6-2018

Design of a Robotic Cownose Ray

Mark Marzotto

Design of a Robotic Cownose Ray

Mark Marzotto, Professor William Keat Ph.D.
Union College, Mechanical Engineering Department, Schenectady NY

Table of Contents

Abstract

Nature often inspires advances in science and technology and a promising example of this is mimicking locomotion of underwater animals. The main ways in which manmade aquatic vehicles propel themselves are water jets and propellers, which are inconsistent with how underwater animals swim. The cownose ray displays very efficient thrust with low frequency strokes by using a combination of oscillations and undulations with its pectoral fins. The goal of this paper is to mechanically replicate the actual fin motion of the cownose ray in order to capture these attractive features. The design will account for both the oscillations by designing a cabling mechanism that simulates muscle and the undulations by using a flexible shaft to control the twist present in the fin. Integrating these two mechanisms will achieve an accurate model of the true motion.

1. Introduction

Many technological advances in engineering are inspired by objects seen in nature such as structures, mathematical series, and animal movements. The cownose ray demonstrates very efficient thrust [1], low frequency fin strokes, and an extremely streamlined body shape, all of which are desirable attributes for underwater vehicles. In addition to being more efficient, underwater locomotion that does not use propellers is desirable for applications such as stealth that need to reduce the noise that is produced from motor vibrations.

The category of batoid fishes, commonly called rays, includes many different types of rays that swim using different manners of propulsion. The types of batoids, in addition to their appearances, can be distinguished by their locomotion. They all utilize a different combination of oscillations and undulations, characterized by varying frequency, amplitude, wave speed, wave

number, stride length, and phase velocity [2]. For this project, the cownose ray (*Rhinoptera bonasus*) is chosen specifically. This ray's locomotion is very efficient and propulsive, which makes it desirable to mimic [2].

A cownose ray is shown in Figure 1.1 and its swimming motion in Figure 1.2. The two pectoral fins of the ray are located laterally on each side of the ray's body and achieve the swimming motion. This motion is a combination of oscillations and undulation. Oscillations are defined as the flapping motion and undulations are defined as the fluttering motion, or the presence of waves in the fin. From a mechanical design perspective, the oscillations are achieved through bending and the undulations through twisting. Some rays that dwell on the ocean floor and do not need high propulsion utilize a more undulatory motion, but the cownose ray, out of all of the batoids, has the greatest oscillation to undulation ratio [2]. This contributes to the high efficiency and propulsion.

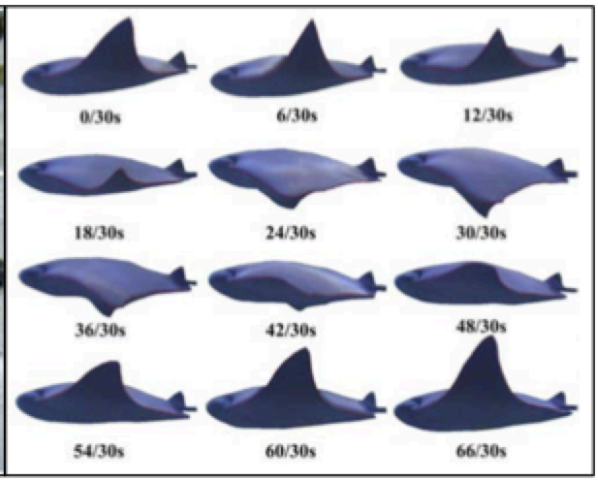


Figure 1.1: (Left) Cownose Ray [2] **Figure 1.2: (Right) Cownose Ray swimming motion [3]**

One aspect of the fin motion that is not characterized is the curvature of the pectoral fins throughout the swimming motion. Their fins are entirely muscle and thus can achieve very flexible positions and motions, but this feature of the movement is often overlooked in existing

attempts at creating mechanical rays. Figure 1.3 shows the extent of curvature in a swimming cownose ray.

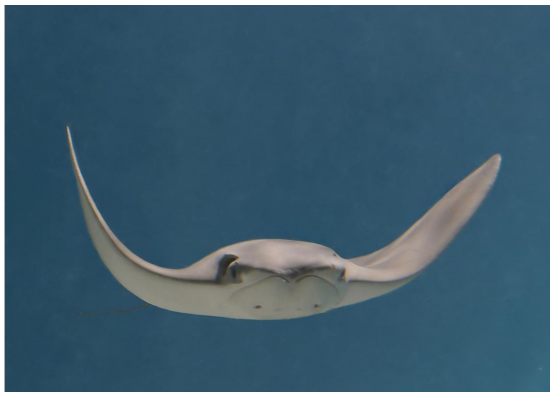


Figure 1.3: Cownose ray swimming with clear curvature present in the pectoral fin [4]

Two main paths have been examined by other researches in mimicking the Cownose Ray's fin motion. The first is by using mechanical actuators and the second by using electroactive polymers (EAP's), also called artificial muscles, which bend when a voltage is applied across them. Both approaches have seen success in some key aspects they set out to mimic, but have not captured the entire fin motion accurately.

In this paper, the swimming motion of actual cownose rays will be analyzed. The previous attempts at creating robotic stingrays will then be reviewed and discussed. Alternative methods of achieving oscillation and undulation will be detailed. Then the formulation of a mathematical model of the swimming motion that is true to the actual motion will be described and finally a physical prototype built that abides to the mathematical model.

2. Analysis of the Swimming Motion

The swimming motion of a cownose ray has been analyzed by researchers such as Rosenberger, but only parameters of interest are reported, such as frequency, amplitude, wave speed, wave number, stride length, and phase velocity [2]. These values for the cownose ray are listed in Table 2.1.

Table 2.1: Key parameters of interest of cownose ray's swimming motion [2]

Parameter	Value	Description / Units
Fin Beat Frequency (f)	1.04	Number of fin-beat cycles per second [Hz]
Mid-disc Amplitude	0.35	Half the dorsoventral displacement of the widest portion of the fin, standardized by disc width
Wavespeed (c)	2.62	Distance wave travels in the middle one-third of the fin divided by the time taken for it to move that distance, standardized by disc length
Wave Number	0.40	Length of fin base divided by the wavelength
Stride Length (U/f)	60.18	Swimming speed (U) divided by frequency (f) [cm]
Phase Velocity (U/c)	0.76	Forward swimming speed (U) divided by wavespeed (c)

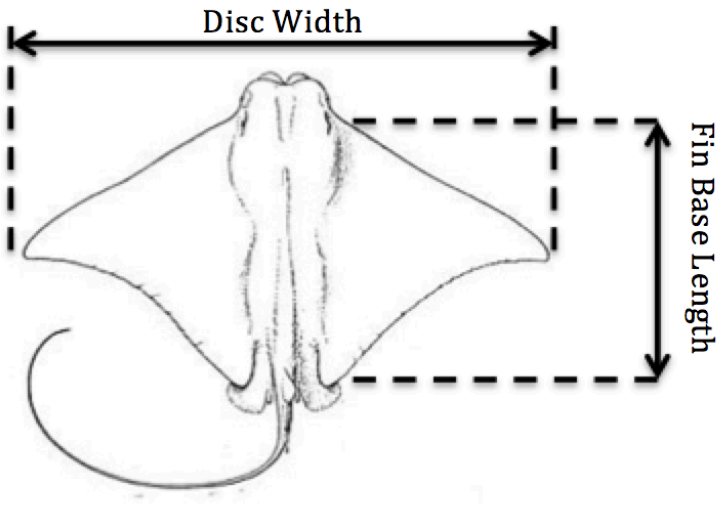


Figure 2.1: Cownose ray fin anatomy terminology

These parameters are helpful in formulating design requirements, but do not provide enough information to model the entire fin motion. In order to obtain data of the fin displacement at any time during a cycle of the fin propulsion, video data of a cownose ray swimming was analyzed [5]. A portion of video of a cownose ray swimming at the New England Aquarium was selected where the ray is swimming straight at the camera and the length of the clip is one cycle of the swimming motion. The clip was then slowed down and three points along the fin were traced at 14 time steps throughout the motion. These points are shown in Figure 2.2.

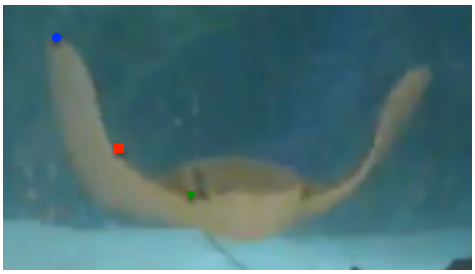


Figure 2.2: Video screenshot of cownose swimming with three points identified.

