Design Optimization of a Quad-Rotor Capable of Autonomous Flight

A Major Qualifying Project Report

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

In Aerospace/ Mechanical Engineering

Submitted by:

Antonio DiCesare	**Kyle Gustafson**	**Paul Lindenfelzer**
ad@wpi.edu	*kgus@wpi.edu*	*plinden@wpi.edu*

Advisor:

Michael A. Demetriou

mdemetri@wpi.edu

Abstract

This report covers the design, analysis, manufacturing, and testing of an autonomous quad-rotor helicopter. A control system was designed and implemented through the use of an onboard microprocessor and inertial measurement system. The goal of the helicopter was to maintain a hover at a used-defined altitude while minimizing lateral drift. In addition to achieving autonomous flight, the helicopter attained a 10% weight reduction from an earlier quad-rotor design and which led to an increased flight time.

Table of Contents

List of Figures

List of Tables

1. Introduction

The helicopter is one of the most complex flying machines due to its versatility and maneuverability to perform many types of tasks. Classical helicopters are usually equipped with a main rotor and a tail rotor. However, other types exist which use a twin rotor. Our specific project is concerned with the design and control of a miniature rotorcraft, known as a quad-rotor helicopter [1].

Quad rotors are symmetrical vehicles with four equally sized rotors at the end of four equal length rods. Early designs of quad rotors were completed in the 1920's by Etienne Omichen, Dr. George de Bothezat and Ivan Jerome. These designs, however, never truly grasped the attention of the public or the in case of Dr Bothezat and Jerome the military. Therefore, neither Omichen's or Bothezat and Jerome's were mass produced. This fact, however, does not discredit the advantages of quad rotors. Unlike their counter parts, quad rotors make use of multiple rotors allowing for a greater amount of thrust and consequently a greater amount of maneuverability. Also, the quad rotors symmetrical design allows for easier control of the overall stability of the aircraft.

Each of the rotors on the quad-rotor helicopter produces both thrust and torque. Given that the front and rear motors both rotate counter-clockwise and the other two rotate clockwise, the net aerodynamic torque will be zero, as seen in Figure 1.

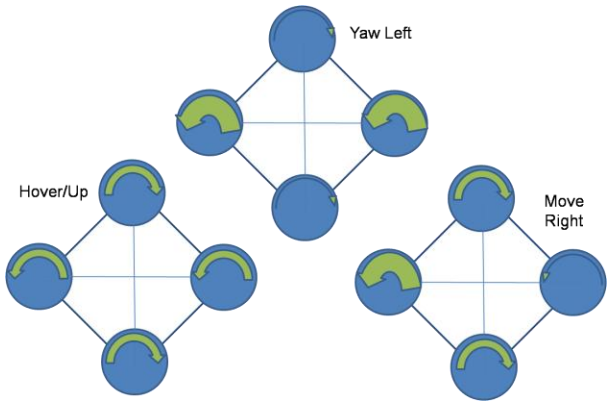


Figure 1- Torque Patterns and Related Motion

The generalized coordinates for a rotorcraft are:

$$q=(x, y, z, \theta, \varphi, \psi) \tag{2.1}$$

Where (x, y, z) denote the position of the center of mass of the rotorcraft relative to the frame, and (θ, φ, ψ) are the three Euler angles which represent the orientation of the craft [9].

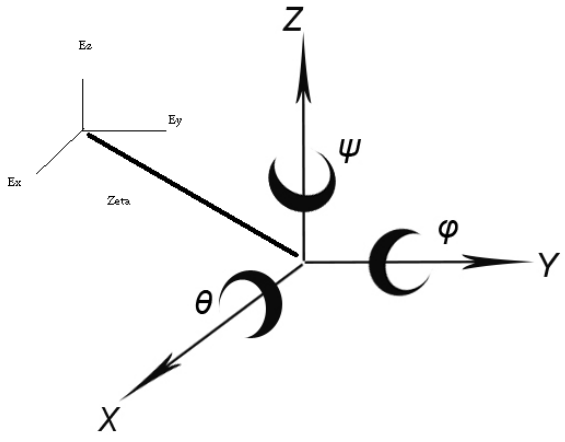


Figure 2- Body and Inertial Frame Coordinate Axis used in this Paper

Our project involves the design of an autonomous quad rotor. Along with optimizing frame design and weight reduction, we have designed controls for the quad rotor with the use of the MSP430 and a custom C code. With the controls on board we were able to program a flight onto the processor of the quad rotor and allow the quad rotor to hover at a given altitude autonomously.

This project is a continuation of the 2007-2008 MQP team's quad rotor, however it is a completely new design in almost every respect. The control equations are similar; their implementation is completely different. The few components carried over from the previous project were the motors, speed controllers, and yaw gyro. The following goals were achieved with our helicopter:

1. Achieve a completely autonomous flight
2. 10% weight reduction over previous prototype
3. Increase total time of flight
4. Obtain on board autonomous control

2. History

Research into the initial development of quad rotors began in the early twentieth century. One of the first engineers to attempt to design a quad rotor was Etienne Oemichen. Oemichen began his research in 1920 with the completion of the Oemichen No.1. This design consisted of four rotors and a 25 Horsepower motor; however, during tests flights the Oemichen No.1 was unable to obtain flight. Two years later Oemichen completed his second design; the Oemichen No.2. His second design consisted of four rotors and eight propellers along with a 125 Horsepower motor. Five of the propellers were used to achieve stable flight while two were used for propulsion and the final propeller being used to steer the aircraft. In April of 1914, the Oemichen No.2 achieved an FAI distance record for helicopters of 360m, which the Oemichen No.2 broke with a distance of 525m [19].

While Oemichen had begun working on his early designs in France, Dr. George de Bothezat and Ivan Jerome began their own research in January 1921 for the United States Army Air Corps. They completed their design in mid 1922, and the first test flight took place in October of 1922 in Dayton, Ohio. Bohezat's and Jerome's design weighed around 1700 kg at the time of take off and consisted of four six-bladed rotors along with a 220-HP motor. After many tests, the quad rotor was only able to achieve a maximum flight time of 1 minute 42 seconds and maximum height of 1.8 meters [20].

Following the research of Oemichen, Bothezat and Jerome, other researchers have attempted to create their own successful vertical flying machines. One such was being the Convertawings Model "A" quad rotor. The Convertawings Model "A" quad rotor was designed and built in the mid 1950's with civil and military purposes in mind. This particular quad rotor

consisted of four rotors, two motors as well as wings. Due to lack of interest, however, the Convertawings Model "A" quad rotor was never mass produced.

Currently Bell Helicopter Textron and Boeing Integrated Defense Systems are doing joint researched on the development of the Bell Boeing Quad Tilt Rotor. The initial design consists of four 50-foot rotors powered by V-22 engines. The main role of the Bell Boeing Quad Tilt Rotor will be that of a cargo helicopter with the ability to deliver pallets of supplies or also deploy paratroopers. The first wind tunnel tests were completed in 2006 and the first prototype is expected to be built in 2012.

3. Quad Rotor Dynamics

To understand how to control the helicopter, we must first understand how it behaves. The derivation of the equations of motion is built of the Lagrangian equations of motion for both translational and rotational kinetic energy, and potential energy. The energy equation for a quad rotor has three terms, the translational kinetic energy, the rotational kinetic energy, and the gravitational potential energy.[3] The vector ξ is defined as:

$$\xi = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3.1}$$

where x, y , and z are the translational unit vectors. The vector η is defined using the rotational axis unit vectors as:

$$\eta = \begin{pmatrix} \theta \\ \varphi \\ \psi \end{pmatrix} \tag{3.2}$$

If we define the Lagrangian kinetic energy equations as:

$$L(q, \dot{q}) = T_{trans} + T_{rot} - U \tag{3.3}$$

with q and $\dot{q}$ being the individual vectors ξ and η. Substituting in mass m and ξ and η, we get the following,

$$L(q, \dot{q}) = \frac{m}{2}\dot{\xi}^T\dot{\xi} + \dot{\eta}^T\mathbb{Z}\dot{\eta} + mgz \tag{3.4}$$

with $\mathbb{Z}$ being the inertial matrix. Using Eq. (3.4) and the following Eq. (3.5) we can determine the motion of the quad rotor:

$$F = \left(\frac{d}{dt}\right)\left(\frac{dL}{d\dot{q}}\right) - \left(\frac{dL}{dq}\right) \tag{3.5}$$

where F is the control input vector $(0\ 0\ u)^T$ and a function of the translational forces and u being the sum of all the forces from the four rotors. From this information we can then determine the translational forces on the quad rotor using the Eq. (3.6):

$$F\xi = R\hat{F} \tag{3.6}$$

In this equation *R* is the direction cosine transformation matrix, where *c* and *s* are cosine and sine, respectively:

$$R = \begin{bmatrix} \mathrm{c}(\theta)\,\mathrm{c}(\psi) & \mathrm{s}(\theta)\,\mathrm{s}(\psi) & -\mathrm{s}(\theta) \\ \mathrm{c}(\psi)\,\mathrm{s}(\theta)\,\mathrm{s}(\varphi) - \mathrm{s}(\psi)\,\mathrm{c}(\varphi) & \mathrm{s}(\psi)\,\mathrm{s}(\theta)\,\mathrm{s}(\varphi) + \mathrm{c}(\psi)\,\mathrm{c}(\varphi) & \mathrm{c}(\psi)\,\mathrm{s}(\varphi) \\ \mathrm{c}(\psi)\,\mathrm{s}(\theta)\,\mathrm{c}(\varphi) + \mathrm{s}(\psi)\,\mathrm{s}(\varphi) & \mathrm{s}(\psi)\,\mathrm{s}(\theta)\,\mathrm{c}(\varphi) & \mathrm{c}(\theta)\,\mathrm{c}(\varphi) \end{bmatrix} \tag{3.7}$$

By carrying out Eq. (3.6) we then determine the equations of motion for the quad rotor. The derivation of the Coriolis term can be found in "Modeling and Control of Mini-Flying Machines", however for simplicity this term can be removed without significant loss effect [1].The end result yields the following non-linear equations of motion:

$$m\ddot{x} = -u\sin\theta \tag{3.8}$$

$$m\ddot{y} = u\cos\theta\sin\varphi \tag{3.9}$$

$$m\ddot{z} = u\cos\theta\cos\varphi - mg \tag{3.10}$$

$$\ddot{\varphi} = \tilde{\tau}_{\varphi} \tag{3.11}$$

$$\ddot{\theta} = \tilde{\tau}_{\theta} \tag{3.12}$$

$$\ddot{\psi} = \tilde{\tau}_{\psi} \tag{3.13}$$

Now that the equations of motion are defined, we can create control inputs based on translational and rotational movement. The first control input is *u* which is defined as follows:

$$u = f_1 + f_2 + f_3 + f_4 \tag{3.14}$$

$$f_i = k\omega^2 \tag{3.15}$$

where f_i is thrust of each motor and *k* is a constant related to the aerodynamics of the rotor [8]. In level flight, this input will correlate to the total thrust in the *z* direction. The next three control inputs correspond to torques about the θ, ϕ and ψ axes. As will be shown in Section 6, these four control inputs are enough to move the helicopter in three dimensions. The τ_i variables are the torques about each axis and are defined as:

$$\tau_\psi = \sum_{i=1}^{4} \tau_{M_i} \tag{3.16}$$

$$\tau_\theta = (f_2 - f_4)l \tag{3.17}$$

$$\tau_\varphi = (f_3 - f_1)l \tag{3.18}$$

where *l* is the length from the center of gravity to the motor and τ_{Mi} is the couple produced by each motor. From these equations it can clearly be seen how for a vertical ascent, all motors must produce the same thrust, otherwise Eqs. (3.11) and (3.12) will not equal zero and there will be a pitching or rolling action. From Eqs. (3.8), (3.9) and (3.10) it is shown that this will create an undesirable translational movement.

The desired outputs for each individual motor can be calculated from the following equation:

$$\begin{bmatrix} u \\ \tilde{\tau}_\theta \\ \tilde{\tau}_\phi \\ \tilde{\tau}_\psi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -c & c & -c & c \\ 0 & -l & 0 & l \\ -l & 0 & l & 0 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{3.20}$$

where c is the torque produced by each motor about the ψ axis and l is the moment arm to each motor.

4. Control Theory

4.1 Overview

The first iteration of the quad rotor used an inertial measurement unit consisting of MEMS gyroscopes and accelerometers and an infrared rangefinder for the *Z* distance measuring; all components were controlled by a Texas Instruments MSP430 on an Olimex development board. It was first thought that a math coprocessor would need to be used to handle floating-point arithmetic; however the MSP by itself was able to run the entire code 1000 times per second, which was more than fast enough for the quad rotor.

4.2 PID Control

The quad rotor will use a Proportional-Integral-Derivative control system, which will be tuned to determine the optimum response and settling time. The PID controller Eq. (4.9) is a closed-loop feedback system which will output a control signal *u* and receive feedback from the inertial sensors. The controller then calculated the difference between the desired position and orientation and the current position and orientation and adjusts *u* accordingly. The equation for a PID controller is as follows:

$$u = P + I + D \tag{4.1}$$

Define:

$$e(t) = e_d(t) - e_a(t) \tag{4.2}$$

where e_d is the desired condition, e_a is the actual (measured) condition and e(t) is the difference (error) between the two at each individual time step. First, the proportional term K_p is defined as a whole number greater than 0 (for a stable system) which is simply some fraction of the error.

As an example set k_p to .5. Our system is traveling from e_0 to e_d. At time *t* it is halfway between the two, so *e(t)* is .5*e_d. The proportional term *P* is defined as follows:

$$P = k_p e(t) \tag{4.3}$$

After plugging in .5 for K_p and .5e_d for e(t), u becomes .25e_d. This is a fourth of the original control input, and thus will lead to a quick exponential convergence to the desired position. This system, while theoretically plausible, doesn't lend itself well to the quad rotor for several reasons. First, there is always a lag between the measured state and the corrective action; combined with the response time of the motors and the inertia of the system, the helicopter could become extremely unstable unless all of those factors are accounted for. The proportional and integral terms in the PID controller are what compensate for rate at which the error is changing and the rate at which the system is changing. Feed-forward control, which has been discussed in previous sections, accounts for the behavioral dynamics of the helicopter, such as its momentum and motor response time.

The integral term determines the magnitude of the accumulated error by summing the instantaneous error over time. This value is "uncorrected error" that was not dealt with during the previous time step. Notice in this equation *t* (instantaneous time) is replaced by τ which is the past time. Adding the integral term increases the overshoot but decreases the settling time.

$$I = K_i \int_0^t e(\tau)\, dt \tag{4.4}$$

The derivative term determines the rate at which the error is changing, and by decreasing the rate of change near the set point it reduces overshoot and increases the settling time [12].

$$D = K_d \frac{de}{dt} \tag{4.5}$$

The determination of K_p, K_I and K_d is determined by tuning the controller until reaching the desired settling and overshoot is reached. Table 1 shows some methods for tuning the controller.

Table 1-PID Controller Tuning Options [16]

<table>
<tr>
<td>Manual Tuning</td>
<td>Ki and Kd are set to zero, and Kp is changed until the system oscillates. From there, it is set to one half that value. I is then increased to reduce settling time without causing instability. Finally, D is increased until the overshoot is minimized without over damping the system.</td>
<td>Pros: Can be changed on the fly; no calculation required

Cons: Not as accurate as other methods; experience required to prevent wasted trial and error</td>
</tr>
<tr>
<td>Ziegler-Nichols (First Method)</td>
<td>Involves studying of the response of a plant to a unit-step input. If the response is an S-shaped curve, the delay time and time constant are determined by a tangent line at the inflection point of the S-curve.

