

Cal Poly Collaborative UAV Project

Final Report



Emil Alejandria
Erik Alsterlind
Jacob Hladky

Department of Computer Engineering
California Polytechnic State University, San Luis Obispo
June 10, 2015

Contents

1	Introduction	1
1.1	Project Overview	1
1.2	Team Roles	1
1.3	Clients and Community Partners	1
1.4	Goals and Objectives	1
1.5	Results	2
2	Background	3
3	Engineering Specifications	4
3.1	Specifications	4
3.1.1	Minimum Avoidance Radius	4
3.1.2	Collision Recovery Time	4
3.1.3	Fall Back to Manual Control	4
3.1.4	Collision Priority Delay	4
3.2	Requirements Table	4
4	Final Design Overview	6
4.1	The Final Design	6
4.2	Safety Concerns	9
4.3	Project Cost	10
5	System Integration and Testing	10
5.1	FMEA	10
5.2	DVP+R	10
5.3	Overall System Analysis	11
5.3.1	Collision Avoidance Algorithm	11
5.3.2	Simulation Environment	11
5.3.3	Code Stewardship	11
5.3.4	Future Plans	11
	Appendix	12

1 Introduction

1.1 Project Overview

The Northrup Grumman UAV project, started three years ago, is a collaborative effort between Cal Poly, San Luis Obispo and Cal Poly, Pomona. This collaboration has resulted in several autonomous vehicles, including several planes, a ground vehicle, and, most recently, a quad-copter. Last year the focus of the project was the implementation of a basic collision avoidance algorithm for the planes. This year, the focus was on maintenance, compatibility, and documentation. Specifically, the goals of this team were to upgrade the autopilot system and to improve the existing collision avoidance algorithm.

The result of this project was a more robust and extensible flight-controller, a more intelligent sense-and-avoid system, and a more accessible code-base.

1.2 Team Roles

- **Emil Alejandria** — Team Lead, Sense and Avoid Architect
- **Erik Alsterlind** — System Architect
- **Jacob Hladky** — Development Tools Specialist

1.3 Clients and Community Partners

Our client on this project is The Northrup-Grumman Corporation (NGC), one of the largest defense companies in the world, and one of the primary contractors for the United States Armed Forces. NGC benefits from continued collaboration with Cal Poly and Cal Poly, Pomona as the universities provide the company a ready supply of talent for potential hire after graduation as well as a possible source for innovation.

Students benefit from NGC's involvement by exposure to the corporate requirements and practices of large, multi-disciplinary projects, and by the opportunity to work on complex hardware systems.

1.4 Goals and Objectives

Goals:

1. Redesign the existing GPS based collision avoidance system to operate more effectively and reliably.
2. Develop a sense and avoidance system for a UAV that is capable of functioning in a GPS denied environment.

3. Design a simulation suite to emulate the real world environment that the planes will be operating in.
4. Emphasize the documentation and organization of the project to aid future capstone teams in improving the project.

Objectives:

1. Understand the technical capabilities, requirements, and limitations of all potential platforms available to us so that we can select the optimal platform for our needs.
2. Draw a complete picture of environmental variables necessary to design a simulator that can sufficiently emulate a field test.
3. Outline the shortcomings in the project's documentation and work to make documentation a strength of the project rather than a weakness.

1.5 Results

Over the course of the year we have implemented a new sense and avoid algorithm that is far more complex and robust than its predecessor. With this algorithm, we have the ability to avoid multiple obstacles and discern an efficient path for reaching our goal safely. The system makes use of GPS information to produce way-points for the auto-pilot to follow. In order to test the algorithm, we feed it sample GPS points that are hard coded into the software to see how it reacts.

We have started the development of a Simulink simulation to try and more accurately model the entire flight system. To run our more sophisticated functionality, we upgraded to a commensurately powerful flight computer. The new hardware is significantly faster than the system it has replaced, and allows us to perform complex calculations with no loss of efficiency. Also, we implemented a communication protocol for sending and receiving GPS information. This protocol was agreed upon with our collaborative partners at Cal Poly, Pomona and provides an effective means of communication between autonomous systems. Finally, throughout this project, we have maintained a standard of stewardship to ensure accessibility for future contributors.

