Design Specification

For

UAS Delivery Verification System

Prepared by:

Group Name: Group 24

Justin Rutschilling	*rutschilling.22@wright.edu*
Carl Schmidt	*schmidt.124@wright.edu*
Nathan Nichols	*nichols.86@wright.edu*
Kirill Kultinov	*kultinov.2@wright.edu*

For:

Wright State University EE 4920/CEG 4981 Faculty

Document Tracking Information:

Document Version *ver 1.0*

Date *1/26/2016*

Table of Contents

1. Executive Summary

UAS delivery is being talked about as the future of the package delivery service industry. One issue with this service is that there is currently no way to ensure proper delivery of a package or a way to ensure that a package is safely received by a customer. This document will outline the system that we are proposing to use to properly verify an order delivered via UAS. The system will be separated into two sections: UDVS-CS and UDVS-SS. The UDVS-CS will be a smartphone application that will be developed for both Android and iOS operating systems. This application will help control whether a user's package will be dropped off to them or not. The UDVS-SS will use a Bluetooth transceiver to receive a signal from the UDVS-CS that will then be used to control an Arduino microcontroller. This Arduino will control an LED that will flash only if the entire verification process was followed and the order was properly verified. This LED will be used to indicate that the package can be dropped off to the user.

1.1 Purpose of this document

The purpose of this document is to provide information detailing how we will design the proposed UDVS. This document will go through how the separate components will function on their own. The goal of this design document is to discuss how these components will interact with each other and the UAS. This document will also review the operating systems we will develop an application for, as well as the software we will use to program this application.

1.2 Design Scope

The goal of this design is to provide a convenient way for companies to ensure that packages delivered via UAS are delivered safely and to the proper location. This design will be a verification system that will allow the customer to verify their order and delivery before the UAS will drop off the package. This proposed verification system can be divided into two main parts. The first part of the verification system is the UDVS-CS. On the customer side of the system we are developing an application that will be used by the user receiving the package to connect to the UAS. The second part of the verification system is called the UDVS-SS. This is the server side of the system, and will be placed on the UAS. It provides a link to the UDVS-CS that once established will allow the verification process to start. Once the verification process is completed, the UDVS-SS will be used to signal the UAS to drop off the package to the customer.

The UDVS-CS application will be designed to be compatible with both iOS and Android devices. This application is used by customers in order to verify the delivery of their products. The application will be compatible with most devices in the market of mobile devices. Moreover, it will require only two actions from users in order to complete the delivery process. Users will have to have prior knowledge on how to pair devices over Bluetooth, and will have to press the verification button when their packages have

arrived. Our goal is to create a simple and functional application with maximum reliability and minimalistic design in order to avoid any confusion among customers. Java and Swift languages have many libraries in order to develop an application that will be reliable and suitable for our needs. We do not consider cases that will cause the application to fail because of problems on the customer's device. These cases include not properly working NIC, Bluetooth, and lack of cellular service.

The UDVS-SS will be a combination of three different sections, one section being a BlueSMiRF Bluetooth Silver transceiver, another being an Arduino Uno device, and the final section being a green LED. The BlueSMiRF Silver transceiver will interact with the Arduino Uno to turn on the LED if a signal has been sent from the UDVS-CS to the UDVS-SS. If no signal was sent, the LED will not turn on and thus the package will not be dropped off. For the UDVS-SS, we will use the C/C++ programming language to write a simple script that will turn on the LED if a signal is found. The customer that is receiving the package will not interact at all with the UDVS-SS, this part of the system only serves to send out and receive signals so that the verification process can be completed.

1.3 Added Value

While we were implementing our design, it became apparent that the hardware side of our project would be completed much sooner than we had originally planned, due in part to an overestimation of the time it would take to order and receive our components. In order to avoid an abundance of downtime, our three electrical engineering students researched ways to add value to the project. These items were not added to our final project due to time constraints.

One possible addition to the project would be to find a way to verify that the correct package was dropped off. Currently, our project only verifies that the person who receives our verification code is present and able to communicate with the UAS at the time of delivery. A way this could be done is through the use of QR codes. A sample of how this process would play out is as follows: package is scanned at the warehouse in order to load package information to a database, user scans the package at the time of delivery and proper message is displayed to user's screen. If codes match, a message would be displayed to the user to signify that delivery was successful. If codes do not match, a message would be displayed to notify the user that the incorrect package was delivered.

In order to scan out at the time of delivery, our application would make use of the existing camera on a user's Android device. Existing open source code using the ZXing (Zebra Crossing) library would allow us to use the camera as a QR scanner. We would use Firebase to store package information.

Another possible feature that we could have worked on had we realized how quickly the electrical engineers on our team would finish the hardware portion of the UDVS is a sensor that could have been used to verify that the delivery drone was has

landed on the ground and can safely drop off the package. For the purpose of being able to detect the presence or absence of the ground the HC-SR04 should suffice. The HC-SR04 is an ultrasonic range finder that sends out sound waves at a frequency of 40kHz and can detect those sound waves reflecting off of surfaces. Typically, an attached microcontroller would receive a voltage pulse from the device when it detects its own reflected sound wave and use the time between the reflected sound wave being detected and the sound wave being sent out to determine by a time of flight calculation how far away the device is from the surface that caused the sound wave to be reflected. However, in this case the device would simply be used to detect if the drone has landed yet and would not necessarily be as precise. The reason that the HC-SR04 was selected was because of its low cost. The sensor is also reasonably lightweight and simple to implement. ["SainSmart HC-SR04 Ranging Detector Mod Distance Sensor (Blue): Cell Phones & Accessories," *Amazon.com: SainSmart HC-SR04 Ranging Detector Mod Distance Sensor (Blue): Cell Phones & Accessories*. [Online]. Available: https://www.amazon.com/SainSmart-HC-SR04-Ranging-Detector-Distance/dp/B004U8TOE6. [Accessed: 21-Apr-2017].]

1.4 Intended Audience and Document Overview

The intended audiences for this document include the product stakeholders and developers, as well as companies and users who are interested in drone delivery. This document is also intended to be read by faculty and students associated with the spring 2017 iteration of EE 4920/CEG 4981 Team Projects II at Wright State University.

Stakeholders may be interested in the market analysis (2.2) and impact of success (2.4) sections of this document.

Companies interested in drone delivery may wish to refer to the market analysis (2.2), design constraints (3.4), and functionality (3.6) sections of this document.

Customers may wish to refer to the functionality (3.6) section of this document.

1.5 Definitions, Acronyms, and Abbreviations

Acronyms:
UAS - Unmanned Aircraft System
UDVS - **UAS** Delivery Verification System
UDVS-CS - **UDVS** Client Side - The developed application portion of the **UDVS**
UDVS-SS - **UDVS** Server Side - The hardware and software components of the **UDVS** that will be installed on the **UAS**

Glossary:
Active – State of the application when it is on the full screen of the smartphone
Code label – It is a **text label** that contains "Verification code: xxxxxx", where xxxxxx is a randomly generated 6-digit code that is sent to a customer after placing an order.

Hub – Location from which orders are processed, filled, and shipped
Material design - Guidelines to follow when developing an application for a
Menu button - It is an interface element that triggers the appearance of a **Navigation drawer**.
Operate – To function as described by requirements.
Navigation drawer - It is an app component that slides in from the left and contains the navigation destinations which are menu items, after a button on the left upper corner is pressed. It spans the height of the screen, with everything behind it visible but darkened by a scrim.
Smart device – Electronic gadget that is able to interact with user, internet, and other
Status label – A **text label** that represents a current status of a customer's order.
Target version – It is the earliest version of a mobile operating system that can support an application.
Text label – A rectangular GUI used to display text inside the defined area.
Verification button – Button used to verify delivery.

1.6 Document Conventions

N/A

1.7 References and Acknowledgments

1.7.1 Resources

[1] Shaw, Ian. "History of U.S. Drones," Understanding Empire, 2012. [Online]. Available: https://understandingempire.wordpress.com/2-0-a-brief-history-of-u-s-drones.

[2] Bluetooth, "Learn about the history of Bluetooth," 2016. [Online]. Available: https://www.bluetooth.com/media/our-history. Accessed: Nov. 12, 2016.

[3] S. Ganesh, J. R. Menendez, and Q. Incorporated, *Patent US9359074 - methods, systems and devices for delivery drone security*. Google Books, 2014. [Online]. Available: https://www.google.com/patents/US9359074. Accessed: Nov. 12, 2016.

[4] C. *, "Pairing,". [Online]. Available at: http://fte.com/WebHelp/BPA600/Content/Documentation/WhitePapers/BTLE/Pairing.htm. Accessed: Oct. 12, 2016.

[5] I. Bartolic, "Bluetooth Download speeds - data rates of the Bluetooth file transfer," in Bluetooth, The Best Wireless Internet, 2014. [Online]. Available at: http://thebestwirelessinternet.com/bluetooth-download.html. Accessed: Oct. 11, 2016.

[6] "Smartphone sales OS market share United States 2013-2016 | Statistic", Statista, 2016. [Online]. Available: https://www.statista.com/statistics/274121/market-share-held-by-smartphone-os-in-the-us/. [Accessed: 28- Oct- 2016].

[7] "App Store - Support - Apple Developer", Developer.apple.com, 2016. [Online]. Available: https://developer.apple.com/support/app-store/. [Accessed: 28- Oct- 2016].

[8] "Dashboards | Android Developers", Developer.android.com, 2016. [Online]. Available: https://developer.android.com/about/dashboards/index.html. [Accessed: 28- Oct- 2016].

[9] "Android versions comparison | Comparison tables - SocialCompare", Socialcompare.com, 2016. [Online]. Available: http://socialcompare.com/en/comparison/android-versions-comparison. [Accessed: 28- Oct- 2016].

