# A L I S

## Aerial Land Inspection System

Vermeer Corporation

## Design Document

Team Members

Brian Gillenwater

Nathan Kent

Quinn Murphy

Bryce Poellet

Jonathan Schlueter

Revision: 2

Date: 12/13/2015

## Table of Contents

## List of Figures

## Project Description

### Project Statement

The Aerial Land Inspection System (ALIS) is a solution to remotely map the terrain of a future work site from the air. This terrain map will allow machinery operators to view a future worksite and identify potential geographical problems before arrival. This document introduces the proposed design of ALIS developed by the team.

### Deliverables

The system will include a quadcopter equipped with a high resolution camera, a powerful Windows PC, and an Android device used for quadcopter control. The quadcopter will capture aerial photos and transfer them to the PC application. The system will then use photogrammetry techniques to generate a high-resolution terrain model. This model will then be sent to a virtual reality system allowing people to examine the worksite without needing to travel there. In summary:

- Quadcopter capable of autonomous flight and image capture
- PC application capable of generating/communicating a flightpath and receiving images
- A high-resolution model generated from aerial photography
- A game engine project including this model, viewable in a head-mounted display

## System Level Design

### System Requirements

#### Functional Requirements

- Sustained flight in adverse weather
- At least 20 minutes of flight time
- Fly up to ½ mile away from controller
- Take 70 or more images with more than 50% overlap
- Model generated in less than 6 hours
- Model is viewable in a virtual reality platform

#### Non Functional Requirements

- User interaction with the system shall be simple and error free.
- A clear and accurate model will be generated, regardless of terrain.
- The device shall work in clear and windy environments (within reason).
- The system shall not crash or become unstable during flight.
- The model will be viewable in a virtual reality platform.
- The system will be safe to operate, and include emergency stop or halt controls.

## Functional Decomposition

The system is started by the user launching the PC application and selecting a location to be mapped. Parameters can then be set that control the area's size and shape, as well as quadcopter altitude. The user then builds a map, which will generate a set of waypoints that will appear on the built-in map interface. Once the user is happy with the generated flight path, the user will select 'Upload', which will search for any connected Android devices. After selecting a device and confirming, the flight plan will be uploaded to the Android platform wirelessly.

After receiving the waypoint data, the Android application will transform the flight path into a format the quadcopter can understand, and pass this information to the quadcopter. The quadcopter will execute the requested flight, taking pictures along its route. Once it returns, images will be transmitted to the PC application.

After receiving all images, the PC application will pass these images to the photogrammetry software. This software will take several hours to run, and will output an .obj model file. Once the model is generated, the model will be imported into a game engine, and a project will be generated that supports the use of a head-mounted display. See Figure 1 below for a pictorial description of the overall system. Figure 2 offers a more in-depth look at the system.