2 Background

At the end of the project's previous year, the team was able to successfully demonstrate a functioning sense and avoidance system. The demonstration entailed two autonomous planes, both running the sense and avoid algorithm, flying straight at each other. Once the vehicles got close enough to each other, they both sensed the collision, and altered their paths to turn right and avoid the other plane. Once far enough away from each other, both planes transitioned back to traveling toward their respective targets. While effective, this functionality is limited in its ability to handle different situations. There is little flexibility in what the system can handle, and the path traveled was far from efficient. To eliminate these limitations we have designed our algorithm using two core concepts: potential fields and path projection. By incorporating both we can handle a wide variety of flight situations and minimize the distance traveled while still reaching our goal. Additionally, the greater complexity of our implementation allows for more room to tune and test. As a result, the system can be improved over time.

Similar to the sense and avoid algorithm, last year's hardware setup functioned fine, but was severely limited. The overall processing power of the two micro-controllers was adequate to perform the task at hand, but allowed for little to no expansion. Seeing as we intended to greatly expand the navigation capabilities of the system, we decided to install a superior flight controller to control functionality. A survey of the available autopilot systems resulted in choosing the 3D Robotics Pixhawk because of its POSIX compatibility and prolific processing power. The value of POSIX compliance is its proven value as an industry standard in software applications, which adds credibility and portability to our project.

3 Engineering Specifications

3.1 Specifications

Discussion of high risk requirements is located in the requirements table below.

3.1.1 Minimum Avoidance Radius

This specifies the minimum distance the UAV must be from another UAV or object when it enters collision detection mode. This specification is critical because it corresponds with the turning radius of the UAV and the time the algorithm takes to determine what to do. If the algorithm detects a collision at a distance less than necessary for avoidance, a collision may occur because there is not enough time or space to avoid the object.

3.1.2 Collision Recovery Time

After entering and exiting collision detection mode, several cleanup operations may need to occur. In an airspace that contains another vehicle in flight, the UAV may need to enter collision avoidance mode repeatedly. Thus, the amount of time to reset the collision detection algorithm is critical to operation.

3.1.3 Fall Back to Manual Control

If the algorithm malfunctions, it may be necessary to land the UAV to avoid a crash. Thus it is absolutely crucial that the autopilot can be disengaged at any time.

3.1.4 Collision Priority Delay

This specification determines which plane has priority over all others during a flight. With the lowest priority, a plane must worry about detection and avoidance at normal specifications. Conversely, a plane with the highest priority does not need to worry about avoidance, and instead stays on a straight path. The purpose of this specification is to create different demonstration situations for testing and validation purposes.

3.2 Requirements Table

Risk can be either Low, Medium, or High

Spec No.	Parameter Description	Requirement with Units	Tolerance	Risk
COLLISION DETECTION ALGORITHM REQUIREMENTS				
01	Detection Radius	500ft	min	H
02	Buffer Distance	50ft	min	M
03	Altitude	70ft	min	L
04	Flight Time	20min	min	L
05	Simulation of Collision Detection Algorithm	—	max	L
06	Plane Turning Radius	20ft	min	M
07	Time for Algorithm to Detect Collision	1s	min	M
08	Collision Detection Recovery Time	5s	1s	H
09	Operating Temperature Range	50-90 deg F	+/- 5 deg F	L
10	Size of Detect Object	1 sq. ft	max	M
11	Time Spent in Collision Avoidance Mode	1min	5s	M
12	Collision Detect Re-evaluation Interval	.1s	max	H
13	Collision Detection in GPS-Limited Area	—	max	M
14	Takeoff/Landing Distance	300ft	50ft	L
15	Data Logging Interval	.01s	max	M
16	Data Communication Interval	1s	.2s	M
17	Collision Detection Control on All Axes	—	max	M
18	Distance for Communication with Partner Plane	1000ft	100ft	M
19	Distance for Communication with Base Station	500ft	50ft	M
SIMULATION ENVIRONMENT REQUIREMENTS				
20	Step Time	1us	min	M
21	Simulation Code Language Support	C++ / C	min	M
22	Platform Support	POSIX-Compliant	min	M
23	I/O Support	Input and Output Files	min	M
24	Error Facility	Report Critical Errors	min	M
CODE STEWARDSHIP REQUIREMENTS				
24	C Code Style	Consistent e.g. K&R or GNU	min	L
26	Code Documentation	Doxygen	min	M
27	Warnings	Code Compiled with -Wall, -Wextra, -Werror	min	L
28	Licensing	Consistent License for All Code	min	L

4 Final Design Overview

4.1 The Final Design

The centerpiece of the hardware setup we installed is the 3D Robotics Pixhawk with PX4 autopilot system. This unit contains a 32-bit ARM processor operating at 168 MHz and running the NuttX Real Time Operating System. This system is replacing the combination of rdupilot Mega and APM 2.6 microcontrollers that were the brains of the previous iteration of this project. The updated hardware system architecture can be seen below in Figure 1.