[10] “Data sheet RN-41 - SparkFun Electronics,” Roving Networks Wireless For Less. [Online]. Available: http://cdn.sparkfun.com/datasheets/wireless/bluetooth/bluetooth-rn-41-ds.pdf. [Accessed: 11-Nov-2016].

[11] M. #687926, M. #524187, M. #706912, M. #339413, M. #691777, M. #772221, and M. #383171, “SparkFun Bluetooth Modem - BlueSMiRF Silver,” - WRL-12577. [Online]. Available: https://www.sparkfun.com/products/12577. [Accessed: 11-Nov-2016].

[12] The Zen Cart™ Team and others, “Bluetooth Shield SHD18,” [EF02022]. [Online]. Available: http://www.elecfreaks.com/store/bluetooth-shield-shd18-p-233.html. [Accessed: 11-Nov-2016].

[13] “Using the BlueSMiRF,” Learn at SparkFun Electronics. [Online]. Available: https://learn.sparkfun.com/tutorials/using-the-bluesmirf. [Accessed: 11-Nov-2016].

[14] "App success made simple," Firebase. [Online]. Available: https://firebase.google.com/. Accessed: Jan. 24, 2017.

1.7.2 Standards

[1] “SUMMARY OF SMALL UNMANNED AIRCRAFT RULE (PART 107),” FAA News, 21-Jun-2016. [Online]. Available: https://www.faa.gov/**UAS**/media/part_107_summary.pdf. [Accessed: 22-Aug-2016].

[2] "IEEE Standard for Local and Metropolitan Area Networks: Overview and

Architecture," 2014. [Online]. Available at: file:///C:/Users/ecslogon/Downloads/802-2014.pdf. Accessed: Oct. 10, 2016.

[3] "12207-2008 - Systems and software engineering -- Software life cycle processes," 2008. [Online]. Available at: http://standards.ieee.org/findstds/standard/12207-2008.html. Accessed: Nov. 8, 2016.

2 Problem Statement

Commercial companies such as Amazon and United Parcel Service (UPS) are continuously trying to provide fast and convenient ways to deliver packages to their customer, the latest and maybe greatest idea that is out there right now is to use UASs to deliver these packages. Using UASs to deliver packages is very much still in the beginning stages and there are a number of obstacles that stand in the way of using UASs to deliver all packages for these companies. One of these obstacles is that there is currently no convenient way to ensure the proper and safe delivery of packages to customers.

To ensure that packages are delivered to the right residence and to help ensure that packages are not stolen off customers' property, companies could use a verification system to achieve both of these things. The verification system that we are proposing is a verification system that will ensure that packages can be sent to the right residence and it will ensure that packages cannot be stolen. This is possible because the only way a package can be dropped off is if the customer verifies the order they placed once the UAS is within the Bluetooth range of the Bluetooth transceiver on the UAS. The proposed verification system is a solution to one of the few technical obstacles that stand in the way of the drone industry taking off in conjunction with the commercial delivery industry. The UAS delivery industry will use this verification system to provide a convenient and reliable option to their customers when delivering packages via UAS.

2.1 Historical Introduction

UAS technology has its roots in military applications. During the Spanish-American war, cameras were attached to kites to collect some of the first ever aerial reconnaissance photos. Aerial surveillance was used extensively throughout World War I. In 1917, Elmer Sperry and Peter Hewitt began construction of the radio-controlled "Hewitt-Sperry Automatic Airplane" or "flying bomb". The 1960's saw rapid advancement in military technology that led to The Air Force launching the Compass Cope program in the 1970s to increase the range and electronic surveillance capabilities of Remotely Piloted Vehicles (RPVs). As technologies, such as GPS, have continued to develop, commercial drone use has become feasible [14].

Bluetooth is a standard for short-range wireless communication. In 1998, five companies (Ericcson, Nokia, Toshiba, Intel, IBM) formed the Bluetooth SIG and, in 1999, Bluetooth version was 1.0 released. In 2002, IEEE approved the 802.15.1 specification, which has since been taken over by the Bluetooth SIG. In 2010, the Bluetooth SIG announced the formal adoption of Bluetooth Core Specification Version 4.0. In 2014, the Bluetooth SIG introduces Bluetooth 4.2, adds features for IP connectivity, privacy and speed. [2]

It is through the advances in these technologies that our solution was able to be designed.

2.2 Market Analysis and Relevant Art

Our target market is companies with products that are able to be shipped via UAS. Our secondary market is made up of the customers who make use of our target market's services. There appear to be products similar to ours being developed by companies such as Qualcomm [3], but we did not find any competitors currently on the market. As such, we were unable to find pricing information on similar products. Secondary customers will not have to pay for both downloading and the use of the application. Since we are developing an application for iOS 9 and later and Android 4.4 and later, more than 95% [6] of all potential customers will be able to use or application. We did not see any similar mobile applications that are being used by companies in order to verify the delivery process yet.

2.3 Alternative Approaches

When deciding what low power near field wireless communication technology we would use, we initially focused on Bluetooth, Wi-Fi, and ZigBee. Table 3.1 depicts what the max range is for each of these technologies, whether it has a high power consumption, low power, or medium power and it shows whether the technology is Android and iOS compatible.

Table 3.1 Comparison of potential wireless technologies

Wireless Technology	Max Range	Power Consumption	Android/iOS Compatibility
Bluetooth	100 meters	Middle	Yes
Wi-Fi	200 meters	Highest	Yes
ZigBee	50 meters	Lowest	No

Using this table as well as doing additional research, it was decided that would use Bluetooth due to its lower power consumption, sufficient range, and iOS and Android compatibility. Additionally, by using Bluetooth we are able to limit the risk of hacking. Bluetooth limits the range at which our system can be communicated with and requiring a verification code to pair with our system reduces the likelihood that our package could end up in the wrong hands. We also considered using Wi-Fi due to its long-range capabilities, but its larger power consumption ruled it out. ZigBee's incompatibility with Android and iOS was the major factor that it was thrown out when choosing which wireless technology to utilize.

We chose to use an application because it will allow for more mobility for the customer and because most smart devices ship with Bluetooth functionality.

While choosing an operating system for our application we have two goals in mind. The first goal is to allow as many potential customers as possible to use our application. This

will also be beneficial for companies because it will allow these companies to get more money since there will be more potential customers. The second goal is to complete the project within the given time range. Table 3.2 contains the information regarding market share of mobile operating systems in the U.S. As you can see, the majority of the market belongs to iOS and Android. Developing for these operating systems will allow us to cover 96.3% potential customers. Also, we have team members who are experienced in developing applications for iOS and Android. There are libraries that will allow us to achieve our goals developing for these operating systems. We did not choose Windows OS and any other OS because their market share is very low and we do not have any team members that know how to develop for these operating systems. Without the required knowledge, our project may not be feasible because there is high probability that it will not be finished on time.

Table 3.2 Market share of mobile operating systems [6]

Operating System	**Market share percentage**
Android OS	65%
iOS	31.3%
Windows OS	2.4%
Other OSs	1.3%

As soon as it was decided that we were going to use Bluetooth to connect our different systems we then had to decide what kind of Bluetooth transceiver we were going to use on the UAS. After doing research on the different options available we concluded on three options, Table 4.1.0 shows these different options as well as the main specifications for each option.

Table 2.3.1: Different options available for use [Sources: 8, 9,10]

	BlueSMiRF Gold	BlueSMiRF Silver	Bluetooth Shield SHD18	Units
Supply Voltage Range	3.0-6.0 Typical: 3.3	3.0-6.0 Typical: 3.3	2.8-3.5 Typical: 3.3	V
Transmission Range	100	18	10	Meters
RF Transmit Frequency	2.402 ~ 2.480	2.402 ~ 2.480	-	GHz
Temperature Range	-40° ~ 70°	-40° ~ 70°	-	Celsius
Serial communications	2400 - 115200	2400 - 115200	9600 - 460800	bps

Dimensions	42x16.5x5.6	45x16.6x3.9	57.4x45.3x19.4	mm
Class of Bluetooth Device	Class 1	Class 2	Class 2	
Weight	-	-	10 +/- 2	grams

After looking through all of the specifications, it was decided to use the BlueSMiRF Silver Bluetooth shield because most smartphones today are either a class two or class three device. Since this is the case, it would not make sense for us to choose the BlueSMiRF Gold device because we could never reach the range of a class one device because we are limited by the Bluetooth in the smartphone. We also chose the BlueSMiRF Silver device because we could supply 5 volts directly to it from the Arduino as compared to the Bluetooth Shield SHD18, which is also a class two device, which could not have 5 volts supplied. Price was also a consideration when deciding which Bluetooth shield to use. The BlueSMiRF Silver was a little bit cheaper than the BlueSMiRF Gold, which was another reason not to choose the Gold device.

2.4 Impact of Success

If our solution is successfully launched and adopted by companies, we expect to see increased consumer confidence in UAS delivery. One would expect this to lead to an increased use of UAS delivery. This increased use could have a variety of global, economic, environmental and societal impacts.

On a global economic scale, countries that allow drone delivery may see an overall increase in business while those who do not embrace this technology may fall behind. This potential lag could lead to many companies moving operations to places that would more conducive to growth.

As previously stated, our solution could allow customers to feel more secure in drone delivery services. When consumer confidence is higher, it leads to more spending. Our solution would also reduce liability for companies, which in turn would lower the costs, and increase the profits, of those companies.

In a societal sense, as drones become more commonplace, society will become more comfortable with them. This increased comfort could lead to drones being used for a wider range of applications.

Environmentally, increasing drone usage will also lead to an increase of waste from their batteries and other components that will need to be disposed of. If an adequate disposal system is not developed, this could lead to negative environmental effects.

