

2016

# Final Report

## PROJECT PEGASUS: UAV PACKAGE DELIVERY SYSTEM

### TEAM AVENGERS

Tushar Agrawal  
Sean Bryan  
Pratik Chatrath

(Systems Lead and Project Manager)  
(Mechanical and Communications Lead)  
(Sensor Lead and Software Developer)

May 5, 2016

## ***Abstract***

*Drones represent a unique opportunity to disrupt the package-delivery industry. However, unlike current upstarts, an application that leverages existing delivery-company infrastructure was developed. Efforts to implement package delivery by UAV are detailed in this report. This includes the systems used and developed, project management considerations, lessons learned, and future work. Significant progress was made to develop a means of delivering packages by drone in real-world conditions. In tests, packages were successfully developed 90% of the time in a simplified scenario that laid the groundwork for future development. The project made use of commercially-available components wherever possible and successfully integrated these components into a coherent system. The work presented represents a repeatable platform for UAV delivery that has the potential to radically change existing models in addition to jump-starting similar projects.*

# Contents

1. Project Overview .....	1
1.1 Objectives .....	1
1.1.1 Background .....	1
1.1.2 Problem Summary.....	1
Delivering packages to a house using UAVs.....	1
1.1.3 Project Description.....	1
2. Use Case.....	1
3. System-Level Requirements .....	3
3.1. Mandatory .....	3
3.1.1 Functional Requirements .....	3
3.2.2 Non-Functional Requirements .....	3
3.2. Desired .....	4
3.2.1 Functional Requirements .....	4
3.2.2 Non-Functional Requirements .....	4
3.3 Performance Requirements .....	4
3.4 Subsystem Requirements: .....	4
S.1 Vision.....	4
S.2 Obstacle Detection and Avoidance.....	5
4. Functional Architecture .....	5
5. System Level Trade Studies.....	6
5.1 UAV Trade Study .....	6
5.2 Vision Board Processing.....	7
5.3 Sensor Suite .....	7
6. Cyber Physical Architecture .....	8
6.1 Mechanical System .....	9
6.2 Electrical System .....	9
6.3 Software .....	10
7. System Description and Evaluations.....	10
7.1 System and Subsystem Descriptions.....	10
7.1.1 Behaviour Subsystem.....	11
7.1.2 Flight Control Subsystem.....	13
7.1.3 Vision Subsystem.....	13

7.1.4 Obstacle Avoidance Subsystem .....	16
7.1.5 Gripper Subsystem.....	19
7.2 Subsystem Modelling, Testing and Analysis .....	19
7.2.1 Flight Control Subsystem.....	19
7.2.2 Vision Subsystem.....	20
7.2.3 Obstacle Avoidance Subsystem .....	21
7.3 System Performance Evaluation against Spring Validation Experiment.....	22
7.3.1 Package delivery without obstacles .....	22
7.3.1 Package delivery without obstacles .....	22
7.4 System Strengths and Weaknesses .....	23
8. Project Management .....	23
8.2 Schedule.....	25
8.3 Risk Management .....	25
8.3.1 Marker Search Algorithm not Robust.....	26
8.3.2 Weather Limiting Testing Opportunities .....	26
8.3.3 Integration of Navigation Stack with Pixhawk .....	27
8.4 Budget.....	27
9. Conclusion .....	28
9.1 Lessons Learned.....	28
9.1.1 Simplify.....	28
9.1.2 Outdoor Drone Considerations .....	28
9.2 Future Work.....	30
10. References.....	31

# 1. Project Overview

## 1.1 Objectives

### 1.1.1 Background

Currently, package delivery truck drivers hand-carry packages door to door. This model is used by Federal Express (FedEx), United Postal Service (UPS), United States Postal Service (USPS), and Deutsche Post DHL Group (DHL). We believe that drones have the potential to expedite this system. According to Amazon, 80% of packages shipped weigh 5 lbs or less -- an ideal weight for current technology.

Amazon is developing Prime Air with the same intent. However, we believe the most efficient system combines delivery trucks with Unmanned Aerial Vehicles (UAV's) which saves time, expense, and improves customer's satisfaction.

### 1.1.2 Problem Summary

Delivering packages to a house using UAVs.

### 1.1.3 Project Description

Given the coordinates of the house, a UAV with a package takes off from point A, autonomously reaches close to the house, scans the outside of the house for a visually marked drop point, lands, drops off the package, then takes off again to land on another platform at point B.

## 2. Use Case

Sam drives a package delivery truck for one of the largest parcel delivery companies. He arrives each morning to a preloaded truck and is handed his route for the day. Even though he has an assigned route, he sometimes is tasked with delivery packages to additional streets. These are often the packages that should have been delivered the day before. Thus it's critical that packages make it to the right house on time today.

Now that his company uses drones, Sam can cover more area in less time. He drives out to his first neighborhood for the day with two packages to deliver. He can quickly deliver the first package, which is heavier. The second package is lighter but a street over. After parking, he quickly attaches the second package to a drone and selects the address on the base station computer. The drone takes off and disappears over a rooftop as Sam unloads the first package.

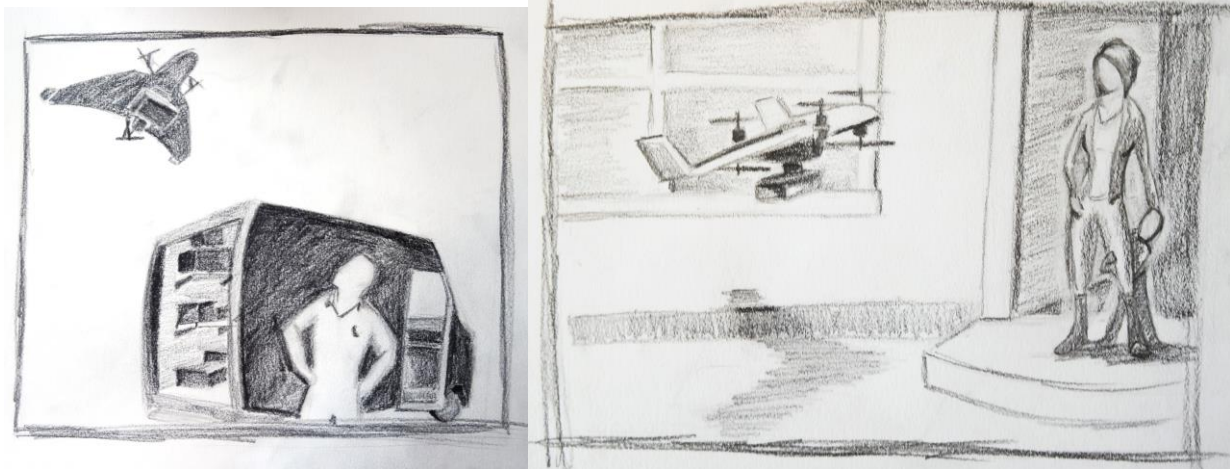


Figure 1 and 2: Artist's Rendition of Sam and one of his drones; Artist's Rendition of Drone Delivering Packages to End Customer

Having delivered the first package, Sam gets back in the truck and starts driving. In the past, he would have driven to the next house and dropped off the package. Nowadays, Sam knows that the drone will deliver the package to the right house and catch up. This saves him a few minutes which adds up over the course of the day to real time savings. This makes Sam a little happy.

Meanwhile, the drone has moved within vicinity of the second house. It begins scanning around for the visual marker outside the house. The drone finds the marker and moves in for a landing. It's able to avoid people on the sidewalk and the large tree outside the house. The drone lands on the marker and does a quick confirmation, it checks the RFID code embedded in the marker. Confirming the correct house has been found, the drone releases the package and notifies the package delivery truck's base station. The base station then updates the drone on the delivery truck's position.

The drone catches up to Sam at a red light and they continue on their way. Sam's day continues this way.

On a major street, he has several packages to deliver in the area. He quickly loads up a few drones, selects the addresses, and watches as the drones do all the work. Sam had to get a gym membership since he's no longer walking as much, but he's happy to be getting through neighborhoods substantially faster. Because the drones allow one driver to do more, the delivery company is able to offer package delivery at a more competitive rate with more margin. This makes customers happy in addition to getting their packages faster. In turn, they are more likely to use the delivery company, which makes the company pleased with their investment.

Late in the day, the base station on Sam's delivery truck notifies him that an adjacent route wasn't able to deliver a package. In the past, this would have meant that the package would be driven back to the warehouse to be resorted and delivered with tomorrow's load. This was a substantial waste of fuel and manpower. Today, routes can be dynamically updated. A drone will deliver the package to Sam's truck. Once he's driven into the correct neighborhood, the drone will deliver the package. The customer will never know there was a problem, and the delivery company saves money.

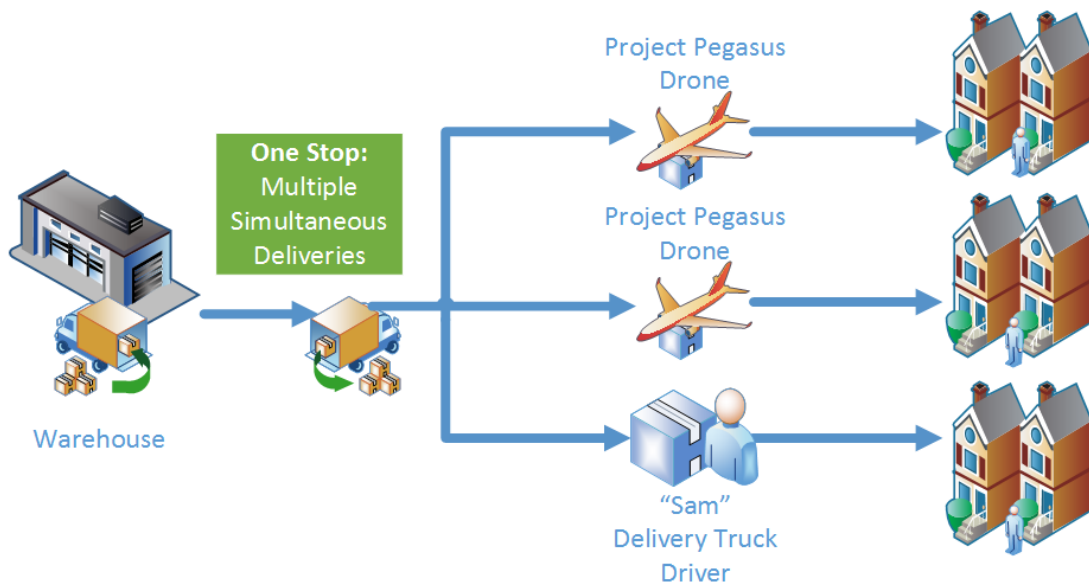


Figure 3: Full Scope of Delivery System

Sam arrives back at the warehouse, his truck empty. He's satisfied in the work he's accomplished, customers are happy that received their packages on time, and the delivery company is exceptionally happy with the improved efficiency and customer retention.

