Requirements Specification Document

Version 1.2

Team Members: Joseph Griffith, Robert McIntosh, Brandon Samz, Corban Stevens

Project Sponsor: Gary Matsch and Harlan Mitchell

Faculty Mentor: Austin Sanders

Created: 11/20/17

Revised: 12/7/17

Accepted as baseline requirements for the project:

Client: _____ Date: _____

Team: _____ Date: _____

Table of Contents

1. Introduction

Every day, over one hundred-thousand flights are scheduled across the globe. With so many flights occurring everyday accidents are inevitable. In 2016 alone, there were sixty five accidents on commercial airlines resulting in ten deaths. Accidents can happen for a number of reasons, most of which are out of the control of aircraft operators and engineers. However, it is the responsibility of the aircraft engineers and operators to minimize the risk of failure as much as possible. Our concern focuses on the needs of the engineer and any engineer, will tell you the best way to see a problem in your system before it happens is to collect ample operation data.

Technology today is data driven, and, as software developers, data is paramount to building and maintaining successful solutions. Our team (BlueSky Group) has been tasked with assisting our client in their goal of obtaining more data. Gary Matsch and Harlan Mitchell with Honeywell Aerospace develop turbine engines and engine control systems for a myriad of private jets. These engines and their connected systems generate data every flight, data that is paramount to the reliability of their product. Currently, this data is downloaded from a computer that monitors the engine's performance through a wired connection. Honeywell Aerospace technicians periodically connect to this Engine Control Unit (ECU) and retrieve the data as often as they can. However, this process does not happen frequently enough. The cumbersome process of physically connecting to a computer and downloading this data on location greatly limits the amount of flight data that Honeywell can collect. Developing a wireless solution to this problem would allow Honeywell to collect ample data but this solution would not bring immediate profit to their company, in other words there is no business case to solve this problem. This is where our team comes in; our clients, Gary Matsch and Harlan Mitchell, see this cumbersome wired process as an embarrassment to the company in an age when everything is wireless. To assist our client in proving to his superiors the importance of this solution we at BlueSky Group will be developing a prototype that solves this problem.

Our prototype will take the form of a mobile app that uses bluetooth to connect to the Engine Control Units and download their data and process it. Currently Honeywell technicians use an archaic Engine Management System called EEI to process flight data. Once they have downloaded the data from the ECU, this system displays the data in a way that can be analyzed by the technicians. Our prototype will emulate the functions of this system on a mobile platform, allowing for ease of access to this data. Our prototype will allow for a safe and easy way to download flight data on a much larger scale meaning greater reliability for the products Honeywell Aerospace creates.

2. Problem Statement

The current problem that we face is the frequency of collecting all of the data that would be needed to analyze and identify problems before it is too late. Currently, the process of extracting this data from the engine control unit is a very long and cumbersome task. This means that, while aircraft are required to download and review this data at set intervals, those are the only times that this data is being looked at, and clearly this is not often enough. The reason that this process is so cumbersome and slow is because the whole process and program are wildly out of date when compared to modern technology. The current process begins with the technician setting up their laptop and connecting an RS-232 cable to it. They then have to walk into the aircraft's cabin and connect the cable into the plane manually. Once that has been completed, the technician can start the download using EEI software. This software is so old that it can only run on Windows XP. Once the download has been started, the download can take up to thirty minutes to if a download has not been completed recently, and if you are running a commercial aircraft that is basically in the air flying twenty four seven, that may be too much time to spend on the ground not making any money. In summary, the problem that our project seeks to solve is this; in order to prevent engine failure, data needs to be downloaded at frequent intervals, and, in its current state, the data is not being downloaded enough. This is because the way that data is collected is cumbersome and slow which is not an option for commercial airlines, and our solution will greatly improve upon this problem.

