2015

Critical Design Review

UAV PACKAGE DELIVERY SYSTEM

TEAM AVENGERS

Tushar Agrawal	(Software Lead and Project Manager)
Sean Bryan	(Mechanical and Communications Lead)
Pratik Chatrath	(Sensor Lead and Software Developer)
Adam Yabroudi	(Systems Engineer and Electrical Lead)

MASTER OF ROBTIC SYSTEMS DEVELOPMENT, FALL 2015 | FRIDAY, OCTOBER 2, 2015

Contents

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.

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.

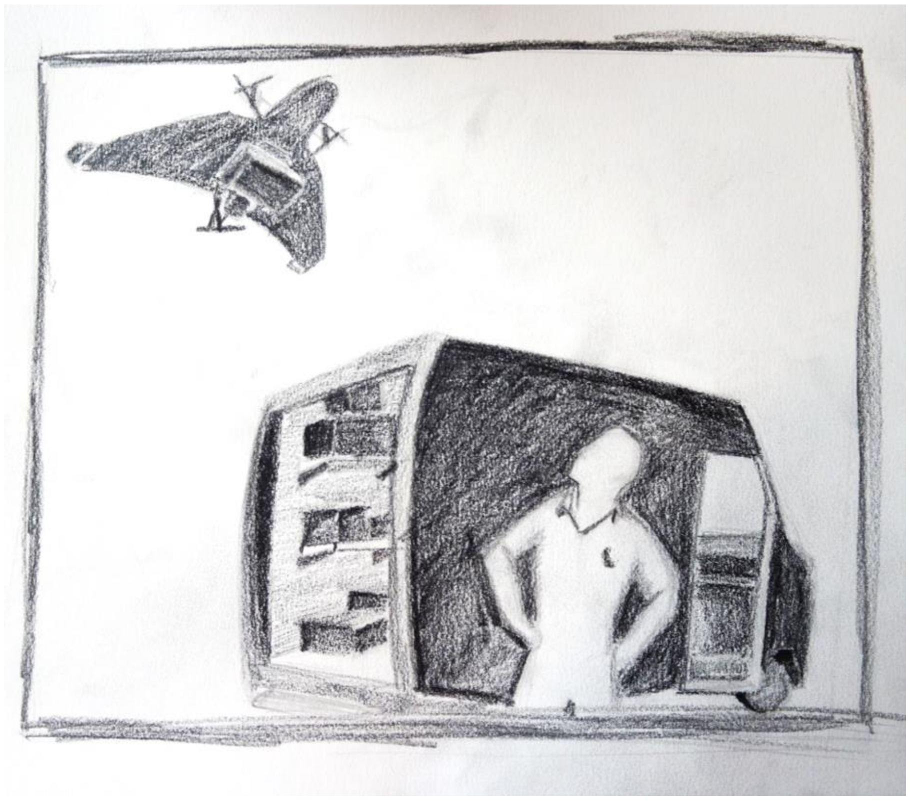


Figure 1: Artist Rendition of Sam and one of his drones

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.

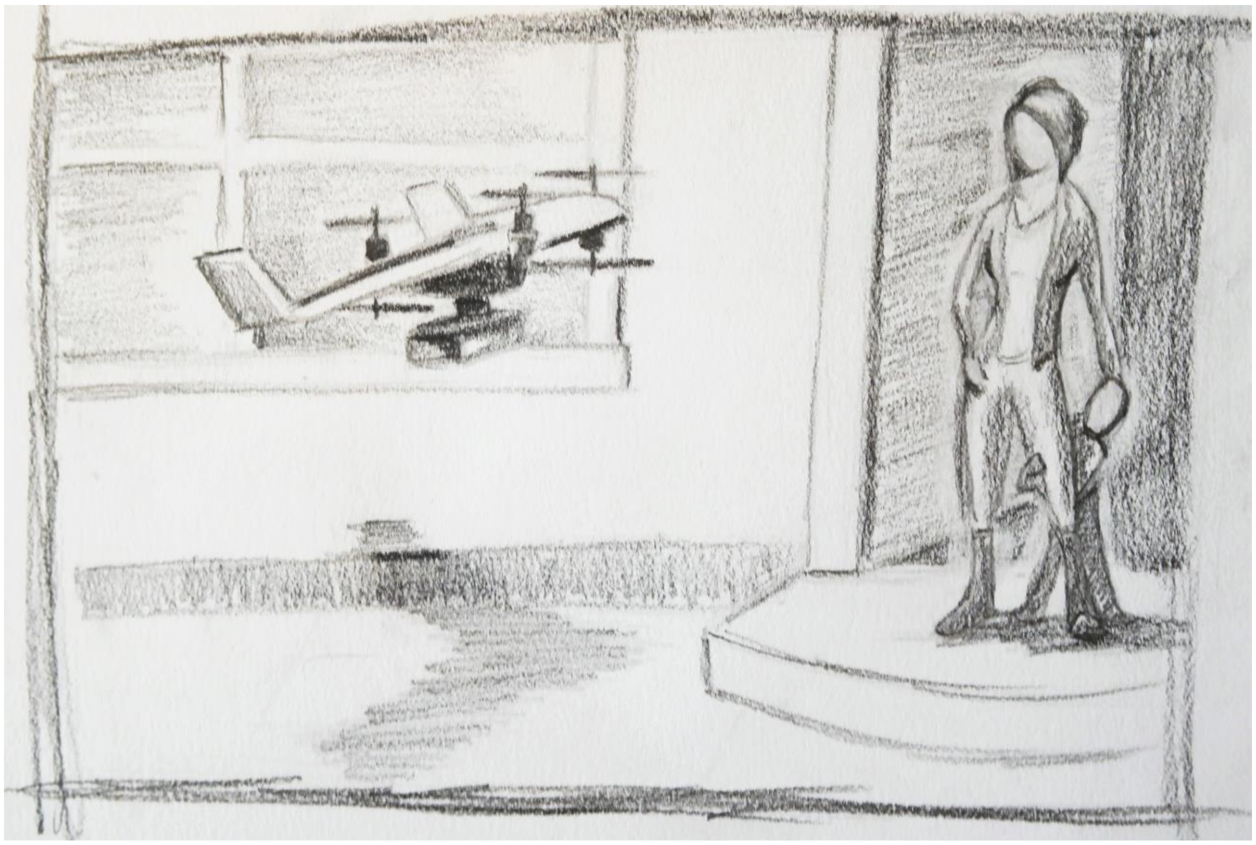


Figure 2: Artist's Redition of Drone Delivering Packages to End Customer

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 to 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 in the area, the drone will deliver the package. The customer will never know there was a problem, and the delivery company saves money.

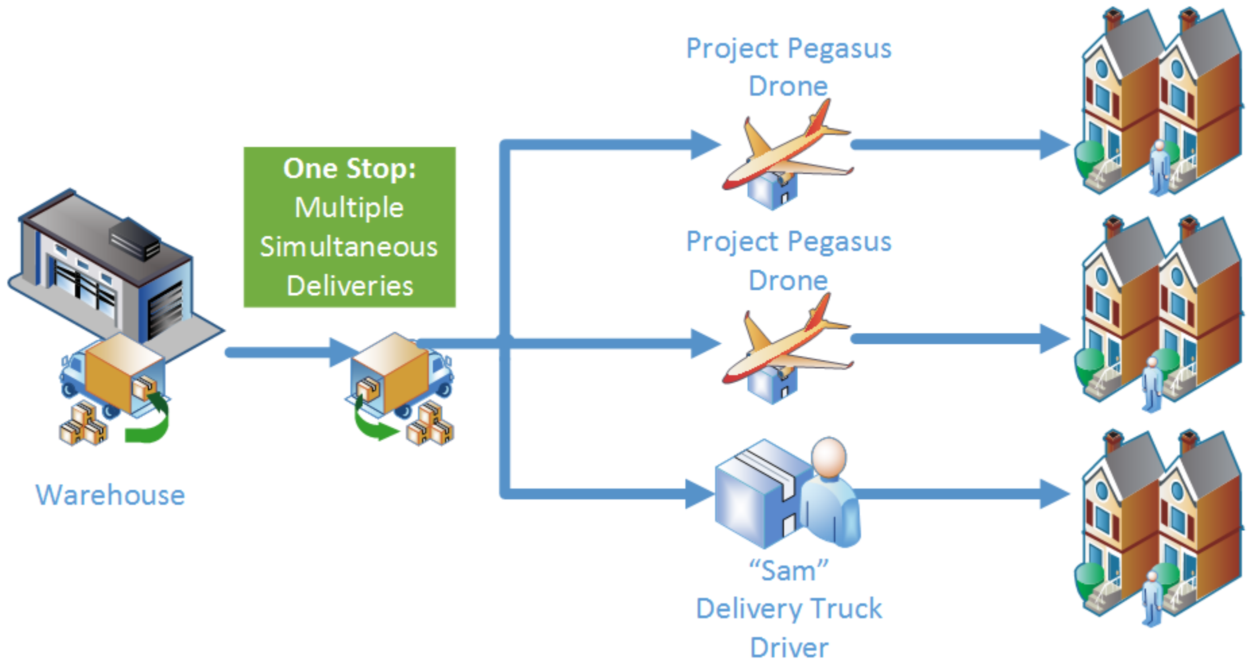

Project Pegasus Drone
One Stop: Multiple Simultaneous Deliveries
Project Pegasus Drone
Warehouse
"Sam" Delivery Truck Driver

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.

M.F.8 Takes coordinates as input from the user.

M.F.9 Communicates with platform to receive GPS updates (intermittently).

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 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.

M.N.5 Sub-systems should be well documented and scalable.

M.N.6 UAV should be small enough to operate in residential environments.

M.N.7 Package should weigh at most 400g and fit in a cuboid of dimensions 30cm x 30cm x 20cm.

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.

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

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 400g.

P.4 UAV carries packages that fit in a cube of 30cm x 30cm x 20cm.

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.

P.9 An edifice with a minimum cross section of 8m x 5m is required to navigate through.

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 upto 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 20cm.