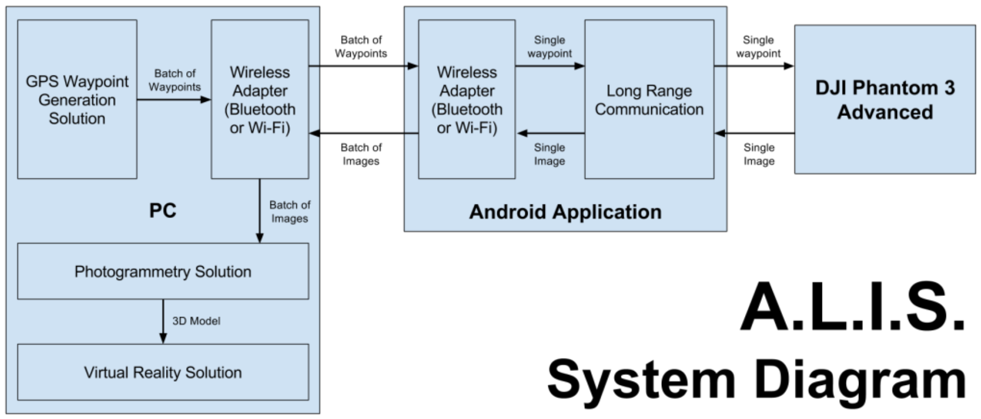


*Figure 1: Functional System Diagram*

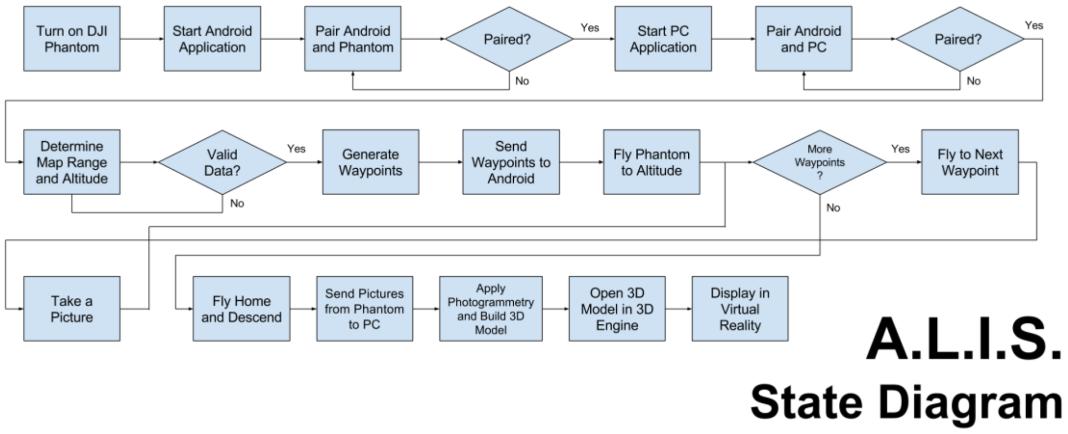
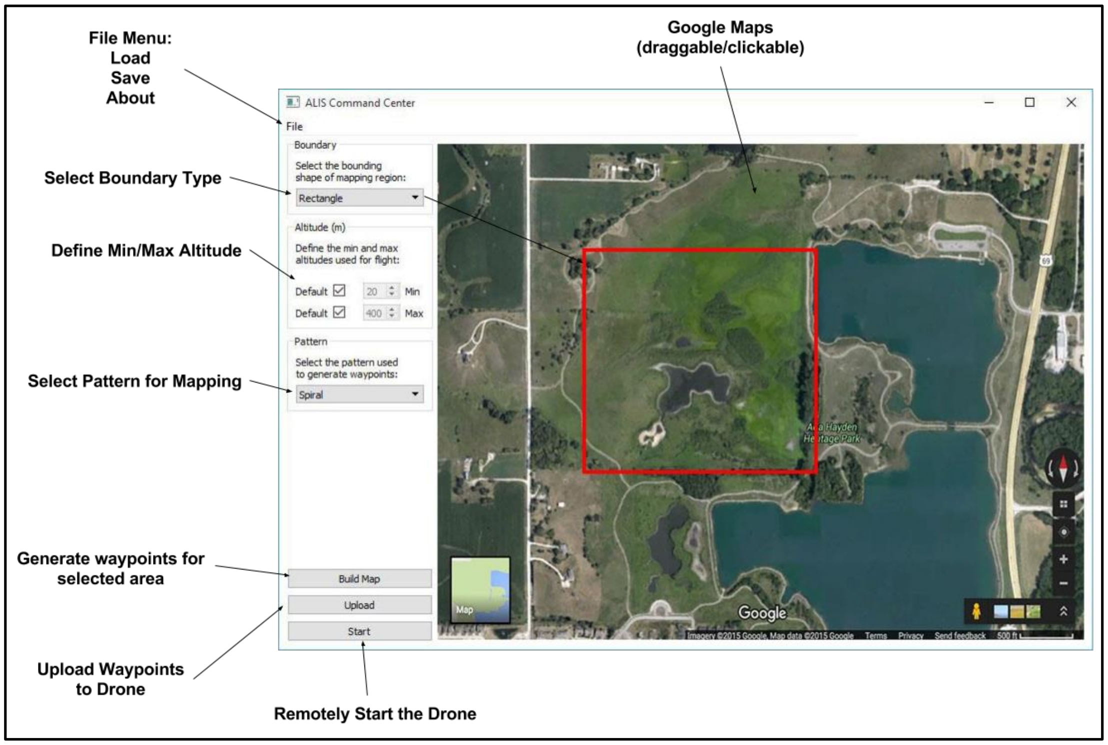


*Figure 2: State Diagram*

## Screen Sketches

### Windows

Figure 3 shows the main page users are presented with when launching the Windows PC application. Parameters are listed in the sidebar, and a drop-down menu provides system control commands such as loading and saving a generated map.