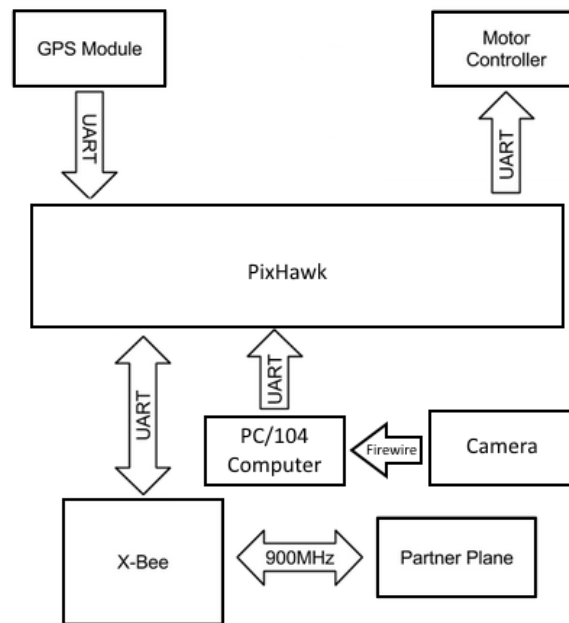


Figure 1: Final System Architecture

The increased word-size of the Pixhawk along with its significantly faster clock rate provides our new system with a much more powerful brain to perform computations and manage modules. We used 3D Robotics GPS modules that came with the Pixhawk to provide GPS points for our sense and avoid algorithm, which plugged into a designated port in the module.

The other hardware component we integrated is the Digi International Xbee S3B Pro radio module for communication between UAVs. The XBee is connected to the Pixhawk via UART serial connection, of which the Pixhawk has three open ports for use. Hardware wise, this connection is relatively simple. The 5 Volt pin of the Pixhawk port goes through a voltage regulator which brings it down to 3.3 Volts for the XBee power pin. The voltage divider is simply comprised of a LM3940 regulator, a 470 nF capacitor across the input, and a 33 uF capacitor across the output. The TX and RX pins of the XBEE and Pixhawk are crossed (TX of Pixhawk port to RX of Xbee and vice versa), and the ground pins are connected.

In terms of software, the operation of the Xbees is performed through three primary functions.

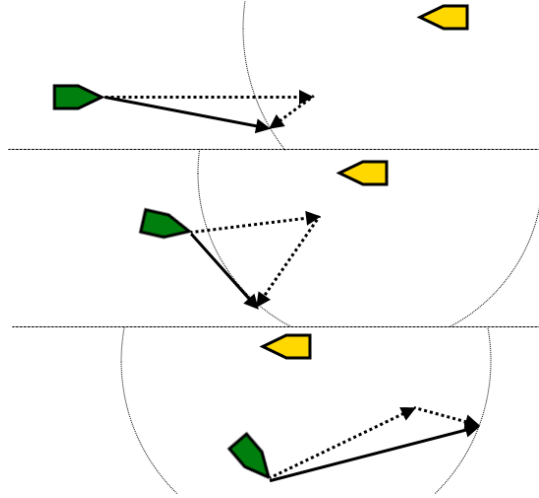


Figure 2: The two vectors represent the push away from the other plane and the pull towards the goal

The first function opens and configures the UART Port of the Pixhawk through software. In our code this function is called `uart_init()`, and can be reviewed in our software documentation which is listed in the appendix of this report. This function initializes UART settings such as baudrate and flow control. After initialization, we have a function that writes a buffer of data to the XBee, and a function that reads a specific number of bytes of data from the XBee into a buffer. Both functions, `xbee_send()` and `xbee_recv()`, can be viewed in the aforementioned software documentation.