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

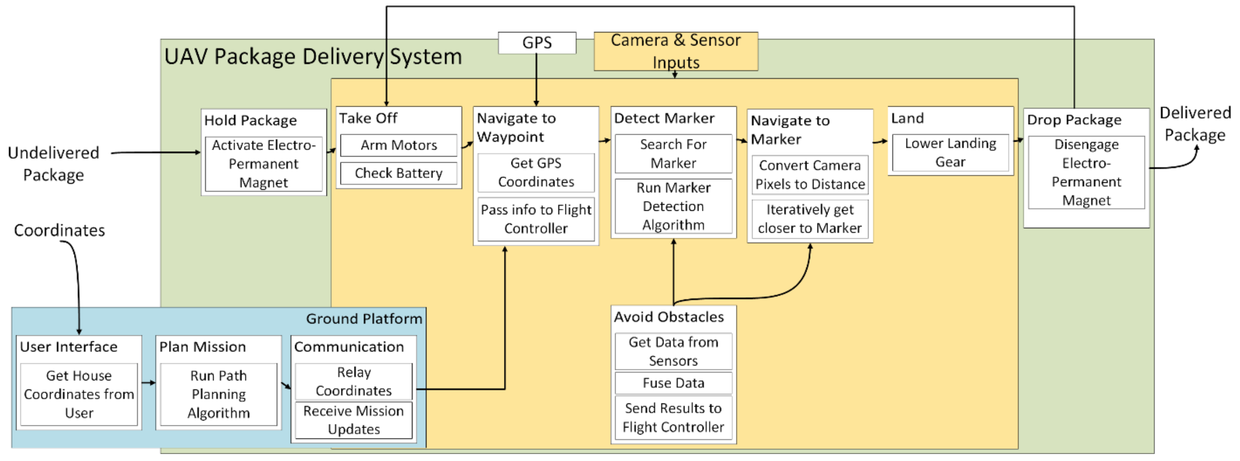


Figure 4: Functional Architecture Diagram

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

Looking in 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 path to destination. This information is then relayed to the UAV by the communication interface. The plan mission software continuously receives the current coordinates from UAV and send updated coordinates back to the UAV.

Meanwhile the UAV checks the battery status. If there is sufficient battery UAV arms the motor and takes off. UAV navigates using the waypoint to the vicinity of the destination using GPS input. It then switches to the marker detection code. 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 marker in image to actual distance of the UAV from the marker. The UAV continuously receives this information and moves towards the marker. UAV finally lowers it landing gear and lands. It drops the package by disengaging the electro-permanent magnet and flies back to base station using waypoint navigation.

During 'Detect Marker' &' Navigate to Marker' function UAV continuously runs an obstacle avoidance algorithms 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. Cyberphysical Architecture

The high level cyberphysical architecture can best be understood by Figure 3. 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.

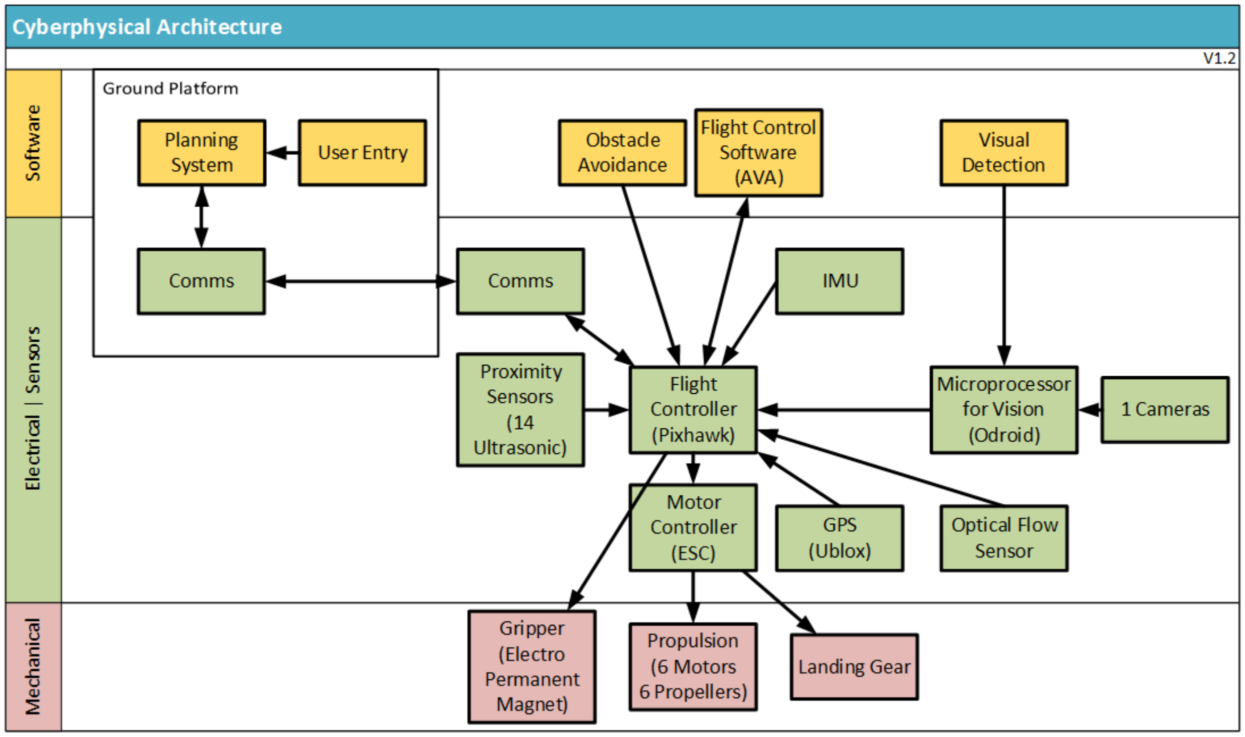

Cyberphysical Architecture
V1.2
Software
Electrical | Sensors
Mechanical
Ground Platform
Planning System
User Entry
Comms
Comms
Obstacle Avoidance
Flight Control Software (AVA)
Visual Detection
IMU
Proximity Sensors (14 Ultrasonic)
Flight Controller (Pixhawk)
Microprocessor for Vision (Odroid)
1 Cameras
Motor Controller (ESC)
GPS (Ublox)
Optical Flow Sensor
Gripper (Electro Permanent Magnet)
Propulsion (6 Motors 6 Propellers)
Landing Gear

Figure 3: Cyberphysical Architecture Diagram

5.1 Mechanical System

We are using FireFLY6 UAV for our project. The mechanical system of our project consists of the landing gear, the propulsion system, and the gripper. The landing gear and propulsion system are part of the FireFLY6 kit that we purchased but must be controlled appropriately by our software. The gripper is NicaDrone - an electro-permanent magnet. 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.

5.2 Electrical System

The electrical system is composed on a high level by flight control board, the vision subsystem hardware, sensors, and communication hardware. The flight controller is Pixhawk the brain of the entire system and runs all critical flight control software. The flight controller interacts with all the sensors on the vehicle except for the cameras. These sensors include the IMU, GPS, and 14 ultrasonic sensors for obstacle detection. The flight takes inputs through communication to the base platform, from the obstacle detection algorithms, and also from the vision processing board. The output from the flight controller goes to the motor controller and is then converted into appropriate signals to control the propulsion system.

Odroid - microprocessor for running visual algorithms connects to the camera and optical flow sensors on board the UAV. Odroid runs vision algorithms and outputs the result to the flight controller.

5.3 Software

The software of our system can be broken into two categories, software that is computed on the platform and software computed on board the UAV. The software on the platform performs two functions. The first is that it is a user interface for human input and control. The second function is path planning algorithms used to optimize the takeoff and return locations for the UAV. This information is then conveyed to the UAV by their communication channel.

The code that is occurring onboard the UAV can be broken down into three major functions. The first is flight control software - Advanced VTOL Autonomy (AVA) code written by FireFly manufacturers. The second major software function is obstacle avoidance. The UAV must avoid obstacles in flight and must sense its changing environment while in flight. The last set of algorithms revolves around computer vision and visual processing. This code converts visual inputs into meaningful outputs for the flight control.

6. Current System Status

6.1 Obstacle Avoidance Subsystem

6.1.1 Sensors Overview

This semester, the following parts of obstacle avoidance subsystem were achieved:

- Selecting appropriate sensors
- Deciding sensor layout
- Gather data from multiple ultrasonic sensor simultaneously by serially pinging the sensors

We are mounting fourteen Maxbotix LV EZ MB1010 ultrasonic sensors around the UAV to cover area of 1.5 m around the UAV. The flowchart of the way obstacle avoidance subsystem will work is shown below in Figure 4.

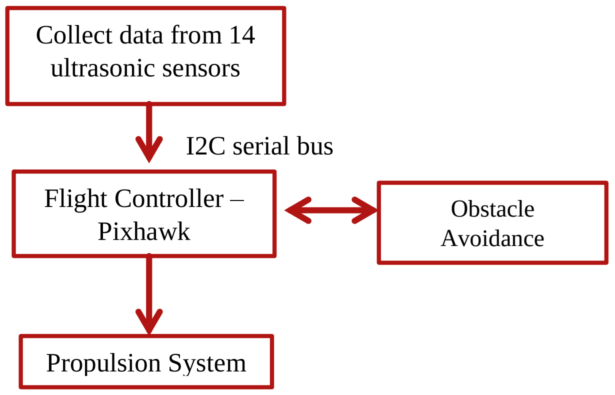

Collect data from 14 ultrasonic sensors
I2C serial bus
Flight Controller – Pixhawk
Obstacle Avoidance
Propulsion System

Figure 4: Flowchart for the Obstacle Avoidance Subsystem

Data from 14 ultrasonic sensors will be passed to Pixhawk through the I2C serial circuit described in the PCB Subsystem. Flight controller will run obstacle avoidance algorithm and correspondingly generate motor commands for the propulsion system.

The CAD design of the sensor layout of obstacle avoidance system are as follows. 7 ultrasonic sensors are facing horizontally while 7 ultrasonic sensors are mounted at 42 degree angle. With this arrangement any obstacle that comes at an angle to the UAV will also be detected.