Boundary
Select the bounding shape of mapping region:
Rectangle

Altitude (m)
Define the min and max altitudes used for flight:
Default ☑ 20 Min
Default ☑ 400 Max

Pattern
Select the pattern used to generate waypoints:
Spiral

*Figure 3: Windows Application Main Window*

Figure 4 below describes the window users are presented with when they click 'Upload' on the main window. This allows users to select a connected Android device, and upload the flight plan over Wifi or Bluetooth.

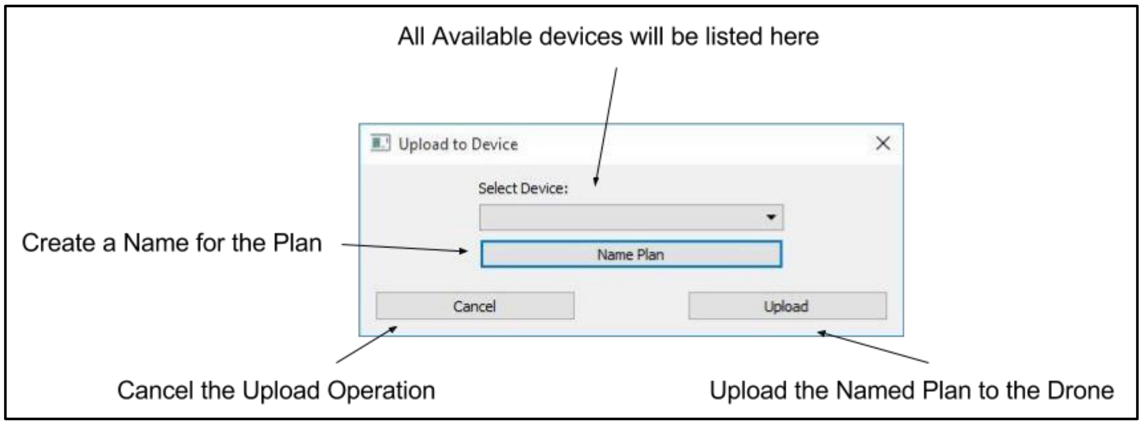

All Available devices will be listed here
Upload to Device
Select Device:
Create a Name for the Plan
Name Plan
Cancel
Upload
Cancel the Upload Operation
Upload the Named Plan to the Drone

*Figure 4: Windows Application Device Upload*

## Android

Figure 5 shows the first screen that users who use ALIS LITE will see. On the top right are different functions that the user can use before actually starting the flight. From left to right, they allow the user to manually upload a binary file of the waypoints, view those waypoints with Google Maps, change the settings of the application, and calibrate the compass. Pressing start will send the quadcopter off to do its mission.

