University of Colorado

Department of Aerospace Engineering Sciences

Design Document

8 May 2015

Drones Versus Zombies (DVZ)

Document Approvals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Name | Affiliation | Approval | Date |
| Customer | Nisar Ahmed | CU/AES |  |  |
| Advisor | Eric Frew | CU/AES |  |  |

Change Summary

|  |  |  |
| --- | --- | --- |
| Revision | Description of Changes | Date Released |
| - | Initial Release | 8 May 2015 |

Team Member Summary

|  |  |
| --- | --- |
| Garrett Brown  Email: [garrett.a.brown@colorado.edu](mailto:garrett.a.brown@colorado.edu)  Approval  Signature: Garrett Brown | Quinn McGehan  Email: [quinn.mcgehan@gmail.com](mailto:quinn.mcgehan@gmail.com)  Approval  Signature: Quinn McGehan |
| Nathan Curry  Email: [Nathan.curry@colorado.edu](mailto:Nathan.curry@colorado.edu)  Approval  Signature: Nathan Curry | Edward Scott  Email: [Edward.scott@colorado.edu](mailto:Edward.scott@colorado.edu)  Approval  Signature: Eddy Scott |
| Tyler King  Email: [tyler.d.king@colorado.edu](mailto:tyler.d.king@colorado.edu)  Approval  Signature: Tyler King | Mark Sakaguchi  Email: [mark.sakaguchi@colorado.com](mailto:mark.sakaguchi@colorado.com)  Approval  Signature: Mark Sakaguchi |
| Austin Lillard  Email: [Austin.lillard@colorado.edu](mailto:Austin.lillard@colorado.edu)  Approval  Signature: Austin Lillard | Benjamin Zatz  Email: [Benjamin.Zatz@colorado.edu](mailto:Benjamin.Zatz@colorado.edu)  Approval  Signature: Ben Zatz |
| Drew Ellsion  Email: Drew.Ellison@colorado.edu  Approval Signature: Drew Ellison | N/A |

Table of Contents

1 Document Scope 6

2 Project Description 6

2.1 Drones Versus Zombies Concept of Operations (CONOPS) 6

2.2 Drone Versus Zombies Architecture 7

3 Drones Versus Zombies System Engineering 8

3.1 Functional Requirements (L1) 8

3.1.1 Drones Versus Zombies (DVZ) Requirements 8

3.2 Element Requirements (L2) 8

3.2.1 Software Element (SWE) Requirements 9

3.2.2 Hardware Element (HWE) Requirements 9

3.3 Subsystem Requirements (L3) 9

3.3.1 Communication Subsystem (CommS) Requirements 9

3.3.2 Mobility Subsystem (MS) Requirements 9

3.3.3 Localization and Mapping/Sensing Subsystem (LAMSS) Requirements 9

4 DVZ Detailed Design 10

4.1 Software Element (SWE) 10

4.1.1 Localization and Mapping/Sensing Subsystem (LAMSS) 10

4.1.2 Communication Subsystem (CommS) 18

4.1.3 Mobility Subsystem (MS) 22

4.2 Hardware Element (HWE) 28

4.2.1 Element Overview 28

4.2.2 Hardware List 28

4.2.3 Design and Analysis Tools and Resources 34

5 Verification and Validation 35

5.1 DVZ Test Philosophy 35

5.2 Test Scenarios 35

5.3 Verification Results 36

6 Project Management 40

6.1 Work Breakdown Structure (WBS) by Phase 40

6.1.1 Phase 1 40

6.1.2 Phase 2 41

6.2 Project Timeline 42

6.2.1 Phase 1 42

6.2.2 Phase 2 43

6.3 Project Budget 44

6.3.1 2014-2015 44

7 Appendix A: DVZ Expenditure Summary 0

List of Tables

Table 2‑1: DVZ CONOPS Steps 7

Table 4‑2: RECUV\_VICON WiFi Network Information 18

Table 4‑3: UCB\_FIXED WiFi Network Information 18

Table 5‑1: Requirements Verification Part 1 39

Table 5‑2: Requirements Verification Part 2 39

Table 7‑1: DVZ Expenditures 0

List of Figures

Figure 2‑1: DVZ Concept of Operations (CONOPS) 6

Figure 2‑2: Spring 2015 DVZ System Architecture 8

Figure 4‑1: Summary of DVZ Sensor Suite 10

Figure 4‑2: Example Occupancy Grid 11

Figure 4‑3: Hector SLAM SW Flow 12

Figure 4‑4: AMCL Step 1 13

Figure 4‑5: AMCL Step 2 13

Figure 4‑6: AMCL Step 3 14

Figure 4‑7: AMCL Step 4 14

Figure 4‑8: AMCL Step 5 15

Figure 4‑9: AMCL Relationship to Pre-generated Map 16

Figure 4‑10: Symmetric Hallway Issue with Laser Scan Matcher 17

Figure 4‑11: ROS Publisher-Subscriber Model 19

Figure 4‑12: DVZ ROS Architecture 20

Figure 4‑13: Pixhawk internal roll/pitch and roll rate/pitch rate controller diagrams 23

Figure 4‑14: PWM to Angle Mapping 24

Figure 4‑15: Altitude PID Controller 25

Figure 4‑16: Velocity PI Controller 26

Figure 4‑17: Position P Controller 27

Figure 4‑18: Flight time analysis results from eCalc 35

Figure 5‑1: Body Velocity Comparison 36

Figure 5‑2: X Hallway Position Hold 37

Figure 5‑3: Y Hallway Position Hold 37

Figure 5‑4: Altitude Hallway Hold 38

Figure 6‑1: DVZ Phase 1 Work Breakdown Structure 41

Figure 6‑2: DVZ Phase 2 Work Breakdown Structure 42

Figure 6‑3: Phase 1 DVZ Timeline 43

Figure 6‑4: Phase 2 DVZ Timeline 44

Figure 6‑5: DVZ Expenditures for 2014-2015 Academic Year 45

Acronyms and Glossary

| Acronym or Term | Definition |
| --- | --- |
| 3DR | 3D Robotics |
| AMCL | Adaptive Monte Carlo Localization |
| BEC | Battery Elimination Circuit |
| CDR | Critical Design Review |
| CommS | Communication Subsystem |
| CONOPS | Concept of Operations |
| DVZ | Drones Versus Zombies |
| EKF | Extended Kalman Filter |
| ESC | Electronic Speed Controller |
| HW | Hardware |
| HWE | Hardware Element |
| IP | Internet Protocol |
| LAMSS | Localization and Mapping/Sensing Subsystem |
| LSM | Laser Scan Matcher |
| MS | Mobility Subsystem |
| PM | Project Manager |
| ROS | Robot Operating System |
| SE | Systems Engineer |
| SegInt | Segment Integration |
| SEIT | Systems Engineering Integration, and Test |
| SW | Software |
| SWE | Software Element |
| TE | Test Engineer |
| UCB | University of Colorado Boulder |
| WBS | Work Breakdown Structure |

# Document Scope

This document describes the detailed design for the Drones Versus Zombies (DVZ) graduate project at the University of Colorado Boulder (UCB). The current version reflects project progress through the Spring 2015 Semester.

# Project Description

The DVZ project began in Fall 2014 and centers on the design and implementation of a cooperative system of multiple unmanned multi-rotor vehicles (multi-rotors) with the goals of environment navigation and target identification as a demonstration of autonomous search functionality in volatile environments. The system being designed by the DVZ team will consist of a minimum of two multi-rotors coupled with a ground station and will be capable of navigating a predetermined map and identifying and differentiating targets based on predetermined identifiers. The multi-rotors comprising the final DVZ system will be capable of coordination with each other via the ground station to provide for efficient search of a given environment.

## Drones Versus Zombies Concept of Operations (CONOPS)

The DVZ system will be capable of autonomously navigating a pre-mapped environment while facilitating coordination between multiple robots to carry out object identification based on the Concept of Operations (CONOPS) presented in Figure 2‑1 and the corresponding steps in Table 2‑1.

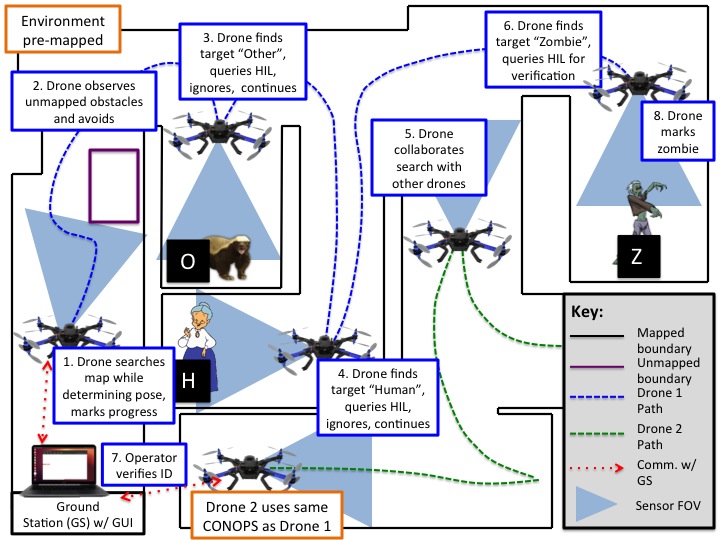


Figure ‑: DVZ Concept of Operations (CONOPS)

Table ‑: DVZ CONOPS Steps

|  |  |
| --- | --- |
| CONOPS Step | Step Description |
| 1 | Each DVZ drone searches a pre-mapped environment while determining its spatial location and orientation (pose) in real time to mark progress and facilitate navigation and control. |
| 2 | While searching the pre-mapped environment, each DVZ drone avoids unmapped and unexpected obstacles to allow for safe operation and to limit damage to the environment and drone. |
| 3 | A DVZ drone identifies an object marked as “Other” based on a predetermined identifier, the letter “O” on a black background, queries the Human in the Loop (HIL), and continues searching after determining that the object marked as “Other” is not of interest. |
| 4 | A DVZ drone identifies an object marked as “Human” based on a predetermined identifier, the letter “H” on a black background, queries the Human in the Loop (HIL), and continues searching after determining that the object marked as “Human” is not of interest. |
| 5 | During the environment search, all DVZ drones coordinate with each other to allow for efficient and thorough search methodology. |
| 6 | A DVZ drone identifies an object marked as “Zombie” based on a predetermined identifier, the letter “Z” on a black background and queries the Human in the Loop (HIL). |
| 7 | The Human Operator verifies that the object marked “Zombie” is of interest via a Graphical User Interface (GUI). |
| 8 | The object marked “Zombie” is included in the environment map and the DVZ drone continues the environment search. |

## Drone Versus Zombies Architecture

The project architecture for the DVZ system was created based on flowdown from customer requirements and the CONOPS and has been approved by the customer and advisor. Key points of the DVZ architecture are detailed in Figure 2‑2 and are summarized as follows:

* The architecture allows for sensor data processing both onboard the Multirotor on the ODROID U3 and on the ground station computer. This flexibility allows for easier implementation of new packages and large fluctuations in necessary processing power.
* Key system software runs on the ODROID U3 while data processing is primarily carried out on the ground station.
* The architecture calls for no communication between the robots comprising the DVZ system, rather the robots communicate with a central ground station.
* Communication between elements of the DVZ system occurs via WiFi which is assumed to be available in the environment per customer direction.
* Communication between subsystems of the DVZ system occurs via the Robot Operating System (ROS) which utilizes a publisher-subscriber model to communicate between hardware and SW nodes. The ground station serves as the ROS master. Further discussion of ROS and the ROS architecture used for DVZ can be found in Section 4.1.2.2 of this document.

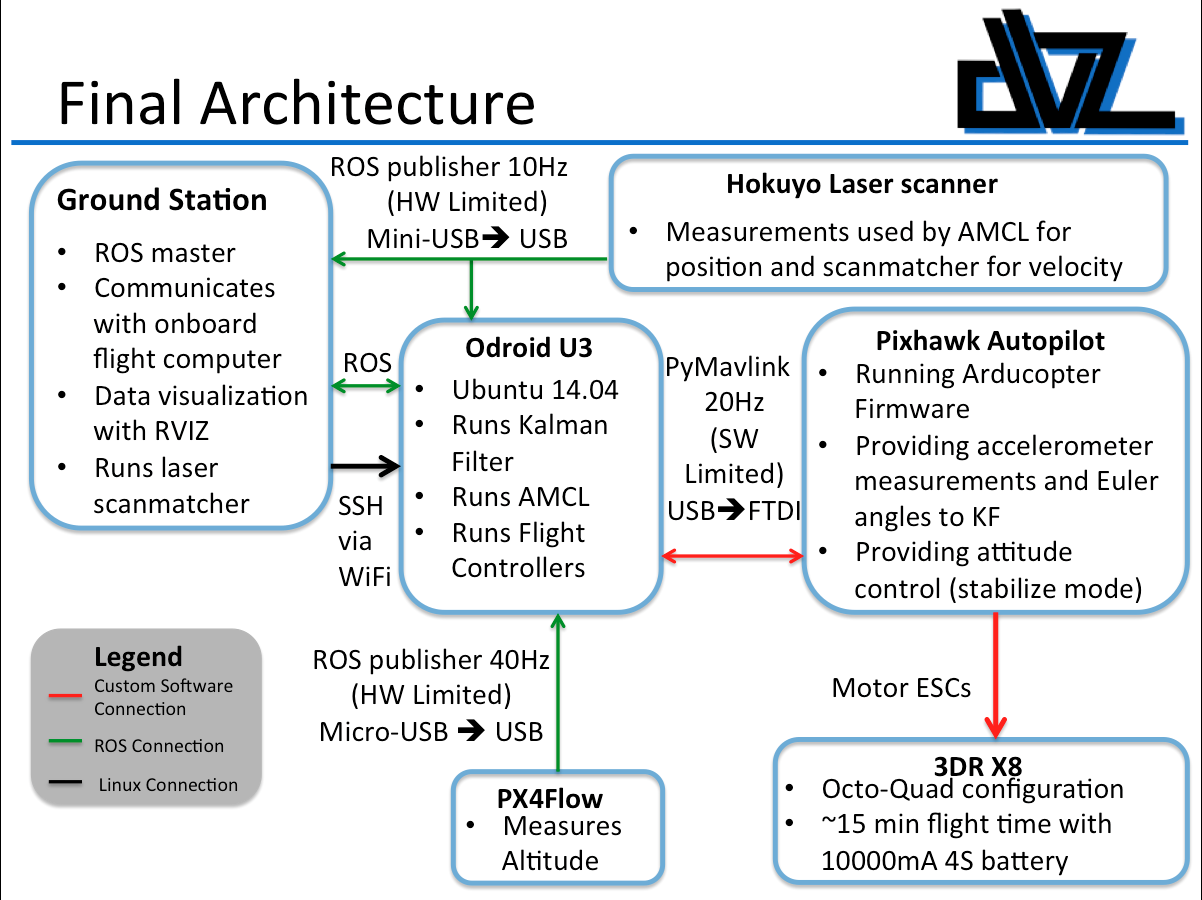


Figure ‑: Spring 2015 DVZ System Architecture

# Drones Versus Zombies System Engineering

Based on the DVZ CONOPS and architecture, DVZ functionalities were defined from top-level customer requirements flowed into lower-level functionalities. DVZ requirements are organized into 3 levels, functional (Level 1), element (Level 2), and subsystem (Level 3) based on increasing specificity. While the DVZ requirements are derived based on the specific test scenario, the Drones Versus Zombies game, the requirements were developed with broader applications in mind to allow for easy modification of DVZ design tenets for more practical applications. DVZ requirements are fully captured in the requirement specification documents and brief descriptions are included here for completeness only.

## Functional Requirements (L1)

### Drones Versus Zombies (DVZ) Requirements

Drones Versus Zombies (DVZ) Requirements represent customer desires regarding system functionality and represent the minimum acceptable functionality for whole system behavior and performance. DVZ requirements can be found at the following location:

* /DVZ/Systems Engineering/Requirements Specification/Drones Versus Zombies (DVZ) Requirements (L1)

## Element Requirements (L2)

DVZ functionality is broken into elements that encompass specific disciplines of functionality including Software development and Hardware development. DVZ software functionality is the responsibility of the Software Element (SWE) and the associated functionality is captured in an associated requirements specification. DVZ hardware functionality is the responsibility of the Hardware Element (HWE) and the associated functionality is captured an associated requirements specification.