### 3. System-Level Requirements

The critical requirements for this project are listed below under Mandatory Requirements. These are the 'needs' of the project. Additionally, the team identified several value-added requirements during brainstorming. These 'wants' are listed below under Desired Requirements.

#### 3.1. Mandatory

##### 3.1.1 Functional Requirements

- M.F.1 Hold and carry packages.
- M.F.2 Autonomously take off from a visually marked platform.
- M.F.3 Navigate to a known position close to the house.
- M.F.4 Detect and navigate to the drop point at the house.
- M.F.5 Land at visually marked drop point.
- M.F.6 Drop package within 2m of the target drop point.
- M.F.7 Take off, fly back to and land at another visually marked platform.

##### 3.2.2 Non-Functional Requirements

- M.N.1 Operates in an outdoor environment.
- M.N.2 Operates in a semi-known map. The GPS position of the house is known, but the exact location of the visual marker is unknown and is detected on the fly.
- M.N.3 Avoids static obstacles.
- M.N.4 Sub-systems should be well documented and scalable.

M.N.5 UAV should be small enough to operate in residential environments.

M.N.6 Package should weigh at most 100g and fit in a cuboid of dimensions 9.5'' x 6.5'' x 2.2''.

## 3.2. Desired

### 3.2.1 Functional Requirements

D.F.1 Pick up packages.

D.F.2 Simulation with multiple UAVs and ground vehicles.

D.F.3 Ground vehicle drives autonomously.

D.F.4 UAV and ground vehicle communicate continuously.

D.F.5 UAV confirms the identity of the house before dropping the package (RFID Tags).

D.F.6 Drop package within 1m of the target drop point.

D.F.7 Takes coordinates as input from the user.

D.F.8 Communicates with platform to receive GPS updates (intermittently).

### 3.2.2 Non-Functional Requirements

D.N.1 Operates in rains and snow.

D.N.2 Avoids dynamic obstacles

D.N.3 Operates without a GPS system.

D.N.4 Has multiple UAVs to demonstrate efficiency and scalability.

D.N.5 Compatible with higher weights of packages and greater variations in sizes.

D.N.6 Obstacles with a cross section of 0.5m x 0.5m are detected and actively avoided.

D.N.7 A landing column with 2m radius exists around the visual marker

D.N.8 Not reliant on GPS. Uses GPS to navigate close to the house. Does not rely on GPS to detect the visual marker at the drop point.

## 3.3 Performance Requirements

P.1 UAV places the package within 2m of the target drop point.

P.2 UAV flies for at least 10 mins without replacing batteries.

P.3 UAV carries packages weighing at least 100g.

P.4 UAV carries packages that fit in a cube of 9.5'' x 6.5'' x 2.2''.

P.5 One visual markers exists per house.

P.6 Visual markers between houses are at least 10m apart.

P.7. A landing column with 3m radius exists around the visual marker

P.8 Obstacles with a minimum cross section of 1.5m x 0.5m are detected and actively avoided.

## 3.4 Subsystem Requirements:

### S.1 Vision

S.1.1 The size of the marker must be within a square of side 1.5m.

S.1.2 Error in the X,Y,Z position of the marker from the camera should be correct up to 10% of distance from it.



## S.2 Obstacle Detection and Avoidance

S.2.1 Obstacles must be detected with a range of 50 cm to 150 cm from the UAV.

S.2.2 Obstacles should be at least in 90% of the situations/positions.

S.2.3 Distance to the obstacle should be correct with a maximum error of 20 cm.

S.2.4 Natural obstacles around a residential neighborhood should be detected.

## S.3 Flight control

S.3.1 UAV must reach the GPS waypoint with a maximum error of 3m.

S.3.2 UAV should be able to fly 10 minutes without replacing the batteries.

## 4. Functional Architecture

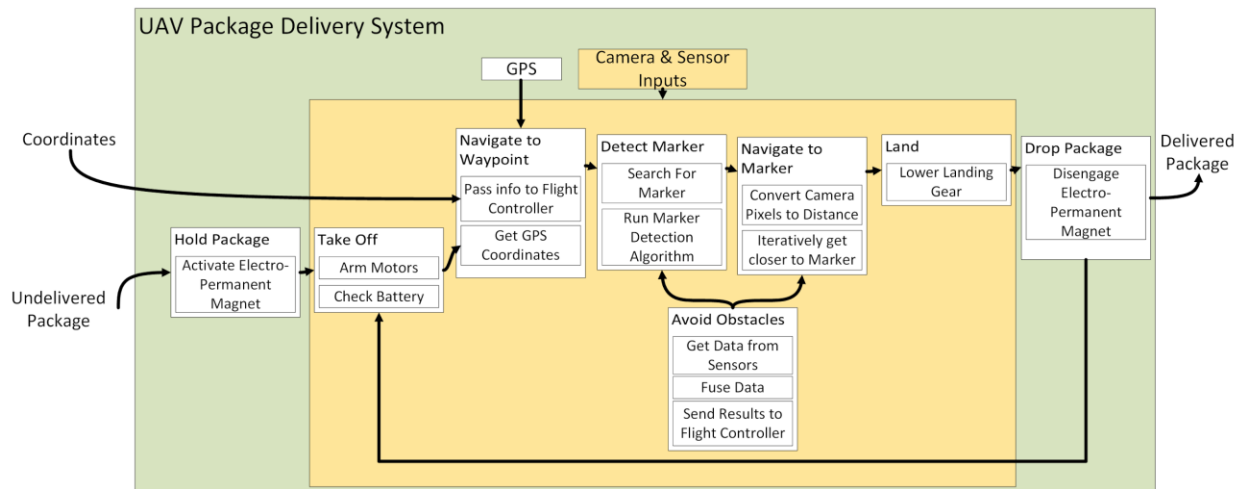


Figure 4: Functional Architecture Diagram

The revised functional architecture of our system is as shown in Figure 4. Viewing the whole system as a black-box, there are 2 inputs – the package to be delivered and GPS coordinates of customer location. The output of the system is the package successfully delivered at the destination.

Looking inside the black box now, the UAV initially holds the package by activating an electro-permanent magnet. Coordinates of the customer location are input to the User Interface. The developed Plan Mission software decides the navigation waypoints and plans a path to the destination. This information is then relayed to the UAV by the communication interface. The mission planning software continuously receives the current coordinates from the UAV and sends updated coordinates back to the UAV.

Meanwhile the UAV checks the battery status. If there is sufficient battery, the UAV arms the motor and takes off. The UAV navigates using the waypoint to the vicinity of the destination using GPS input. It then switches to the marker detection code. The UAV takes input from the camera and starts to scan the vicinity of the customer destination for the marker put up by the customer. It moves in a predetermined trajectory for scanning. Once the marker is detected the vision algorithm maps the size of the marker in the image to actual distance of the UAV from the marker. The UAV continuously receives this information and moves towards the marker. The UAV finally lowers lands on the marker. It drops the

package by disengaging the electro-permanent magnet and flies back to the base station using waypoint navigation.

During the ‘Detect Marker’ & ‘Navigate to Marker’ functions, the UAV continuously runs an obstacle avoidance algorithm on-board. The obstacle-avoidance algorithm continuously receives data from sensors, fuses the data and asks the flight controller to alter its trajectory if there is an obstacle in its path.

## 5. System Level Trade Studies

Our project contains three major trade studies. We had to choose the UAV platform, our visual processing board and our sensor suite for obstacle detection.

### 5.1 UAV Trade Study

There are many factors that go into selecting a proper UAV platform such as cost, shipping time, payload capacity, and flight time. In the end, we used 10 different metrics to evaluate the best choice for our application. The top three highest weights were payload capacity, price, and flight time. Payload was given the highest priority since it was crucial to our project application. A UAV without the ability to carry a package was useless in our project. Price was second highest because it was a large constraint on our project. Flight time was given the third highest weight because of the necessity that the vehicle consistently reach the door of the house and return to the platform no matter how far the house was from the street.

The remaining factors were derived from our performance requirements and scaled appropriately based on their effect on the project timeline and ease of integration into the complete system and vision of our project. The results of our trade study can be seen below:

<b>Metric</b>	<b>weights</b>	<b>BEV FireFly6</b>	<b>3DR X8+</b>	<b>Erle Hexcopter</b>	<b>3DR Solo</b>	<b>3DR X8-M</b>
Payload	22	6.0	4.5	10.0	0.5	4.5
Price	20	5.0	6.5	7.5	10.0	0.5
Flight time	17	10.0	3.3	3.3	3.3	3.3
Size of vehicle	10	2.0	10.0	10.0	8.0	10.0
Shipping time	10	9.0	9.0	1.0	9.0	9.0
Flight controller capabilities	8	7.0	7.0	6.0	10.0	7.0
Firmware documentation	8	10.0	10.0	2.0	10.0	10.0
Ease of adding mechanical subsystems to frame	5	6.0	10.0	3.0	1.0	10.0
<b>Total</b>	<b>100</b>	<b>6.8</b>	<b>6.6</b>	<b>6.2</b>	<b>6.0</b>	<b>5.4</b>

Table 1: Trade study for the UAV platform

As per our trade study we bought and used BEV FireFly6 UAV for Fall Semester. However by the end of the Fall semester we realized that FireFly6 was the weakest subsystem of our project. We faced several issues getting the UAV up and running. Primary reason for that was using the beta code from BirdsEyeView which in turn had little documentation and compass compatibility issues. Hence we decided to fall back on our second best option - 3DR X8 Plus for spring semester.