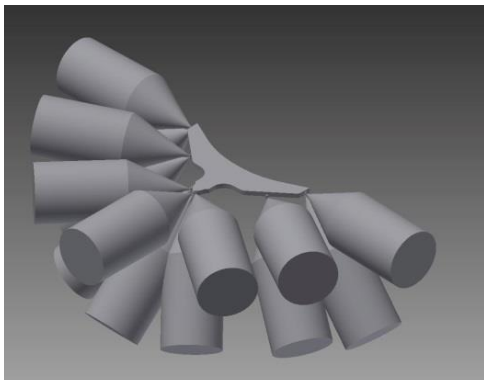


Figure 5: Arrangement of 14 Ultrasonic Sensors around The UAV

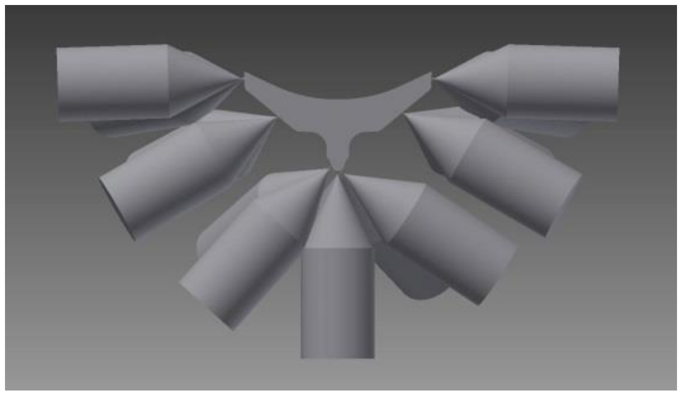


Figure 6: Top View of Sensor Layout showing arrangement of 14 ultrasonic sensors

For the fall validation experiment we demo simultaneous functioning of 6 ultrasonic sensors mounted at the nose of the UAV [Figures 7 & 8]. Figure below shows the arrangement of the sensors and the sensor visualization in Rviz [Figure 9].

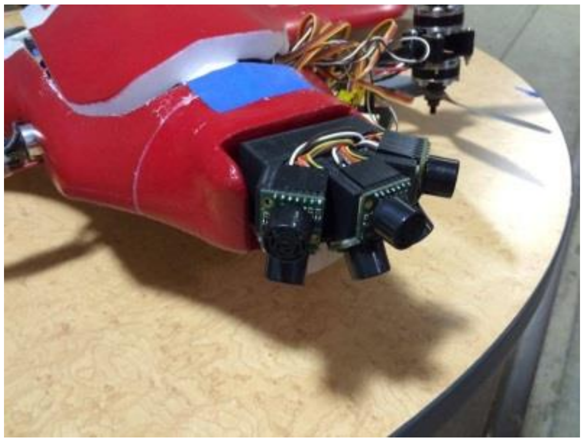


Figure 7: Close Up of 6 Ultrasonic Sensors Mounted at the Nose of UAV

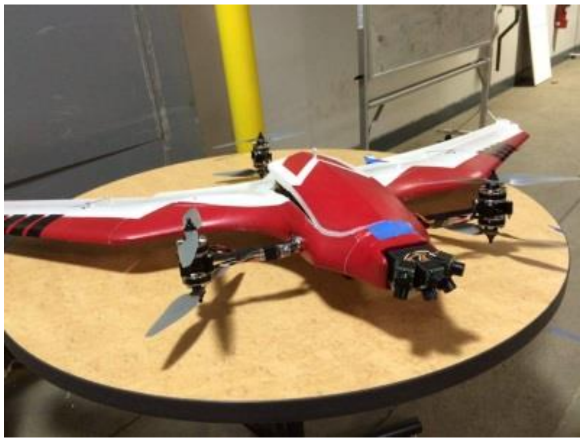
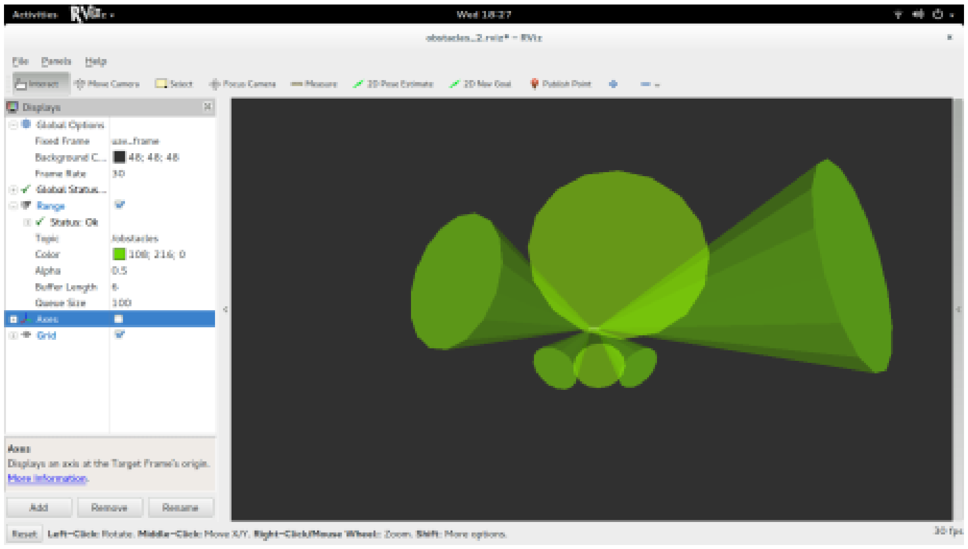


Figure 8: Image Showing 6 Ultrasonic Sensors Mounted at the Nose of the UAV



Global Options	
Fixed Frame	axe_frame
Background C...	■ 48; 48; 48
Frame Rate	30
✓ Global Status...	
Range	✓
✓ Status: Ok	
Topic	/obstacles
Color	🟩 108; 216; 0
Alpha	0.5
Buffer Length	6
Queue Size	100
Axes	☐
Grid	✓

Axes:
Displays an axis at the Target Frame's origin.
More Information.

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.

Figure 1 Rviz Visualization of 6 ultrasonic sensors

The flowchart below explains the experiment arrangement of sensors [Figure 10]. 6 ultrasonic sensors were connected to the arduino. Sensor readings from arduino were passed to a laptop on which sensor visualization was shown in Rviz.

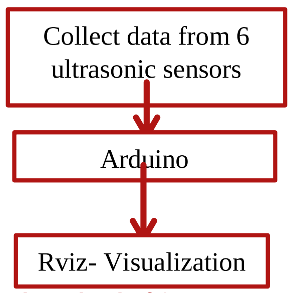


Figure 10: Flowchart of Sensor Experiment Arrangement

6.1.2 Sensor Modeling, Test, and Analysis

According to our system requirements we wish to detect any obstacle that comes within an area of 1.5m radius around the UAV. For this our first step was to decide the type of sensors to be used. We did tests with IR, ultrasonic and lidar to decide which sensor was best for our system. Following are the analysis of the tests we did.

Sensor	Analysis
IR	For our system requirement we need 39 IR sensors to cover the whole area. Too many sensors.
Lidar	Can detect only in one plane. 3D Lidar are too costly Mounting lidar on servo and rotating introduces more complexity
Ultrasonic	Need 14 ultrasonic sensors to meet the system requirement. Cost within budget

Table 1: Analysis of Sensor Tests

Based on the above analysis we decided to use ultrasonic sensors. In the above analysis we mentioned we required 14 ultrasonic sensors. This number of sensors was obtained by geometry. We divided the area to be covered around the UAV with the detection area of each sensor (obtained from sensor datasheet) and obtained the number of sensors required.

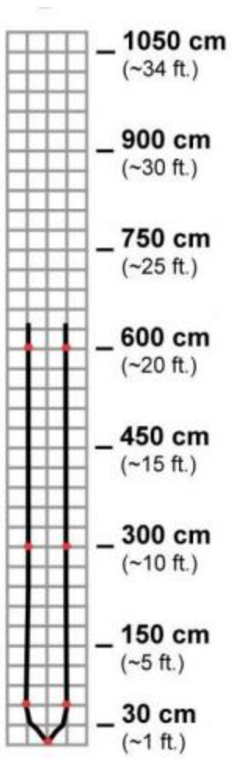
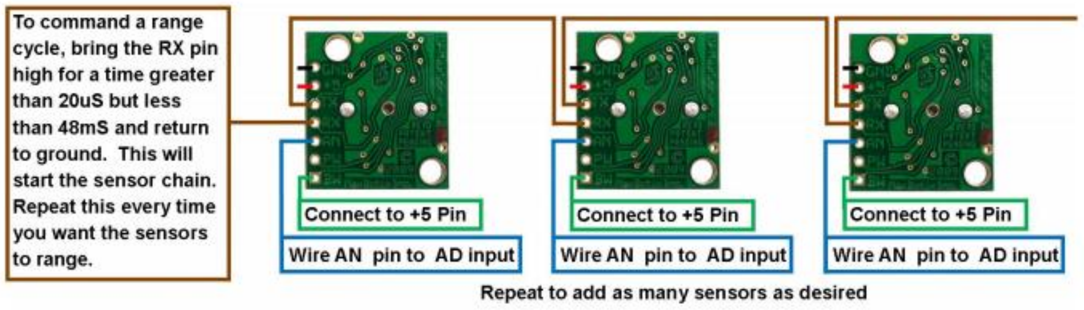


Figure 11: Detection Area of Maxbotix MB1010 Ultrasonic Sensor. Source: Sensor Datasheet

Ultrasonic sensors face issue with interference. Hence, we serially ping each sensor using the pinging hardware arrangement provided by Maxbotix as shown in the figure below.



To command a range cycle, bring the RX pin high for a time greater than 20uS but less than 48mS and return to ground. This will start the sensor chain. Repeat this every time you want the sensors to range.
Connect to +5 Pin
Wire AN pin to AD input
Connect to +5 Pin
Wire AN pin to AD input
Connect to +5 Pin
Wire AN pin to AD input
Repeat to add as many sensors as desired

Figure 12: Maxbotix serially pinging arrangement. TX pin of one sensor is connected to RX pin of another. Once one sensor collects reading it pings next sensor to take reading.

Next we performed test to find out the time taken by each sensor to take 1 reading. As shown below in figure 13, each sensor requires 42ms to take one reading.