### Software Element (SWE) Requirements

DVZ Software Element (SWE) requirements represent high level software (SW) functionalities necessary for successful implementation of the DVZ system. SWE requirements were developed via system architecture analysis and represent the minimum acceptable behavior and performance for all DVZ SW functionalities. SWE requirements can be found at the following location:

* /DVZ/Systems Engineering/Requirements Specification/Element Requirements/Software Element (SWE) requirements

### Hardware Element (HWE) Requirements

DVZ Hardware Element (HWE) requirements represent high level hardware (HW) functionalities necessary for successful implementation of the DVZ system. HWE requirements were developed via system architecture analysis and represent the minimum acceptable behavior and performance for all DVZ HW functionalities. HWE requirements can be found at the following location:

* /DVZ/Systems Engineering/Requirements Specification/Element Requirements/Hardware Element (HWE) requirements

## Subsystem Requirements (L3)

Due to the complexity of the DVZ SW system, DVZ SWE functionality is broken into SW subsystems that encompass specific SW functionalities necessary for successful operation of the DVZ system. Sensing and localization functionality is the responsibility of the Localization and Mapping/Sensing Subsystem (LAMSS), system mobility and path planning are the responsibilities of the Mobility Subsystem (MS), and communication and modeling functionalities are the responsibilities of the Communications Subsystem (CommS).

### Communication Subsystem (CommS) Requirements

DVZ Communication Subsystem (CommS) requirements represent the low level SW functionalities necessary for successful implementation of the DVZ communication system. CommS requirements were developed via system architecture analysis and represent the minimum acceptable behavior and performance for all DVZ communication functionalities. CommS requirements can be found at the following location:

* /DVZ/Systems Engineering/Requirements Specification/Subsystem Requirements/Communication Subsystem (CommS) requirements

### Mobility Subsystem (MS) Requirements

DVZ Mobility Subsystem (MS) requirements represent the low level SW functionalities necessary for successful implementation of the DVZ mobility and path planning system. MS requirements were developed via system architecture analysis and represent the minimum acceptable behavior and performance for all DVZ mobility functionalities. MS requirements can be found at the following location:

* /DVZ/Systems Engineering/Requirements Specification/Subsystem Requirements Mobility Subsystem (MS) requirements

### Localization and Mapping/Sensing Subsystem (LAMSS) Requirements

DVZ Localization and Mapping/Sensing Subsystem (LAMSS) requirements represent the low level SW functionalities necessary for successful implementation of the DVZ sensing, localization, and mapping system. LAMSS requirements were developed via system architecture analysis and represent the minimum acceptable behavior and performance for all DVZ sensing, localization, and mapping functionalities. LAMSS requirements can be found at the following location:

* /DVZ/Systems Engineering/Requirements Specification/Subsystem Requirements Localization and Mapping/Sensing Subsystem (LAMSS) requirements

# DVZ Detailed Design

## Software Element (SWE)

The DVZ Software Element (SWE) is responsible for the design and implementation of all SW aspects of the DVZ system and for integration of Free and Open Source Software (FOSS) into the DVZ system. The SWE is composed of three subsystems which are organized based on related functionality: The Localization and Mapping/Sensing Subsystem (LAMSS), the Communication Subsystem (CommS), and the Mobility Subsystem (MS).

### Localization and Mapping/Sensing Subsystem (LAMSS)

The Localization and Mapping/Sensing Subsystem (LAMSS) is responsible for all aspects of the DVZ system necessary to map a given environment and then interpret that environment to allow for localization of the drone. LAMSS functions include design and implementation of environment mapping along with localization and pose estimation. During future DVZ work, visual sensing necessary for target identification and differentiation will also fall under the purview of the LAMSS.

To accomplish all sensor functions necessary for DVZ sensor functionality, a sensor suite consisting of a 3D Robotics (3DR) PX4 Flow and a Hokuyo URG-04LX was selected to provide attitude, velocity, and ranging measurements as shown in Figure 4‑1. The PX4 Flow provides an altitude measurement via an ultrasonic sensor and is also capable of providing velocity estimates based on a downward facing camera that analyzes differing pictorial frames. The velocity capabilities provided by the PX4 Flow are not currently called for in the DVZ design, however, the capability remains an option for lateral velocity estimation. The Hokuyo implements range finding via LiDAR and can be used to estimate both position and velocity when coupled with the Laser Scan Matcher (LSM) software discussed below. In addition to the PX4 Flow and Hokuyo, acceleration and angular rate measurements are taken from the IMU included in the 3DR Pixhawk. Specific selection details for the DVZ Sensor suite are discussed in greater depth in the Hardware section of this document.

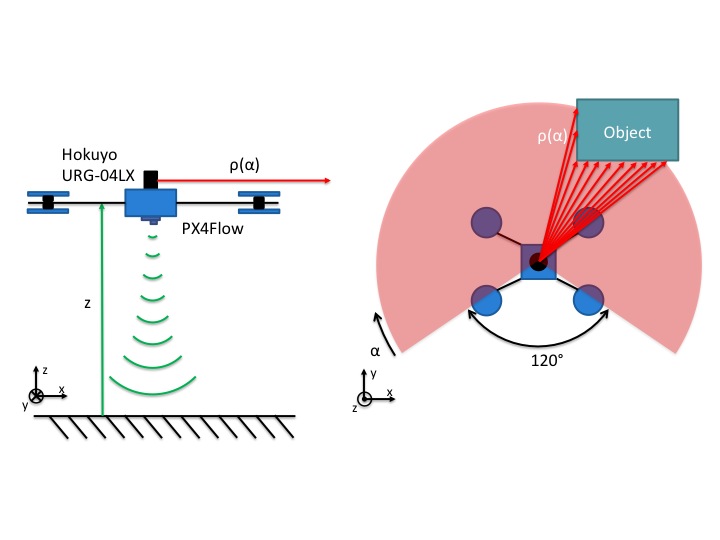


Figure ‑: Summary of DVZ Sensor Suite

#### Environment Mapping

A machine-readable map for the purpose of robotics can be generated by creating a discrete representation of a given environment and then using numerical identifiers to denote cells which are occupied. Such a discrete representation, an example of which can be found in Figure 4‑2 below, is termed an occupancy grid and can be used to represent an arbitrarily complex environment.

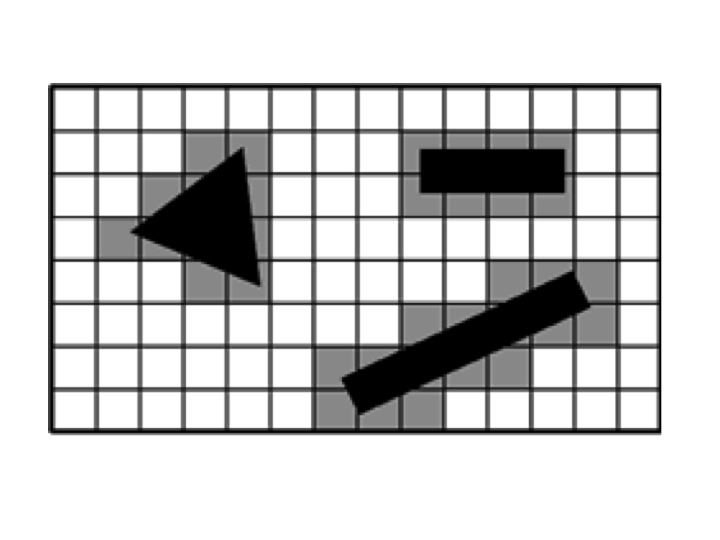


Figure ‑: Example Occupancy Grid

While the occupancy grid in Figure 4‑2 represents a two dimensional environment, an occupancy grid can be extended to three dimensions by creating a three dimensional discretization of space. Arbitrarily complex environments can be represented by adjusting the fidelity of an occupancy grid either locally or globally, however, the size of the occupancy grid can impact the computational speed of the system.

An environment can be reduced to an occupancy grid via many methods ranging from manual entry of environment details into a computer environment to real-time mapping via a sensor or fusion of sensors, however, because the DVZ system already employs sensors capable of rangefinding, the most logical method for map generation is to leverage existing sensing capabilities. To that end, offline environment mapping for DVZ is implemented via laser scan measurements processed with the Robot Operating System (ROS) Hector Map package which is part of the larger Hector Simultaneous Localization and Mapping (SLAM) package. While the DVZ design will not leverage SLAM capabilities provided by Hector SLAM, the Hector Map package does provide a method for generating an occupancy grid corresponding to an arbitrary environment. The Hector SLAM package employs the I/O architecture depicted in Figure 4‑3 where it should be noted that generation of the occupancy grid requires knowledge of the drone location (but not necessarily the full pose). While it is possible to use a fully coupled SLAM solution as part of the DVZ architecture, the introduction of multiple vehicles and desire for efficient path planning and search is more easily handled via a separate architecture for mapping and localization.

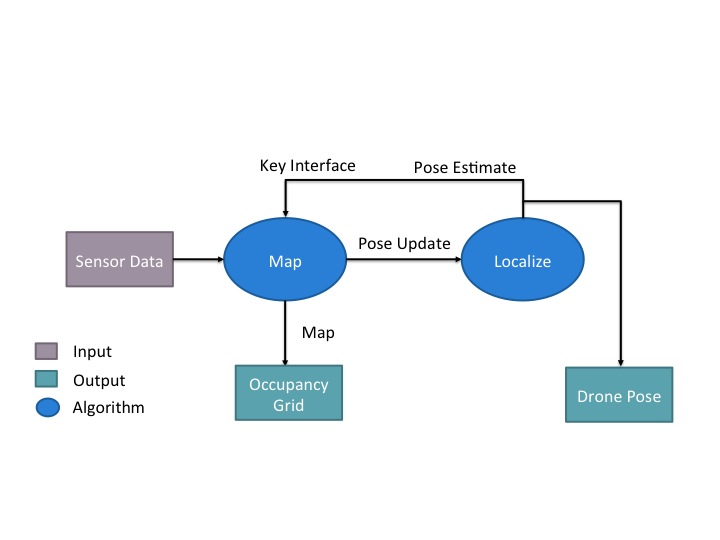


Figure ‑: Hector SLAM SW Flow

Because the DVZ system is assumed to fly at constant or near-constant altitude, the flight environment is assumed to be constant with respect to the vertical direction, thus decrease processing complexity and the necessary storage required, the occupancy grid used as an input to the DVZ system is two dimensional. During testing, the assumption of a two-dimensional map proved acceptable, however, because the laser scanner only provides a two dimensional measurement, it was necessary to account for even minor variations in the roll and pitch angles of the flight platform. When the platform rolls or pitches, the Hokuyo records a distance greater than the actual distance, thus, to ensure that the distances reflected in the map correspond to actual laser measurements, a trigonometric correction factor must be applied. Further discussion of sensing adjustments to account for vehicle dynamics is made in the following sections.

#### Localization

Based on the pre-mapped environment generated using Hector mapping, the DVZ system implements a localization via Adaptive Monte Carlo Localization (AMCL) which uses statistical modeling to estimate the position and orientation of a robot within an environment. AMCL requires sensor inputs which defined the robot velocity and position estimates for a given time along with an estimate of the robot motion captured by the Odometry frame. For the purposes of DVZ, the Odometry estimate is provided by a Laser Scan Matcher (LSM) while the sensor information is fused via an Extended Kalman Filter (EKF). The AMCL, LSM, and EKF implementations are discussed below.