## 5.2 Vision Board Processing

As will be depicted in the cyber physical architecture, our system has a separate board for obstacle avoidance and visual detection subsystem so that we can do processing in real time during flight. The second board also allows the system to meet safety requirements by allowing all safety critical functions to be run on the flight controller which won't get bogged down by computer vision algorithms.

Our main criteria was processing speed and documentation. Documentation was critical because the board will have to integrate with the rest of the system and we will be designing this ourselves. As a result, being able to debug errors will ensure that the system functions as a whole. Ports/Interfaces were also critical because they affected how the board interacted with the cameras and the rest of the UAV. Our last criteria was price to ensure we met our project budget.

<b>Metric</b>	<b>weights</b>	<b>Raspberry Pi 2</b>	<b>BeagleBoard-xM</b>	<b>Odroid-U4</b>	<b>BeagleBone Black</b>
Processing power	30	4	6	10	6
Documentation online	30	10	9	5	9
Ports/interfaces	25	7	10	8	6
Price	15	10	3	6	8
<b>Total</b>	<b>100</b>	<b>7.45</b>	<b>7.45</b>	<b>7.4</b>	<b>7.2</b>

Table 2: Trade study for microprocessor (for vision)

Raspberry Pi 2 and BeagleBoard-xM tied for the first place. However we ended up using Odroid because of the following reasons:

- We found out that BeagleBoard-xM were no longer sold. So we would not have any spares which would turn to be a high risk scenario for our project.
- Our second option Raspberry Pi scored 4/10 in processing power whereas Odroid scored 10/10. Hence we found Odroid to be a better choice.

## 5.3 Sensor Suite

The sensor system was critical for the obstacle avoidance functionality of our UAV. We considered three sensors: IR sensors, Ultrasonic Sensors and Lidar. We developed prototype version of obstacle detection system using all the three sensors. Based on the results of the tests conducted with the prototypes we finally decided to use Hokuyo UTM-30LX LIDAR sensor for our obstacle avoidance system.

The table 3 shows the result of our sensor trade study.

Metric	weights	IR	Ultrasonic	Hokuyo URG - 04 LIDAR	Hokuyo UTM-30lx LIDAR
Number of sensors required	1	39	14	1	1
Cost (Cheap, Costly)	1	Cheap	Cheap	Costly	Costly
Detection Range	1	0.5 - 1m	1.5m	4m	30m
Other	1	-	Interference issue	Does not function outdoor	Works outdoor Can borrow one from another Lab
<b>Total</b> <b>Red = -1 point</b> <b>Green = +1 point</b>	<b>5</b>	<b>-1</b>	<b>0</b>	<b>0</b>	<b>2</b>

Table 3: Trade study for sensor suite (for obstacle avoidance)

## 6. Cyber Physical Architecture

The high-level cyber physical architecture can best be understood by Figure 5. On a high-level, the system can be broken down into three major categories: mechanical components, electrical components, and software. The electrical components are the bridge between the software and the mechanical actuation.

The obstacle avoidance and vision system architecture show the flow of information between the software, electrical and mechanical components for their corresponding systems.

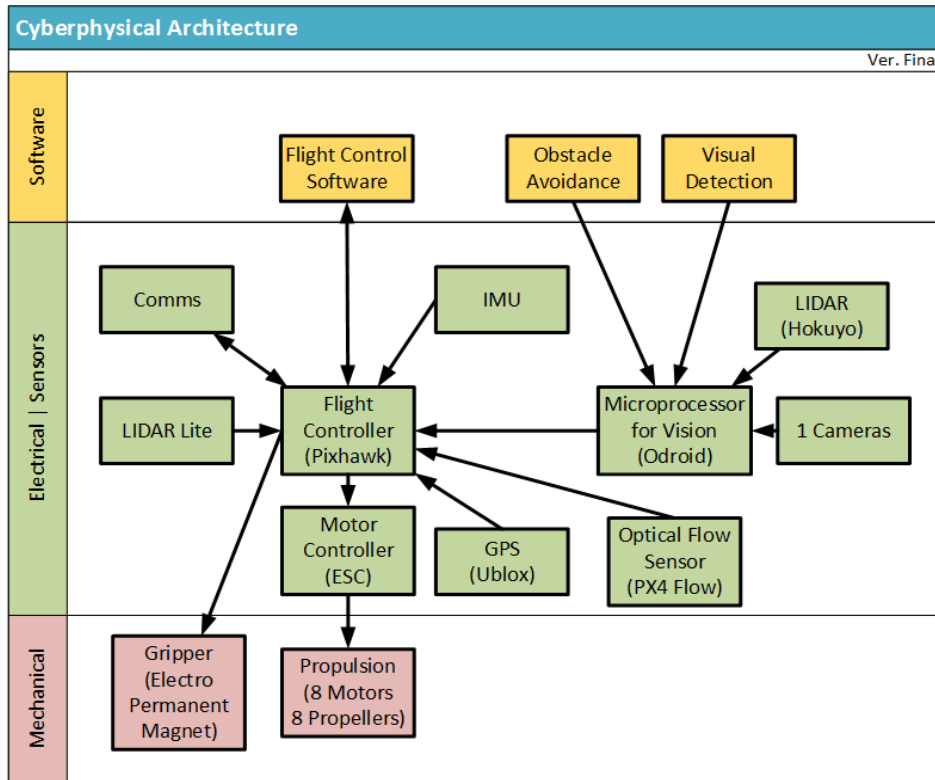


Figure 5: Cyber Physical Architecture Diagram

### 6.1 Mechanical System

We are using 3DR X-8+ UAV for our project. The mechanical system of our project consists of the propulsion system and the gripper. The propulsion system is part of the 3DR kit that we purchased but must be controlled appropriately by our software. The gripper is made by NicaDrone -- an electro-permanent magnet [2]. This gripper is the interface between the vehicle and the package and must allow the package to be dropped off upon arriving at the destination. It will be controlled by our flight control system which is the brain of the UAV.

### 6.2 Electrical System

The electrical system is composed on a high-level by the flight control board, the vision subsystem hardware, sensors, and the communications hardware. The flight controller is Pixhawk [1] the brain of the entire system and runs all critical flight control software. The flight controller interacts with two sensors on the vehicle: the IMU and GPS. The flight controllers takes commands from the main processing board which directs it to avoid obstacles and land on visual markers. The output from the flight controller goes to the motor controller and is then converted into appropriate signals to control the propulsion system.

Odroid[3] - microprocessor for running visual algorithms connects to the camera and Lidar on board the UAV. Odroid runs vision and obstacle detection algorithms and outputs the result to the flight controller.

### 6.3 Software

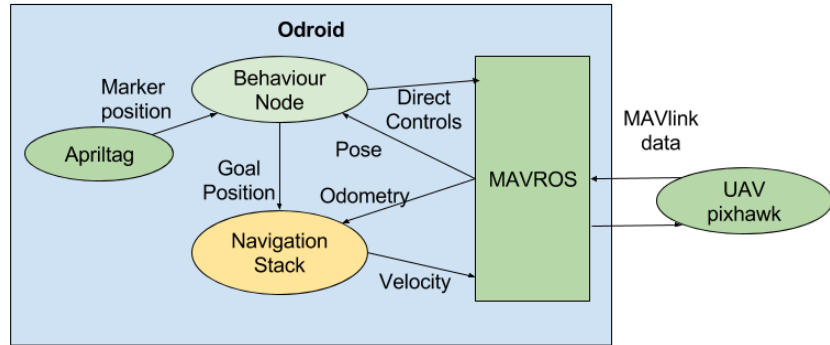


Figure 6: Software architecture of the system

The software of the system comprises of 2 main levels. The lower level control is implemented in the flight controller. This runs the control environment to monitor the position and orientation of the UAV to maintain stable flight. It uses the GPS, Compass, IMU and a Barometer as sensors.

The higher level control runs the application specific program and controls the UAV through the lower level control. It interfaces with the camera and the Lidar. It runs the behavior program in addition to the vision processing algorithms (AprilTag detection), obstacle detection and planning algorithms.

