**Graduate Projects**

University of Colorado at Boulder

Aerospace Engineering Sciences

ASEN 5018/6028 –Spring 2015

|  |
| --- |
| **Drones Versus Zombies (DVZ)**  **Mobility Subsystem (MS)**  **Summary/Continuity Document** |

|  |  |  |
| --- | --- | --- |
| **Approvers List** | | |
|  | Title | Name |
| Prepared By | MS Lead | Mark Sakaguchi |
|  | MS Team Member | Drew Ellison |
|  |  |  |
| Approved By | MS Lead | Mark Sakaguchi |
| Approved By | Systems Engineer | Edward Scott |
| Approved By | Project Manager | Garrett Brown |

# Introduction & Summary

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.

The DVZ team has selected a 3DRobotics X8 quadcopter platform with a Pixhawk autopilot, a 3DRobotics PX4Flow optical flow sensor, an Odroid U3 single board computer running Lubuntu 14.04, and the Robot Operating System Indigo release to handle communication between the different hardware pieces.

# Semester Report

## Objectives and Tasks List

Completed:

1. Developed functional altitude controller that used either the Vicon system or the 3DRobotics PX4Flow optical sensor’s ultrasonic sensor for feedback to determine altitude.
2. Developed functional velocity controller that used either the Vicon system or Kalman Filter for feedback to determine inertial and body velocities.
3. Developed functional position controller that used either the Vicon system or Kalman Filter for feedback to determine inertial position.

Incomplete:

1. Final tuning of Pixhawk autopilot internal yaw and yaw rate controllers to control heading.
2. Path Planning Design and determination of specific path following/map search algorithms

## Issues

3DRobotics PX4Flow optical flow sensor issues:

1. This semester it was determined that the 3DRobotics PX4Flow optical flow sensor could not output reliable drone body (x,y) velocity values after testing of the sensor in the Lockheed Martin Hallway basement area. The PX4Flow optical flow camera could not output reliable/accurate velocity measurements on the concrete floor surface or on the carpeted area of the hallway. Current speculations for why this happens are poor lighting conditions and/or insufficient texture to the floor surface in the hallway to obtain reliable/accurate velocity measurements. Velocity measurements for testing in the Lockheed Martin Hallway currently come from the output of the Kalman Filter which uses the Hokuyo laser scanner, Euler angles, and accelerometer measurements from the drone to fuse an estimate of the drone’s velocity.
2. The ultrasonic sensor on the PX4Flow optical flow sensor would also occasionally return a 0m altitude reading during flight. After plotting the output altitude measurements directly from the ultrasonic sensor, it was determined that the sensor would have numerous 0m altitude readings returned even during a short duration flight. To remediate this issue, a simple filtering scheme was written within the altitude controller script to check if a zero altitude measurement was recorded and to account for this false value by using the most recent non-zero altitude value.
3. The controllers from the Fall 2014 semester were completely redesigned for the Spring 2015 semester after deciding to fly the drone using the Pixhawk autopilot’s internal “Stabilize” mode and internal roll, pitch, and yaw controllers. Therefore it was determined that the velocity controller script would output desired roll and pitch angles to achieve the desired inertial/body (x,y) velocities. The desired roll and pitch angles would then pass through an angle2pwm function which uses the RC Handset channel calibrated values to map the desired angle and convert it to a PWM value. This PWM value would then be sent to the Pixhawk autopilot where the PWM value would be converted back to the desired angle and this angle would then be controlled by the “Stabilize” mode’s internal roll and pitch controllers.