The transfer function then equals

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1}$$

The values for Kp, Ti and Td are determined by
<table>
<tr><th>Type of Controller</th><th>K_p</th><th>T_i</th><th>T_d</th></tr>
<tr><td>P</td><td>T/L</td><td>∞</td><td>0</td></tr>
<tr><td>PI</td><td>0.9 T/L</td><td>L/0.3</td><td>0</td></tr>
<tr><td>PID</td><td>1.2 T/L</td><td>2L</td><td>0.5L</td></tr>
</table>
</td>
<td>Pros: Can be changed on the fly; more accurate than manual tuning

Cons: Difficult to set exactly to critical gain</td>
</tr>
</table>

<table>
<tr>
<td>Ziegler-Nichols (Second Method)</td>
<td>Is completed by setting $T_i = \infty$ and $T_d = 0$ in a closed-loop system with a proportional controller and increasing K_p to K_{cr}. Where at K_{cr} the output would consist of sustained oscillations.

K_p, T_i and T_d are determined using
<table>
<tr><th>Type of Controller</th><th>K_p</th><th>T_i</th><th>T_d</th></tr>
<tr><td>P</td><td>$0.5K_{cr}$</td><td>∞</td><td>0</td></tr>
<tr><td>PI</td><td>$0.45K_{cr}$</td><td>$1/1.2\ P_{cr}$</td><td>0</td></tr>
<tr><td>PID</td><td>$0.6K_{cr}$</td><td>$0.5\ P_{cr}$</td><td>$0.125\ P_{cr}$</td></tr>
</table>
</td>
<td>Pros: More accurate than manual tuning

Cons: May be difficult to determine K_{cr}</td>
</tr>
</table>

For the quad rotor, the Ziegler-Nichols method or manual tuning will be used to minimize cost and training time.

This quad rotor uses a variation of the PD controller; that is no integral term is calculated. In terms of the quad rotor, the proportional term is defined as a constant a_i multiplied by the difference between the measured and desired position, be it lateral or angular position. The derivative term is a constant a_j multiplied by the measured velocity. More on the actual control equations is in Section 4.3.1.

4.3 Inertial Measurement

Inertial Measurement Systems sense inertial forces on a body and from those forces linear and angular position and velocity can be calculated. There are two types of INS: inertially stabilized and strapdown system. The difference comes from the frame of reference the unit is aligned to. In an inertially stabilized unit the gyroscopes remain fixed in reference to an inertial

(navigation) reference frame (Fig. 2) and do not rotate with the vehicle. The unit is mounted on gimbals which allow it to rotate on all three axes; gyroscopes are mounted orthogonally on the gimbal to cancel the precession of the unit and thus keep it aligned to the inertial frame rather than the body frame. A pickoff is a sensor which measures the angles of the gimbal with respect to the vehicle, thus determining angular orientation. The gyroscopes do not measure angles directly, they are only used to keep the unit aligned with the reference frame [10].

In a strapped-down system the gyroscopes rotate with the body, and by integrating angular velocities from an initial position the orientation and position of the vehicle can be determined. In this system, however, the body accelerations must be converted to accelerations in the navigation frame through the following equations [10].

$$(a_x)_{nav} = a_x \cos\theta - a_y \sin\theta \tag{4.6}$$

$$(a_y)_{nav} = a_x \sin\theta + a_y \cos\theta \tag{4.7}$$

In Eqs. (4.6) and (4.7) a_x and a_y are the body accelerations measured by the strapdown accelerometers. The *z* accelerations are measured using a direction cosine transformation matrix *R* as shown in Eq. (3.7) from Section 3.1.

For the quad rotor, a strapdown system consisting of a MEMS gyroscope and accelerometers was used. Although stabilized platform INS's are more accurate, they are far more complex, expensive, and heavy for an indoor autonomous helicopter. An off-the-shelf gyro from SparkFun Electronics and a two axis accelerometer from Analog Devices were used.

Also, to get a precise altitude measurement, an infrared rangefinder was used. The sensor provides distance measurements to +/-.3mm. This was much more accurate than could be integrated from accelerometer measurements.

4.3.1 Fundamental Equations

The position and orientation of an aircraft can be described by 12 variables which correspond to the translational and rotational positions and their respective velocities. Utilizing a state-space transformation we can create a matrix which includes our dynamic equations from Section 4 along with the information measured from our INS. The state matrix is shown in Eq. (4.8). By initializing the aircraft at a known position, the current state can be found by integrating the measured accelerations and velocities.

$$X = \begin{bmatrix} x \\ y \\ z \\ \theta \\ \varphi \\ \psi \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \\ \dot{\varphi} \\ \dot{\psi} \end{bmatrix}, \dot{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \\ \dot{\varphi} \\ \dot{\psi} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\theta} \\ \ddot{\varphi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ -\frac{u}{m}\sin x_7 \\ -\frac{u}{m}\cos(x_7)\sin(x_8) \\ \frac{u}{m}\cos(x_7)\cos(x_8) - g \\ x_{10} \\ x_{11} \\ x_{12} \\ \tau_\theta \\ \tau_\varphi \\ \tau_\psi \end{bmatrix} \tag{4.8}$$

We can simulate an open-loop response MatLab [13] using a constant *u* as seen in Figure 3. The graph on the bottom shows an example PID control (Section 4.1) combining these two equations will give us the desired control scheme.

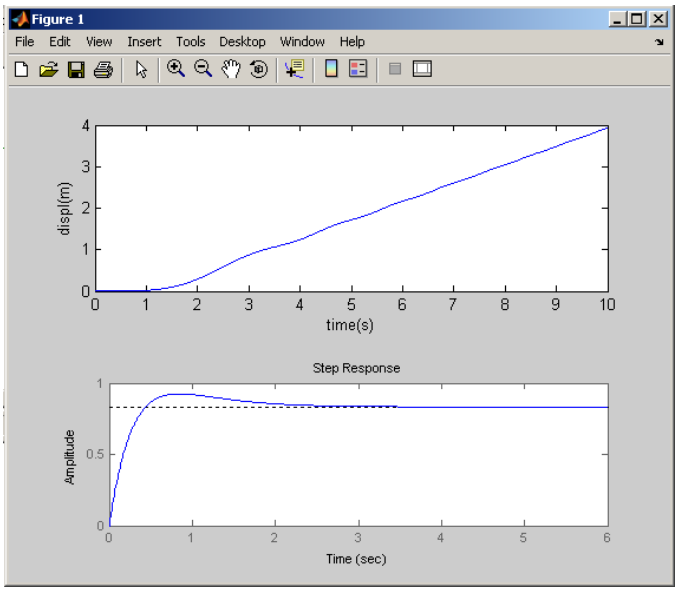


Figure 3-Z Displacement (top) and PID Control

The original sensor package called for a three-axis accelerometer and two-axis gyro, along with a third yaw gyro. The five-axis IMU (Figure 4) was damaged during soldering, and a new control scheme had to be developed using accelerometers in place of the gyros.

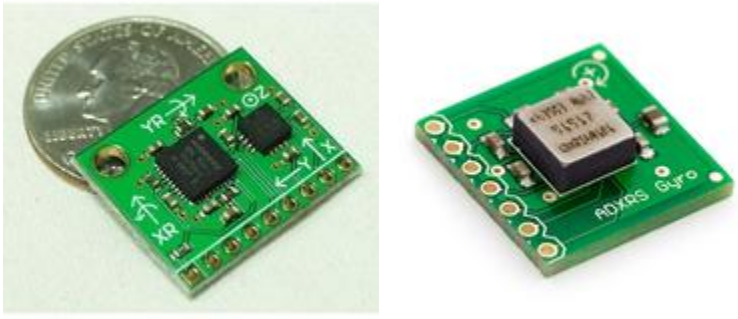


Figure 4-Sparkfun 5-Axis IMU and Gyro Breakout Board © Sparkfun Electronics

Altitude Control

The control for the altitude is modeled using only proportional and derivative terms; this simplifies the equations and reduces the processing power needed. It is also assumed that the roll and pitch angles are very small. The basic control equation is:

$$u = (-k_{dz}\dot{z} - k_{pz}(z_a - z_d) + mg)\frac{1}{\cos\theta \sin\phi} \tag{4.9}$$

where z_a is the actual (measured) position and z_d is the desired altitude, k_{dz} and k_{pz} are the damping constants from the PID controller. The same principle is applied to the yaw control (Eq. 4.10), to ensure the helicopter doesn't pirouette about the z-axis as it ascends or descends.

$$\tilde{\tau}_\psi = -k_{d\psi}\dot{\psi} - k_{p\psi}(\psi_a - \psi_d) \tag{4.10}$$

Combining the equation for *u* with the quad rotor dynamics equations in Section 4.1, the following equations arise:

$$\ddot{x} = -\frac{1}{m}\left(-k_{dx}\dot{x} - k_{px}(x_a - x_d) + mg\right)\frac{\tan\theta}{\cos\phi} \tag{4.11}$$

$$\ddot{y} = \frac{1}{m}\left(-k_{dy}\dot{y} - k_{py}(y_a - y_d) + mg\right)\tan\phi \tag{4.12}$$

$$\ddot{z} = \frac{1}{m}\left(-k_{dz}\dot{z} - k_{pz}(z_a - z_d)\right) \tag{4.13}$$

$$\ddot{\psi} = -k_{d\psi}\dot{\psi} - k_{p\psi}(\psi_a - \psi_d) \tag{4.14}$$

Pitch and Roll Control

The following equations (4.15 and 4.16), which were derived by [1], were overly complex for the needs of the quad rotor; it wasn't necessary to have precise (x,y) control or to travel great horizontal distances. These equations feature saturation functions to ensure that limits on motor speed, tilt angle, or angular rate aren't exceeded. If we assume the helicopter will be relatively stable with small pitch and roll and angles and slow movement, then we can use the

same form of the control equation we used for altitude and yaw. Note the saturation functions have been removed for clarity.

$$\tilde{\tau}_{\phi} = -(\dot{\phi} + \left(\phi + \dot{\phi} + \left(2\phi + \dot{\phi} + \left(\dot{\phi} + 3\phi + 3\frac{\dot{y}}{g} + \frac{y}{g} \right) \right) \right)) \tag{4.15}$$

$$\tilde{\tau}_{\theta} = -\left(\dot{\theta} + \left(\theta + \dot{\theta} + \left(2\theta + \dot{\theta} + \left(\dot{\theta} + 3\theta + 3\frac{\dot{x}}{g} + \frac{x}{g} \right) \right) \right) \right) \tag{4.16}$$

The simplified pitch and roll equations are as follows:

$$\tilde{\tau}_{\phi} = -a_{1\phi}\dot{\phi} - a_{2\phi}(\phi - 0) \tag{4.17}$$

$$\tilde{\tau}_{\theta} = -a_{1\theta}\dot{\theta} - a_{2\theta}(\theta - 0) \tag{4.18}$$

4.3.2 Implementation

While the aforementioned equations seem to be able to control the helicopter all by themselves, there is actually quite a lot more besides just the equations. The IMU sensors measures angular rate and linear acceleration, however the control equations need absolute angles as well as linear velocity. Since we are not concerned with horizontal drift, we only need

to worry about integrating the angular rates into angles, and differentiating the *z*-position to get z-velocity.

The integration is performed by using Simpson's Rule (Eq. 4.19), which is exact for third-order or fewer systems. This simple numerical integration technique is both simple to program and easy on processing power. In this case, between time steps t_1 and t_2 the angle would be found (in this case ψ) by Simpson's Rule.

$$\int_{t_1}^{t_2} \dot{\psi}\, dt = \frac{t_2 - t_1}{6}\left(\dot{\psi}_{t1} + 4\frac{\left(\dot{\psi}_{t1} + \dot{\psi}_{t2}\right)}{2} + \dot{\psi}_{t2}\right) = \Delta\psi \tag{4.19}$$

Notice that the integration only solves for the *change* in the ψ; software code continuously adds all of the angle measurements to get the current value. Over a long period of time or if there were drastic changes in angle then this system would accumulate error fast. Fortunately that is not a problem with this helicopter.

One issue with using this straightforward integration with Simpson's rule is that it doesn't filter out noise; instead the noise is carried throughout the arithmetic and the final answer is actually $\Delta\psi + \delta$, where δ is the signal noise. Time permitting, a Kalman filter could have been implemented to give an much better result for our yaw angle. The code required for a Kalman filter is beyond the team's programming capability, and thus is left as a recommendation for future projects.

Differentiation of the *z*-position is much simpler, the algorithm is:

$$\dot{z} \equiv \frac{z_2 - z_1}{t_2 - t_1} \tag{4.20}$$

As mentioned in section 4.2, the Sharp GP2D12 Infrared Rangefinder was used to measure altitude. It outputs a nonlinear analog voltage that corresponds to distance. The distance to voltage function can be seen in Appendix G [20].

Figure 5-Sharp GP2D12 Infrared Rangefinder

4.4 Electronic Speed Control

One critical factor to ensure stable flight is knowledge of the exact rotor RPM, which in turn will mean a better idea of torque. At a constant PWM, the motors will spin at various RPM depending on the dynamic loads placed on them. This can create unwanted forces which create more errors. One option to ensure motor speed is to place an optical encoder on the motor shaft and combine it with a PID controller to ensure desired RPM. No encoders were found that were small and light enough to be placed on the motors; however the Phoenix 10 Electronic Speed Controller can be programmed to maintain a constant RPM through the use of a "governor mode". In this mode, PWM is varied to maintain a constant RPM whereas in "throttle mode" a steady PWM is generated.

The Phoenix 10 is the same speed controller used in the previous quad rotor; they cost about $60 each and the Phoenix had all the functionality we needed so there was no reason to choose another ESC. The Phoenix 10 works by monitoring the back EMF (the "voltage induced

in a motor wire by the magnet spinning past its coils [17]") to know exactly where the rotor is. Then, a PID algorithm in the controller adjusts the PWM to maintain a constant RPM.

Figure 6-Phoenix 10 Electronic Speed Controller © castlecreations.com

5. Mechanical Design

5.1 Overview

Typical quad-rotors utilize a four-spar method, with each spar anchored to the central hub like the spokes on a wheel. The previous MQP used a machined Aluminum 6061 hub and square tubular carbon fiber rods. The propeller used was made of nylon by APC Propellers, and was nine inches in diameter with a pitch of six inches. Propellers will be discussed in subsequent sections.

The mechanical design of the quad-rotor is relatively simple. The primary consideration, as with most aircraft, is weight. The group's goal of 10% weight reduction came primarily in the hub design. Instead of the solid block, a truss frame was developed, with a cantilevered spar design, which can be seen in Figure 3. Stress calculations for the rod were based on readily available hollow carbon fiber tubes. The stress distribution can be seen in Figure 7. The maximum stress areas, shown in red at about 6.9 GPa, are well under carbon fiber's 220GPa yield strength. This test assumed a 2 kg load, almost four times the expected weight of the aircraft.