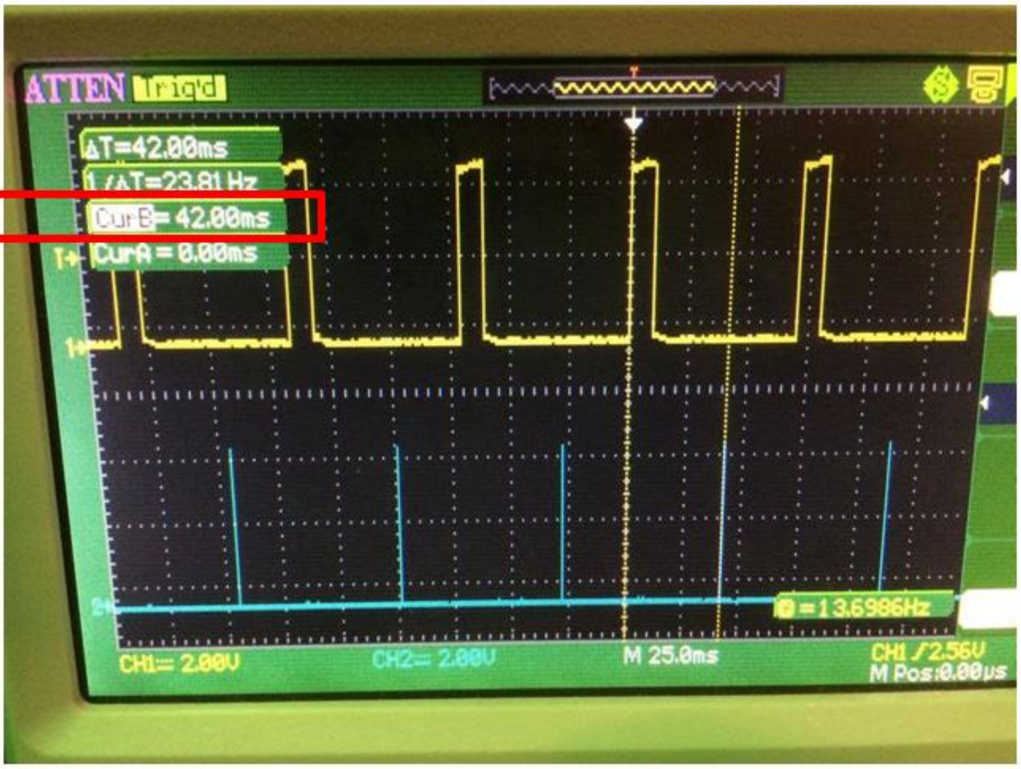


Figure 13: Cycle Length of Sensor Ping

As the time it takes for 1 sensor to collect reading is 42 ms other sensor should not collect readings during the same time or interference issue will cause incorrect reading. The total time taken by 14 sensors if they are all pinged one after the other: 42 X 14 = 588ms. Now we use a median filter of size 5 on top of it. Hence total time taken to take one reading: 588 X 3 = 1.76s 1.76s update rate isn't acceptable for the system. Hence we subdivided the 14 ultrasonic sensors into geometrically 3 subsystem facing in different direction so that they do not interfere with each other as shown in figure below.

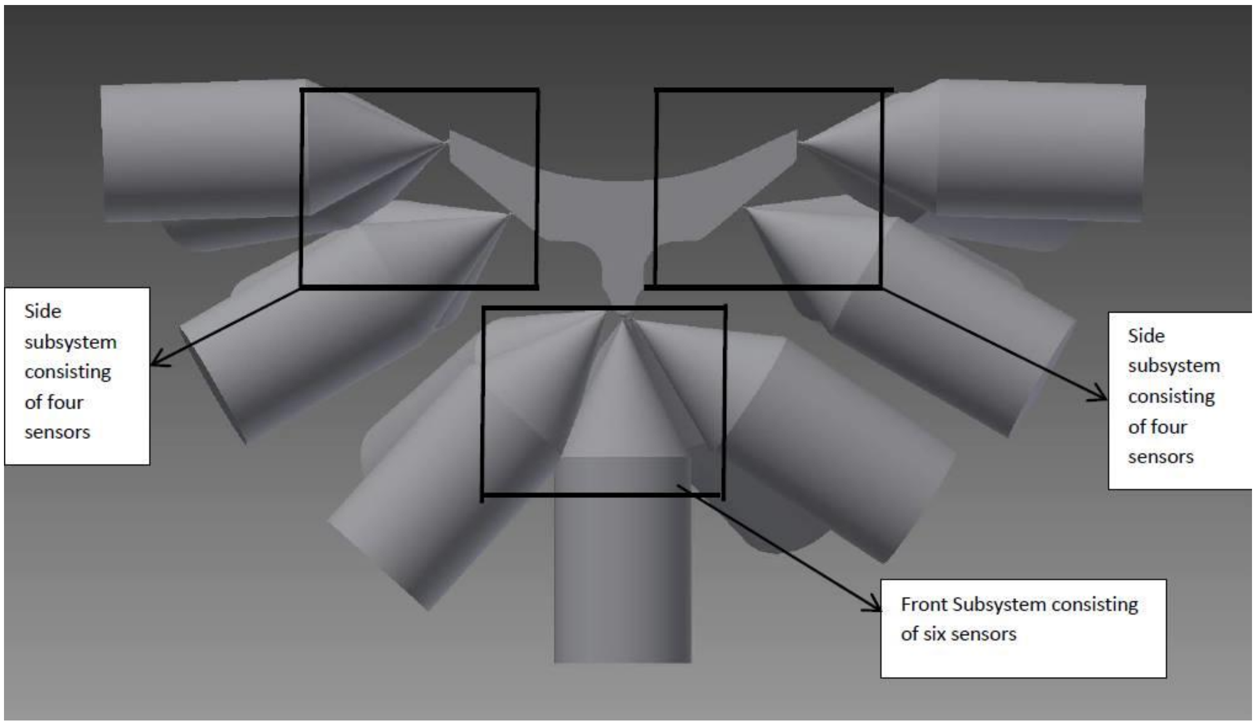

Side subsystem consisting of four sensors
Side subsystem consisting of four sensors
Front Subsystem consisting of six sensors

Figure 14: Dividing the sensors into 3 subsystems to reduce update rate of the system

We further divided each subsystem into sensors on the top layer in one group and other system consisting of sensors in bottom layer.

Analysis of the serial pinging arrangement are:

- Serially pinging sensors helps to get rid of interference
- Pinging sensors facing in different direction simultaneously helps reduce update rate
- Using pinging pattern a subsystem of 6 sensors achieves around 250 ms update rate

6.2 Master-Slave Sensor Boards

6.2.1 PCB Overview

As mentioned in the obstacle avoidance subsystem we are using I2C communication for connecting 14 ultrasonic sensors to the flight controller. Hence we designed master board to handle combining of I2C lines and 5V regulator. Slave boards were designed to take sensor inputs and reduce analog line noise by converting it straight to digital.

The flowchart below explains the connection of sensors with the master-slave board.

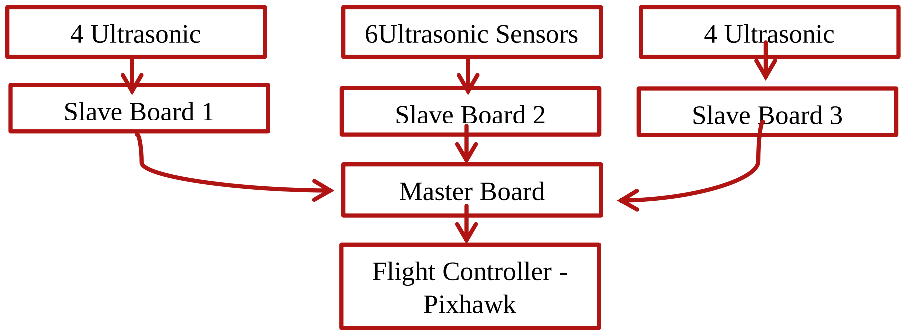


Figure 15: Flowchart of Sensor Interface using Master-Slave Boards

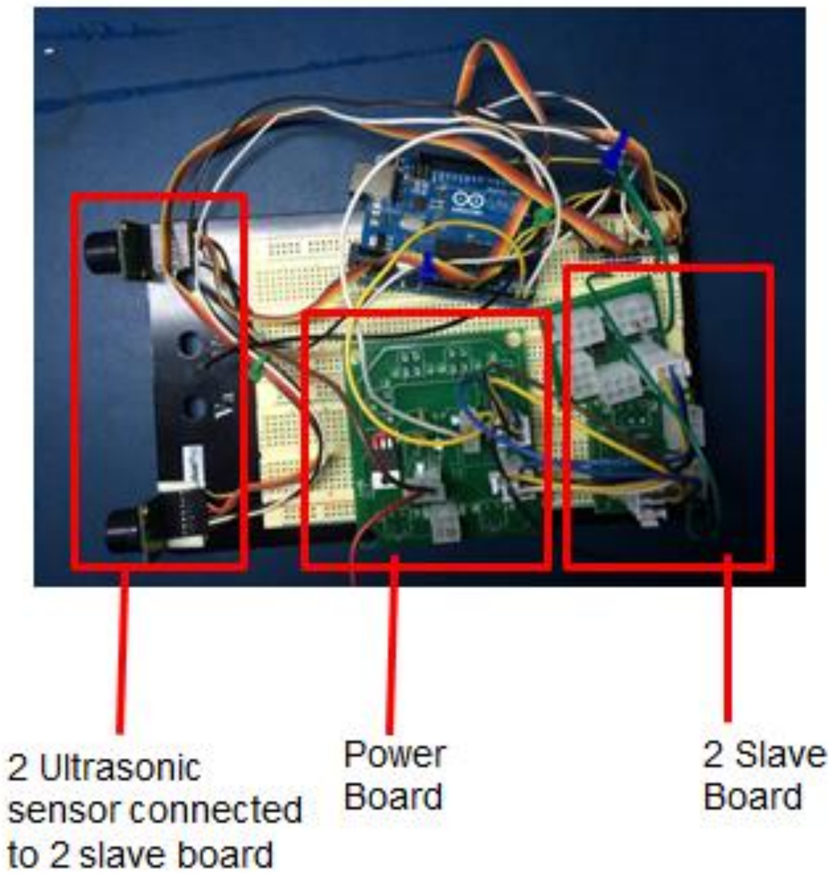


Figure 26: Fall Validation Senosr Setup.

In the FVE, 2 sensor connected to 2 slave board. Slave board pass data to master board which relays data to Arduino. Data received by Arduino is visualized on laptop through serial terminal

6.2.2 PCB Modeling, Test and Analysis

Design of master-slave board is made in Eagle. The schematic of the board are as bellow.