## 7. System Description and Evaluations

### 7.1 System and Subsystem Descriptions

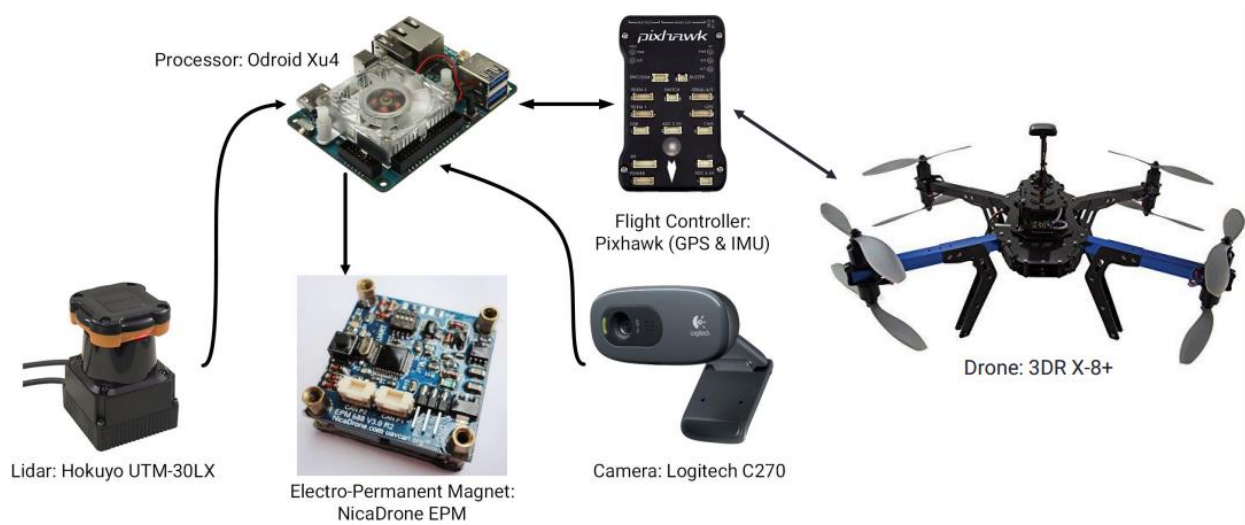


Figure 7: System architecture with all components

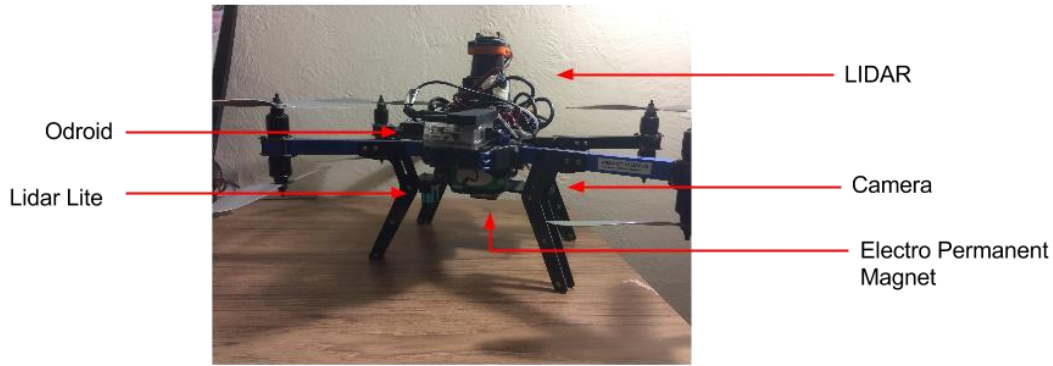


Figure 8: Image depicting UAV with all system components

The system consists of multiple subsystems. The 3DR X8+ is the UAV body. It comes with the Pixhawk Flight controller [1] which is responsible for maintaining the stability while flying the drone. It receives commands from the main processor, Odroid[3], which runs the application specific algorithms. The details of the subsystems are given below:

### 7.1.1 Behaviour Subsystem

The behavior subsystem is responsible in orchestrating the various subsystems towards the mission of package delivery. It decides which subsystems should function at which stages in the mission and how the outputs from them flow to other subsystems. It runs on the Odroid[3].

#### 7.1.1.1 State machine formulation

A simple state machine is used here. Every state denotes a unique stage of the system and determines the associated process and subsystems that must be functioning at that stage.

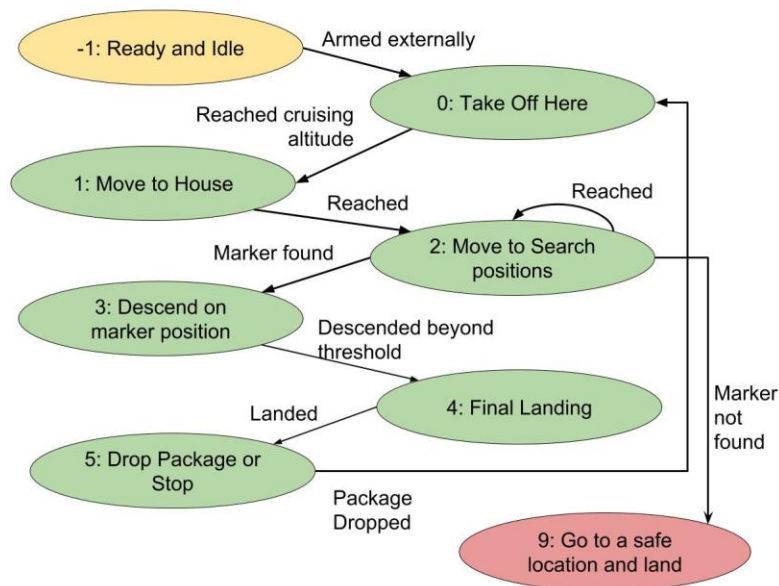


Figure 9: State Machine for Behavior control

As seen in Figure 9 above, the following states are used with the corresponding functionalities:

- State -1: Idle.
- State 0: Take off from current location.
  - Navigation is blocking until height is achieved.
  - Vision subsystem inactive.
- State 1: Move to Waypoint - House or Truck GPS position
  - Navigation is blocking until waypoint is reached.
  - Vision subsystem is inactive.
- State 2: Search for marker
  - Navigation is blocking on individual goals for the search path
  - Vision subsystem is active - Updated position goals are generated and override waypoints.
- State 3: Marker found. Descend.
  - Vision subsystem maintains position in X and Y.
  - Z position is reduced as and when the position is held well over the marker, until a threshold minimum height is attained
  - Navigation is non-blocking. Position updates from the marker detection system override waypoints.
- State 4: Final Descend/Landing.
  - Remembers the filtered marker location and continues descend on that position without continuous updates
  - Vision subsystem is inactive.
- State 5: Drop package.
  - Initiates drop off of the package
- State 9: Destination marker not available. Safe landing
  - Calls states 1 and 4 in emergency mode for reaching a predefined safe waypoint and safely landing there.

#### *7.1.1.2 Search Patterns*

A few search patterns were realized and tested. The formulations used were parametric with the following parameters:

- View area: The area seen by the camera. Values used: 2m x 2m (at 15m height)
- Area to be searched. The area to be searched around the GPS position of the house while the house is at the center. Values used: 8m x 6m.

The following search patterns were tested (figure 10):



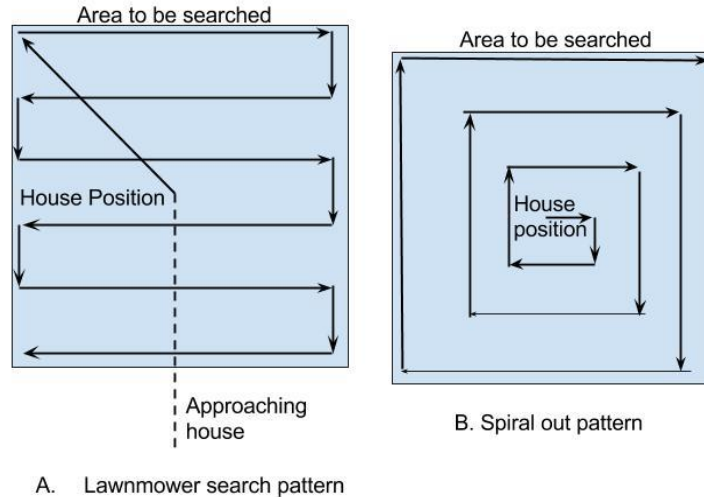


Figure 10: Search patterns tested for marker search

The lawnmower search pattern was finally used as it was robust and simple.

### 7.1.2 Flight Control Subsystem

Flight control was done using the Pixhawk flight controller [1], running the PX4 firmware. State estimation was done using the IMU, GPS, Compass, Barometer and Lidar lite range sensor.

Pixhawk internally uses feedforward and PID feedback control for waypoint following. Mavros [6] is a ROS layer over MAVlink, the communication link with Pixhawk. It exposes an API for controlling the UAV in the OFFBOARD mode. As seen in figure 6 (Software architecture), the Odroid runs this Mavros layer and uses the following API for controlling the UAV in different situations:

- **Position set-points:** Publish to ROS topic /mavros/setpoint\_position
- **Velocity commands:** Publish to ROS topic: /mavros/setpoint\_velocity/cmd\_vel
- **Odometry from UAV:** Subscribe to ROS topic: /mavros/local\_position/odom
- **Position from UAV:** Subscribe to ROS topic: /mavros/local\_position/pose
- **Actuator Control:** (for Gripper subsystem) Publish to /mavros/actuator\_control
- **Mode switching:** Command ROS service mavros/set\_mode

The following modes were used:

- **OFFBOARD:** For controlling positions/velocities using the Odroid.
- **Auto.LAND:** For autonomous landing.

### 7.1.3 Vision Subsystem

The vision subsystem is responsible for detecting the marker and guiding the UAV for a precision landing. It consists of a simple webcam, the Logitech C720 connected to the main processor, the Odroid XU4[3].

After conducting trade study between different markers, AprilTags were selected due to their precision detection, and reasonable speed. In the current state, the AprilTags were not directly usable as markers due to range and speed limitations. The following upgrades were made:

### 7.1.3.2 Range - Nested AprilTags

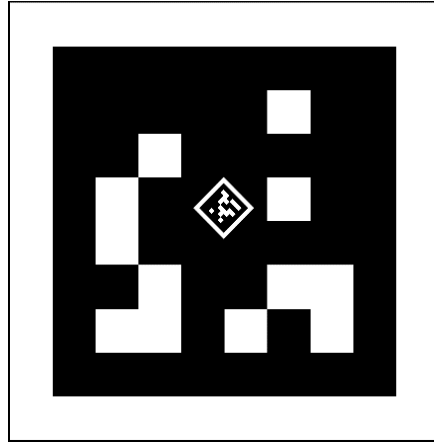


Figure 11: Nested AprilTag marker. Outer AprilTag can be seen from far-off distances, and the inner one from nearby distances. (Tag id 166 outer, 138 inner, rotated -45 degrees)

As the marker had to be detected from a high height (around 15-20m), the AprilTag marker needed the side length to be 58cm. But then, it was not possible to detect the marker below 2m height. As a result, we developed Nested AprilTag makers (Figure 11). They consist of a bigger outer marker with a certain Tag ID, and a smaller inner AprilTag with a different Tag ID. The inner tag is one-tenth the size of the outer tag and is rotated 45 degrees counter-clockwise, to ensure it does not interfere with the detection of the outer marker.

Different sizes of nested tags were tested to determine upto what ranges they can be detected.

S.No.	Detection distances for different nested April Tag markers			
	Outer April Tag		Inner April Tag	
	Size	Range	Size	Range
1	3.6cm	8 cm to 1.8m	0.36 cm	Not detected
2	14.4cm	40 cm to 7.2m	1.44 cm	4 cm to 50cm
3	57.5cm	1.6 m to 30m	5.75 cm	16 cm to 2m

Table 4. Table depicting tested size-range relationships for nested AprilTags.

Based on our requirements we chose the 57.5cm Nested April Tag for our application.

### 7.1.3.3 Speed - Detection to Tracking

The most important part of the vision system is the marker detection algorithm which detects the markers in the image. These algorithms needed to be robust to noise as the environment around the house may be cluttered, but they also need to be fast so that fast updates can be sent to the control algorithms after state estimation.

