI is implemented using the Python programming language and the Tkinter library, Python’s standard GUI package. Tkinter is a thin object-oriented layer on top of Tcl/Tk and the most commonly used method for GUI creation. The GUI has several features such as “check in traveler,” “Simulate Carousel 7,” “Start and Stop Simulating,” as well as entries for the user to enter specific details, such as his or her name, flight status, destination, and origin. The GUI utilizes special features and widgets such as buttons, entry text, labels, and a new window feature. Labels are used to describe specific actions in the GUI structure, while buttons provide the user with a “click on” feature that performs a specific command in the GUI. Entry text gives the user an opportunity to enter text in specific required fields and the new window feature allows auxiliary windows to be opened while performing a specific task. All these features come together to make the GUI for the senior design project. A graphical representation of the GUI can be seen in Figure 3.6.5 on the next page.
To connect the GUI to the database in Python, the user needs to import the pymsql library on the computer in which the GUI is being run. To connect the GUI to a database the user only needs to run the command below:

```
Connection = db.Connect (host= " ", port= " ", user=USER, password=PASSWORD, db=DB).
```

### 3.7 Software Problems

A major software issue was connecting the GUI to the database and implementing MySQL commands and was resolved by installing pymsql on the computer.

Another issue involved the remote database not updating each time the local database was updated or changed using the GUI, which was corrected by using pymysql’s commit function. The function makes permanent changes to the remote database when called and pushes all changes to the remote database.
Placing entry text, a button and other widgets used in the making of the GUI was another hurdle. The initial idea was to use the grid function that comes with the Tkinter package in the Python programming language, but this was not successful, as the rows and columns did not fit the size of the frame of the application. An alternative method was to use the “place” function, which utilizes the x and y axis, giving the user an opportunity to place widgets on at specific points within the window. Finally, each time the assign button was clicked, the program would crash instead of closing the window. This issue was resolved by switching the “break” method with a “destroy” function call, provided by the Tkinter library.

Finally, an obstacle encountered during the mobile app software development process was the synchronization of objects between different views. The project involved a large amount of web requests and view changing, which created delays pertaining to the displayed View. In some instances, the class would not pass the object desired to the new class. The difficulty was that the team did not have many iOS development resources available, so everything needed to be solved independently. While the mobile phone could retrieve information from the database through the web, our desktop software was unable to gain access for a time. Eventually it was discovered that network port 443 for MySQL was not enabled and with a simple security alteration, the issue was overcome.

**Chapter 4: Testing and Verification of Design**

**4.1 System Testing**

The team conducted testing of three main components: the mounted airport reader; the mobile bluetooth reader; and the iOS application. Precision was the team’s top priority when implementing and testing the design, whereas refinement was secondary. The primary functionality that had to be in correct operation for the project to be considered a success were the abilities to:

- identify a single RFID tag
- identify multiple RFID tags simultaneously - (the team had 15 tags to work with)
- reflect changes in location status of a tag through an iOS application
When testing the mounted, airport reader, distance of a successful read was prioritized above quantity of tags read. Because an RFID tag would be affixed to the outside of a bag - either adhered to the barcode tag or the body of the luggage - line-of-sight was the testing method used most often. Without obstruction, the airport reader produced the results in Table 4 below.

<table>
<thead>
<tr>
<th>Distance</th>
<th>Single Tag</th>
<th>Maximum Tags (15)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0’</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>5’9”</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>7’7”</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4.1.1: Airport Reader Reading Range**

As shown in the table, the maximum reading range of a single RFID tag was 7’7”, while the maximum reading range of all of the team’s tags was 5’9”. The sensitivity of the antenna can be adjusted such that the maximum reading range of a single tag is approximately 15’, however, this level of sensitivity often resulted in incomplete or incorrect data. Through trial and error, the optimal sensitivity was set which led to the results gathered in Table 4.1.1.

Because the motivation behind the mobile reader was the ability for a traveler to correctly identify a single item, the focus of the mobile testing resolved around the identification of a single tag. Additionally, the Bluetooth connectivity range of the scanning reader to an iPhone was an area of focus. The testing results of the mobile reader can be seen below in Table 4.1.2 below.

