Test Plan

Drone Swarm – Group 1

Andrew Azores - 1048083

Jazz Kersell – 1041571

Evan Holtrop - 1059591

Darren Kitamura - 0854359

Table of Contents

Index of Tables

Change History

Version	Date	Author	Comments
0	21 Oct 14	AA, JK, EH, DK	Initial commit

Table 1: Change History

Test Factors and Rationale

Factor	Rational
Reliability	For the main application/demo, the UAVs are potentially very dangerous if they lose control, therefore it is extremely critical for this software to be reliable. Should some error occur there must be a recovery mechanism. The algorithm itself will also rely depend upon having a reliable distributed network and reliable connections and communications chanells to its peers.
Continuity of Processing	While this software will not be designed to meet real time requirements there is little room for computation to slow or stop. If a device enters a failure state but fails to notify the rest of the network in a timely fashion, then the system may be operating in an unacceptable state for an extended period of time before the fault is detected.
Correctness	This software is a proof of concept for a research paper, therefore it is essential that the implementation be correct according to the specification. A correct implementation is necessary to adequately demonstrate that the research is sound.

Performance	For the main demo/application, the system must be functional for up to 10 UAVs. It does not need to be able to spontaneously add or remove UAVs from the swarm. The system must be able to monitor constraints that apply to individual UAVs but not global constraints. The performance of the research paper algorithm itself is largely dependent upon the computational complexity of the algorithm – the software should be implemented such that it matches the same computational complexity.

Table 2: Test Factors and Rational

Testing Method

Testing will consist of unit tests and manual testing. Initially the main effort of testing will be unit testing. This is necessary because the system has very little user interaction which makes manual testing somewhat unnatural at lower levels. Unit tests will provide whitebox and blackbox test coverage, and will ensure that individual components are functioning correctly. Manual testing will be conducted mostly at a high level to verify the system as a whole. There will be some lower level manual testing to verify the Device Communication Module. Unit tests will also be written to verify the state monitoring algorithm; mock devices can be created with simulated state transitions, and conditions about the evaluation of the state monitoring algorithm can be tested.

Types of Testing

Manual

Manual testing will be used mostly for high level testing of the whole system. This is due to the fact that UAVs by definition have minimal interaction with users. There are two aspects of testing that can be done manually: whole system testing, and testing of the Device Communication Module. Interim demonstration applications will also be constructed, which may also be used for high-level testing. For example, a demonstration in which users can enter values for variables in a mathematical expression on various devices connected to the distributed network, and the state monitoring algorithm on each device will monitor that certain constraints are always met. For example, a test case could include ensuring that the algorithm correctly asserts failure of a non-negativity constraint when an expression is entered which may in fact become negative.

Whole system tests will consist of loading a set of constraints and corresponding monitor automata onto each drone, and observing the behaviour produced. Testing the Device Communication Module will consist of attempting to initiate communication between multiple (up to 10) Android devices and ensuring that all devices are aware of all other devices and are able to open communications channels to each other device.

Regression

Regression testing allows developers to verify that their changes have not produced new problems in the code. Since these tests should be done frequently it is most efficient to use automated testing for this. This project will use unit tests as our primary means of regression testing.

Functional

Functional testing, also known as black box testing, is used to verify that a piece of software is correct according to its requirements. Functional testing will ensure that: low level components return correct results when given correct inputs, up to 10 devices can connect and communicate with each other, and the system as a whole can interpret and execute based on given conditions, which is dependent upon the state monitoring algorithm functioning correctly.

Recovery Testing

For safety and economic reasons, losing control of a UAV is an unacceptable circumstance. Recovery testing is therefore necessary to ensure that any problems that the system encounters will not lead to disaster. Testing will be conducted by injecting error conditions and observing recovery as governed by the state monitoring algorithm.

Compliance Testing

Compliance with the specifications set out in our requirements document will be observed will throughout the manual testing phase.

Test Cases

Test Cases	Scenario	Expected Behaviour	Negative Result
1. Inputting Coordinates for drones to fly	The starting point of the project before the drones take flight	Drones will begin the pre-determined operation	Nothing, the mission will need to be re-entered
2. Global State Transfer between devices	Drones are in flight and are sending and receiving state data of other drones	Drones in the swarm communicate their states to each other	The drone is not able to transmit and receive the state data and will retry

3. Recognition of drones being out of parameters	The flight path of one or more drones leads to them flying out of the parameters	One or more of the offending drones will reposition itself in the swarm	New drones will be elected to have their position readjusted
4. Leader direction change	The swarm leader determines the route must be adjusted	A new flight vector is sent to the swarm	The swarm does not receive the new flight vector and continues on the original path
5. Drone being added to network	A new drone comes online and joins the swarm	All other drones query the status of the newly joined device to determine its state	The drone is not added and will attempt to re-join, if it is already in flight the drone will land
6. Drone leaving network	The drone's mission is complete and disconnects	The drone leaves the network	The drone shuts down anyway
7. Drone being dropped from the network	During flight a drone or drones are dropped from the network	Re-establish connection and continue on with the pre-determined mission	Abort the mission and attempt to land
8. Drone flight vector correction	During a flight, conditions change and all drones in the swarm need to be adjusted	Determine the course correction and issue all drones in the swarm the new location	All drones in the swarm should attempt to land
9. State monitoring algorithm evaluating mathematical expressions	Values or coefficients for variables in a mathematical expression are input manually on Android or iOS devices for the distributed state monitoring algorithm to evaluate	The state monitoring algorithm should correctly report whether the specified constraints are fulfilled, violated, or are indeterminate	

Table 3: Test Cases

Test Schedule

Date	Summary
15 Sep 14	Manual testing of Device Communication Module started
10 Nov 14	Proof of concept test cases finished

14 Nov 14	Proof of concept build
11 Nov 14 – 28 Feb 15	Refine test cases according to changing requirements
15 Mar 15	Final build

Table 4: Test Schedule