3 Context of Design Solution

Section 3.1 of this document discusses the design objectives of the system; it also discusses any assumptions that will be made when designing. For the verification, system we are designing to be considered complete it will need to have followed all the design objectives and adhere to all the standards set forth in section 3.5.

3.1 Design Objectives

The solution that is being designed by this team is a UAS delivery verification system. This system is used to ensure proper and safe delivery of packages to a designated delivery location. This system should be able to complete the following objectives:

> Objective 1. Establish connection between a user's smart device and a Bluetooth transceiver.
>
> Objective 2. Allow user to verify delivery in the application developed.
>
> Objective 3. Give visual confirmation that the verification process has been successful.

In order to complete Objective 1 the system will utilize a commercially available Bluetooth transceiver and the built in Bluetooth capabilities of customers' smart devices. Table 3.1 outlines the wireless technologies that we considered to help us establish a connection between the customer and the UAS.

The system will complete Objective 2 by allowing users to verify delivery with an Android/iOS application that we will develop. We chose to use an application because it will allow for more mobility for the customer and because most smart devices ship with Bluetooth functionality.

After the delivery has been verified, the system will complete Objective 3 with an LED that will let the user know that the verification signal has been received.

Our design does not specify how the drone will reach its destination, nor does it specify the means for the UAS to drop off the package.

3.2 Design Assumptions

The entire system will have the following assumptions, which affect who can use the system and how the system can be used:

1. The application that will be developed on the **UDVS-CS** will be developed for iOS operating system version 9 and later, it will be developed for the Android operating system version 4.4 and later. For iOS devices, the application will be designed using Xcode IDE and for Android devices the application will be designed using Android Studio IDE.

2. Our system will utilize a low power near field wireless communication device that will be used on the **UAS** to connect to the application on a user's smartphone.

3. In conjunction with the low power near field wireless communication device, an Arduino Uno will used to control the LED on the **UAS.** This LED will flash if the low power near field wireless communication device receives a signal from the **UDVS-CS** saying the order was verified and the UAS is ready to drop off the package.

4. A 9 volt battery will be used to power the low power near field wireless communication device and the Arduino Pro.

5. It is assumed that all off the shelf components will be commercially available to us for purchase

3.3 Design Requirements

3.3.1 User Interfaces

UI-1: Our application shall follow **Material design** guidelines
UI-2: Our software shall have a **Navigation drawer**
UI-3: The application shall contain a **Verification button**

3.3.1.1 Test Plan User Interfaces

TUI-1: Open the application on both the iOS and Android devices and make sure the Material design guidelines have been followed.
TUI-2: Open the application on both the iOS and Android devices and ensure that both applications have a Navigation drawer included in them
TUI-3: Open the application and go through the verification process and ensure that during that process a Verification button is displayed to the user

3.3.2 Hardware Interfaces

HI-1: The application shall be constantly checking if a customer's device is paired with the drone while the application is in use
HI-2: Devices shall have working wireless modules
HI-3: Devices shall be able to connect to the Internet

3.3.2.1 Test Plan Hardware Interfaces

THI-1: A greyed out verify button shall signal that a pairing has not been made. An enabled verify button shall signal that a pairing has been made. If the button is enabled once a pairing has been made, we will know that HI-1 has been satisfied
THI-2: Ensure that the smart device can pair to other Bluetooth devices before pairing with UAS. We will use a Bluetooth speaker to make sure that the Bluetooth functions as intended
THI-3: Open the smartphone and ensure that the device is either connected to internet via Wi-Fi or LTE

3.3.3 Software Interfaces

SI-1: Xcode version 8 IDE shall be used to develop an application for iOS

SI-2: Android studio version 2.2.3 IDE shall be used to develop an application for Android

SI-3: Firebase real time database shall be used to send the current delivery status to the user's device.

SI-4: External Accessory framework shall be used to implement data transfer from the user's iOS device to the drone

SI-5: Swift 3 and Objective-C programming languages shall be used to write code for the iOS application

SI-6: Android API shall be used to implement data transfer from the user's Android device to the drone

SI-7: C/C++ programming language shall be used to write code for the Android application

3.3.3.1 Test Plan Software Interfaces

TSI-1: Open the Xcode IDE and make sure that the current version is 8

TSI-2: Open the Android Studio IDE and make sure that the current version is 2.2.3

TSI-3: Change data in the Firebase database and make sure that these changes are also displayed on the mobile device

TSI-4: Check the connectivity by calling on appropriate functions from the framework. Send sample packets of data and check if the data is received on the UAS's side by the Bluetooth transmitter

TSI-5: Inside IDE, make sure that each new file is created as swift 3 file. make sure that Objective-c code is saved with either .m or .h extensions

TSI-6: Check the connectivity by calling an appropriate functions from the framework. Send sample packets of data and check if the data is received on the UAS's side by the Bluetooth transmitter

TSI-7: Open the Arduino IDE and ensure that the program is in fact written in C/C++

3.3.4 Communication Interfaces

CI-1: The Security Manager Protocol shall be used to send data between the user's device and the UAS

3.3.4.1 Test Plan Communication Interfaces

TCI-1: Inside the Bluetooth framework for mobile application, make sure that the Security Manager Protocol is set as a Protocol for data exchange.

3.3.5 Functional Requirements

FR- 1: After order is placed, verification code shall be sent to **UDVS-CS**

FR- 2: After correct verification code has been entered by user, **UDVS-CS** shall establish connection with **UDVS-SS**

FR- 3: After connection between **UDVS-CS** and **UDVS-SS** has been

established, there shall be a visual indication to the user that delivery may now be verified

FR- 4: After verification has been made to confirm delivery, a light located on the **UAS** shall turn on

FR- 5: After package has been delivered, user shall be notified that transaction is complete

FR- 6: To achieve a successful authentication process, when the user is signing into the application, the user shall enter correct login credentials.

3.3.5.1 Test Plan Functional Requirements

TFR-1: Use the real time database to simulate a placed order and see if a verification code is sent to the user's smart device. If the smart device receives a verification code, we will know that FR-1 is satisfied.

TFR-2: We will open the smartphone and go to the Bluetooth settings menu, select "UAS name" (The name that will come with the device). Once you select the name, you will be prompted to enter the code. If you enter the correct code, pairing will happen, if you enter the incorrect code the system will not pair.

TFR-3: Open the application and ensure that once the UDVS-CS and UDVS-SS have been paired via Bluetooth that the verify button in the application lights up and is able to be pressed

TFR-4: Ensure that once the verification process has been completed that an LED placed on the UAS lights up

TFR-5: Traverse through the entire verification process and once the LED has turned on ensure that a message is sent to the smartphone that says the transaction has been completed

TFR-6: The user enters correct username and password and then verifies successful authentication process by looking at the console on FireBase's website.

3.3.6 Performance Requirements

PR- 1: After correct verification code has been entered, the **UAS** shall wait 45 seconds for connection between **UDVS-CS** and **UDVS-SS** to be established

PR- 2: After connection has been established, the **UAS** shall wait 20 seconds for **Verification button** to be pressed by user

PR- 3: After verification has been made to confirm delivery, a light located on the **UAS** will light up within 5 seconds

3.3.6.1 Test Plan Performance Requirements

TPR-1: Use a stopwatch to time the how long it takes the UDVS-SS and the UDVS-CC to successfully pair. This should take 45 seconds or less

TPR-2: Use a stopwatch to time the process between the verification code being entered and the verify button being hit in the application, by the user. Ensure that, that time is at 20 seconds or less

TPR-3: Use a stopwatch to time how much time passes from when the verify button has been pressed and the LED illuminating on the UAS, this should

take no longer than 5 seconds

3.3.7 Safety Requirements

SR- 1: A verification code shall be used to confirm delivery for security purposes

SR- 2: **UAS** shall weigh under 55 lbs including cargo and all equipment [14]

3.3.7.1 Test Plan Safety Requirements

TSR-1: We will confirm delivery using the verification code provided then we will attempt to confirm delivery without using the verification code and make sure that we cannot.

TSR-2: Use a scale to make sure that the UDVS-SS and the UAS as a whole weighs under 55 lbs.

3.3.8 Software Quality Requirements

N/A

3.3.9 Hardware Quality Requirements

HR-1: A power source shall provide a voltage to the Arduino Uno that meets an objective of 9 volts with a threshold of +/-0.5 volts.

HR-2: The Arduino Uno and Bluetooth module shall be connected as depicted in section **4.1 Hardware Components.**

HR-3: The Arduino Uno shall provide a voltage to the Bluetooth module that meets an objective of 5 volts with a threshold of +/-0.5 volts

HR-4: An LED shall be attached to the Arduino to help signify a successful verification process has been completed. This LED shall have a 220 ohm resistor attached to it.

3.3.10 Test Plan Hardware Quality Requirements

THR-1: Using a multimeter we will make sure that 9 volts +/- 0.5 volts is provided to the Arduino by placing the positive lead of the multimeter to the positive of the power jack and the negative lead of the mulimeter to the ground of the power jack.

THR-2: We will use the wiring diagram provided in section **4.1 Hardware Components** and make sure we wire the Arduino Uno and Bluetooth module as it is depicted in that section.

THR-3: Using a multimeter we will ensure that 5 volts +/- 0.5 volts is provided to Bluetooth module. We will do this by placing the positive lead of the multimeter to the VCC pin on the Bluetooth module, then place the negative lead to Ground pin on the Bluetooth module.

THR-4: We will ensure that the right resistor is used by verifying the color code on the resistor signifies it is a 220 ohm resistor. We will also ensure the LED is wired correctly by referring the wiring diagram in section **4.1 Hardware Components**.

