

Landing Platform Deck

A Major Qualifying Project Report:

Submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

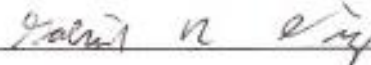
Degree of Bachelor of Science

In Aerospace Engineering

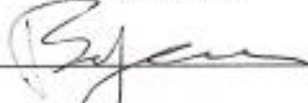
by



Travis Lee Austin



Gabriel R. Diaz



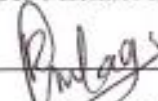
Zhaolong Li

Date: March 5, 2014

Approved by:



Professor Michael Demetriou



Professor Raghendra Cowlagi

Abstract

The project's goal is to build a motion platform that emulates the motion of an air carrier deck under varying sea conditions. This is the first of a series of projects that will lead to an autonomous helicopter landing on a platform that can measure weight distribution and impact force. The motion platform, run by three 14.4W motors is 1 meter long and 0.49 meters wide. Designs of the motion platform were performed using Solidworks. Kinematics, motion and ocean dynamics studies were performed using Matlab and Simulink. The ocean conditions were calculated and translated into gear rotations using Simulink as sine waves degrees per time. An Arduino, a single-board microcontroller was then programmed to take these gear rotations and send pulses to the motors to execute the maneuver.

Acknowledgements

We would like to thank the following individuals for their help and support throughout the entirety of this project. We would like to thank our project advisors, Professor Michael Demetriou and Professor Raghvendra Cowlagi for leading us towards the right direction into completing the project and for gathering money outside of our budget when we needed it. We would like to thank Gary Feldman for helping us understand how RC helicopter works and how to fly and fix the mCP X. We would also like to thank Randy Robinson for providing the group with a working PC power supply for the project.

Table of Authorship

Travis Lee Austin

- Material Analysis
- Emulating the motion of an Air Carrier
- Ocean Dynamics
- Simulink Results

Gabriel R. Diaz

- Introduction
- Motion Platform Design
- Motor Selection
- RC Helicopter Selection
- Construction of the Platform
- Conclusion and Recommendations

Zhaolong Li

- Micro-Controller Selection
- Programming Approaches
- Driving a Stepper Motor with an Arduino
- Stepper Motor Repositioning

- Implementation of the Arduino

Table of Contents

Abstract.....	2
Acknowledgements.....	2
Table of Authorship	2
Table of Contents.....	3
List of Figures	4
List of Tables	4
Introduction	5
Motion Platform Designs	5
Design 1.....	6
Design 2.....	6
Design 3.....	7
Design Choice.....	7
Material Analysis.....	8
Motor Selection	9
RC Helicopter Selection.....	11
Emulating the Motion of an Air Carrier	13
Ocean Dynamics.....	14
Simulink Results	23
Construction of the Platform	24
Motor Mounts.....	25
Power Source	26
Micro-Controller Selection.....	26
Programming Approaches	29
Precise Orientation	29
Uncertain/Random Orientation.....	29
Driving a Stepper Motor with the Arduino	32
User Interface	34
Stepper Motor Repositioning	36
Implementation of the Arduino.....	38

Movement Pattern.....	40
The Hardware and Software Limitation.....	40
Operation Manual.....	41
Conclusion and Recommendations	41
Works Cited.....	42

List of Figures

Figure 1: Motion Platform Design 1.....	6
Figure 2: Motion Platform Design 2.....	6
Figure 3: Motion Platform Design 3.....	7
Figure 4: Torque vs PPS for Stepper Motor	11
Figure 5: mCP X	12
Figure 6: Fossen's Simulink Program	20
Figure 7: Simulink Block for Heave Motion.....	21
Figure 8: Matlab Function into Simulink Block	22
Figure 9: Multilayer Simulink System.....	23
Figure 10: Oscilloscope Readouts	24
Figure 11: Constructed Motion Platform.....	25
Figure 12: Inside the Motor Mount/Enclosure	26
Figure 13: Arduino, BeagleBone or Raspberry Pi?	27
Figure 14: Arduino vs raspberry Pi vs BeagleBone	28
Figure 15: Normal Curve, Standard Deviation	30
Figure 16: SN754410 Motor.....	33
Figure 17: Motor Driver Function Table	33
Figure 18: Stepper Motor Connection	34
Figure 19: Connection Diagram for Bluetooth Module	35
Figure 20: Initializing the Motors.....	36
Figure 21: Configuration for Unipolar Motor Driver.....	39
Figure 22: Arduino Uno Connection to Potentiometer	39
Figure 23: Arduino Uno	40

List of Tables

Table 1: Advantages and Disadvantages of Motor Types.....	10
Table 2: mCP X Description	12
Table 3: Coefficient A.....	15
Table 4: Coefficient B	15
Table 5: Coefficient C.....	16

Introduction

Helicopter pilots can have a difficult time landing onto an air carrier deck, especially during a storm. The pilot would have to take into consideration the motion of the deck when landing. The pilot does not want to land at an angle that may cause the helicopter to tip over onto the deck. As the air carrier moves, the pilot may experience disorientation and vertigo. (Tormes 1974) The pilot may “fail to conceptualize his speed, motion, and altitude relative to the speed and motion of the ship.”¹ Research has been performed on creating autonomously landing helicopters. These groups tried different approaches such as using a tether cord as a guide, which was done in the University of Delaware. (Oh, Pathak and Agrawal 2005) This MQP is the first of a series of MQP’s that will ultimately conclude with a helicopter that can land autonomously under different sea conditions. The final objective of this project is to create a motion platform that can simulate the motion of an air carrier deck under different sea conditions.

Motion Platform Designs

Motion platforms have been utilized for different environments such as for flight simulators, entertainment rides and medical research. (Borta 2002) One popular motion platform for small projects is called the Stewart Platform, a small platform with six degrees of freedom created by six actuators connected to each corner or side at a 45° angle. (Roth and Roth 1996) The Stewart Platform was the starting point to understanding how motion platforms work and the parts and joints that makes up one. The initial platform idea was to have 6DOF and be large enough for future MQP’s should they want to experiment using RC airplanes or large RC helicopters. After doing research on the motions of the deck, the group decided to remove 3DOF: the yaw, sway, and translational motions and keep the pitch, roll, and heave. Though each of the six motions impact the pilot’s ability to successfully land a helicopter, the latter three mentioned are the main DOFs that can be more problematic for the pilots. The pitch and roll can cause the helicopter to tip if the timing of the landing is off while the heave can cause the pilot to land hard or not find the deck where the pilot expected to land. To have the 3DOF at least three actuators would have to be working together under the platforms with their positioning carefully determined. The group constructed three designs incorporating only three actuators.

¹ Tormes, Felix (1974). “Disorientation Phenomena in Naval Helicopter Pilots”, pg. 7

Design 1

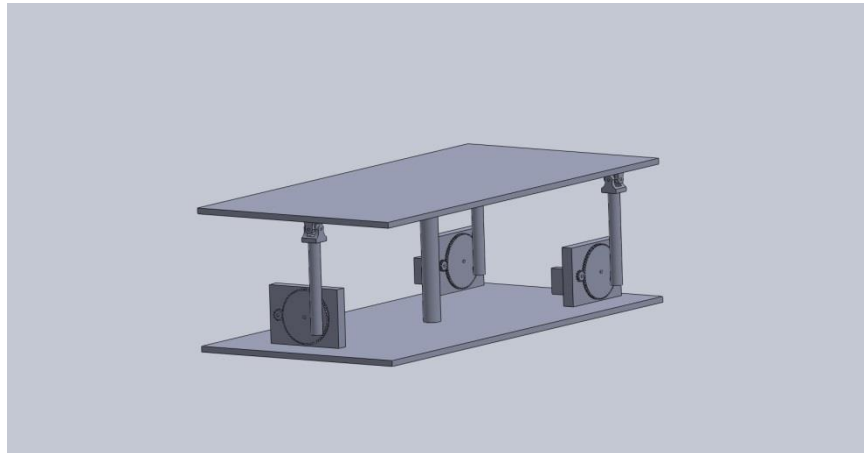


Figure 1: Motion Platform Design 1

Design 1 is the more complex of the three designs. This platform has two actuators attached to the corners at opposite ends of the shorter edge and a third actuator centered at the opposite edge. All three actuators must work together to create all 3DOF of the platform. Every DOF has all three actuators moving concisely together. The rod connecting the gears to the platform will have a ball joint connecting one end to the gear and a universal joint connecting it to the platform. A center pivot point was added as a safety feature so as to stop the rod from rotating away from their respective gears, which would create a swaying motion that is not wanted in the final product. To create this center pivot point, a tube, firmly connected to the bottom base, will encase a rod. This rod will have a spring under it connected it to the tube and a ball joint at the other end connecting it to the platform. The rod will be able to freely move linearly in the z-direction. This is the most stable of the three designs.

Design 2

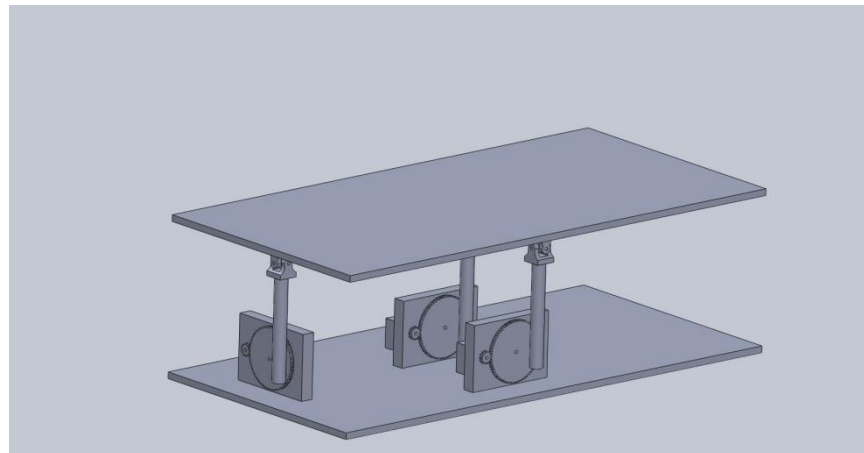


Figure 2: Motion Platform Design 2

Design 2 is very similar to the first design in which there are three actuators located between two planes, but the positioning of the actuators differs. In this design, one actuator is centered at the

small edges, one centered at the longer edge, and another located mid center of the platform. The reasoning for placing the actuator as such is to give each actuator fewer degrees of freedoms to control. The former actuator mentioned will only be a part of the pitch and heave motions, the second with the roll and heave motion, and the third with only the heave motions. In this way, programming the actuators is simplified because now, instead of having all three actuators needed to move for a single motion, for the pitch and roll; only two actuators would be needed to move concisely. The only DOF that would require all three actuators working jointly would be the heave. A problem with this design, is that half of the platform would become a cantilever beam and would create more stress onto the platform should the helicopter land or even collide at this section.

Design 3



Figure 3: Motion Platform Design 3

This design follows the same train of thought as the second design, trying to simplify the programming. Each actuator is only responsible for a single DOF of the platform. The top two actuators are positioned in the same spots as the non-center mid actuators in the second design, and control the pitch and roll motions, respectively. Under them is a third actuator, which controls solely the heave motion. There is a rod centered between the top and second planes, in which connects at the top by a ball joint. In between the second and last planes, there are four rods at each corner, connected through springs onto the higher-level plane. This design, though the easiest to program to emulate the deck, would be bulky in size and need the most material. The problem with this design is the amount of torque needed by the bottom motor to lift the top half of the motion platform. This design is also bulky and would be harder to carry.

Design Choice

While choosing a design three factors were considered: stability, portability, and ease of programming. The most important factor is the platform's stability. Should the helicopter crash into the platform, the platform should not break or dent especially with repetitive crashes. The second factor of importance is the portability. The platform should be able to be easily carried by the group from one

location to the next easily by the group. This is important because when experimenting, it would be safer to fly the helicopter outside rather than inside the lab. The third factor is the ease of programming. The motion of an air carrier on the ocean has many different levels of turbulence from a calm day to a severe hurricane. Programming and creating these levels of turbulence through the actuators can be a very difficult aspect of this project but it is the least important factor in deciding the design for the platform. Using these factors as a guide, the first design was chosen as the group's model platform.