3. Solution Vision

Our primary goal with this project is to bring Honeywell's data acquisition methods into the 21st century. To do so we will be implementing a mobile application with Bluetooth capability to access their flight data. The mobile application will emulate the functionality of Honeywell's current engine monitoring software, EEI, while displaying the flight data in a way that is fitting for a mobile application. The final piece of our puzzle will be to emulate the engine control unit itself using Linux Virtualbox. This solution will afford Honeywell a certain level of efficiency through some key features;

- The ability to access flight data without a wired connection
- The ability to transmit this data to Honeywell immediately
- The ability to access flight data anywhere in the world regardless of internet connectivity
- The ability to download all flight data kept by the ECU
- The ability to download flight data with only minor intrusion in the aircraft's cabin
- The ability to view and process flight data mobily

In this way and with these features we can ensure that Honeywell will collect ample flight data. As mentioned before the flight data itself is created by the turbine engine's control unit. It will be our mission to access this data wirelessly and present it in a way that can be used by technicians. Honeywell's current software allows the technician to see warnings on faulty parts, failures in non-critical systems otherwise unseen to the pilot, and various graphical data on engine performance. Our solution will do just the same while adhering to the restrictions of a mobile platform. For the mobile platform itself we have elected to use Android Studio as it is the platform our team has the most experience with and direct access to. We considered using IOS for this project but quickly realized we did not have the means of developing in this platform. However, we do not see this as a major issue as Android Studio will meet all of our mobile platform requirements. Of course our mobile application will rely heavily on the data it is fed. Unfortunately for our team however, we will not be given direct access to an actual turbine engine and its ECU. So for the purpose of testing our system, we must recreate this system in a way that is true to the original function. Our plan here is to use a Linux Virtualbox as the ECU used on Honeywell turbines is a Linux machine. Furthermore, we must create our own flight data as we will not be provided any real data from Honeywell. We see this as our greatest challenge moving forward but plan to mitigate the risks by studying EEI (Honeywell's engine monitoring software) to understand what information is required by it. We originally planned to emulate the engine's control unit using a Raspberry Pi. While this solution is feasible it is not very practical in that it would require

a great deal of time to merely set up Bluetooth functionality. Given the time our team may pursue this avenue as it would add a level of complexity to our demonstrations but for the time being we have elected to focus on achieving full system functionality first. Now when it comes to the wireless connectivity we briefly considered Wifi but quickly chose Bluetooth for a number of reasons. First it inherently includes a level of security as the devices must be within range and consent to creating a connection. Second Bluetooth allows connection to be made anywhere in the world regardless of internet connectivity.

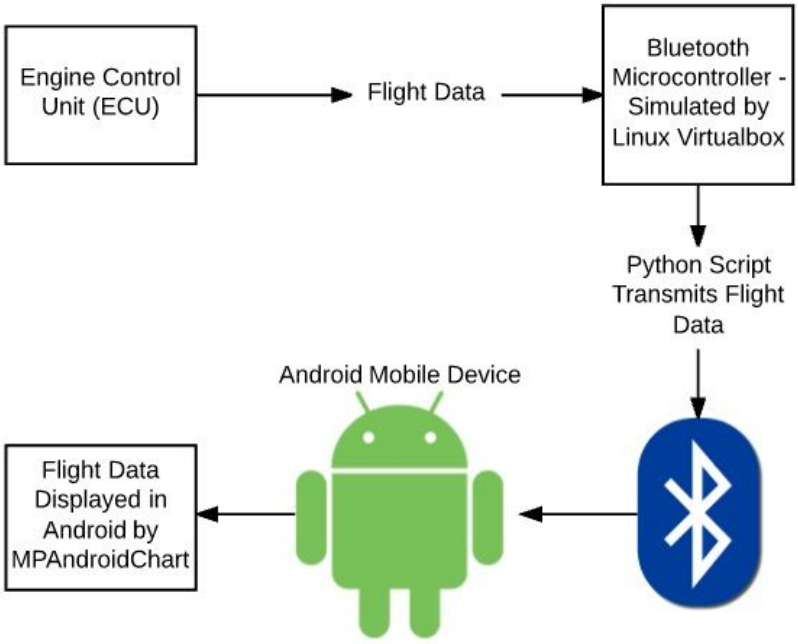

Figure 1: *Overview of System Functionality*

Our system could be expanded in many ways that would benefit the company as well. For instance the app could theoretically be made to monitor a live feed of flight data. The app could also be expanded to automatically download and deliver data to Honeywell on certain intervals or upon arrival at an airport. In summary our system will provide Honeywell ease of access to their flight data. This access will ensure greater reliability of their systems by ultimately allowing for data to be downloaded more often.