The vertical displacement of these points was measured and then both the displacement and the corresponding time were converted to true scale. To do this, the 14 seconds for one cycle was scaled to 2 seconds and the displacement was scaled so that the maximum amplitude is 12.8 inches. The maximum amplitude of 12.8 inches comes from manipulating the mid-disc amplitude from Table 1. Mid-disc amplitude is defined as half the dorsoventral displacement of the widest portion of the fin, standardized by disc width so multiplying this vale by 2 and by the disc width of 18.3 inches gives the full amplitude [2]. Figure 2.3 shows these three points plotted through one complete cycle of swimming motion and polynomials may be fit to the data to fully characterize the fin motion.

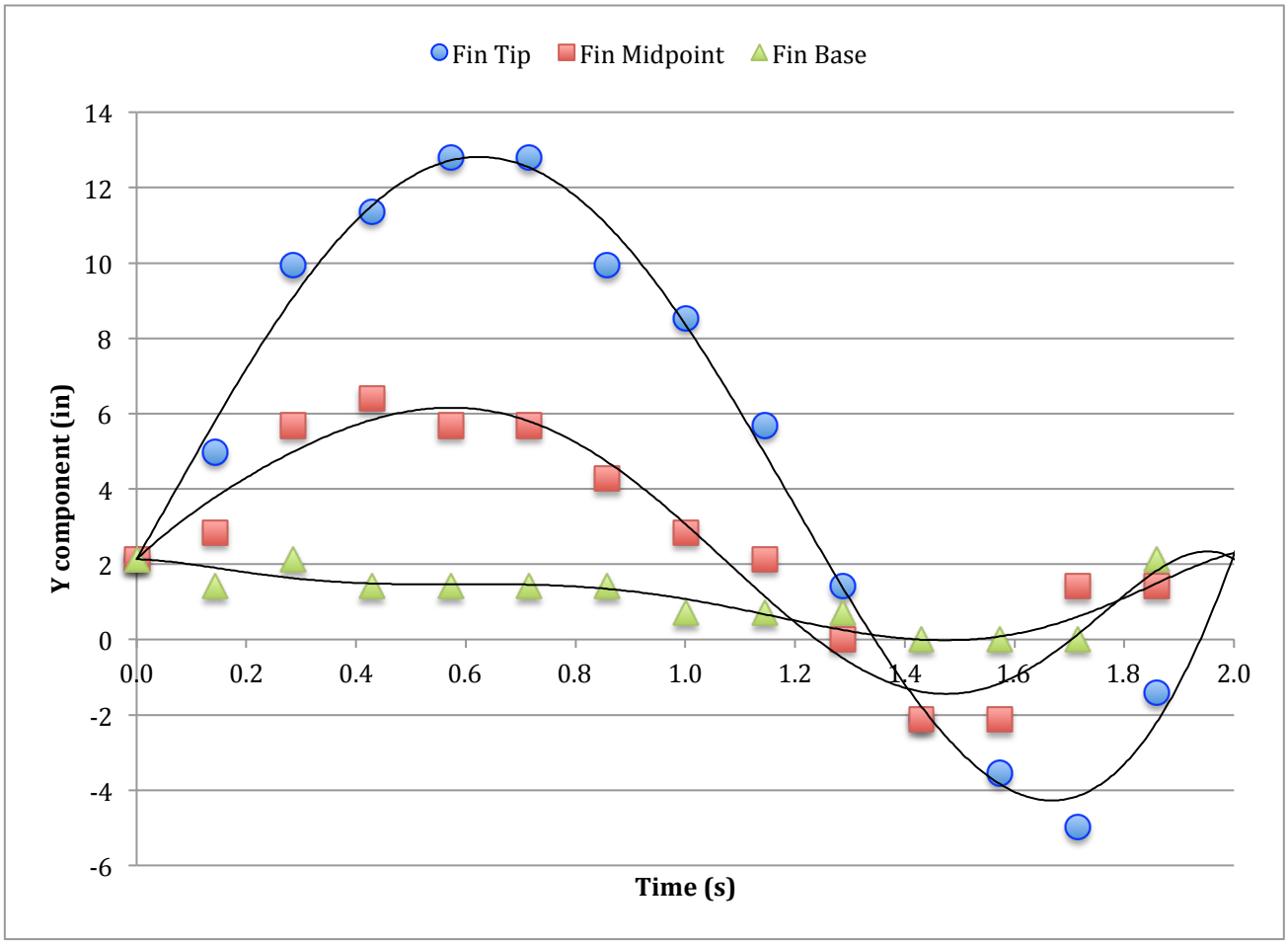


Figure 2.3: The three points on the pectoral fin (shown in Figure 4) plotted throughout one cycle of swimming motion.

Equation 1.1, 1.2, and 1.3 characterize the fin tip, midpoint, and base point respectively, where y is the vertical displacement and x is time.

$$y = -2.01x^6 + 6.95x^5 + 9.55x^4 - 40.51x^3 + 6.79x^2 + 25.48x \quad (1.1)$$

$$y = -9.97x^6 + 48.10x^5 - 75.80x^4 + 46.16x^3 - 21.37x^2 + 13.8x \quad (1.2)$$

$$y = -4.63x^6 + 26.07x^5 - 51.68x^4 + 42.95x^3 - 13.34x^2 - 0.42x \quad (1.3)$$

With these equations, the mechanical cownose fin location can be compared to them to check for accuracy and they may be used if certain displacements are needed to design any aspects of the mechanisms.

The undulation portion of the motion must also be characterized in order to check accuracy of the mechanical model. Rosenberger uses the parameter wave number for this, which is defined as the length of fin base divided by the wavelength [2]. This means that if there is one full wave present on the fin, equal to the length of the fin base, the wave number would be equal to 1. More than one wave present on the fin results in a wave number larger than 1 since the wavelength is less than the length of the fin base and less than one full wave results in a wave number less than one since the wavelength is greater than the length of the fin base. The cownose ray has a wave number of 0.4, meaning the wavelength is 40% greater than the fin base length. Figure 2.4 shows varying degrees of undulation in a few different batoid rays, where column 'C' shows a cownose ray.

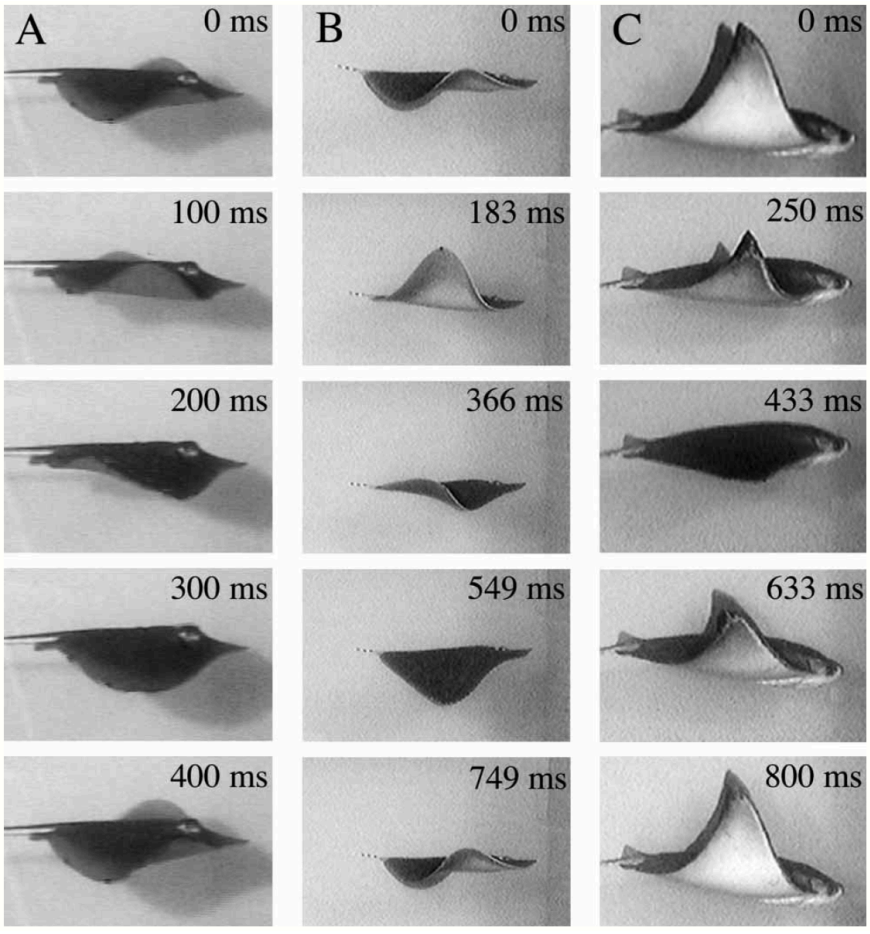


Figure 2.4: Three different batoid rays with varying degrees of undulation shown. (A) Wave number of 1.31. (B) Wave number of 0.63. (C) Wave number of 0.4.