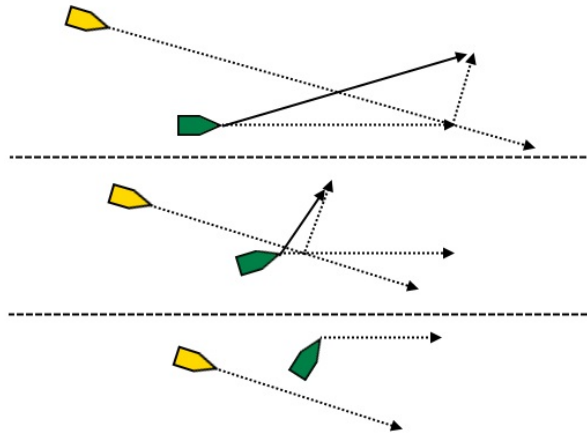


Figure 3: The algorithm sees that it can safely cross the other plane's trajectory and prevent a future collision

The design of `xbee_recv()` required the accommodation of the real time nature of the system we developed in. Normally, a `read()` system call in a POSIX environment will return a given size of data as long as it is available to be read. In the Pixhawk, only one byte of data can be read at a time. As a result, we needed to write our function to deal with this difference without losing any data. On top of the main functions we use wrapper functions to provide an additional level of functionality

and customization to the software. In Figure 4 below, you can see a software flow diagram of how the uses of the XBee are integrated into the software of the system using our functions.

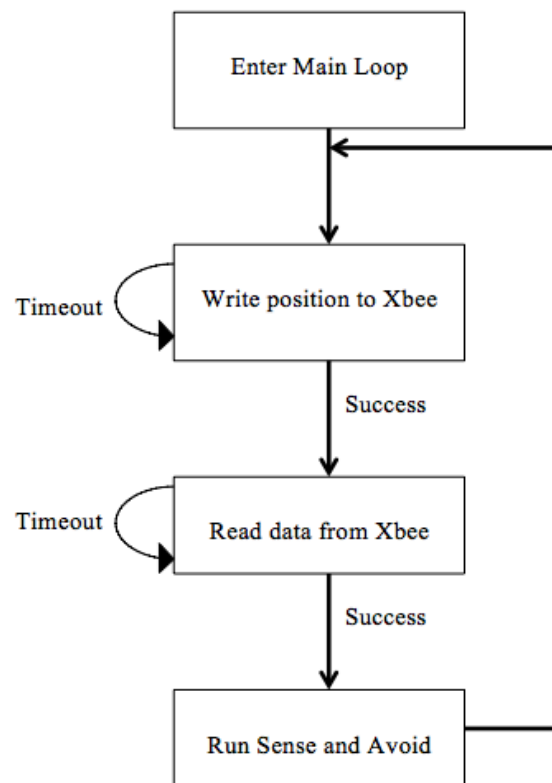


Figure 4: The software flow architecture of our program

Part of our collaboration with the Cal Poly, Pomona team was a common system of communication between autonomous vehicles. The standard that we agreed upon resulted in two types of packets that can be sent on the network. The structure of the bodies of the packets can be seen below in Figures 5 and 6. One is a system status packet that the Pomona team uses in its Ground Station to map the functionality of planes during a demonstration.

The other is a global position packet that contains GPS data that the planes need to communicate with each other for collision avoidance to function properly. The bodies of these packets contain different information, but both utilize the same general header structure. This header contains values that define what packet is being sent as well as a sync word and a checksum value to validate a packet upon its arrival. Complete documentation of the packet structure can be found in the source code.

```

typedef struct {
    int32_t sync;
    uint8_t source_id;
    uint8_t dest_id;
    uint8_t seq;
    uint8_t ttl;
    uint16_t message_type;
    uint16_t message_length;
    int8_t checksum;
} header_t __attribute__((packed));

```

Figure 5: Structure of status packet body

```
typedef struct {  
    int64_t time_stamp;  
    uint16_t vehicle_id;  
    uint8_t vehicle_mode;  
    uint8_t vehicle_state;  
} status_body_t __attribute__((packed));
```

Figure 6: Structure of position packet body

The sense and avoidance system utilizes two algorithms to determine how best to avoid any potential collisions. The algorithms each generate a pair of north and east vectors which are summed to produce a vector pointing in the optimal direction for preventing collisions. The first algorithm, called the potential field method, pushes the plane directly away from obstacles and pulls the plane toward the objective. The magnitude of the pull is constant, but the magnitude of the push varies which helps to improve pathing efficiency. The second algorithm attempts to predict and avoid future collisions. It calculates whether the plane's trajectories intersect and, if they do, the time until any collisions. It then attempts to decide whether the safest and most efficient course of action would be to proceed normally, speed up, or deviate slightly from the current flight path. The first algorithm doesn't take into consideration future situations and the second algorithm neglects more immediate hazards. By combining these two algorithms we can utilize the strengths of both as a work around to each other's weakness.