After trying multiple algorithms and markers, we finally settled with an AprilTag detection algorithm as developed by University of Michigan [7] and its C++ version as developed in MIT [8].

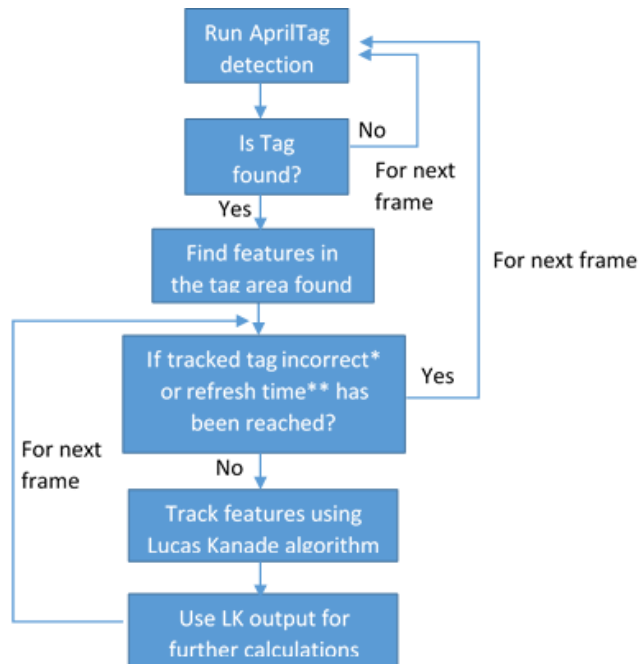


Figure 12. Flowchart showing the April Tag detection algorithm interspersed with the Lucas-Kanade tracking Algorithm

The April Tag detection gives up to 8 fps on the Odroid. This rate is too slow for controlling the UAV. As a workaround, we tried to combine the April Tag detection with Lucas-Kanade tracking algorithm. The basic idea of the same is illustrated in the flowchart [Figure 12].

We use the April Tag detection as the primary algorithm. After the first frame in which the tag is detected, the features were obtained from this frame and tracked in the following frames using the Lucas Kanade tracking. Currently, only the four corner points of the tag are used as features, as they are minimum set required to calculate the pose of the April Tag. The output obtained from the tracking results was be verified for correctness\*. In case no tag is obtained or the tag obtained is incorrect, we shifted back to the April Tag detection for the next frame. As tracking results may start deviating from the actual detections, it is a good idea to refresh the estimates using the April Tag detection once every few frames\*\*.

\*correctness of the tag can be verified in multiple ways: (the basic version has been tested to be a good enough measure of correctness)

- Basic: verify that the tracked points form a sensible quadrilateral.
- Advanced: also include using the decoding logic of AprilTags to verify the tag

\*\*use a refresh time (or number of frames) after which the full detection is run to refresh the tracking results. This is done as the tracking results can deviate due to errors and occlusions. A refresh every 30-60 frames gives a good output.

These algorithms were tested for speed on the laptop and the Odroid [3]. The speeds have been compared in Table 5.

Algorithm	FPS on Laptop (i3 4th gen)	FPS on Odroid (Quad core ARM)
April Tag detection	14	8
Lucas Kanade Tracking (Only tracking)	30	29
<b>Merged (LK + April Tag detection)</b>	<b>29</b>	<b>28</b>

Table 5: Table listing the comparison for individual algorithms with the combined algorithm

The resulting FPS can be taken up to around 29fps on the Odroid, which makes the state estimation and subsequent control possible. A few tracking results can be seen in these videos.

- Marker Detection Test #1 [<https://youtu.be/zJ2rNg4Q4Vc>]
- Marker Detection Test #2 [[https://youtu.be/0qn28RghF\\_o](https://youtu.be/0qn28RghF_o)]
- Marker Detection Test #3 [<https://youtu.be/5TkUyuMBI2E>]

#### 7.1.4 Obstacle Avoidance Subsystem

The flow chart (figure 13) shown below describes the functional blocks of the obstacle avoidance subsystem

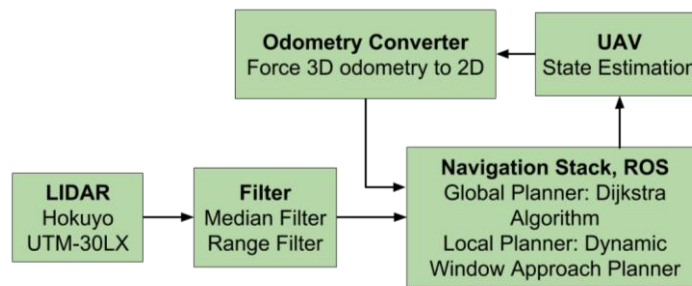


Figure 13: Flowchart depicting the functional blocks of the obstacle avoidance subsystem

##### 7.1.4.1 LIDAR

We use Hokuyo UTM-30LX LIDAR [4] for obstacle detection. It has a detection range of 30m and 270 degree scanning range.

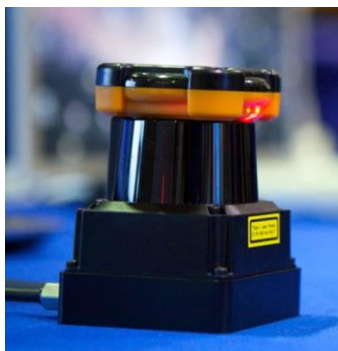


Image 1: Hokuyo UTM-30LX LIDAR. Image source: <http://electronicdesign.com>

#### 7.1.4.2 Filter:

In sunlight the lidar sensor picks up noise due to IR interference. This noise is detection of obstacles even when there are no obstacles in front of the sensor. As shown in figure 14, the red spots are noisy unfiltered output from the lidar. To get rid of this issue we use combination of the Median Filter and the Range Filter. The range filter discards the sensor readings beyond 4m distance. Median filter takes the median of 25 observations and helps reduce the noise. In figure 14, the 2 green dots are the filtered output of the lidar data.

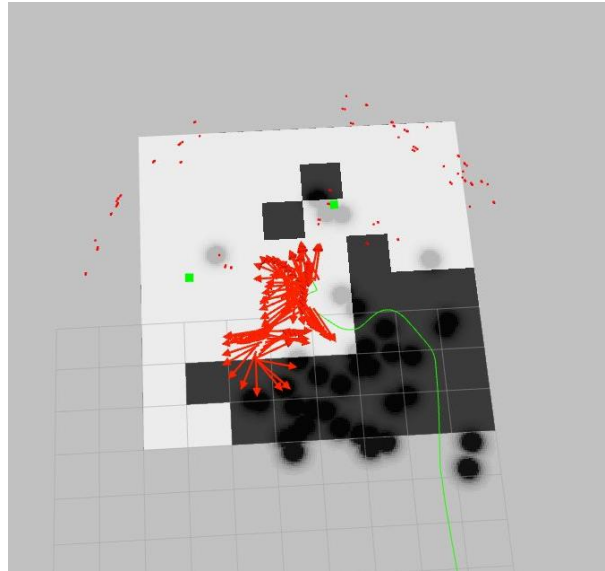


Figure 14: The figure shows the rviz visualization of one of the test run of the UAV. The red lines are the positions of the UAV. The black spots are the recorded obstacles in the cost map generated by the navigation stack as described in the next section. The red spots are the noisy unfiltered output from our previous indoor sensor. The 2 green dots are the filtered output. As seen the noise is greatly reduced by median and range filter.

#### 7.1.4.3 Navigation Stack, ROS

ROS Navigation Stack takes in information from odometry, sensor stream, start pose & goal pose to output safe velocity commands for the robot. ROS Navigation Stack is developed for 2D navigation. We tweaked it for our application. As a **prerequisite**, **navigation stack** takes following inputs:

- Coordinate Frame information using tf
- Sensor\_msgs/LaserScan message from LIDAR
- Odometry information using tf and nav\_msgs/Odometry messages

As an output the stack gives velocity commands on /cmd\_vel topic. The navigation stack contains the following parts:

- **Local costmap & Global costmap** - Navigation stack maintains two costmap.
- **Global path planner** – It uses the global costmap to compute paths ignoring the kinematic and dynamic vehicle constraints. It uses Dijkstra's algorithm to do this.

- **Local path planner** – It accounts for the kinematic and dynamic vehicle constraints and generates feasible local trajectories in real time while avoiding obstacles using the local costmap. We use Dynamic Window Approach local planner.
- **Move\_base** – Implements the state machine

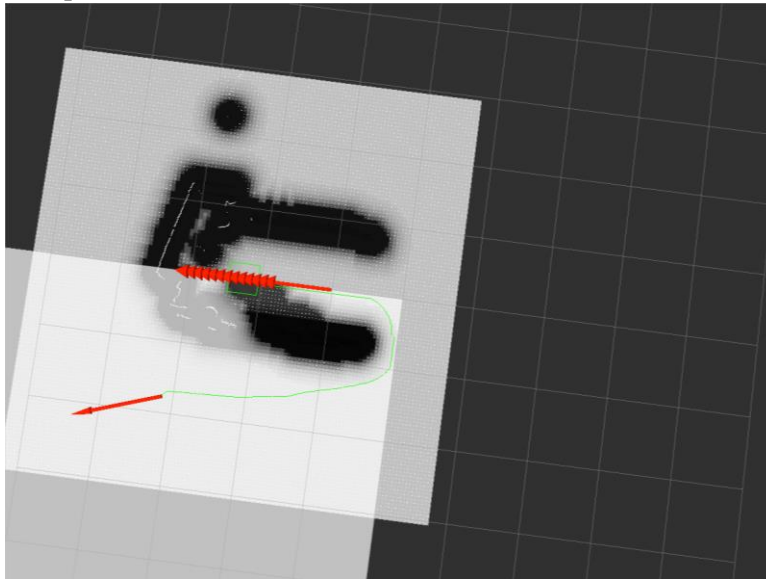


Figure 15: The figure show the rviz visualization of the navigation stack. The back spots depict the obstacles in local and global costmap. The red lines are the positions of the UAV. The green line is the global plan generated by the navigation stack. Local plan is not clearly visible in the figure as it is a very small path near the UAV base generating plan for the next 2/3 seconds for the UAV.