3. Alternative Mechanical Methods for Producing the Swimming Motion

3.1. Introduction

With an understanding of the fin's motion throughout the swimming cycle, a mechanical method of achieving this complete motion must be chosen and designed. All of the methods must include some number of motors to actuate the movement, but there are several ways to use motors to produce different movements. This section will detail how the designs of current cownose ray robots attempt to mimic the swimming motion. It will also discuss alternative mechanical concepts for both achieving the oscillations and undulations. Finally, the methods that were chosen will be explained further.

3.2. Alternative Mechanical Implementations

3.2.1. Current Attempts

Existing robotic rays can be broken into two main categories based on how they actuate the fins. The first achieves the fin motion by utilizing material properties. Chen et. al, for example use electroactive polymers (EAP's), also called artificial muscles [1]. When a specified voltage is applied across these polymers, they deflect (Figure 3.2.1). The maximum deflection then is matched to the profile of the fin at the top of the swimming motion. The front portion of the fin is constructed with this material and the trailing portion of the fin is made of a passive membrane that allows for the motion in the lead edge to propagate through the fin (Figure 3.2.2). This combination achieves the oscillation and undulation motion.

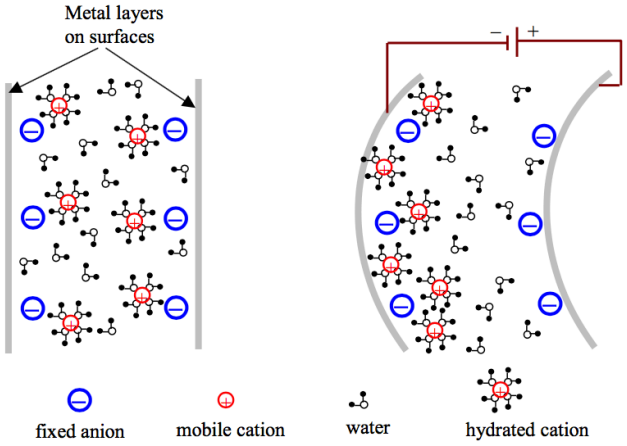


Figure 3.2.1: Cross section view of actuation mechanism

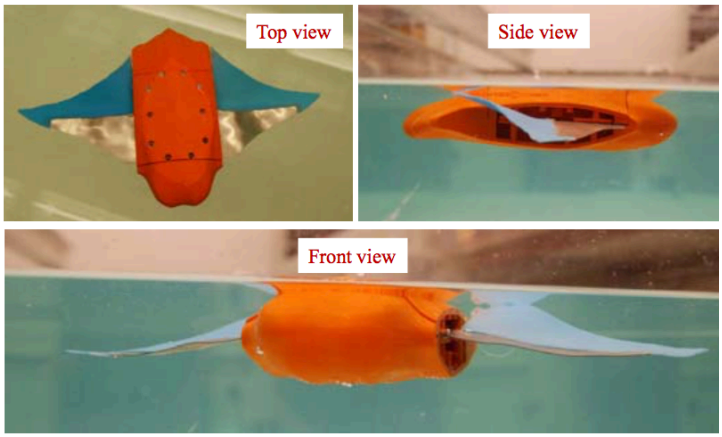


Figure 3.2.2: Robot with both sections of the fin visible.

This design relies heavily on material selection. The voltage can be altered to change the amplitude, but the shape of the deflection and the wave propagation is passively controlled by the material properties. This means that the fin location at intermediate parts of the swimming motion and the degree of undulations cannot be controlled. In addition, this method requires a smaller scale robot, because of the high loads from the water, so it is not true size.

The other category contains mechanical actuators and the distinction often lies in how the undulation is integrated into the oscillation motion. Both Ma et. al [3] and Yang [2] utilize at least one motor with a rigid shaft to control the oscillations, but differ in how they control the undulations. Ma et. al attaches the entire fin to a dc motor and passes a flexible shaft through fin cross-sections and attaches it to the fin tip. The motor that controls this flexible shaft is located on the fin. As the fin is oscillating up and down due to the DC motor, the flexible shaft controls the twisting of the fin (Figure 3.2.3).

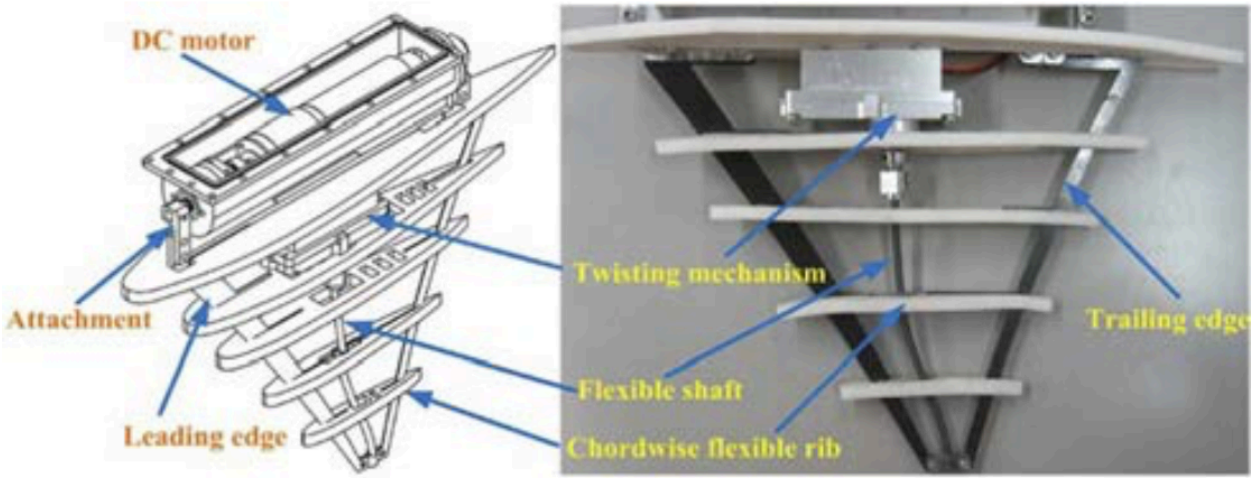


Figure 3.2.3: Mechanical method of achieving swimming motion with DC motor and flexible shaft [3]

This method integrates the two motions well, allowing them to be independently controlled, but it cannot control the curvature of the fin. The leading and trailing edge material, along with the material that covers the fin will dictate how the fin bends or curves.

Yang et. al uses a series of servo motors with a specific timing to simultaneously control the undulation and twisting (Figure 3.2.4).

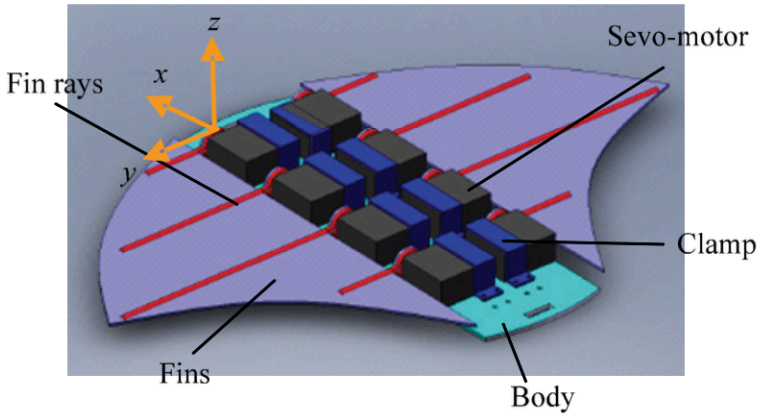


Figure 3.3.4: Mechanical method of achieving swimming motion by timing servo-motor-actuated shafts (fin rays) [6].

The design contains 4 shafts on each fin that actuate upwards and downwards to prescribed amplitudes and the phase shift between them creates a wave through the fin. While this design

can control the undulations precisely, it also does not achieve the curvature that actual cownose rays display. In order to fully capture the swimming motion, the oscillation and undulation motions must both be produced, but the fin curvature must be included in the oscillatory motion.

3.2.2. Discussion of Alternative Oscillating Concepts

To achieve the proper oscillations, taking into account both amplitude and curvature, other mechanical methods must be examined. The first method expands on the methods of Yang et. al and Ma et. al. If instead of a rigid or semi-flexible shaft being the only element attached to the servo-motor at the fin base, several servo-motors may be linked together. Similar to how a curve is composed of many small straight lines, the fin could approximate a curve by using these segments (Figure 3.2.2.1).