4.1 Functional Requirements

In this section, we will outline the functionality necessary to effectively meet our client's needs. All of the functional, performance-based, and environmental requirements will be expanded on and relate to the following set of top level domain requirements.

1. *Top Level Domain Requirement:* Engine download application is accessible to users with a smartphone or tablet.
    1.1. *Accessible with a smartphone:* Most users of the engine download application will be using a smartphone. Smartphones are much smaller than tablets, so less detailed data will be displayed.
        1.1.1. The application should have a navigation menu so that different screens can be easily navigated to.
        1.1.2. The application should support scrolling so that it can be navigated easily.
    1.2. *Accessible with a tablet:* Tablets allow the user to see more of the data without having to scroll or go to another window. They are also very helpful when the technician has to view graphs in detail or manipulate something on a densely crowded screen.
        1.2.1. The application should scale and take advantage of increased screen size on a tablet.
        1.2.2. The application should have an expanded view of the graphs so that greater detail is displayed.
2. *Top Level Domain Requirement:* Engine data can be downloaded any time or place the plane has landed, with only a smartphone or tablet running the engine download application.
    2.1. Application should be able to download flight data regardless of internet connection.
        2.1.1. Download should be possible as long as the user is within 10 meters of the aircraft.
        2.1.2. Application should list available engine control units to download data from.
            2.1.2.1. Application should keep track of recently accessed engine control units.
            2.1.2.2. Application should list the engine control units in a formatted string of numbers, which are randomly generated and assigned to each ECU.

3. *Top Level Domain Requirement:* Application should be able to download all of the data from the ECU.
    3.1. Application should display progress of the download.
        3.1.1. Application should verify when a download is completed.
        3.1.2. If a download fails to complete the user should be notified.
            3.1.2.1. If download fails to complete on the first run user should be able to restart the download.
    3.2. Application should be capable of storing all of the engine's flight data.
        3.2.1. If device does not have enough available storage to complete download the user should be notified.
    3.3. Application should display the file size of the ECU's current available flight data.
4. *Top Level Domain Requirement:* Application should allow for review of engine data, with functionality similar to EEI.
    4.1. The application should function similarly to the EEI software provided to the group.
        4.1.1. The application should have engine data that includes the engine serial number, the aircraft id, the download date and time, engine position, and the operator who performed the download.
        4.1.2. The application should include engine maintenance data that includes the number of maintenance conditions, number of exceedances, number of events, number of chips, and the number of impending oil filter bypasses.
        4.1.3. The application should include statistics on the engine and the ECU.
            4.1.3.1. Information regarding engine usage and engine landing should be included.
        4.1.4. The application should show data regarding different systems for aircraft takeoff, climb, and cruise.
        4.1.5. The application should show a list of faults that occur with a timestamp so that the technicians will know when they occurred.
5. *Top Level Domain Requirement:* The application will be able to establish a secure connection with the laptop for testing using Bluetooth. In order to achieve this both the laptop and the application must abide by protocols established by Bluetooth. The laptop will act as a server that will provide the data that has been gathered from the ECU, and the the application will act as the client which is going to be receiving the information sent by the server.

5.1. *Establishing connection as a server:* Our laptop will operate as the server for our Bluetooth connection. In order to do this first it will have to maintain an open Bluetooth server socket. This server socket will listen for incoming connection requests, and provide a connected Bluetooth socket after a request is accepted. When the Bluetooth socket is acquired from the server socket the server socket will then be discarded in order to make sure that our laptop will not accept any more connections.

5.1.1. In order to get a Bluetooth server socket our application will call the method listenUsingRfcommWithServiceRecord(). This will automatically write a new service discovery protocol database entry on the device. The UUID of the app will also be included in the entry which establishes the basis for the Bluetooth communication

5.1.2. To start listening for connections, our program will call accept() on the server socket

5.1.3. Our program will then call close() in order to close the connection to make sure that other devices cannot join the same connection.