## Lessons Learned

1. Fully characterize a sensor before integrating it within the platform. After numerous tests it was determined that the PX4Flow optical flow sensor’s ultrasonic sensor would not work in the carpeted section of the Lockheed Martin Hallway. Testing of this sensor on the carpeted section concluded that the altitude measurement would return a 0m value after being placed immediately above the carpeted section in the hallway even though it would return reliable/accurate altitude measurements on the concrete section.
2. Tuning of the altitude controller involved using Research and Engineering Center for Unmanned Vehicles (RECUV)’s Vicon motion capture system. RECUV’s Vicon motion capture system allowed the DVZ team to put infrared motion capture markers on the drone system which allowed the drone’s inertial (x,y,z) position and Euler angle orientation to be captured during flight. The Vicon system allowed precise closed loop control of the drone’s altitude such that the altitude controller’s Proportional, Integral, and Derivative (PID) gains could be varied and tuned for accurate flight.
3. Tuning of the velocity controller involved using RECUV’s Vicon motion capture system. RECUV’s Vicon motion capture system allowed the DVZ team to put infrared motion capture markers on the drone system which allowed the drone’s inertial (x,y,z) position and Euler angle orientation to be captured during flight. The Vicon system allowed precise close loop control of the drone’s inertial (x,y) velocities such that the velocity controller’s Proportional, Integral and Derivative gains could be varied and tuned for accurate flight.
4. Tuning of the position controller involved using RECUV’s Vicon motion capture system. RECUV’s Vicon motion capture system allowed the DVZ team to put infrared motion capture markers on the drone system which allowed the drone’s inertial (x,y,z) position and ( Euler angle orientation to be captured during flight. The Vicon system allowed precise closed loop control of the drone’s inertial (x,y) position such that the position controller’s Proportional, Integral and Derivative gains could be varied and tuned for accurate flight.

## Procedures

### Controllers

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 controllers 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.

In order to control the drone using the controller scripts, the scripts are located at:

* DVZ\Software Element\Mobility Subsystem\Controller Scripts\\*

Depending on which platform is being used, either the X8 (Wolverine) or the X8+ (Magneto) one should use navigate to the appropriate folder since the controller gain values and RC Handset channel values are different. The most up to date controller scripts for Wolverine can be found in:

* DVZ\Software Element\Mobility Subsystem\ControllerScripts\Wolverine\ with\_px4flow\_collision\_avoidance\_localization\_log\\*

The most up to date controller scripts for Magneto can be found in:

* DVZ\Software Element\Mobility Subsystem\Controller Scripts\Magneto\with\_px4flow\_collision\_avoidance\_localization\_log\\*

It should be noted that both sets of scripts can be utilized using feedback from either the Vicon system or the Kalman Filter. The “stabilize\*.py” script is the main script that should be run while the other python scripts house the other necessary functions. The “stabilize\*.py” script defines the functions for subscribing to the necessary ROS topics to pull in AMCL, Kalman Filter Statistics, Kalman Filter, PX4Flow, Hokuyo, and NAV Goal data that is needed to run autonomously. It also defines the function for using VRPN to grab the Vicon position and orientation data. The “stabilize\*.py” script also defines all of the controller bounds and controller gains for the chosen platform.

In order to fly the drone using the controller scripts:

1. Ensure Vicon is on and the selected tracked object is labeled either as “wolverine” or “magneto” as seen in the “stabilize\*.py” script.
2. On the Odroid, in the ~\Documents\flight\_scripts\ folder, ensure that the vrpn scripts and transformations.py scripts are present.
3. On the Odroid, in the ~\Documents\flight\_scripts\ folder, “scp” **all** of the scripts from the selected \Magneto\ or \Wolverine\ folder.
4. On the ground station computer, run “roscore”, “px4flow\_launch”, and “hokuyo\_launch” from the command lines to start the PX4Flow sensor and Hokuyo Laser Rangefinder.
5. Determine which source you would like the controller feedback to be coming from, either Vicon or the Kalman Filter (for running the Kalman Filter, see the Localization and Sensing Subsystem Continuity Document). Make the appropriate changes in the “stabilize\*.py” and “controllers\_func\*.py” scripts to select where the feedback source is coming from.
6. On the Odroid, in the ~\Documents\flight\_scripts\ folder, from the command line run “python stabilize\*.py” to run the scripts. Follow the prompts as outlined by the script. Takeoff of the drone should be conducted manually. To hand control to the drone, for Wolverine:
   1. On the Wolverine RC Handset, Channel 5 and 6 should be in their “low” positions i.e. PWM values < 1400.
   2. Start the “stabilize\*.py” script.
   3. Once the script is running, manually takeoff the platform to a desired altitude and stable flight.
   4. Flip Channel 6 “high” to hand control of the drone to the Odroid i.e. PWM values > 1400. Depending on which controller scripts are commented in or out, you will lose stick control on the following channels:
      1. Altitude controller running – lose Channel 3 control i.e. throttle
      2. Velocity controller running – lose Channel 1 and 2 control i.e. roll and pitch
      3. Position controller running – lose Channel 1 and 2 control i.e. roll and pitch
   5. To end the script and hand control back to the pilot, flip Channel 5 “high” to hand full control of all of the stick to the pilot i.e. PWM value > 1400 and manually land and then disarm the drone.