Material Analysis

The material choices for the platform will be focused on two major pieces, the platform itself and the rods connecting the platform to the motors. The main criteria for these components is weight, since any weight would have to be pushed or pulled by the motors and the additional weight would potentially require a higher class of motors which would significantly increase the cost. The other basic criteria is strength or the ability to handle the stress caused from repeated landings and even a potential crash of larger helicopter such as the T-Rex 450.

Metal materials were looked at for the rods, starting with aluminum 6061 and aluminum 3004. Aluminum 6061 was cheaper while actually providing a higher strength. Since aluminum 6061 met all requirements for the rods and any alternative metals would add weight the analysis for the rods stopped there. However the initial selection for board materials was much broader, several different types of materials were looked at such as aluminums, plastics, and woods. Two specific materials in each of these categories were chosen for more detailed analysis: aluminum 6061, aluminum 3004, ABS plastic, PLA plastic, mahogany, and oak. After weight calculations aluminum was removed as an option for the board. Mahogany and oak were considerably equal to plastic in most respects but after detailed review; they were considered too inconsistent. Their values of strength and weight would vary greatly depending on water content which would be difficult to specify with any supplier. PLA was removed due to its lack of availability in the correct dimensions, ABS proved to be easily obtained at a low cost in almost any dimensions required and was able to pass all stress calculations.

Initial calculations for the platform were made based on the idea that the T-Rex 450 would land at the farthest corner from any supporting rods; this setup applies the largest bending stress to the platform. This calculation was later altered to account for a crash using numbers for a top speed attack dive. It should be noted that this is not an expected scenario but gives a good margin of safety or an idea of how the platform would handle the worst possible scenario. To calculate the maximum amount of stress the board could handle calculations that were made to see what the maximum repeatable stress would be, or the fracture stress of the board. These values were compared to the bending stress that would be applied to the board for a maximum velocity downward crash of the helicopter on the farthest corner edge. For the selected material ABS plastic the stress from this scenario proved to be several magnitudes less than the amount of stress the board could handle.

The condition for the rods was altered slightly to provide the worst possible case for the rods specifically. The effects of the board that realistically would distribute any crashes force were ignored, giving a simplified scenario where the helicopter would hit a rod head on in a vertical crash at top speed.

The material of aluminum 6061 initially gave a magnitude of strength higher than was needed that the equations were redone with a thinner rod of quarter inch diameter, which the material was also able to handle. This reduction in diameter resulted in a reduction in price by almost half.

The following equations are the final results for the stress analysis of both the board and rod during worst case scenarios.

$$V_{max} = 70 \frac{\text{mile}}{\text{hr}} = .0194 \frac{\text{mile}}{\text{s}}$$

$$a = \frac{(V_{max} - 0)}{0.2s} = 0.097 \frac{\text{miles}}{\text{s}} = 156.106 \frac{\text{m}}{\text{s}^2}$$

$$F_{impact} = (0.68 \text{ kg}) * a = 106.15 \text{ (N)}$$

$$\sigma_B = \frac{3FL}{2wt^2} = \frac{3 * 106.15 * 0.5}{2 * 0.5 * (0.00635)^2} = 3948787.898 \text{ (Pa)} = 3.948787 \text{ (MPa)}$$

$$\sigma_{rod} = \frac{F}{A} = \frac{106.1}{\pi * (\frac{0.0063}{2})^2} = 724520.5866 \text{ (Pa)} = 0.724520 \text{ (MPa)}$$

$$\sigma_c = \frac{K_c}{\sqrt{\pi a_d}}$$

$$a_d - 1(\text{mm}) = .001(\text{m})$$

$$\sigma_{ABS} = 17.07 \text{ (MPa)}$$

$$\sigma_{AL6061} = 305.5 \text{ (MPa)}$$

Motor Selection

There were three different types of motors investigated for this project: brushless DC motors, servo motors, and stepper motors. The factors put into selecting a motor are ease of positioning, cost, and torque/speed ratio. Each of their advantages and disadvantages are listed in a table below. (Design Trends: 'The Stepper Versus Brushless Servo Myth...' 2007)

Table 1: Advantages and Disadvantages of Motor Types

	Brushless Motor	Stepper Motor	Servo Motor
Advantages	Ability to maintain or increase torque at various speeds	Large amount of poles for motor positioning	Faster than Stepper Motor
	Low Power Loss	Constant holding Torque	Low Power Loss
	-	Does not require Driver	High torque at high speeds
Disadvantages	High Cost	Torque decreases as motor reaches maximum speed	Require driver and feedback loop
	Require Driver and Feedback Loop	Vibration if not properly loaded	Need power to hold position

The number one factor in choosing a motor was the ease of positioning. For each sea level state, the positioning has to be as precise as possible to collect accurate data. The stepper motor exceeds the other motors in this aspect. Stepper motors come in many different amounts of poles, making the positioning more defined for programming. The second factor of importance is the cost of the motor. Looking at the Digikey site, the servo motor was the cheapest and the Brushless motor the most expensive when compared at similar power output. The torque/speed ratio is a complicated factor because depending on how much money is put into the motor, one could get a motor of either selection that could satisfy the torque/speed requirement of the application. The brushless motor though has torque control, meaning that the torque produced by the motor is an independent, controlled variable and not a part of other aspects of the motor such as the speed. They all require a motor driver that controls the start and stop timing of the motor, rotation direction, regulating speed and torque, and setting fail-safes. The brushless motor and the servo also need a driver for the feedback loop so the Arduino can figure out the distance the shaft had turned in any given pulse. The group selected the stepper motor because of the high importance put into the positioning. The stepper motor does decrease in torque as the speed increases, but if the motor is carefully chosen so that at the maximum needed speed it still has enough torque to lift the platform, then the stepper motor will work well for the project.

The max torque and angular velocity were calculated to be 0.3466 nm (49.083 oz-in) and 2.85 rad/s (27.21 rpm), respectively. The stepper motor selected was a 14.4W, 1.8° step angle motor made by Hurst. The motor was bought from Allied Electronics at \$53.39 each. The torque-frequency curve taken from the datasheet is shown below. (Series H23R Hybrid Stepper Motor n.d.) The x-axis is the pulses per second (pps) and the y-axis is the torque in oz-in.

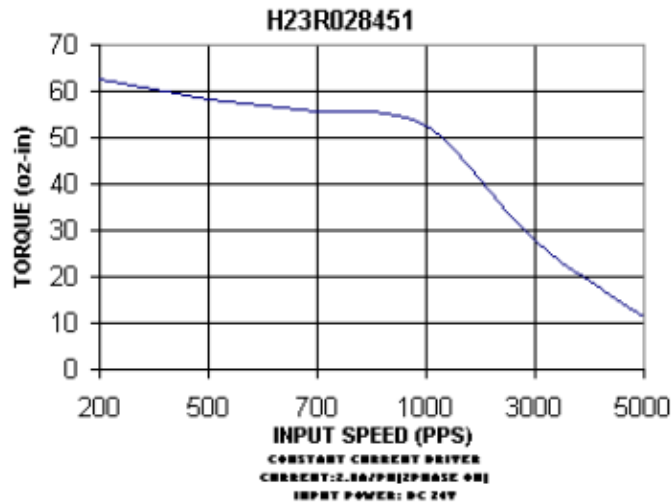


Figure 4: Torque vs PPS for Stepper Motor

The pull-out torque curve represents the maximum torque the motor can handle at specified speeds. If the load creates a torque larger than shown on the graph for that specific speed, the motor will stall. To calculate the pps, the angular velocity had to be changed from rpm to rps. The equation for pps is shown below. (Step Motor Basis n.d.)

$$pps = (rps) \left(\frac{\text{pulses}}{\text{revolution}} \right) (\text{microsteps})$$

The calculated pps for the max angular velocity of 27.21 rpm is 90.7 pulses per second with no extra microstepping between poles. Looking at the graph the torque is over 60 oz-in, which is higher than the max torque to lift the platform and small helicopter of 49.083 oz-in. The group took in consideration the torque needed for microstepping 4x should a more precise degree be wanted and the speed was not a problem. The pps for this case would be 362.8 pulses per second, which still achieves a torque higher than the expected load. This motor will be able to handle the max outputs needed for the project and also more weight for future projects.

RC Helicopter Selection

The project requires a RC helicopter in which to test land on the platform. There are two routes in which the group could have gone with in choosing the helicopter since no member has adequate experience in flying RC helicopters. The first route is to either purchase a large T-REX helicopter or fix the one that the advisor already had in possession. These large helicopters are more stable and easier to fly but should any piece should break, the repair cost would be greater than that of the small RC helicopters. The second route would be to purchase a small, single-rotor helicopter. The learning curve would be greater but with the helicopter bought ready-to-fly, more time could be spent on the building of the platform, programming, and testing. Should the small helicopter crash and break pieces, the parts are cheaper and repair is easier than the T-REX. The group decided to go with route two but build the platform to suit the formerly mentioned helicopter as well.

The helicopter must be able to maneuver similarly to a full-size helicopter and also have the same ground effects that a real helicopter would encounter when landing. The control options for the helicopter must include the roll, elevator, tail rotor, and a collective pitch over fixed pitch. A helicopter with collective pitch has the ability for the change the main rotor blades at different angles to change the applied lift. The RC helicopter must also be exceptionally responsive so should the helicopter be given pre-set instructions for future work on autonomous landing, the helicopter should be able to receive and perform the desired motions as quickly as possible. The Blade mCP X was recommended by an avid RC helicopter hobbyist as well as the hobby store owner of Turn4 Hobbies located in West Boylston when given the project requirements for the helicopter to the group. The specifications of the Blade mCP X is listed in the table below. (Blade mCP X n.d.)

Table 2: mCP X Description

Gross Weight	1.60 oz (45.5 g)
Length	9.25 in
Main Rotor Diameter	9.65 in
Rotor Type	Collective Pitch Single Rotor
Tail Rotor Diameter	1.40 in



Figure 5: mCP X (Blade mCP X BNF 2014)

Emulating the Motion of an Air Carrier

The platform dynamics have been worked on separate from the ocean dynamics, in an attempt to set up multiple parts of the overall equations of motion simultaneously. These dynamic equations will not be able to produce any actual numbers until the ocean dynamics are calculated giving the required angles of motion that the platform must be able to achieve. The platform dynamic equations will relate these angles to the moments and forces the motors will have to generate. These equations require repeat iterations for each set of motions the platform will have to perform, such as a rotation about the x-axis after a rotation about the y-axis has already been performed and vice versa.

In order to write out functions in Matlab that relates the desired motion of the platform to the required motion of the motors, it was necessary to do a three dimension mathematical calculation. These calculations were also necessary in order to accurately know how much torque would be required from the motors, this in turn will affect every other part of the platform as the motors are the largest part of the budget. The equation that relates motor motion to platform motion is based on a vector representing the arm bar or connecting rod between the motor arm and the corner of the platform. By doing some vector addition, relating the center of the board before motion to the gear, a second vector describing the motion of the gear arm in terms of α , plus the heave motion at the center of the board, and using an Euler angle transformation on the vector between the center of the board and the corner of the board, resulted in the following position equation was derived as seen in the equation below.

$$r_{B/C} = \frac{w \times \cos(\varphi) + L \times \sin(\varphi) \times \sin(\theta) - w}{L \times \cos(\theta) - L - R \times \cos(\alpha)}$$

$$w \times \sin(\varphi) - L \times \sin(\theta) \times \cos(\varphi) + z - m - R \times \sin(\alpha)$$

Since $r_{B/C}$ is not known and α is what needs to be solved for $r_{B/C}$ must be squared to equal its magnitude which is the known length of the bar and each of the x, y, and z terms must be squared. As shown below.

$$(r_{B/C})^2 = r_x^2 + r_y^2 + r_z^2$$

This gives the full position equation with only α unknown. Phi, theta, and "z" can be gathered from the Simulink program written by Fossen, while L, R, m, and W can be adjusted for the geometric measurements of the actual platform. Re-arranging the equation and attempting to solve for α is impossible to do by hand. Given the different inputs and constants, the simplest approach was to reduce the equation until all terms except for the variable α , were equal to A, B, and C.

$$A * \sin(\alpha) + B * \cos(\alpha) = C$$

Using a trigonometric identity of tangent this equation can be solved for α . There are some mathematical limitations that have to be understood involving which quadrant α is in, however the programming can be adjusted to account for this. All of this allows for α to be calculated with only the heave, pitch, and roll as inputs with all other constants being given from the dimensions of the platform.