5.2. *Establishing connection as a client:* The application will operate as the client in this project. The application will initiate a connection with the laptop that is accepting an open connection on an open server socket. Our application will create a Bluetooth device object that represents the remote device. It will then use that Bluetooth device to acquire a Bluetooth socket and initiate the connection.

5.2.1. Our application will initialize a Bluetooth socket object that will allow our application to connect to a Bluetooth device. The UUID that is given in this creation must match the UUID that is used by the server or else the connection will not be allowed. This will be made sure because the UUID value will be hard coded into the application

5.2.2. Our application will then establish a connection by calling connect() which will initiate a lookup to find a remote device with a matching UUID. if the lookup is successful then the connection is created.

5.2.3. If the method times out then it will throw an IOException which we will then handle by letting the user know a connection could not be completed.

4.2 Performance Requirements

This section details the performance requirements for this project. These performance requirements specify how the functional requirements are expected to perform in aspects such as speed, accuracy, etc.

1. Application downloads data with zero data loss or corruption.
    1.1. It is important that all engine data received is accurate so that the application does not misinterpret or fail to report important data.
        1.1.1. Bluetooth connections manage data using packets similar to TCP. As such, the underlying Bluetooth protocol handles this requirement by ensuring that each packet is received and is not corrupt, and resends the packet if not. However, during the test phase, this requirement may also be tested by manually hashing the download file on each end and comparing the hash.
2. Application downloads data within 1 minute.
    2.1. One of the main focuses of this application is to speed up the data download process. As such, we want the application to download the data as fast as possible, while still providing some leeway due to unavoidable connection issues.
        2.1.1. This time was determined using the 5 MB max file size of the data to be downloaded and average download speeds of Bluetooth 3 or 4. Verifying this requirement is met should involve averaging download times from multiple downloads that minimize outside factors (signal interference, phone distance from microcontroller, etc).
3. Application reports no incorrect data.
    3.1. As previously mentioned, it is important that the application is highly accurate with data to ensure that issues aren’t mistakenly diagnosed or missed, as this could result in major engine failure or unnecessary maintenance.
        3.1.1. As download data will be generated for testing purposes, the team should be able to verify that the application properly displays data by manually parsing it and comparing with what is displayed in the application. This requirement is verified by ensuring that all data in the testing process is correctly reported.

4. Application will establish a connection between the laptop and the device in under 15 seconds.
    4.1. In order to have an application that is focused on speeding up the data recovery from the ECU, speed is a key factor in everything that this application needs to do. When it comes to establishing the Bluetooth connection, the application should be able to establish the connection within 15 seconds.
        4.1.1. The application will be able to establish a connection within 15 seconds because we will be operating our Bluetooth connection in the way that is directed by the Bluetooth documentation. This will ensure that the application is establishing a connection as soon as possible because it is operating in the intended way.
        4.1.2. The application will also be able to establish a Bluetooth connection in under 15 seconds because the application will be able to pair with the laptop because the laptop will be looking for a Bluetooth socket which can only be provided by our application. This Bluetooth socket will be initiated once the application is prompted to connect to a device. This means that the devices should automatically pair once they have been once they have been initially paired with each other.

4.3 Environmental Requirements

This section details the environmental requirements for this project. These are requirements that are imposed upon the project due having been specifically asked for by the client or as a result of chosen technologies.