##### Adaptive Monte Carlo Localization (AMCL)

To provide for localization in the

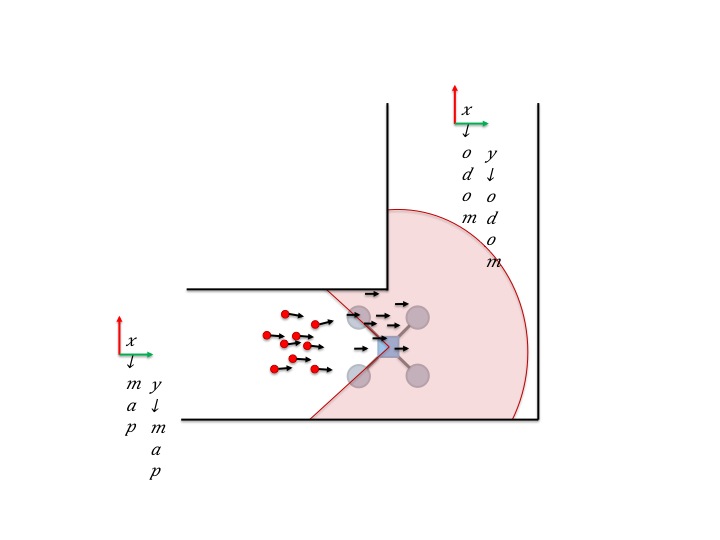


Figure ‑: AMCL Step 1

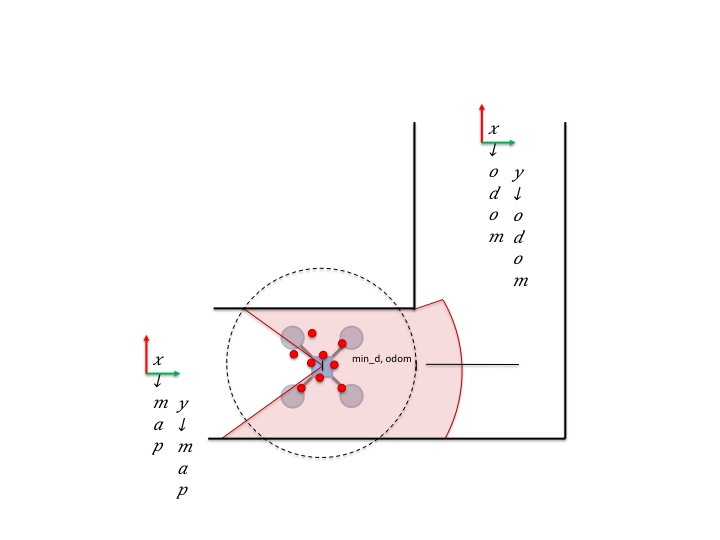


Figure ‑: AMCL Step 2

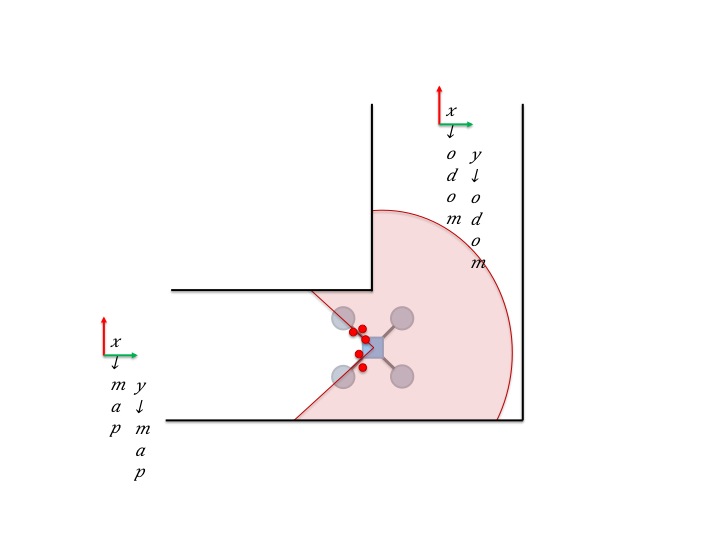


Figure ‑: AMCL Step 3

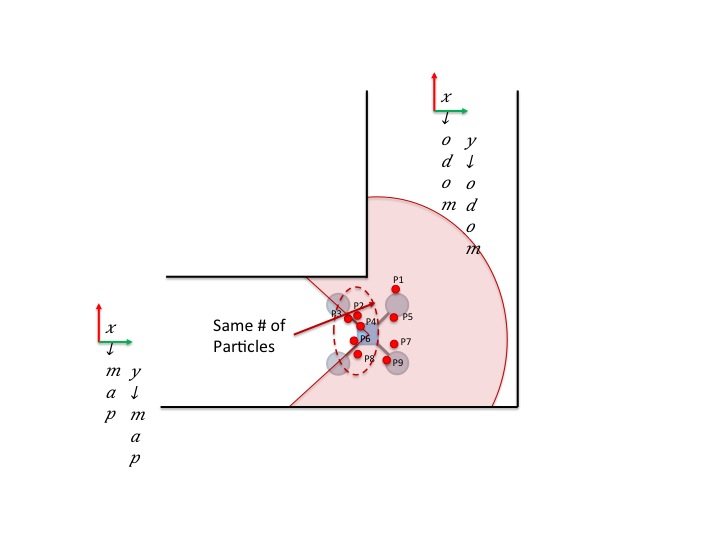


Figure ‑: AMCL Step 4

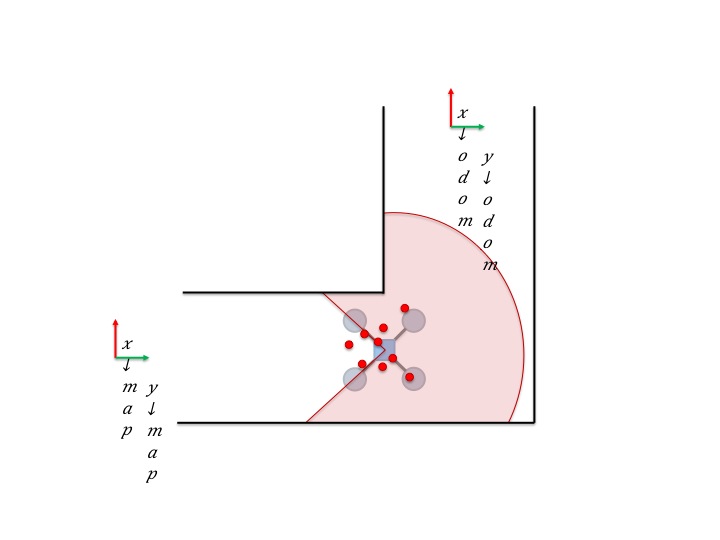


Figure ‑: AMCL Step 5

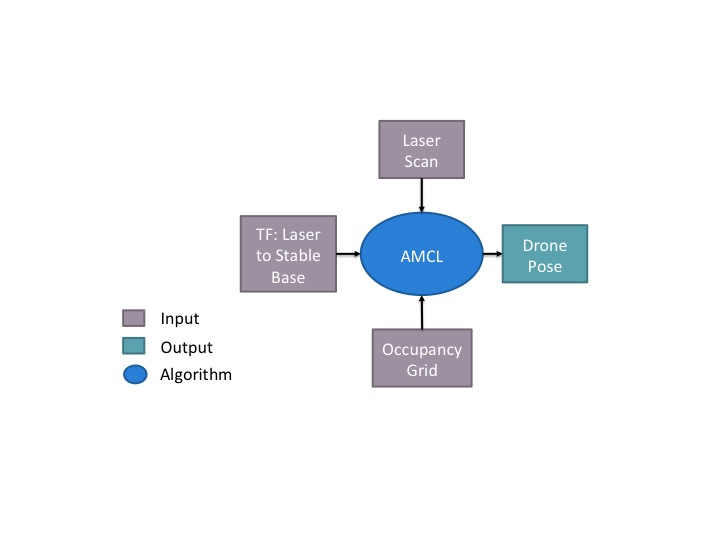
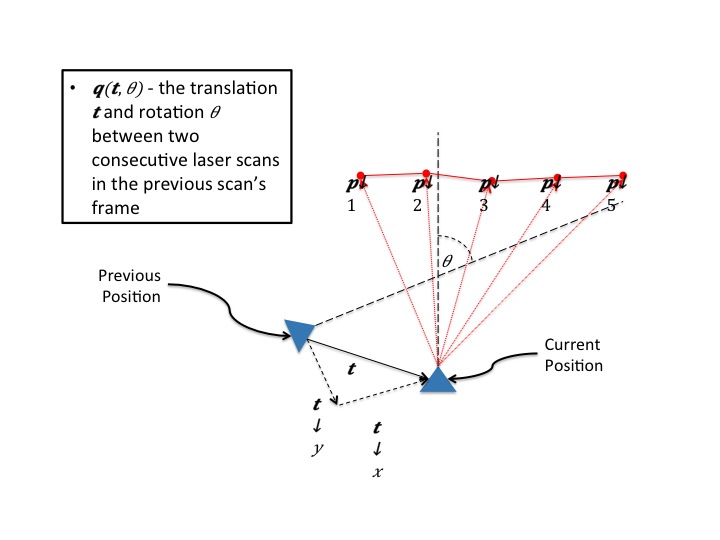
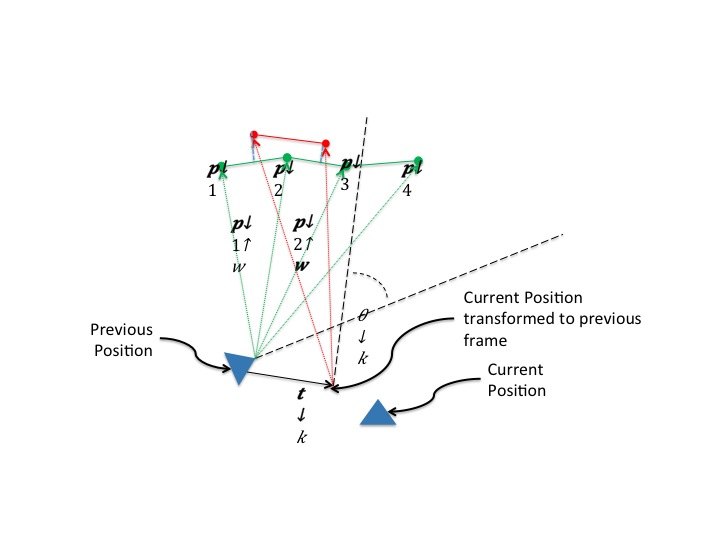


Figure ‑: AMCL Relationship to Pre-generated Map

##### Odometry and Laser Scan Matcher





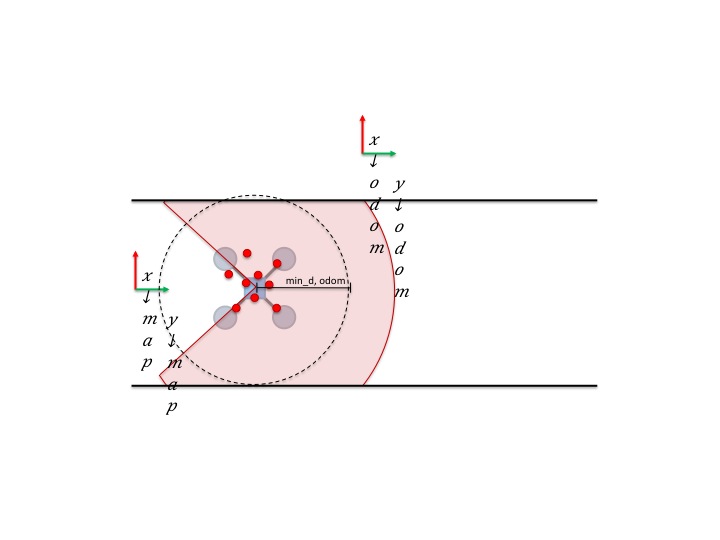


Figure ‑: Symmetric Hallway Issue with Laser Scan Matcher

##### Sensor Fusion and Filtering

Discuss Kalman Filter with equations

### Communication Subsystem (CommS)

The Communication Subsystem (CommS) is responsible for all aspects of the DVZ system necessary to facilitate communication between software and hardware components and between robots and the ground station. In addition to communication functionalities, the CommS is responsible for full system modeling via Gazebo SW.

#### Low Level Communication

##### Wireless Network

Based on customer requirements and due to the implementation of the DVZ system in an indoor environment, all low level communication for the DVZ system will be implemented via WiFi. In order for the high level communication protocols to function properly, the ground station must have the ability to ssh into the robots’ onboard computers. To achieve this, either the hostname or the ip address of all platforms on the network must be known. Due to the complexity of the UCB network, and firewall restrictions, static IP addresses and DNS names for the ground station and the robots were obtained from OIT. Thus, when testing in the Lockheed Martin Hallway the platforms on the network are assigned static IP addresses on the SSID UCB\_FIXED (as opposed to UCB\_WIRELESS). More specifically the MAC address of the particular WiFi card is assigned an address, so each WiFi dongle should only be used for one robot. This is not an issue for the ground station as it has an onboard WiFi card.

In order to maintain as consistent testing environments as possible, static ip addresses were also set up on the RECUV\_VICON network. The range of static ip addresses for DVZ on the RECUV network is currently 192.168.20.40 to 192.168.20.49. Ip addresses can be added to RECUV\_VICON via connecting to the RECUV\_VICON router in a web browser. For specific instructions and login info refer to the actual router. The ip addresses for both networks for the current platforms are summarized in Tables XX and XX2. The hostnames in the /etc/hosts files on each of the computers have been changed to the DNS name that was provided by OIT. This name is also the DNS name given to each specific platform on RECUV\_VICON. Consistency in the DNS names associated with each ip address and the hostnames onboard each of the platforms helps keep the functional on either RECUV\_VICON or UCB\_FIXED with no effort needed to transition between them.

Table ‑: RECUV\_VICON WiFi Network Information

|  |  |  |  |
| --- | --- | --- | --- |
| Device | Platform | MAC Address | IP Address |
| Cerebro | Ground Station | D0:7E:35:07:D4:B0 | 192.168.20.40 |
| Wolverine | Quad | 7C:DD:90:52:13:98 | 192.168.20.41 |
| Magneto | Quad | 7C:DD:90:52:1A:AD | 192.168.20.42 |

Table ‑: UCB\_FIXED WiFi Network Information

|  |  |  |  |
| --- | --- | --- | --- |
| Device | MAC Address | IP Address | DNS |
| Cerebro | D0:7E:35:07:D4:B0 | 10.201.0.12 | Engr2-0-12-fixed |
| Wolverine | 7C:DD:90:52:13:98 | 10.201.0.10 | Engr2-0-10-fixed |
| Magneto | 7C:DD:90:52:1A:AD | 10.201.0.11 | Engr2-0-11-fixed |

#### High Level Communication

##### Robot Operating System (ROS)

The higher level communication system is implemented via the Robot Operating System (ROS). ROS was selected because of the significant library of pre-existing open source packages that help minimize the amount of custom code. In addition to providing significant existing code, ROS also facilitates the low level communication between functions and sensors, thus as long as all DVZ entities have sufficient WiFi connectivity, ROS can be employed to carry out all necessary data gathering and distribution throughout the system.

A system implemented with ROS consists of nodes that interact via a publisher/subscriber model as depicted in Figure 4‑11. The publisher/subscriber model means that each hardware or software node (e.g. the Hokuyo or AMCL respectively) publishes a specific, configurable set of data to a topic which can then be retrieved or utilized either in part or in whole by other nodes. The ROS publisher/subscriber model allows for the passage of information across networks, so nodes residing on individual platforms can pass information via various communications protocols including WiFi and Ethernet. To maintain order within the ROS architecture, one platform is designated as the master which makes it responsible for maintaining the topic list and facilitating data transfer while the other platforms comprising the system are slaves and only access or publish data. In addition to the publisher/subscriber model, and as can be seen in Figure 4‑11, ROS nodes can also communicate via services which are commands sent directly from one node to another. Currently the DVZ system does not utilize any services. Additional information on ROS, including tutorials, can be found at:

* [www.ros.org](http://www.ros.org).

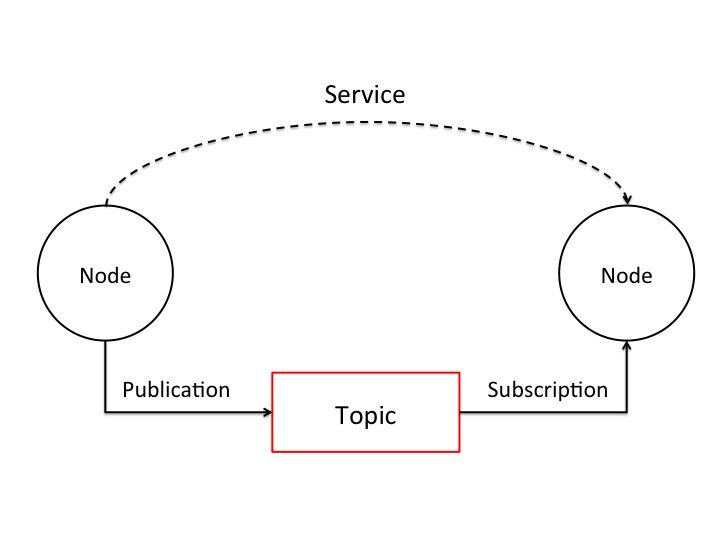


Figure ‑: ROS Publisher-Subscriber Model

The ROS architecture implemented for the DVZ system can be seen in Figure 4‑12. For the purposes of DVZ, all DVZ entities are running ROS Indigo on Ubuntu 14.04 operating systems. The figure is a depiction of the system architecture with the software nodes depicted as blue ovals, and the information they publish depicted as red squares. The communication between the sensors, depicted as black squares, is handled by pre-existing ROS packages. This information is then published to topics that are subscribed to by other pre-existing and other custom scripts. For this iteration all of the custom scripts have been developed in Python.

The localization node in Figure 4‑12 can be seen broken out further in Figure 4‑13. AMCL is a pre-existing package that has been discussed at length in Section 4.1.1.2.1. This node subscribes to a pre-existing map as well as topics containing odometry and range information. The range information is published as a topic called \scan by the pre-existing package used to interface with the Hokuyo. The odometry information is published to a topic called \odom by a custom script that integrates output of the custom implemented Kalman filter. AMCL, in turn, publishes to a topic that contains information about the position of the robots within the predefined map. Once fully developed, this information will then be passed to the path planner, and the custom Nav node to determine where to move and how to get there.

Because ROS handles the low level communication the physical layout of the architecture is not depicted in Figure 4‑12 and Figure 4‑13. This is because most of the nodes can be run either on the ground station or on the mobile robots. The only packages that must be run onboard the robots are the packages that interface directly with the sensors. Currently, the idea is to run all of the localization and control packages onboard the robots, and leave the path planning for the ground station. With this configuration the cpu usage is more spread out, as opposed to being centralized on the ground station, which would make scaling of the project more difficult.