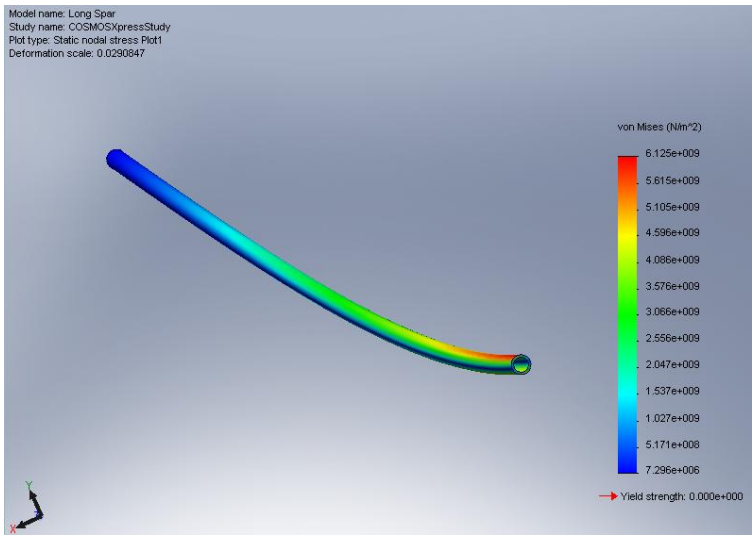


Figure 7- Von Mises Stresses on Carbon Fiber Rod

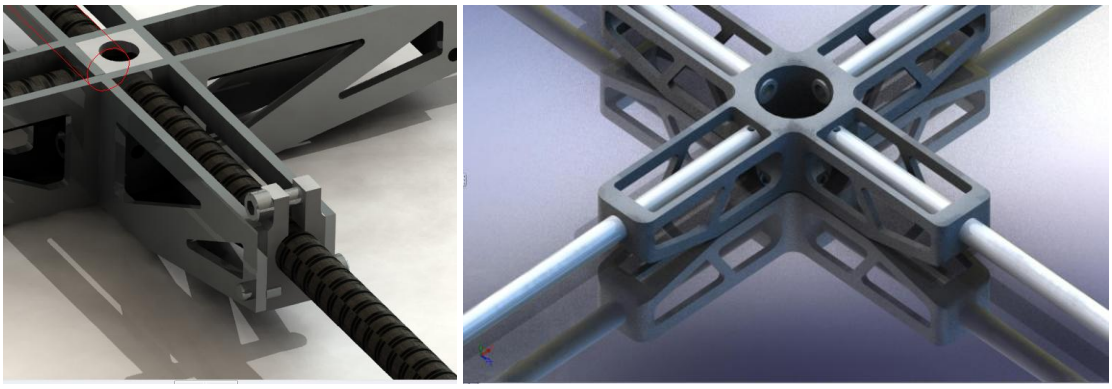


Figure 8- Render of First Hub Design (left) vs Final Unibody Design

The primary load on the quad-rotor is in the *z* direction, which is along the strongest axis of the truss. The entire hub was made on a rapid prototyping (RP) machine out of ABS plastic. The original design, while light and strong, could not be manufactured to tight enough tolerances on the RP machine. The unibody design ensured the spars, and thus the motors, were all perfectly aligned.

5.2 Material Selection

When designing an autonomous quad-rotor, there are several material options which must be considered. Any design must consider different materials based on durability, machinability, and price. When dealing with a machine capable of flight, then one must consider weight a major factor. The materials in consideration for our design include aluminum, plastic, and carbon fiber.

Aluminum, historically, has been the material of choice for RC helicopters. Aluminum is light and strong, dissipates heat well, and is relatively inexpensive in comparison to some of the other possibilities. The negative for aluminum is that it tends to be too heavy for small aircraft models. Also, aluminum can develop cracks over time from vibrations.

Plastic absorbs vibration much better the previously mentioned aluminum. Also, it is fairly durable and will return to its original shape if bent. Plastic is also very inexpensive, light, and can be machined very easily. Various types of plastics were explored, including Nylon®, polypropylene, Delrin®, Ultem®, polyethylene, and ABS. The loads placed on these parts are well within the yield strength of the materials, so ultimately the decision came down to price, and raw Nylon was the cheapest plastic available.

Another plastic option presented itself, and that was rapid prototyping of ABS using fused-deposition modeling (FDM). FDM uses a 3-D printer to lay down thin lines of plastic. The printer builds pieces in layers, and this must be taken into consideration in the design. Holes which are perpendicular to the horizontal plane will be smooth, but holes whose center axis is parallel to horizontal will be built in “steps”, and thus aren’t suitable for applications where smoothness or threads are required (see Figure 23.b for details). The resolution of the machine is the thickness of lines, which is about .2cm. This means that machined pieces will not be close to the accuracy of CNC machining of metal. Finally, certain thin features aren’t possible because

two lines of fused plastic aren't as strong as a similar size of machined ABS. The main selling point for FDM is its extreme simplicity to use (export directly from SolidWorks into Catalyst, which then determines proper orientation and prints the part), its cost ($6-8 per cubic inch), and speed of manufacture (about 2 hours for all of this helicopter's parts).

Carbon Fiber is currently the best material available for RC helicopters. It is stronger and lighter than aluminum and absorbs vibration better than plastic. It can be molded to be super stiff in one direction and flexible in the other. But, it is also much more expansive than other materials. Also, it is difficult to machine so it would require an outside source to manufacture the required parts.

Table 2- Material Comparison

Material	Modulus of Elasticity (GPa)	Tensile Strength (MPa)	Density (g/cm^3)
Nylon 6.6	2.61	82.8	1.14
ABS	.001	29.0	1.02
Ultem®	3.45	114	1.28
Delrin®	2.55	52.4	1.42
Carbon Fiber	220	760	1.7
Stainless Steel 404	200	1790	7.80
Aluminum 7075	71	572	2.80

The group determined that it would be using carbon fiber as the material of choice for the spars used on the aircraft. Also, the FDM machine was used to manufacture all other necessary frame components due to its price and machinability.

5.3 Prototype

After making an FDM prototype, it was determined that the rapid prototype parts were suitable for use on the prototype helicopter. A strength test showed that the frame could support 1.5kg at the center of the hub while being supported from each motor mount. Also, the entire frame including spars and motor mounts had a mass of 39g, compared to 39.9g for just the aluminum hub on the previous version of the quad rotor.

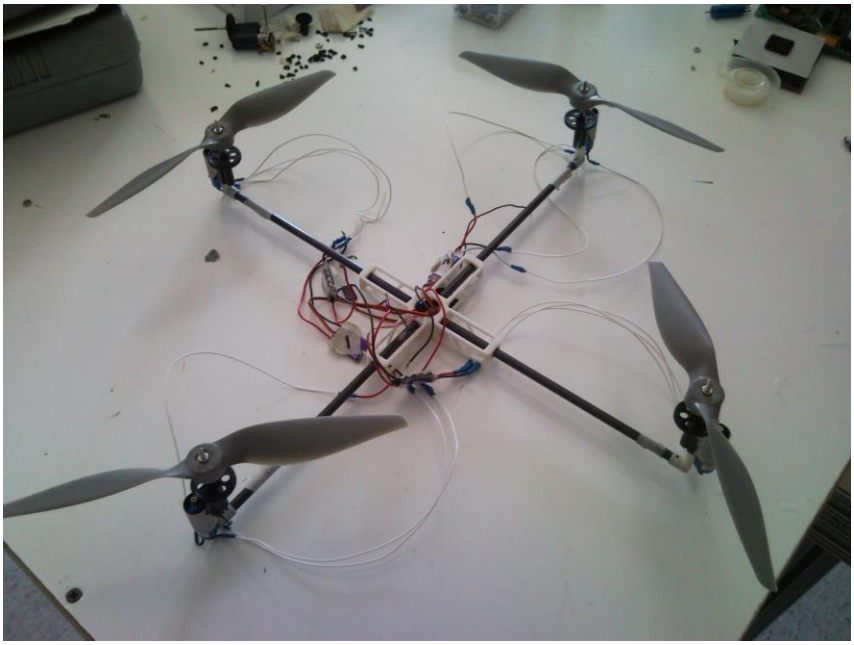


Figure 9-Complete Quad Rotor Assembly

The FDM parts did have drawbacks however. Because the resolution of the machine is about .02”, there were issues getting parts to fit. The motor mounts had to be sanded considerably to get the gearboxes to fit on them. This was done by hand and resulted in the mounts not being perfectly straight. This problem was mitigated by realigning the gearboxes as they were screwed in.

The .254” OD carbon fiber rods selected proved difficult to machine as well. Cutting of the rods was accomplished by wrapping them in masking tape and using a hacksaw, and were sanded and glued to the ABS parts with cyanoacrylate glue.

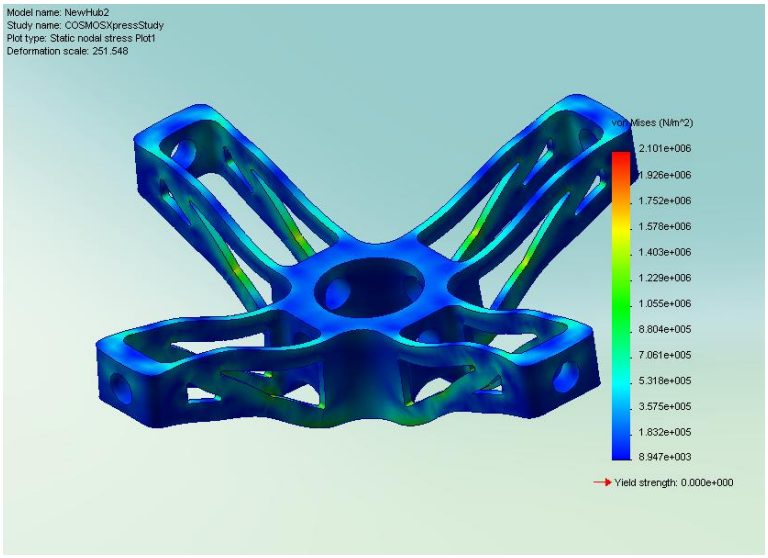


Figure 10-Central Hub Showing Stress Distribution from Z-Loading

5.4 Motor Alignment

As previously mentioned, the act of mounting the motor mounts to the frame was extremely difficult due to the fact that the holes needed to be drilled to within +/- $.1^{o}$ of each other. Any successful flying device must be perfectly balanced, and this is very prevalent when dealing with a quad rotor. In order to achieve a balanced vertical flight, the group needed to be certain that the motors were perfectly straight. If the motors were not perfectly aligned, achieving a balanced, smooth flight would be nearly impossible. With this in mind, the group devised a way to be sure that all the motor mounts were aligned correctly. By setting the frame of the quad rotor on a flat surface, the group used a T-square held against the motor mounts. If the mounts were directly against the edge of the square, then the mount would be aligned in a 90 degree angle. An image of this process can be seen below in Figure 7.

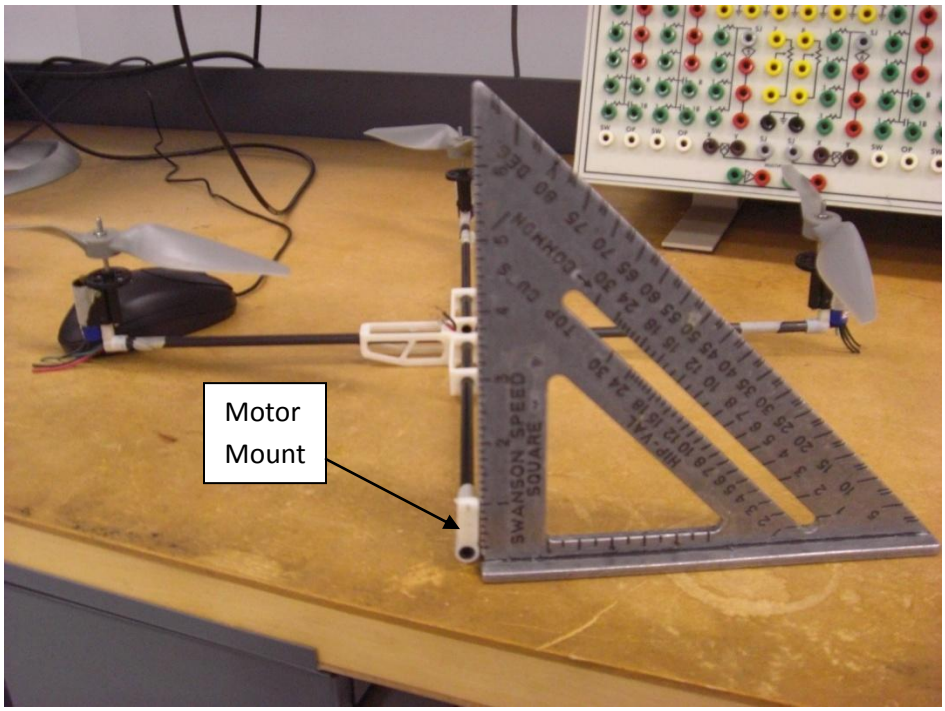


Figure 11- Motor Alignment

5.5 Propeller Balancing

Due to the fact that the propellers are operating at such a high RPM, it is crucial to the quad rotors performance that the propellers are balanced. Although theoretically the propellers are designed to have symmetric blades, in reality there are slight imperfections. These imperfections cause the propellers to vibrate uncontrollably, making smooth flight almost impossible. By balancing the propellers, vibrations can be significantly reduced.

The group balanced the propellers by examining the relative weight distribution of each propeller. There are several types of commercially available propeller balancing devices, but a similarly effective device was made. The method used consisted of attaching the propeller to a spindle held up between two blocks, allowing the propellers to rotate freely. If one of the blades is heavier than the other, then the propeller will rotate towards the heavier blade. In such a case,

the blades trailing edge was sanded down to compensate for the mass difference. Once the blades weights were evenly distributed, the propeller balanced horizontally.

5.6 Quad Rotor Balancing

The quad rotor, which includes the hub, spars, and motor mounts, was carefully designed and assembled with the idea of having a helicopter which is as close to perfect in terms of its weight distribution throughout the entire aircraft. As previously mentioned, achieving a steady and controllable flight is almost impossible to achieve unless the aircraft is as close to perfect balance. The addition of the electronic components and battery onto the frame of the quad rotor leaves its weight distribution inconsistent throughout. As a result, the group has devised a way to make sure that the frame is as balanced as possible.

The way to do this is by attaching a piece of string to the center of the hub and allowing the quad rotor to float freely. If the weight is not evenly distributed, then the quad rotor will lean towards the heaviest part. The battery packs, which are mounted below the central hub by rubber bands, were then adjusted to ensure the quadrotor was balanced in the x and y axes.

5.7 Wake Interaction

The wake interaction of an aircraft is the turbulence, which forms from the movement of air caused by the rotation of blades, as seen in Figure 10**.** The wake generated by a helicopter can be much greater that those formed from fixed wing aircraft. The wake created can cause uncontrollable aircraft movements. Because our device is propelled by four separate motors there will be a greater amount of wake produced interacting on the aircraft frame which may have an impact on how the aircraft is controlled. The wake created may cause an increase in vibration, which will hinder the overall performance.

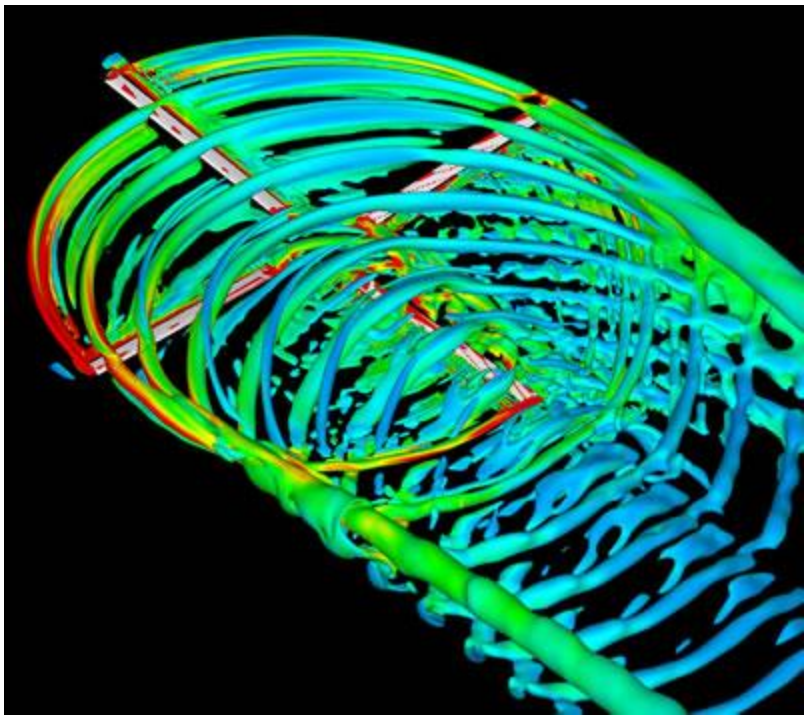


Figure 12-Wake Generation -©http://www.onera.fr/photos-en/simulations/images/helicopter-pale-wake-interaction.jpg

An elementary study on blade wake shows that the wake flows in a funnel shape directly downward from the blades, thus there is very little propwash directed radially from the propeller. The minimum distance between propeller tips is 1", which is more than enough to avoid interference.