<table>
<thead>
<tr>
<th>Function</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bluetooth Connectivity</td>
<td>46’</td>
</tr>
<tr>
<td>Single Tag Scan</td>
<td>3.5”</td>
</tr>
</tbody>
</table>

**Table 4.1.2: Mobile Reader Characteristics**

After budget analysis and verification of proper operation, a more expensive antenna was ordered, which will increase the reading range of the mobile reader to 24”. Testing of the iOS application was more of an exact science than the readers’ testing due to the fact that the app
either displays the correct information or it does not. Design issues such as the application not saving a traveler’s luggage information were identified and quickly overcome during this testing. The application successfully added the itinerary associated with a piece of luggage to a traveler’s profile and displays the status of the bag while concealing the RFID tag ID. Additionally, the application enables the bluetooth reader and correctly emits a sound when the desired tag has been placed within the reading range.

4.2 Testing Result Analysis

The mobile reader component was supplementary to the requirements presented to the team initially, so the testing results yielded in section 4.1 show that all design functionality desired by the team’s sponsor was achieved and exceeded. In reference to the design criteria set by the team in section 2.1, the following notes can be made:

- **Integration** - The design was implemented in such a way that it could be integrated seamlessly into a variety of systems. The scripts created to handle the passage of the data from the readers to the database and application can be easily changed to execute different actions based on the systems involved. For example, an airline would need only grant access to its servers and slightly alter the behavior of the GUI to carry out its needs.

- **Accessibility** - The mobile application was created in a way that is easy for a traveler to understand and use. With the ability for a traveler to track the location of his or her luggage using information known only to that traveler, the application is friendly to users of all technical skill levels.

- **Cost** - The cost of UHF RFID tags is roughly five cents when ordered in bulk. This cost could easily be absorbed by an airline or simply transferred to the ticket cost of a traveler. The costs of the readers - roughly $250 - pale in comparison to the $4 billion lost annually by airlines. Additionally, the cost to create the mobile reader was $350 cheaper than the cheapest available commercial product.

- **Time** - The implementation of the design was often completed ahead of the deadlines set at the beginning of the project. All functionality was completed two weeks prior to design day, allowing ample time to prepare for Design Day.
Chapter 5: Summary and Conclusion

5.1 Final Cost

The final cost of the project was $699.34 and is broken down into an itemized list in Table 5.1 below. Each design team was initially given a budget of $500, however due to the nature of the technology involved in this project, the team had to request an increase in funding. The three most expensive required items - USB reader, UART reader, and Apple Developer account - totaled over $550. Additional costs such as wires, additional tags, battery components, Bluetooth Arduino board, and mounting hardware accounted for the rest of the allocated budget.

<table>
<thead>
<tr>
<th>No</th>
<th>Item Name</th>
<th>Cost-US$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RFID scanner - airport side</td>
<td>248</td>
</tr>
<tr>
<td>2</td>
<td>RFID scanner - mobile side</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>battery and Charger</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>RedBear BLE Arduino</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>Apple Developer account</td>
<td>105</td>
</tr>
<tr>
<td>6</td>
<td>PVC pipes</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>USB programmer</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>Misc.</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>700</td>
</tr>
</tbody>
</table>

Table 5.1: Budget Chart

5.2 Schedule

The team was able to meet all deadlines set at the beginning of the semester, often completing them early. This timely success can be attributed to an efficient delegation of duties as well as the hard work put in by each team member.

5.3 Conclusion

RFID technology can be used in a variety of applications with new possibilities discovered daily. With little to no experience in many of the technological fields involved in this
project, the team was able to quickly research and implement a design that required microcontroller programming, RFID data analysis, iOS development, and database management. When presented with unfamiliar obstacles, the team was always able to overcome through research and dedication. The result of the team’s four months of work has been a product that could be easily integrated into nearly every industry.
Appendix 1: Technical Roles

Appendix 1.1 Brian Prange

My technical contribution to the project involved bridging the gap between the two sub-teams identified as the hardware and software teams. I created the scripts necessary to transfer the data from the Radio Frequency Identification (RFID) reader to the database and mobile application, and vice-versa. My previous academic and work experience using the Python programming language, along with online research, allowed me to create the code necessary to transfer the data to and from a well-organized MySQL database.