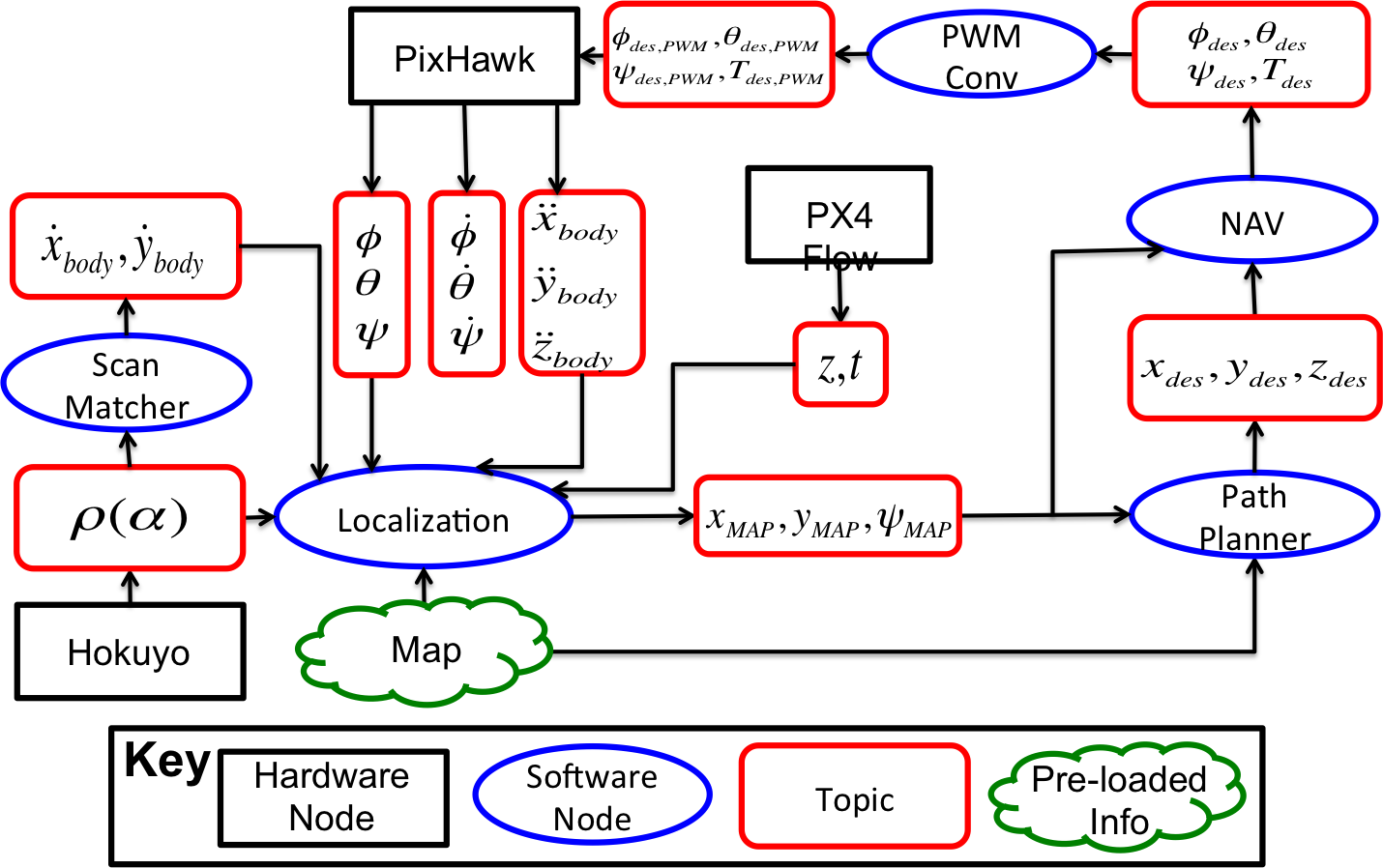


Figure ‑: DVZ ROS Architecture

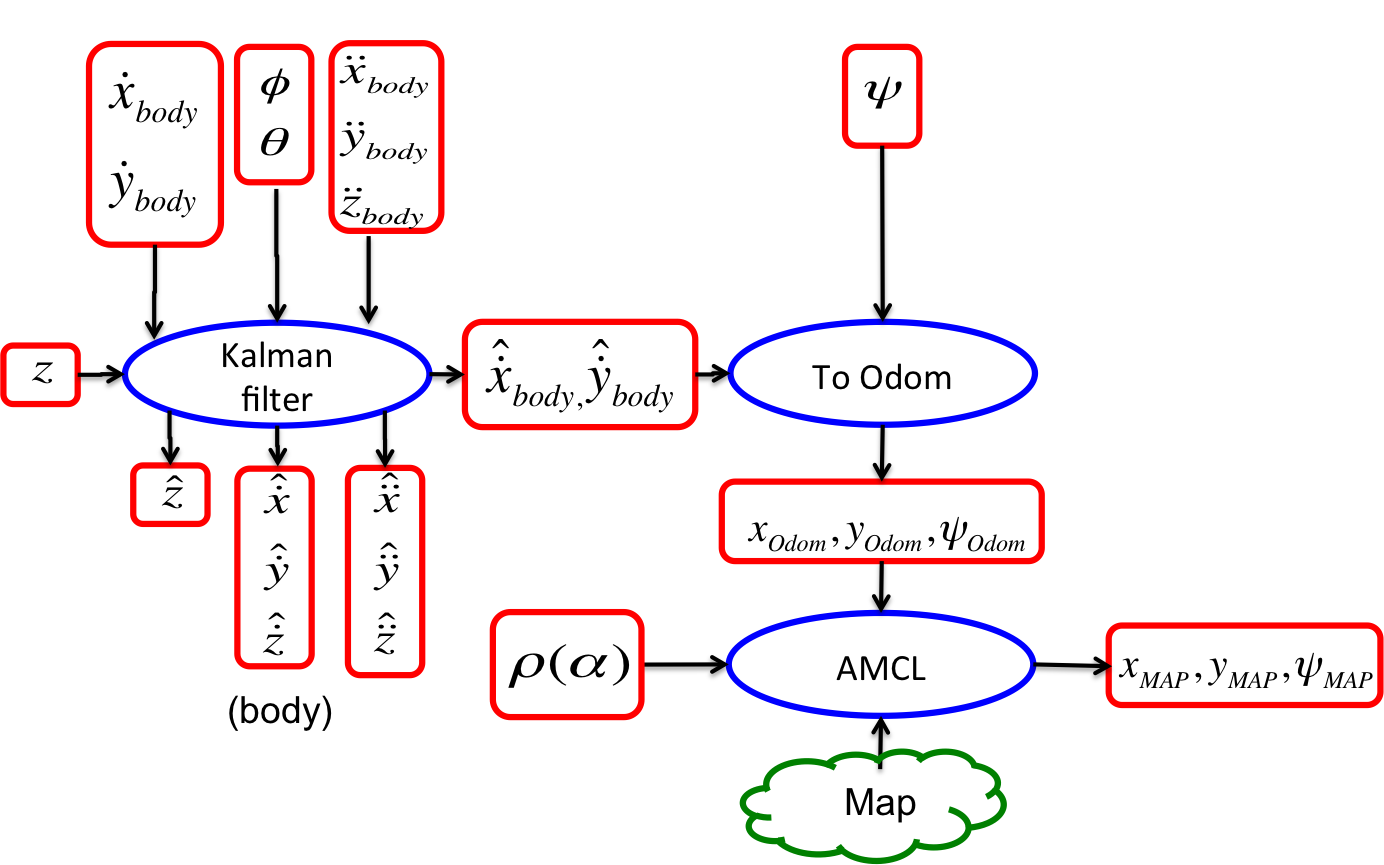


Figure 4-13: Localization ROS Architecture

The previous figures depict the high level ROS architecture. A figure of the actual implementation running real time can be found on the dropbox under:

* /DVZ/Verification\ Validation\ Test/rosgraph\_everything\_running\_parallel.png.

Once again, the nodes are represented as ovals and the topics as squares. This figure was obtained using the package rat\_graph, which is a helpful external tool for visualizing the full system in real time. Further, it is able to provide data statistics about various components of the network including latency, data usage, and subscription rates. It is highly recommended that this package be utilized for debugging any potential network problems.

#### Modeling and Gazebo

One of the major bottlenecks in project development is in algorithm testing. Currently as an algorithm is developed it is tested via flight test. However, this takes quite a bit of time to setup the flight tests, wait for batteries to recharge, etc. Thus, a robust accurate simulation of the system is quite desired. A software simulation would ultimately allow testing to occur much more rapidly than it currently does. To this end, the current simulator of choice is Gazebo. Similarly to ROS, Gazebo was selected as the platform to simulate the system upon due to its pre-existing support of robot simulation and integration with ROS. It is recommended by the creators that version 2.x of Gazebo be used when integrating with ROS.

The system modeling is currently still in more of a research phase than an implementation phase. The current plan is to modify an existing Gazebo simulation for use by DVZ. The existing simulation is called Quadrotor Indoor SLAM Demo, and can be found at <http://wiki.ros.org/hector_quadrotor>. The simulation currently uses the ROS Hector SLAM package, and can be controlled via an xbox controller. However, examining the rqt\_graph image of the ros architecture found in:

* /DVZ/Verification\ Validation\ Test/rosgraph\_hector\_slam\_demo.png

allows insight into how it might be modified for use by the DVZ system. First off, the xbox controller controls the quad in gazebo via publishing to a rostopic called \geometry\twist, which commands the desired translational and rotational velocities of the quad. If the DVZ system were to publish its desired velocities to this topic, it would essentially replace the xbox controller. Next, the Hector Slam section needs to be replaced by AMCL. Finally, the environment that the quad is in must be changed to the desired environment (e.g. the Lockheed Martin Hallway). All of the Hector quad simulation code is available on git, and can be found via the previous link.The modification of the simulation to be DVZ compatible has not begun yet though.

### Mobility Subsystem (MS)

The Mobility Subsystem’s main function is to provide the drone system the ability to move throughout its environment. The Mobility Subsystem stretches from the high level path planning ability of the drones down to the low level controllers that control the altitude, velocity, and position of the drones. The Mobility subsystem does not include control of the drone’s individual motors as that level of control is handled by the drone’s autopilot system which the Mobility Subsystem interfaces with.

#### Controls

The altitude, velocity, and position controllers were written in Python using the Pymavlink API. For initial testing, no external sensors i.e. the PX4Flow and Hokuyo Laser Rangefinder were used in the development and tuning of the controller to prevent unnecessary damage. The development and tuning of the controllers involved using RECUV’s Vicon motion capture system to feedback inertial (x,y,z) position and the Pixhawk’s internal measurements of the Euler angles to control the drone system.

The controller scripts communicate to the Pixhawk autopilot system using the Pymavlink API which allows MAVLink (Micro Air Vehicle Communication Protocol) messages to be received and transmitted to and from the Pixhawk autopilot. In order to control the drone platform, the controller scripts, which are written in Python, outputs PWM values for the roll, pitch, throttle and yaw channels and are sent to the Pixhawk autopilot over the “RC\_CHANNELS\_OVERRIDE\_SEND” MAVLink message channel. By using the “RC\_CHANNELS\_OVERRIDE\_SEND” Mavlink message channel, the controller scripts are basically overriding the PWM values that would be received from the RC Handset and are instead being received from the controller scripts. In addition to the roll, pitch, throttle and yaw channels begin overwritten, Channel 5, 6, and 7 (toggle switch channels) are also being overwritten such that they can be used to start/stop the controller scripts.

The Pixhawk autopilot is currently set to fly in “Stabilize Mode”. In manual flight, Stabilize Mode allows the pilot to effectively control the roll, pitch and yaw angles of the drone platform i.e. a roll PWM value of 1600 (mid-stick at ~1500) will roll the drone platform to the right by a small roll angle. If the pilot hold the roll stick at a PWM value of 1600, the drone platform will stay at the achieved roll angle and then go back to level hover flight once the roll stick is re-centered. This holds true for the roll, pitch and yaw channels. Stabilize Mode was chosen to take advantage of the internal angle and angle rate controllers (roll, pitch, and yaw) which are controlled by the Pixhawk autopilot. The Stabilize Mode setup can be seen below.

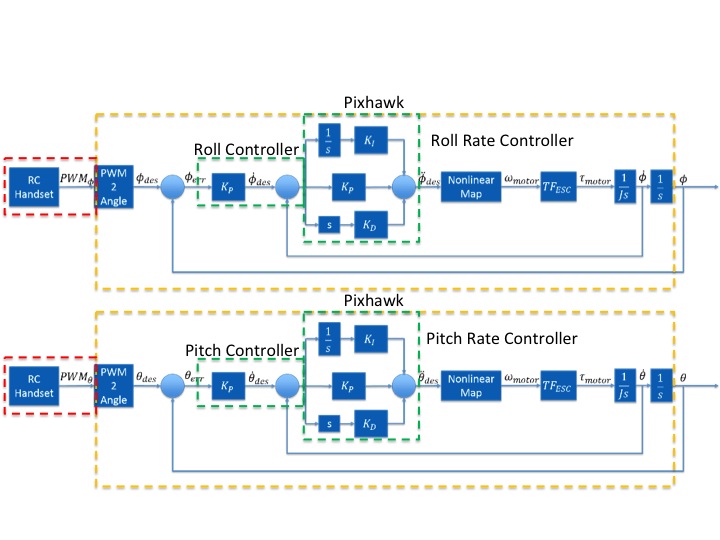


Figure ‑: Pixhawk internal roll/pitch and roll rate/pitch rate controller diagrams

As one can see, when flying the drone platform in Stabilize Mode, the PWM values sent from the RC Handset first pass through a PWM2Angle function internal to the Pixhawk. This PWM2Angle function takes in the sent PWM value and, depending on which channel it is being sent over (Channel 1 = Roll, Channel 2 = Pitch, Channel 4 = Yaw), uses the RC Handset calibrated channel-max, channel-min, channel-trim and channel-deadzone parameters along with the angle-max parameter to convert the sent PWM value to a desired roll, pitch or yaw angle. This conversion assumes a linear mapping from a PWM value (PWM on the approximate range ) to angle (angle on the range ) with a set channel-deadzone width centered around channel-trim value. An example of this mapping PWM2Angle can be seen in Figure 4‑14.

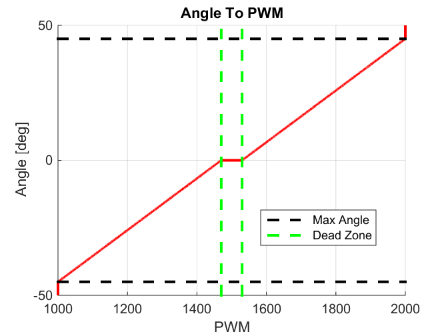


Figure ‑: PWM to Angle Mapping

Once the desired PWM values have been converted to desired Euler angles, the desired Euler angles are sent to first the angle controller which is just a proportional controller which then outputs a desired Euler angle rate. This desired Euler angle rate is then sent to the angle rate PID controller. This angle rate controller then outputs a desired Euler angle acceleration where then additional conversions are applied and the individual motor speeds are adjusted to achieve the desired Euler angle.

For the development of the controller scripts, it was crucial to understand the PWM2Angle function that was implemented in the Pixhawk autopilot firmware. This function was then inverted to create an Angle2PWM function that was implemented in the controller scripts such that a desired Euler angle could be generated by the velocity controller, converted to the corresponding PWM value, sent to the Pixhawk autopilot, converted back to the correct desired Euler angle, and then sent to the internal Euler angle and angle rate controllers. The Angle2PWM function implemented in the controller scripts can be found in the “controllers\_func\*.py” scripts for both the Wolverine and Magneto platforms in “DVZ/Software Element\Mobility Subsystem\Controller Scripts\” and then the corresponding Wolverine or Magneto folder.

#### Altitude Controller

The altitude controller implements a PID controller scheme to control the altitude of the drone platform. This PID architecture can be seen below.

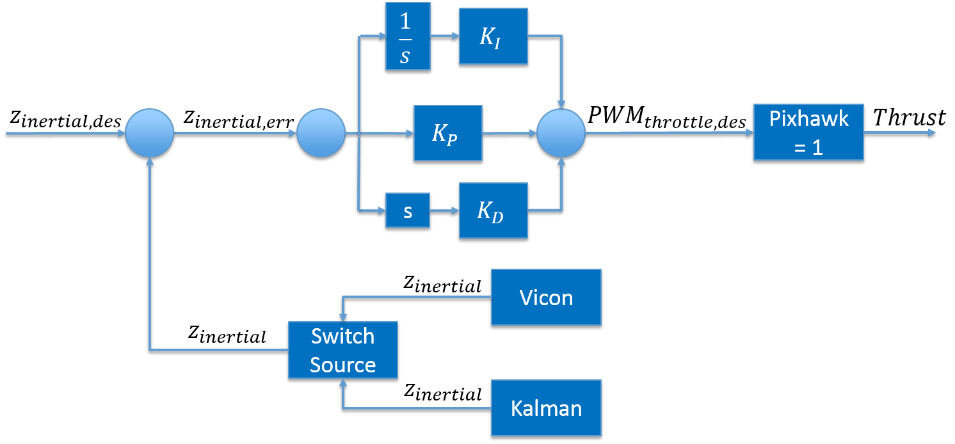


Figure ‑: Altitude PID Controller

Inertial z-measurements (altitude measurements) are input to the controller either from the Vicon motion capture system or the output from the Kalman Filter. The desired altitude and current altitude are input to the PID gain terms which are then summed together along with a base-throttle PWM value to calculate the desired PWM throttle value which is sent to the Pixhawk autopilot over RC Channel 3. It should be noted that for the current implementation of all of the controller scripts, control of the drone platform is not handed over to the Odroid and controller scripts until the pilot flips Channel 6 (for Wolverine) or Channel 7 (for Magneto) high, i.e. PWM > 1400. Before Channel 6 or Channel 7 is flipped high, the pilot manually takes off the drone platform until it is in stable flight near the desired altitude. For the current implementation of the Altitude Controller, the base-throttle PWM is sampled from the current RC Channel 3 throttle PWM value during the manual piloted flight. As soon the pilot flips Channel 6 or Channel 7 high, the most recent RC Channel 3 throttle PWM value is permanently saved as the base-throttle PWM which is then added to the PID controller values. This handoff from the piloted manual flight to the autonomous controller script flight allows a smooth flight performance transition and prevent the integral term in the Altitude Controller from having to accumulate to get the drone platform up to desired altitude.

If feedback is provided by the Vicon motion capture system:

* z-velocity measurements used in the derivative term of the controller are calculated by taking the current z measurement, subtracting the previous z measurement and then dividing it by a Δt term.

