Prototype Sprint #1

Project Overview

The capability of UAVs to rapidly search a large area, especially an area that is difficult to traverse by foot or vehicle, would be invaluable to operations such as search & rescue. However, small UAVs have a very limited flight time.

A system incorporating a UVG equipped with a landing pad that also serves as a charging station would allow the UAV to be delivered to areas of limited access. The UAV could then, being provided with way-points by the user, autonomously take-off, and navigate through the waypoints. After moving through the waypoints, or when the UAV requires recharging, the UAV will return to the UVG and safely land in such a way that the charging unit can connect to the UAV.

Team Expeditus aims to specifically create and deliver software that allows a UAV to autonomously take-off, navigate to way-points designated by the user, and return to the landing pad in manner that allows for charging and redeployment.

Goals

- Autonomous Take-off
  - Reaching a specific height before navigation
  - Provided no obstacles in reaching operating height
- Autonomous Navigation of Waypoints
  - Provided an obstacle free operation space
- Autonomous Landing of UAV that is:
  - Accurate within +/- .1 meters
  - Oriented correctly

Approach

Phase I
This first phase requires the testing of hardware and ensuring the quadrotor is properly configured and is capable of stable and controllable flight. This will mean many iterations of manual flight. Take-off and way-point navigation is solved by the use of MavLink. Testing of these capabilities will first be tested through the use of Mission Planner or APM Planner, which provides a handy interface for setting up take-off and

way point navigation. The last goal of this phase will be a simulation environment to simulate landing algorithms.

Objectives

1. Test of Manual Flight
2. Test of Mavlink Autonomous Control
3. Setting up Simulation Evironment for Quadrotor

Phase II

Computer vision has been determined to hold the most promise to solve the landing goal. Autonomous landing provided by the GPS (detailed in the hardware section below) provides an accuracy of nav-point navigation, given optimal conditions, to within $+/-10$ meters. This is obviously unsatisfactory to use for purposes of landing on a small platform, but it should bring us within a distance of our goal to detect the landing platform from our downward facing camera.

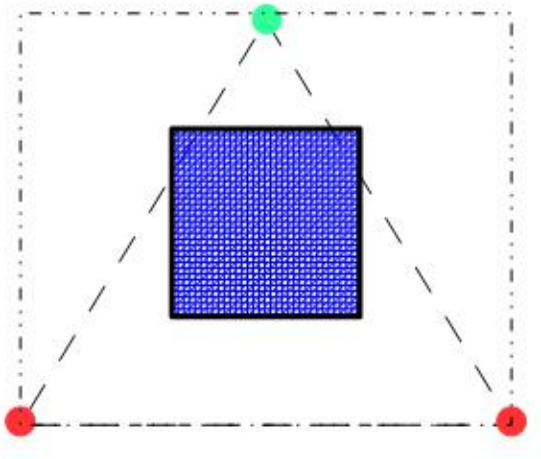

Figure 1: Landing Pad Light & QR Code Layout

Lights: Colored lights arranged in either a rectangle or triangle with different colors on selected lights(figure 1) to provide orientation information for the UAV. The warping and scale of the shape would provide the orientation by the operation needed to unwarp the shape, and the scale would provide distance information. The UAV will use the lights to provide information to the flight controller to manuever above the landing pad at a certain height, and with a specific orientation before switching state to use the QR code.

QR Code: Similar to the lights, warping and scale will provide the UAV with information regarding the orientation and distance between the UAV and the QR code. The QR code will be centered on the landing pad(figure 1).The UAV will be generally centered above the landing pad and at a much closer height, and so this state will be the last fine maneuvers of the UAV to land accurately and with the correct orientation.

ROS will be used as the glue to allow the communications(figure 2) needed between camera, flight controller, and Odroid. Although ROS is not a real-time operating system, it has near real-time capability provided through use of ROS communication tools such as actions, which can provide needed interrupts to activities, and switch state. As the UAV nears the platform, the state will change with the initial image acquisition of the landing platform. This state change will initiate controls to be fed from the software on the Odroid to the flight controller to bring the UAV onto the landing pad with accuracy and correct orientation.

1. Test Autonomous Landing Capability in Simulation
2. Implement Autonomous Landing Capability
3. Test Autonomous Landing Capability