5.8 Vibration Analysis

Ever since the discovery of resonance by Galileo Galilee in the 17th Century, the study of vibrations of mechanics has become a vital part of the design of any mechanical system. Because mechanical systems tend to have natural modes, when a certain force is applied any of the natural modes can be excited which in turn may lead to catastrophic failure of the system. This in turn leads to the importance of studying the resonance frequency of the quad rotor. The resonance frequencies of a system are the frequencies at which the system will be "excited"; therefore, it is imperative to determine the correct resonant frequencies of the quad rotor in order

to ensure that the natural modes of the system will not be disturbed. To do these we have determined the natural frequencies of the propellers, which we then relate to the resonance frequencies of the quad rotor to ensure the stability of the system.

5.8.1 Analysis and Results

To determine the natural frequencies of the quad rotor the following Equation 6.1 for the Strouhal number was used:

$$St = (f * L)/v \tag{5.1}$$

Where *St* is the Strouhal number, *f* is the vortex shedding frequency of the propeller, and *v* is the velocity of the flow past the propeller. The Strouhal number is an experimentally determined quantity derived in wind tunnel testing and a quantity of 0.2 is acceptable in our project. The velocity of the flow, v, is determined analytical by multiplying the angular velocity of the propeller in RPM's with the radius of the propeller. The characteristic length, *L*, is determined analytically as well by determining the thickness of the propeller as well as the chord length multiplied by the sine of the angle alpha, which is the angle of attack. The larger quantity is then substituted for the characteristic length. As a result, we were able to determine the vortex shedding frequency of the propeller. The following are values for the two bladed experimentally tested propellers.

Table 3-Empircal Data for 2-Blade Propellers

# of blades	L (m)	½ Radius (m)	Radius (m)
2	0.017	0.056	0.112

Using the values presented above the vortex shedding frequencies, f, were determined for values of omega ranging from 0 to 40000, which is the maximum angular velocity of the propellers. The following graph represents the calculated values of vortex shedding frequency as a function of ω for the two bladed propeller.

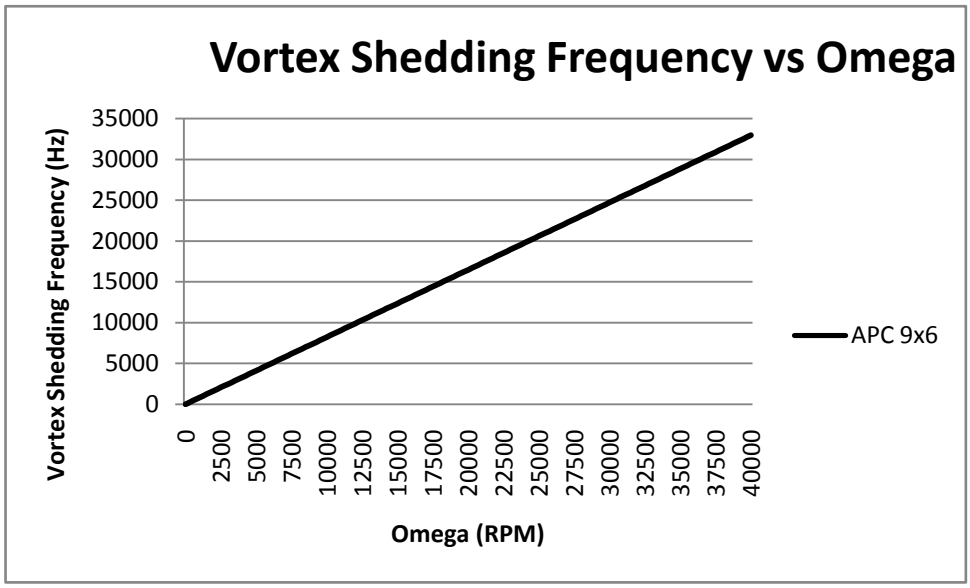


Figure 13-Vortex Shedding Frequency vs. Angular Velocity

As presented in the graph, there is a strong linear correlation between the increase in vortex shedding frequency and the increase in omega. The values for the vortex shedding frequency are an average for each propeller as the speed of the flow past the propeller was calculated at one-half the total radius of the propeller.

These results are significant when compared to the analysis of determining what the natural resonance frequency of the structure is completed. To determine the natural resonance frequency of the structure we assumed a rigid body with the carbon fiber spars representing beams with one free end and one fixed end. From this assumption, we determined what the effects on the spar will be when there was a force applied (the force of the propeller). Below is a

table representing the natural modes of a cantilevered rod and the corresponding natural frequency of the carbon fiber rod of our quad rotor.

Table 4-Natural Modes of Cantilevered Rod Compared to Natural Frequency of Carbon Fiber Rod

knL	omega n	freq Hz
1.8751	0.845615372	0.134652129
4.6941	5.299424099	0.843857341
7.854757	14.83848321	2.362815798
10.99554	29.07750792	4.63017642
14.13717	48.0672017	7.654013009
17.27876	71.80407572	11.43377002

To determine the natural frequency of the carbon fiber rod the following equation is used [21].

$$frequency\ (Hz) = \ (KnL^4 * E * I)/(2 * \pi * \rho * A) \qquad (5.2)$$

where E is the modulus of elasticity and I is the second moment of area which are defined by the properties of carbon fiber. A is the area of the carbon fiber, while ρ is the density. In our analysis, we determined that for the natural resonance frequencies shown above, at no time does the vortex shedding frequency of the propellers excite any of the modes. This leads us to conclude that barring any resonance within the electronics our quad rotor is structurally sound.

5.9 Rotor Theory

Unlike other helicopters which require complex mechanical rotor mechanisms to control pitch of the blades, a quad-rotor relies solely on differential torque and thrust, and thus uses

fixed-pitch blades. The previous version of the quad-rotor, as well as almost all other quad-rotors that were researched, have two bladed propellers. The terms propeller and rotor are used interchangeable because the rotors used in this project are marketed as model airplane propellers, and since they are both fixed pitch are essentially the same. One of the goals of the project is to increase total thrust; increasing thrust is accomplished by increasing the velocity of the air being moved by the rotor, or by increasing the amount of air being moved by the rotor. Explicitly:

$$T = \dot{m}(u_9 - u_0) \tag{5.3}$$

Where T is thrust, $\dot{m}$ is the mass airflow, and u_9 and u_0 are the exit and incoming velocity velocities, respectively. From a simple physics perspective, using the kinetic energy equation:

$$K = .5mv^2 \tag{5.4}$$

doubling the thrust by double *m* would lead to double the energy required. Doubling the thrust from doubling differential velocity however, would require four times the energy. Thus, for this project increasing mass air flow was the primary goal when choosing propellers.

All else being equal, two primary ways to increase mass air flow is to increase the diameter of the rotors, or to add blades. Since this quad rotor is going be used indoors, the maximum diameter was limited to the width of a doorway, which is approximately 32” or 81.2cm. A 2” (5.08cm) clearance was added giving the helicopter a final maximum dimension of 28” or 71.12cm. From this dimension, the maximum rotor diameter is 10”; at 10.5” the adjacent rotors would intersect, and a 2.54cm gap between them prevents tip vortex interaction. Limited to a 10” rotor, which is 1” more than the previous quad rotor, the only option was to increase the number of blades. Several 3- and 4-bladed rotors were selected for testing. The results for thrust testing can be seen in following sections.

Table 5-Propeller Comparison Table

Propeller	Supplier	Price
GP3109070 9x7	Hobby-Lobby	$8.80
GP310070 10x7	Hobby-Lobby	$9.30
GP310083 10x8.3	Hobby-Lobby	$9.40
FS400 9x7 4blade	Hobby-Lobby	$5.00
MA0970T	Hobby-Lobby	$7.29
FS300 9x7	Hobby-Lobby	$4.00

The design of propellers is a complex subject, and as there were already dozens of readily available propellers on the market, it was both cost and time effective to select a commercial blade. The propellers are designed for radio control applications, and their airfoils are highly proprietary. The standard measure of propellers is diameter x pitch, where pitch refers to the angle of incidence at ¾ of the radius. Using this angle, the pitch is converted to inches by how far the propeller would move after one revolution if it were "screwed" into a solid substance.

5.9.2 Thrust Testing Part I

Using the apparatus seen below in Figure 14, different rotor blades were tested for maximum thrust production versus power input. The test was conducted by placing the motor and the blade on one side of the balance and as the thrust increased, it had the opposite effect on the opposing side of the balance and pressed down on a scale to demonstrate the downward

force. Five different blade designs were tested with various airfoils, diameters, pitch, and number of blades.

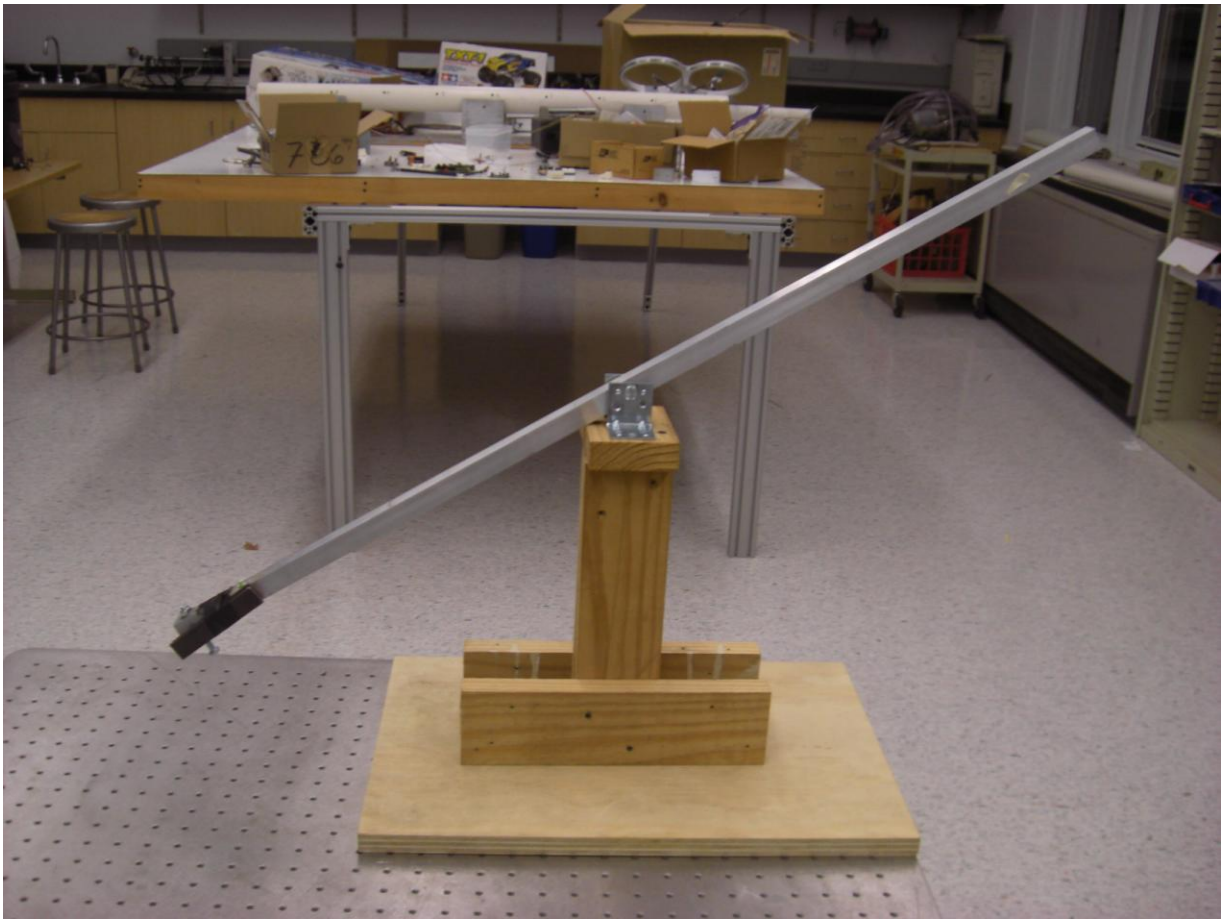


Figure 14-Thrust Testing Stand

The test was conducted for each blade at four fixed pulse widths (the ESC was in Throttle mode so RPM's differed between blades, put power input was constant) values which were 1.4, 1.5, 1.6, and 1.7ms. Each value produced a different thrust amount. Each blade was tested at the same duty cycle. Once the test was completed, the results were plotted as seem in Figure 15.

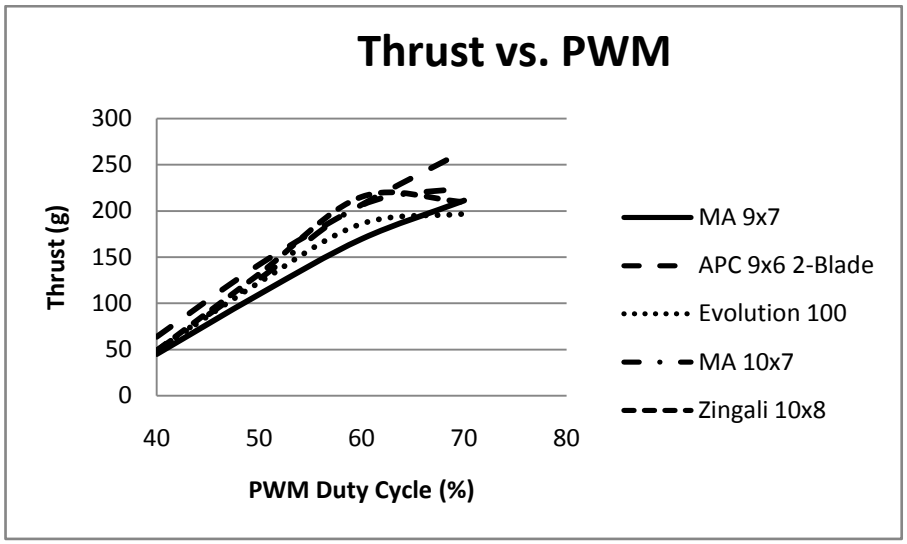


Figure 15-Thrust vs. PWM Input for Various Propellers

The thrust testing demonstrated that the APC 9x6 blade outperformed the others in terms of total thrust produced. The design of the APC 9x6 proved to have a better aerodynamic airfoil design compared to the three bladed rotors we also tested. As a result, the APC 9x6, as seen in Figure 16 is the blade of choice for our quad-rotor.

Figure 16- APC 9x6 blade © http://www.chiefaircraft.com/rcmsec/Models/ElectricAirplanes/Accessories/APCProps.html

The APC 9x6 has a far more aggressive airfoil and platform than the other propellers tested. At about 1/8 span there is an extremely high angle of incidence increase and a large chord increase. The chord reaches its maximum at about 1/3 span, and tapers off towards the tip. At the tip there is a lot of washout, allowing for a higher tip speed and the leading edge is rounded, presumably to reduce vortex generation and reduce drag.

In order to determine the power draw from the motor for various level of thrust, the PWM was adjusted until the desired thrust was measured on the scale. For a comparison the second best performing propeller (the Master Airscrew 10x7) was measured as well.