1. The application will run on Android OS.
    1.1. As specified by the project, the application must be mobile. This means that it must be able to run on a mobile phone. As not all members of the team have Mac computers, and thus are not able to write iOS applications, we have decided to write the application for Android OS.
2. Application should be able to run on all Android devices.
    2.1. Application should transfer user interface seamlessly between devices
        2.1.1. Application interface should be scalable between tablets and mobile smart phones.
    2.2. Application interface should be able to display in both portrait and landscape mode.
3. As specified by our project sponsor the application will be able to mimic the basic functions of EEI in a different layout that is newer than what is currently being used.
    3.1. The application will mimic the functionality of EEI. From our analysis of the current EEI, the most important feature that we need to implement is the functionality to analyze the data and identify if there are any immediate faults that need to be recognized. For example, if a sensor has failed and is in need of repair the EEI will be able to recognize that and report that to the user.
        3.1.1. The EEI is able to identify when a sensor has failed and is in need of repair because, if a sensor is broken, then it will not be reporting data. When this is happening, it either means that the sensor or its connection needs to be repaired or that it is just broken altogether. The application will be able to identify these conditions and report them back to the user.
    3.2. The application will be able to display the ECU data to the user.
        3.2.1. The application will display data to the user through use of a library with the ability to display the data in a graphical format that will be appealing to the user.
4. At least 5 dummy engine data files should be created and be able to be downloaded by the application.
    4.1. The file type must be consistent across these 5 files.
    4.2. The fields and units of measure should be consistent with those found in an actual .dld file.

5. Potential Risks

Currently, the team has identified three major risks that are important to consider while implementing our application. These three risks are Bluetooth connectivity issues, misinterpreted data, and long download times. These risks have been deemed important due to their effects on the function of the plane and its diagnostics as well as possible maintenance costs to the owner.

The first major risk to consider is Bluetooth connectivity issues. This could be caused by having too much interference with the signal or possible errors with the code and device that could prevent Bluetooth connection. In either of these cases, the result would be no connection to the ECU at all, preventing flight data from being downloaded. This issue has a medium severity, as the result would be either having to reschedule the data download or having the data download take longer than expected. There is low risk for this issue, however, as newer standards of Bluetooth and well written code make it extremely likely that there will not be connectivity issues. Mitigation for this issue would need to be implemented in the form of a secondary connection option. This could look like either continuing to equip the plane with the current serial connection as a backup and using the current EEI solution, or providing functionality within the application to function using a wired serial connection (USB). This second option would, however, impose more requirements on the system than currently listed.

Another risk is that of the application having programming errors that cause data from the download to be interpreted incorrectly. This could result in issues with the engine not being diagnosed properly, which could ultimately result in a malfunction leading to even higher repair costs or possibly engine failure/plane crashes. Another result is the possibility of the application to report that something is wrong with a specific component, when there is actually no issue. Although this won't mask an actual issue, this false reporting could cause unnecessary time and money to be spent, as the part in question may be replaced when it doesn't need to be. The best mitigation strategy for these types of errors is extensive testing of the application, specifically the module that parses and interprets received data. This will help ensure that the application can handle a large variety of data reliably.

Although seemingly minor, there is also a risk that the data download takes a longer time than expected. There are multiple factors that could contribute to a longer download time, such as larger data sizes, Bluetooth interference, and older hardware with slower Bluetooth adapters. The direct impact of longer download times is more time being wasted. The severity of this issue is low, as, although it may cause delays in the plane being ready for use again or extra maintenance being needed to be scheduled, the likelihood of this issue occurring is also low as long as downloads are occurring as frequently as expected. As such, it is important that the application is as efficient as possible to limit download times. If the application is as efficient as possible, then factors contributing to slowdown will be minimized.

6. Project Plan

Having researched feasibility and laid out requirements, it is important to have a schedule and plan moving forward. As there are multiple components required in the application to be built, the team has identified key components of the system and prioritized those over others. The plan, presented below, serves as an outline of tasks that have already been accomplished and tasks that will be accomplished in the future.