For Magneto:

* 1. On the Magneto RC Handset, Channel 5 and 7 should be in their “low” positions i.e. PWM values < 1400.
  2. Start the “stabilize\*.py” script.
  3. Once the script is running, manually takeoff the platform to a desired altitude and stable flight.
  4. Flip Channel 7 “high” to hand control of the drone to the Odroid i.e. PWM values > 1400. Depending on which controller scripts are commented in or out, you will lose stick control on the following channels:
     1. Altitude controller running – lose Channel 3 control i.e. throttle
     2. Velocity controller running – lose Channel 1 and 2 control i.e. roll and pitch
     3. Position controller running – lose Channel 1 and 2 control i.e. roll and pitch
  5. To end the script and hand control back to the pilot, flip Channel 5 “high” to hand full control of all of the stick to the pilot i.e. PWM value > 1400 and manually land and then disarm the drone.

Note: No autonomous control of the drone’s yaw has been implemented yet as the yaw and yaw rate controllers internal to the Pixhawk have not been tuned.

# Next Semester/Future Expectations

## Prioritized List of Tasks and Objectives

1. Tune the yaw and yaw rate controllers internal to the Pixhawk using Mission Planner.
2. Implement yaw control into the controller scripts such that a desired heading can be achieved.
3. Fine tune the altitude controller for both the Wolverine and Magneto platforms.
4. Fine tune the velocity controller for both the Wolverine and Magneto platforms.
5. Fine tune the position controller for both the Wolverine and Magneto platforms.
6. Finalize path planning architecture and decide on which ROS packages will be implemented.
7. Implement Path Planning Design.

## Starting Points

1. Plug the Pixhawk autopilot into a computer and connect to it using the APM Mission Planner software. Click on the “Config/Tuning” tab and find the PID gain values for the yaw and yaw rate controllers. Change these values such that the desired heading control is achieved.
2. The controller scripts can be found in:

* DVZ\Software Element\Mobility Subsystem\Controller Scripts\\*

Depending on which platform is being used, either the X8 (Wolverine) or the X8+ (Magneto) one should navigate to the appropriate folder since the controller gain values and RC Handset channel values are different. The most up to date controller scripts for Wolverine can be found in:

* DVZ\Software Element\Mobility Subsystem\Controller Scripts\Wolverine\with\_px4flow\_collision\_avoidance\_localization\_log\\*

The most up to date controller scripts for Magneto can be found in:

* DVZ\Software Element\Mobility Subsystem\Controller Scripts\Magneto\with\_px4flow\_collision\_avoidance\_localization\_log\\*

See the Section 2.4 Procedures for information on running the controllers.

* The task of determining the final path planning ROS architecture is opened ended and involves additional research. The intent is to have a high level ROS path planning architecture send altitude, yaw, and (x,y) position commands to the altitude, velcoity, and (x,y) position controllers to control the quadcopter as it moves through the environment.

## Improvement, Updates, Verification

In the development of the altitude, yaw and (x,y) position controllers, no simulations or models were developed to aid in the tuning of the PID gain values. Since much of the tuning of the PID gain values involves changing the gain values and verifying the behavior in a flight test, effort should be put forth into developing an accurate control model of the drone system such that tuning of the PID gain values can be approached more intelligently.