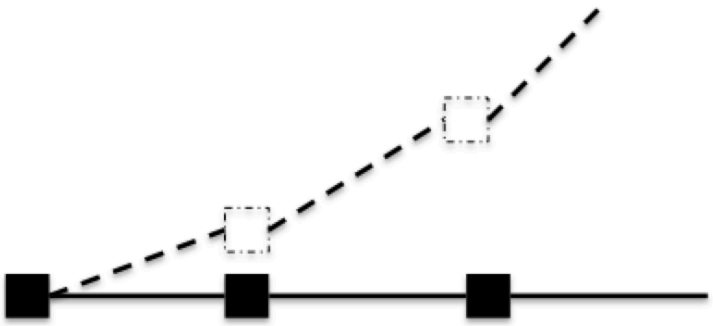


Figure 4.2.2.1: Servo-motors with rigid shafts in series to approximate the needed curvature.

This method requires a larger number of motors and timing calculations so that the motion is smooth. Also, because of size requirements, the fin segments would not be able to be small enough to produce a smooth curve.

Another mechanical method of producing the oscillations is with the use of four-bar linkages. These have been used in actuating wings by researchers such as Bejgerowski et. al [7] (Figure 3.2.2.2).

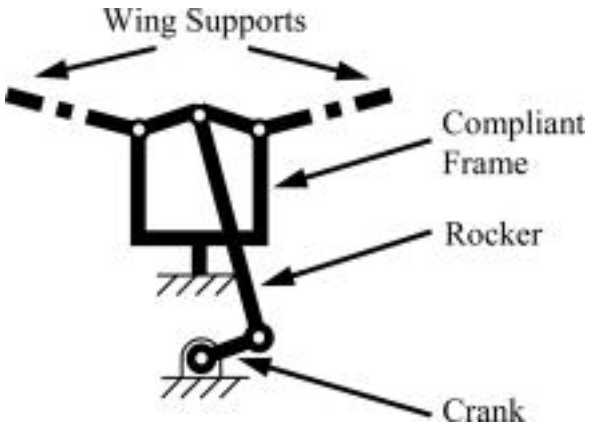


Figure 3.2.2.2: Crank and rocker four bar linkage to actuate wings [7]

When the crank is attached to a DC motor, the end of the rocker actuates the wings so that they flap. This produces a fluid continuous motion that does not rely on any restoration affects, but rather is actively controlling the motion at all time through the motion. This does not account for the needed curvature so a different method of achieving the curvature would need to be integrated into the fins. This additional method, along with the size of four-bar linkages in general may be too large to fit on a life-size cownose ray.

The last mechanical method of producing the oscillatory motion while capturing the curvature is with a cabling mechanism. Cabling mechanisms still use a motor, but instead of controlling the position of a shaft, they lengthen and shorten cables or wire that when attached to parts in various manners, can actuate the part. This can be viewed as how a muscle in your body that is attached to a bone moves the bone when it is contracted. With muscles on both sides of a bone and joint, the combination of contracting and expanding can move the bone in both directions. Cabling mechanisms work based on the same principles and are an effective method for mimicking muscle movement. These mechanisms are used in animatronics, especially in the movie industry, and in automation lines very commonly.

Cabling takes advantage of the concept of beam bending. Attaching the cable to a beam in certain spots creates couples on the beam. This couple applies pure bending to the beam,

which creates a curvature. A more flexible material will deflect more and have a smaller radius of curvature than a stiffer material. These two concepts together allow for flexible motions to be created mechanically. Figure 3.2.2.3 shows an example of a cabling mechanism.

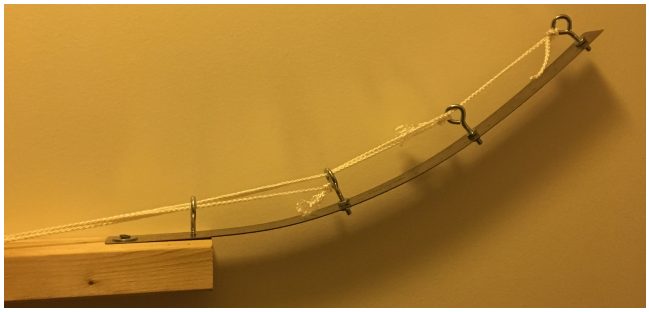


Figure 3.2.2.3: Cabling mechanism that introduces pure bending to the beam when the cable is contracted.

Applying a force to the string applies a couple to the eyelet that the string is anchored to, causing the metal beam to bend. By controlling the location of the couples, their magnitude, and the material properties of the beam, a desired motion can be achieved. Having a second cabling mechanism paralleled on the bottom of the beam would allow for control of the beam in both directions, which is needed for the fin oscillations.

3.2.3. Discussion of Alternative Undulating Concepts

Alternative undulation concepts must also be compared to ensure an accurate production of the swimming motion. The first method considered is passively through material selection. This is the same as how Chen et. al choose a material for the fin that would allow the motion initiated in the front edge to propagate through the fin. While this is a very smooth and natural motion, choosing a material that passively emulates a cownose ray's fin with a high degree of accuracy is a difficult challenge that is coupled to the internal design of the actuating mechanisms.

The second method is to have several motors along the base of the fin that actuate shafts or cabling mechanisms. This is the method used by Yang et. al (Figure 3.2.4). Timing the different motors so that they create a wave in the fin is a more controllable and deterministic method of producing the motion, but requires a lot of motors and hardware. The undulation and oscillation are also controlled by the same mechanism, which limits independent control.

The final method is by using a flexible shaft. This is seen in the robotic stingray produced by Ma. Et al. A motor near the fin base can be attached to a flexible shaft that runs to the tip of the fin. The properties of the flexible shaft allow it to be bent, so that it can comply with the needed curvature, without sacrificing its torsional strength. Actuating the angular position of the fin tip provides the required undulation. This is achievable by only controlling one point because there is less than half a wave present on the fin at a time [2].

3.3. Conclusions

The main considerations in choosing between oscillation and undulation methods were accuracy of motion replication, number of parts, robustness, and if the two motions could be independently controlled. Decision matrices for both oscillation and undulation are shown in Appendix A. The method that most accurately reproduces the oscillation with the correct curvature is the cable-actuated beam. The flexible shaft provides the highest degree of undulation control. These two methods when used in combination are independent, compatible, and produce the most accurate swimming motion.

4. Models of Cable-Actuated Large Displacement Beam Bending

4.1. Mathematical Model of the Cable-Actuated Beam

In order to control the cable-actuated beam so that it both reaches the correct maximum amplitude and has the correct shape profile throughout the swimming motion, a mathematical model is needed. This model must output vertical displacements along the beam based on beam material, geometry, and the couples present along the beam. Linear and non-linear models of beam deflection were developed.

The linear model of cable-actuated beam bending comes from Euler's Equations and the concept of superposition for beam deflection. The beam was modeled as cantilevered with three equally-spaced couples along the beam. The deflection that results from double integrating the internal moment equation is given in Equation 4.1.1 which relates beam displacement to position, where y is vertical displacement, x is x-position, c_i is the i^{th} couple along the beam, E is modulus of elasticity, and I is area moment of inertia.

$$y = \frac{1}{EI}\left[\frac{1}{2}(c_1 + c_2 + c_3)x^2 - \frac{1}{2}c_1 < x - \frac{L}{3} >^2 - \frac{1}{2}c_2 < x - \frac{2L}{3} >^2\right] \qquad (4.1.1)$$

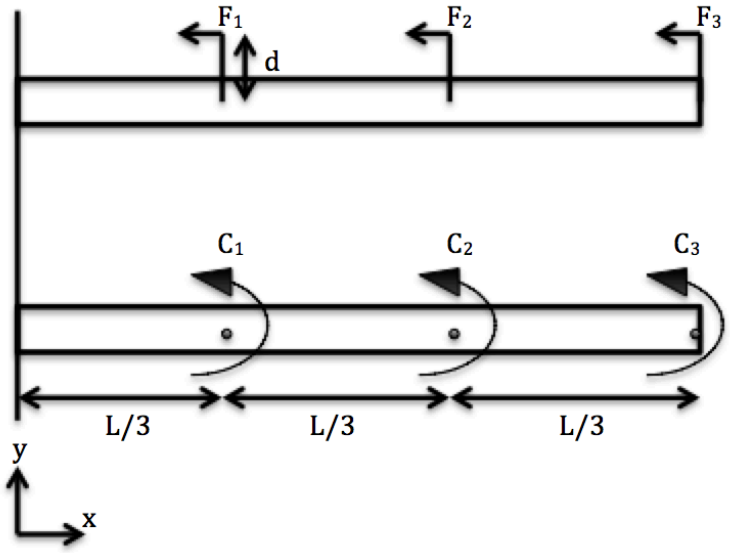


Figure 4.1.1: Linear model schematic with three equally spaced couples produced by a horizontal force and a moment arm.