3.4 Design Constraints

3.4.1 Technical Constraints

Technical constraints are to ensure that the **UAS** delivery verification system functions properly. The constraints detailed in this section have to do with the nuts and bolts of the design of the **UAS** delivery verification system.

TC- 1: When attempting to pair the communication device on the **UAS** and the communication device the customer is using, errors could be incurred and could prevent proper connection from being made.
TC- 2: The wireless communication device used (**UDVS-SS**) on the **UAS** must be successfully **installed** and it must have a proper communication link to the **UAS.**
TC- 3: The operating system that is developed must be compatible with customers' devices.
TC- 4: Power to the **UDVS-SS** will be supplied by a power source.
TC- 5: Corruption of the **UDVS-SS** may occur if a virus is sent to it from the **UDVS-CS**.
TC- 6: When connecting devices, using low power near field wireless communication systems, there is a certain amount of time delay before proper connection is made.
TC- 7: The device will emit a light that will show that the order was verified.
TC- 8: The light that is pulsed will be pulsed at a frequency in the range of 10-20 Hz.
TC- 9: The light emitted will be in between the range of 50-200 lumens.

3.4.2 Non-technical Constraints

Whereas technical constraints mainly dealt with ensuring that the design itself worked, non-technical constraints mainly deal with regulations, standards, and ease-of-use issues. The main purpose of addressing non-technical constraints is to ensure that the product is legal, safe, marketable, and is compatible with existing technology.

NTC- 1: The **UDVS-CS** will be able to be easily navigated, so no training or tutorials will be needed to use the application.
NTC- 2: The **UAS** that is delivering the packages must comply with Part 107 of the FAA Small **UAS** Rule for Unmanned Aircraft System [14].
NTC- 3: The application that will be developed will send out a verification code and an approximate delivery time to the customer.
NTC- 4: In order to establish proper connection between the **UDVS-CS** and **UDVS-SS** the customer will be outside and in range or the **UDVS-SS** and will enter the verification code once in range.
NTC- 5: To ensure that the application will be adopted by the operating

system we design it for, we will be required to follow the application guidelines for said operating system.

NTC- 6: The **UAS** will **Operate** at temperatures in the range of -20-40 degrees Celsius.

NTC- 7: The **UAS** will **Operate** at humidity's in the range of 0-80 percent humidity.

NTC- 8: The wireless communication system that is used will comply with the standard: IEEE 802 the standard for wireless communication [15].

3.5 Master Test Plan/ Test Results

3.6 Design Standards

Rules for operating an unmanned aircraft are necessary to ensure public safety. SUMMARY OF SMALL UNMANNED AIRCRAFT RULE (PART 107) provides regulations for operating small unmanned aircraft [14]. This document states the operation guidelines such as altitude, time of day that the aircraft can be flown, rules for pilot certification, and aircraft's weight limit requirements. This standard will be used to ensure the operation of the **UAS** will adhere to the rules set by the FAA.

The IEEE 802.15 standard provides specifications that the communication systems on the **UDVS-SS** will need to meet in order for it to function properly as a Bluetooth device [15].

ISO/IEC 12207:2008 provides guidelines for how the **UDVS-CS** software is handled and used from development until disposal [16]. This standard will ensure that the **UDVS-CS** is properly maintained and reliable.

The ANSI/IEC 60529-2004 standard describes a system that can be used to ensure that the **UDVS-SS** hardware portion of the device that is attached to the **UAS** remains functional even when operating in bad weather [15].

3.7 Design Functionality

The proposed design will have three main functions. The first function will be to establish a Wireless connection between the user's smart device and a Bluetooth transceiver on the UAS. The BlueSMiRF Silver Bluetooth module, by SparkFun Electronics, was chosen as our transceiver due to it being a class two Bluetooth device. This was chosen because most smartphones are either a class two or a class three device so a class one Bluetooth module is not needed for connection.

The second function of our design is to allow users to verify delivery. Once the Bluetooth pairing has been made, the application will advance to the next state. During this state,

the verification button will be enabled so that the user may use it to verify the delivery. When the user presses the verification button, a verification signal will be sent from the smart device to the Bluetooth transceiver. Once this signal is received by the transceiver, it will send a signal to the Arduino which will turn the LED on. This LED will serve as a visual confirmation to the user that the verification has been received.

3.8 User Characteristics

User-1: Companies that use **UAS** delivery services

- Companies are those people who are planning on using a **UAS** in order to deliver products to customers.
- Companies that use **UAS** delivery services will use our product to ensure that packages are safely delivered to their customers so that they have a strong customer satisfaction.

User-2: Customers

- Customers are those people that utilize the services of User-1.
- Customers are those people who have login credentials for a certain company that provides **UAS** delivery.
- Customers are those people who know how to pair to devices via Bluetooth.
- Customers are those people who has a device running either Android OS 4.4 and later or iOS 9 or later.

User-3: Mobile device users

- Mobile device users are those people who have a device running either Android OS 4.4 and later or iOS 9 or later.
- Mobile device users are those people who do not have login credentials.

User-1 is allowed to wire our product to their UAS's drop off tool in order to get a signal of successful/unsuccessful delivery only. Any other changes to our product's hardware are not allowed for this type of users. A User-1 will be provided with the source code for the Bluetooth transmitter in order to send a randomly generated passcode to each customer.

The intended frequency of use for our mobile application is every time a User-2 gets a product delivered by UAS from a certain company. Also, whenever a User-1 delivers products using UAS. A User-2 may want to review a manual for our mobile application. In this case, a User-2 will have to use the application. User-2 will not be able to make any changes to their login credentials and any delivery information. A User-2 will have access to all features of our application. A User-3 Mobile will be able to see a manual for the application and the application's version number.

The downloading and installation of our application is required in order to use it on a device that runs on either iOS or Android operating system. That's why, a User-2 and a User-3 have to know how to download and install mobile applications from AppStore and Google Play.

The mobile application is able to properly operate when a mobile device is able to operate. All functionalities of the application are available when a mobile device has the internet connection and properly working Bluetooth. Users will be able to access the user manual only if the internet connection is unavailable and Bluetooth does not work properly.

3.9 Operating Environment

3.8.1 Operating systems and versions

Our application will be available at the AppStore for all devices supporting iOS and at the Google Play for devices running Android operating system. It is uncomfortable to stay outside with a laptop or a PC and confirm the delivery. It is much easier to do it holding a small device in one hand. This is why we decided to develop an app for mobile operating systems.

We had to choose what operating system our application will be available on. Our primary concerns were feasibility of the project and availability of the final product for customers. We chose both Android and iOS because one of our team members has experience developing and designing applications for these operating systems. Because of differences in the development process for Android and iOS, it is going to take almost twice longer than developing for one of these operating systems.

We did not choose Windows because we have no members in our team experienced in developing for this operating system. Also, Windows' market share in the United States is only 2.4 percent [6]. We did not choose any other existing operating system for the same reason. Their market share is 1.3 percent [6]. Also, market share of mobile operating systems are represented in Figure 3.8.1. This figure is showing that developing for iOS and Android will allow us to cover 96.3 percent of the market. It means that almost all customers will be able to use our mobile application.

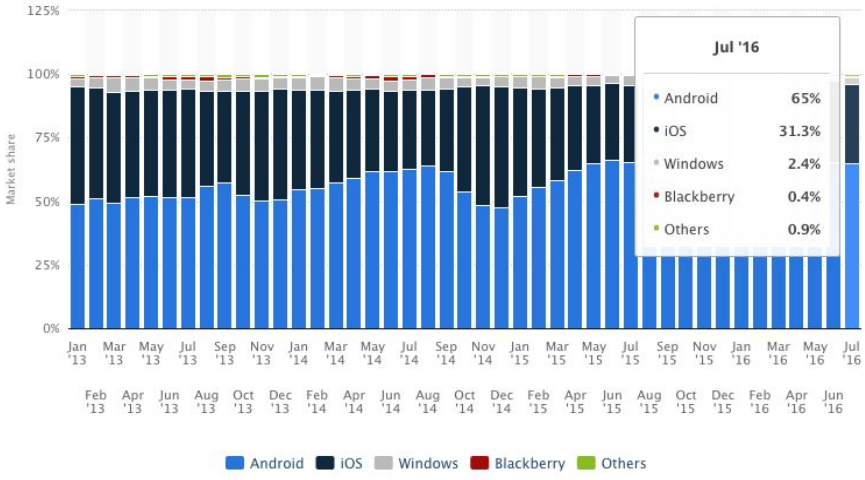

Figure 3.8.1 Mobile OSs market share in the US [6].

We also had to choose an operating systems version that our application will be developed for. One of our primary concerns was to cover as many potential customers as possible. Also we needed to make sure that it is possible to make the application have the same design for both iOS and Android because we are implementing Material design rules. It means that Both iOS and Android application have to follow Material design rules. It is important to mention that Material design is made by Google. It means that we have to be careful with choosing iOS version because it can be impossible to implement Material design on some versions of iOS. We chose iOS 9 as the target version. The ninth version of mobile operating system from Apple was chosen because there is a SWRevealViewController library that allows us to implement Material design following all rules. Earlier versions do not support libraries that would satisfy our needs. Also, all versions below iOS 9 are used on 8 percent of all devices running iOS [7]. Figure 3.8.2 shows that 92 percent of all iOS devices are running iOS 9 or later. It means that we will make our application available for majority of iOS users..