Ocean Dynamics

For the purposes of creating a moving landing platform that simulates an aircraft carrier's landing deck it is important to understand the fundamental equations that govern the ocean and ship dynamics. These equations affect the motion of the aircraft carrier, and assuming an entirely rigid structure, the landing deck of the aircraft carrier as well. These equations are used to gather an understanding of how the platform should move and to allow for future variations in programming for different models of ocean conditions or varying ship parameters. The initial base equation involves both the ocean dynamics and the ship dynamics and models all effects on the ship. (Fossen 2002)

$$M * \dot{v} + C(v) * v + D(v) * v + g(\eta) = \tau + g_o + w$$

For this project, the effects of tau, which represents the control forces and moments of the ship that are used to maneuver, are neglected. It is assumed that the ship is not making any advanced or aggressive maneuvers while a helicopter is attempting to land. The moments and forces, "g," caused by the ballast system of the ship is also considered negligible as the ship is assumed to be one completely rigid piece. These variables could easily be added back into the equations for more advanced work if needed, however this project only requires a general equation to model the major motions of an aircraft carrier. This simplification leaves only the terms for the system inertial matrix M, Coriolis Effect C, Damping effects D, gravity/buoyancy forces g, and environmental disturbances w. The remaining environmental disturbance "w" on the right side of the equation can now be expanded into wind and ocean matrices. Since data about aircraft carriers center of gravity and moments of inertia are not public knowledge due to national security concerns they can be substituted with similar data from ships classified as very large crude carriers (VLCCs). VLCCs are a good approximation to an aircraft carrier since the only major differences are in weight and length. Making this assumption allows the use of the following equations for the wind forces and moments.

$$\begin{aligned} X_{wind} &= \left(\frac{1}{7.6}\right) * C_X * \gamma_r * \rho_a * V_r^2 * A_T \\ Y_{wind} &= \left(\frac{1}{7.6}\right) * C_Y * \gamma_r * \rho_a * V_r^2 * A_L \\ N_{wind} &= \left(\frac{1}{7.6}\right) * C_N * \gamma_r * \rho_a * V_r^2 * A_L * L \end{aligned}$$

C_x, C_y, and C_n are coefficients that are defined based on the following set of equations.

$$\begin{aligned} C_X &= A_0 + A_1 * 2 * \frac{A_L}{L^2} + A_2 * 2 * \frac{A_T}{B^2} + A_3 * \frac{L}{B} + A_4 * \frac{S}{L} + A_5 * \frac{C}{L} + A_6 * M \\ C_Y &= -(B_0 + B_1 * 2 * \frac{A_L}{L^2} + B_2 * 2 * \frac{A_T}{B^2} + B_3 * \frac{L}{B} + B_4 * \frac{S}{L} + B_5 * \frac{C}{L} + B_6 * A_{SS}/A_L) \\ C_N &= -(C_0 + C_1 * 2 * \frac{A_L}{L^2} + C_2 * 2 * \frac{A_T}{B^2} + C_3 * \frac{L}{B} + C_4 * \frac{S}{L} + C_5 * \frac{C}{L}) \end{aligned}$$

The coefficients A, B, and C in these equations numbered one through six are further defined by the following tables based on the angle between the ship's heading and the direction of the wind, called

gamma, which is also a key term in the general wind equations that could be used to run different simulations of varying weather conditions.

Table 3: Coefficient A

$\gamma_r(deg)$	A_0	A_1	A_2	A_3	A_4	A_5	A_6	S.E.
0	2.152	-5.00	0.234	-0.164	-	-	-	0.086
10	1.714	-3.33	0.145	-0.121	-	-	-	0.104
20	1.818	-3.97	0.211	-0.143	-	-	0.033	0.096
30	1.965	-4.81	0.243	-0.154	-	-	0.041	0.117
40	2.333	-5.99	0.247	-0.19	-	-	0.042	0.115
50	1.726	-6.54	0.189	-0.173	0.348	-	0.048	0.109
60	0.913	-4.68	-	-0.104	0.482	-	0.052	0.082
70	0.457	-2.88	-	-0.068	0.346	-	0.043	0.077
80	0.341	-0.91	-	-0.031	-	-	0.032	0.09
90	0.355	-	-	-	-0.247	-	0.018	0.094
100	0.601	-	-	-	-0.372	-	-0.020	0.096
110	0.651	1.29	-	-	-0.582	-	-0.031	0.09
120	0.564	2.54	-	-	-0.748	-	-0.024	0.1
130	-0.142	3.58	-	0.047	-0.7	-	-0.028	0.105
140	-0.677	3.64	-	0.069	-0.529	-	-0.032	0.123
150	-0.723	3.14	-	0.064	-0.475	-	-0.032	0.128
160	-2.148	2.56	-	0.081	-	1.27	-0.027	0.123
170	-2.707	3.97	-0.175	0.126	-	1.81	-	0.115
180	-2.529	3.76	-0.174	0.128	-	1.55	-	0.112

Table 4: Coefficient B

$\gamma_r(deg)$	B_0	B_1	B_2	B_3	B_4	B_5	B_6	S.E.
10	0.096	0.22	-	-	-	-	-	0.015
20	0.176	0.71	-	-	-	-	-	0.023
30	0.225	1.38	-	0.023	-	-0.29	-	0.03
40	0.329	1.82	-	0.043	-	-0.59	-	0.054
50	1.164	1.26	0.121	-	-0.242	-0.95	-	0.055
60	1.163	0.96	0.101	-	-0.177	-0.88	-	0.049
70	0.916	0.53	0.069	-	-	-0.65	-	0.047
80	0.844	0.55	0.082	-	-	-0.54	-	0.046
90	0.889	-	0.138	-	-	-0.66	-	0.051
100	0.799	-	0.155	-	-	-0.55	-	0.05
110	0.797	-	0.151	-	-	-0.55	-	0.049
120	0.996	-	0.184	-	-0.21	-0.66	0.34	0.047
130	1.014	-	0.191	-	-0.28	-0.69	0.44	0.051
140	0.784	-	0.166	-	-0.209	-0.53	0.38	0.06
150	0.536	-	0.176	-0.029	-0.163	-	0.27	0.055
160	0.251	-	0.106	-0.022	-	-	-	0.036
170	0.125	-	0.046	-0.12	-	-	-	0.022

Table 5: Coefficient C

$\gamma_r(deg)$	C_0	C_1	C_2	C_3	C_4	C_5	S.E.
10	0.0596	0.061	-	-	-	-0.074	0.0048
20	0.1106	0.204	-	-	-	-0.17	0.0074
30	0.2258	0.245	-	-	-	-0.38	0.0105
40	0.2017	0.457	-	0.0067	-	-0.472	0.0137
50	0.1759	0.573	-	0.0118	-	-0.523	0.0149
60	0.1925	0.480	-	0.0115	-	-0.546	0.0133
70	0.2133	0.315	-	0.0081	-	-0.526	0.0125
80	0.1827	0.254	-	0.0053	-	-0.443	0.0123
90	0.2627	-	-	-	-	-0.508	0.0141
100	0.2102	-	-0.0195	-	0.0335	-0.492	0.0146
110	0.1567	-	-0.0258	-	0.0497	-0.457	0.0163
120	0.0801	-	-0.0311	-	0.0740	-0.396	0.0179
130	-0.0189	-	-0.0488	0.0101	0.1128	-0.420	0.0166
140	0.0256	-	-0.0422	0.01	0.0889	-0.463	0.0162
150	0.0552	-	-0.0381	0.0109	0.0689	-0.476	0.0141
160	0.0881	-	-0.0306	0.0091	0.0366	-0.415	0.0105
170	0.0851	-	-0.0122	0.0025	-	-0.220	0.0057

The other have of the environmental effects are the wave equations which are initially written as the following.

$$\begin{aligned}
 X_{waves} &= \frac{K_{\omega_1} * s}{s^2 + 2 * \lambda_1 * \omega_{e1} * s + \omega_{e1}^2} * \omega_1 + d_1 \\
 Y_{waves} &= \frac{K_{\omega_2} * s}{s^2 + 2 * \lambda_2 * \omega_{e2} * s + \omega_{e2}^2} * \omega_2 + d_2 \\
 N_{waves} &= \frac{K_{\omega_3} * s}{s^2 + 2 * \lambda_3 * \omega_{e3} * s + \omega_{e3}^2} * \omega_3 + d_3
 \end{aligned}$$

The right side of the equation can be further simplified through manipulation and basic assumptions, such as ignoring the effects of drift “d” which for the purposes of this project are not large enough over the projected time frame of a helicopter landing to significantly affect ship motion.

$$\frac{2(\lambda\omega_0\sigma)\omega}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4(\lambda\omega_0\omega)^2}}$$

ω_0 is substituted with ω_e which is the encounter frequency of the waves calculated with the following equation.

$$\omega_e(U, \omega_0, \beta) = \left| \omega_0 - \frac{\omega_0^2}{g} U \cos \beta \right|$$

This equation allows the equations of motion to be changed based on the velocity of the ship, “U” and the angle between the waves direction and ships heading, “B.” Since these equations are based

on the modified Pierson-Moskowitz equations λ equals 0.26 and sigma can be solved using the significant wave height in the following form.

$$H_s = 4\sigma$$

These equations allow the wave forces to be calculated based on the ship speed, ship direction relative to wave direction, wave frequency, and the significant wave height. These variables can easily be changed to imitate different ship parameters and different sea states once the equations are inputted into Matlab.

The other half of the equation consists of variables representing the system inertia matrix, Coriolis-centripetal, damping, and gravitational/buoyancy forces and moments. The system inertia matrix is broken into two parts, one for the rigid body of the ship and the other for the added mass of water around the ship.

$$\begin{aligned} M_{RB} &= H^T(r_g^b)M_{RB}^{cg}H(r_g^b) \\ &= \begin{bmatrix} ml_{3x3} & -mS(r_g^b) \\ mS(r_g^b) & I_c - mS^2(r_g^b) \end{bmatrix} \\ &= \begin{bmatrix} m & 0 & 0 & 0 & mz_g & -my_g \\ 0 & m & 0 & -mz_g & 0 & mx_g \\ 0 & 0 & m & my_g & -mx_g & 0 \\ 0 & -mz_g & my_g & I_z & -I_{xy} & -I_{zx} \\ mz_g & 0 & -mx_g & -I_{xy} & I_y & -I_{yz} \\ -my_g & mx_g & 0 & -I_{xz} & -I_{yz} & I_z \end{bmatrix} \end{aligned}$$

$$\begin{aligned} M_A &= H^T(r_g^b)M_{RB}^{cg}H(r_g^b) \\ &= \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & 0 \\ 0 & 0 & -Z_{\dot{w}} \\ 0 & z_g Y_{\dot{v}} & -y_g Z_{\dot{w}} \\ -z_g X_{\dot{u}} & 0 & x_g Z_{\dot{w}} \\ y_g X_{\dot{u}} & -x_g Y_{\dot{v}} & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -z_g X_{\dot{u}} & y_g X_{\dot{u}} \\ z_g Y_{\dot{v}} & 0 & -x_g Y_{\dot{v}} \\ -y_g Z_{\dot{w}} & x_g Z_{\dot{w}} & 0 \\ -z_g^2 Y_{\dot{v}} - y_g^2 Z_{\dot{w}} - K_p & x_g y_g Z_{\dot{w}} & x_g z_g Y_{\dot{v}} \\ x_g y_g Z_{\dot{w}} & -z_g^2 X_{\dot{u}} - x_g^2 Z_{\dot{w}} - M_{\dot{q}} & y_g z_g X_{\dot{u}} \\ x_g z_g Y_{\dot{v}} & y_g x_g X_{\dot{u}} & -y_g^2 X_{\dot{u}} - x_g^2 Y_{\dot{v}} - N_{\dot{r}} \end{bmatrix} \end{aligned}$$