4. Tie Capabilities together
5. Final Testing of UAV with Landing Pad

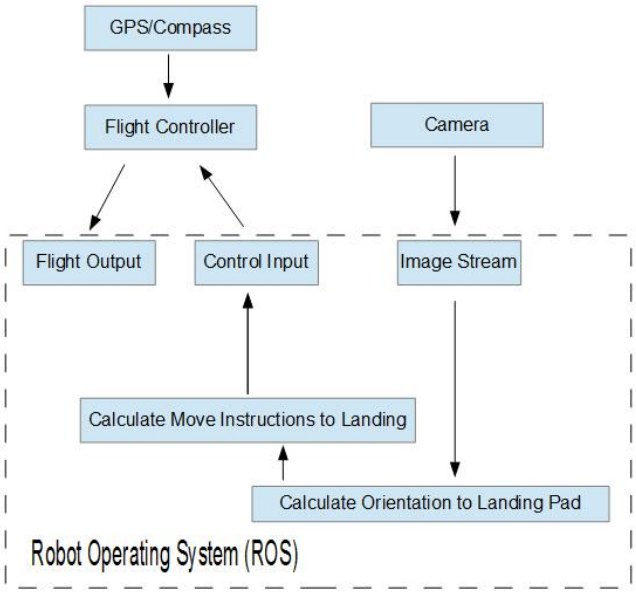

Figure 2: General ROS & Flight Control Setup

Components

Figure 3: Current UAV Build

UAV Hardware
The hardware platform is a quadrotor that had been assembled from a previous iteration of the project(figure 3). To achieve our goals, replacement parts will need to be ordered, as well as some additional equipment to allow the Odroid to coordinate specific landing commands given evaluation of images it receives(figure 4). The platform consists of:

- *APM 2.6+ Assembled*: This serves as the flight controller for the quadrotor. It includes a 3-axis gyro, accelometer, and barometer. It is protected within an enclosure.
- *3DR uBlox GPS with Compass Kit*: GPS and Compass unit with enclosure, external to flight controller to be located on the quadrotor where magnetic interference is least.
- *DIY Quad Kit*: Kit consisting of body and other components such as:
  - Landing Struts(4)
  - 850kV motors(4)
  - Power Distribution Board
  - Electronic Speed Controllers(4)
  - Propellers(4)
  - Power Module & Adapter
  - Radio module
  - Transmitter

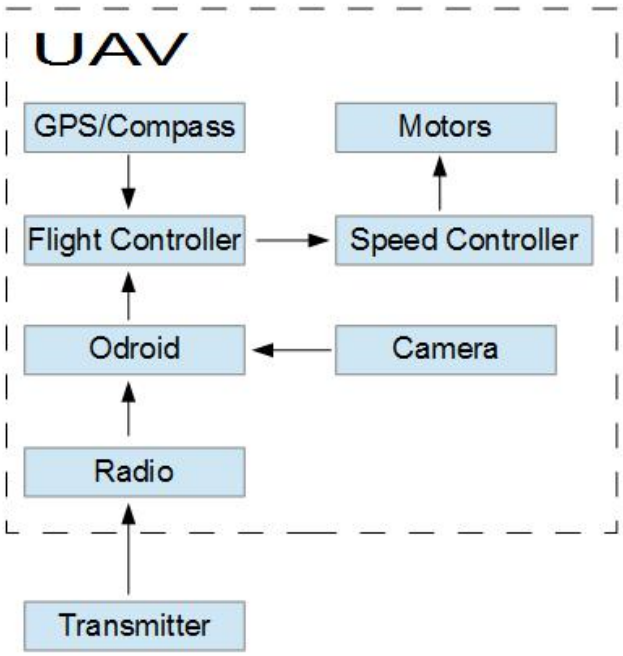

Figure 4: General Hardware Configurations

Additional Hardware

- *USB Camera(2)*: To be used in conjunction with Odroid to provide image stream.
- *Odroid XU4*: To be used to process images and provide instructions to the flight controller during the autonomous landing phase.

UAV Software

Team Expeditus will take advantage of developed software that interfaces and controls the quadrotor. As stated previously, this will provide the autonomous take-off and waypoint navigation. To achieve autonomous landing, the flight controller API will be used as a node within a ROS system.

- *Mission Planner (Windows)*: To be used during testing autonomous take-off flight and way-point navigation.
- *APM Planner(Windows/Linux)*: Alternative to Mission Planner, has Linux support.
- *MavLink*: API to receive output from the flight controller, as well as providing instructions to the flight controller.

Additional Software

- *OpenCV*: Provides a open-source library with image processing and computer vision capabilities.
- *ROS (Robot Operating System)*: An open-source pseudo-operating system employing network communications to link together component functionality to allow fast prototyping and implementation of real robots.
- *MavROS*: A ROS-ified version of MavLink. It functions as a node within the ROS framework.
- *Gazebo*: A simulation environment that can be linked with ROS.

Development Environment

Hardware:

- *Assembly & Repair*: The team has access to the Robotics Lab within the McLaury Bldg on the SDSMT campus. This lab is equipped with all necessary tools for removing, replacing, or otherwise altering the hardware configuration.
- *Flight Training*: The SDSMT UAV Team has provided orientation and access to a UAV flight simulator, so that team members are able to train on manual flight control.

Software:

- *Language*: C++ will be used for the project. No other language is currently forecasted for use.
- *OS*: Development will mostly be conducted in Ubuntu 14.04 which supports the current version of ROS & Gazebo. Current versions of Windows may be used briefly to test hardware through the use of Mission Planner.

Immediate Needs

Hardware:

- *APM 2.6+ Assembled*: Unit was unresponsive during testing.
- *3DR uBlox GPS with Compass Kit*: Unit was unresponsive during testing. Visual inspection revealed a crack across entire enclosure.
- *Odroid XU4*: Missing.
- *Landing Struts(8)*: Missing.
- *USB Cameras(2)*: Required for image capture.

Software:

- None