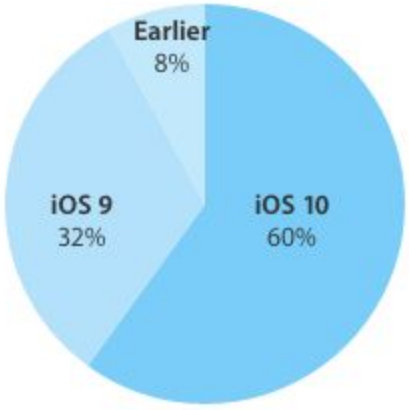

Figure 3.8.2 iOS versions distribution [7]

We used the same strategy in choosing an Android version which meets all our needs. We chose Android 4.4 because 97.3 percent of all Android devices are running Android 4.4 or later [8]. It also means that 97.3 percent of all Android devices will be able to run our application. Table 3.8.1 shows Android version distribution. Also, we chose Android 4.4 because it supports large payloads over Bluetooth and multi-action notifications [x]. These features are important for our project because it will allow us to create a user-friendly application.

Table 3.8.1: Android OS versions distribution [6]

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%

2.3.3 - 2.3.7	Gingerbread	10	1.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.3%
4.1.x	Jelly Bean	16	4.9%
4.2.x		17	6.8%
4.3		18	2.0%
4.4	KitKat	19	25.2%
5.0	Lollipop	21	11.3%
5.1		22	22.8%
6.0	Marshmallow	23	24.0%
7.0	Nougat	24	0.3%

3.8.2 Environmental conditions

Our application will be able to run as long as the device it is running on is able to operate. Devices may not be able to operate in some environmental conditions. We do not specify these conditions because our application will be supported by many different devices. Each device operates in its own environmental conditions. A device that runs our application must have working Bluetooth and active connection to the internet.

3.10 User Documentation

Our mobile application will have user manual inside the about section, which will be under the menu option. The user manual will explain the application's purpose and how it works. It will also explain how the delivery will be verified in a step by step manner. The manual will explain what each element under the delivery status information means and how to establish a communication link between the smartphone and UAS's Bluetooth transmitter

4 Technical Approach

The system that is being developed will provide a means of verifying an order placed by a customer through the use of an application that is ran on both iOS and Android devices, this application will be used to connect the cellular device to a Bluetooth shield used on the UAS. When an order is placed a verification code and approximate delivery time is sent to the application on the user's device. The user will use this verification code to connect to the drone once the drone is within the range that is outlined in section 3.3. Once proper connection has been made the user will use the application to verify the order by pressing the verify button in the application. Figure 4.0.1 shows the state diagram for the entire system and depicts how the delivery process will be traversed through.

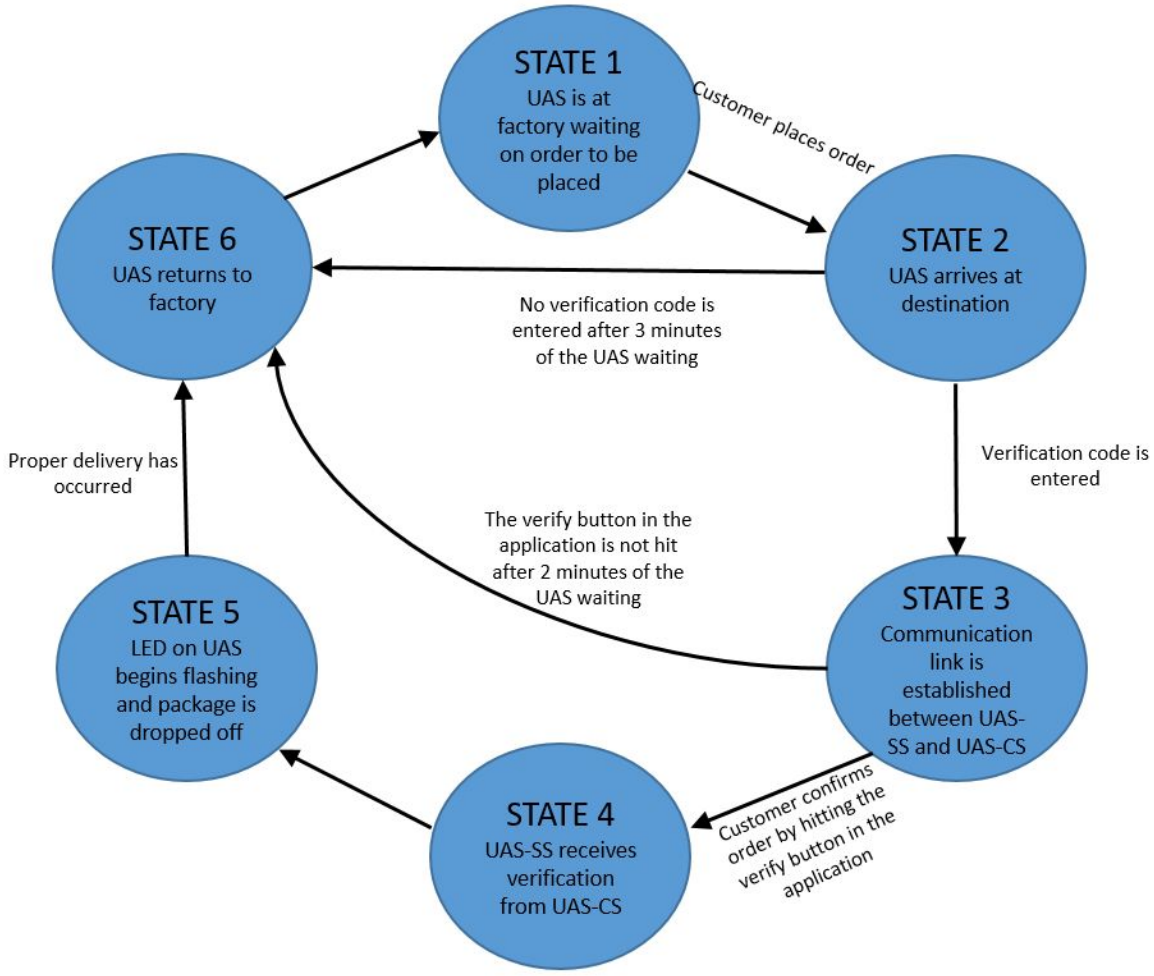

Figure 4.0.1: State Diagram of System [Source: Author]

The application that is being developed will be compatible with iOS operating system version 9 and later and Android operating system version 4.4 and later. The application

will serve as the UDVS-CS, and it will be used to verify the order that the customer has placed. This application will be designed using Xcode IDE for the iOS portion and Android Studio IDE for the Android portion of the application. When the UAS is within range of the user's Smart device the user will enter the verification code they were given to pair with the BlueSMiRF that will be used on the UAS. Once a proper connection link has been established between the devices the user will go back into the application and use the green verify button to confirm the delivery.

On the UDVS-SS of the system a Bluetooth shield will be used in conjunction with an Arduino Uno to control when the UAS will drop off the package. The Bluetooth shield that will be used is a BlueSMiRF Silver Bluetooth shield. This is a commercially available product that can be easily attached to an Arduino. An Arduino Uno will be used along with this BlueSMiRF Bluetooth shield. This Arduino will control the functionality of an LED that is placed on the UAS. The Arduino can be integrated into the UAS to control whether a package will be dropped off or not. The functions of the Arduino will be controlled by the Bluetooth shield, if the Bluetooth shield does not receive a signal from the UDVS-CS the Arduino will not turn on the LED. If a signal is received by the Bluetooth shield the Arduino will go through the process of turning on the flashing LED. To control the Arduino the C/C++ programming language will be utilized to code the system. We will use C/C++ to tell the Arduino to turn on the LED when a signal is received from the drone. When the drone is waiting for a signal to come in the C/C++ code will tell the Arduino to wait three minutes for it to receive a signal from the Bluetooth shield and if no signal is received the UAS will return to factory where it came from. If a signal is received by the Bluetooth shield but the verify button was not pressed in the application, by the user, the C/C++ code will be used to tell the UAS to return to the factory where it came from after waiting for two minutes.

4.1 Hardware

The system that is being designed has a mixture of software and hardware components that will need to be able to interact with each other. The assumptions that are made for what is being used in terms of the hardware used are specified in section 3.2. The constraints that the hardware adheres to are described in section 3.4. While the hardware system that is designed will only be used on **UDVS-SS**, the entire system is reliant on the user having a smartphone to use our **UDVS-CS** in conjunction with the **UDVS-SS**. The following figure Diagram 4.1.0 shows a block diagram of how the UAS and the delivery process reacts to certain decisions made by the user in the application.