Navigation stack has lot of parameters to tune. These parameters include robot configuration parameters - maximum, minimum linear velocities, rotational velocities, linear acceleration, rotational acceleration, goal tolerance parameters - path distance bias, goal distance biases, trajectory scoring parameters, raytracing and others [14]. We tuned these parameters for our system.

#### *7.1.4.4 UAV State Estimation*

UAV's flight controller-Pixhawk[1] gets velocity commands from the navigation stack running on the odroid through the Mavros [6]. Pixhawk passes velocity commands to the motors and runs its own state estimation EKF filter to estimate its current position. This odometry data is then passed to the Odometry Converter.

#### *7.1.4.4 Odometry Converter*

The Pixhawk outputs 3D odometry data. As mentioned above the Navigation Stack is designed for 2D navigation and requires 2D odometry data. This means that the robot should always have  $z=0$  or it doesn't generate correct velocity outputs. Hence we use a ros node which receives 3D odometry data from pixhawk and converts it into 2D odometry data.

### 7.1.5 Gripper Subsystem

The gripper subsystem is responsible for holding the package firmly while it is being delivered. It should be able to engage to grip the package and disengage to let it drop.

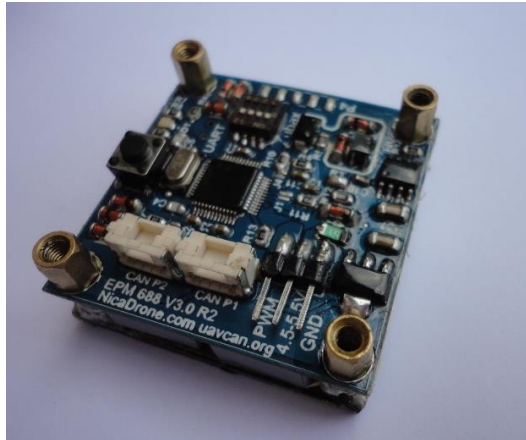


Image 2: NicaDrone Electro-Permanent Magnet



Image 3: Package

The NicaDrone Electro-Permanent magnet (EPM) [2] is used as the gripper (Image 2). This is an electro-permanent magnet that can engage a permanent magnet to stick to a package with a small steel plate on the top. (Image 3). When disengaged, the magnetic field is removed, letting the package fall down due to gravity.

The EPM is controlled through the pixhawk, using the Actuator control mode through mavros. A simple command to engage the EPM needs to be sent when attaching the package, and another similar command to disengage.

## 7.2 Subsystem Modelling, Testing and Analysis

### 7.2.1 Flight Control Subsystem

The accuracy of the system primarily depends on the state estimation, which relies heavily on the GPS position estimates for the UAV.

Based on our testing, the health of the GPS depends on the area of flight. Areas surrounded with buildings usually have degraded GPS availability and hence lower accuracy in state estimation. After taking a few minutes to gather GPS connections, in a clear environment, the state estimation was accurate up to 60cm.

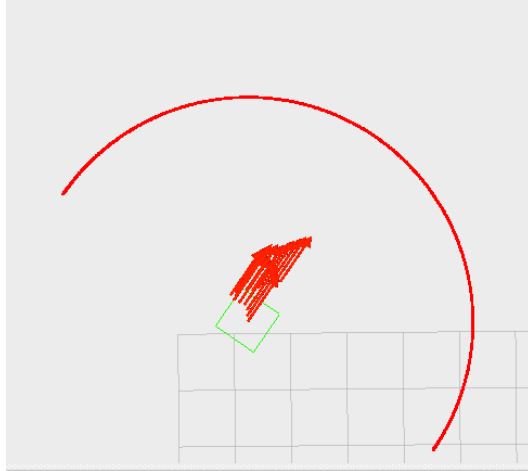


Image 4: shows the deviations in the position of the UAV while trying to hold position.

### 7.2.2 Vision Subsystem

The testing system was tried in multiple conditions to test for robustness and error. Following were some of the inferences drawn:

- With different lighting conditions, exposure adjustments are required. Automatic exposure control is unable to handle cases with direct sunlight on the marker.
- Marker is detected and tracked with high precision in all orientations (roll, pitch and yaw).
- The primary constraint for finding error in X,Y,Z values has been the calibration of the setup to obtain ground truth.
- Errors in X,Y,Z offsets were recorded and plotted (Figure 16). An error of <5% from the distance to the marker is seen in each X,Y and Z. These results exceeds the expectations set in the requirements (10%).

As viewed from the camera, X axis is the camera view axis, Y axis is sideways axis (longer side in the image captured) and Z axis is the upwards axis (shorter side in the image captured).

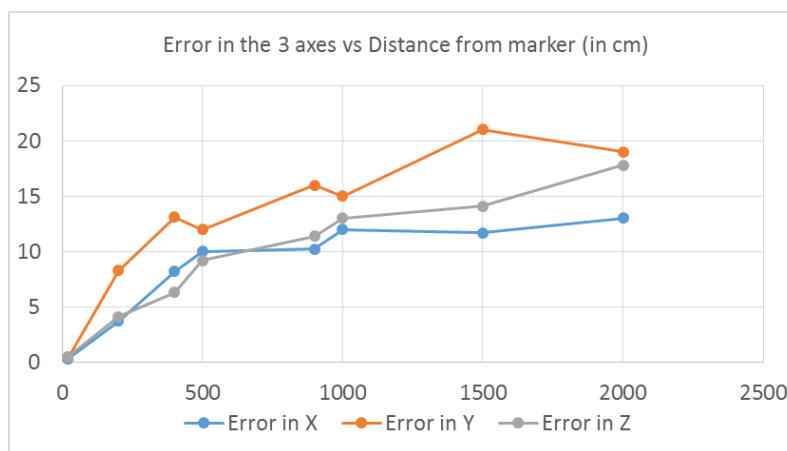


Figure 16: Graph depicting error in detection distances to Marker



### 7.2.3 Obstacle Avoidance Subsystem

The requirement of our project is for the UAV to avoid obstacles of size 1.5m X 0.5m.

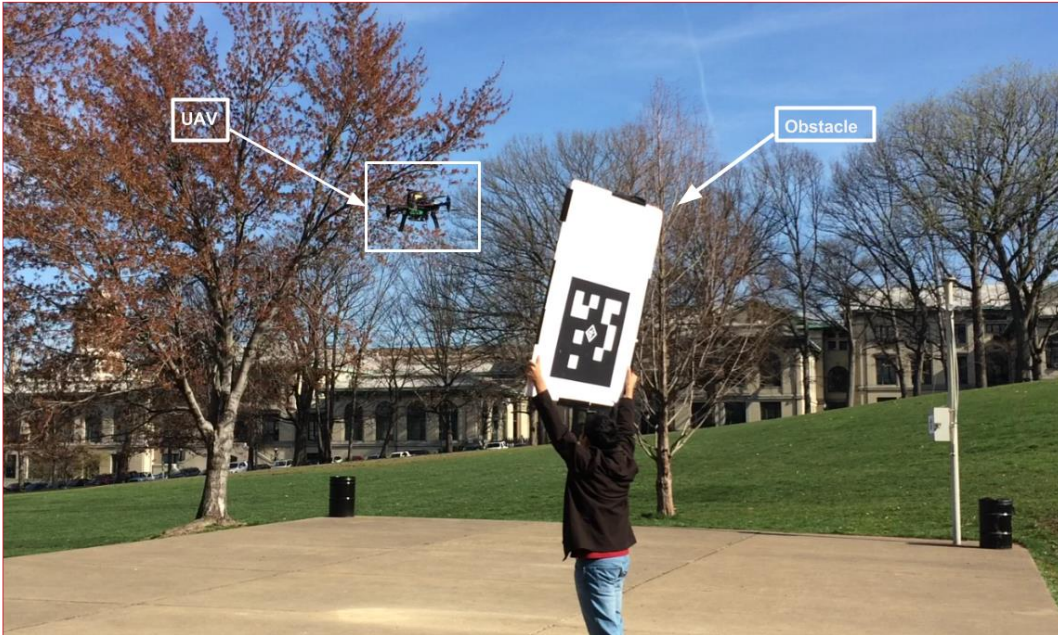


Figure 17: The figure shows the UAV and a test obstacle brought in front of it.

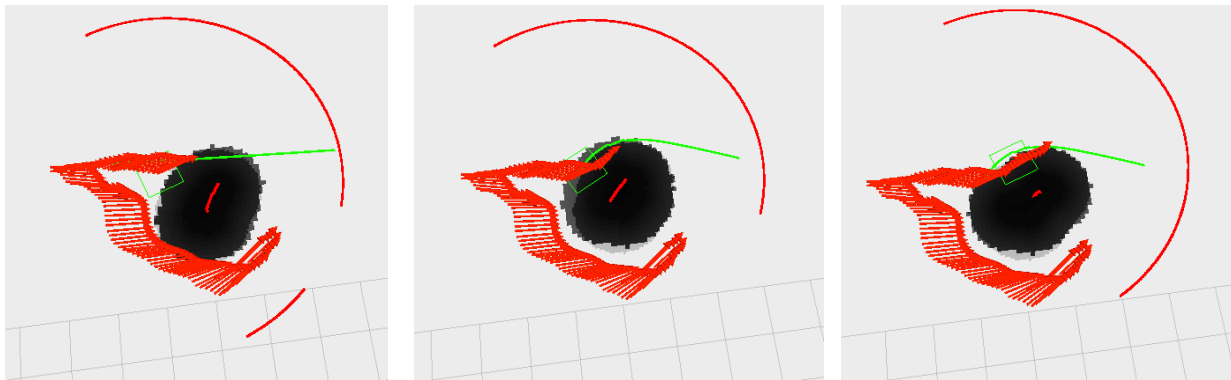


Image 5: Rviz visualization of the UAV avoiding obstacles

For the various tests that we carried out this are the results we obtained:

Attributes	Obtained result
Minimum distance maintained between the UAV and obstacles	80 cm
Distance at which UAV starts to avoid obstacle	100 cm

Table 6: Obstacle Avoidance test results

### 7.3 System Performance Evaluation against Spring Validation Experiment

Full system demonstration video can be seen here <https://www.youtube.com/watch?v=vT5HnfHKzuY>

#### 7.3.1 Package delivery without obstacles

A 9.5” x 6.5” x 2.2” package weighing 200g was delivered 30cm from the center of the marker. The UAV took off from a starting position around 25m away from the house and landed back on a the truck position another 20m away from the house.