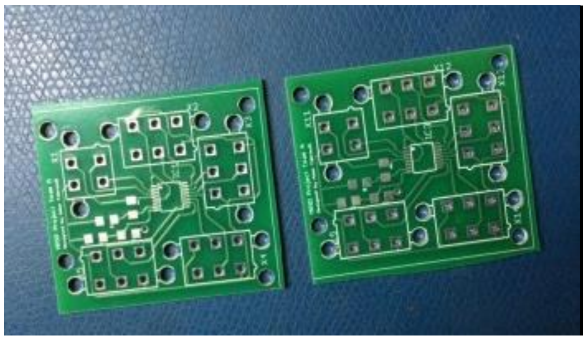


Figure 37: Two Printed Slave Boards

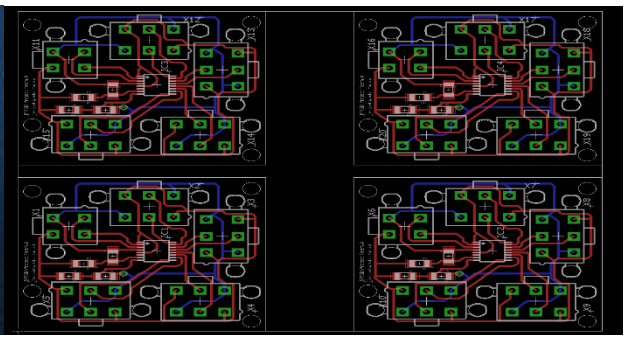


Figure 4 Schematic of four slave boards

Printed circuit boards are populated. Power and data lines are tested. No significant errors detected.

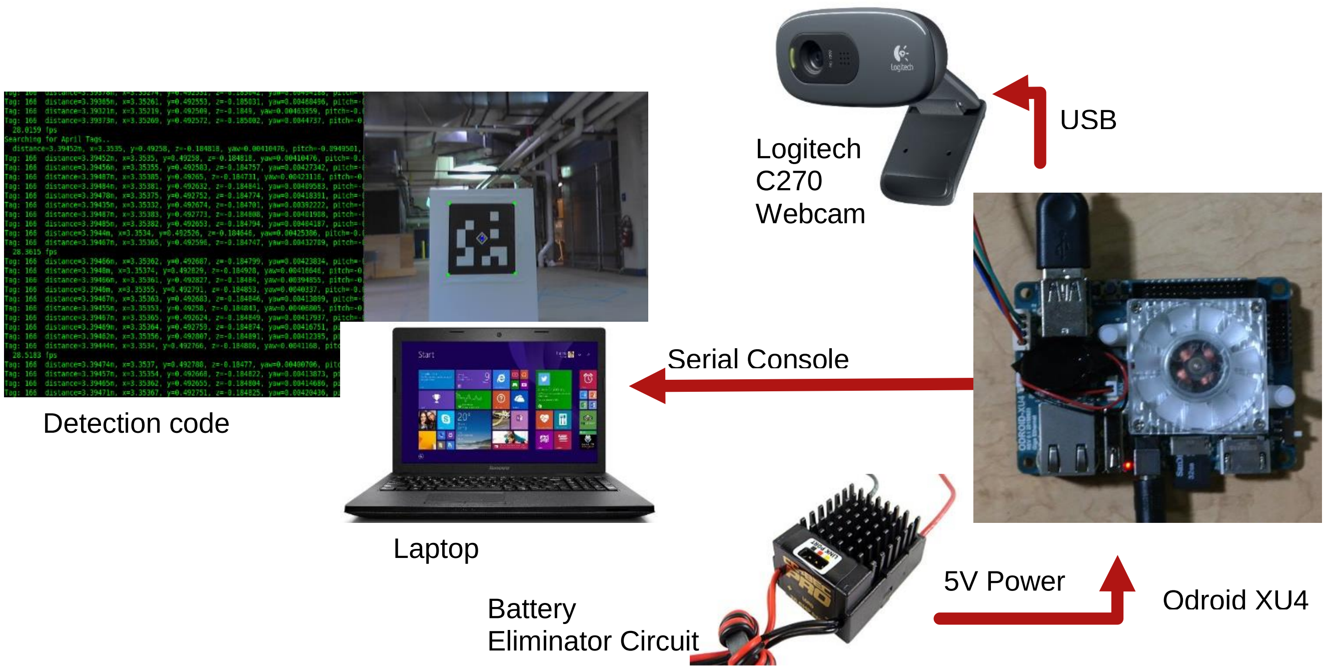
6.3 Vision Subsystem

6.3.1 Single Board Computer

The Single Board Computer is responsible for high level control of the UAV. This includes instructions for navigation and determining the drop point for accurate landing and delivery. After trying multiple SBCs, we have finalized on the Odroid XU4. It has two quad core processors each core with more processing power compared to the Beagleboard xM and the BeagleBone Black, and is light enough to be mounted on the vehicle.

As seen in the Cyber-Physical Architecture, the Odroid shall directly communicate with the Pixhawk (Flight controller) and control the UAV. When the UAV is near the house, the camera connected to the Odroid with identify the visual marker near the house and instruct the pixhawk on landing accurately on it.

For the Fall Validation, the architecture replaced the Pixhawk with a laptop to see the output data from the Odroid (refer figure XX).



```
Tag: 166  distance=3.39365m, x=3.35261, y=0.492553, z=-0.185031, yaw=0.00460496, pitch=-
Tag: 166  distance=3.39321m, x=3.35219, y=0.492509, z=-0.1849, yaw=0.00463959, pitch=-0.
Tag: 166  distance=3.39373m, x=3.35269, y=0.492572, z=-0.185002, yaw=0.0044737, pitch=-0
  28.0159 fps
Searching for April Tags..
  distance=3.39452m, x=3.3535, y=0.49258, z=-0.184818, yaw=0.00410476, pitch=-0.0949501,
Tag: 166  distance=3.39452m, x=3.3535, y=0.49258, z=-0.184818, yaw=0.00410476, pitch=-0.
Tag: 166  distance=3.39456m, x=3.35355, y=0.492583, z=-0.184757, yaw=0.00427342, pitch=-
Tag: 166  distance=3.39487m, x=3.35385, y=0.49265, z=-0.184731, yaw=0.00423116, pitch=-0
Tag: 166  distance=3.39484m, x=3.35381, y=0.492632, z=-0.184841, yaw=0.00409583, pitch=-
Tag: 166  distance=3.39478m, x=3.35375, y=0.492752, z=-0.184774, yaw=0.00418391, pitch=-
Tag: 166  distance=3.39435m, x=3.35332, y=0.492674, z=-0.184701, yaw=0.00392222, pitch=-
Tag: 166  distance=3.39487m, x=3.35383, y=0.492773, z=-0.184808, yaw=0.00401988, pitch=-
Tag: 166  distance=3.39485m, x=3.35382, y=0.492653, z=-0.184794, yaw=0.00404187, pitch=-
Tag: 166  distance=3.3944m, x=3.3534, y=0.492526, z=-0.184646, yaw=0.00425386, pitch=-0.
Tag: 166  distance=3.39467m, x=3.35365, y=0.492596, z=-0.184747, yaw=0.00432789, pitch=-
  28.3615 fps
Tag: 166  distance=3.39466m, x=3.35362, y=0.492687, z=-0.184799, yaw=0.00423834, pitch=-
Tag: 166  distance=3.3948m, x=3.35374, y=0.492829, z=-0.184928, yaw=0.00416646, pitch=-0
Tag: 166  distance=3.39466m, x=3.35361, y=0.492827, z=-0.18484, yaw=0.00394855, pitch=-0
Tag: 166  distance=3.3946m, x=3.35355, y=0.492791, z=-0.184853, yaw=0.0040337, pitch=-0.
Tag: 166  distance=3.39467m, x=3.35363, y=0.492683, z=-0.184846, yaw=0.00413899, pitch=-
Tag: 166  distance=3.39455m, x=3.35353, y=0.49258, z=-0.184843, yaw=0.00406805, pitch=-0
Tag: 166  distance=3.39467m, x=3.35365, y=0.492624, z=-0.184849, yaw=0.00417937, pitch=-
Tag: 166  distance=3.39469m, x=3.35364, y=0.492759, z=-0.184874, yaw=0.00416751, pi
Tag: 166  distance=3.39462m, x=3.35356, y=0.492807, z=-0.184891, yaw=0.00412395, pi
Tag: 166  distance=3.39444m, x=3.3534, y=0.492766, z=-0.184886, yaw=0.0041168, pitc
  28.5183 fps
Tag: 166  distance=3.39474m, x=3.3537, y=0.492788, z=-0.18477, yaw=0.00400706, pitc
Tag: 166  distance=3.39457m, x=3.35354, y=0.492668, z=-0.184822, yaw=0.00413873, pi
Tag: 166  distance=3.39465m, x=3.35362, y=0.492655, z=-0.184804, yaw=0.00414686, pi
Tag: 166  distance=3.39471m, x=3.35367, y=0.492751, z=-0.184825, yaw=0.00420436, pi
```

Figure 19: Testing and Validation Platform for the Vision Subsystem

6.3.2 Marker

We considered various markers that would be simple and robust for our application. Finally, we have developed a custom marker which uses one small AprilTag nested into another larger one.

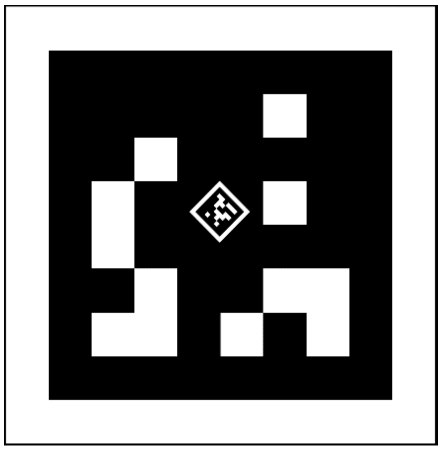


Figure 20: Nested AprilTag marker. Outer AprilTag can be seen from far-off distances, and the inner one from nearby distances. (Tag id 166 outer, 138 inner, rotated -45 degrees)

As seen in the figure above, the inner marker is one-tenth the size of the outer marker and is rotated 45 degrees counter-clockwise so it does not hinder the detection of the larger one. The nested AprilTag helps in increasing the range of a simple AprilTag.

Different sizes of nested tags were tested to determine upto what ranges they can be detected. Table 2 depicts the final comparisons

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

Table 2: Table depicting tested size-range relationships for nested AprilTags

6.3.3 Detection Algorithm

The most important part of the vision system is the marker detection algorithms which read and detect the markers. 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.