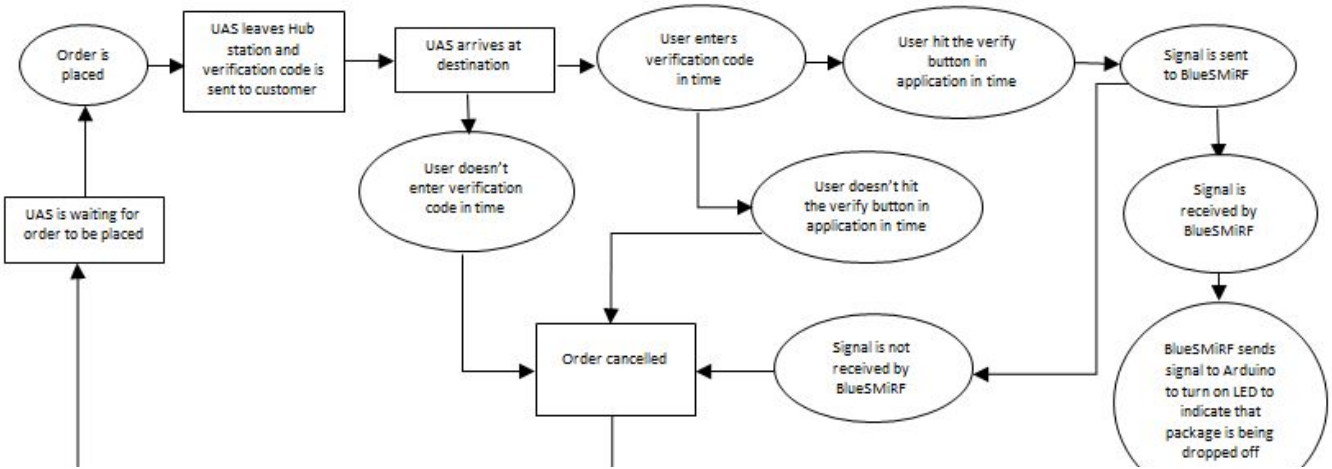

Diagram 4.1.0: Block Diagram of Delivery Process

One of the assumptions that is made in section 3.2 is the use of Bluetooth technology using a BlueSMiRF Silver. The BlueSMiRF Silver device is a type of Bluetooth shield that can be compatible with different microcontrollers to run certain functions [11]. As seen in the table comparing the BlueSMiRF Silver to two other Bluetooth shields in section 2.3 the reason we decided to use the BlueSMiRF Silver as opposed to the other devices is because it was cheaper than the BlueSMiRF Gold, it is a class two device so it has the same range of most smartphones, and it has a lower power consumption compared to other Bluetooth devices. Figure 4.1.0 is a graph of received power (for the BlueSMiRF Silver) vs. range to twenty meters. This graph was made using Friis transmission equation. We assumed when using this equation that the BlueSMiRF Silver was the receiver and the smartphone was the transmitter. For a Bluetooth device the transmitted power at max is around four dBm, we used this value for our power transmitted. We assumed that both gain values for the receiver and transmitter were 1 because we are using isotropic antennas. We also used a frequency of 2.44 GHz because this was the average for the frequencies ran on the BlueSMiRF Silver. Finally, we used the range between zero and twenty meters. Having all these values, we found the received power at different lengths away from the smartphone. This graph shows the reader how much power we should expect to see at certain distances.

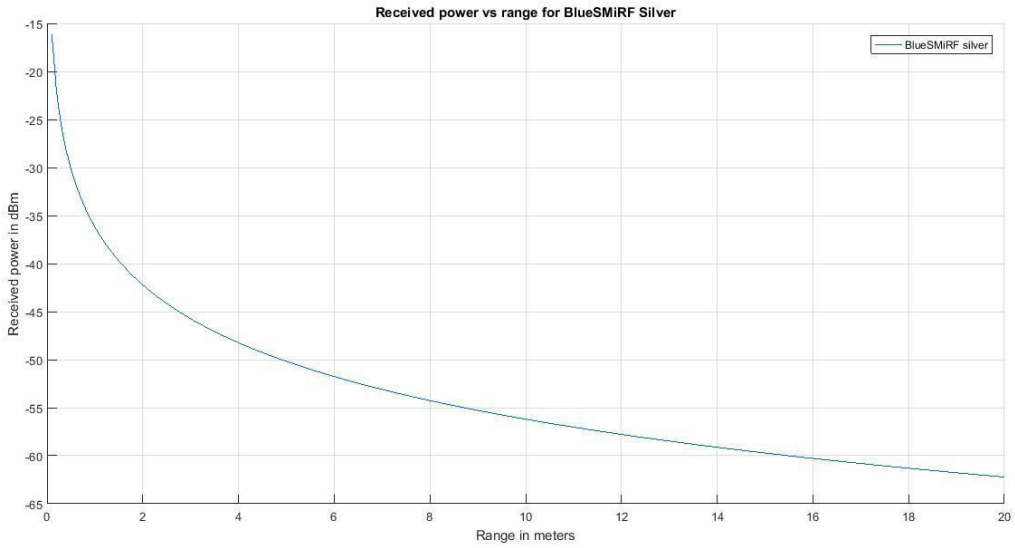

Figure 4.1.0: Received Power (dbm) of BlueSMiRF Silver vs. Range (meters) [Source: Nathan Nichols]

Another assumption that was made in section 3.2 was to use an Arduino Uno. This was decided because while doing research for the Bluetooth shield it was found that the majority of people who used the BlueSMiRF recommend using the Arduino Uno especially when just wanting to control an LED, like we will be doing. A Raspberry Pi is not needed because we do not need a lot of memory space. The following image Figure 4.1.1 shows how all of the hardware components will be connected together so that the components can communicate together:

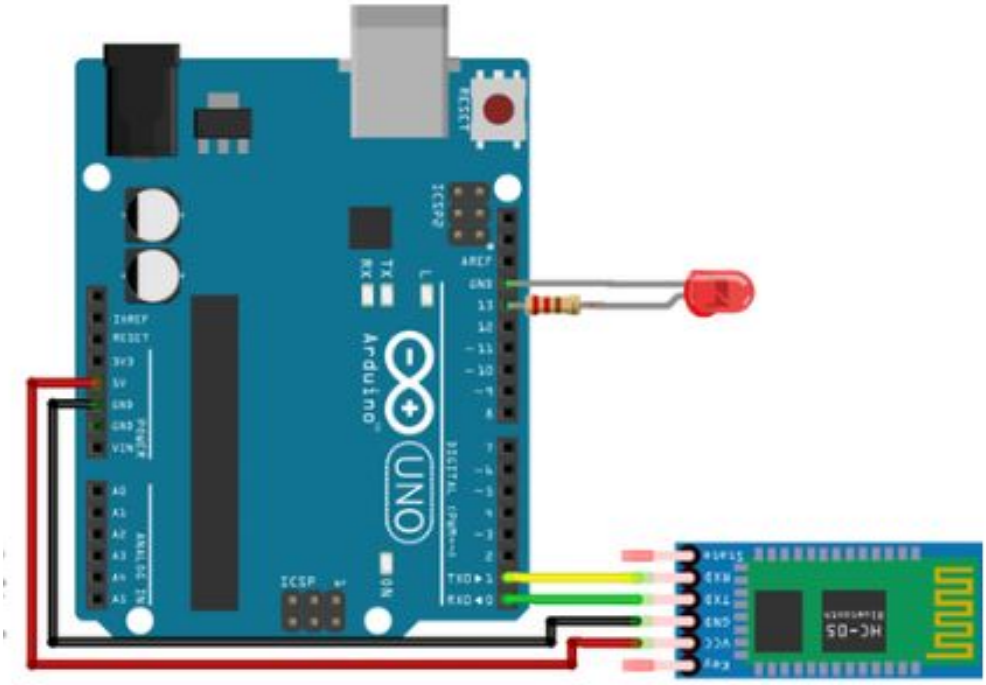

Figure 4.1.1: How the Hardware components are connected [Source: 13]

Power will be provided to the Arduino via a 9-volt battery, this battery will supply voltage to the Arduino and then the Arduino circuit will supply the 3.3 volts needed for the BlueSMiRF Bluetooth transceiver. Figure 4.1.2 explains the pin layout that is used on the BlueSMiRF, these pins will be used to power the BlueSMiRF and to send and receive information.

Pin Label	Pin Function	Input, Output, Power?	Description
RTS-O	Request to send	Output	RTS is used for hardware flow control in some serial interfaces. This output is not critical for simple serial communication.
RX-I	Serial receive	Input	This pin receives serial data from another device. It **should be connected to the TX** of the other device.
TX-O	Serial transmit	Output	This pin sends serial data to another device. It **should be connected to the RX** of the other device.
VCC	Voltage supply	Power In	This voltage supply signal is routed through a 3.3V regulator, then routed to the Bluetooth module. It should range from **3.3V to 6V.**
CTS-I	Clear to send	Input	CTS is another serial flow control signal. Like RTS, it's not required for most, simple serial interfaces.
GND	Ground	Power In	The 0V reference voltage, common to any other device connected to the Bluetooth modem.

Figure 4.1.2: Pin Layout on the BlueSMiRF Bluetooth Transceiver that Provides Interface Between the BlueSMiRF and the Arduino Uno [Source: 11]

4.2 Software

This section explains few optimistic and pessimistic cases that happen while customers are using our mobile application. Figure 4.2.5 shows a screen that appears when a user opens an application for the first time. A user can either enter the login credentials in order to access information about the delivery or click the menu button that will trigger the appearance of a navigation drawer. If user enters invalid credentials then a prompt will appear that asks a user to re-enter the login credentials. The navigation drawer contains a menu of available options like shown on figure 4.2.6. A user can either go to the login page or the About section.

The About section will provide the information about the current version of the application and a user manual that explains how to use the application in order to verify the delivery. The navigation drawer slides to the left and a currently chosen screen appears every time a user clicks on an item from the menu. For example, if the user clicks on the delivery status then Delivery Status screen appears where a user is either asked to enter the login and password or the delivery status is shown if the entered login credentials are valid (Figure 4.2.1).