The two primary functions involved in the project were associating an RFID tag with a traveler and updating the location status of the luggage as it progresses through the airline baggage system. The first Python script I created involved reading the data that the airport reader produced as it identified tags. The data showed details such as time of the read, frequency of the signal, signal strength, and the unique epc code. Because all epc codes have a unique pattern of twelve sets of two hexadecimal character, separated by hyphens, I was able to create a regular expression pattern to extract the code from the reader information. After obtaining the epc code, I connected remotely to the team’s MySQL database on our hosted Amazon Web Server. Upon successful connection, a sql statement would insert a row into the “baggage” table, associating the traveler information with the tag.

After a bag had been checked in, simulation of the bag progressing through the baggage system was required. Following a similar procedure to the assignment script, I created a second Python script that would indefinitely read the data produced by the RFID scanner. When a tag was recognized, an “update” sql statement was executed changing the “bag_status” of the flyer,
associated with the epc code, from “Checked In” to “On Carousel 7.” This script would run until manually terminated, updating the status of any tag to pass under the reader.

To make this code more user-friendly, I worked alongside Emmanuel to create a Graphical User Interface (GUI) that allowed the scripts to be executed in a more efficient manner. Using the Python library “Tkinter,” Emmanuel and I created intuitive buttons and entry fields to facilitate the processes mentioned above.

Appendix 1.2 Ziye Xing

My technical role in the project was to build the software and database necessary for the system. I developed an iOS phone application and implemented a MySQL database and API on a commercially-available server. I also helped develop the webpage of this project. As a computer engineer, with a focus mainly on a software track, I had competent experience on C/C++, Python, and Java programming.

In order to serve better as a software developer for the project, I learned Swift and iOS development for mobile client, and I also learned server implementation-related knowledge such as PHP and TCP/IP.

The application was developed on the iOS platform using the Swift language. The development environment is the latest Xcode 7.0 platform. The main purpose of this application is to allow the user to check the current location of his or her baggage and keep track of it by using a mobile device. The application was developed for any iPhone running iOS 8. The application also involved synchronized computing about a web service. The uses Bluetooth to communicate with a handheld RFID scanning device, so the application also uses a Bluetooth...
module and handles the data receiving and processing. For more general iOS development detail, please refer to my application note.

I signed up for Amazon Web Server for a year-free service on basic needs to implement the server for the database. I configured the Apache server, setup the TCP ports for preferred service, and made the MySQL database ready to be used. The PHP API was compatible with our iPhone app requests and also echoes the queried data from the database in a preferred format to the client. I also helped develop the team website, mainly using the bootstrap framework.

Appendix 1.3 Emmanuel Wadieh

My technical contribution to the project mainly involved creating a Graphical User Interface. The purpose of this Graphical User Interface (GUI) was to serve as an interface between the user and the database in MySQL, hosted on the amazon web server. A connection was also made to the database in the GUI. GUI’s usually have buttons, graphical icons, text and labels that allow users to implement commands as opposed to text-based interfaces and typed command text in command line prompts. This GUI was made purposely for the senior Design Team 1, and allows users to “Check In” by entering specific information such as the user’s name, flight number, flight origin and flight destination. Once the user is “Checked In”, the user’s tag number associated to the luggage is recorded in the database alongside the user's information hosted by the amazon web server. Once this information is in the database, each time the user goes through specific checkpoints the information in the database is updated allowing the user to know specifically where his or her luggage is. In the case of our senior design project, the only and final checkpoint is carousel 7. This GUI is implemented in Python using the Tkinter Python package.
I also worked alongside Ziye in creating the User Interface (UI) for the mobile application. The purpose of the mobile application is to allow users the opportunity to get updates on where exactly their luggage is during a flight on a mobile device. This UI was implemented in Xcode using the swift programming language. This mobile application is also connected to the main database.

My experience with the Python programming language in prior classes here at Michigan State University and research work involving GUI’s and database in MySQL, helped me contribute to the success of this project.

Appendix 1.4 Marwan Baraya

Since the beginning of this project, I have been involved in the technical details of it. My largest two technical contributions to the project were selecting the parts that would be needed and designing and implementing the handheld bluetooth RFID reader.