This model works on the assumption that the beam is only displacing a small amount. This assumption is not consistent with the displacements that the fin must experience, but is still used as a starting point. The benefit of this model is that it is linear so the effects of multiple couples can be superimposed to obtain the total deflection.

The non-linear model also relates vertical displacement to the x-component. It segments the beam based on the locations of the couples and finds the radius of curvature associated with each segment (Equation 4.1.2) and the change in angle over the segment (Equation 4.1.3). R_i is the radius of curvature of the ith segment, M_i is the constant internal bending moment over the ith segment, E is the modulus of elasticity, I is the area moment of inertia, and L is the length of the beam (see Figure 4.1.2).

$$R_i = \frac{EI}{M_i} \tag{4.1.2}$$

$$\theta = \frac{\frac{L}{3}}{R_i} \tag{4.1.3}$$

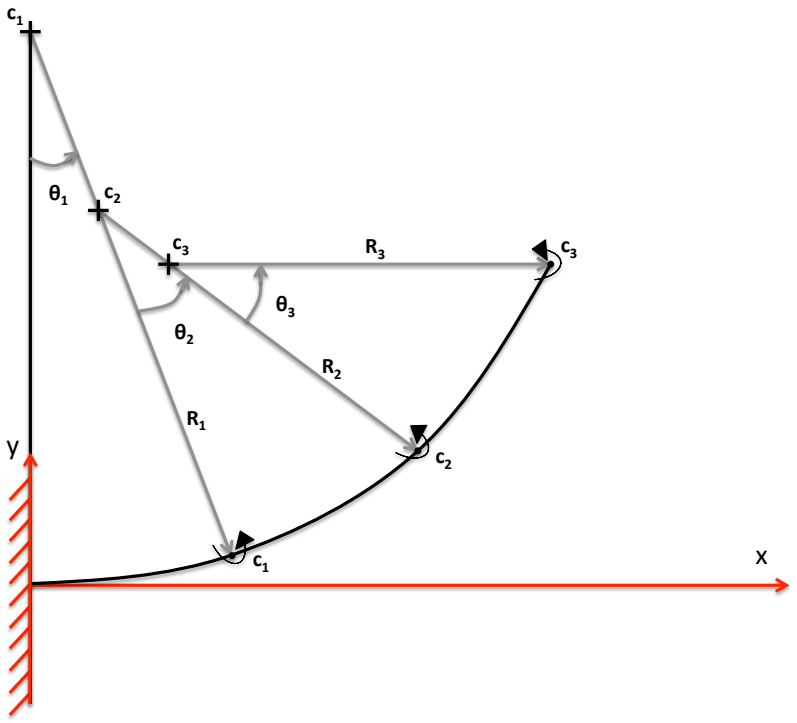


Figure 4.1.2: Beam deflection composed of piece-wise curvatures defined by a radius and angle

Creating a piecewise function of these radii and interpolating displaced coordinates along the circular arcs results in the non-linear model. This model is not limited to small displacements. Equations 4.1.4 and 4.1.5 show how the coordinates are interpolated along the second arc as an example.

$$x = x_{c_2} + R_2 \sin(\theta_1 + \Delta\theta_2) \tag{4.1.4}$$

$$y = y_{c_2} - R_2 \cos(\theta_1 + \Delta\theta_2) \tag{4.1.5}$$

Here x_{c_2} and y_{c_2} are the coordinates of the center of curvature of the second segment and $\Delta\theta_2$ is a value between 0 and θ_2 that locates the point along the second arc.

4.2. Physical Prototype of the Cable-Actuated Beam

With two mathematical models of the cable-actuated beam in hand, a physical prototype was built to test the accuracy of the models. Figure 3.2.2.3 is the prototype, which is an 18 inch ruler cantilevered on a block of wood. 15 inches of the ruler are free hanging. The ruler has three eyebolts equally spaced along the 15 inches with a piece of string tied to each one. When the strings are contracted, the beam deflects upwards, with the magnitude of displacements depending on which strings are being pulled and how much.

4.3. Results

The material and geometry properties of the cable-actuated prototype are used with the linear and non-linear model to calculate theoretical deflection curves. Vertical displacements of the prototype were also recorded and plotted together with these curves to check the consistency between the mathematical models and actual motion of the cable-actuated beam (Figure 4.3.1). Figure 4.3.1 shows the generated non-linear and linear deflection curves, as well as the

experimental beam displacements, with only one end couple MATLAB code for both mathematical models is shown in Appendix B.

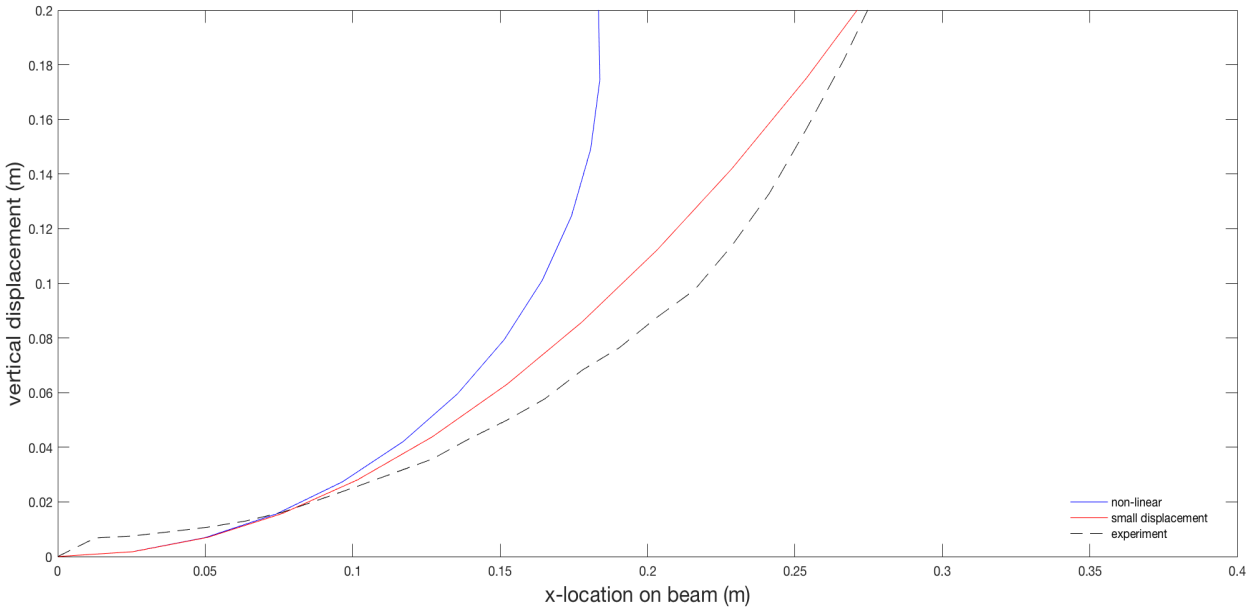


Figure 5.3.1: Linear model, non-linear model, and prototype vertical displacement at the max deflection position, with only a couple applied at the fin tip.

4.4. Conclusions

The figure shows that neither the linear nor non-linear models accurately model the prototype geometry. This suggests that the forces acting on the beam are more complex than just a moment at the free-end of the beam. The downward force at the free end, the weight of the fin, and the lifting forces where the cable pulls upwards on the cross-sections could have significant effects on the fin profile. It is also possible that the code is not properly calculating the beam location. A new method of generating these curves is being developed to more accurately predict the beam deflection curve.

5. Fin Prototype Design and Sizing

5.1. Fin Geometry

As both of a stingray's fins are the same and move simultaneously with no phase change, only one fin was prototyped. The prototype, as seen in Figure 5.1.1, consists of a rigid "spine," with a flexible beam cantilevered perpendicularly from it. There are 4 air-foil shaped cross sections along the flexible beam that have holes above and below the beam for the cables pulling the beam, an off-centered hole to the side of the top cable hole for the flexible shaft, and two holes on the tips that maybe used for covering the fin in the future. The last cross-section's flexible shaft hole has the same shape as the keying on the end of the flexible shaft. These cross-sections are shown in Figure 5.1.2 and the SolidWorks drawings are shown in Appendix C.