Performance Criterion	Required	Achieved	Fulfilled?
P.1: Package drop accuracy.	Less than 2m	30cm	Yes
P.2: UAV flight time	At least 10 minutes	10 minutes	Yes
P.3: Package weight requirements	At least 100g	200g	Yes
P.4: Package size requirements	9.5” x 6.5” x 2.2”	9.5” x 6.5” x 2.2”	Yes

Table 7: Performance comparison for system performance requirements (P5-P7 were environment requirements)

#### 7.3.1 Package delivery without obstacles

A 9.5” x 6.5” x 2.2” package weighing 200g was delivered 80cm from the center of the marker. The UAV took off from a starting position around 12m away from the house and landed back on a the truck position another 20m away from the house.

Performance Criterion	Required	Achieved	Fulfilled?
P.1: Package drop accuracy.	Less than 2m	80cm	Yes
P.2: UAV flight time	At least 10 minutes	10 minutes	Yes
P.3: Package weight requirements	At least 100g	200g	Yes
P.4: Package size requirements	9.5” x 6.5” x 2.2”	9.5” x 6.5” x 2.2”	Yes
P.8 Obstacles are avoided.	Size 0.5m x 1.5m	Avoided	Yes

Table 8: Performance comparison for system performance requirements (P5-P7 were environment requirements)

## 7.4 System Strengths and Weaknesses

Overall system performance is strong and robust. Package delivery without obstacles is stable and repeatable. Obstacle avoidance has issues related to field of view.

Detailed strengths and weaknesses of the subsystems are listed in table 9.

Subsystem	Overall	Strengths	Weaknesses
Vision	Strong	Precise and Accurate	Manual exposure control
Behaviour	Strong	Robust	
Flight Control	Strong	Accurate and Robust	Drift during yaw velocity control
Gripper	Strong	Firm and quick	Package blows away after drop
Obstacle Avoidance	Neutral	Responsive and accurate	Low field of view. May hit obstacles not in view.

Table 9: Strengths and Weaknesses of the subsystems

## 8. Project Management

The following section outlines the high-level Work Breakdown Structure and schedule. For this project, we made a concerted effort to integrate existing technologies wherever appropriate. To ensure project success, great attention has been given to integration testing leading toward a full scenario test. The work for this project has been broken down at the highest level into Systems Engineering, Fabrication and Procurement, Systems Integration, and Testing.

As you can see, the system is nearly complete with items in blue deliberately eliminated. Work has started on both the drone and obstacle avoidance has started. Team A has yet to start work sections related to the gripper, user interface, and final integration.

Significant revisions were made to the project at the start of the Spring Semester due to changes in personnel and a re-evaluation of priorities. The largest change was a reduction in quality -- define as "the degree to which requirements are met" -- and a minor reduction in scope. In order for focus better on the true intent of the project, the Graphical User Interface and VTOL were eliminated. We shifted to the better-supported 3DR X-8+ and to the integration of subsystems.



## 8.2 Schedule

The following schedule was made using a Gantt chart and our best estimates of both development time and system dependencies.

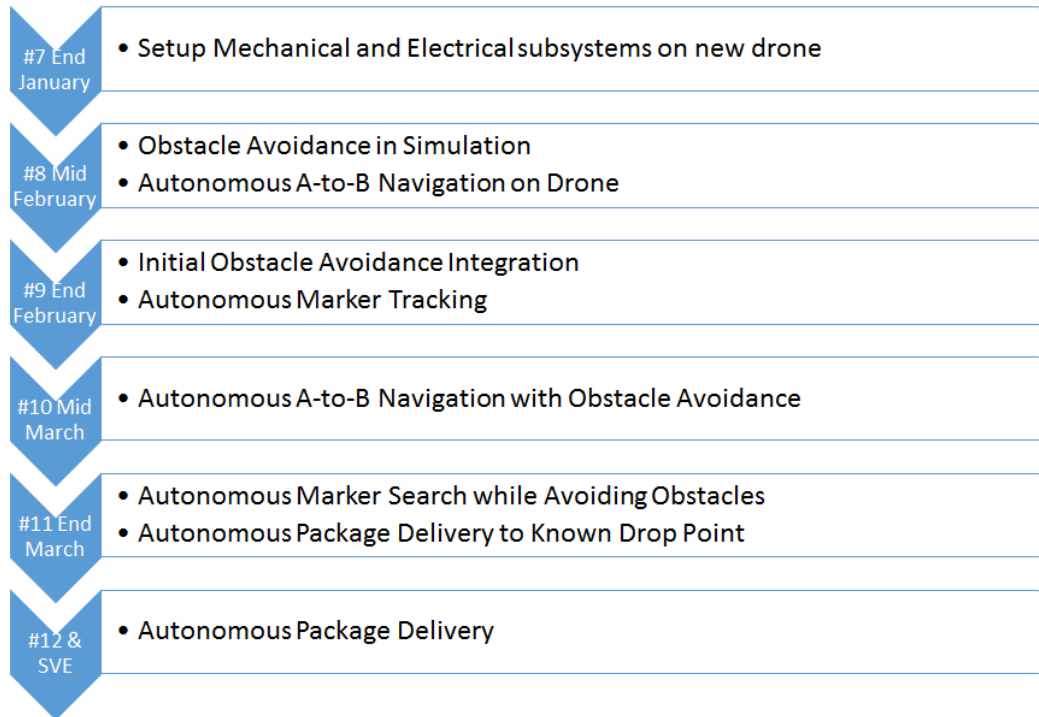


Table 10: Project Schedule of Deliverables

This schedule was a refinement on the original, developed during the fall semester. Team A held themselves to this strict timeline until week #10. The team wasn't able to complete Obstacle Avoidance due to delays from weather and numerous crashes. Thanks to appropriate risk management, the team was able to repair the drone and recover the timeline the following week.

## 8.3 Risk Management

The Team identified 3 risks to the project:

1. The Drone may not Locate the Marker
2. Lack of Testing Opportunities due to Weather
3. Possible trouble Integrating the Navigation Stack with the Pixhawk

### 8.3.1 Marker Search Algorithm not Robust

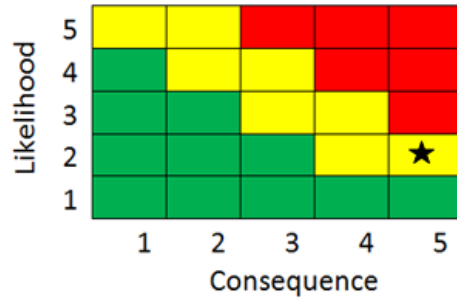


Figure 18: Risk Matrix for Risk 1

This technical risk is a real-world consideration. It’s important that the possibility that the drone is unable to find the marker in vicinity of the house be considered. The consequence is that the drone will be unable to deliver the package.

This risk was mitigated by having the drone return to a known-good position following completion of the search pattern. Additionally, we experimented with different search parameters to ensure that the marker was discovered. One parameter that we didn’t anticipate, but eventually tuned was the exposure setting on the camera. Light and shadows in different areas and at different times of day behaved differently.

This risk was successfully minimized using the above techniques.

### 8.3.2 Weather Limiting Testing Opportunities

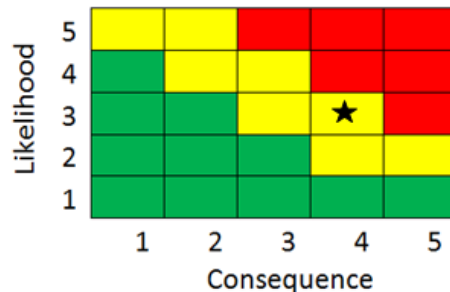


Figure 19: Risk Matrix for Risk 2

Due to the decision to operate outdoors, this operational risk was realized several times during the project. The largest impact was delayed testing and integration since the system had to be tested were there was a GPS signal. We mitigated this by testing early and often. Additionally, we paid close attention to weather reports in order to anticipate windows for testing.

An unexpected benefit of dealing with weather also forced the team to develop robust algorithms. This benefit was shown during the SVE and SVE encore when wind gusts of up to 15 MPH almost pushed the drone off the marker. The marker tracking algorithm correctly compensated for the wind, allowing the package to be successfully delivered.

### 8.3.3 Integration of Navigation Stack with Pixhawk



Figure 20: Risk Matrix for Risk 3

The final, significant risk to the project that the team anticipated involved integrating the ROS Navigation Stack with the Pixhawk. Our concern was that this process would take too long and delay integration of additional subsystems. To combat this, the team prioritized its integration early in the schedule. Back up to this, was proper mission planning to eliminate the need for obstacle detection. However, thanks to its prioritizing integration and frequent testing, the backup wasn't needed.

### 8.4 Budget

Item	Price
<b>Electro Mechanical System</b>	\$ 2,706.67
Firefly 6 UAV and Spares	\$ 977.98
3DR X8+ and spares	\$ 1,079.68
Spare Electronics	\$ 545.01
Nica Drone Electro Permanent Magnet	\$ 104.00
<b>Vision System</b>	\$ 422.88
Webcams	\$ 29.93
Odroid and accessories	\$ 219.95
PIX4FLOW kit (optical flow) and SD cards	\$ 173.00
<b>Obstacle Avoidance System</b>	\$ 555.00
Lidar-Lite v2	\$ 115.00
MaxBotix Ultrasonic Rangefinder	\$ 440.00
IR sensors	\$ -
Hokuyo Lidar	\$ -
<b>Test Environment</b>	\$ 167.74
<b>Reimbursement</b>	\$ 20.00
Sean -- wood/home depot	\$ 13.00
Adam -- amazon connectors	\$ 7.00
<b>Total</b>	\$ 3,872.29

Table 11: Project Budget

Even after switching to a new platform, our expenditure was below the maximum budget allotted (\$ 4000).



## 9. Conclusion

Numerous changes were made during the Spring Semester. Through it all, Team A was able to deliver on the original intent of the project. The majority of the progress of the project was completed during this last semester. Along the way, we learned several key lessons and developed ideas for further development of the platform.