4.2 Safety Concerns

The primary concern with the system is for the safety of the planes in flight, and for any bystanders that might view future demonstrations. To ensure that no damage is done we have developed methods of simulating our sense and avoid algorithm before putting anything in the air. The first aspect of simulation involves hard coding sample data points into the software being run on the Pixhawk. By providing a variety of different sample GPS data, we can comprehensibly test how the algorithm responds to different situations, and tune our system as necessary. The second aspect is a Simulink simulation developed to model the entire system and see how it responds as realistically as possible. Having a simulation outside the Pixhawk environment allows us to make relatively major changes and test them without having to touch the actual functioning software. With these tools, we can make sure our system is prepared for physical testing without having to take major risks with costly hardware or anyone's wellbeing.

4.3 Project Cost

5 System Integration and Testing

5.1 FMEA

Our FMEA Identified 3 major points of failure, each with a few specific ways it could fail. Those points of failure are:

1. UAV Platform - the hardware keeping the UAV in the air fails in some way. Methods of failure:

- Loss of control
- Engine failure
- Loss of power

These failures are addressed by the platform team within the UAV group.

2. Pixhawk Failure - software or hardware problems related to the Pixhawk. Methods of failure:

- Algorithm failure
- Hardware failure

Algorithm failure means that we located an obstacle, but failed to avoid it. Hardware failure, on the other hand, would be electric failure of the Pixhawk and its peripherals.

3. XBee Failure - the communication is interrupted or invalidated somehow. Methods of failure:

- Loss of connection
- Bad data

These and other possible failures are all addressed in our risk management document included with this report.

5.2 DVP+R

The following tests were explored in our DVP+R:

- **Pixhawk Resilience Test:** subjecting the Pixhawk to physical stress similar to what it would endure while in flight to ensure integrity.
- **Object Detection Test:** place multiple objects / planes in the environment and ensure that the system identifies all of them.
- **Beacon Test:** Place GPS beacons in the flight area and ensure that the system avoids them.

- **Waypoint Test:** Have plane navigate through an environment with obstacles and ensure it arrives within 10m of waypoint.
- **Connection Verification Test:** Send packets via XBee in potentially restricted environments (different planes, different orientations, etc) and ensure data arrives.
- **Packet Verification Test:** Send packets via XBee in potentially restricted environments and ensure data validity.
- **IMU Test:** Move the hardware while the IMU is running to test for accuracy

Other tests explored applied to the platform team but not to this team. They can be found in the DVP+R Section of the appendix.

5.3 Overall System Analysis

5.3.1 Collision Avoidance Algorithm

Current tests indicate that our algorithm is able to make limited predictions to prevent collisions and is able to maintain a minimum distance from any obstacles while taking an efficient path. The performance of the algorithm meets and exceeds expectations, giving us the potential to continually build upon the algorithm without concern of straining system resources. More testing is necessary as our test suite is currently limited to emulated algorithm behavior.

5.3.2 Simulation Environment

The simulation suite currently consists of emulating the algorithm behavior in software and modeling a realistic system in Simulink. The algorithm emulation is sufficient for determining basic functionality and making tweaks to the algorithm. The system modeling is done in Simulink where parameters are used to more closely replicate the plane's movements as well as environmental factors.

5.3.3 Code Stewardship

We have adhered to an established standard of documentation for everything we develop in order to enhance accessibility for future contributors as well as anyone curious about the project. We have also placed all our code on a github repository for easy access and management. The private repo is provided to us by the IEEE club and is called the "IJCal Poly IEEE/firmware" repository.

5.3.4 Future Plans

The sense-and-avoid algorithm is depends on consistent, accurate GPS data. A line-fitting algorithm should be implemented to improve trajectory projection. This will also prevent failure in the

case of communication loss by predicting and adjusting the plane's path. A Kalman filter should be implemented on the GPS data to improve both the current accuracy of calculations and the accuracy of any future path prediction algorithms.

The software simulation for the algorithm needs minor improvements, however emphasis will be placed on developing the simulator in Simulink. While the framework is in place to model the system, the algorithm needs to be correctly ported into Simulink in order for testing to be completely useful.