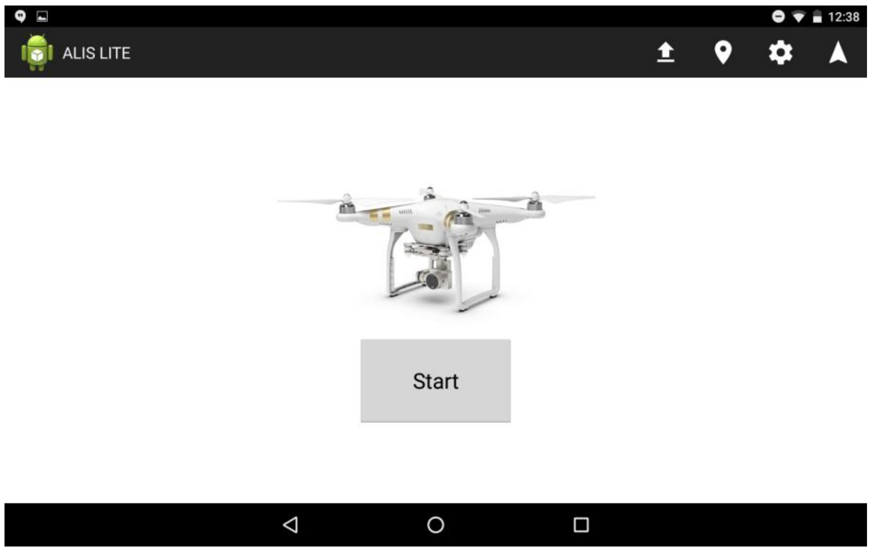
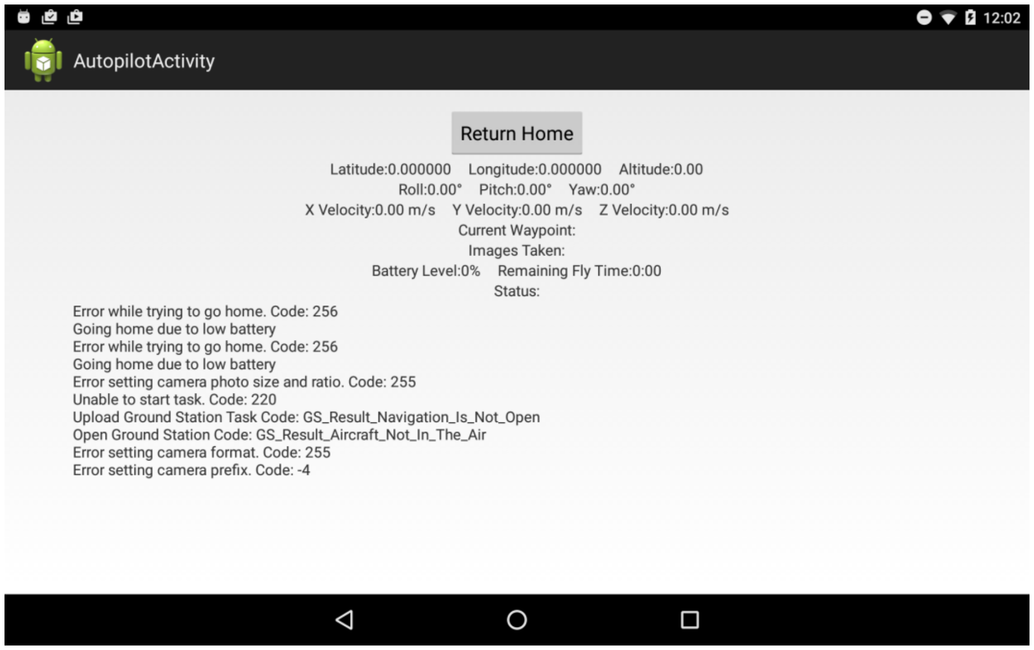


*Figure 5: ALIS LITE Connection Screen*

Figure 6 shows the screen that is shown during a mission. It reports back the status of the quadcopter (location, attitude, battery life, mission status) as well as printing out any status messages like errors or updates in a scrolling list.



Latitude:0.000000 Longitude:0.000000 Altitude:0.00
Roll:0.00° Pitch:0.00° Yaw:0.00°
X Velocity:0.00 m/s Y Velocity:0.00 m/s Z Velocity:0.00 m/s
Current Waypoint:
Images Taken:
Battery Level:0% Remaining Fly Time:0:00
Status:

```
Error while trying to go home. Code: 256
Going home due to low battery
Error while trying to go home. Code: 256
Going home due to low battery
Error setting camera photo size and ratio. Code: 255
Unable to start task. Code: 220
Upload Ground Station Task Code: GS_Result_Navigation_Is_Not_Open
Open Ground Station Code: GS_Result_Aircraft_Not_In_The_Air
Error setting camera format. Code: 255
Error setting camera prefix. Code: -4
```