### 9.1 Lessons Learned

Several key lessons were identified during the project's life cycle. They include simplifying the platform and components early, important considerations for outdoor drone projects, the importance of spares, and how to reduce scope while maintaining a project's intent.

#### 9.1.1 Simplify

The need for simplification was seen repeatedly. In fact, reevaluation of a project is continuous. Over the last two semester, the team was forced to decide between different tradeoffs. Through these decisions, a trend emerged in going with more proven or more reliable subcomponents. The lesson is not that complicated components should be avoided. Rather, we realized that integration was itself a difficult challenge and fighting unwieldy subsystems only multiplied this difficulty.

We realized that the FireFly6 is the weakest subsystem of our project at this point in time. The UAV is also the most important subsystem of our project and must be made operational as soon as possible. Due to this realization, we are contemplating as part of our risk mitigation to change platforms entirely and go with an octocopter capable of doing everything the FireFly6 does just at slower speeds and with less flight time. Cutting our losses and modifying our project was the best thing for our project long-term.

On the Obstacle Avoidance end, using 14 ultrasonic sensors proved to be imprecise and noise-prone. These were scrapped in place of a Hokuyo Lidar due to its superior accuracy and ease of integration. The numerous sensors, while within the tolerances of our system requirements, were heavy and easily fooled. By eliminating them, we were able to better focus on navigation and obstacle avoidance.

These are just two examples of when the choice to simply, without cutting corners, allowed for greater progress of the system as a whole. Future teams are strongly encouraged to get systems streamlined from the start since integration itself is complex. Finally, by focusing on integration, teams will have a greater time to evolve their systems as they learn the technical details they couldn't have anticipated from the start.

#### 9.1.2 Outdoor Drone Considerations

Team A made the choice to operate outdoors. We stand by this decision as it was rewarding to operate in a real-world environment. However, it came at a significant risks and opportunity costs which weren't evident at the start.

The first challenge to the team was the current regulatory environment in the United States. In fact, FAA and local regulations changed midway through our project. The team was able to adapt to a degree, but this should be weighed by future teams before electing to operate drones outdoors.



Team A interacted with the University's Office of Risk Management and Insurance in addition to Legal Counsels. We were offered assistance to submit for a Section 333 Waiver with the FAA issuing a joint "blanket" Certificate of Waiver of Authorization (COA) for operating under 400 feet. Details can be found on the [FAA's website](#) about how to file a petition.

However, the expected turnaround time is 120 days normally, but that has been lengthened due to excessive applications caused by the FAA's new rulings. With less than 20 days till SVE, It is unlikely that we would receive authorization. Rather than waste resources, Team A has made alternate arrangements and will be documenting the appropriate process for next year's class. The notes below come directly from Daniel Munsh:

As of April 2016, there are four general government requirements for non-recreational (called "civil") outdoor drone operation:

1. The aircraft must be registered with the FAA;
2. The aircraft must have a Section 333 Exemption from the FAA (this exempts the drone from the FAA regulations for traditional aircraft that would otherwise apply);
3. The aircraft must have a COA from the FAA (this authorizes the specific parameters for flight operations). Section 333 Exemptions automatically come with a "blanket" COA for operations under 400 feet.
4. Comply with any state or local laws regulating drones. For example, the City of Pittsburgh prohibits drone operations in public parks. So even if you have all the proceeding items from the FAA, teams still can't operate in city parks in Pittsburgh.

Important points of contact for next year's class include Daniel J. Munsch, AVP and Assistant General Counsel ([dmunsch@andrew.cmu.edu](mailto:dmunsch@andrew.cmu.edu)) and Diane Patterson, Senior Risk Management and Insurance Specialist ([dianep@andrew.cmu.edu](mailto:dianep@andrew.cmu.edu)).

The current legal landscape is expected to change sometime in calendar year 2016. Future classes should reference [CMU's Office of the General Counsel's Page on Drones](#). Separately, the Senate is considering a proposed law, the [Higher Education UAS Modernization Act](#), that would create a separate regulatory structure for drone operations related to research at institutions of higher education. Finally, teams should also be aware that there is a blanket prohibition of drones in the city's parks, which eliminates nearby Schenley Park as a test area.

The next challenge to operating outdoors is in mitigating the effects of the environment. Team A lost significant testing time due to weather and use of green spaces around campus.

First, weather is unpredictable and has the greatest impact on testing time. This was less of a concern early in the process when individual subsystems are being developed and tested. The effect worst during the spring semester with the cold and heavy snow early in the semester followed by wind and rain in the later months. These limit testing of the system at exactly the time that it is needed most. As our team demonstrated, these obstacles are able to overcome, but require preparation and tenacity. Teams will need to monitor weather forecasts and plan their schedules accordingly.

Additionally, events are often hosted on the "Mall" and "Cut" on campus. Teams will need to check that areas are clear of crowds and that passers-by be redirected around the testing area for their own public safety. Caution tape and stakes were sufficient for the latter purpose, but this is an additional time constraint. Furthermore, if teams are able to register time on the stadium, then they have a controlled

environment to work in. The point of contact for the stadium is Sara Gauntner ([sarap@andrew.cmu.edu](mailto:sarap@andrew.cmu.edu)). This isn't the only time constraint, which is our final consideration.

Teams seeking to operate outdoors, drones or not, need to be aware that travel and setup time come with significant opportunity costs. The time and effort in transporting robots and supporting materials back-and-forth should not be overlooked. Teams are encouraged to develop a field kit and keep it staged. Discipline is required in ensuring that any item removed from the kit is returned in order to prevent wasted time travelling back to the lab. The ease of transport for these kits is another factor. Weight should be minimized while still allowing for possible contingencies. Potential items for kits include spare parts, extra batteries, extension cords, tools, crowd-control materials, and the test environment itself. These will grow as the project develops.

The time taken is a substantial opportunity cost. For our team, it was rewarding to operate in real-world conditions and not be confined to a netted cube in the basement. We do wish to arm future teams with the knowledge of the trade-offs involved in operating drones outdoors.

## 9.2 Future Work

Our team began our project with a specific vision for its application. We see the project as commercially viable if specific supporting infrastructure could be implemented. Our project is predicated on the integration with delivery trucks. For this to work, both a physical interface between the drone and truck, and a user interface for drivers.

The physical interface between the drone and truck itself would warrant another MRSD project. The loading of packages, charging (or replacement) of batteries, and communication link between truck and drone are a worthy challenge. We would need to leverage the system of staging packages used by delivery companies. There is a specific order in loading trucks that could be leveraged to simplify this problem. Packages are currently loaded from front-to-back and top-to-bottom in order of delivery. However, what the physical system of landing, loading, and launching drones would like like is unknown at this time. This is the next major challenge for our application.

Additionally, since we eliminated the GUI from our project, this needs to be built in order to ensure the utility of the project. Currently, the drone needs to be reinitialized during each battery change since we elected to not use predefined coordinates and don't have a system of passing that information to the drone.

A GUI is technically simple, but the user experience should be paramount to ensure its adoption and use commercially. Ease of adoption by delivery truck drivers would expedite its adoption commercially by the large organizations we targeted.

During the 2015-2016 school year, we successfully implemented the core technology required to deliver packages by UAV. The team may release this as an open-source project in order to jump-start similar projects. We still believe strongly in our original application, of leveraging existing package-delivery networks, and believe that with additional development (and regulatory permission) that this goal could be achieved.

## 10. References

- [1] Meier, L.; Tanskanen, P.; Fraundorfer, F.; Pollefeys, M., "PIXHAWK: A system for autonomous flight using onboard computer vision," in Robotics and Automation (ICRA), 2011 IEEE International Conference on , vol., no., pp.2992-2997, 9-13 May 2011
- [2] Web link: "Electro Permanent Magnet" from Nica Drone, [http://nicadrone.com/index.php?id\\_product=59&controller=product](http://nicadrone.com/index.php?id_product=59&controller=product), accessed 10/2/15
- [3] Web link: "Odroid" from HardKernel [http://www.hardkernel.com/main/products/prdt\\_info.php?g\\_code=G143452239825](http://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825)
- [4] Web link: Hokuyo UTM-30LX [https://www.hokuyo-aut.jp/02sensor/07scanner/utm\\_30lx.html](https://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html)
- [5] Edwin Olson, "AprilTag: A robust and flexible visual fiducial system" in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2011
- [6] Web link: "MAVROS - Ros node for MAVlink communications" <http://wiki.ros.org/mavros>
- [7] Web link: AprilTags developed at university of Michigan <https://april.eecs.umich.edu/wiki/index.php/AprilTags>
- [8] Web link: AprilTags C++ library from MIT, <http://people.csail.mit.edu/kaess/apriltags/>
- [9] Web link: "Precision Land ArduCopter Demo", [http://diydrone.com/profiles/blogs/precision-land-arducopter-demo?xg\\_source=activity](http://diydrone.com/profiles/blogs/precision-land-arducopter-demo?xg_source=activity), accessed 10/2/2015
- [10] Web link: "3D Robotics Partners with Intel, Develops New Drone Power", <http://3drobotics.com/3d-robotics-partners-intel-develops-new-drone-power/>, accessed 10/2/15
- [11] Web link: "Petitioning for Exemption under Section 333", [https://www.faa.gov/uas/legislative\\_programs/section\\_333/how\\_to\\_file\\_a\\_petition/](https://www.faa.gov/uas/legislative_programs/section_333/how_to_file_a_petition/), accessed 5/5/2016
- [12] Web link: "CMU's Office of the General Counsel's Page on Drones", <https://www.cmu.edu/ogc/drones/>, accessed 5/5/2016
- [13] Web link: "Sen. Moran Introduces Legislation to Support Educational Use of Unmanned Aircraft Systems", [http://www.moran.senate.gov/public/index.cfm/news-releases?ContentRecord\\_id=608B2DA3-7434-4C04-B69F-D38DD32FFA5C](http://www.moran.senate.gov/public/index.cfm/news-releases?ContentRecord_id=608B2DA3-7434-4C04-B69F-D38DD32FFA5C), accessed 5/5/2016
- [14] Web link: "DWA Local Planner ROS" [http://wiki.ros.org/dwa\\_local\\_planner](http://wiki.ros.org/dwa_local_planner)