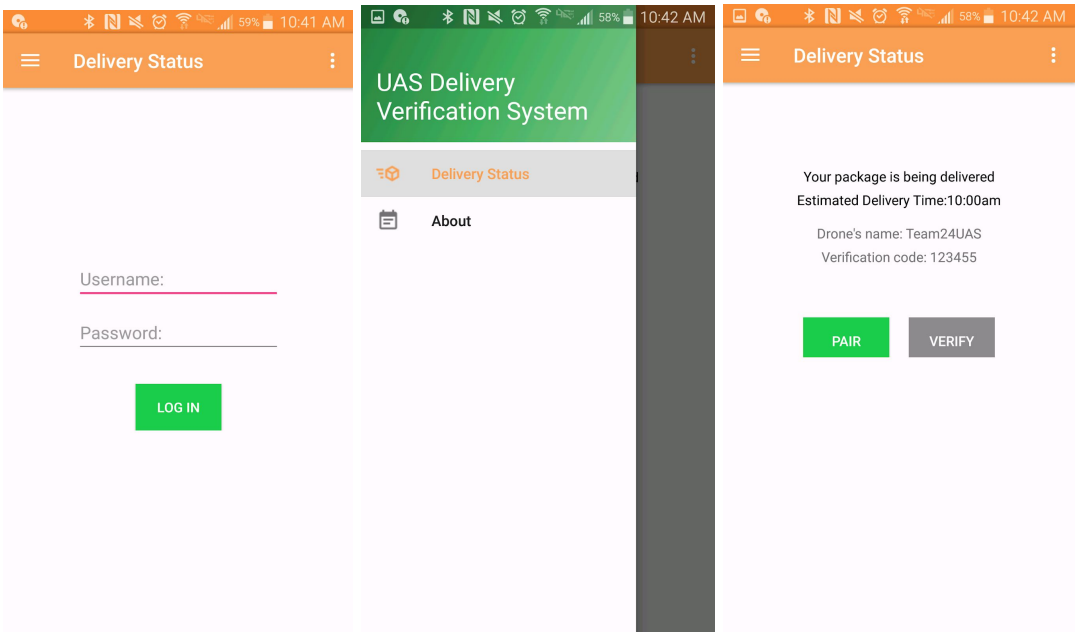

Figure 4.2.5 [Kirill Kultinov] Figure 4.2.6 [Kirill Kultinov] Figure 4.2.1 [Kirill Kultinov]

If a user enters valid credentials then a page of the delivery status will appear on the screen. Here a user can see the current status of the delivery. If items are not being currently delivered, then information about estimated date and time of delivery will be shown. If no orders have been made, then the delivery status will indicate this. If delivery is in progress and everything goes right, then screens will be changed depending on the actual delivery status (Figures 4.2.1 - 4.2.3). If something makes the delivery unavailable, then a notification will be shown on a screen that explains the reason why the delivery cannot be made and asks a user to contact a certain company that is responsible for the delivery. If a user does not verify the delivery on time and the delivering UAS leaves, then the appropriate massage will be displayed in the Delivery Status section. If a delivery is successfully completed, the application will show a user a corresponding message as shown on Figure 4.2.4

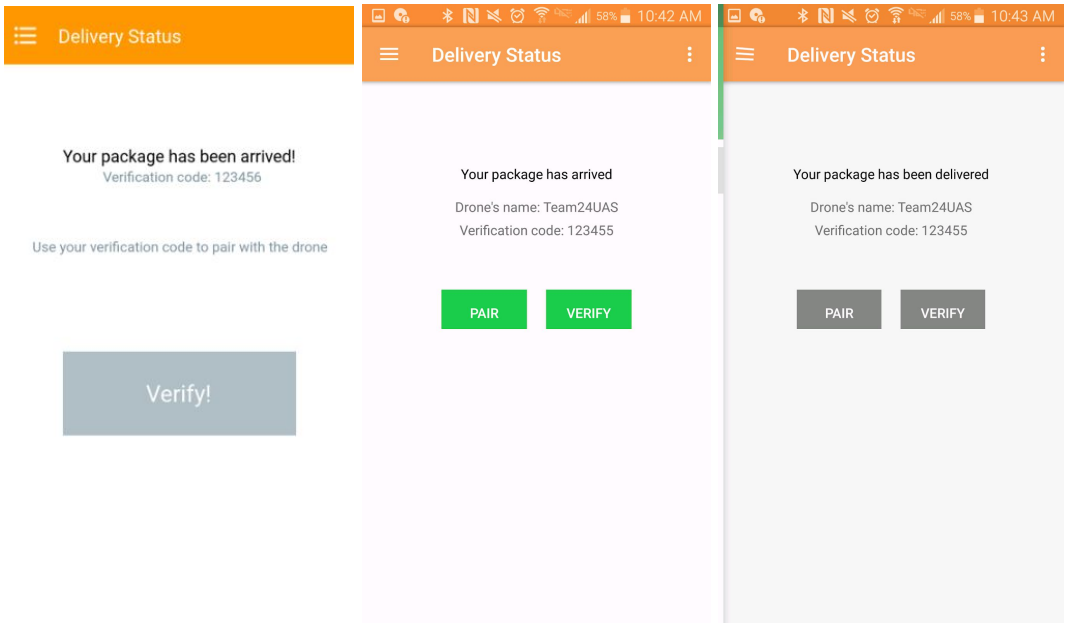

Figure 4.2.2 [Kirill Kultinov] Figure 4.2.3 [Kirill Kultinov] Figure 4.2.4[Kirill Kultinov]

In order to send and receive data we will use built-in Bluetooth frameworks for iOS and Android. Diagram 4.2.1 shows how we will enable an ability to send data from user's smartphone to the BlueSmirf. We will create Bluetooth adapter object required to establish the connection between two devices. After this, Bluetooth adapter receives a MAC address of the BlueSmirf and passed as an argument to a Bluetooth device object. Bluetooth socket is created in order to send and receive data using the created Bluetooth device object. Finally, input and output streams are gotten from the socket and initialized. Having this setup, we can send data via Bluetooth by accessing the output stream of the Bluetooth socket and passing a string we want to send. A separate thread will be created for receiving data from BlueSmirf. The thread will be constantly running until the data is received and listening for incoming data on the input stream

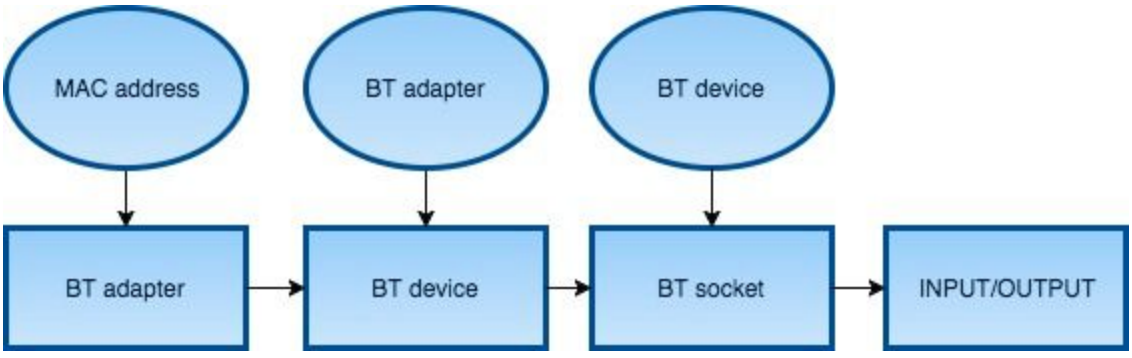

Diagram 4.2.1: Bluetooth connectivity implementation

Diagram 4.2.2 shows how a user can get access to the delivery status. This diagram shows steps and actions that are made by user and how it affects software. Each action performed by a user is represented as a rectangle with corresponding text field. Each action performed by our software is represented as an oval with a corresponding text

field. When a user opens our application, software checks the internal memory of the device in order to determine if there are stored login credentials. If credentials are found then a user can either re-enter the credentials or press login button. If no stored login credentials exist in the device's memory then a user has to enter login information. An access to the delivery status can be either granted or denied after the login button is pressed. If the access is denied then the software tries to retrieve saved login credentials. All steps described above are repeated until the user enters correct login credentials and the access is granted.

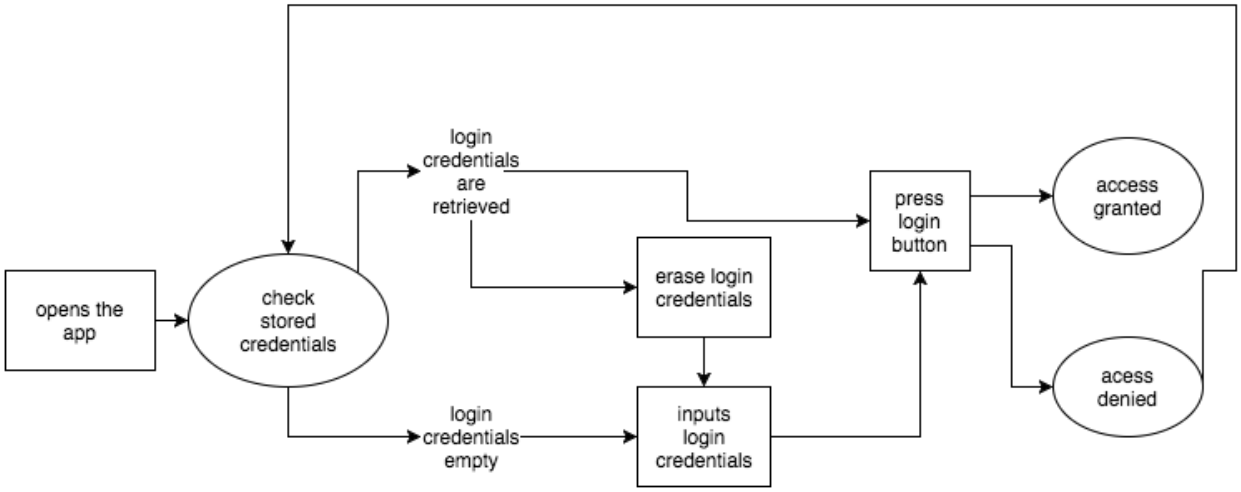

Diagram 4.2.2: A user gets access to the delivery status

Our application has to interact with a company's database in order to display current delivery status and check the user's login credentials. We will be using Firebase real-time database for the simulation. Firebase is free to use and has many advantages. It allows smartphones to connect to a database and update all the information on the screen in real time, whenever the data is changing in the database remotely. Firebase allows the implementation of login credentials' checking and sending push notifications on users' devices [14]. Sample users with basic information will be created. During testing, we will enter a sample information for a user, including login credentials and initial delivery status. The status will be updated over time in order to simulate the complete the delivery process until it is completed. Also, the software interacts with a Bluetooth transceiver installed on a UAS that delivers a package to the user. Diagram 4.2.3 combines states of the delivery process and how each affects a state of our application. All representation conventions are preserved from diagram 4.2.2. Hexagons represent actions made by company. Whenever delivery is cancelled the message is sent to a user and our application displays a corresponding message on the screen of the user's device. The application also receives messages from a company server when a delivery status changes. When a UAS arrives at the destination point of the delivery and waits for confirmation, a user enters the verification code. The package is dropped off after the delivery is verified.