*Figure 6: ALIS LITE Status Screen*

## Detailed Description

### I/O Specification

The Windows application and Android application will communicate over a Wi-Fi network (IEEE 802.11 b/g/n) using TCP/IP. The Android application will run as a server, and the Windows application will connect to it and begin the data transfer. In the future, this transfer may be encrypted, but for the moment it is not. The first data sent will be a 256-bit verification key to verify that the client has the proper authority to program a route. This should be the SHA-256 hash of some string/password. The next set of data will be a unique ID for the route that is being sent. This ID is a SHA-256 hash of the route being sent and is used to double check that the route has not changed in between uploading and initiating the run. Next, the number of points that are to be expected is sent in a two byte integer. Finally, the data points are sent in the order that they are to be executed and the connection is closed. If any of the data is out of the valid range, an error message will be returned to the Windows application and the upload of the route will be cancelled. The data shall be sent in big endian order and sent in the format shown in figure 7.

| **Description** | **Byte Offset** | **Data Type** | **Minimum** | **Maximum** |
|---|---|---|---|---|
| Latitude of the Waypoint | 0 | f64 | -90 | 90 |
| Longitude of the Waypoint | 8 | f64 | -180 | 180 |
| Altitude of the Waypoint (meters) | 16 | f32 | 0 | 120 |
| Heading of the Quadcopter (degrees) | 20 | i16 | -180 | 180 |
| Pitch of the Gimbal (degrees) | 22 | i8 | -90 | 30 |

*Figure 7: Data Packet Format*

At this point, the Windows application is set up as a server and waits to hear back from the Android application. Upon completion of this route, the Android application will download the photos from the quadcopter and transmit them back to the Windows application following another exchange of the 256-bit verification key.

## UI Specifications

### Windows Application

The Windows application will be where the majority of interaction and processing happens for ALIS. A user will be able to do the following:

- Waypoint Planning & Generation
  - Set the bounding shape of the region being mapped (rectangle, oval, etc.)
  - Define a min and max flight altitude (can use defaults if unknown)
  - Set the waypoint generation pattern (spiral, grid, etc.)
  - Build a waypoint map using the bounding shape and pattern
  - View the generated map and waypoints with a built-in map utility
  - Name the generated map
  - Save the current map and its waypoints
  - Load a previously saved map
- Communication & Control
  - Select a recognized Android device and upload the current map
  - View status of the map upload to the device
  - Verify a successful map transfer to the device
  - Initiate the generated waypoint route (thru the Android app)
- Other
  - Upon a completed flight, signal the photogrammetry software to generate a model with the retrieved aerial images
  - View basic information about the project such as version number, targeted hardware, and group members

### Android Application

Because the Android application is a thin-client, the user interface will be minimal and mostly comprised of debug information. The user will be able to do the following things:

- Perform an emergency landing of the quadcopter
- Force the quadcopter to return to home base
- Start the quadcopter’s flight sequence
- View the status of the transfer of data points from Windows to Android
- View the status of the transfer of images from the quadcopter to Android
- View the status of the transfer of images from Android to Windows
- View errors associated with any data transfer
- View the waypoints that have been loaded onto the quadcopter
- View the current location, altitude, speed, and attitude of the quadcopter
- View the current battery life of the quadcopter

## Hardware/Software Specifications

- Interfaces from Desktop and Android app will be located in appendix
- Software Libraries and Version Info
    - Windows Application
        - C++ 2011
        - Qt5 Cross-Platform UI Framework v5.5
    - Android Application
        - DJI SDK downloaded 10/26/2015
        - Minimum Android SDK level 16 OS version 4.1
- The PC and Android application need to be on the same network.
    - Need to pick a port to bind to or allow the user to select one.
    - The software will prompt to open a firewall port
- The software will run on Microsoft Windows 7 and later

## Simulations and Modeling

The Android code cannot be easily tested due to the proprietary nature of the DJI Phantom. However, testing constantly with the quadcopter outside is out of the question due to FAA regulations and battery life. To remedy this, we will test the Android application in DJI's Simulation application by connecting our remote control to a PC and running the Android application in that environment. There, it will be easier to identify bugs without needing to drive outside of Ames to test the application.