For the Coriolis and centripetal matrices apply to the rigid body and added mass of a surface ship.

$$C_{RB}(v) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -m(y_g q + z_g r) & m(y_g p + w) & m(z_g p - v) \\ m(x_g q - w) & -m(z_g r + x_g p) & m(z_g q + u) \\ m(x_g r + v) & m(y_g r - u) & -m(x_g p + y_g q) \\ m(y_g q + z_g r) & -m(x_g q - w) & -m(x_g p + v) \\ -m(y_g p + w) & m(z_g r + x_g p) & -m(y_g q - u) \\ -m(z_g p - v) & -m(z_g q + u) & m(x_g p + y_g q) \\ 0 & -I_{yz}q - I_{xz}p + I_z r & I_{yz}q + I_{xy}p - I_y r \\ I_{yz}q + I_{zx}p - I_z r & 0 & -I_{zx}q - I_{zy}p + I_z r \\ -I_{yz}q - I_{zy}p + I_y r & I_{zx}q + I_{zy}p - I_z r & 0 \end{bmatrix}$$

$$C_A(v) = \begin{bmatrix} 0 & 0 & 0 & 0 & -a_3 & a_2 \\ 0 & 0 & 0 & a_3 & 0 & -a_1 \\ 0 & 0 & 0 & -a_2 & a_1 & 0 \\ 0 & -a_3 & a_2 & 0 & -b_3 & b_2 \\ a_3 & 0 & -a_1 & b_3 & 0 & -b_2 \\ -a_2 & a_1 & 0 & -b_2 & b_1 & 0 \end{bmatrix}$$

$$\begin{aligned} a_1 & X_{\dot{u}}u + X_{\dot{v}}v + X_{\dot{w}}w + X_{\dot{p}}p + X_{\dot{q}}q + X_{\dot{r}}r \\ a_2 & Y_{\dot{u}}u + Y_{\dot{v}}v + Y_{\dot{w}}w + Y_{\dot{p}}p + Y_{\dot{q}}q + Y_{\dot{r}}r \\ a_3 & Z_{\dot{u}}u + Z_{\dot{v}}v + Z_{\dot{w}}w + Z_{\dot{p}}p + Z_{\dot{q}}q + Z_{\dot{r}}r \\ b_1 & = K_{\dot{u}}u + K_{\dot{v}}v + K_{\dot{w}}w + K_{\dot{p}}p + K_{\dot{q}}q + K_{\dot{r}}r \\ b_2 & M_{\dot{u}}u + M_{\dot{v}}v + M_{\dot{w}}w + M_{\dot{p}}p + M_{\dot{q}}q + M_{\dot{r}}r \\ b_3 & N_{\dot{u}}u + N_{\dot{v}}v + N_{\dot{w}}w + N_{\dot{p}}p + N_{\dot{q}}q + N_{\dot{r}}r \end{aligned}$$

For the dampening matrices there is no added mass, but instead the dampening forces are broken into linear and non-linear matrices.

$$D = H^T(r_g^b)D^{cg}H(r_g^b)$$

$$= \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & 0 \\ 0 & 0 & -Z_w \\ 0 & z_g Y_v & -y_g Z_w \\ -z_g X_u & 0 & x_g Z_w \\ y_g X_u & -x_g Y_v & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -z_g X_u & y_g X_u \\ z_g Y_v & 0 & -x_g Y_v \\ -y_g Z_w & x_g Z_w & 0 \\ -z_g^2 Y_v - y_g^2 Z_w - K_p & x_g y_g Z_w & x_g z_g Y_v \\ x_g y_g Z_w & -z_g^2 X_u - x_g^2 Z_w - M_q & y_g z_g X_u \\ x_g z_g Y_v & y_g x_g X_u & -y_g^2 X_u - x_g^2 Y_v - N_r \end{bmatrix}$$

$$D_n = H^T(r_g^b)D_n^{cg}(v)H(r_g^b)$$

$$= \begin{bmatrix} -X_{u|u}|u| & 0 & 0 \\ 0 & -Y_{v|v}|v| & 0 \\ 0 & 0 & -Z_{w|w}|w| \\ 0 & z_g Y_{v|v}|v| & -y_g Z_{w|w}|w| \\ -z_g X_{u|u}|u| & 0 & x_g Z_{w|w}|w| \\ y_g X_{u|u}|u| & -x_g Y_{v|v}|v| & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -z_g X_{u|u}|u| & y_g X_{u|u}|u| \\ z_g Y_{v|v}|v| & 0 & -x_g Y_{v|v}|v| \\ -y_g Z_{w|w}|w| & x_g Z_{w|w}|w| & 0 \\ -z_g^2 Y_v - y_g^2 Z_{w|w}|w| - K_{p|p}|p| & x_g y_g Z_{w|w}|w| & x_g z_g Y_{v|v}|v| \\ x_g y_g Z_{w|w}|w| & -z_g^2 X_{u|u}|u| - x_g^2 Z_{w|w}|w| - M_{q|q}|q| & y_g z_g X_{u|u}|u| \\ x_g z_g Y_{v|v}|v| & y_g x_g X_{u|u}|u| & -y_g^2 X_{u|u}|u| - x_g^2 Y_{v|v}|v| - N_{r|r}|r| \end{bmatrix}$$

The gravitational and buoyancy forces can be written into one matrix as follows:

$$G = H^T(r_g^b)G^{cg}H(r_g^b)$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -Z_z & y_g Z_z & -x_g Z_z & 0 \\ 0 & 0 & y_g Z_z & -y_g^2 Z_z - K_\phi & x_g y_g Z_z & 0 \\ 0 & 0 & -x_g Z_z & x_g y_g Z_z & -x_g^2 Z_z - M_\theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Arranging these matrices into a coherent set of equations will prove incredibly difficult to do by hand and will be incredibly time consuming to calculate for multiple iterations with varying values for wave height, wave frequency, and ship angles relative to both the wind and the waves. Fossen includes several Matlab codes for the majority of these matrices, which prove to be instrumental in solving these equations and allow for quick corrections for any future changes to the projects simulation needs.

The mathematics required to calculate for the various variables regarding ship motion in different ocean conditions proved to be highly difficult, resulting in the need for a computer program that could perform these calculations quickly. While Matlab was an initial choice, literature review uncovered a Simulink file that was already capable of calculating the majority of required variables. Figure 6 below shows the original Simulink program as written by Thor Fossen.

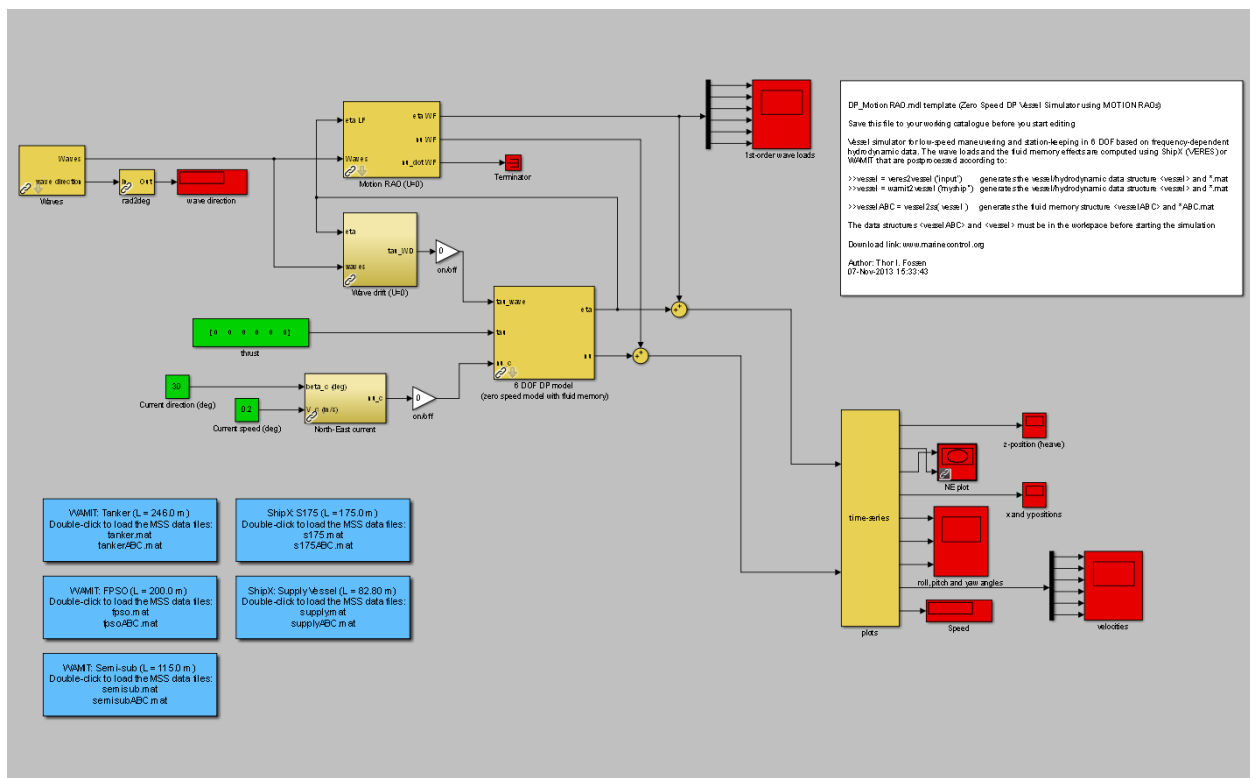


Figure 6: Fossen's Simulink Program

This code allowed for the input of several different ocean conditions, and had preloaded ship information for more accurate calculations. However, the code only outputted some of the necessary variables for the project: heave, pitch, and roll. Since the project required the velocity rates and the acceleration rates of these variables, several block additions were needed. All three variables had several Simulink blocks added to their original outputs in much the same way as shown for heave, in Figure 7 below.