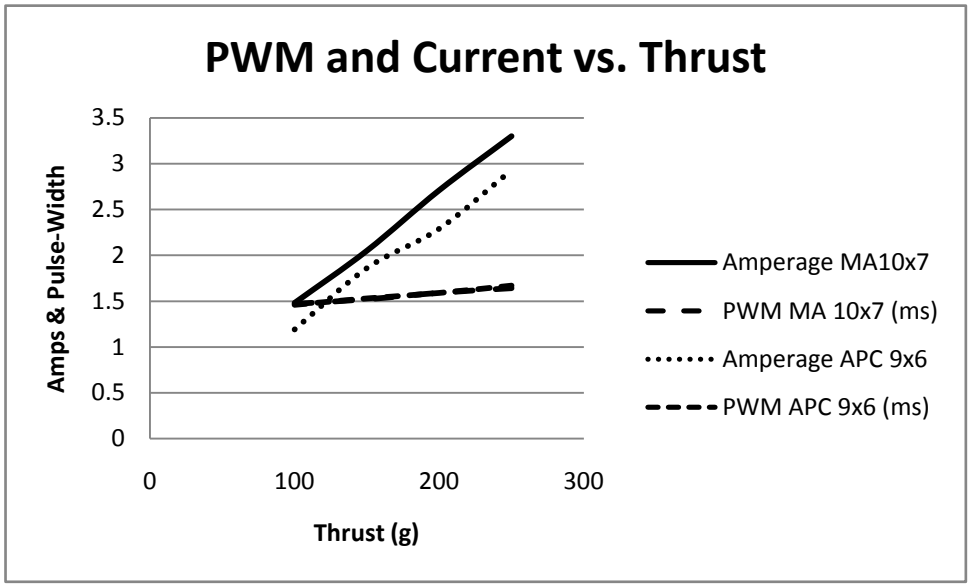


Figure 11-Current and Thrust for MA 10x7 and APC 9x6

In Figure 16 it can be seen that the APC propeller drew about 250mA less for a given amount of thrust, thus confirming that it was clearly the best option.

5.9.3 Thrust and Torque Testing Part II

Once we decided upon the APC 9x6 propellers, we conducted further test in order to determine what discrepancies existed between the four different motors and propellers. The following table represents the data we collected for the thrust testing of the different motors.

Table 6-Thrust Curves for Each Motor/Gearbox Combination

	Motor 1 (g)	Motor 2 (g)	Motor 3 (g)	Motor 4 (g)
Initial Mass	2074	2082	2078	2070
PWM				
1.1	68	72	72.6	67.9
1.2	84	90	89.2	84.2
1.3	108.4	114.6	113.3	105.2
1.4	146.5	156.6	150	134.4
Regression line	y = 2.5496x + 0.3605	y = 2.7311x + 0.3796	y = 2.5143x + 0.414	y = 2.1631x + 0.4199

The test was completed with a thrust scale as well as a signal generator which provided the PWM signals to the speed controllers. For each motor, at each interval the difference between the initial mass of the motor and the reading on the scale was recorded and represents the mass that motor could lift at that given PWM signal. This information was plotted to obtain the regression line (Figure 18) which relates thrust to PWM; unfortunately this curve isn't linear so a close linear approximation was used. Notice that the pulse-width ranges from 0-.5ms, this is because normally between 0-1ms the motor is off, and so to include this range in the regression equation would have resulted in the thrust curves being offset. Notice also that the measured mass was converted to Newtons, which are the correct units of force instead of grams.

Next, the torque produced by one motor was tested. All of the motors were removed except for 1, and the helicopter was held perpendicular to the scale. The torque vs. PWM is shown in Figure 17.

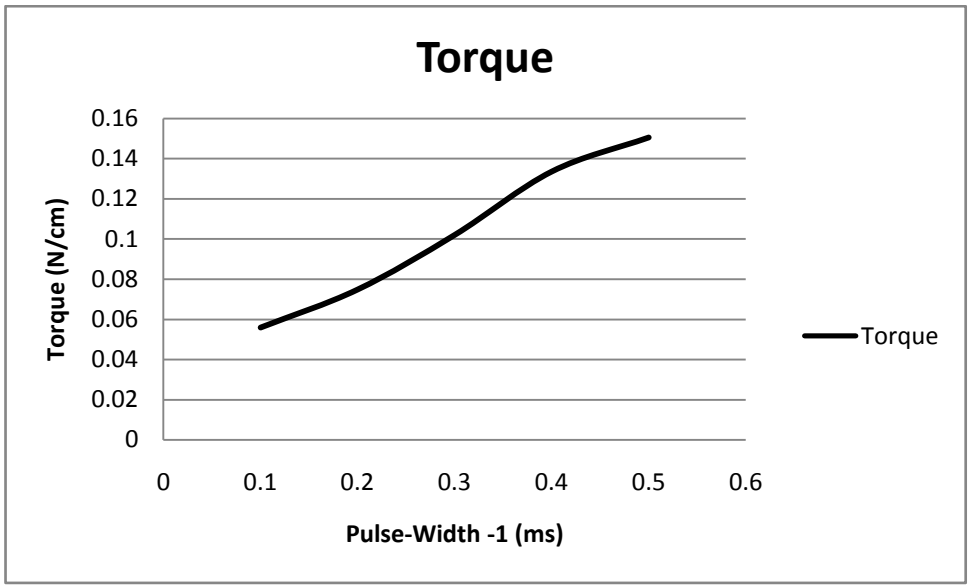


Figure 17-Torque vs. Pulse-Width

From these two graphs, we can find out $c = 1.83$ for Eq. (4.20), where c is the ratio of torque to thrust.

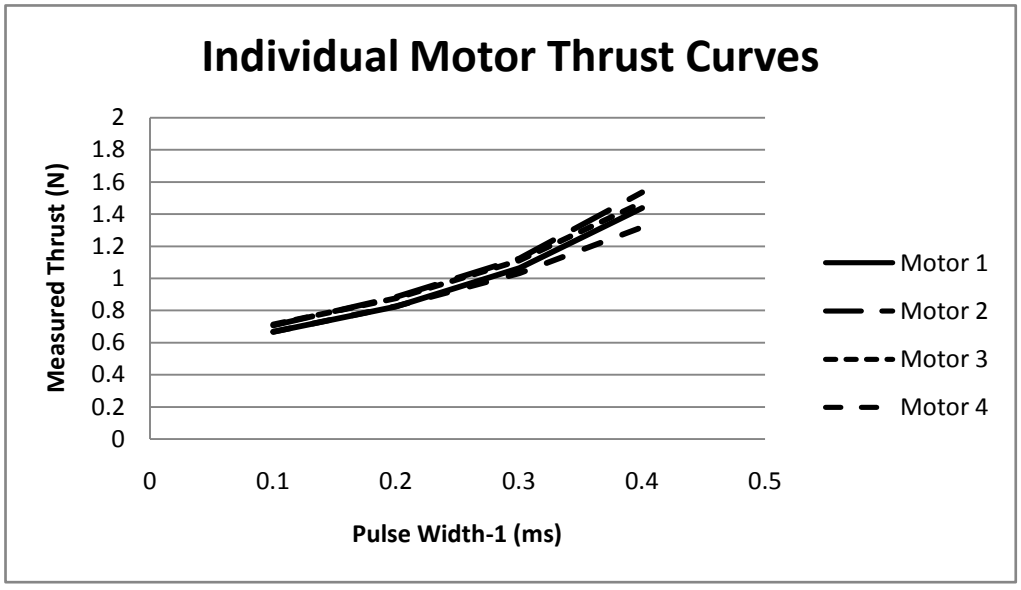


Figure 12-Motor Thrust Curves

6. Electronic Control System

6.1 Overview

The electronic controller required for the quad rotor helicopter had to be a very fast computational machine that would get all the sensor values, calculate initial motion data and produce an output to vary the speed of each individual motors to acquire desired motion. In order to achieve the computational requirements the MSP430F149 microprocessor was chosen. It is fast, has low power requirements and has a large enough code space.

6.2 Battery Selection

In determining what battery would best suite our needs for this project, we had to take many factors into account. These included, but were not limited to our total flight time and the total current draw from our four motors. The four motors which we have used in this project are the MR-012-030-4000 (as seen in subsequent section) motors from Medusa Research. Each motor weighs approximately 15 grams and has a max current draw of six amps. Multiplying that by four, we get a total current draw of 24 amps from our four motors. To have a substantial amount of flight time while keeping the total mass of the quad rotor to a minimum, we decided on the Thunder Power 3 cell 11.1 volt 2000 mAh battery from Dragonfly Innovations as shown below. To determine the total flight time you must divide the battery's power rating by the current draw of the four motors and multiplying by sixty.

Figure 13-Dragonfly Innovations LiPoly Battery © http://www.rctoys.com/rc-products/TP-2000-3SPL.html

$$tof(min) = \frac{battery\ power\ rating\ (AmpHr)}{current\ draw\ motors\ (Amp)} * 60\ min \tag{6.1}$$

This formula allows us to determine that our total flight time is approximately 8 minutes 34 seconds. One more important parameter that was taken into account is what is known as the current rating "C". The battery we have chosen has a continuous current rating of 16 C. This current rating allows us to determine the maximum discharge rate of the battery. To determine this you multiply the current rating by the total power rating of the battery.

$$Discharge\ rate\ (mAh)\ = \tag{6.2}$$

$$current\ rating\ *\ battery\ power\ rating\ (mAh)$$

The maximum discharge rate of the battery is 3200 mAh. However, since we are drawing only 1400 mAh, we are within the limits of our battery's specifications. Figure 20 is a manufacturer's diagram of what the discharge curve would be for our battery.

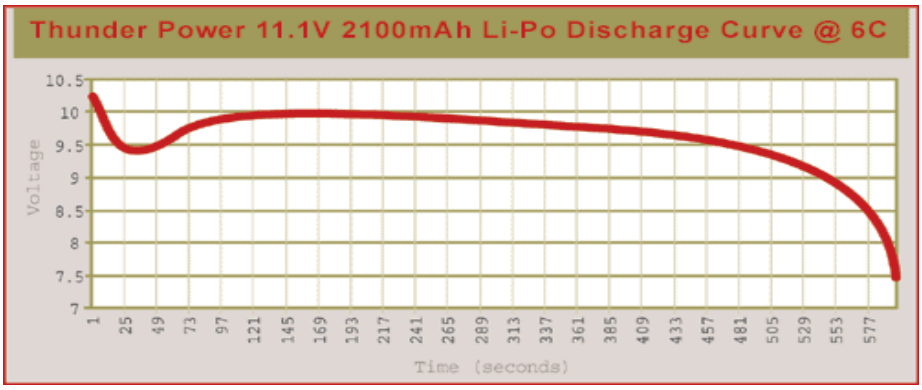


Figure 14-Battery Pack Discharge Curve © www.rchobbies.org/lithium_battery_breakthrough.htm

6.3 Motor and Speed Controller

Medusa Research MR-012-030-4000 brushless DC motors were chosen for the quad rotor. They were selected for their price, power output, and weight. Using this motor with the aforementioned APC propeller produces about 1.95N thrust, which is plenty considering an operating weight of the helicopter to be 4.22N. The motors have a max current draw of 6A, and at hovering thrust the total current draw should be 14A with a maximum draw of 24A.

The DC motors are brushless, meaning an electronic speed control (ESC) is required to control the motors; as compared to a brushed motor where a varying DC voltage could adjust the speed. The speed controller used in the Phoenix-10 from Castle Creations. This is a very complex device which performs many functions besides varying thrust. The ESC takes in a 50Hz Pulse-Width Modulated (PWM) signal from the controller and by varying the pulse-width from 1ms to 2ms determines motor RPM. The ESC has three outputs, one going to each pole in the motor. It fires each pole in succession as the magnets move around the armature; the length that the pole is "on" for determines RPM of the motor.

6.4 Sensors

6.4.1 Sharp GP2D12 Infrared Rangefinder

The infrared rangefinder has an effective range of 10cm to 80cm with an accuracy of +/-.3mm. The sensor's analog output is a nonlinear function which much be linearized in order to be useful in the control system. The user manual recommends a using

$$V = \frac{1}{(R + k)} \tag{6.3}$$

Where V is the voltage, R is the distance in meters and k is the calibration constant, which according to Sharp should be around .42. Test data showed this number to not be accurate, and a new linearized function was created in the form of $y=mx+b$, where $y=V$. To find the constants, different values of k were plugged into Excel and a linear regression curve was plotted. The value of k with the highest correlation factor R^2 was used, and the constants m and b were taken from the regression equation. The final range (z position) equation is:

$$z = \frac{1}{.7569V + .3371} - .25 \tag{6.4}$$

6.4.2 1DOF Sparkfun Gyro SEN-00394 (ADXRS300 chip)

The SEN-00394 breakout board contains the ADRXS401 chip with all the necessary circuitry to be used for angular rate measurement. ADXRS401 is a complete angular rate sensor (gyroscope) that uses Analog Devices surface-micromachining process to make a functionally complete and low cost angular rate sensor integrated with all of the required electronics on one chip. The manufacturing technique for this device is the same high volume BIMOS process used for high reliability automotive airbag accelerometers. The output signal, RATEOUT is a voltage proportional to angular rate about the axis normal to the top surface of the package.

The results are delivered as 5mV every degree per second. (i.e. for 3^o per second, the output will be about 15 milli volts).

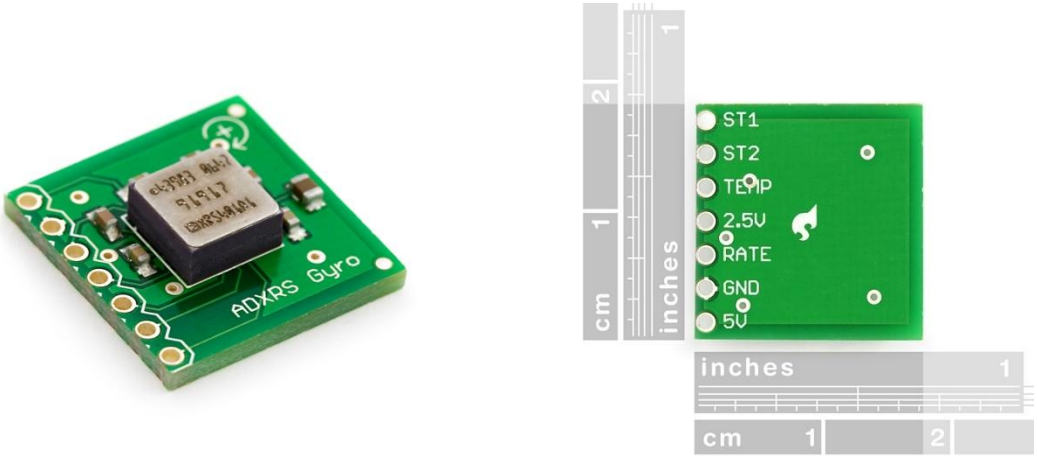


Figure 15-ADRXS401 Chip and Pin Diagram

The ADRXS401 chip has a strict requirement of 5 volts with a tolerance of ±.25V whereas our power supply is a 6V battery. Hence it is require to use a voltage regulator circuit to ensure reliability.

6.4.3 Accelerometers

The accelerometer used is the dual-axis iMEMS ADXL203. It has a range of +/- 1.7*g* (where *g* is acceleration due to gravity) with a sensitivity of 1000mV/*g*. It takes an input voltage V_s of 5V, with a nominal output of 2.5V. It has a low noise output, at 4kHz the output noise is 1mV RMS. The resolution is 1/1000 *g* at 60Hz.

Assuming small angles on the quadrotor ($0 \leq |\theta| \leq 5^o$) typical outputs will be between 2412mV and 2587mV, although when horizontal acceleration is taken into account these numbers can become larger.

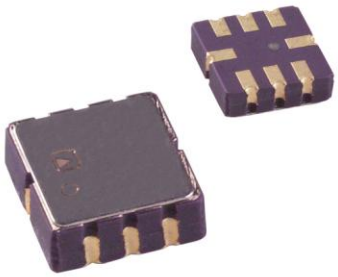


Figure 16-ADXL203CE Dual Axis iMEMS Accelerometer © Digi-Key Electronics

6.5 C Code Overview

6.5.1 The "main()" Funtion:

The Control system was designed with a single microprocessor chip. The goal was simple: calculate acceleration, angular rate, velocity, and implement control equations to generate required PWM. To instruct the Microprocessor the program was written in C code. The code comprised of five functions:

- void Time_count();
- void Init_PWM();
- void Setup_Sensors(void);
- void Compute_motion_data();
- void Calc_PWM();
- void Set_PWM();

In the main loop the two funtions: Void_Init_PWM and void Setup_Sensors, are called at first. The Void_Init_PWM function configure the timerB registers and set the Pheonix controllers to stand by mode. Void Setup_Sensors setup the ADC12 registers to sample voltages from five inputs. The SHTx bit is used for extended sampling time. To account for any stray capacitance in the wires and on the PCB the sampling time is set to a very high value .The main part of the program is the while loop within the main loop. In the while loop are the functions Calc_PWM(),

Compute_motion_data(), Set_PWM(). Every time the while loop is executed the **ADC12CTL0 |= ADC12SC;** instruction starts the ADC conversion and all the functions are called and finally the Set_PWM() function configures the TimerB CCR0 registers to acquire the desired PWM. The variable "time" calculates the time it takes to execute the loop. The average value of time is 9ms which means that the program is executed 100 times each second and the worst case "time" value was measured 11ms.