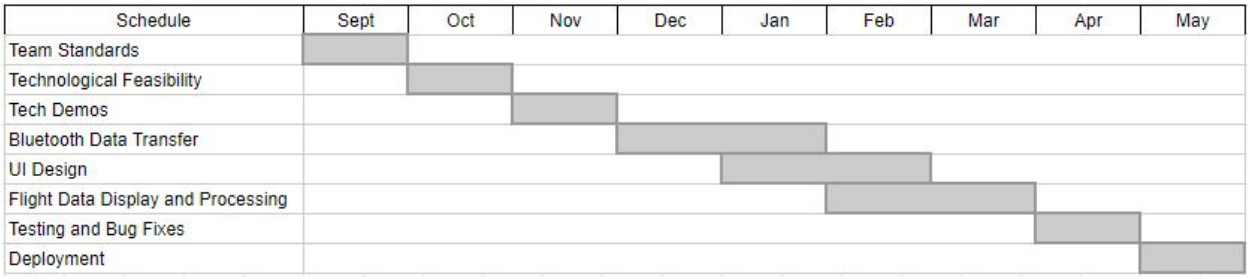

Figure 2: *Schedule of Operations Sept 2017 - May 2018*

Technological Feasibility (10/1 - 10/31)
The first task that relates to the requirements outlined by our client is the Technological Feasibility. In this step, our team researched if certain parts of our solution vision are actually viable to take the time to design and implement. Figuring this out during the beginning of the project will prevent us from finding out later that some of our design decisions will not work the way we planned for them to. This phase of the project was completed in late October.

Tech Demo's (11/1 - 11/30)
The second portion of the project will be the Technical Demos for the various technologies that we will be using. This part will familiarize the team with how these technologies work. This will also provide the team with examples of how the different parts of the product will work. This phase of the project will be executed during the month of November.

Bluetooth Data Transfer (12/1 - 1/31)
The next portion of the project will deal with the Bluetooth Data Transfer. This is dealing with how we will transfer data through a Bluetooth connection from one device to another. Being able to transfer the data is a very crucial requirement that our client has given us. The whole point of this project is to allow for the engine flight data to be sent wirelessly instead of having a wired connection. This phase of the project will begin at the start December and last until the end of January.

User Interface Design (1/1 - 2/28)
This UI Design is the next portion of the project that will be worked on after the team figures out how to send our data through a Bluetooth connection. This portion is important because it will allow our team to create an easily navigable application. This will allow our users to not be frustrated when they need to use it. This phase of the project will start at the beginning of January and finish at the end of February.

Flight Data Display and Processing (2/1 - 3/31)
After the completion of our UI design, our team will then work on the Flight Data Display and Processing. Our application has to display the flight data that it has received and display it in a way that the technicians be able to easily understand so that they know how the plane is performing. This is the area where we will accomplish this. This portion of the project will start at the beginning of February and be completed by the end of March.

Testing and Bug Fixing (4/1 - 4/30)
Testing and Bug Fixing will be the next major part of the project. Since our application will be used to create a business case to allow the wireless transfer of flight data so that airplanes are checked more often, we want or application to work when it is tested and presented. This is why finding and fixing any bugs that may occur is very important. This portion will start at the beginning at the beginning of April and finish at the end of April.

Deployment (5/1 - 5/31)
The final portion of the project will be the Deployment phase. This is when we will give our client the finished project so that they can get approval to switch their data transfer method to a wireless system instead of the wired one that they are currently using. This phase will begin during May.

7. Conclusion

Problems with aircraft engines can be fatal, and the best way to ensure that the engine is properly functioning is to review data as often as possible. Our team, BlueSky Group, is working with Gary Matsch and Harlan Mitchell of Honeywell to solve the problem of lack of data downloads and complicated wired connections. The current process for downloading and reviewing engine data is outdated and time-consuming, which make for downloads not happening very often and Honeywell not getting as much data as they would like. BlueSky Group's solution to this is to create a mobile application that will connect to the ECU and download engine data wirelessly, via Bluetooth. This application will allow the technician to view the received data on the spot and identify any potential issues. This solution will make engine downloads a simple process which uses tools most technicians already carry around with them, eliminating the use of a bulky laptop attached to long wires.

This document serves to identify key requirements of the system to be built. With necessary functionality and requirements in place, the team will be able to focus on building software that meets these requirements instead of trying to build software with a constantly evolving idea of what needs to be done. This will allow for a consistent and coherent application that will function well and properly meet the needs of the client.

Before the team was able to build these requirements, it was necessary to research various technologies to determine any possible limitations, issues, or risks. As such, the team is well on its way to demonstrating that these technologies will work well together to meet the application requirements. Furthermore, we are confident that we will be able to then expand upon these technology demos to implement the fully functioning application.