If feedback is provided by the Kalman Filter:

* z measurements are coming from the subscribed to PX4Flow ROS topic. During testing it was noted that the PX4Flow sometimes returns a 0m altitude measurement even in flight. To account for these false 0m readings, the z measurements coming from the PX4Flow ROS topic are first filtered by the Altitude Controller. If a 0m altitude measurement is received, the most recent non-zero altitude measurement is used to account for these dropouts.
* z-velocity measurements should come from the Kalman Filter, which outputs an estimated z-velocity value for the drone platform. However, as of Spring 2015, the z-velocity measurement coming from the Kalman Filter has not proved to be accurate nor reliable and thus the z-velocity measurement should be differentiated from the current and previous z measurements as done when using feedback from the Vicon Motion Capture System.

#### Velocity Controller

The Velocity Controller implements a PI controller scheme to control the velocity of the drone platform. This PI architecture can be seen below.

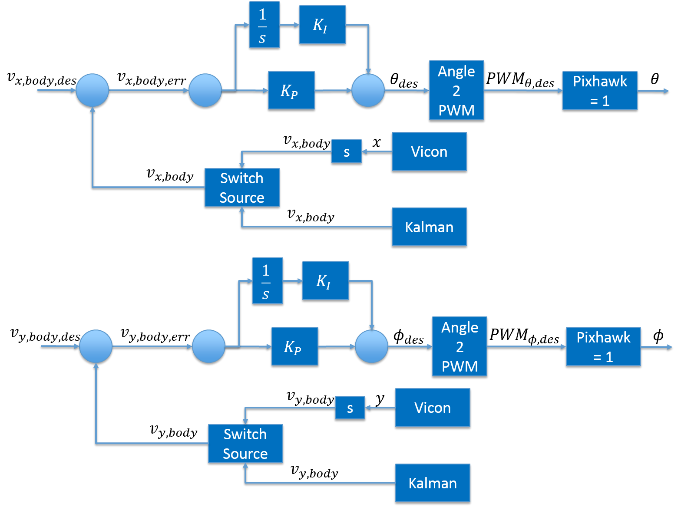


Figure ‑: Velocity PI Controller

Inertial velocity measurements represented in the body platform frame (x out the front of the drone, y out the right of the drone, z pointing down) are input to the controller from either the Vicon motion capture system or the output from the Kalman Filter. The desired body velocities and current body velocities are input to the PI gain terms which are then summed together to calculate the desired pitch angle that will achieve the desired x-body velocity and the desired roll angle that will achieve the desired y-body velocity. The calculated roll and pitch angles are then sent to the Angle2PWM function as mentioned in the 4.1.3.1 Controls section. This function calculates the corresponding desired roll and pitch PWM values which are then sent to the Pixhawk autopilot over RC Channel 1 and Channel 2 respectively. It should be noted that for the current implementation of all of the controller scripts, control of the drone platform is not handed over to the Odroid and controller scripts until the pilot flips Channel 6 (for Wolverine) or Channel 7 (for Magneto) high, i.e. PWM > 1400.

If feedback is provided by the Vicon motion capture system:

* Velocity measurements are calculated by inputting the inertial x and y position of the drone platform measured by Vicon along with the inertial yaw measurement measured by Vicon. Since these measurements are in the Vicon inertial frame (which is a right-handed coordinate frame but with the z-axis pointing up, different from the drone platform body frame where the z-axis is pointing down) differentiating the inertial x and y position measurements results in the inertial velocity of the drone platform represented in the inertial frame. Using the inertial yaw measurement measured by Vicon, the inertial velocities of the drone platform are rotated into the drone platform’s body frame such that the PI Velocity Controller can directly calculate the needed roll and pitch angles to achieve the desired velocity.

If feedback is provided by the Kalman Filter:

* Velocity measurements from the Kalman Filter are already calculated in the body frame so no rotation from the inertial frame to the body frame need to occur.

#### Position Controller

The Position Controller implements a P controller scheme to control the velocity of the drone platform. This P architecture can be seen below.

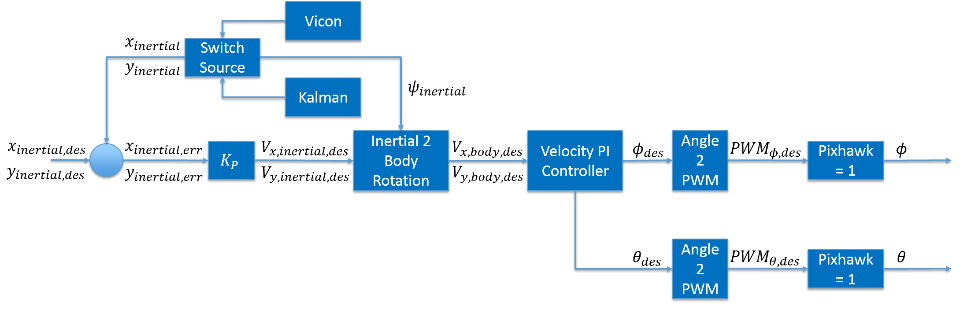


Figure ‑: Position P Controller

Inertial position measurements are input to the controller from either the Vicon motion capture system or the output from the Kalman Filter. The desired inertial x and y positions and current inertial x and y positions are input to the proportional gain term which scales the inertial x and y position error to calculate the desired inertial x and y velocities needed to achieve the desired inertial x and y positions. The calculated inertial x and y velocities are then sent to the Velocity Controller along with the current inertial yaw measurement such that the Velocity Controller can calculate the needed x and y body velocities to achieve the desired inertial x and y position. It should be noted that the current implementation of the position controller takes in a desired inertial x and y position but discretizes the space from the drone’s current inertial x and y position to the desired location. This effectively prevents a “step” input to the position controller which, throughout testing, was found to produce unwanted behavior. The discretization takes in a desired position-step size, discretizes the distance from the current and desired position and sends the discrete steps to the Position Controller. This implementation can be seen in the “stabilize\*.py” controller script for both Wolverine and Magneto in the “DVZ/Software Element\Mobility Subsystem\Controller Scripts\” and then corresponding Wolverine and Magneto folders.

#### Path Planning

Path planning is beyond the scope of the DVZ design and implementation work carried out during the Fall 2014 and Spring 2015 semesters, however, some thought has been given to the path planning implementation. Current (Spring 2015) conclusions regarding path planning design are as follows:

* Path planning must not only account for physical nodes, but will most likely also need to account for search nodes to ensure that a space larger than the camera field of view can be searched. There are several implementations which could prevent or mitigate the need for search nodes, but search nodes should be considered.
* Path planning for multiple robots should be considered during the early stages of path planning design to ensure that the chosen algorithm or methodology is capable of handling multiple robots. A suggested approach is to treat the n robots with k degrees of freedom as a single system with n\*k degrees of freedom and implement a single planning algorithm, however, there are other implementations which may prove more efficient.
* The marking of unexpected obstacles in the environment is likely best implemented via heavily weighting search nodes and adjusting the path plan based on the modified weights.
* Due to the heavy dependence on ROS, the path planning implementation should ideally be implemented via a package which easily interfaces with ROS. The leading choice appears to be the Open Motion Planning Library which provides a large variety of path planning algorithms and modeling capabilities and which is designed to interface with ROS.

## Hardware Element (HWE)

### Element Overview

The Hardware Element of this project involves ensuring functionality of the overall system. This includes selecting components to be used on the drone, making sure all components can be mounted safely and securely, making sure all components interface with each other and the main power system on the drone, repairing or replacing broken parts, and any other tasks related to keeping the drone in a functional state.

The major tasks for Spring 2015 semester included choosing a 2nd base platform (the drone), updating the mass and power budget for the drone and components, improving and making more blade guards, and upgrading to a 4S battery. The base platform selected was the 2013 3DR RTF X8, an 8 rotor copter. Other components are discussed in the following sections of this report. The new platform selected was the 2014 3DR RTF X8+. Testing has shown that the drone has an endurance of over ten minutes (14 minutes during flight with Vicon control input). Blade guards were strengthened with metal splints and foam blocks. Other parts made and component choices will be discussed in detail below.

### Hardware List

Here is a detailed list of all the hardware components used on the drone along with information about why that piece of hardware was selected. More specifications for each component are documented in the following location: Dropbox > DVZ > Hardware Element > Mass\_Power\_Budget.xlsx. Some of the other hardware options and more specs are listed in the following location: Dropbox > DVZ > Hardware Element > Hardware Specifications.xlsx.

1. Main Platform: 2013 3DR RTF X8 [1] and 2014 X8+
   1. Component Explanation

The main platform is the frame of the drone and the Power Distribution Board (PDB) and wiring to ESCs. The RTF (ready to fly) X8 comes with motors, props, ESCs and most other hardware necessary for manual flight. This hardware will be discussed below.

* 1. Other Options Considered

3DR RTF quad, 3DR RTF Y6, and the DJI Flamewheel F450

* 1. Reason Chosen

The X8 was chosen mostly because it was already available from a prior CU project. When compared to the other options, it also was expected to provide the most payload capacity, which would allow for more sensors to be carried on-board if we wanted to do that. The choice to use the X8 was a good one as the X8 has proven to be a good platform for testing our system and mounting various components.

* 1. Modifications

The RTF X8 initially had an Ardupilot autopilot system, which was switched out for the Pixhawk autopilot system. Two mounting plates were also made to mount the Hokuyo laser scanner and the PX4Flow sensor. These plates are both made by drilling holes in the proper locations into clear plastic plates. The material was chosen purely based on availability (out of scrap piles at the aero machine shop and the ITLL laser cutting room). Rigid composite sheets would be less likely to break during a crash, but are not necessary under normal operating conditions.

1. Battery: Traxxas 3S2P 8400mAh LiPo, 25C
   1. Component Explanation

Provides power for the whole flight system (motors, autopilot, sensors, etc). The name of the battery indicates its specs. This battery is a 3 cell (3S), which means it has three LiPo cells attached in series. It includes 2 sets of 3 cell LiPo batteries attached in parallel (2P) inside the packaging to hold a total capacity of 8400 mAh. The battery is rated to be able to constantly discharge at 25C. This means it can provide power at 25 times its capacity in amps, or 25\*8.4 = 210 amps.

* 1. Other Options Considered

Four cell (4S) batteries were also considered for this project. James Mack, a project advisor, suggested using 4S batteries to achieve longer flight times. Two 4S batteries owned by RECUV are also occasionally used for testing purposes and the system has functioned properly while using them, but the 3S batteries are used primarily. During the semester there have been 2 ESC failures, and both times were while using the 4S battery, though all the components are rated to work with 4S batteries. It has not been determined whether the relationship between battery use and ESC fault is a coincidence or not.

* 1. Reason Chosen

The 3S batteries were available, and proved to function properly as well as meet our requirement of providing enough power for a flight time over 10 minutes. If new batteries need to be purchased, a 4S battery is recommended, assuming the current 4S battery available has been proven to not harm the drone. The method used to select batteries was to test various configurations using eCalc, explained more in Section 4.2.7 [2].

* 1. **Modifications:**

N/A

1. Power Module: (3DR)
   1. Component Explanation

Takes power from battery and sends appropriate power to main Pixhawk power plug.

* 1. Other Options Considered

None.

* 1. Reason Chosen

It was standard on X8 and verified to work with Pixhawk as well, since it was used to power an Ardupilot originally.

* 1. Modifications

None

1. Autopilot: Pixhawk, with APM Arducopter firmware
   1. Component Explanation

An open source platform used to perform low-level control of the drone and includes some sensors. Look online for full documentation [3].

* 1. Other Options Considered

Ardupilot hardware with APM firmware or Pixhawk hardware with PX4 firmware.

* 1. Reason Chosen

The Pixhawk hardware with APM firmware was chosen because the Pixhawk provides more processing ability to interface with other tasks and interfaces more easily with other components. The APM Arducopter firmware was chosen because it is more robust and well tested. This setup choice is becoming more common among open source drone users and is now the standard package included with the 2014 3DR RTF X8, which helped to verify our decision.

* 1. Modifications

Typically, the Pixhawk is meant to be powered through a power port on the top of the device, with backup power through one set of the “Main Out” pins 1 – 8. In the current setup, power is instead supplied through the micro USB input, coming from our USB hub. This may be the source of some problems, but is still being worked out. [4]

1. ESC: 20 A
   1. Component Explanation

Electronic Speed Controllers rated to 20 Amps. There is one for each motor, and each is used to control the power sent to a motor based on a signal input from the Pixhawk.

* 1. Other Options Considered

None

* 1. Reason Chosen

Standard component on X8

* 1. Modifications

Two have been replaced. Reports exist in the Dropbox > DVZ > Hardware Element > Problems folder.

1. Motor: 850 kv brushless
   1. Component Explanation

There is one motor to turn each propeller, 8 total.

* 1. Other Options Considered

None

* 1. Reason Chosen

Standard on X8

* 1. Modifications

None. One has been replaced; see report in Dropbox > DVZ > Hardware Element > Problems folder.

1. Propeller: APC Slow Fly 10 X 4.7
   1. Component Explanation

10 inch diameter and 4.7 pitch

* 1. Other Options Considered

None

* 1. Reason Chosen

Standard on X8.

* 1. Modifications

None

1. Radio Receiver: FrSky TRF4
   1. Component Explanation

Receives radio signals from RC controller and sends signal to Pixhawk

* 1. Other Options Considered

None

* 1. Reason Chosen

This replaced the Futaba R617FS receiver, which worked with the Ardupilot, but did not interface directly with the Pixhawk. This is because it was a PWM receiver and the Pixhawk works with PPM receivers. The FrSky TRF4 is a PPM receiver and was recommended by James Mack. It has been proven to function sucessfully.

* 1. Modifications

None

1. Telemetry Transceiver: 3DR 915 MHz
   1. Component Explanation

Communicates wirelessly with Mission planner, or APMplanner ground stations. This piece of hardware may not be necessary for this project. In the scope of this project, communication with the drone is established through the wifi connecting to the on-board supervisory computer (Odroid). Since wireless communication is not used for connecting with mission planning software mentioned above, this component can be removed.

* 1. Other Options Considered

None

* 1. Reason Chosen

Standard with X8.

* 1. Modifications

None.

1. GPS: 3DR GPS+MAG (ublox LEA-6H)
   1. Component Explanation

Provides GPS coordinates and magnetometer input to Pixhawk. The GPS part is not used in this project because we are flying indoors, but the magnetometer provides more accurate heading indication.

* 1. Other Options Considered

None.

* 1. Reason Chosen

Standard on X8.

* 1. Modifications

None

1. Supervisory Computer: O-Droid U3
   1. Component Explanation

This is an ARM Processor used for on-board computing and processing and communicating with the ground station laptop. The case is useful for mounting but not necessary to function.

* 1. Other Options Considered

Something with an Intel processor instead of ARM.

* 1. **Reason Chosen:**

Used by others in RECUV. Shown to work for similar projects.

* 1. Modifications

Power cable was cut and soldered to the output of the BEC (discussed below) so that it could be powered by the battery instead of from the wall.

1. Sensor 1, Laser Scanner: Hokuyo URG-04LX-UG01
   1. Component Explanation

2D laser scanner to provide distance measurements from the drone to its surroundings. Range of 4m and scans over a 240 degree field of view, centered in the front.

* 1. Other Options Considered

Kinect 2 for Windows, ASUS Xtion Pro Live, Hokuyo UTM-30LX, Other Hokuyo models.

* 1. Reason Chosen
* Seen a lot in papers; other researchers using it for SLAM
* Able to simulate in Gazebo if desired
* Low cost (compared to other Hokuyo models)
* Low range was a concern, but we believed that it would be sufficient for use indoors. Testing to this point has shown that it suffices. Needs to be tested to see if it works for localizing in ECEA building.
* Low computational cost of data processing because it only takes one measurement (compared to Kinect or Xtion, which require lots of processing)
  1. Modifications

None

1. Sensor 2, Optical Flow and Altitude: PX4Flow
   1. Component Explanation

Includes a visual camera and a sonar range sensor mounted to a board which does some signal processing. Returns optical flow (position and velocity) using camera data, and altitude using sonar data).

* 1. Other Options Considered

Sonar sensors alone (HRLV module) and not using optical flow at all.