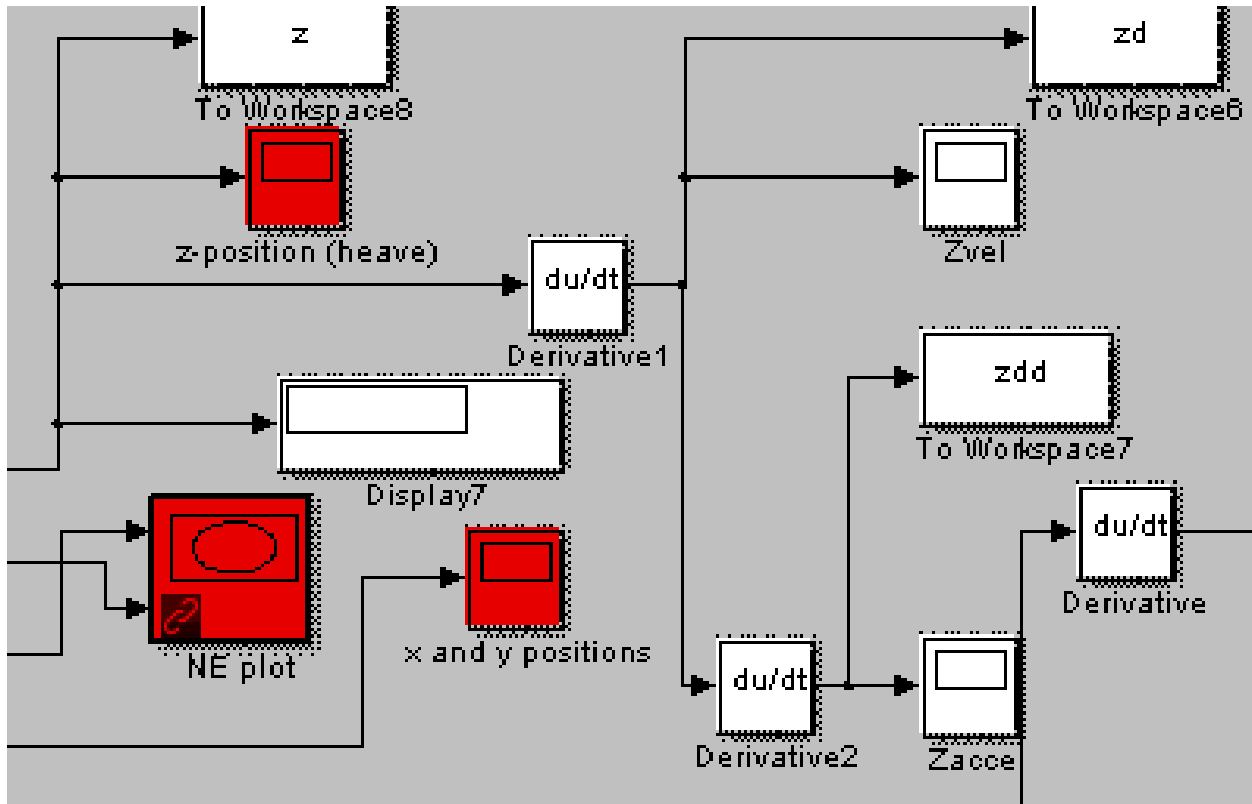


Figure 7: Simulink Block for Heave Motion

The red blocks show the original Simulink code, while the white blocks show the new additions to the program. The major additions are the derivative blocks that take the heave motion and transform it into heave velocity and heave acceleration. The other blocks shown are output blocks, which are outputting to the oscilloscope blocks allowing for an easy check as to whether or not the variables are behaving in a logical manner. The “To workspace” blocks are one of the important outputs, as this allows Simulink to send the variables to Matlab for further calculations. Using functions in Matlab that can relate these Simulink outputs to the Alpha positions and velocities, the motors can be positioned correctly. If the Matlab functions were brought into Simulink, this would increase the processing time, allowing for a simple one-click operation to output everything needed for programming the motors that move the platform.

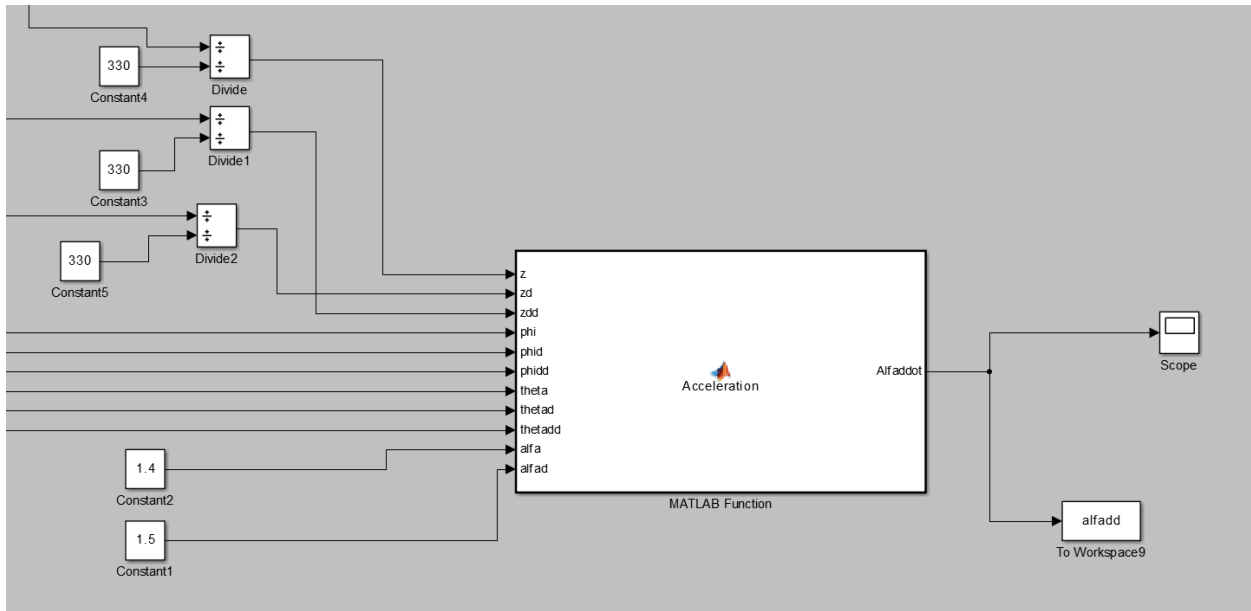


Figure 8: Matlab Function into Simulink Block

An example of turning a Matlab function into a Simulink block is shown in Figure 8 above. This block will be able to take the variable calculated through previous Simulink adaptations and turn them into data needed for operating the motors of the platform. The Matlab blocks have significant problems in regards to what kind of programming they can use. In order to bypass the limitations of the Matlab block and keep the entire process in either Simulink or Matlab as opposed to switching back and forth, certain adaptations were made. Since certain solve commands could not be used inside the Matlab block the trigonometric identity discussed in the mathematics section was substituted. The trigonometric identity technique was already a Matlab function, however the “if” statements it required did not function inside the Matlab block. Instead, the trigonometric function was “translated” into Simulink manually. This multilayered Simulink system was then reduced into a single block for simplicity as shown by Figure 9. The A, B, and C variables needed for the trigonometric function were solved for using a Matlab block, shown as the blocks on the right in Figure 9. They contain all of the mathematics as well as the variables for platform dimensions. This allows the program to be adapted for multiple platforms of different sizes as long as the set up remains the same. This also allows the trigonometric solver block to remain independent, since it should not require adjustments. The original trigonometric identity function script contained several “if” statements to account for the multiple quadrants that the angle could be found in, these statements were incorporated into the Simulink version as well but were fed into a separate oscilloscope block as seen on the far right side of Figure 9. These oscilloscope blocks only read zero for typical ocean conditions used with this platform, but would allow the same block to be applied to other scenarios.

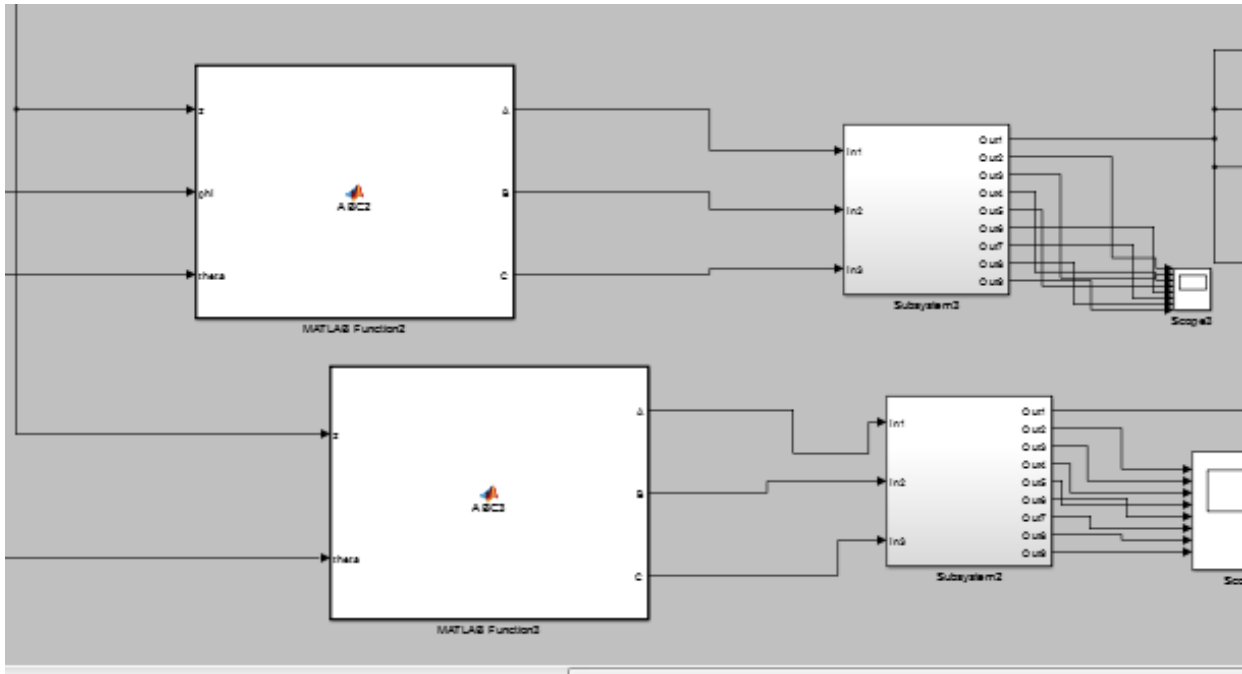


Figure 9: Multilayer Simulink System

Once the numbers leave the trigonometric block the selected output is fed into a separate oscilloscope as well as a Matlab output block. The oscilloscope block allows for quick “sanity” checks for de-bugging. The Matlab output blocks give the actual results to Matlab where the time steps can be sent for manual insertion to another software program that is more compatible with operating a stepper motor. For future use this step could possibly be bypassed with a specialized Simulink block that could possibly connect the stepper motor directly to the ocean and dynamic conversion Simulink programs in real time. However, no translator blocks were applicable at present.

Simulink Results

The results from Simulink are detailed in both oscilloscope graphically as well as time stamped results into Matlab. The oscilloscope gives immediate visual results and a better visual of how the gears are supposed to move. The time stamped results can be uploaded easily into Arduino for the programming but are difficult to use for troubleshooting purposes. The graphs from set known numbers are easy to visualize and identify any errors in coding.

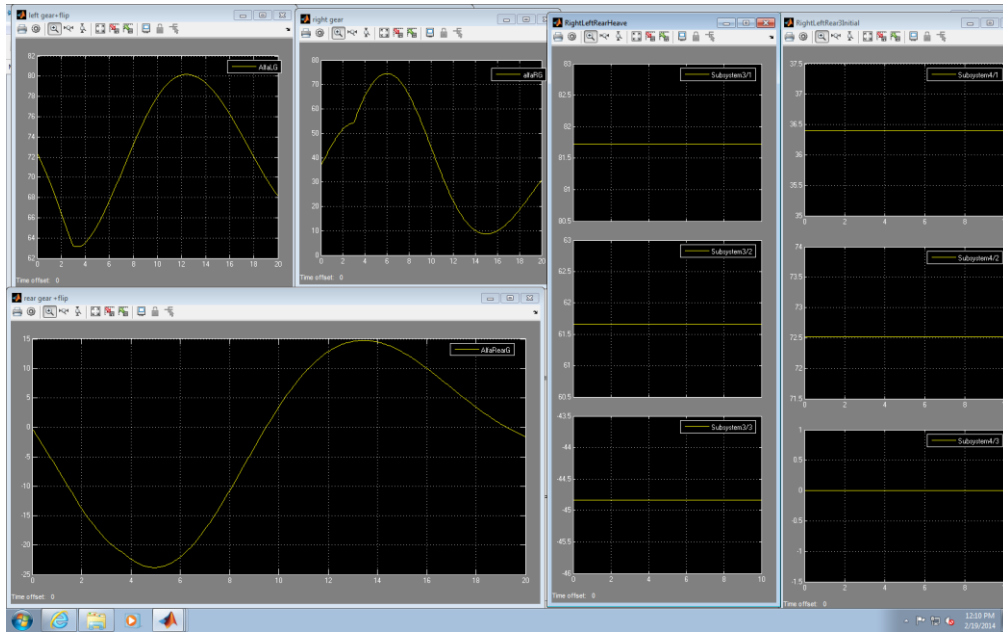


Figure 10: Oscilloscope Readouts

As shown in the oscilloscope, the readouts given are visuals of how each individual gear should move. Figure 10 is an example of one of the test runs for the program, showing results for a set of heave without roll or pitch. The x-axis is time in seconds while the y-axis is the degrees the motor lead rotates. Running multiple scenarios allows for narrowing down of any program issues. This makes it easier to figure out that the right gear would need to be programmed slightly offset because of an error at zero pitch and roll. This was only a programming error, but was almost impossible to see without the visual feedback form the oscilloscope results. The above figure gives readouts for each of the gears on the left window going clockwise with readouts for the left, right, and rear gears while the right two column graphs give the heave and initial conditions with left, right, and rear gears stacked from top to bottom. These are just one of example of the multitude of scenarios that were run.