More thorough testing needs to be performed on the autopilot software. This year we were not permitted to fly so we were unable to perform any practical testing. We have reviewed the autopilot documentation and have a fairly good idea of how it will function within our environment, but that understanding needs to be implemented and tested before any major demonstration can be performed. The autopilot functionality of the previous year's system could be used as a reference.

The voltage regulator for the XBee radios should be implemented on a PCB or proto-board. The current circuit is prototyped on a breadboard and may not withstand the rigors of flight.

UAV		Engineering Requirements																			
System Functions	Potential Solutions (From Convergent Thinking Exercise)	Best Benchmark from QFD																	Weighted Sum +	Weighted Sum -	Weighted Sum S
			Range	Power Consumption	Speed	Recovery Time	Operating Temperature	Ease of Access													
Specification Weight			40	20	20	5	5	10													
Collision Sensor	IR	S	-	+	S	+	+	+											40	40	20
	Computer Vision		+	-	S	+	+	S											50	20	30
	Radar		+	-	S	+	+	-											50	30	20
Autopilot	APM 2.6	S	S	+	-	-	+	S											25	25	50
	PixHawk		S	S	+	+	+	S											30	0	70
Simulation Environment	Unit Step	S	S	S	S	+	S	+											15	0	85
	Event Based		S	S	+	S	S	S											20	0	80
Camera	CCD Firewire	S	S	S	+	+	+	+											40	0	60
	CMOS USB		S	S	S	S	+	S											5	0	95
Processing Solution	PixHawk	S	S	+	S	S	+	+											35	0	65
	PC104 FW35208HR		S	-	+	+	+	S											30	20	50

CPE 450 DESIGN VERIFICATION PLAN AND REPORT													
Report Date		Sponsor						Component/Assembly		REPORTING ENGINEER:			
TEST PLAN										TEST REPORT			
Item No	Specification	Test Description	Acceptance Criteria	Test Responsibility	Test Stage	SAMPLES		TIMING		TEST RESULTS			NOTES
						Quantity	Type	Start date	Finish date	Test Result	Quantity Pass	Quantity Fail	
1	Physical	Pixhawk Resilience Test	No functional change in pixhawk	Emil	PV	20	C	2/4/2015	2/20/2015				
2	Software	Object Detection Test	All detectable object are detected	Emil	PV	20	C	2/11/2015	2/25/2015				
3	Software	Beacon Test	Planned route avoids all beacons	Taylor	PV	20	C	2/11/2015	2/25/2015				
4	Communications	Packet Verification Test	Check outgoing packets to ensure valid data is being sent	Erik	PV	100	C	2/18/2015	3/4/2015				
5	Communications	Connection Verification Test	Ensure packets are being received by other plane	Jacob	PV	100	C	2/18/2015	3/4/2015				
6	Software	IMU Backup Test	In event of GPS failure, use IMU to navigate to waypoint	Taylor	PV	20	C	2/18/2015	3/4/2015				
7	Physical	Glide Test	No damage to plane	Chris	PV	1	C	N/A	N/A				
8	Physical	Engine Test	Engines run for 1 minute with no issues	Chris	PV	5	C	N/A	N/A				
9	Software	Waypoint Test	Plane arrives within 10M of designated waypoint	Jacob	PV	5	C	2/18/2015	3/4/2015				
10	Physical	Battery Test	Verify battery holds >95% of listed charge	Emil	PV	2	C	2/11/2015	2/25/2015				
11													
12													
15													
16													
17													
18													
19													
20													
21													
22													
23													
24													
25													
26													
27													
28													
29													
30													

Failure Mode and Effects Analysis

Description

This template illustrates a Failure Mode and Effects Analysis (FMEA), also referred to as a Potential Failure Mode and Effects Analysis (PFMEA) or Failure Modes, Effects and Criticality Analysis (FMECA). A detailed discussion can be found at www.ASQ.org

[Learn About FMEA](#)

Instructions

- Please follow the link for detailed instructions for data entry
- Initiate action to reduce the RPN
- Re-evaluate the RPN value after completion of the recommended actions

Learn More

To learn more about other quality tools, visit the ASQ Learn About Quality web site.