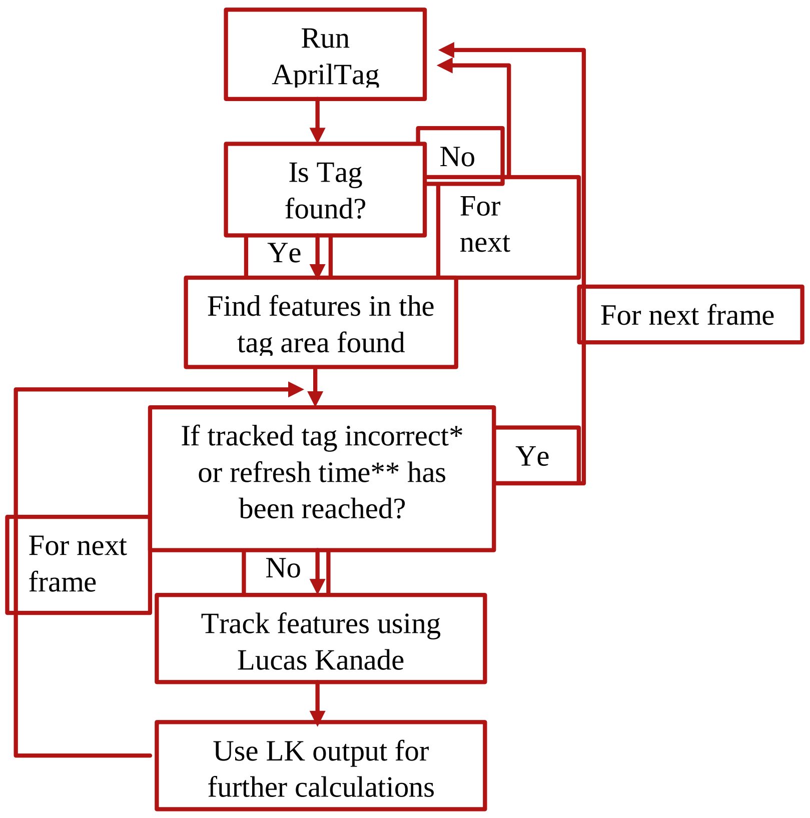

Run AprilTag
Is Tag found?
No
For next
Ye
Find features in the tag area found
For next frame
If tracked tag incorrect* or refresh time** has been reached?
Ye
For next frame
No
Track features using Lucas Kanade
Use LK output for further calculations

Figure 51: Flowchart depicting the algorithm for detection and tracking for apriltag

After trying multiple algorithms and markers, we finally settled with an AprilTag detection algorithm as developed by University of Michigan [1] and its C++ version as developed in MIT [2]. The Apriltag detection gives upto 8 fps on the Odroid. This rate is too slow for controlling the UAV. As a workaround, we tried to combine the AprilTag detection with Lucas-Kanade tracking algorithm. The basic idea of the same is illustrated in the flowchart [Figure 21].

We use the AprilTag 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. 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 AprilTag detection for the next frame. As tracking results may start deviating from the actual detections, it is good idea to refresh the estimates using the AprilTag 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)

1. Basic: verify that the tracked points form a sensible quadrilateral.

2. 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.

Different algorithms were tested for speed on the laptop and the Odroid. The speeds have been compared in Table 3.

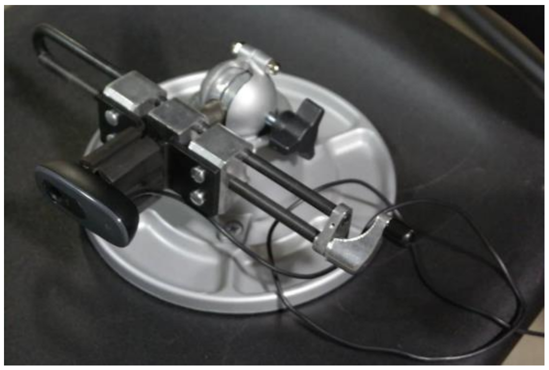
Algorithm	FPS on Laptop (i3 4th gen)	FPS on Odroid (Quad core ARM)
AprilTag detection	14	8
Lucas Kanade Tracking	30	29
Merged (LK + AprilTag detection)	29	28

Table 3: Speed comparisons for different algorithms

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]
- Marker Detection Test #3 [https://youtu.be/5TkUyuMBI2E]

6.3.4 Vision Subsystem - Testing and Analysis


Camera mount for calibration

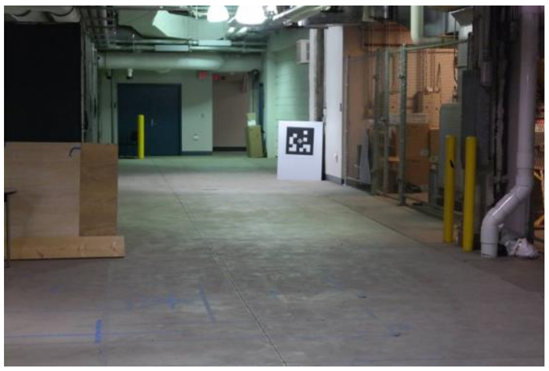

Marker setup

Figure 22: The testing setup used for comparing detection results.

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 works best in outdoor conditions whereas high exposure work better in indoor conditions.

Marker is detected and tracked with high precision in all orientations (changes in pitch roll and yaw).

The primary constraint for finding error in detected 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. 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).

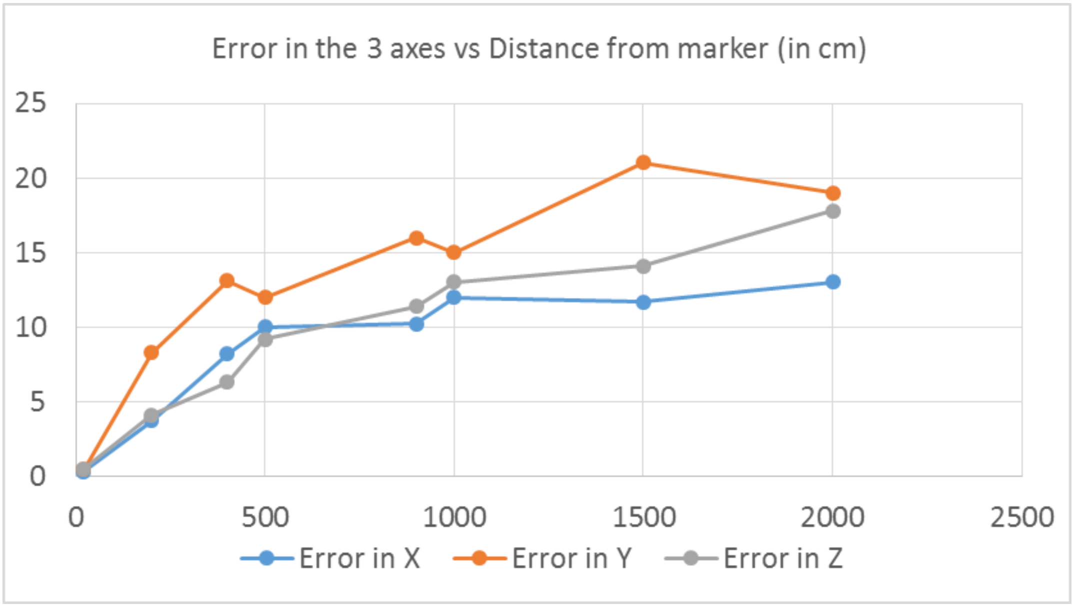


Figure 23: Graph depicting error in detection distances to Marker

6.4 UAV and Flight Control System

The primary focus for the UAV and flight control system this semester was to get it to fly via RC control and waypoint navigation. The system had to be procured, assembled, tested, programmed, and tuned to achieve this goal.

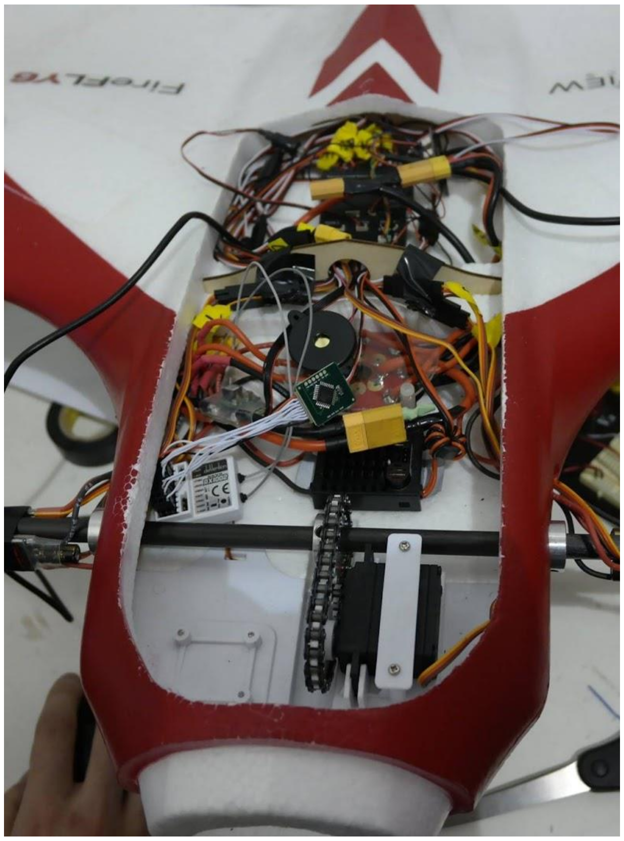


Figure 24: Electronics Installed in FireFLY6

Our current status is that the vehicle is procured, assembled, and semi-functional. The electronics can be seen inside the vehicle in Figure 24. Due to the fact that we are using beta code from BirdsEyeView Aerobotics, they have many significant changes from the traditional Pixhawk code found online. BirdsEyeView sent us their own version of Mission Planner and their own compiled version of their firmware which has a controller to combine Arduplane and Arducopter Y6 configuration code. Their code has stronger PreArm safety checks than the traditional Ardupilot code and it requires an external compass to be present to arm the motors.

Our major bottleneck in getting this vehicle up and running was the PreArm software checks. Due to faulty hardware, the Pixhawk was unable to find or calibrate the external compass. Even when we successfully calibrated, the readings would be lost when we power cycled the vehicle. We were able to get the vehicle flying on the night before the FVE but it failed to arm after that.