6.6 Testing the Phoenix Controllers

The PWM input to the Phoenix controllers needed to start the controllers is 1.1ms . Below this range, the controller is in stand-by mode and the power to motor is cut off. Before the Phoenix controller powers up, the PWM signal should be kept less than the cutoff signal. The Phoenix controller first checks the signal, if the signal is below the cutoff signal, it goes to standby mode, and awaits the pulse width to increase. If the signal is greater than the cutoff signal, the Phoenix controller shows an error by repeatedly flashing an LED unless the signal is set to below cutoff width. Therefore the PWM signal should be increased once the Phoenix controller powers up and verifies that the input signal is within the range.

The PWM signal was generated with Tektronix AFG 3021 signal generator. The signal was varied between 1.2ms to 2ms wide pulses of amplitude 5V. At approximately 1.7 ms pulse, the servos are at full throttle and increasing it anymore causes the Phoenix controllers to shut down the motors.

For the motor control we cannot just assume the RPM of the rotors by the pulse width. For the same PWM each motor was measured for thrust and was found to have a slightly

different speed. Their starting frequency seemed to vary as well. Therefore for accurate control an additional device for measuring RPM is required.

6.76 Using the ADC to measure sensor values

6.7.1 Configuring the ADC12

The MSP430 provides 16 ADC inputs that can be sampled and stored in the ADC memory registers. Once the conversion is done the ADC12 IFGx bits are set, an automatic interrupt is generated and within the ISR, a code is written so that the corresponding inputs are stored in the variables A0results to A6results. Once the data is moved from the ADC12memory registers, the IFG x bits are automatically cleared.

The MSP430 ADC supports four operation modes. For our purpose the multiple channel single conversion is selected. The reference voltage is set to +AVCC which is +3V. Conversion and IFG bits are enabled. The ISR copies the values form ADC12MEM registers to the variables A0results to A6results and clears the IFG bits.

The sampling time can be varied by the SHT bit.

tsample = 4 □ □ tADC12CLK□ □ □

SHT0	0	1	2	3	4	5	6	7	8	9	10	11	12–15
n	1	2	4	8	16	24	32	48	64	96	128	192	256

SHT10 was chosen to account for any stray capacitance in the wires and on the PCB. This is to ensure that enough time is provided for the input voltage to stabilize before it is written to the ADC12MEMx registers.

6.7.2 Computing motion

The values obtained from ADCMEM registers have to used to first calculate the raw sensor inputs and then the actual motion data**.**

A0 results= **4096 * sensor voltage/3000mv**

Hence the sensor input voltage in millivolts is: (**A0results*3000mv/4096**)

This equation gave us the raw input voltage. To calculate the actual motion data (i.e acceleration, angle etc) any offset voltage is subtracted from the result and then divided by the rate from the spec sheets.

For example:

Acceleration= (Raw voltage- Offset)/ 200mv/deg/sec

Though this is the concept, when the actual code was written, some of the formula was simplified to reduce execution time. For example, instead of writing A0 results= **4096 * sensor voltage/3000** the code was written as A0 results= **1.365 * sensor voltage.**

Using these concept each sensor input was converted into motion data which were used by the function Calc_PWM to implement equations of motion and produce 4 integer values that would control the PWM signals.

6.8 Using TimerB to generate PWM

The TimerB of the MSP430 has 7 capture compare modes. The code takes advantage of the TimerB to produce 4 different PWM pulses of the same frequency. At startup the pulse width required was 1ms. Then the pulse width has to be increased to 1.5 ms in order to start the motors. The internal clock frequency is set to 1048576 Hz by the TBSSEL_2 command. The CCR0 value sets the frequency for the PWM signal. The servos require a 50Hz PWM in order to function.

CCR0 = clock frequency / required frequency

= 1048576Hz/50Hz = 20972

The corresponding CCRx values (CCR1-CCR5) for the positive width of the PWM signal was calculated by the equation:

CCRx= High time*1048576

The CCRx values can be varied from 1049 to 2097 to represent a PWM signal with a +ve pulse width of 1ms to 2ms.

Each of the capture compare register is set to a variable PWM1, PWM2, PWM3, and PWM4. For the first few seconds the pulse width is set to approximately 1ms. The swDelay() function is a simple loop that produces a delay before the PWM is increased.

The **Set_PWM()** function simply sets the TBCCRx registers to the required PWM values calculated by the Calc_PWM() function.

6.9 The Time_count() function

The Time_count function is used to configure timerA to run in upmode producing an interrupt request every millisecond. The ISR simply increments the variable “time” which is used to measure the time difference between each ADC sample. The time count is in milliseconds.

TACCR0 value= SMCLK frequency/1000Hz

TACCR0= 800,000Hz/1000Hz = 800

SMCLK was selected for TimerA as it is synchronized with the CPU clock. Otherwise the results could be unpredicted.

6.10 Power supply consideration

The ADXL203 can be powered with VS as low as 4 V or as high as 6 V. However the output is ratiometric, therefore, the output sensitivity (or scale factor) varies proportionally to the supply voltage. At VS = 5 V, the output sensitivity is typically 1000 mV/*g*.

At V_S = 4.75 V, the output sensitivity is typically 187mV/V/*g*, and at V_s=5.25V the sensitivity is 215mV/V/*g*. The zero *g* bias output is also ratiometric, so the zero *g* output is nominally equal to VS/2 at all supply voltages.

The **IDG-300** Rate-Out is not ratiometric to the supply voltage. The scale factor is calibrated at the factory and is nominally independent of supply voltage.

No additional capacitors or resistors were added as the required filtering circuitry is already integrated on the PCB.

6.11 1DOF Sparkfun Gyro SEN-00394 (ADXRS300 chip)

The SEN-00394 breakout board contains the ADRXS401 chip with all the necessary circuitry to be used for angular rate measurement. ADXRS401 is a complete angular rate sensor (gyroscope) that uses Analog Devices surface-micromachining process to make a functionally complete and low cost angular rate sensor integrated with all of the required electronics on one chip. The manufacturing technique for this device is the same high volume BIMOS process used for high reliability automotive airbag accelerometers. The output signal, RATEOUT is a voltage proportional to angular rate about the axis normal to the top surface of the package.

The results are delivered as 5mV every degree per second. (i.e. for 3 _per second, the output will be about 15 milli volts).

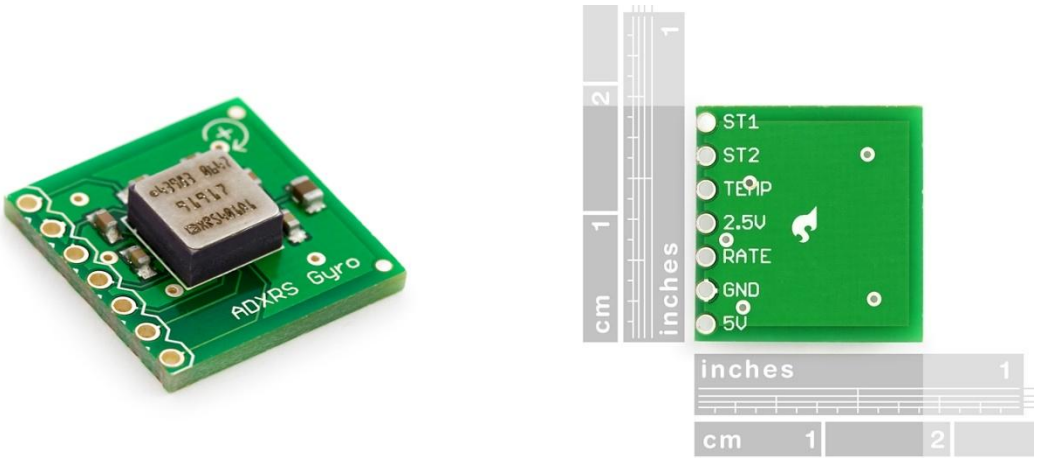


Figure 17-ADRXS401 Chip and Pin Diagram

The ADRXS401 chip has a strict requirement of 5 volts with a tolerance of ±.25V whereas our power supply is a 6V battery. Hence it is required to use a voltage regulator circuit to ensure reliability.

6.12 Connecting the ADRXS401 with the MSP430F149:

The ST1 and ST2 inputs can be for automated self test. However these features were not used for the project and the inputs were connected to ground. The RATE output has a range of .5v to 4.5 volts and had to be scaled with a voltage divider circuit and connected to the ADC input A2. The ground was connected to both the power supply ground and the ground connection on the MSP430 board. Once the sensors were connected to the board, they were calibrated using actual readings taken with a DMM and tested to ensure accuracy.

7. Results

7.1 Mechanical Design Results

The goal of 10% weight decrease was easily surpassed through the use of the ABS rapid prototype frame. The entire quad rotor, including controller and battery packs, has a mass of 435g. The previous quad rotor had a mass of about 650g, which is about 66% more. The ABS frame wasn't a complete success however; the "layering" of the rapid prototype machine caused delamination in both the hub and the motor mounts. 3-D printed parts have problems with thin sections, and the truss in the hub and the tube where the motor mounts slide over the spars proved troublesome. As can be seen in Figure 23.a and 23.b, the printer left noticeable gaps in crucial structural areas; needless to say this caused numerous parts to be broken accidentally as the motor mounts were screwed on and various tests were performed.

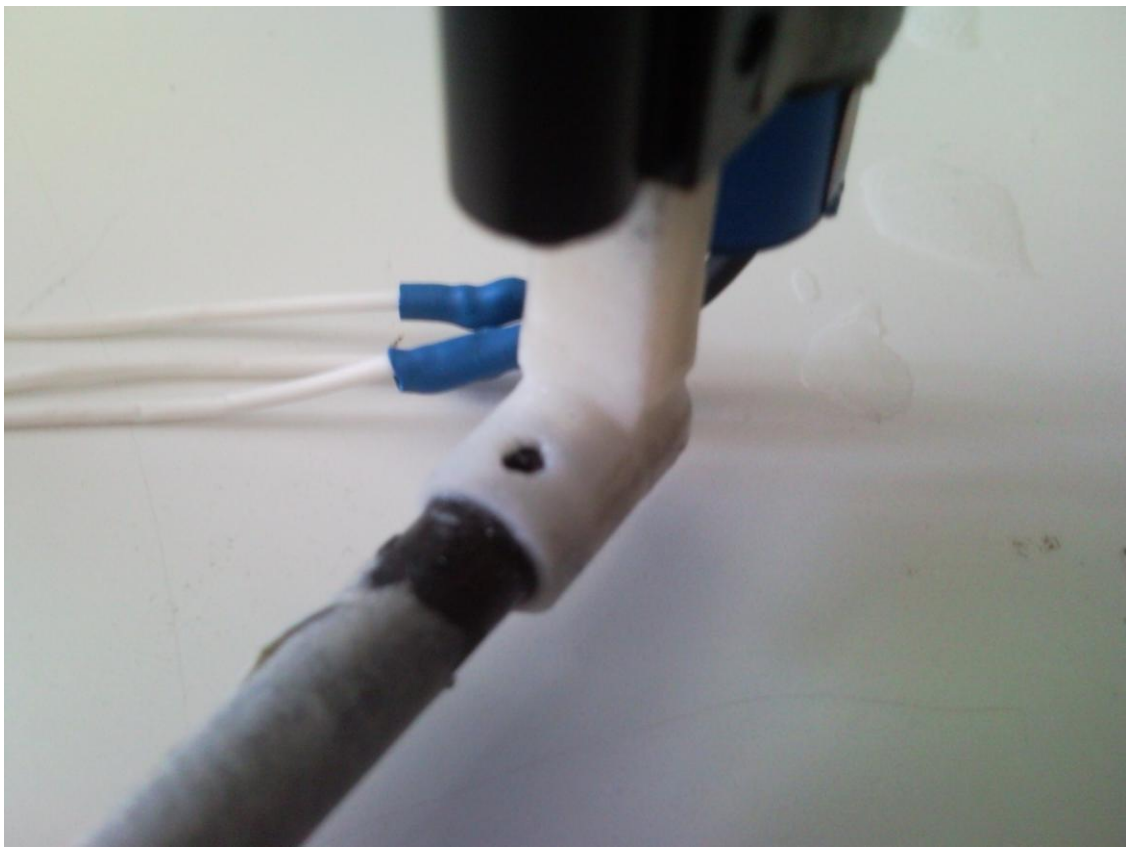


Figure 18.a-Gaps in FDM Modeled Motor Mount

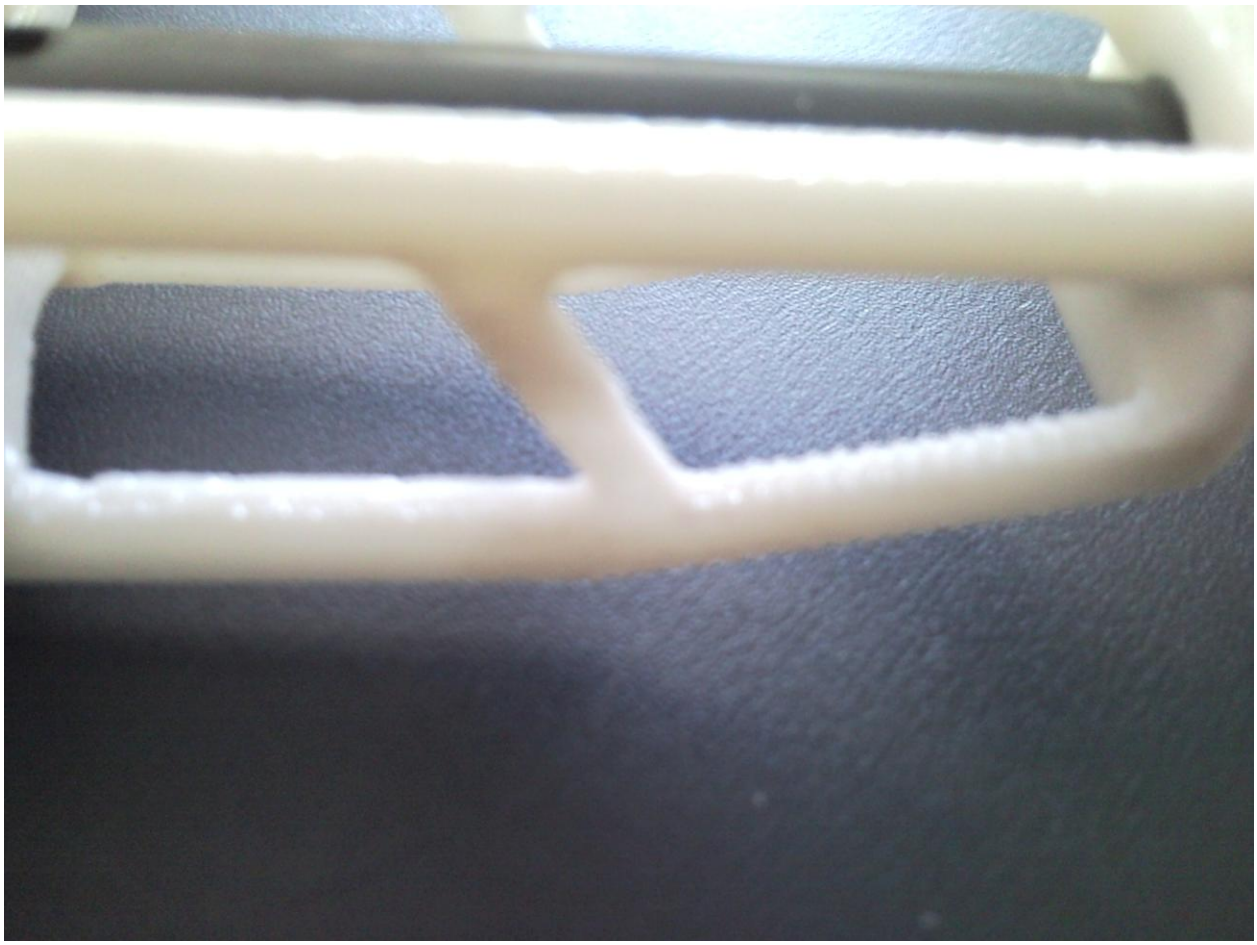


Figure 23.b Gaps and "Stepped Edges" on FDM Central Hub

Screwing the spars to the hub and motor mounts also proved extremely difficult. The screw holes had to be drilled perfectly into the spars to prevent motor misalignment; this was near impossible on the drill press so instead cyanoacrylate glue was used to successfully secure everything together and ensure all necessary alignments were made.