[Learn About Quality](#)

FAILURE MODE AND EFFECTS ANALYSIS

Item: UAV Responsibility: UAV Team
 Model: Current Prepared by: UAV Team
 Core Team: Taylor Hutchinson, Emil Alejandria, Erik Alsterlind, Jacob Hladky

FMEA number: _____
 Page : 1 of 1
 FMEA Date (Orig): 1/22/2015 Rev: 1

Process Function	Potential Failure Mode	Potential Effect(s) of Failure	Severity	Cause(s)/ Mechanism(s) of Failure	Detection	Current Process Controls	Detectability	RPN	Recommended Action(s)	Responsibility and Target Completion Date	Action Results				
											Actions Taken	Severity	Occurrence	Detectability	RPN
UAV Airframe	Loss of control	Collision, crash	9	Control system disconnect, loss of power, out of range	1	Built-in glide on power loss	4	36	Verify glide functionality	Chris before next flight					0
	Engine failure	Collision, crash	9	Mechanical failure, Power loss	2	Built-in glide on power loss	4	72	Test engines before flying, make sure batteries are charged properly	Chris before next flight					0
Autopilot (PixHawk)	Hardware Failure	Collision, crash	10	Wire disconnect, PCB failure	2	None	10	200	Check connections before flying, make sure everything is strapped down securely	Capstone team (testing), Chris (before flight)					0
	Algorithm Error	Collision, crash	8	Bugs in code	2	Heavy testing & verification	3	48	Test, Test, Test	Capstone team					0

[illegible]

Taylor Hutchinson

Jacob Hladky

Erik Alsterlind

Emil Alejandria

Assignment 5 – Personas

Persona 1 – Military UAV Controller



Paul's life was fairly unremarkable, though certainly not unsuccessful. He grew up near the city, but was lucky enough to go to school in the suburbs. He was a C student in high school, so he enlisted in the US air force straight afterward. He went through boot camp then was sent to college after the aptitude tests. He did well in college, graduating in the upper half of his class. Then the air force sent him to the Middle East, but he was one of the lucky ones there. Rather than being asked to hunt insurgents, his work was on an airfield doing repair work. He helped maintain and repair various air force vehicles, specializing in UAVs.

Last week, he got a stroke of luck – a promotion. He was no longer a simple repair man, and he was one step closer to his dream. He was now a member of the UAV control team. One day he hoped to be its leader. The training period lasted a week, teaching him about the various planes they used and the level of control each required. Some needed to be manually controlled during only takeoff and

landing, while others had little to no autonomous capability. He already knew how the planes worked and the basics on how to fly them, but he had to learn more detail on flight and how they were monitored.

Today he would finally get to fly. Or, at least handle a takeoff or two. He walked into the sparse, functional air force building as he had many other days in his crisp uniform. He saluted a few senior officers who walked by, as per normal. Today, though, he was headed for his desk instead of the airfield. He could hardly stop smiling as he approached the desk and greeted his fellow teammates. There wasn't time to celebrate now – he didn't want to ruin things on his first day. He would do that later, after his shift was finished. For now, he went to work.

Person #2: A commercial company purchases navigational systems from Northrop Grumman. For example, Amazon to use with an automated package delivery system using UAVs.

Eric is one of the programmers working on Amazon's drone delivery service. After a delayed, but otherwise normal, morning routine he eagerly heads off to work, looking forward to the day before him. He greets his coworkers as he arrives at his desk, some with enthusiasm and others with a perfunctory wave. Setting his coffee aside for a moment, he brings up Reddit to update himself on the world's news. It's only a few moments before he pulls up his work from the day before: some of the drone routines to be implemented in their delivery system.



The navigational system, acquired elsewhere, is the least of their concerns as they try to convince the FAA, FCC, and various other overly regulatory branches of the government of their system's reliability. They've put in enough work to account for every possible contingency from drone malfunctions to theft to hawk attacks. The only way to cut through the red tape is to show that their system will continue to function when faced with disasters of cataclysmic proportions. Above that, though, the system needs to be quick and efficient. Not only do they need to still cut a profit, they need to provide a service that meets the standards for which they're known.

After reexamining the results of their recent navigation tests Eric spent the past couple hours looking through the drone's package drop-off subroutine. The system worked flawlessly in every field test presented, but he still felt that the process could be improved. He had looked through these lines so much they he could probably write the entire subroutine from memory. Little had changed over the past couple weeks of going through the code, but there was nothing else to do but try and improve what they had until it cleared the desks in D.C..