We worked with BirdsEyeView Aerobotics' head firmware engineer and performed many tests to attempt to fix the issue. We tried removing connectors and soldering wires directly into ports, placing aluminum foil under the compass as a homemade faraday cage, attempting to calibrate indoors and out in open fields, and many other tests. Ultimately we were unable to bypass this bug no matter what solution we attempted to throw at it and consequently we were unable to meet our FVE requirements.

Currently we realize 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 to meet Spring validation tests. Due to this realization, we are contemplating as part of our risk mitigation to change platforms entirely and go with a octocopter capable of doing everything the FireFly6 does just at slower speeds and with less flight time. Cutting our losses and modifying our project will be the best thing for our project long term and so we believe it is the right move to take at this time.

7. 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.

7.1 Work Breakdown Structure

1 Systems Design

1.1 Project Planning

1.1.1	Design System Architecture	25 days
1.1.2	Design Test Environment	4 days

1.2 Drone

1.2.1	Choose Drone	1 day
1.2.2	Select Flight Controller	1 day
1.2.3	Design Drone Underbelly	2 wks
1.2.4	Design Marker Search Algorithm	4 weeks

1.3 Ground Platform

1.3.1	Design Base Station	1 week

1.4 Vision System

1.4.1	Design Vision System	1 day
1.4.2	Select Camera	3 days
1.4.3	Select Vision Board	1 day
1.4.4	Design Visual Markers	1 day

1.5 Obstacle Avoidance

1.5.1	Analyze Obstacle Sensors	1 wk
1.5.2	Design Obstacle Avoidance	2
1.5.3	Design Sensor Layout	1 wk

1.6 Communications System

1.6.1	Design Communications	3 days
1.6.2	Select Radio Module	1 day

1.7 User Interface

1.7.1	Determine User I/O	4 days
1.7.2	Design User Interface	2 weeks

2 Procurement and Assembly

2.1 Drone

2.1.1	Procure Drone	2 wks
2.1.2	Assemble Drone	1 wk
2.1.3	Procure Flight Controller	2 wks
2.1.4	Mount obstacle sensors	1 wk
2.1.5	Fabricate underbelly	1 wk

2.3 Ground Platform

2.3.1	Build Ground Platform	3 wks

2.4 Vision System

2.4.1	Procure Camera	2 wks
2.4.2	Procure Vision Board	1 week
2.4.3	Fabricate Visual Markers	1 wk

2.5 Obstacle Avoidance

2.5.2	Procure Obstacle Sensors	1 week
2.5.3	Procure Optical Flow	1 week
2.5.4	PCB iterations	2 wks

2.6 Communications System

2.6.1	Procure Radio Module	2 wks

2.7 User Interface

2.7.1	Build User Interface	1 week

2.8 Package Handling

2.8.1	Build/Procure Gripper	2 mo

3 Testing & Integration

3.1 Project Planning

3.1.1	Build Test Environment	1 week
3.1.2	Full Scenario Test	2 weeks

3.2 Drone

3.2.1	Test Flight Controller	1 week
3.2.2	Test Drone R/C-only Control	4 days
3.2.3	Tune and test forward flight	2 wks
3.2.4	Understand code	1.5 wks
3.2.5	Waypoint using hover	4 days
3.2.6	Waypoint using FF	1.5 wks
3.2.7	Autonomously control UAV	1 wk
3.2.8	Test Visual Landing of Drone	1 wk

3.3 Vision System

3.3.1	Test Camera and Board	3 days
3.3.2	Integrate and test Visual system on board	4 days
3.3.3	Test Visual Markers with Vision System	3 days
3.3.4	Integrate vision info into control	4 days

3.4 Obstacle Avoidance

3.4.1	Integrate Optical Flow	2 wks
3.4.2	Integrate obstacle avoidance system	1.5 wks
3.4.3	Table Test Obstacle Avoidance Sensors	3 wks
3.4.4	Test Waypoint Following with Obstacle Avoidance	1 mo
3.4.5	Test Visual Landing with Obstacle Avoidance	2 mo

3.5 Communications System

3.5.1	Electronic Test of Radio Module	1 day
3.5.3	Data Test of Radio Module	2 days

1.8	**Package Handling**	
1.8.1	Design Gripper System	3 weeks
1.8.2	Select Gripper Mechanism	1 wk
1.8.3	Design Package Modifications	1 wk

3.6	**User Interface**	
3.6.1	Test User Interface	3 days

3.7	**Package Handling**	
3.7.1	Test Gripper Electronics	4 days
3.7.2	Test Gripper and Package Modifications	1 week
3.7.3	Integrate and test gripper with drone	1 week

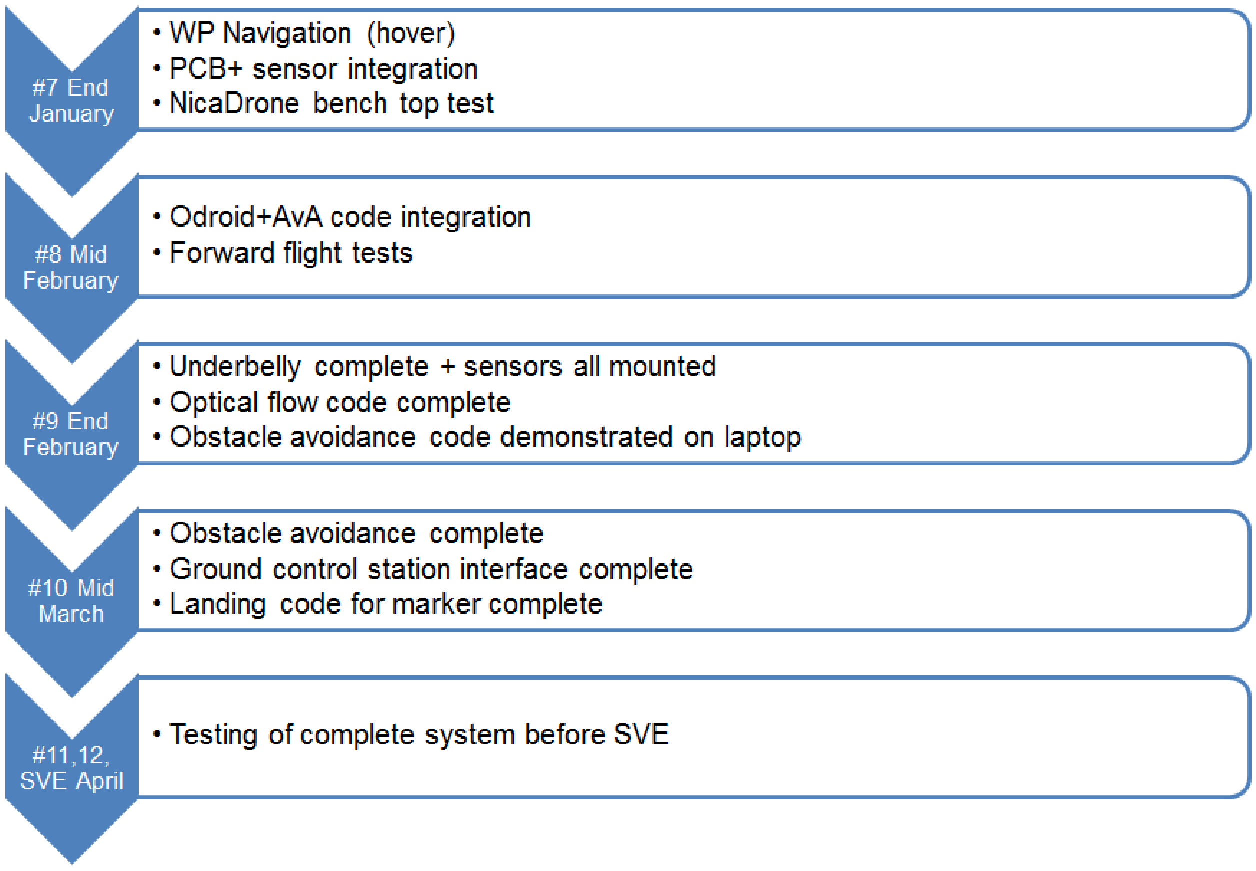
Figure 25: Work Breakdown Structure

As you can see, the vision system is nearly complete. 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.

7.2 Schedule

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

Current priorities include fixing current UAV problems, initiating obstacle avoidance code, and integrating subsystems.



#7 End January	• WP Navigation (hover) • PCB+ sensor integration • NicaDrone bench top test
#8 Mid February	• Odroid+AvA code integration • Forward flight tests
#9 End February	• Underbelly complete + sensors all mounted • Optical flow code complete • Obstacle avoidance code demonstrated on laptop
#10 Mid March	• Obstacle avoidance complete • Ground control station interface complete • Landing code for marker complete
#11,12, SVE April	• Testing of complete system before SVE

Table 5: Project Schedule of Deliverables

8. Test Plan

8.1 Capability Milestone for spring-semester Progress Review

The table listed below describes the high-level capability milestone for each of the progress review.

Milestone	Capability	Description
Progress Review 7	Waypoint Navigation	Successfully demonstrate the UAV following waypoints in hover mode
	Obstacle Detection	Connect all sensors to flight controller using master-slave board. Display the readings of the sensors
	Gripper	Demonstrate hold and release action of gripper mechanism
Progress Review 8	Vision and Flight controller communication	Demonstrate communication between Odroid and AvA code (flight controller code)
	Forward Flight	Demonstrate forward flight motion of UAV
	Underbelly Design	Mount gripper, camera and sensors on UAV
Progress Review 9	Optical Flow	Demonstrate optical flow integrated to the flight code
	Obstacle Avoidance	Show in simulation UAV avoiding static obstacles
	Obstacle Avoidance	Demonstrate UAV flight integrated with functioning obstacle avoidance system
Progress Review 10	Ground control	Demonstrate continuous update of UAV status on the ground control station
	Landing	Demonstrate autonomous landing of UAV on marker
Progress Review 11	Integration	Buffer period

Progress Review 12	Integration	Buffer period

Table 6: Deliverables by Progress Review

8.2 Spring Validation Experiment

We have divided our Spring Validation Experiment in 3 tests. Package carrying mechanism test demonstrates that the UAV can fly with the package and deliver it. Obstacle-less package delivery test demonstrates the complete functionality of package delivery with the obstacle avoidance system functioning. The third test will demonstrate all the systems – package carrying mechanism, obstacle avoidance and navigation functioning successfully.