* 1. Reason Chosen

Using an optical flow sensor was expected to make localization a simpler task. There are examples of this sensor providing accurate positioning when used on quads. It is designed to integrate easily with the Pixhawk. It is convenient because it integrates optical flow with the sonar sensor onto one board and does some signal processing for us as well.

* 1. Modifications

None.

1. Visual Camera: TBD
   1. Component Explanation

A camera has not been chosen for certain. The camera is going to be used for taking pictures to identify zombies, etc, in the space.

* 1. Other Options Considered
* SONY Super Had CCD II TVL D-WDR DNR
* WGE-100 camera
* Raspberry Pi with Camera Module

1. Blade Guards
   1. Component Explanation

The implemented blade guards were made out of carbon rods that were already available from a past CU project. This proved to be a very simple, easily manufactured, low weight, and low cost solution.

* 1. Other Options Considered
* Carbon fiber rigid plates offset with plastic standoffs – too costly and heavy
* Foam guards – weak and still heavier than carbon rods
* Plastic molded blade guards (similar to James’s design for Flamewheel) – not as strong or light
  1. Reason Chosen

Carbon rods were the lightest and simplest design.

* 1. Modifications

Blade guards are attached to the quad by inserting a rod into the end of each arm. Place the screw at the end of the arm (holding the motor plates on) through a hole in the carbon rod to prevent it from spinning or sliding in and out. Care must be taken to ensure that the carbon rod does not damage any motor wires as it is inserted.

1. BEC: HKU5 UBEC
   1. Component Explanation

Splits power from the battery to supply 5V to the Odroid and the USB hub

* 1. Other Options Considered

None.

* 1. Reason Chosen

Availability and functionality. Lightweight.

* 1. Modifications

Soldered into battery leads.

1. USB Hub
   1. Component Explanation

Powered USB hub with 4 usb ports and 1 mini usb port.

* 1. Other Options Considered

Non-powered USB hub.

* 1. Reason Chosen

Needed a powered usb hub with enough ports for all components.

* 1. Modifications

Power source cable modified and soldered to output of BEC.

### Design and Analysis Tools and Resources

In designing this drone, analysis was done to estimate flight time (endurance) and power draw. An online tool for RC modeling and analysis called eCalc [2] is very useful for doing analysis of a quod copter. It allows a user to enter in the main quad components and specifications, and then outputs useful data such as endurance, power drawn by the motors, etc. eCalc was used in this manner to estimate endurance. Figure 4‑17 below shows results from this analysis. The expected endurance was about 12 minutes for a hover flight, which was proven to be a conservative estimate when a manual hover flight lasting 13 minutes was performed.

The eCalc website was also useful for comparing batteries. Different battery configurations could be loaded and then analyzed with the drone system to see how endurance varied. It was shown that the current chosen batteries (listed above) are a good choice with this drone, but certain 4 cell LiPo batteries may produce slightly longer flights.

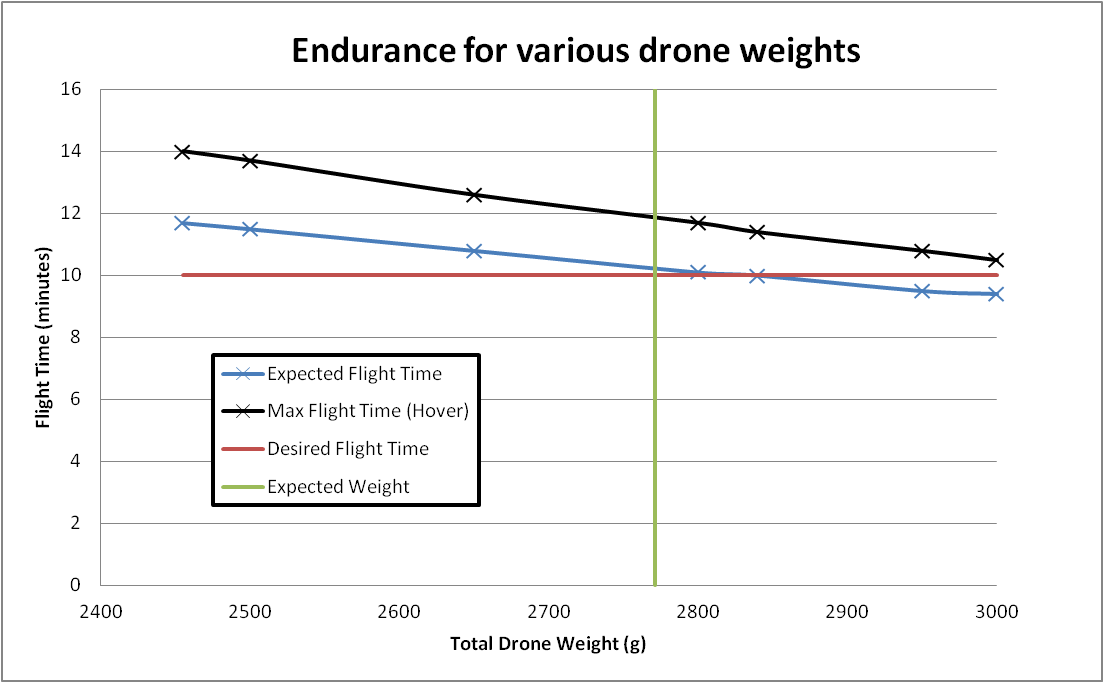


Figure ‑: Flight time analysis results from eCalc

# Verification and Validation

## DVZ Test Philosophy

The structure of the DVZ project did not lead to having a typical test philosophy. The basis for this project was to acquire and customize algorithms and methods that already existed to implement autonomous flight. Due to this fact a lot of focus was put on finding different methods and testing them with the utmost efficiency. This approach allowed us to learn about the differences between each method through physical implementation rather than through an in depth study of the approach. While this dictated how much of the initial research phase was performed this did not take away from more traditional verification and validation techniques. The requirements were broken down into three main levels for customer, element and subsystem necessities. The final validation test which involves playing the game of drones versus zombies is outlined clearly in the CONOPS and drove most of the customer requirements.

## Test Scenarios

The baseline test plans are broken down into three main groups for non-flight, manned flight, and unmanned flight scenarios. The non-flight tests include pushing the octorotor around on the cart to perform sensor characterization tests. The bulk of initial testing was done in this manner for both safety and efficiency sake. Based on changes to the Kalman filter implementation system verification was switched from non-flight to manned flight cases. This was due to the use of the Euler angles as a dead reckoning sensor measurement instead of the accelerometers. For unmanned flight this is broken down further into cases using either VICON or a combination of the Kalman filter and AMCL. The VICON system served as the main verification technique because it could track the octorotor’s position in 3D space. This was used rigorously in the tuning of controller gains as well as characterizing the sensor fusion techniques used. The second case involved the system functioning autonomously using only on board sensors. While the fully autonomous tests were taking place the VICON system was tracking the octorotor and storing its position for system comparisons to take place. The bulk of the testing was performed in the RECUV flying lab in the VICON space but due to the location of the final demonstration being set in the engineering center the system had to be tested in that environment. Many of those tests were performed with the cart for safety considerations but as the system progressed manned flight took place as well as one unmanned flight.

## Verification Results

This section serves to show the progress that has been made towards verifying the requirements. The figures below quantify the level of success that was achieved as well as outline improvements that need to be made moving forward.

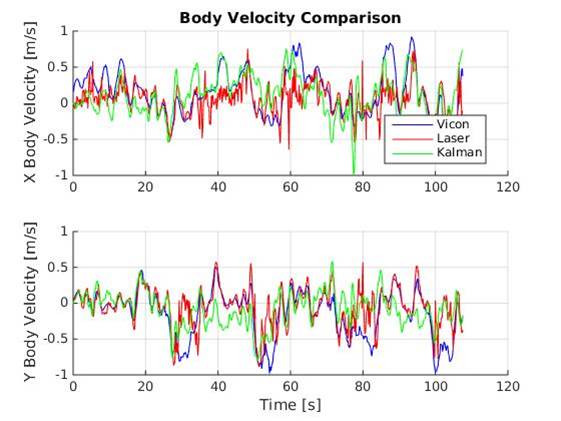


Figure ‑: Body Velocity Comparison

In Figure 5‑1 data from two methods is compared to the VICON tracker. The red lines on the plots show how the laser scanmatcher tracked the velocity changes as the octorotor was flown in a unique space. It is known that the Lockheed Martin Hallway is not a fully unique space so scanmatcher will not be able to provide accurate reading the entire time. This is one part that makes up the Kalman filter fusion which means that the Kalman filter should be able to achieve equal or better performance than scanmatcher on its own. As can be seen from the graphs above this does not hold true in all cases. At some points the Kalman filter is much closer to the VICON data such as around 40 seconds on the X body plot. This is an issue that will have to be remedied moving forward in the project design.

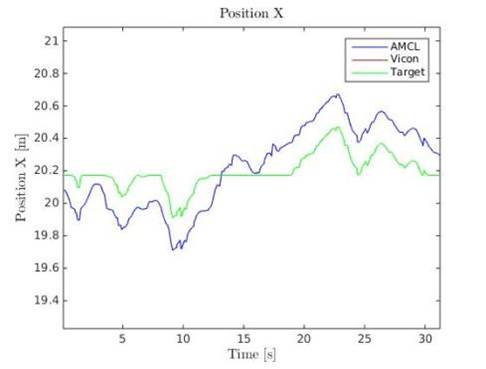


Figure ‑: X Hallway Position Hold

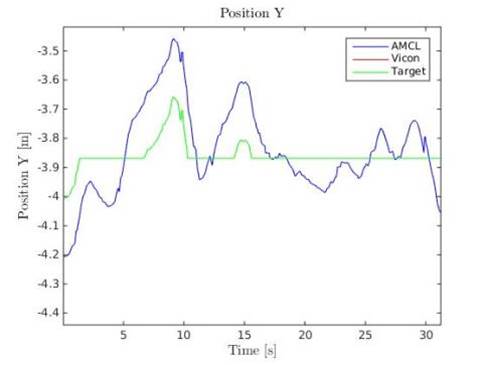


Figure ‑: Y Hallway Position Hold

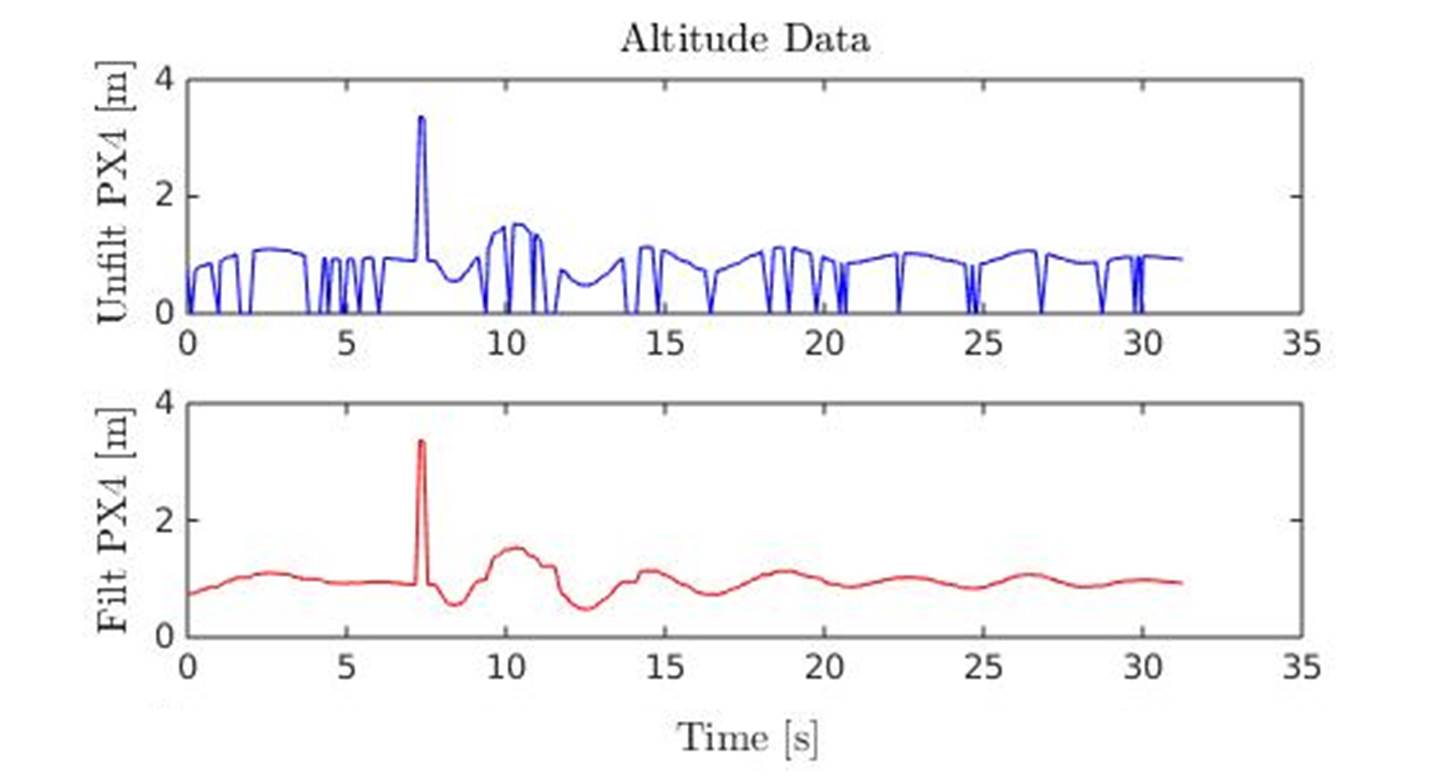


Figure ‑: Altitude Hallway Hold

The most important requirement centers on being able to fly autonomously in the hallway with only onboard sensors. This was broken down into smaller goals such as altitude hold and way point following in a unique portion of the hallway. Figure 5‑4 shows the data from the PX4 Flow ultrasonic sensor as the octorotor was performing am altitude hold in the hallway. The top plot shows the unfiltered data which as can be seen has many dropouts which needed to be filtered out so that the altitude controller could perform properly. In its current state the filter only deals with zero dropouts which is why the spike at around 7 seconds still shows up in the filtered plot. Due to the success of the altitude hold the system was tested next with a position hold as well. The X and Y position data from this test can be seen in Figure 5‑2 and Figure 5‑3. The current implementation dictates that the controller only produces a limited step response as to limit any possible large jump in the system. This was implemented to improve the safety factor for autonomous flight.

The following tables outline the progress that has been achieved on the customer requirements. The items marked in green show requirements that were met in their full capacity. Yellow dictates requirements that were met in a partial capacity and finally red outlines requirements that either were decided to be out of scope for this first phase of the project or were not achieved due to a lack of system robustness.