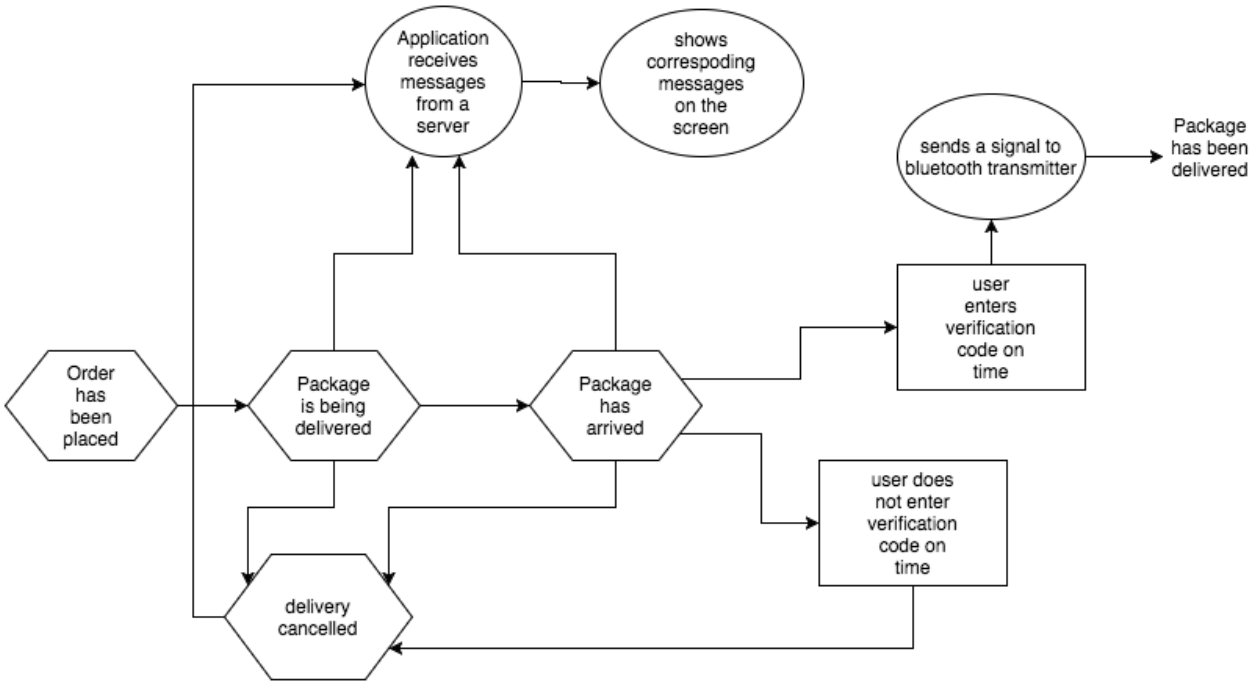

Diagram 4.2.3: Software interaction with a company's server and **UAS** depending on delivery status

As described above, our application will retrieve login credentials from the device's internal memory. This feature will allow users to avoid entering login and password every time the application is launched. This can be implemented by using Core Data on iOS. Core Data allows developers to store data on the iOS device in order to retrieve it inside an application at any time. It can be done by creating an object called ManagedObjectContext and invoking FetchRequest specifying an object that we want to retrieve. In our project, it is going to be a customer object. A similar technique will be used in order to retrieve login credentials inside Android application. However, cache will be used instead because Android OS does not have Core Data feature. Users will be able to prevent saving their login credentials in our application. This feature is implemented to increase the security of user's' login credentials.

5 Project Results

5.1 Accomplished

Through our testing above we were able to verify that our finished project functioned as we had expected when designing it. On the hardware side, all requirements were met. On the software side, we were able to fully complete and integrate our Android application into our overall system. To see fully what was accomplished please refer to **Volume 4, Test Plan/Test Results For UAS Delivery Verification**.

5.2 Project Alterations

Throughout the process of completing our project, there were alterations which were made from our original design. The original design for our application called for the user to manually open their Bluetooth settings, outside of our application. We were able to include a button that allowed the user to access their Bluetooth settings directly from our application. We originally intended on using only one LED to signify that verification had been received, but decided to add two additional LEDs to make it easier for the user to see.

5.3 Project Shortcomings

Unfortunately, we were not able to complete every aspect of our design. We were unable to get our iOS application to discover our BlueSMiRF Silver module. This led to us not being able to communicate, via our iOS application, with our hardware components. Given more time we do feel, however, that we could have completed this task. With our time constraints and not having demoed our project with our Android application we did not want to do anything that might have prevented our Android application from pairing with our BlueSMiRF silver.

We were not able to implement login credential storage in our applications for both iOS and Android due to time limitations. Also, we could not implement notifications for our applications because of the difficulty of the Firebase and time limitations. We could send notifications manually from Firebase when the package arrives. However, we decided it would not be satisfiable for our project. In addition to that, a server in the middle between Firebase and smartphone is required in order to send notifications automatically. Due to time limitations we could not implement the server and add Firebase functionality.

6 Appendix: Résumés of Team Members

The following pages present one page resumes of all of the team members.

KIRILL A. KULTINOV

EDUCATION

Bachelor's of Science in Computer Science **May 2017**
Wright State University

WORK EXPERIENCE

Software Developer **2015-Present**
K&K Cinema, Rybinsk, Russia

- Designed and developed application for Android & iOS devices
- Modify the functionality and the interface of the existing application

Programming Assistant **2014-Present**
Computer Science Help Room, Wright State University

- Provide help to students who struggle or have questions about CS projects and lab assignments

SKILLS

- Java
- Swift
- Visual Basic
- Objective-C
- C++
- Oracle SQL
- Python
- Reverse engineering
- Assembly

RELEVANT COURSEWORK

- Introduction to Java Programming I & II
- Networking
- Cyber Network Security
- Linear Algebra
- Introduction to Robotics

- Data Structures and Algorithms in C++
- Comparative Languages
- Technical Writing for Engineers
- Computer Organization
- OS Internals and Design
- Calculus I & II
- Logic for Computer Scientists

NATHAN D. NICHOLS

EDUCATION

Bachelor of Science in Electrical Engineering **05/17**
Wright State University
Dayton, Ohio
Associate in Science (A.S.) **05/13**
Ohio University
Chillicothe, Ohio

WORK EXPERIENCE

Tutor **2015-2015**
Wright State University

- Tutoring students in mostly engineering-related courses

SKILLS

- Matlab
- Multisim 12.0
- COMSOL
- Microsoft Word
- Excel
- Power Point

RELEVANT COURSEWORK

- Introduction to Electromagnetics
- Digital Communication
- Linear Systems 1 &2

JUSTIN M. RUTSCHILLING

EDUCATION

B.S., Electrical Engineering **April 2017**
Wright State University
Dayton, Ohio

WORK EXPERIENCE

Electrical Engineering Co-op Operations **May 2016-**
Midmark Corporation **August 2016**

- Designed a comparator circuit to verify proper output voltage of a wall mount transformer
- Ran diagnostic work on a circuit board to determine why it was failing testing as a part of a team
- Reverse engineered a circuit to properly document it after it was already built
- Utilized soldering skills to solder a comparator circuit that was previously designed

Electrical Engineering Co-op Sustaining **August 2015-**
Midmark Corporation **December 2015**

- Properly documented a test fixture that tested circuit boards
- Ran testing and properly documented that testing
- Gained experience using 3-d modeling tools

- Performed troubleshooting techniques to fix a test program

Electrical Engineering Co-op New Product Development **August 2014-**
Midmark Corporation **December 2014**

- Attained vital experience using AutoCAD to layout circuit designs
- Designed circuits to test upholstery heaters
- Partook in continuous improvement events

SKILLS

- AutoCAD experience
- Very knowledgeable in Microsoft office tools
- MatLab experience
- Strong interpersonal skills
- Control system design knowledge
- Highly dependable

RELEVANT COURSEWORK

- Digital/Continuous Control Design
- Intro to C Programming
- Technical Communication
- Intro to Electromagnetics
- Digital Integrated Circuit Design with PLDs and FPGAs
- Digital Communication

HONORS AND AWARDS
Dean's List **May 2016**
Wright State University

CARL W. SCHMDIT IV

EDUCATION

Bachelor's of Science in Electrical Engineering **April 2017**
Wright State University, Dayton, Ohio

WORK EXPERIENCE

Server and Bartender **January**
Logan's Roadhouse**.** Beavercreek, Ohio **2008-September 2016**

- Ensured guest satisfaction
- Took and placed orders
- Prepared drinks for entire restaurant
- Maintained meal schedule through proper timing of salads, refills, cleaning, etc.
- Processed payment at end of meal
- Managed food-out window

SKILLS

- Microsoft Word
- Microsoft Excel
- Microsoft PowerPoint
- MATLAB
- C Programming
- HDL Coding
- Supervision & Training
- Time Management
- Interpersonal Communication

RELEVANT COURSEWORK

- Technical Communications
- Linear Systems
- Non-Linear Devices
- Electronic Integrated Systems
- C Programming
- HDL Programming
- Digital Integrated Circuit Design