Simulink Operation Manuel

Construction of the Platform

The platform had to be large enough should a future MQP want to use it to experiment using larger RC helicopters or even airplanes. For this reason, the platform was design to be 1 meter by 0.5 meter. The 1-meter length was chosen by researching the average amount of runway needed to land an RC airplane. The motion platform is being modeled from the Nimitz Class air carrier which flight deck is 317 meter long and 77 meters wide. This would scale the platform's width to be 0.25 meters wide. (The centerpiece of Navy operations: Aircraft Carriers n.d.) The group was worried that this width would not suffice for the larger helicopters such as the T-REX whose length is exactly the same size as the width of the platform. (T-REX 500 n.d.) Instead, the group decided to cut the Nimitz flight deck in half, which

would work better for the larger helicopters and give more room to future MQP's to add arresting gear. This turned the platform to be 1 meter by 0.49 meters.

Scaling the maximum lift that an air carrier can experience on a large wave would create a maximum pivot angle of 15° at the center point, which creates a maximum height of about an eighth of a meter (5 in). This would mean that the connection of the rod would have to be the same distance away from the motor lead to achieve this pivot angle.

While building the motion platform, it was realized that the project might only need the motors connected to the rods with a ball joint directly rather than using gears if the motor is strong enough. A reason for this is so that the maximum speed is not hindered for more torque. The motor's speed and torque will be the limiting factor. Another change was replacing the ABS plastic base that holds everything together to plywood for budgeting reasons. Figure 11 below reveals the built motion platform.



Figure 11: Constructed Motion Platform

Motor Mounts

The motor mounts were handmade utilizing the extra pieces of ABS plastic, ceiling boxes, and blocks of 2x4 wood blocks due to budgeting reasons. Holes were drilled into the ceiling box through the large area side. These holes were sized to be a quarter inch, same size as the mounting holes for the motors. The ceiling box also would enclose the motor when connected, with the ABS plastic covering the rear. This is shown in Figure 11. The block of wood is to raise the motor to the desired height so that the servo horn can rotate without hitting the base.

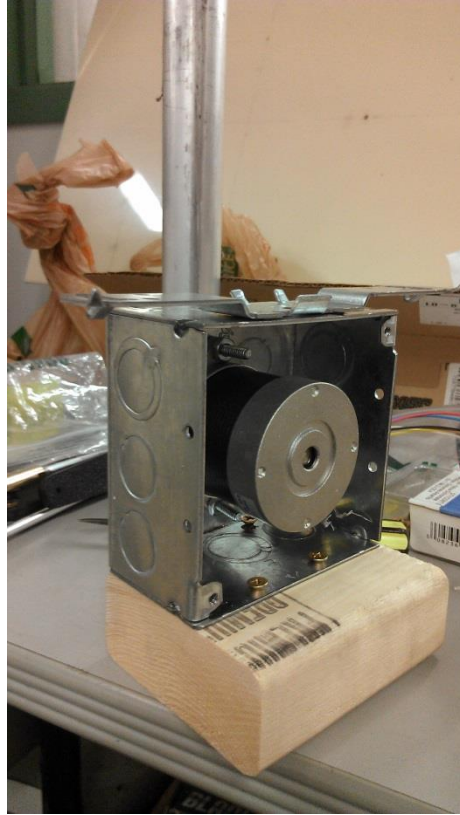


Figure 12: Inside the Motor Mount/Enclosure

Power Source

If the budget had permitted, a rechargeable external battery would be perfect to power the motors. The motors require 12V with a nominal power of 14.4W. The best and cheapest way to power the motor was to use a PC power supply that is available for free from Randy. The PC power supply would be perfect because it provides a current with a regulated voltage ranging from 3.3V to 12V depending on the color of the wire. For the power supply acquired, the yellow wire provides the 12V. The disadvantage is that they power supply needs to be connected to an outlet, making the platform not useful outside unless near a power outlet. To have the power supply work the green wire must either have a load or shorted to have power supply work. The reason for this is because the green wire is the Power On wire meaning it checks to make sure that the power supply is properly connected to something before it lets any current through the other wires. A jumper cable was used to short this green wire, having the power supply always ready to supply the current when the switch is turned on. Once the jumper cable was placed and the insulation for the yellow (12V) and black (ground) were stripped, they can be easily connected with the motor and motor drivers to a breadboard.

Micro-Controller Selection

The micro-controller is the brain of the landing platform; it controls the behavior of the actuators and receives feedback. There are three most popular micro-controllers in the market: the Arduino, the

Raspberry Pi and the BeagleBone. Each of these controllers has its strengths and weaknesses. These three types of controllers looked into were carefully before making the decision.



Figure 13: Arduino, BeagleBone or Raspberry Pi? (Allan 2013)

The table below described their basic characteristics (Allan 2013):

Name	Arduino Uno	Raspberry Pi	BeagleBone
Model Tested	R3	Model B	Rev A5
Price	\$29.95	\$35	\$89
Size	2.95"x2.10"	3.37"x2.125"	3.4"x2.1"
Processor	ATMega 328	ARM11	ARM Cortex-A8
Clock Speed	16MHz	700MHz	700MHz
RAM	2KB	256MB	256MB
Flash	32KB	(SD Card)	4GB(microSD)
EEPROM	1KB		
Input Voltage	7-12v	5v	5v
Min Power	42mA (.3W)	700mA (3.5W)	170mA (.85W)
Digital GPIO	14	8	66
Analog Input	6 10-bit	N/A	7 12-bit
PWM	6		8
TWI/I2C	2	1	2
SPI	1	1	1
UART	1	1	5
Dev IDE	Arduino Tool	IDLE, Scratch, Squeak/Linux	Python, Scratch, Squeak, Cloud9/Linux
Ethernet	N/A	10/100	10/100
USB Master	N/A	2 USB 2.0	1 USB 2.0
Video Out	N/A	HDMI, Composite	N/A
Audio Output	N/A	HDMI, Analog	Analog

Figure 14: Arduino vs raspberry Pi vs BeagleBone

When comparing computational power, the Raspberry Pi and the BeagleBone are very powerful; in fact they have the computational power similar to an average personal desktop manufactured 10 years ago. On the other hand, the computational power of an Arduino seems negligible compare to the other two (16MHz compare to 700MHz).

However, the Raspberry Pi and the BeagleBone maintain a real Linux operating system, which will consume a significant amount of the computational power. The Arduino on the other hand will only perform a single task uploaded to its memory. What this means is that for a simple task, an Arduino is most likely a better choice, but for task that requires heavy computation and multitasking, a Raspberry Pi or a BeagleBone should be used.

In the moving platform, there must be some Servos or Motors involved; controlling these components requires the use of the GPIO (General Purpose Input/Output) pin, and PWM (Pulse width modulation). Therefore, the second consideration for the micro-controller is the number of GPIO pins and the number of PWM capacity. As can be seen, the Raspberry Pi has only 8 GPIO pins and none of them can do PWM, therefore it is not a good idea to use it to control multiple motors or servos. The Arduino has 14 GPIO pins and 6 of them can be used to do PWM, the team thinks this number of GPIO pins is sufficient to control the number of motors and servos that the platform will need, so going for the BeagleBone (66 GPIO and 8 PWM) would be a waste of budget.

Other factors to consider are the price for each controller and how to program them. The price for a standard Arduino Uno is about \$30~\$35, which is the cheapest amount the three. The price for a Raspberry Pi is about \$45, and the price of a BeagleBone is about \$90, which might be too expensive since the budget of this project is \$480. For the IDE, the Arduino comes with its own IDE and programming language, with the syntax pretty similar to C. For the Raspberry Pi and BeagleBone, the main programming language will be Python because there are extensive Python libraries to communicate with the GPIO pins. However, since the Raspberry Pi has a real Linux Operating System, so it can be program with many Linux supported languages such as Java, C, Ruby or even shell script.

Having considered all of the factors, the team decided that an Arduino would be a better choice for the moving platform. However, based on the intensity of the math computation needed for the microcontroller, a BeagleBone will also be considered.

Programming Approaches

Precise Orientation

For precise orientation, every movement of the platform is carefully planned and calculated. In other words, for every computational cycle of the microcontroller, the platform will move in a specific rate so that it would reach a desire state at a specific time. Each rates, desire states and desired times is calculated or inputted by operator before the process start.

Pros:

1. Operator will have full control of the platform.
2. Motion of the platform is predictable, the orientation and motion of the platform at any time is known.
3. Better actuator cooperation since the motion of each actuator is preloaded into the controller.

Cons:

1. Fix/periodic motion, which eliminate the uncertainty of the ocean.
2. Required more computational power. Heavy math computation might interrupt the controlling of the actuators.
3. Some motion needs to be hard coded, result in large and un-clear program.

Uncertain/Random Orientation

The major characteristic of the ocean movement is that it is unpredictable. This uncertainty can be used to develop the moving platform solution. Similar to the previous approach, for every computational cycle of the microcontroller, the platform will move in a specific rate to achieve a specific goal state (notice that it is not enforce that it reaches the state at a specific time). The different is that for

every computational cycle, the rate is randomly picked from a normal distribution with an empirical mean and standard deviation base on the ocean state (or mode for the platform). Once the platform reaches a desired state, the next desire state is chosen by the same method.

To demonstrate this, assume the curve below in Figure 15 is the distribution of the wave speed, with mean of 3ft/sec and standard deviation of 1ft/sec. For the next wave speed, it is chosen at random based on the distribution curve below, so there are 38.2% chance the next wave speed will in between 2.5ft/sec and 3.5ft/sec, and 15% chance it will be in between 3.5ft/sec and 4ft/sec.

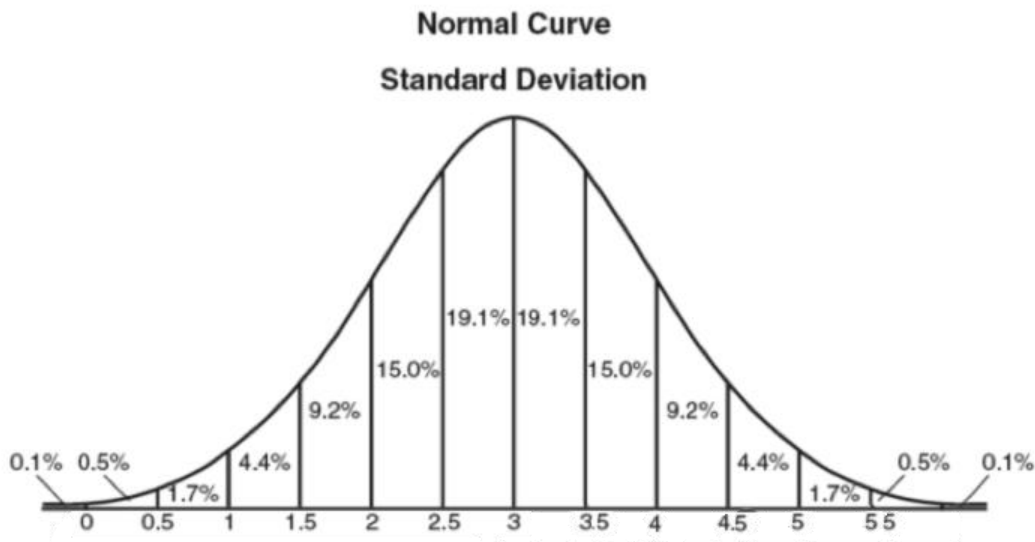


Figure 15: Normal Curve, Standard Deviation (Roberts 2012)

Pros:

1. Reduce computational power
2. More realistically mimic the ocean movement
3. More ocean states can be simulated.

Cons:

1. Need gyro to determine the current platform orientation
2. Need to carefully choose empirical data for every ocean state.
3. Need a certain motor/servo orientation for this method to work best.