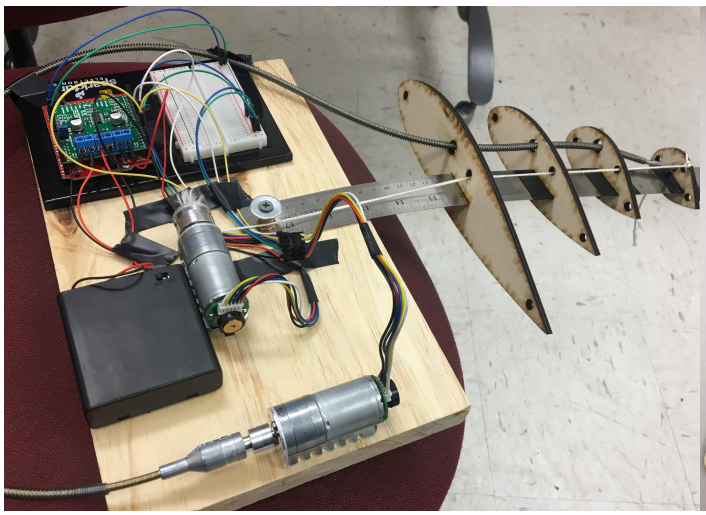


Figure 5.1.1: Prototype of one cownose ray fin

Figure 5.1.2: Fin cross-section supports

The prototype excludes the motor and cable on the bottom of the fin to pull the fin downwards. As this prototype is being tested only in air, the restoration forces of the beam along with gravity returns the fin to the starting position without needing the opposing motor.

The flexible shaft used has a collar for a motor shaft and an output metal rod with a keying on it. The shaft size of the motor selected for the undulations is smaller than the hole on the flexible shaft collar, so an aluminum adapter was fabricated to couple the motor shaft to the flexible shaft (Figure 5.1.3).

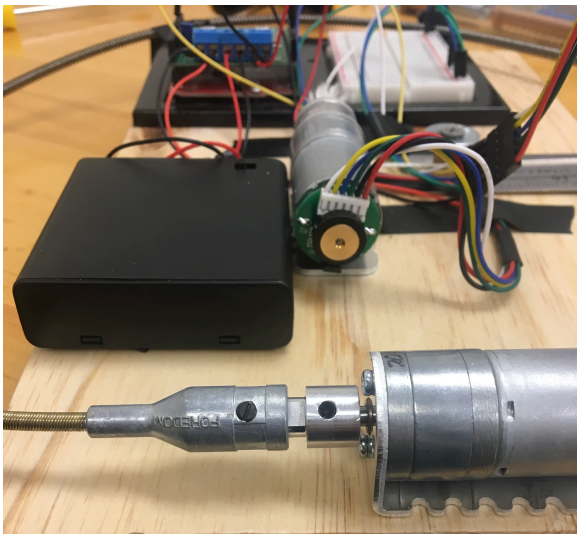


Figure 5.1.3: Aluminum adapter that couples flexible shaft to motor shaft

5.2. Cable-Actuated Oscillation Motor Sizing

As seen in Figure 5.1.1, the motor controlling the cable that pulls on the flexible beam is located on the rigid "spine" perpendicular to the cable. A scooter wheel adapter hub is used as a spool that the cable is attached to (Figure 5.2.1).

Figure 5.2.1 – Pololu Aluminum Scooter Wheel Adapter for 4mm Shaft used as spool [8]

The end cap for the adapter hub is screwed down and the cable fed through the setscrew hole on the small diameter section of the hub. The cable is then twisted around the cap screws to secure it. This spool diameter, used to calculate the torque needed in the motor, is 18 mm.

With the combination of a moment and downward force at the end of the beam where the cable is connected, upward lifting forces where the cable pulls up on the intermediate cross-sections, and the non-uniform weight along the beam, it is difficult to predict the force required to deflect the beam so the tip is vertical. Instead, a force gauge was attached to the cable and pulled until the tip was vertical. The measured force for this is 25 N. Dimensions of the prototype are shown in Figure 5.2.2, where the beam is 0.02 inch thick stainless steel.

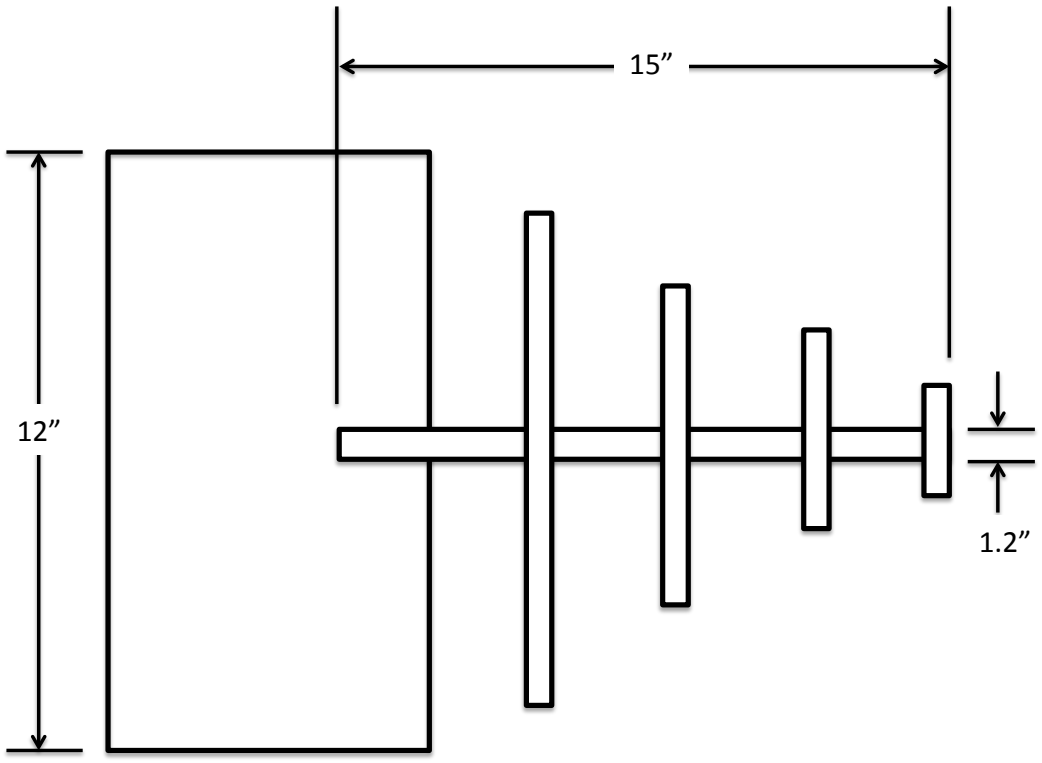


Figure 5.2.2 – Overall dimensioned schematic of prototype with stainless steel beam

With the force and diameter of the spool known, the required torque and power, assuming a 1 second cycle, was calculated (Appendix D). A motor from Polulu was then chosen and shown in Figure 5.2.3, with specifications provided in Table 5.2.1.

Figure 5.2.3: Polulu gear motor with encoder [9]

Table 5.2.1: Polulu Gear Motor with Encoder Specifications [9]

Stall Torque	33 oz-in
No Load Angular Velocity	290 rpm
Operating Voltage	6 V
Shaft Diameter	4mm

5.3. Undulation Motor Sizing

The flexible shaft used to twist the fin was chosen before the motor was sized. To measure the torque applied to the flexible shaft collar to twist the fin, the prototype was assembled with everything except the motor. The flexible shaft collar was then placed on a fixed horizontal shaft so that it could be twisted. Finally, a cable was attached to the collar, wrapped around it, and pulled on with a force gauge. This applied a couple to the flexible shaft, which was attached to the final cross-section, thus twisting the fin. The measured force to achieve the required twisting was also found to be around 25 N, with a smaller diameter shaft, so the same motor was chosen for consistency.

5.4. Results

When the fin prototype was assembled with the motors, the required twisting was achieved, but the required beam deflection was not. The stall torque of the motor with the wheel adapter was tested separately from the prototype and it was found that the motor had an experimental stall torque of 900 g-cm, instead of the 2400 g-cm listed in the specifications.

5.5. Conclusions

To achieve the required beam deflection, the flexible beam was redesigned. The original beam, shown in Figure 5.1.1, is an 18 in stainless steel ruler, approximately 1.2 inches wide and 0.02 inches thick. An aluminum beam of the same length, 0.6 inches wide, and 0.016 inches thick replaced the stainless steal beam. This reduces the area moment of inertia by approximately 72% and the stiffness modulus by approximately 33%. With this change, the fin was able to be properly controlled with the motors and reached all of the maximum deflections required. The stiffness was perhaps reduced too much, however, as the fin at rest was slanted too low down below the midsection of the stingray.