Table ‑: Requirements Verification Part 1

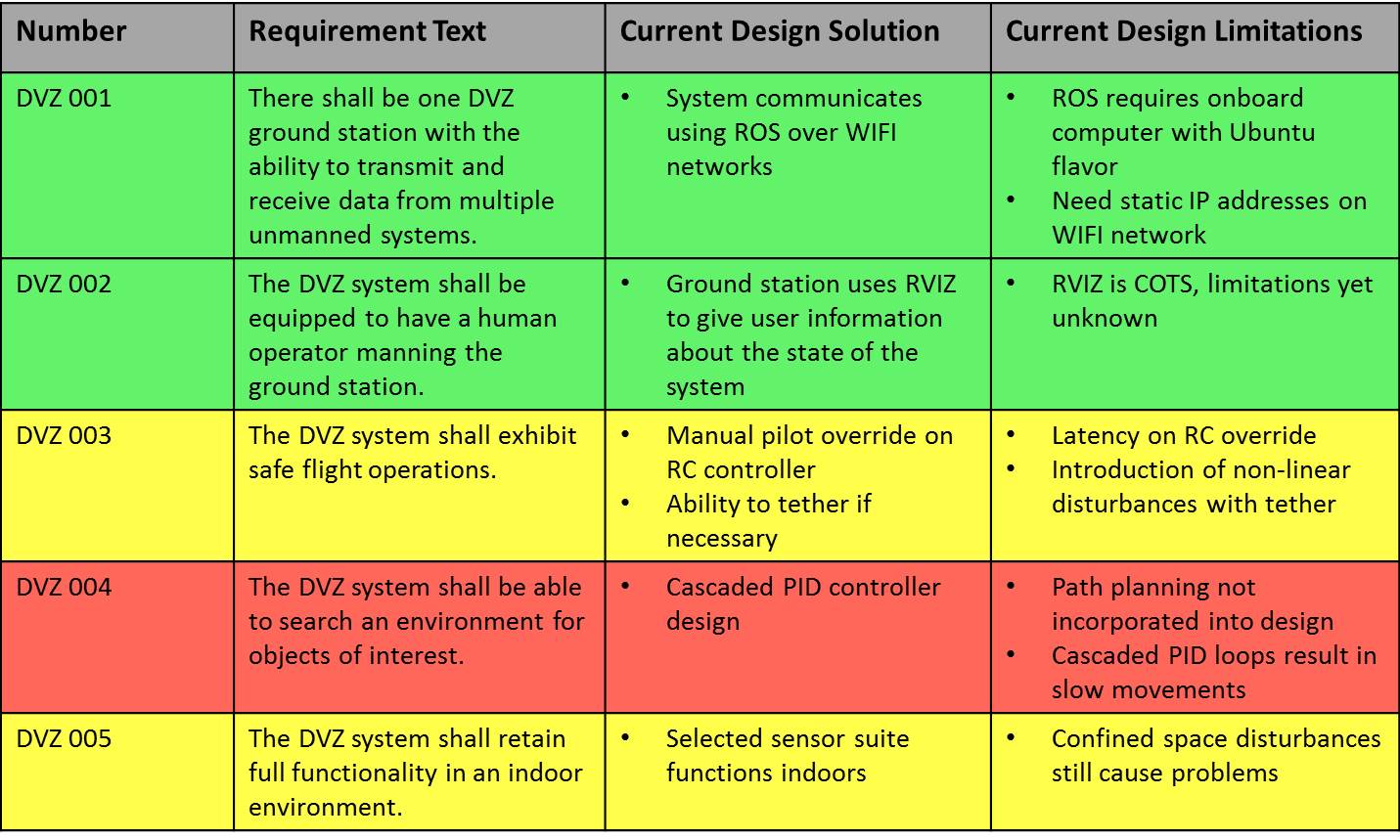
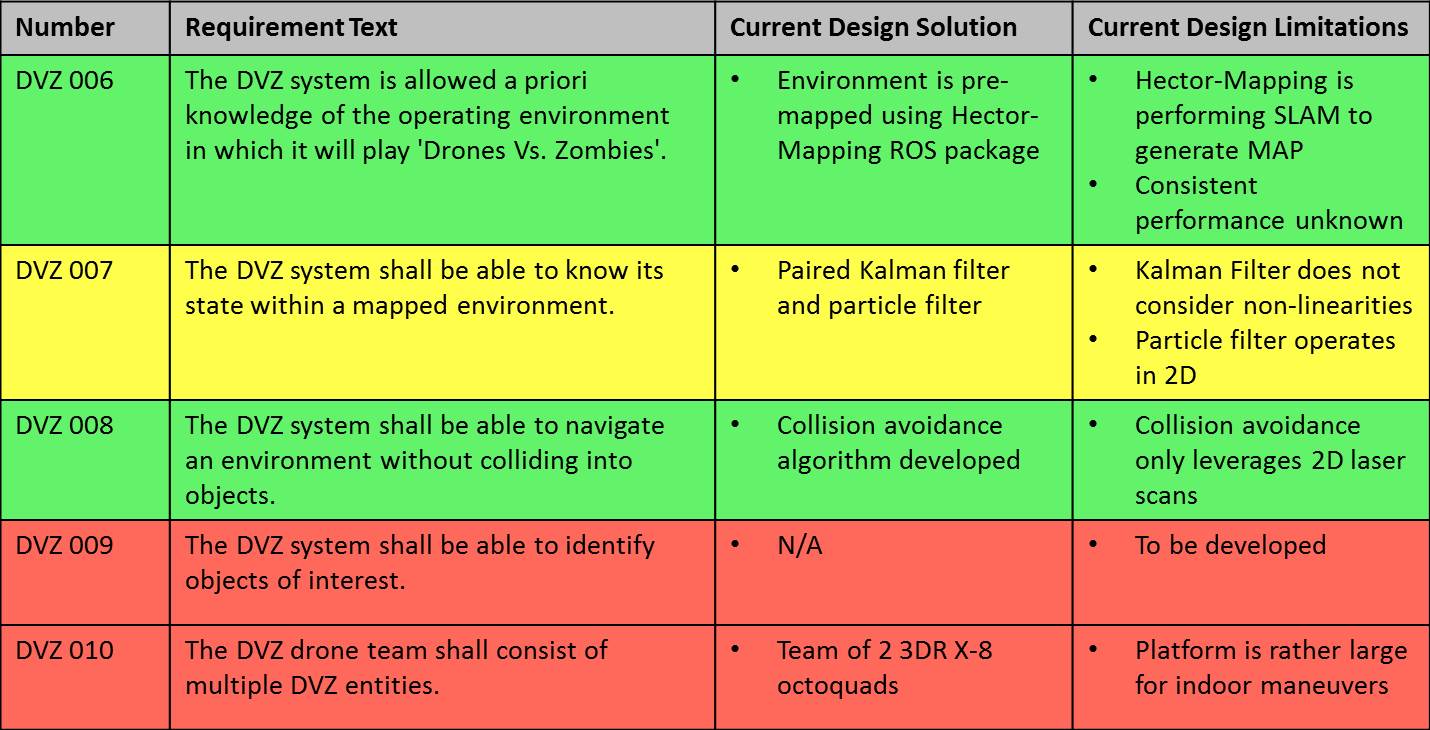


Table ‑: Requirements Verification Part 2



From Figure 5‑1 and Table 5‑2 it can be seen that about half of the customer requirements were met while the remaining portion were either partially completed or not attempted. For requirement DVZ\_003 the system was able to exhibit safe flight operations in some but not all cases. This is one area where the robustness of the system was not there to allow for these requirements to be tested to its fully potential and verified. Requirement DVZ\_005 relates back to Figure 5‑2, Figure 5‑3, and Figure 5‑4 where the results for position holding in the hallway was shown. At this stage the indoor functionality is in a preliminary stage where options are limited. This correlates directly with DVZ\_007 which looks at the element of localizing in the hallway. Finally of the three remaining requirements were determined to be out of scope for this first phase. In depth path planning and object identification was determined to be secondary to the ability to localize and fly autonomously. Due to the partial success that was achieved on this front these items were not addressed and therefore verification results at this stage do not exist.

# Project Management

## Work Breakdown Structure (WBS) by Phase

### Phase 1

Tasking necessary to complete the 2014-2015 scope for the DVZ project will be broken into two phases nominally completed per semester.

DVZ Phase 1 was predominately completed during the Fall 2014 semester and included all work necessary to generate a solution for position estimation and localization. DVZ Phase 1 also included the initial design work and implementation necessary for the mobility and communication portions of the problem along with all necessary HW work to support SW and sensor integration. As the solution was developed, documentation and organizational work products will be generated to provide continuity documentation and description of the baseline. The high-level work products that comprised DVZ work for the Fall 2014 semester are shown in Figure 4‑1 where specific sets of tasking (e.g. PM) correspond to the roles defined in Figure 3‑1.

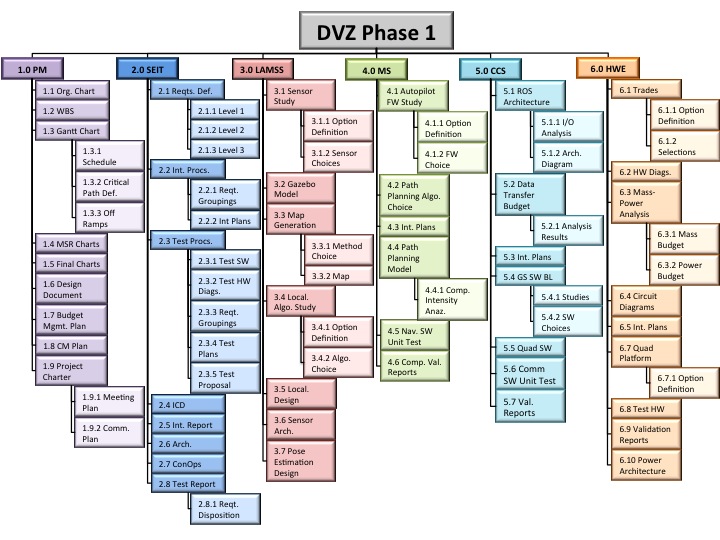


Figure ‑: DVZ Phase 1 Work Breakdown Structure

### Phase 2

DVZ Phase 2 was predominately completed during the Spring 2015 Semester and included all work necessary to finalize and implement the localization and mobility solution determined during Phase 1. Work for DVZ phase 2 centered on design and implementation of a Kalman Filter to allow for sensor fusion along with the combination of the Kalman Filter with AMCL and laser odometry to provide a robust localization solution. In addition to a solution to the Localization problem, DVZ Phase 2 included final implementation of the control algorithms and integration of the localization and controls portions of the DVZ architecture. While technical work was ongoing for Phase 2, documentation and support activities were ongoing to ensure that the system hardware functionality was maintained and that the system design was thoroughly documented. The high level work products that comprised DVZ for the Spring 2015 semester are summarized in Figure 6‑2 below.

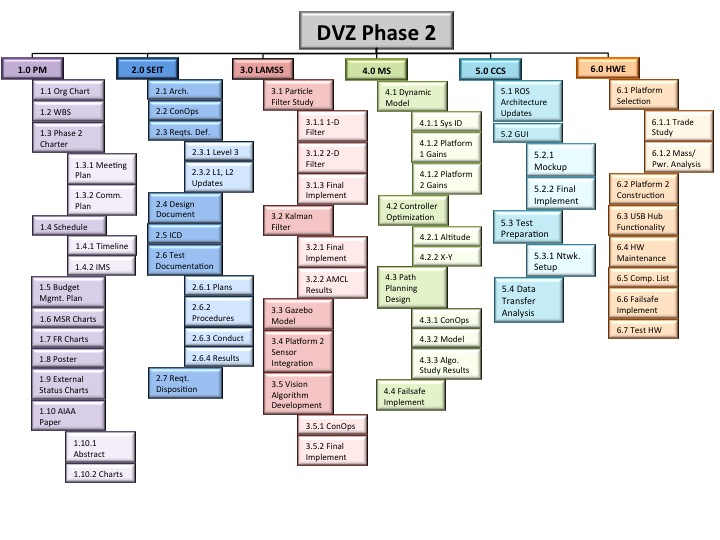


Figure ‑: DVZ Phase 2 Work Breakdown Structure

## Project Timeline

### Phase 1

Tasking comprising the DVZ Phase 1 WBS was be carried out based on the timeline depicted in Figure 5‑1 and work was be tailored to the following two major milestones:

* Requirements Signoff: Customer Acceptance of DVZ Requirements to Level 2 (Oct 24, 2014)
* Phase 1 Critical Design Review (CDR): Detailed review of Phase 1 design by applicable stakeholders (Dec 19, 2014)

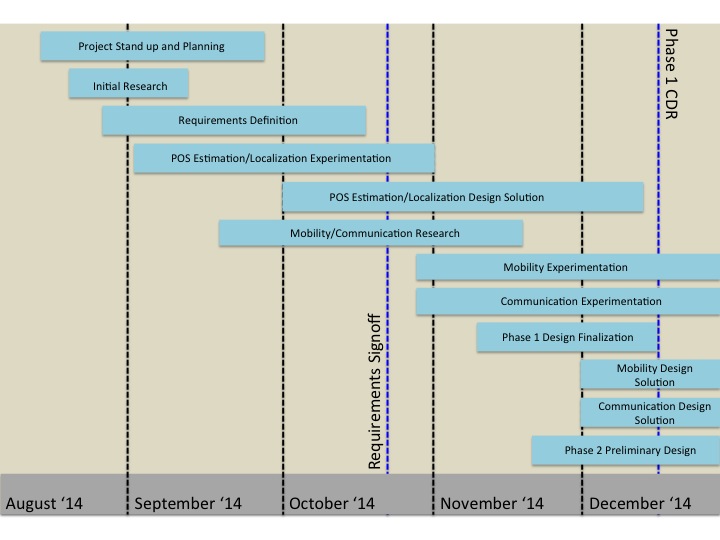


Figure ‑: Phase 1 DVZ Timeline

### Phase 2

Tasking comprising the DVZ Phase 2 WBS was be carried out based on the timeline depicted in Figure 6‑4 and work was be tailored to the following two major milestones:

* Midterm Status Review (MSR): Status of DVZ design and implementation at mid semester (Mar 6, 2015)
* Phase 2 Spring Final Status Review (SFSR): Detailed review of Phase 2 progress at the end of Semester 2 (May 1, 2015)

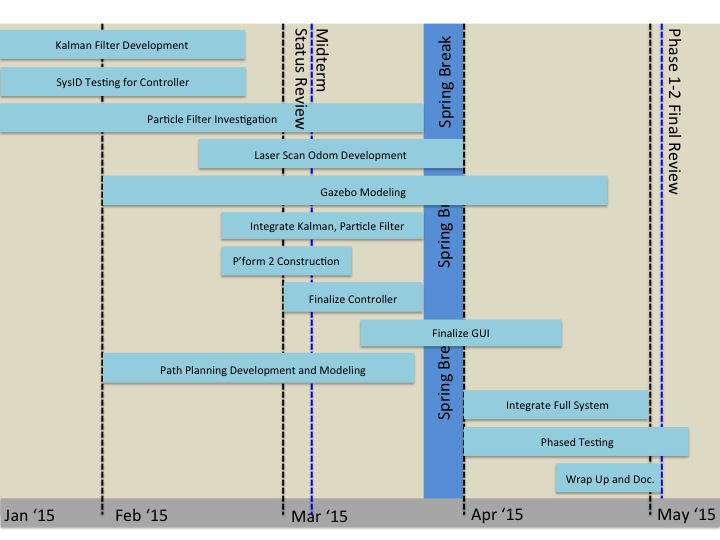


Figure ‑: Phase 2 DVZ Timeline

## Project Budget

### 2014-2015

The DVZ project was allocated $10,000.00 for the 2014-2015 academic year and the budget was underrun due to re-use of existing hardware. Budget not used during the 2014-2015 academic year was returned to the customer and may be returned to the team as additional funding for the 2015-2016 academic year. A summary of DVZ spending for the 2014-2015 academic year is provided in Figure 6‑5, and a detailed cost breakout is provided in Appendix A.

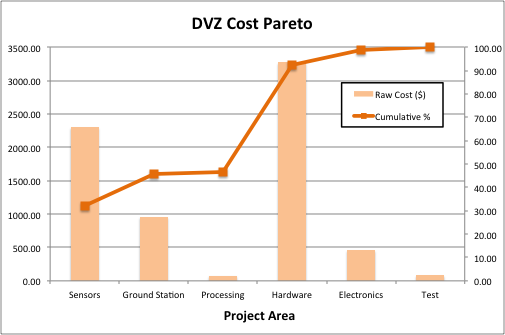


Figure ‑: DVZ Expenditures for 2014-2015 Academic Year

## Additional Documentation

In addition to this document, details regarding the detailed design for DVZ and the DVZ project as a whole can be found at the following locations:

* DVZ-CM-001 DVZ Document Listing
* DVZ/Project Management/Configuration Management
* DVZ-PM-009 DVZ Project Charter:
* DVZ/Project Management/Configuration Management/Documents/DVZ-PM-001 DVZ Project Charter
* DVZ-PM-007 DVZ System Requirements Review (SRR) Charts:
* DVZ/Project Management/Configuration Management/Formal Presentations/DVZ-PM-007 DVZ Phase 1 SRR Charts
* DVZ-PM-008 DVZ Critical Design Review (CDR) Charts:
* DVZ/Project Management/Configuration Management/Formal Presentations/DVZ-PM-008 DVZ Phase 1 CDR Charts
* DVZ-PM-010 DVZ Midterm Status Review (MSR) Charts:
* DVZ/Project Management/Configuration Management/Formal Presentations/DVZ-PM-010 DVZ Phase 2 MSR Charts
* DVZ-PM-011 DVZ Spring Final Status Review Charts:
* DVZ/Project Management/Configuration Management/Formal Presentations/DVZ-PM-011 DVZ Phase 2 SFSR Charts

# Appendix A: DVZ Expenditure Summary

The following table includes all expenditures incurred by the DVZ project during the 2014-2015 academic year and may not represent the most current expenditure summary. Current expenditures are detailed in the expenditure report at DVZ/Project Management/Financial/Expenses.