Persona #3: Search and Rescue



This persona summary is that of a search and rescue member or team that might need the services of our UAV to perform their job. This person or group is most likely either militarily trained or gone through some civilian form of rescue training. They are extremely focused on getting their job done because it could mean deciding the fate of people's lives. It's unlikely that they are trained in a computer engineering related field, so repairing the functionality of the aircraft in the field most likely isn't a possibility. As far as behavior patterns, a member of a search and rescue team would be very serious about their job as well as selfless. Often, rescue personnel have to put themselves in dangerous situations to get their job done, so effectiveness is at the top of his concerns. His or her goal is simply to get the job done as efficiently as possible. If our UAV is launched in an effort to locate a lost individual or object and it took 12 hours to accomplish its task because it didn't operate efficiently then the rescue team wouldn't be satisfied. The quicker the task can be accomplished the better, as far as the rescue personnel is concerned. Their skills are generally some combination of military and rescue training. This can include military training, parachuting, rock climbing, or medical training; basically, any skills necessary to help someone who is lost or in trouble. It is unlikely that they have any skills related to the UAV project unless he or she is an officer trained at an academy like West Point where there are strong engineering programs. They are disciplined and goal oriented as their profession is too intense for them

not to be. The general environment is a search and rescue mission, for an individual or a group of people. Lives are most likely on the line so the mood of the rescue team is intense and focused. While the general attitude would be the same across situations, the actual physical environment could vary drastically. The mission could occur in the ocean with a coast guard type unit, in a war zone where a squad needs to be located for extraction, or in a forested area in the midst of a wild fire. The possibilities are only limited by the different ways people can get lost or into trouble.

Sources:

1. <http://www.uscg.mil/top/about/>
2. <http://www.airforce.com/careers/detail/pararescue-males-only/>
3. http://a1.g.akamai.net/f/1/15157/1h/dodairforce.download.akamai.com/15157/airforce/careers/career_details/i/detail/Pararescue_694x306.jpg

Processes for Handling Risks

1. Loss of communication with the base station
 - a. Consequences:
 - i. Loss of camera follow capability
 - ii. Loss of real-time data updates
 - b. What to do:
 - i. Keep at least two pairs of eyes on the plane at all times
 - ii. Switch to manual control
 - iii. Land plane as soon as possible
 - c. Prevention methods
 - i. Communication test suite before each flight
 - ii. Redundant communication links (future)
2. Failure to receive data from other plane
 - a. Consequences:
 - i. Loss of real-time updates on other plane
 - b. What to do:
 - i. Keep at least two pairs of eyes on the plane at all times
 - ii. Take manual control if the plane looks like it's going to fail to avoid an obstacle
 - iii. Ground plane if necessary
 - c. Prevention Methods:
 - i. Test before flight
 - ii. Component checks
3. Failure to initialize communication to other plane or base station
 - a. Consequences:
 - i. Loss of real-time updates on other plane
 - b. What to do:
 - i. Keep at least two pairs of eyes on the plane at all times
 - ii. Take manual control if the plane looks like it's going to fail to avoid an obstacle
 - iii. Ground plane if necessary
 - c. Prevention Methods:
 - i. Test before flight
 - ii. Component checks
4. Electrical Failure
 - a. Types:
 - i. Loss of battery power
 - ii. Loss of connection to battery
 - iii. Short in electrical system
 - b. Consequences:
 - i. Plane shut down
 - ii. Plane slows down
 - iii. Crash landing

- c. Prevention Methods
 - i. Redundant connections / batteries
 - ii. Follow battery charging processes
 - iii. Checklist all connections before flying
 - iv. Visual confirmation of all connections
 - v. Only use locking connectors
5. Avoidance Failure
- a. Causes:
 - i. Bad data from other planes (incorrect GPS points)
 - ii. Terrain features (Ex: Glacial Circe)
 - iii. Undetectable aerial hazards
 - b. Prevention Methods:
 - i. Require multiple messages with the same GPS point
 - ii. Cross check data to ignore clearly invalid points
 - c. What to do:
 - i. Switch to manual control until danger is gone
 - ii. Ground plane if necessary
6. GPS Denial
- a. Causes:
 - i. GPS Module Disconnection
 - b. Prevention Methods:
 - i. Line-fitting prediction in sense & avoid algorithm
 - c. What to do:
 - i. Keep eyes on the plane
 - ii. Take manual control if the plane looks like it's going to fail to avoid an obstacle
 - iii. Ground plane if necessary