6. Prototype Control

6.1. Motor Control

The motors on the fin prototype are controlled and driven by an Arduino Uno board and a motor driver. The motors operate at 6 V so a 6 V battery pack is used to power the Arduino board and motor driver. The motor driver is a dual motor shield that is designed to be mounted to an Arduino board and to be able to drive 2 motors. The motor driver is used for two reasons. First, the Arduino board only has a 5V output, but the motor shield allows for a higher output voltage. Second, the H-bridge integrated into the driver allows for the DC motor to be spun in both directions without having to change the positive and negative ends of the motor. Figure 6.1.1 shows the motor driver and Arduino board. In addition, Polulu sells the motor driver and recommends it be used with the selected motors, so they are compatible.

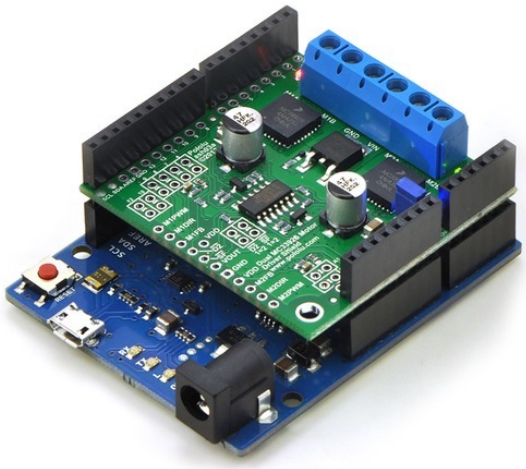


Figure 6.1.1: Arduino board with dual motor driver [10[

6.2. Motion Control

The Arduino code used to control the motor outputs consists of two tasks: reading the motor encoders and spinning the motors.

6.2.1. Reading the motor encoders

In order to continuously keep track of the motors positions while running the code that spins the motors, interrupt pins on the Arduino board are used. The Arduino Uno board has two interrupt pins that may be used to jump out of the main loop of the code and run different commands when something defined happens. For keeping track of the motor positions, the interrupt pins recalculate the position every time the encoder changes value.

The encoders on the motors have two signals allowing them to be quadrature encoders. Attaching both of these to the two interrupts allows the Arduino code to recalculate the position when either signal changes. When this signal changes, the code checks to see if the first changed pin switches from HIGH to LOW or LOW to HIGH, and then checks the other signal as well. With this information, the direction of the motor is known and 1 count is either added or subtracted to the current position. Figure 6.2.1 shows a visual of this. For example, if Channel A changes, the code checks to see if it goes from “1” to “0” or “0” to “1”. If it is “1” to “0,” it then checks Channel B. If the motor is spinning clockwise, Channel B will be “1,” but if it is spinning counter clockwise, Channel B will be “0,” per Figure 6.2.1. With clockwise defined as the positive direction, “1” is added to the current position of the motor in counts if Channel B is “1.” Using the counts per revolution specifications of the encoder, the position of the motor is converted to degrees and can be printed to the serial port if needed.

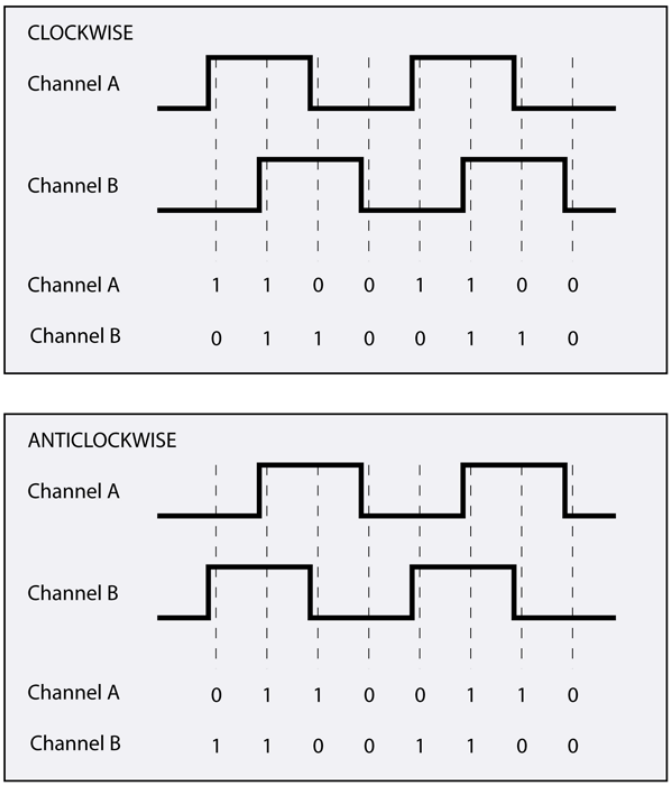


Figure 6.2.1: Two signals of the encoder and how to determine direction from them [11]

Since the fin prototype has two motors with encoders, the Arduino board must read four signals. With only two interrupt pins, each encoder has one signal on an interrupt and one signal on a non-interrupt analog pin. The code then only checks for one of the signals to change to recalculate the position of the motor, which halves the resolution of the encoders, but is still accurate enough for this application. The Arduino code that implements this is shown In Appendix D.

6.2.2. Spinning both motors

Both motors, used for the oscillations and undulations, are spun at the same time, but different amounts. To do this, if-else loops are set up for both motors to power them while their current positions are less than their desired amounts. When the positions reach their desired values, the motors are stopped. If the motor input tries to switch from its HIGH to LOW value

suddenly like this, the motor overshoots its desired position. To account for this, the input signal to the motor is not constant. As the position of the motor approaches its desired angle, the voltage fed to the motor is reduced so that it has 0 voltage when at its desired value. This smoother transition from HIGH to LOW eliminates the overshoot, but introduces a new problem. There is a level of non-zero voltage that is not enough to spin the motors; therefore, the motor will stop spinning before the final angle is reached. A controller value is added to the motor input signal to correct this. The value of this control is experimentally determined and is specific to the system and to the specific stage of the motion. For example, there is a different control value for the same motor when it is spinning clockwise and when it is spinning counterclockwise. With these controller values experimentally determined, they are implemented and it is observed that the motor stops at the desired positions reliably. After the motors reach their desired values for the portion of the swimming motion they are performing, the Arduino code steps to similar code for the next portion of the motion. The four parts of the motion are shown in the schematic in Figure 6.2.2. See Appendix E for the full Arduino Code.

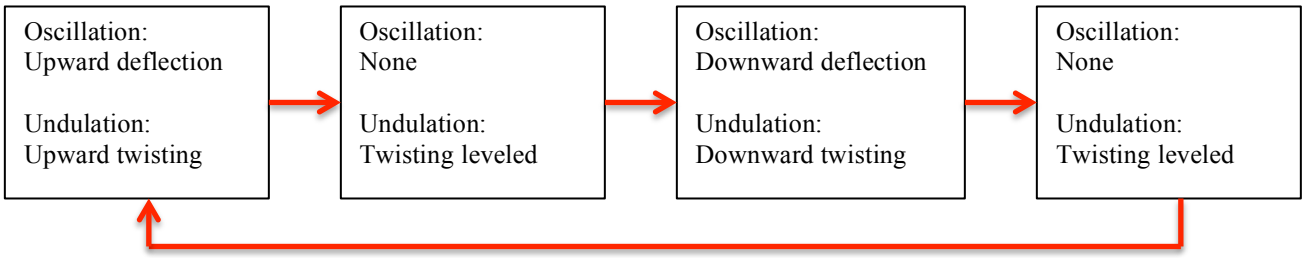


Figure 6.2.2: Swimming motion chronological steps. Arduino code consists of nested if/else statements to step through and loop these steps

6.3. Results

Implementing the voltage error signal with a controller value to the motor input signal is an accurate and precise method of spinning the motors of the prototype to desired angles. The motor driver also effectively works at delivering the desired voltage and spinning both motors in both directions.

In addition, the fin prototype does operate correctly and provide a consistent, looped swimming motion. Appendix F shows a progression of the fin through this motion, taken from a video of the prototype in operation.

6.4. Conclusions

If the prototype changes – a different beam geometry, different motors, or a skin is added for example - the system changes. This does not effect reading the encoders or the logic of the code, but the controller values used to stop the motors at the right positions throughout various parts of the code will need to be updated by printing the positions to the serial port and altering the values until the actual position is the same as the desired position.

7. Conclusions

The fin prototype was successful in demonstrating the ability to create a robotic stingray fin that has the capability of mimicking the motion of an actual ray. The achieved motion was not compared to the true motion for accuracy, but the mechanisms used to control the oscillations and undulations may be altered to refine the fin motion.