Table ‑: DVZ Expenditures

| Item | Description | Part Number | Vendor | Unit Cost | Quantity | Shipping | Tax | Total Cost | Group |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PX4Flow | PX4 optical flow smart camera | PX4-KIT-0009 | 3DRobotics | $149.99 | 1 | $8.99 | N/A | $158.98 | Electronics |
| Pixhawk | Pixhawk autopilot with u-blox GPS with compass unit | PX4-KIT-0011 | 3DRobotics | $199.99 | 1 | $0.00 | N/A | $199.99 | Hardware |
| 3DR uBlox GPS with Compass Kit | 3DR uBlox GPS with Compass Kit | GPS-KIT-0003 | 3DRobotics | $79.99 | 1 | $0.00 | N/A | $79.99 | Hardware |
| 5V/2A Power Supply | 5V/2A Power Supply for ODroid | N/A | Hard Kernel | $5.50 | 1 | $30.00 | N/A | $35.50 | Hardware |
| HDMI Cable (Micro, Type D) | HDMI to Micro HDMI cable | N/A | Hard Kernel | $5.70 | 1 | $0.00 | N/A | $5.70 | Hardware |
| WiFi Module 3 | WiFi module dongle | N/A | Hard Kernel | $4.00 | 2 | $0.00 | N/A | $8.00 | Hardware |
| ODROID-U3 | ODroid single board computer | N/A | Hard Kernel | $65.00 | 1 | $0.00 | N/A | $65.00 | Hardware |
| 16 GB eMMC Module U Linux | Linux module for ODroid | N/A | Hard Kernel | $39.00 | 1 | $0.00 | N/A | $39.00 | Processing |
| FrSky TFR4 2.4GHz Receiver | FrSky TFR4 4ch 2.4GHz Surface/Air Receiver FASST Compatible | TFR4 | Hobby King | $29.47 | 2 | $8.31 | N/A | $67.25 | Hardware |
| Turnigy 3000mAh 4S Battery | Turnigy nano-tech 3000mah 4S 25-50C Lipo Pack | N3000.4S.25 | Hobby King | $27.92 | 2 | $0.00 | N/A | $55.84 | Hardware |
| Lenovo Y40 Laptop | Lenovo Y40 Laptop: Black | 59423032 | Lenovo | $959.00 | 1 | $0.00 | $80.17 | $959.00 | Ground Station |
| Hokuyo Laser Range Finder | Hokuyo URG-04LX-UG01 Scanning Laser Rangefinder | RB-Hok-07 | RobotShop | $1,140.00 | 1 | $0.00 | $0.00 | $1,140.00 | Sensing |
| MicroSD Card | 16 GB MicroSD Ultra Plus | N/A | Best Buy | $17.99 | 1 | $0.00 | $0.00 | $17.99 | Hardware |
| Male-Female Hex Standoff | Metric Nylon 6/6 Male-Female Thread Hex Standoff, 6mm Hex, 31mm Length, M3 Screw Size | 95783A463 | McMaster-Carr | $2.06 | 1 | $5.54 | $0.00 | $7.60 | Hardware |
| Phillips Machine Screw | Metric Nylon Pan Head Phillips Machine Screw, Black, M3 Size, 6mm Length, 0.5mm Pitch, packs of 100 | 92492A116 | McMaster-Carr | $7.36 | 1 | $0.00 | $0.00 | $7.36 | Hardware |
| Phillips Machine Screw Zinc-Plated | Metric Pan Head Phillips Machine Screw, Zinc-Plated Steel, M3 Size, 16mm Length, 0.5mm Pitch, packs of 100 | 92005A126 | McMaster-Carr | $2.88 | 1 | $0.00 | $0.00 | $2.88 | Hardware |
| Thin Hex Nut | Class 04 Steel Thin Hex Nut - DIN 439B, Zinc Plated, M3x0.5 Thread Size, 5.5mm Wide, 1.8mm High, packs of 100 | 90695A033 | McMaster-Carr | $3.10 | 1 | $0.00 | $0.00 | $3.10 | Hardware |
| 5V/2A Power Supply | 5V/2A Power Supply for ODroid | N/A | Hard Kernel | $5.50 | 1 | $30.00 | $0.00 | $35.50 | Hardware |
| HDMI Cable (Micro, Type D) | HDMI to Micro HDMI cable | N/A | Hard Kernel | $5.70 | 1 | $0.00 | $0.00 | $5.70 | Hardware |
| WiFi Module 3 | WiFi module dongle | N/A | Hard Kernel | $8.00 | 1 | $0.00 | $0.00 | $8.00 | Hardware |
| ODROID-U3 | ODroid single board computer | N/A | Hard Kernel | $65.00 | 2 | $0.00 | $0.00 | $130.00 | Hardware |
| 16 GB eMMC Module U Linux | Linux module for ODroid | N/A | Hard Kernel | $39.00 | 1 | $0.00 | $0.00 | $39.00 | Processing |
| ODROID-U3 Case | Case for ODROID computer | N/A | Hard Kernel | $4.00 | 3 | $0.00 | $0.00 | $12.00 | Hardware |
| ESC 20 Amp with SimonK | Electronic Speed Controller 20 Amp | N/A | 3DRobotics | $25.99 | 2 | $2.32 | N/A | $54.30 | Hardware |
| PX4Flow | PX4 optical flow smart camera (NOT COMPATIBLE WITH ARDUPILOT FIRMWARE) | PX4-KIT-0009 | 3DRobotics | $149.99 | 1 | $2.32 | $0.00 | $152.31 | Electronics |
| Wall 2”x4” | 2”x4” boards for walls | N/A | Home Depot | $2.83 | 8 | $0.00 | $0.00 | $22.64 | Test |
| Wall 1/4” washers | 1/4” washers for walls | N/A | Home Depot | $8.98 | 1 | $0.00 | $0.00 | $8.98 | Test |
| Wall 1/4” hex nuts | 1/4” hex nuts for walls | N/A | Home Depot | $5.98 | 1 | $0.00 | $0.00 | $5.98 | Test |
| Wall 1/4”x5” hex bolts | 1/4”x5” hex bolts for walls | N/A | Home Depot | $0.40 | 20 | $0.00 | $0.00 | $8.00 | Test |
| Wall 15/32” 4'x8' boards | 15/32” 4'x8' boards for walls | N/A | Home Depot | $9.95 | 4 | $0.00 | $0.00 | $39.80 | Test |
| FTDI Cable | FTDI Cable 5V VCC-3.3V I/O | DEV-09717 | Sparkfun | $17.95 | 1 | $0.00 | $0.00 | $17.95 | Hardware |
| Zip Ties 200 pack | 8" Neutral/Black Cable Tie Set 200pk | N/A | Home Depot | $6.47 | 1 | $0.00 | $0.00 | $6.47 | Hardware |
| Painters Tape | Scotch Blue .94" Painters Tape | N/A | Home Depot | $3.30 | 1 | $0.00 | $0.00 | $3.30 | Hardware |
| Hokuyo Laser Range Finder | Hokuyo URG-04LX-UG01 Scanning Laser Rangefinder | RB-Hok-07 | RobotShop | $1,140.00 | 1 | $23.74 | $0.00 | $1,163.74 | Sensing |
| Traxxas 25C 11.1V 3S 8400mAh Lipo Battery | Traxxas 25C 11.1V 3S 8400mAh Lipo Battery | N/A | Amazon | $134.99 | 1 | $2.00 | $0.00 | $136.99 | Hardware |
| 700nm 12.5mm Diameter, OD 4 Shortpass Filter | 700nm 12.5mm Diameter, OD 4 Shortpass Filter | 84-701 | Edmund Optics | $160.00 | 1 | $17.99 | $0.00 | $177.99 | Hardware |
| X8+ Multirotor | X8+ Multirotor platform | N/A | 3DRobotics | $1,350.00 | 1 | $0.00 | $0.00 | $1,350.00 | Hardware |
| X8+ Battery Pack | X8+ Battery Pack | N/A | 3DRobotics | $149.99 | 2 | $0.00 | $0.00 | $299.98 | Hardware |
| X8+ Propellers | X8+ Propellers Push-Pull Set | N/A | 3DRobotics | $8.00 | 4 | $0.00 | $0.00 | $32.00 | Hardware |
| PX4Flow | PX4 optical flow smart camera | PX4-KIT-0009 | 3DRobotics | $149.99 | 1 | $0.00 | $0.00 | $149.99 | Electronics |
| Pack of AA batteries | AA Batteries needed for X8+ RC Handset | N/A | Best Buy | $12.49 | 1 | $9.00 | $0.00 | $21.49 | Hardware |
| Carbon Fiber Rods | Rigid Carbon Fiber Shapes, Round Tube, 0.315" OD, 0.237" ID, 48" Length | 2153T37 | McMaster-Carr | $18.40 | 4 | $6.66 | $0.00 | $80.26 | Hardware |
| USB 2.0 4-Port Hub | Plugable USB 2.0 4-Port High Speed Hub with 12.5W Power Adapter | N/A | Amazon | $15.95 | 1 | $0.00 | $0.00 | $15.95 | Hardware |
| FTDI Cable 5V VCC-3.3V I/O | FTDI Cable 5V VCC-3.3V I/O | DEV-09718 | Sparkfun | $17.95 | 1 | $4.42 | $0.00 | $22.37 | Hardware |
| Safety Glasses | 17 safety glasses | N/A | McGuckin Hardware | $69.23 | 1 | $0.00 | $0.00 | $69.23 | Hardware |
| Expo Pen Style Markers | Expo Low Odor Dry Erase Pen Style Markers, 12 Colored Markers | N/A | Amazon | $9.15 | 2 | $0.00 | $0.00 | $18.30 | Hardware |
| Expo Chisel Tip Markers | Expo 2 Low-Odor Dry Erase Marker Set, Chisel Tip, 16-Piece, Assorted Colors | N/A | Amazon | $12.39 | 2 | $0.00 | $0.00 | $24.78 | Hardware |
| Hokuyo Mounting Screws | 18-8 Stainless Steel Metric Pan Head Phillips Machine Screw, M3 Size, 8MM Length, 0.5MM Pitch, Pack of 100 | 92000A118 | McMaster-Carr | $4.66 | 1 | $0.00 | $0.00 | $4.66 | Hardware |
| Leg Standoff Mounting | 18-8 Stainless Stee Metric Pan Head Phillips Machine Screw, M3 Size, 5MM Length, 0.5MM Pitch, Pack of 100 | 92000A114 | McMaster-Carr | $6.72 | 1 | $0.00 | $0.00 | $6.72 | Hardware |
| Propeller Lock Washers | Type 18-8 Stainless Steel Internal-Tooth Lock Washer, Number 10 Screw Size, 0.195" ID, 0.381" OD, Pack of 100 | 91757A107 | McMaster-Carr | $3.63 | 1 | $0.00 | $0.00 | $3.63 | Hardware |
| PX4Flow Mounting Standoff | Metric Nylon 6/6 Male-Female Thread Hex Standoff, 6MM Hex, 5MM Length, M3 Screw Size | 95783A403 | McMaster-Carr | $1.32 | 12 | $0.00 | $0.00 | $15.84 | Hardware |
| PX4Flow Mounting Plate | Optically Clear Cast Acrylic Sheet, 3/32" Thick, 6" x 12" | 8560K188 | McMaster-Carr | $2.57 | 1 | $6.91 | $0.00 | $9.48 | Hardware |
| X8+ Mounting Screws | Metric Nylon 6/6 Male-Female Thread Hex Standoff, 4.5mm Hex, 10mm Length, M3 screw size | 95783A312 | McMaster-Carr | $1.47 | 10 | $6.11 | $0.00 | $20.81 | Hardware |
| Fedex Shipping to Repair January 2015 Hokuyo | Fedex Shipping to Sentek Solutions, Inc, 2019 Van Buren Ave, Suite A - Indian Trail, NC 28079 - Attn: Grant Keil/RMA 032715-1 | N/A | Fedex | $29.87 | 1 | $0.00 | $0.00 | $29.87 | Hardware |
| 10x4.7 SF Propeller for X8 | 10x4.7 SF Propeller | LP10047SF | APC Propellers | $3.06 | 8 | $0.00 | $0.00 | $24.48 | Hardware |
| 10x4.7 SFP Propeller for X8 | 10x4.7 SFP Propeller | LP10047SFP | APC Propellers | $3.06 | 8 | $12.16 | $0.00 | $36.64 | Hardware |
| Steel tube splint for blade guards | Easy-to-Weld 4130 Alloy Steel Round Tube, 0.375" OD, 0.028" Wall Thickness, 3' Long | 89955K469 | McMaster-Carr | $13.77 | 1 | $0.00 | $0.00 | $13.77 | Hardware |
| 1Ft Mini USB Cable | 1Ft USB 2.0 A to 5-Pin Mini B Cable | N/A | Amazon | $3.99 | 3 | $0.00 | -$0.60 | $11.37 | Hardware |
| (10-Pack) 1Ft Micro USB Cable | (10-Pack) 1Ft USB Cable 2.0 A Male to Micro B Cable | N/A | Amazon | $14.95 | 1 | $0.00 | $0.00 | $14.95 | Hardware |
| Nylon T-Connectors 10 Pairs | Nylon T-Connectors 10 Pairs | 30685 | Hobby King | $4.14 | 1 | $0.00 | $0.00 | $4.14 | Hardware |
| Nylon XT60 Connectors Male/Female 5 Pairs | Nylon XT60 Connectors Male/Female 5 Pairs | 52411 | Hobby King | $4.15 | 1 | $0.00 | $0.00 | $4.15 | Hardware |
| HKU5 5V/5A UBEC | HKU5 5V/5A UBEC | 33290 | Hobby King | $6.03 | 1 | $6.44 | $0.00 | $12.47 | Hardware |
| Plain-Backed Fiberglass Strip, 0.063" Thick, 1" Width, 10' Length | Plain-Backed Fiberglass Strip, 0.063" Thick, 1" Width, 10' Length | 8817K63 | McMaster-Carr | $2.60 | 1 | $0.00 | $0.00 | $2.60 | Hardware |
| Medium Grit Grinding Bit, 1/8" Shank, 1/8" Head Diameter, 3/8" Head Length, B97, Packs of 5 | Medium Grit Grinding Bit, 1/8" Shank, 1/8" Head Diameter, 3/8" Head Length, B97, Packs of 5 | 4522A176 | McMaster-Carr | $6.32 | 1 | $0.00 | $0.00 | $6.32 | Hardware |
| Double Cut Carbide Bur with 1/8" Shank Diameter, Cylindrical-Flat End, 1/8" Head Diameter | Double Cut Carbide Bur with 1/8" Shank Diameter, Cylindrical-Flat End, 1/8" Head Diameter | 8175A11 | McMaster-Carr | $5.50 | 1 | $0.00 | $0.00 | $5.50 | Hardware |
| J-B Weld Adhesive, 8280, 10 Ounce Tube | J-B Weld Adhesive, 8280, 10 Ounce Tube | 7605A12 | McMaster-Carr | $17.50 | 1 | $0.00 | $0.00 | $17.50 | Hardware |
| Plastic Cup, Polypropylene, 2-1/2 oz, 2-3/8" Top OD X 1-5/8" H, packs of 125 | Plastic Cup, Polypropylene, 2-1/2 oz, 2-3/8" Top OD X 1-5/8" H, packs of 125 | 1865T35 | McMaster-Carr | $7.87 | 1 | $0.00 | $0.00 | $7.87 | Hardware |
| Heat-Shrink Tubing, 0.19" ID Before, 0.09" ID After, 1 ft L, Black, packs of 10 | Heat-Shrink Tubing, 0.19" ID Before, 0.09" ID After, 1 ft L, Black, packs of 10 | 7486K42 | McMaster-Carr | $9.47 | 1 | $0.00 | $0.00 | $9.47 | Hardware |
| Heat-Shrink Tubing, 1/8" ID Before, 1/16" ID After 12" L, Black, packs of 10 | Heat-Shrink Tubing, 1/8" ID Before, 1/16" ID After 12" L, Black, packs of 10 | 7856K63 | McMaster-Carr | $7.97 | 1 | $0.00 | $0.00 | $7.97 | Hardware |
| Total |  |  |  |  |  |  |  | $7,236.39 |  |