The motors also proved to be very problematic. The leads on the motors are four strands of fine gauge wire which are very fragile. After soldering, resoldering, and constant manipulation

of the leads the motors would not get a clean signal and wouldn't run. The only solution was to buy new motors and have them professionally soldered. The motors were soldered to the speed controllers by the ECE shop on campus, and they also applied protective heat shrink tube over the joint. Unfortunately, this meant that the motor wires could not be routed through the spars as originally planned, because this would mean the leads would have to be resoldered every time a motor was removed for testing.

7.2 Flight Testing

7.2.1 Preflight Testing

Before the quadrotor could be tested in free flight, it had to be determined that it would not respond in a way which would damage itself. First, all of the sensors were connected to the helicopter, but not actually mounted on it. This allowed manipulation of the sensors while allowing the motors to run, but prevented the quadrotor from actually moving. This was a qualitative test simply to ensure that the motors were responding quickly to variations in attitude and altitude.

The second test calibrated the hover thrust. The helicopter was attached to a scale and thrust was increased until the thrust canceled out the weight. This number was then used as the *mg* term in Eq. (4.9).

7.2.3 Flight Attempts

The quadrotor is still going through flight testing which includes tuning of the damping constants and tweaking of the code to get the most accurate data out of the sensors. The rangefinder seems to have problems where certain distances are accurate to within a few millimeters while others can be off by more than a centimeter. Current solutions include keeping the altitude of the quadrotor within the range where the sensor is most accurate; unfortunately this means that any altitude between 10-80cm cannot be chosen and the helicopter is more limited in its flight regime.

The flight testing looks promising however and the quadrotor is expected to be fully functional by April.

8. Conclusion and Recommendations

We have successfully met our design goals and demonstrated autonomous flight. This is something the previous MQP, who had twice as many members, could not do. We would have liked to progress further, such as having the ability to fly to a destination, but we were unable to do this in just three terms. Part of the reason for this was because the group had no programming or electrical engineering experience so the control scheme was planned entirely different until the addition of Tanvir Anjum two months into the project.

As a recommendation to future MQP students, this project should be undertaken only by electrical engineering students or students with a strong controls background. This will save months of studying and research to just understand how the previous group got to where they did.

Second, and we apologize for the pessimism, but prepare for everything to go wrong. The design of the quad rotor is straightforward in theory; however parts will break, won't arrive on time, or simply will refuse to run. The more planning and testing done early on will save numerous hours later fixing problems that could have been prevented.

Finally, we'd like to recommend a few key steps to save time for future MQP's. The sooner these are completed, the sooner progress can be made.

1. Learn how to use a function generator to operate the motors; this is key for thrust and torque experiments and will allow the programmers to work on the code while other tests are being run. See Appendix G for details.

2. Fully understand the control equations. They can be daunting at first, but with a little study will make perfect sense.
3. Test everything as soon as possible. If you order a new component, immediately test it to not only ensure functionality but so that you understand how it works and how you must use it.
4. Use alligator clips or crimp-on pins for temporary connections. This prevents cold soldering and will save leads in the long run.

References

[1] Castillo, P., Dzul, A., Lozano, R., Modeling and Control of Mini-Flying Machines, Berlin: Springer, 2005.

[2] Wikipedia. Org, http://en.wikipedia.org/wiki/Quadrotor. Date Accessed: September 12, 2008

[3] RC Helicopter Fun, http://www.rchelicopterfun.com/rc-helicopter-material.html. Date Accessed: September 24, 2008

[4]X-UFO SHOP, http://www.xufo-shop.de. Date Accessed: September 23, 2008

[5] Bramwell's Helicopter Dynamics, Second Edition. Done, George. David Balmford. *Library of Flight Series.* © 2001

[6] Boyd, Stephen. *The Kalman Filter.* www.Stanford.edu/class/ee363/kf.pdf. Date Accessed: October 8, 2008.

[7] Hussein, Islam. Observers and the Kalman Filter. October 2008.

[8] Last Year's MQP

[9] Bouabdallah, S., Noth, A., Siegwart, R.,PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor: Autonomous Systems Laboratory. Date Accessed: December 3, 2008

[10] Modern Inertial Technology: Navigation, Guidance, and Control, Second Edition. Lawrence, Anthony. *Springer* ©1998

[11] Bucknell University. *Using ODE45*. http://www.facstaff.bucknell.edu/maneval/help211/usingODE45.pdf. Date Accessed: December 5, 2008.

[12]Control Tutorials for Matlab: PID Tutorial. http://www.engin.umich.edu/group/ctm/PID/PID.html Date Accessed: December 5, 2008

[13] Nwe, T.T., Htike, T., Mon, K.M., Naing, M.Z.,Myint, M.Y., *Application of an Inertial Navigation System to the Quad-rotor UAV using MEMS Sensors*. August 2008. WASET.org Volume 32

[14] Dupuis, M., Gibbons, J., Hobson-Dupont, M., Knight, A., Lepilov, A., Monfreda, M., Mungai, G., *Design Optimtimization of a Quad-Rotor Capable of Autonomous Flight*. Worcester Polytechnic Institute 2008

[15] Escareno, J., Salazar-Cruz, S., Lozano, R., *Embedded control of a four-rotor UAV*. American Control Conference, June 2006

[16] Ogata, Katsuhiko. Modern Control Engineering. 4th ed. Upper Saddle River: Prentice Hall, 2002.

[17] Interview with Jonathan Feldkamp, Castle Creations. December 15, 2008

[18] "Oemichen." 12 Oct. 2008 <http://http://www.aviastar.org/helicopters_eng/oemichen.php>.

[19] "De Bothezat." 12 Oct. 2008 <http://http://www.aviastar.org/helicopters_eng/bothezat.php>.

[20] “Linearizing Sharp Ranger Data” 28 Feb. 2009 <http://www.acroname.com/robotics/info/articles/irlinear/irlinear.html>

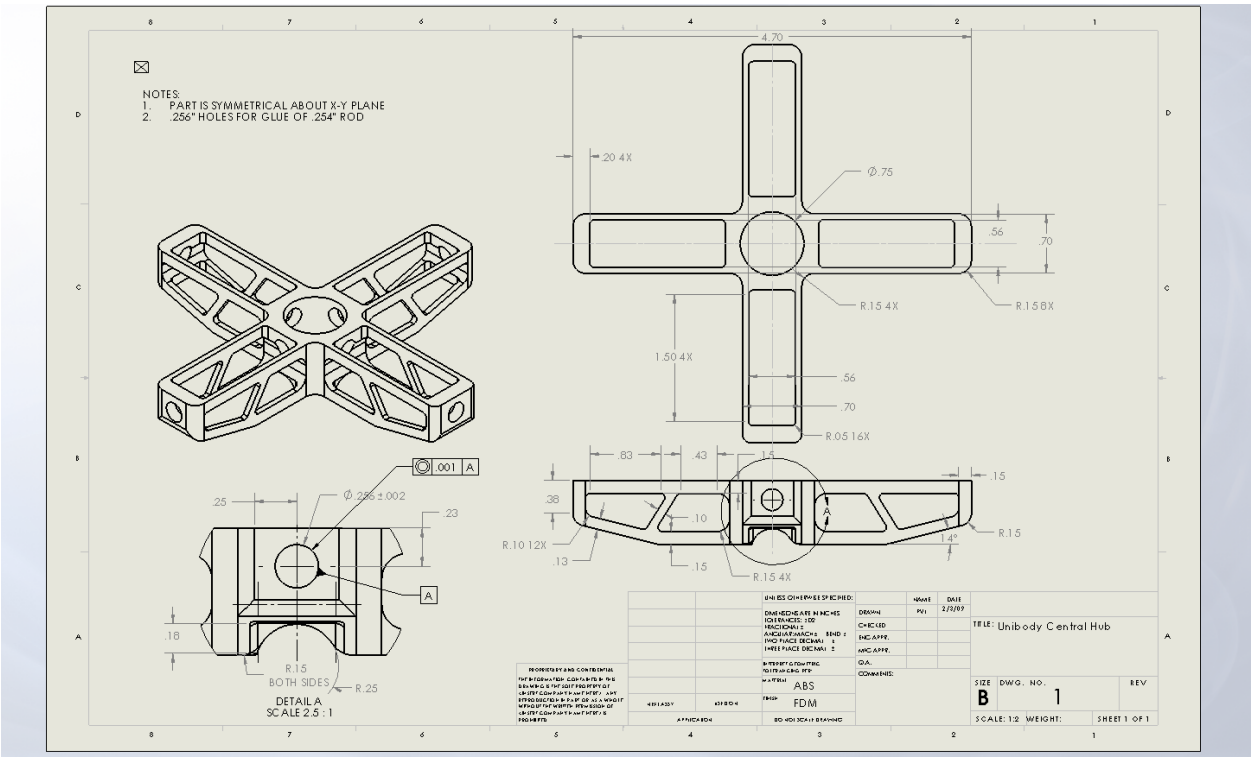
[21] “Deflection of Cantilever Beams” 21. Dec. 2008 < http://emweb.unl.edu/Mechanics-Pages/Scott-Whitney/325hweb/Beams.htm>

[22] “Convertawings Model ‘A’ Quadrotor” 14 Oct. 2008 <http://www.cradleofaviation.org/exhibits/jet/quadrotor/index.html>

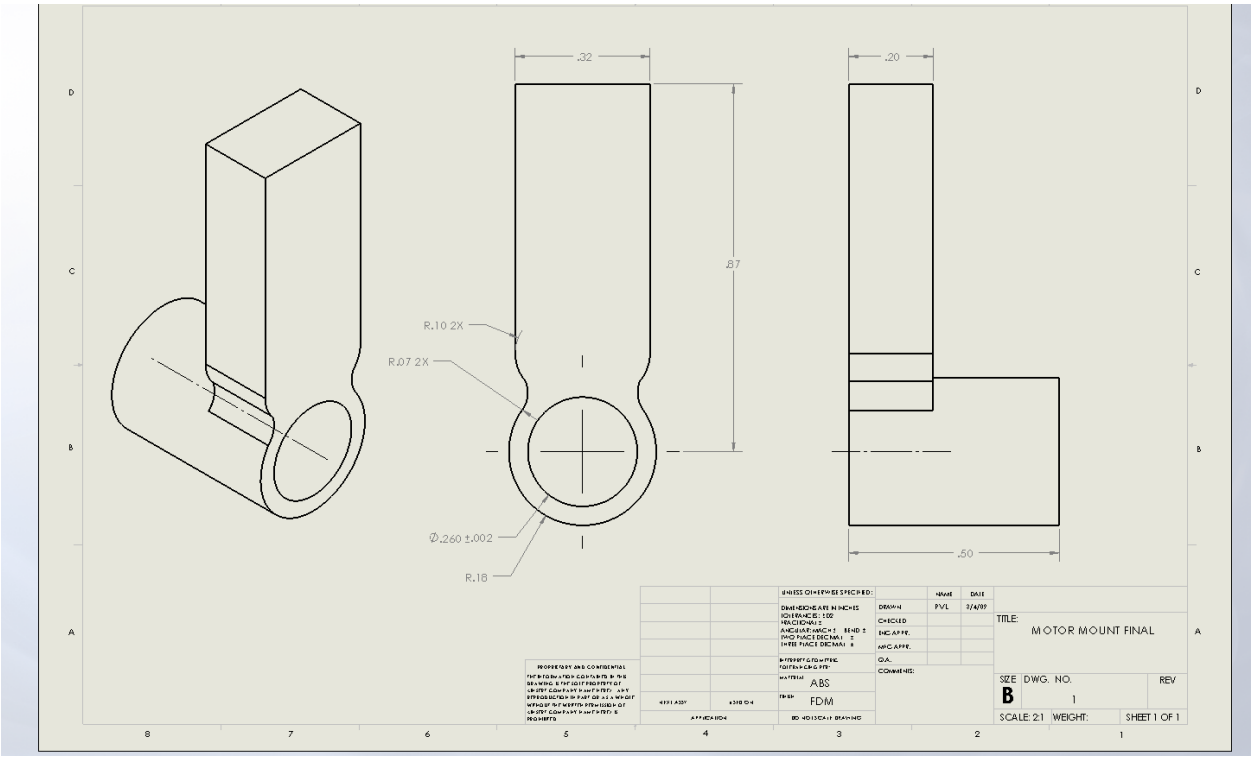
[23] “Simpson’s Rule” 18 Feb. 2009 <http://mathworld.wolfram.com/SimpsonsRule.html>

Appendix A: FDM Part Drawings

Central Hub:



Motor Mount:


.32
.20
.87
R.10 2X
R.07 2X
Ø.260 ±.002
R.18
.50
TITLE:
MOTOR MOUNT FINAL
DWG. NO.
1
REV
B
SCALE: 2:1
WEIGHT:
SHEET 1 OF 1
ABS
FDM

Appendix B: Matlab Code

PID Controller Simulation Step Function m-File

```
Kp=10;
Kd=8;
num=[Kd Kp ];
den=[2 2+Kd 2+Kp];

t=0:0.01:6;
step(num, den,t)
u=num/den;
```

MDS Simulation in Z Direction

Function m-File:

```
function pdot=mds(t,p)
%Spring Example Problem
m=2; g=-9.81; Kp=10; Kd=8;
%global u tautheta tauphi taupsi
%Parameters-damping coefficient and natural frequency
%forcing function

u=1;
tautheta=1;
tauphi=1;
taupsi=1
pdot=zeros(size(p));
pdot(1)=p(4);
pdot(2)=p(5);
pdot(3)=p(6);
pdot(4)=-u/m*sin(p(7));
pdot(5)=u/m*cos(p(7))*sin(p(8))
pdot(6)=u/m*cos(p(7))*cos(p(8))-g
pdot(7)=p(10);
pdot(8)=p(11);
pdot(9)=p(12);
pdot(10)=tautheta;
pdot(11)=tauphi;
pdot(12)=taupsi;
```

ODE Solver m- File:

```
clear; close all; clc;

dt=0.01;
```

```
tspan=[0:dt:10];
global u;
% t=1:.1:10;

p0=[0;0;0;0;0;5;0;0;0;0;0;0];

[t,p]=ode45('mds', tspan, p0);

xdot=p(:,2);
x=p(:,1);

figure
subplot(2,1,1)
plot(t,x);
xlabel('time(s)'); ylabel('displ(m)');
subplot(2,1,2)
plot(t,xdot);
```

Appendix C: PWM Generation C-Code

```
void main(void)

int a,b,c,d=1000;

{

WDTCTL = WDTPW + WDTHOLD;                 // Stop WDT

  FLL_CTL0 |= XCAP18PF;                   // Configure load caps

  P2DIR |= 0x0c;                          // P2.2 - P2.3 output

  P2SEL |= 0x0c;                          // P2.3 - P2.3 TBx options

  P3DIR |= 0xF0;                          // P3.4 - P3.7 output

  P3SEL |= 0xF0;                          // P3.4 - P3.7 TBx options

  TBCCR0 = 21000;                         // PWM Period

  TBCCTL1 = OUTMOD_7;                     // CCR1 reset/set

  TBCCR1 = a;                             // CCR1 PWM duty cycle

  TBCCTL2 = OUTMOD_7;

  TBCCR2 = b;

  TBCCTL3 = OUTMOD_7;

  TBCCR3 = c;

  TBCCTL4 = OUTMOD_7;

  TBCCR4 = d;

   TBCTL = TBSSEL_2 + MC_1;                 // Selects SMCLK clock source , sets timerB in up
mode

  swDelay(100);

a=1500;

b=1500;

c=1500;

d=1500;
```

```
_BIS_SR(CPUOFF);                        // Enter LPM0


}
void swDelay( int max_cnt)
{
unsigned int  cnt1=0, cnt2;
while (cnt1 < max_cnt)
  { cnt2 = 0;
    while (cnt2 < 6000)
      cnt2++;
    cnt1++;
  }
}
```

This C code configures the timer B to produce the range of required PWM pulses. The variables a, b , c and d can be varied independently to control the positive width of the 4 respective PWM signals.