The handheld reader idea originated when our sponsor suggested that we create a mobile phone attachment to scan RFID tags. The main constraint with that idea was that the team lacked access to the required hardware licenses to build a mobile device compatible gadget. My approach was to use Bluetooth communication to connect the handheld device to the mobile phone. In this case, we would have a lower cost design since we did not require any licenses to access the hardware of 'iPhone', and our design would be more generic since the handheld device would be compatible with other platforms like Android and Windows phones.

I put together a list of everything we would need in order to implement the handheld device. After that, I demonstrated the idea to my teammates, our sponsor, and facilitator and after getting approvals, I contacted the vendor in China and made sure the right parts were ordered within a reasonable time frame. After getting the parts, I started testing the integration of the
parts together. After a few days of testing and debugging, I found out that the RFID reader and the Bluetooth-enabled Arduino operated on different baud rates, so I used a USB debugger and a converter circuit I built myself to update the firmware on the RFID reader and slow it down from 115200 bps to 9600 bps. At that point, I was able to make the RFID reader and the microcontroller communicate. The next step was to program the microcontroller so it could communicate with the mobile device using Bluetooth. After I was done with that part, I forwarded the device to my teammates who were responsible for developing the mobile application. I also provided them with the software libraries they would need to make the software development easier. After that part was done, I helped testing, debugging, and presenting our project to our sponsor and facilitator.

Appendix 1.5 Henry Nguyen

Throughout the semester, I have been involved in the testing, troubleshooting, and presentations of the product. The majority of my technical contribution to the project was oriented towards the hardware portion of the project. Along with Marwan, I worked on connecting the components of the mobile device: the antenna, the reader/writer kit, the Blended Micro Bluetooth and the rechargeable battery. The first step in integrating the Blended Micro and the RFID reader was the physical hardware connections which required header pins and jumper wires. The Blended Micro did not come with any pin headers so I soldered those on separately. While the user can solder a pin header to every pin hole on the board, only the V33, GND, D1/Tx and D0/Rx pins were needed for this application. Unlike the blended micro, the RFID reader came with pre-soldered pin heads for the GND, Tx and Rx pins. The second phase of the assembly consisted of programming the blended micro to receive and output that the data picked
up from the RFID tags from the reader. The RedBear Lab Blended Micro was chosen due to its ability to connect with both iOS and Android devices. There was also a step-by-step initial setup guide provided by RedBear Labs that was very helpful for the integration. I played a large role in finding a suitable power source for the mobile scanner after testing the device in the lab with the power supply I found that it required approximately 1.3A for initial power up and 3.3V. I then found a suitable battery -- a 2000mAh, 3.7V wireless rechargeable lithium ion battery. After finding all of the required parts and testing their ability to work with one another, I assisted in interconnecting the various parts, testing, and storing the various parts into the portable case.
Appendix 2: References

Appendix 2.1 Works Cited

"Table 1-65: Mishandled-Baggage Reports Filed by Passengers with the Largest U.S. Air Carriers (a) | Bureau of Transportation Statistics." Table 1-65: Mishandled-Baggage Reports Filed by Passengers with the Largest U.S. Air Carriers (a) | Bureau of Transportation Statistics. N.p., n.d. Web. 07 Dec. 2015.

Appendix 2.2 Datasheets

http://redbearlab.com/blendmicro/

Appendix 3: Technical Attachments

Appendix 3.1 Software

The full software for the project has been omitted from this report and the application notes for proprietary reasons as the team is considering developing this product into a business model

Appendix 3.2 Parts List

<table>
<thead>
<tr>
<th>No</th>
<th>Part Name</th>
<th>QTY</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AS3992 UHF RFID reader</td>
<td>2</td>
<td><img src="image" alt="AS3992 UHF RFID reader" /></td>
</tr>
<tr>
<td>2</td>
<td>RedBear Labs BLE Arduino</td>
<td>1</td>
<td><img src="image" alt="RedBear Labs BLE Arduino" /></td>
</tr>
<tr>
<td>3</td>
<td>8dBi UHF RFID antenna 915MHz</td>
<td>1</td>
<td><img src="image" alt="8dBi UHF RFID antenna 915MHz" /></td>
</tr>
<tr>
<td></td>
<td>Item Description</td>
<td>Quantity</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>----------------------------------------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5dBi PCB UHF RFID 902-928M Antenna</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2000 mAh Li-ion Battery</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Li-ion single cell charger</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SPDT Switch</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Assorted 5mm LED’s</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>