Pseudo code for this approach

```

//0 means board is reaching max, 1 means board is reaching min

turn = 0;

current_position = 0; //initial position

servos.initial(current_position); // this should make the board flat

// Unit of current_position and goal will be in degree

goal = pick_random_max(max_mean, max_standard_deviation); // choose a goal state

while(1){

  if(turn == 0){

    while(current_position < goal){

      current_position++;

      servo.position(current_position);

      // use wait time to control the speed of the servo

      // if we have multiple servos, another clock technique will

      // be use instead of waiting.

      wait_time = pick_random_delay_time(delay_mean, delay_std);

      delay(wait_time)

    }

    // choose the minimum goal state

    goal = pick_random_min(min_mean, min_standard_deviation);

    // reverse the turn counter to indicate the board is going to min value

    turn = 1 - turn;

  }

  // Same as above be just reverse the sign

```

```

if(turn == 1){
  while(current_position > goal){
    current_position--;
    servo.position(current_position);
    wait_time = pick_random_delay_time(delay_mean, delay_std);
    delay(wait_time)
  }
  // choose the maximum goal state
  goal = pick_random_max(max_mean, max_standard_deviation);
  // reverse the turn counter to indicate the board is going to min value
  turn = 1 - turn;
}
}

```

Driving a Stepper Motor with the Arduino

To drive a stepper motor with the Arduino, a motor driver will be needed with separate power source for the motors. The motor cannot be directly to the Arduino board because this will cause a large amount of current flow through the Arduino board; the amount of current will exceed the Arduino's capacity, therefore breaking the Arduino board.

A double H-bridge was used as the simple motor driver; the model we use is the SN754410 Motor Driver IC. Below is the schematic:

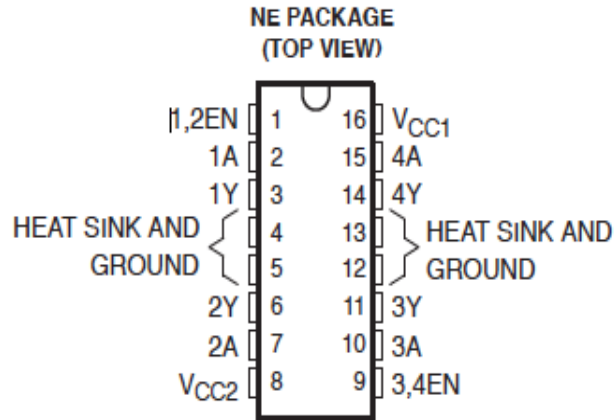


Figure 16: SN754410 Motor (Texas Instruments 1995)

**FUNCTION TABLE
(each driver)**

INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high-level, L = low-level

X = irrelevant

Z = high-impedance (off)

† In the thermal shutdown mode, the output is in a high-impedance state regardless of the input levels.

Figure 17: Motor Driver Function Table (Texas Instruments 1995)

For the power source, we will use 3 AA batteries for prototyping and testing. Below is a diagram for the stepper motor connection.

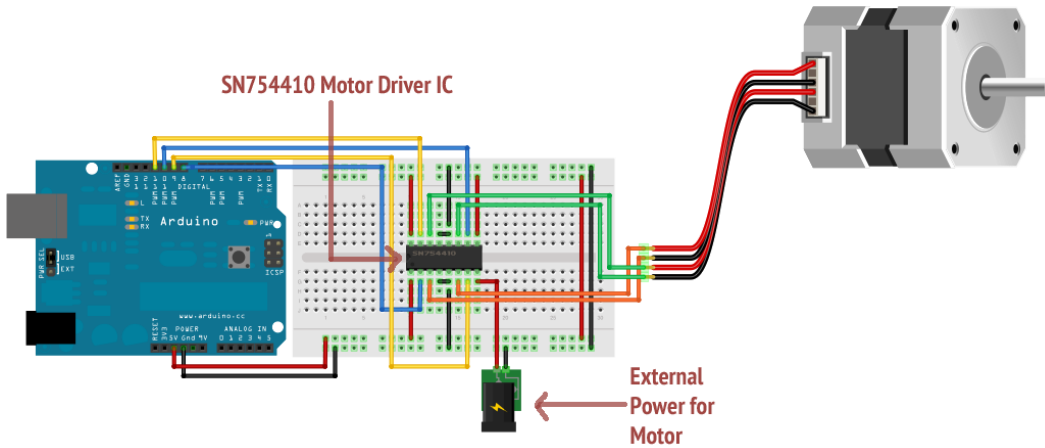


Figure 18: Stepper Motor Connection (Arduino 2014)

The Arduino has its own stepper motor library. However, this library only supports one stepper motor to be run at a given time; it cannot be used to run multiple stepper motor simultaneously. In order to run multiple stepper motor simultaneously, third party stepper motor library “AccelStepper” was obtained from GitHub (AccelStepper 2012).

The position, velocity and acceleration of the stepper motors can be changed by calling the AccelStepper functions:

`Stepper.moveTo (integer)`

`Stepper.setMaxSpeed (double)`

`Stepper.setAcceleration (double)`

The unit for these functions will be in steps, steps/second and steps/second².

User Interface

Because the Arduino is a simple electronic, it does not have a display or a terminal to interact with the human operator at runtime. In order to do user interaction, a communication channel needs to be built, and all the interaction patterns preprogrammed.

There are many ways to build the communication channel. One simple approach is to use a series of buttons, each one represent an ocean states. The advantage of this approach is that it is simple and very easy to build. The weakness is that it required too many input pins, which might not be available since 12 pins would be needed to drive the 3 stepper motors.

Another approach is to use a potentiometer and read a multiplier for each of the ocean states. For example, suppose the reading from the potentiometer is ranging from 1 to 10, the ocean state data (angular velocity, angular acceleration) for an ocean state will be:

$$(\text{Constant}) * (\text{Potentiometer_multiplier})$$

The advantages of this approach are:

1. Easy to build and program
2. Few pin required
3. Can simulate large number of ocean states
4. Easy to operate

The weaknesses of this approach are:

1. The exact data for each of the states is unknown
2. Fix pattern for each of the states

The third method is to connect a Bluetooth module to the Arduino board so that communication with the Arduino can be accomplished with any Bluetooth enable devices. The module that will be used is the HC-06 Bluetooth to serial adapter, it can be used to send and receive Bluetooth signal. Below is the connection diagram for this Bluetooth module:

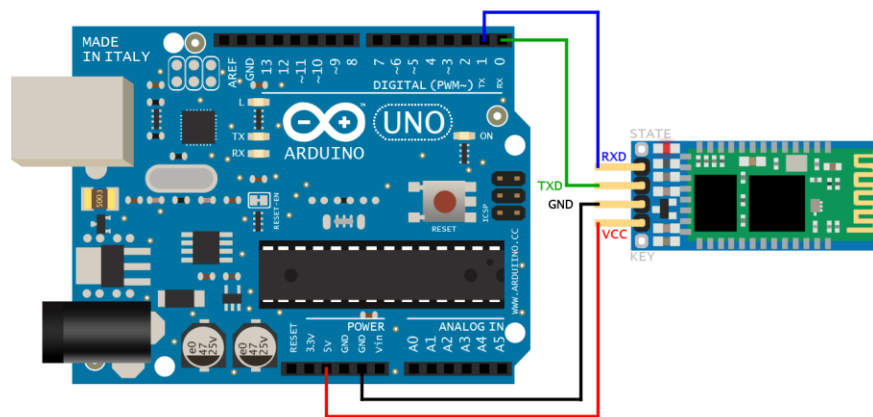


Figure 19: Connection Diagram for Bluetooth Module (AccelStepper 2012)

Smartphones can be used as the controller for the platform. Out of the many open source Bluetooth controller Apps the “Bluetooth Serial Controller” on an Android phone are the one that have been looked into for the controller. The controller can be programmed to send out different strings or characters as its signal. Integers 1 to 9 will be used for the Bluetooth signal, representing ocean state 1 to ocean state 9. The “R” signal is used as the reset button; every time the module receive the “R” signal, it will reset the platform to the initial position.

Stepper Motor Repositioning

The platform must be leveled every time the program starts so it can behave properly. The Arduino and the stepper motors do not have hardware or software support for this. The stepper motor cannot be moved to position zero because position zero will always be defined as the position when the motor starts by the AccelStepper library. Addition hardware is needed to define the fix zero position.

The hardware used to define the zero position is a simple photo resistor and a LED. The photo resistor will be fix to a point below the gear; an LED will be mounted to the gear. The image below shows how this looks like:

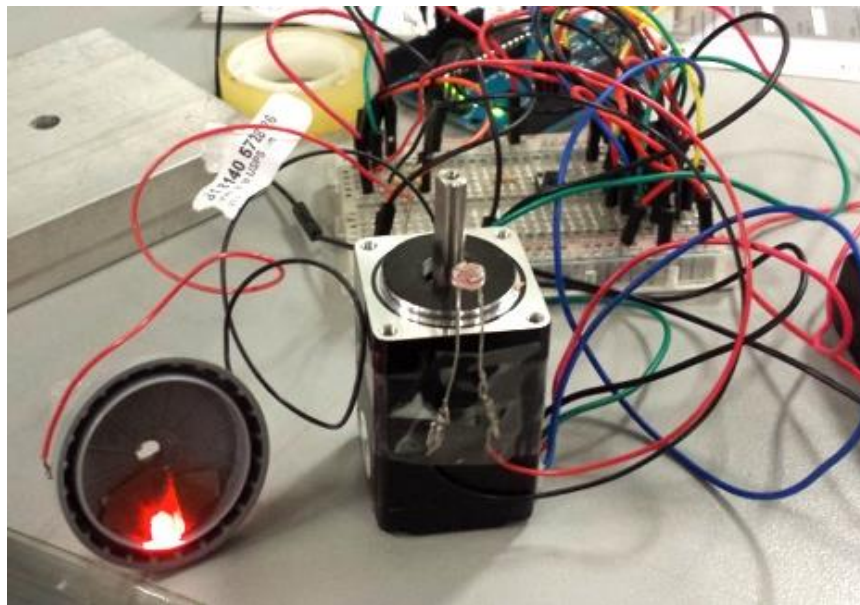


Figure 20: Initializing the Motors

Every time the motor starts, it will rotate one full revolution. As the gear is spinning, the photo resistor will continue to read the light level under the gear. Every time the photo resistor read a maximum value (when the LED is right above the photo resistor), it will remember the location where the maximum happen. Once the initializing revolution is finished, the Arduino will reverse the gear to the position where the maximum light level is read. This will be defined as the zero position.

Arduino code for this method

```
#include <AccelStepper.h>

AccelStepper stepper; // Defaults to 4 pins on 2, 3, 4, 5

int stepToGo;

int i;

int sensorMax = -1000;

int sensorPin = A0;

int sensorValue;

int stepBackward = 0;

int k = 1;

void setup(){

    Serial.begin(9600);

    stepper.setSpeed(100);

    stepper.moveTo(200);

    while(k == 1){

        stepper.run();

        stepToGo = (int)stepper.distanceToGo();

        if (stepToGo == 0){

            k = 0;

        }

        sensorValue = analogRead(sensorPin);

        Serial.println(sensorValue);
```

```

        if (sensorValue > sensorMax){
            sensorMax = sensorValue;
            stepBackward = stepToGo;
        }
    }

    stepper.setSpeed(-100);

    Serial.print("stepBackward: ");

    Serial.println(stepBackward);

    stepBackward = 200 - stepBackward;

    stepper.runToNewPosition(stepBackward);
}

```

Implementation of the Arduino

The final implementation of the platform is slightly different than the original design. First, the actual motors for the platform are unipolar stepper motors, which require a different motor driver than the original design.