## Testing Specifications

### Windows Application Testing

The Windows application shall be tested to ensure that the generated flight path for the quadcopter navigates the specified area as needed to obtain the photogrammetry images without breaking any of the flight restrictions. Once a more detailed design of the Windows application construction is complete, these specifications can be expanded to provide testing details for the components of the application.

Testing the Windows Application will involve the following:

- Checking generated flight path for accuracy and user parameter inclusion
- Make sure any generated waypoints fully comply with FAA regulations
- Checking that flight path covers the specified area sufficiently for the photogrammetric reconstruction to be successful
- Checking that the flight path is transmitted to the Android Application in the correct format and with the correct data

### Android Application Testing

The flight code for the quadcopter shall be tested to ensure that the flight path can be received from the Windows application and sent to the quadcopter as waypoints. The transfer of images from the Android application to the Windows application will also be tested.

Testing the Android Application shall involve the following:

- Checking that a correct flight path can be received from the Windows application
- Testing that the data can be successfully used on the quadcopter leading a the flight specified by the Windows application
- Transferring images from Android to Windows

### Photogrammetry Testing

The photogrammetry process will be tested to confirm that the generated model closely matches reality. In addition, the process also must be tested to ensure that it completes smoothly and consistently. These specifications will be updated to match any new information gathered from our tests with the quadcopter.

Testing the Photogrammetry process shall involve the following:

- Loading the pictures into the photogrammetry process
- Converting the pictures into a point-cloud
- Transforming the point-cloud into a textured model usable in Unity
- Import into Unity to confirm that it accurately depicts the area in question

## System Prototyping

### Windows Application Prototyping

The initial development of the Windows application will lead to a prototype that can accept user input to specify the flight plan, generate a sequence of waypoints, and transmit those waypoints to the Android application.  The goal of this prototype will be to demonstrate the base functionality of the system before time is spent to add convenience functionality and the make the user interface intuitive for a new user.

### Android Application Prototyping

There will be no formal prototype of the Android application.  This application is meant to be as thin as possible and is only being used because the quadcopter selected for the project did not support onboard navigation.  Therefore, the application will be developed to bridge the Windows application and quadcopter.  Once the base functionality of this application is complete, there will be no need to add additional features.

### Photogrammetry Prototyping

There will be no true prototype of the Photogrammetry process. Once it is confirmed that the process works, it will simply exist as an automated script. There will be no need to add additional features to the process, especially considering that the Photogrammetry process is simply a series of applications developed by other parties.

## Challenges

### FAA Drone Regulations

The FAA has strict rules regarding the operation of drones. Adding to this complexity is the fact that these regulations are currently undergoing changes. We will need to research these regulations to ensure we do not violate them initially, and then continue to monitor the regulations as the changes are made to ensure that our system always complies with the law.

### Photogrammetry with Snow Cover

The photogrammetry process is likely to be less reliable during the winter months due to the snow obscuring many features. While we are unable to change how well the photogrammetry will work once there is snow on the ground, we can plan ahead knowing this issue will exist. We will plan on getting as many test sets of images as possible before it snows so that we have data to work with during the winter and we cannot gather any more.

### Photography Pattern

We do not know the optimal pattern to take the photos to generate the best model with the photogrammetry software. Taking photos at different angles or altitudes will affect the quality of the final model. We will need to consider and test multiple flight and photography patterns to find the best way to generate the waypoints in order to optimize the quality of the model and ensure that it contains the needed resolution.

## Other Documents

### External Documentation

Qt 5.5

Android API Level 16 SDK

DJI Android Mobile SDK

## Conclusion

The final goal of this project is to design a system that uses a quadcopter to autonomously map a remote worksite and develop a working prototype of the system. We plan to have only as much user-interaction as necessary, and limit that interaction to the Windows application. The Android application servers as a thin client between the Windows application and the quadcopter. The goal is to develop the system so that, in the future, a different quad may be used with the system by only making minor changes. The output of the system will be a three-dimensional model of the worksite that can be imported into a virtual reality system allowing users to examine the worksite without travelling there, as the worksite is potentially in a remote area with limited access.