Test Name	Package carrying mechanism test	
Test Description	Validates the package carrying and dropping capabilities of the UAV	
Test Location	Schenley Park	
Equipment Required	UAV fitted with the package carrying mechanism, Small package.	
Step	Step Description	Success condition
1	Place the UAV with the package attached to the carrying mechanism	
2	Initiate take off manually	
3	UAV lifts off and hovers 5m over the ground for 1 minute	The package remains securely attached to the UAV
4	UAV descends and lands	Package still attached
5	UAV drops package onto the ground	Package is released and lands on the ground

Table 7: Spring Validation, Package carrying Test

Test Name	Obstacle-less package delivery test	
Test Description	Validates that packages can be delivered to a house without any obstacles in the path	
Test Location	Schenley Park	
Equipment Required	Platform, Open space (outdoor environment), Fully equipped system.	
Step	Step Description	Success condition
1	Place UAV with package at platform with visual marker	
2	Initiate system by entering GPS coordinates for the house and GPS coordinates of the return to point	Delivery begins
3	UAV takes off autonomously towards the house	
4	Reaches waypoint near the house using GPS	Autonomously and accurately navigates using GPS to reach near given GPS coordinates
5	Identifies and navigates to the visually marked drop off point (with no obstacles in the path)	Identifies the visual marker near the GPS destination and autonomously navigates to it
6	Lands and drops the package	The package should be upto 2m from the center of the visual marker
7	Autonomously takes off and navigates to the GPS of the return back point (communicated in the beginning)	Package should remain delivered. UAV departs
8	Detects, navigates and lands at the platform with the visual marker	Should land within 2m of the center of the visual marker

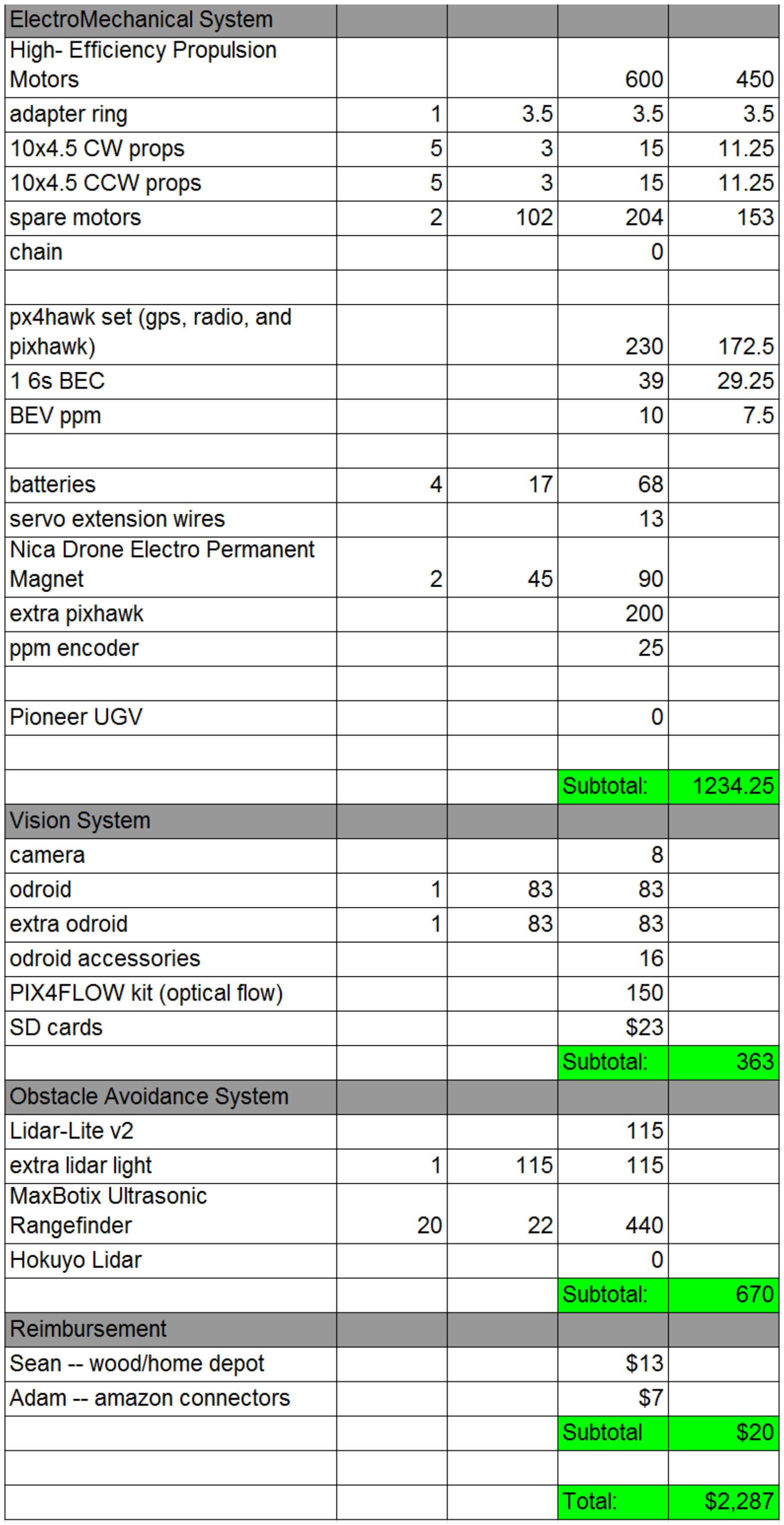
Table 8: Spring Validation, Obstacle less package delivery test

Test Name	Complete Package delivery test with obstacles	
Test Description	Validates that packages can be delivered to a house even with static obstacles in the path	
Test Location	Schenley Park	
Equipment Required	Platform, Open space (outdoor environment), Fully equipped system, Obstacles (cross section: 1.5m x 0.5m).	
Step	Step Description	Success condition
1	Place UAV with package at platform with visual marker	
2	Initiate system by entering GPS coordinates for the house and GPS coordinates of the return to point	
3	UAV takes off autonomously towards the house	
4	Reaches waypoint near the house using GPS	
5	Identifies the marker and plans path to navigate to it	Identifies the visual marker near the GPS destination.
6	Place an obstacle (2mx2m) in the path of the UAV	Avoids the obstacle
7	Place an obstacle (2mx2m) on the side of the UAVs intended path	Avoids the obstacle
8	Repeat with 1.5m x 0.5m obstacles	Avoids the obstacle
9	Repeat with both obstacles, placed in the front and the sides	Avoids the obstacles
10	Lands and drops the package	
11	Autonomously takes off and navigates to the GPS of the return back point (communicated in the	

	beginning)	
12	Detects, navigates and lands at the platform with the visual marker	

Table 9: Spring Validation, Complete Package delivery test with obstacles

8.3 Budget



ElectroMechanical System				
High- Efficiency Propulsion Motors			600	450
adapter ring	1	3.5	3.5	3.5
10x4.5 CW props	5	3	15	11.25
10x4.5 CCW props	5	3	15	11.25
spare motors	2	102	204	153
chain			0	
px4hawk set (gps, radio, and pixhawk)			230	172.5
1 6s BEC			39	29.25
BEV ppm			10	7.5
batteries	4	17	68	
servo extension wires			13	
Nica Drone Electro Permanent Magnet	2	45	90	
extra pixhawk			200	
ppm encoder			25	
Pioneer UGV			0	
			Subtotal:	1234.25
Vision System				
camera			8	
odroid	1	83	83	
extra odroid	1	83	83	
odroid accessories			16	
PIX4FLOW kit (optical flow)			150	
SD cards			$23	
			Subtotal:	363
Obstacle Avoidance System				
Lidar-Lite v2			115	
extra lidar light	1	115	115	
MaxBotix Ultrasonic Rangefinder	20	22	440	
Hokuyo Lidar			0	
			Subtotal:	670
Reimbursement				
Sean -- wood/home depot			$13	
Adam -- amazon connectors			$7	
			Subtotal	$20
			Total:	$2,287

Table 10: Project Budget

This semester we spent $2,061 of our projected budget of $2,287. The majority of our budget has been spent on the FireFly6 platform and its spares. The difference between what we spent and our projected budget is because we haven't bought the spare Odroid, spare camera, and spare LidarLite yet. Even when we purchase those spares, we will have used 57% of our $4,000 budget which gives us plenty of money to purchase components in the Spring if necessary.

9. Conclusion

As already stated, we realize 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 will be the best thing for our project long term and so we believe it is the right move to take at this time.

On the Obstacle Avoidance end, 14 ultrasonic sensors are sufficient to cover area of 1.5 m radius around the UAV. Using serial pining we can get rid of interference. The update rate for the system is around 250 ms which is good. The sensors are not very precise and give around +-20cm error when obstacle are not exactly perpendicular to the sensor. However the error of the system is with the limits of our system requirements.

The vision system has been developed to run using a Logitech webcam on an Odroid. We use nested AprilTag markers and AprilTag detection coupled with Lucas Kanade tracking algorithm to track the marker location. It is able to achieve upto 29 frames per second update rate and can detect the marker upto 20m. To summarize, our vision system looks strong and ready to be integrated. The algorithm used is fast, robust and accurate and based on initial estimates should be able to guide the UAV to land.

Significant work was completed this semester. Despite this, Team Avengers was unable to get the UAV flying in time. Due to properly applied project management methods, the team has the time and resources to aggressively adapt subsystems in order to meet SVE. As an ambitious group, we intend to meet those goals.

10. References

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

Figure 4: Dronecode Software Architecture https://www.dronecode.org/dronecode-software-platform, accessed 10/2/15

Web link: “Electro Permanent Magnet” from Nica Drone, http://nicadrone.com/index.php?id_product=59&controller=product, accessed 10/2/15

Web link: “Using LIDAR-Lite to sweep an arc for sense-and-avoid”, http://diydrones.com/profiles/blogs/using-lidar-lite-to-sweep-an-arc-for-sense-and-avoid?xg_source=activity, accessed 10/2/15

Web link: “Precision Land ArduCopter Demo”, http://diydrones.com/profiles/blogs/precision-land-arducopter-demo?xg_source=activity, 10/2/2015

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