The actual driver for the platform is the ULN2003A by Texas Instruments. They are high-voltage high-current Darlington transistor arrays. Each consists of seven npn Darlington pairs that feature high-voltage outputs with common-cathode clamp diodes for switching inductive loads. (Instruments 2004) For the configuration of the platform, 4 GPIO pins from the Arduino will be responsible for sending signal to the 4 input pins of the motor driver in order to control the stepper motor (pin 2 – 5 for motor 1, 6 – 9 for motor 2, 10 – 13 for motor 3). Below is a diagram for the configuration:

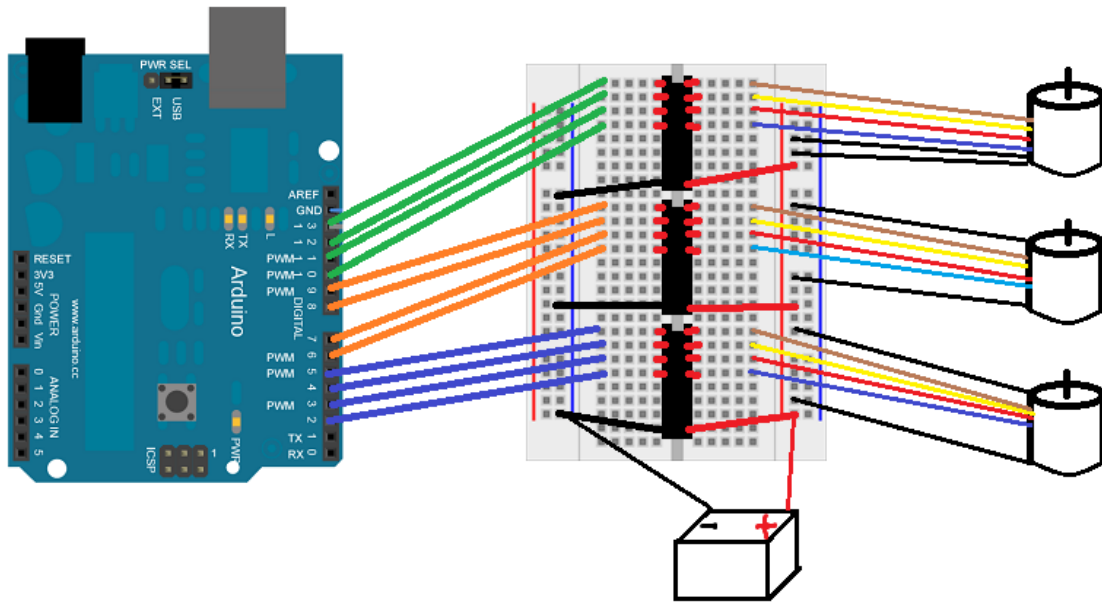


Figure 21: Configuration for Unipolar Motor Driver

To run the three motor simultaneously with different parameters, the Arduino will use the 3rd party open source library AccelStepper as discussed in earlier chapter.

In the original design, it includes a Bluetooth module that acts as the communication channel between the Arduino and the human operator. After consideration, this Bluetooth module will be replaced by a potentiometer in order to reduce cost and simplify the controlling procedure. Based on the position of the potentiometer, the Arduino will select the appropriate pre-loaded ocean state to execute. The below diagrams, taken from ITP Physical Computing will demonstrate the connection:

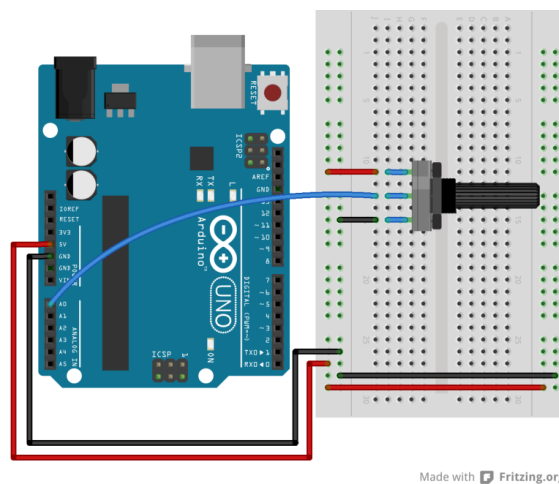


Figure 22: Arduino Uno Connection to Potentiometer (Using a transistor to control high current loads with an Arduino 2013)

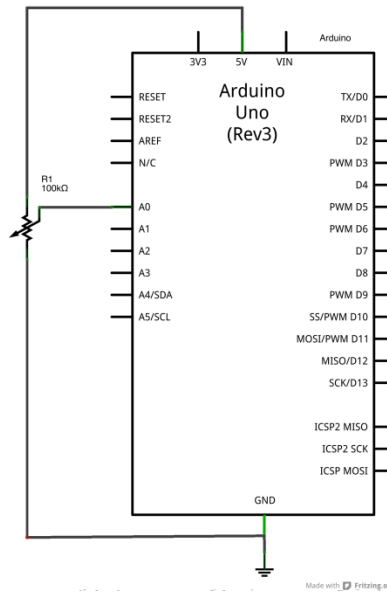


Figure 23: Arduino Uno (Using a transistor to control high current loads with an Arduino 2013)

Movement Pattern

The movement of the three motors is based on the simulation obtained from the Simulink model discussed in the previous session. The movement of different ocean state for each motor will be stored inside the Arduino program as an array of position and angular rate. The pattern will repeat itself once the cycle ends.

For future development, the team suggests to add an Arduino SD card shield for the Arduino, and store the ocean state parameters inside a SD card in different files. To read the data in the SD card in real-time, the Arduino can use the SD library. This method can create a more robust moving platform.

The Hardware and Software Limitation

According to the Simulink model, for calmer ocean state (e.g. wave height less than 15 meters), the different between the highest and lowest motor position is very small (~ 2 degree). For the current setup, it is very difficult to achieve this small change since the minimum movement for the motor is one step, which is 1.8 degree. For any smaller movement than 1.8 degree, it will require micro-stepping, which require additional hardware for the Arduino or very complex and precise programming.

For future development, the team suggests that replacing the hinge mounted on the motors with 2 gears with big gear ratio. Doing this has multiple advantages, first, big gear ratio reduces the torque requirement for the motors, and therefore the motors can lift the platform more easily. Second, using big gear ratio and placing the supporting rod in the mid-way of the big gear radius can make the platform move more smoothly and precisely for smaller changes. Therefore eliminate the need for micro-stepping.

Operation Manual

The platform is still in testing phase, therefore the circuit is connected with a breadboard. Before turning any power on, make sure that the motors connection is same as the diagram shown in the previous session, with the color in the right order. The black and white wires from the motors should be connected to the positive of the 12V power supply (In the diagram they are both shown in black).

Once they are connected properly, turn on the 12V power supply first. And then connect the 9V cell battery to the Arduino. Shortly after the 9V battery is connected, the platform should start its initialization process. In this process, all the three hinges will be repositioned to a horizontal position. Once the three hinges are in their horizontal position, the main loop of the program will start. And now the operator can select the wave intensity with the potentiometer.

To exit the program, just turn off the PC power supply and disconnect the 9V battery from the Arduino.

Conclusion and Recommendations

Researching and understanding the ocean dynamics and its effect onto an air carrier has been the highlight of this project. Although there are a few issues with the built motion platform, they can be easily fixed. The first problem is that the stepper motor chosen had a torque too weak with the maximum speed needed to lift the platform. The reason for this error came from miscommunication between the group members. Three correctly sized motor will fix this problem. The second problem is the ball joint that connects the motor horn to the rod. This joint allows unwanted motion perpendicular to the motor's rotation. This causes the platform to have an unwanted sway from side-to-side as it is lifted up and lowered down. Two quick-solve ways the group used to stop this sway was to connect the two back rods together so that they would sway together and not cause harm to the platform and to clamp the joint to the motor horn to limit the amount of motion in this unwanted direction. Though this will temporarily limit the impact of the problem, the future students to tackle this MQP should purchase a new joint that already constricts this movement. Throughout this project the group has gained a much better understanding of the programs Matlab and Simulink while looking through Fossen's program and making additions to fit the project's objectives. For the motor, it is advised to get a motor with smaller steps for example a step of 0.125° rather than 1.8° . This is because with larger steps, the platform will jerk as it hits each step during a slow movement rather than move smoothly. Should such a small-step motor cannot be bought, the original idea with gears would also remedy the jerking motion.

Works Cited

1995. *Texas Instruments*. November. Accessed February 26, 2014.
<http://www.ti.com/lit/ds/symlink/sn754410.pdf>.
2014. *Arduino*. Accessed February 26, 2014. http://arduino.cc/en/uploads/Tutorial/bipolar_bb.png.
2012. "AccelStepper." *GitHub*. Accessed February 26, 2014. <https://github.com/adafruit/AccelStepper>.
- Allan, Alasdair. 2013. *Arduino Uno vs BeagleBone vs Raspberry Pi*. April 15. Accessed 10 1, 2013.
<http://makezine.com/2013/04/15/arduino-uno-vs-beaglebone-vs-raspberry-pi/>.
- n.d. "Blade mCP X." *Horizon Hobby*. Accessed December 18, 2013.
www.horizonhobby.com/products/blade-mcp-x-bnf-BLH3580#t2.
2014. *Blade mCP X BNF*. Accessed February 18, 2014. <http://www.horizonhobby.com/products/blade-mcp-x-bnf-BLH3580>.
- Borta, R. T. 2002. Electric Motion Platform and a Control System for Controlling the Same. USA Patent 6,445,961 B1. September 3.
2007. *Design Trends: 'The Stepper Versus Brushless Servo Myth...'*. November. Accessed December 15, 2013. www.motion-designs.com/images/DTrends_Nov_2007.pdf.
- Fossen, T. 2002. *Marine Control Systems Guidance, Navigation, and control of Ships, Rigs, and Underwater Vehicles*. Trondheim: Marine Cybernetics.
- Instruments, Texas. 2004. *High Voltage, High Current Darlingtton Transistor Arrays*. November.
- Oh, S., K. Pathak, and S. K. Agrawal. 2005. "Autonomous Helicopter Landing on a Moving Platform Using a Tether." *International Conference on Robotics and Automation*. Barcelona.
- Roberts, Donna. 2012. *Normal Distribution*. Accessed February 26, 2014.
<http://www.regentsprep.org/Regents/math/algtrig/ATS2/NormalLesson.htm>.
- Roth, N., and B. Roth. 1996. *The Direct Kinematics of the General 6-5 Stewart Gough Mechanism*. Stanford University: Design Devision, Mechanical Engineering Department.
- n.d. "Series H23R Hybrid Stepper Motor." *Allied Electronics*. Accessed February 10, 2014.
http://www.alliedelec.com/images/products/datasheets/bm/HURST_MANUFACTURING/70030140.pdf.
- n.d. "Step Motor Basis." *Lin Engineering*. Accessed December 18, 2013.
www.linengineering.com/LinE/contents/stepmotors/pdf/Product_Guides/Lin_RG_StepMotorBasics.pdf.
- n.d. *The centerpiece of Navy operations: Aircraft Carriers*. Accessed March 5, 2014.
<http://www.navy.com/about/equipment/vessels/carriers.html>.

Tormes, Felix R. 1974. *Disorientation Phenomena in Naval Helicopter Pilots*. Pensacola: Naval Aerospace Medical Research Laboratory.

n.d. *T-REX 500*. Accessed March 5, 2014. http://www.aligntrexstore.com/T-REX-500_c_12.html.

2013. "Using a transistor to control high current loads with an Arduino." *ITP Physical Computing*. September 21. Accessed February 27, 2014. <http://itp.nyu.edu/physcomp/Tutorials/HighCurrentLoads>.