Appendix D: The C Code

```
#include <msp430x44x.h>

#include <math.h>

void Time_count();

void swDelay(int max_cnt);

void Init_PWM();

void Setup_Sensors(void);

void Calc_PWM();

void Compute_motion_data();

void Set_PWM();

int PWM1, PWM2, PWM3, PWM4;

// These variables need to be global otherwise the compiler removes them after executing  the funtions

float A0results, A1results, A2results, A3results, A4results, A5results, A6results;

float F1,F2,F3,F4;

int count,time;

float c, l;

float t1,t2;

float aPHI1, aPHI2, atheta1, atheta2, aPSI1,aPSI2;

float PWM1, PWM2, PWM3, PWM4;

float tilt,PHIDOT,PSIDOT,thetaDOT,ZDD, Zpos;

float PHIDOT1,PHIDOT2,PSIDOT1,PSIDOT2,thetaDOT1,thetaDOT2;

float Ttheta, TPSI, TPHI,u;

float PHI,thetaDES,PSI,theta,PHIDES,PSIDES;

void main (void)

{
```

```
WDTCTL = WDTPW +WDTHOLD; // Stop WDT

PWM1=1000,PWM2=1000,PWM3=1000,PWM4=1000;        // all PWMs set to initial value.

                                                    +ve pulse width around 1 ms

Init_PWM(PWM1, PWM2, PWM3, PWM4);              //  sets the motor controller to standby

                                                    mode by increasing +ve pulse width
                                                    of the PWM from 1ms to 1.2ms.

Setup_Sensors();                                    // Configures ADC12

while(1)

{

ADC12CTL0 |= ADC12SC;              // starts coversion. reads and stores raw data from sensors

Compute_motion_data();                  // computes position, acceleration,    velocity.. etc
                                        //using raw sensor data
Calc_PWM();                             // implements control equations and calculates
                                        //the required PWM signal

Set_PWM();                              // Sets PWM to desired value

}

}
```

```
void Setup_Sensors()
```

```
{
P6SEL = 0x0F;                                   // Enable A/D channel inputs
  ADC12CTL0 = ADC12ON+MSC+SHT0_10;          // ADC12 on, extend sampling time
                                                // to avoid overflow of results
    ADC12IE = 0x08;                             // Enable ADC12IFG.3
  ADC12CTL1 = SHP+CONSEQ_1;                 // Use sampling timer, repeated seq
  ADC12MCTL0 = INCH_0;                        // ref+=AVcc, channel = A0
  ADC12MCTL1 = INCH_1;                        // ref+=AVcc, channel = A1
  ADC12MCTL2 = INCH_2;                       // ref+=AVcc, channel = A2
  ADC12MCTL3 = INCH_3;                    // ref+=AVcc, channel = A3
  ADC12MCTL5 = INCH_4;                       // ref+=AVcc, channel = A4
  ADC12MCTL6 = INCH_5;                     // ref+=AVcc, channel = A5
  ADC12MCTL7 = INCH_6+EOS;             // ref+=AVcc, channel = A6, end sequence
  ADC12CTL0 |= ENC;                         // Enable conversions
  _EINT();                                    // Enable interrupts



}




#pragma vector=ADC12_VECTOR__interrupt void ADC12ISR (void)
{
 A0results = ADC12MEM0;             // Move A0 results, IFG is cleared
```

```
 A1results = ADC12MEM1;          // Move A1 results, IFG is cleared
 A2results = ADC12MEM2;          // Move A2 results, IFG is cleared
 A3results = ADC12MEM3;          // Move A3 results, IFG is cleared
 A4results = ADC12MEM4;          // Move A3 results, IFG is cleared
 A5results = ADC12MEM5;          // Move A3 results, IFG is cleared
 A6results = ADC12MEM6;          // Move A3 results, IFG is cleared
 }
```

```
void Init_PWM()
{
 FLL_CTL0 |= XCAP18PF;                  // Configure load caps
  P2DIR |= 0x0c;                        // P2.2 - P2.3 output
```

```
  P2SEL |= 0x0c;                          // P2.3 - P2.3 TBx options

  P3DIR |= 0xF0;                         // P3.4 - P3.7 output

  P3SEL |= 0xF0;                        // P3.4 - P3.7 TBx options

  TBCCR0 = 20972;                        // PWM Period

  TBCCTL1 = OUTMOD_7;                          // CCR1 reset/set

  TBCCR1 = PWM1;                          // CCR1 PWM duty cycle

  TBCCTL2 = OUTMOD_7;

  TBCCR2 = PWM2;

  TBCCTL3 = OUTMOD_7;

  TBCCR3 = PWM3;

  TBCCTL4 = OUTMOD_7;

  TBCCR4 = PWM4;

   TBCTL = TBSSEL_2 + MC_1;                // Selects SMCLK clock source , sets timerB in up
mode


swDelay(10);


 }




void Set_PWM()

{

TBCCR1 = PWM1;
```

```
TBCCR2 = PWM2;
TBCCR3 = PWM3;
TBCCR4 = PWM4;
}

/******************** swDelay() ************************/
void swDelay( int max_cnt)
{   unsigned int  cnt1=0, cnt2;

 while (cnt1 < max_cnt)
   {
     cnt2 = 0;
     while (cnt2 < 6000)
       cnt2++;
     cnt1++;
   }
}




void Compute_motion_data()
{
  Zpos=A0results;                  //  625mv/cm
PHIDOT=A1results*1.4648;        //   2mv/deg/sec
thetaDOT=A2results*1.4648;     //   2mv/deg/sec
```

```
PSIDOT=A3results*3.662;       //   5mv/deg/sec
ZDD=A4results*219.73;        //   300 mv/9.8 m/s^2 ADXL330
   }

void Calc_PWM()
{
TPHI=-(aPHI1)*(PHIDOT)-(aPHI2)*(PHI-PHIDES);
 theta=((t2-t1))/6*(thetaDOT1+(4*((thetaDOT1+thetaDOT2)/2) + thetaDOT2));
 Ttheta= -(atheta1)*(thetaDOT)-(atheta2)*(theta-thetaDES);
 PSI= ((t2-t1)/6)*(PSIDOT1+4*(PSIDOT1+PSIDOT2)/2 + PSIDOT2);
 PHI= ((t2-t1)/6)*(PHIDOT1+4*(PHIDOT1+PHIDOT2)/2 + PHIDOT2);
 F1=.25*u+c*TPSI+0+l*TPSI;
 F2=.25*u-c*TPSI+-l*Ttheta+0;
 F3=.25*u+c*TPSI+0-l*Ttheta;
 F4=.25*u-c*TPSI+l*Ttheta+0;
}

void Time_count()
{
  CCTL0 = CCIE;                          // CCR0 interrupt enabled
  CCR0 = 800;                        // for 1ms interval
```

```
  TACTL = TASSEL_2 + MC_1;         // SMCLK, ~800KHz cont mode: up
}

// Timer A0 interrupt service routine
#pragma vector=TIMERA0_VECTOR__interrupt void Timer_A (void)
{
  time++;            // increment time
}
```

Appendix E: Sharp GP12D2 Calibration Information

Voltage as a function of distance:

$$V = 1/(R + .42)$$

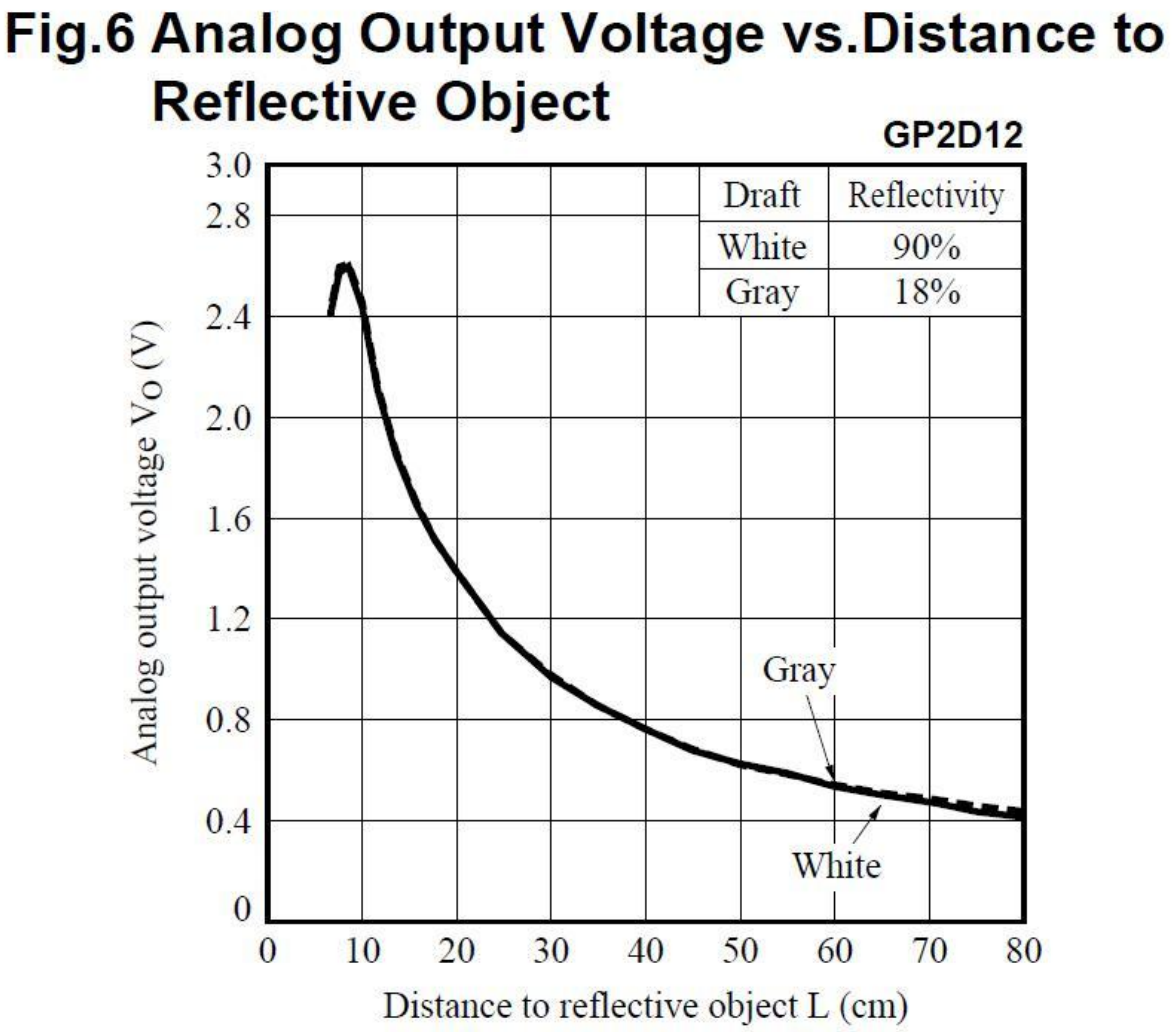


Figure 19-Analog Output Voltage vs. Distance to Reflective Object © Sharp Electronics

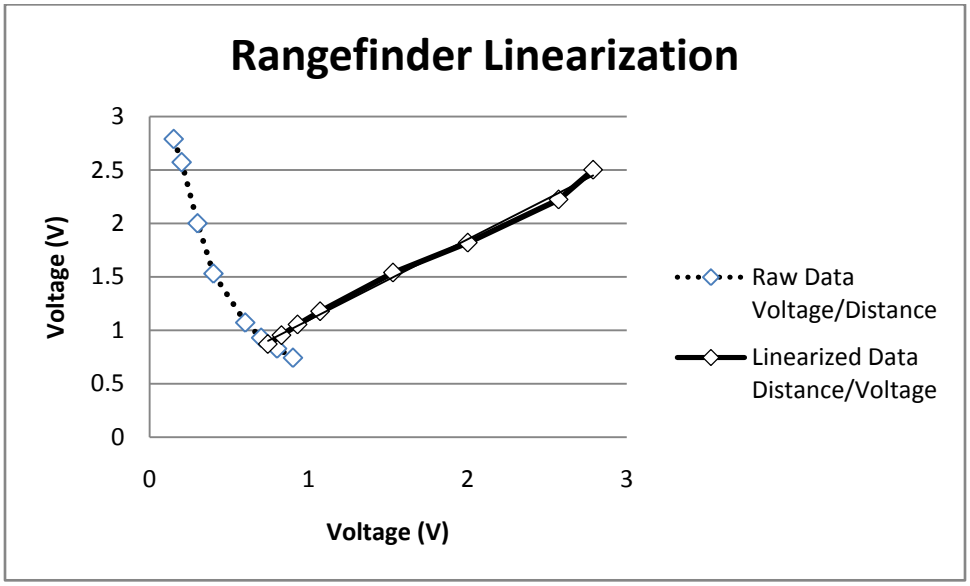
Appendix F: Function Generator Control of PWM Signal

Follow these steps to control the MR012-30-4000 motors with the Phoenix 10 ESC either individually from the function generator and DC power source.

1. Connect red, white, and black speed controller outputs to the three poles on the motor using alligator clips. The motor fires each pole sequentially if it is spinning the wrong way simply switch two of the wires.
2. Connect the red and black power wires to the DC power source, but do NOT turn the output on yet.
3. Next you will need to use a small piece of 27AWG wire to get the PWM signal to the ESC. Stick the wire into the slot with the orange wire coming from it, and connect this to the positive output from the function generator.
4. Connect the negative output of the function generator to the negative pole on the DC power source.
5. Set the output on the power source as follows:
    a. Voltage=11.1V
    b. Current=3A
6. Set the ouput on the function generator as follows
    a. Function=Square Wave, Continuous
    b. Frequency=50Hz
    c. Amplitude=5V
    d. Pulse Width=1.0ms
7. Turn on the output from the function generator, then turn on the output from the DC power supply. The ESC will beep 3 times, pause, and then beep again twice. The motor speed can now be controlled by adjusting the pulse-width between 1-2ms.
8. To use multiple motors, make parallel connections from both the voltage and function generator. The use of multiple motors will draw more current than most power supplies can handle, so you might need to use power supplies in series to bump up the current.

Figure 20-Speed Controller Hookup Diagram

Appendix G: Rangefinder Calibration Data



Measured Range	V	k	y	m	b	Linearized Range
0.15	2.788	0.25	2.5	0.7569	0.3371	0.158607
0.2	2.571	0.25	2.222222222	0.7569	0.3371	0.188003
0.3	2	0.25	1.818181818	0.7569	0.3371	0.290278
0.4	1.53	0.25	1.538461538	0.7569	0.3371	0.418826
0.6	1.072	0.25	1.176470588	0.7569	0.3371	0.620703
0.7	0.93	0.25	1.052631579	0.7569	0.3371	0.710599
0.8	0.829	0.25	0.952380952	0.7569	0.3371	0.786731
0.9	0.743	0.25	0.869565217	0.7569	0.3371	0.861758