The prototype with the aluminum beam was able to reach the maximum deflection needed and fin twisting of up to 60° in both the clockwise and counter clockwise directions. In addition the Arduino code that was implemented, which reads the encoders, accurately controls the motors, spinning them to their desired values ± 1°. It also continuously loops through the full swimming motion without accumulating any error in the motor positions.

With the prototype of one fin working, the future work for this project consists of refining this motion and advancing in the direction towards getting a stingray in water. Specific tasks include comparing the shape of the fin curvature to that of a real cownose ray and adjusting various system parameters until these align. It is already known that the fin droops too low while at rest so that is one aspect that needs to be addressed. Another goal is to cover the fin with a flexible skin material. Related to this is also waterproofing the controls of the system. My spring term goal is to refine the motion as accurately as possible and to cover the fin with a skin.

Appendix A: Decision Matrices for Oscillation and Undulation Methods

Table A.1: Decision Matrix for Oscillation Methods				
		Options		
Criteria	**Weight**	Multiple Servo Motors	Four-Bar Crank and Rocker	Cable-Actuated Mechanism
Motion Accuracy	5	7	5	9
Number of Parts	2	3	6	8
Robustness	1	5	8	8
Control	2	9	6	9
Total	**10**	**64**	**57**	**87**

Table A.2: Decision Matrix for Undulation Methods				
		Options		
Criteria	**Weight**	Passively (Material)	Series of Servo Motors	Flexible Shaft
Motion Accuracy	5	9	7	8
Number of Parts	2	9	4	8
Robustness	1	8	6	7
Control	2	2	8	9
Total	**10**	**75**	**65**	**81**

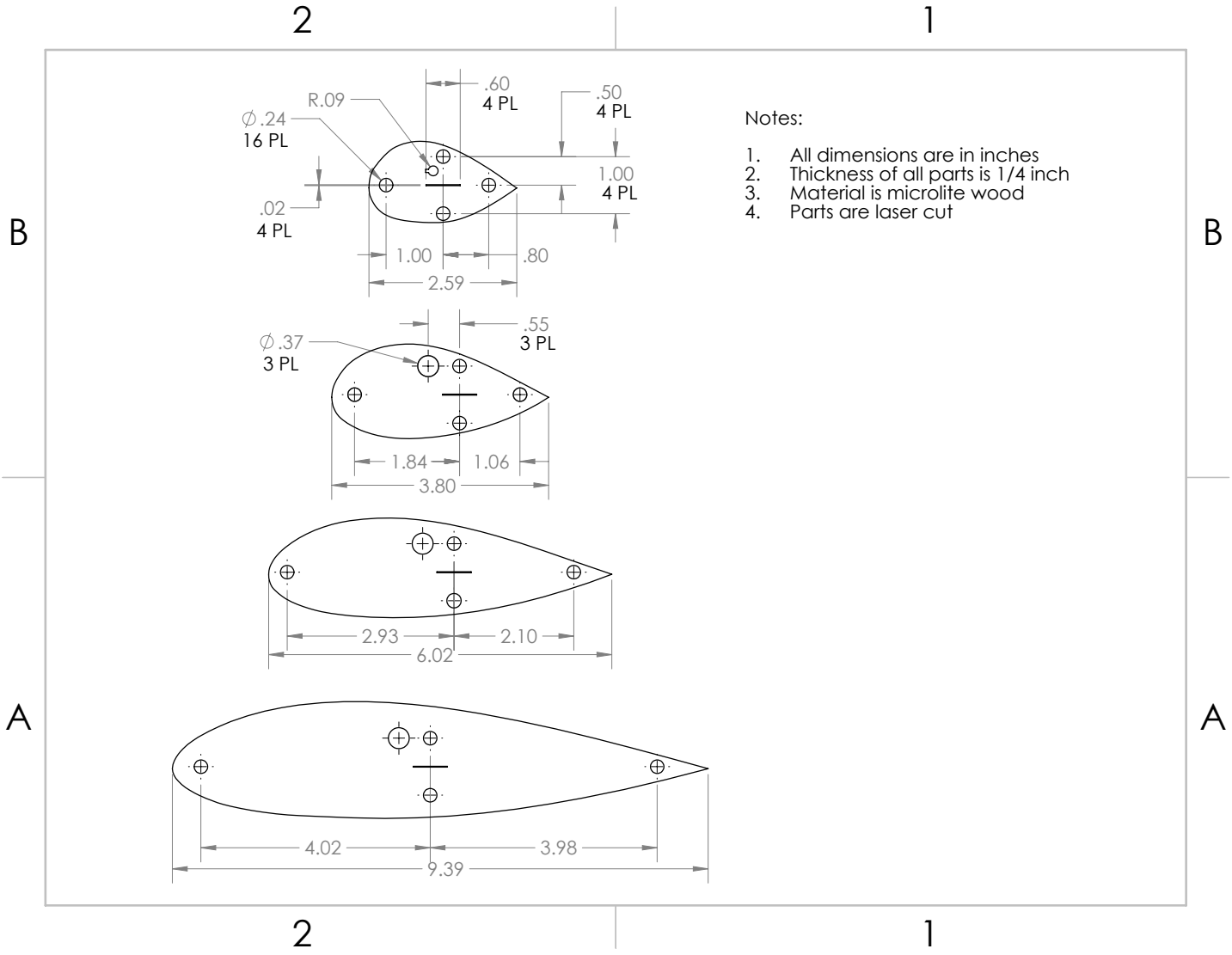
Appendix B: MATLAB Code for Linear and Non-Linear Models

```
%nonlinear beam model assumes piecewise circular arcs induced by
%a series of couples along the beam

%inputs
E=2.0E11;    %elastic modulus
I=1.323E-12;   %area moment of inertia of beam cross-section
L=0.381;       %total length of the beam
NC=3;       %number of equally-space couples along the beam
C(1)=0.6;    %mag of 1st couple (closest to the wall;CCW is pos)
C(2)=0.6;    %mag of 2nd couple
C(3)=0.6;    %mag of 3rd couple (at the tip)
Nint=5;     %number of equally spaced intervals between couples

%calculate the length of each beam segment
Lseg=L/NC;

%calculate internal bending momemts along the beam
Msum=0;
for i=NC:-1:1
    Msum = C(i) + Msum;
    M(i)=Msum;
end

%calculate radii of curvature for each beam segment
for i=1:NC
    R(i)=E*I/M(i);
end

%calculate the change in angle (in radians) over the each arc
for i=1:NC
    theta(i)=Lseg/R(i);
end

%-------------------------------------------------------------------------
%calculate the non-linear deformed geometry, one segment at a time
%-------------------------------------------------------------------------

%coordinates of first point
x(1)=0.;
y(1)=0.;
cnt=1;

%coordinates of first center of curvature
XC(1)=0.;
YC(1)=R(1);

%calculate coordinates along the 1st arc;
dtheta=theta(1)/Nint;
for i=1:Nint
    cnt=cnt+1;
    x(cnt)=XC(1) + R(1)*sin(i*dtheta);
    y(cnt)=YC(1) - R(1)*cos(i*dtheta);
end
```

```
%coordinates of 2nd center of curvature
XC(2)=XC(1) + (R(1)-R(2))*sin(theta(1));
YC(2)=YC(1) - (R(1)-R(2))*cos(theta(1));

%calculate coordinates along the 2nd arc;
dtheta=theta(2)/Nint;
for i=1:Nint
    cnt=cnt+1;
    x(cnt)=XC(2) + R(2)*sin(theta(1)+i*dtheta);
    y(cnt)=YC(2) - R(2)*cos(theta(1)+i*dtheta);
end

%coordinates of 3rd center of curvature
XC(3)=XC(2) + (R(2)-R(3))*sin(theta(1)+theta(2));
YC(3)=YC(2) - (R(2)-R(3))*cos(theta(1)+theta(2));

%calculate coordinates along the 3rd arc;
dtheta=theta(3)/Nint;
for i=1:Nint
    cnt=cnt+1;
    x(cnt)=XC(3) + R(3)*sin(theta(1)+theta(2)+i*dtheta);
    y(cnt)=YC(3) - R(3)*cos(theta(1)+theta(2)+i*dtheta);
end

%-------------------------------------------------------------------------
%calculate small displacement deflections for comparison purposes
%-------------------------------------------------------------------------

%generate x-coordinates
xs(1)=0.0;
dx=L/(NC*Nint);
for i=1:NC*Nint
    xs(i+1)=dx*i;
end

%calculate small displacement deflections
for i=1:NC*Nint+1
    add1 = 0.5*(C(1)+C(2)+C(3)) * (xs(i))^2;
    if(x(i)>Lseg)
        add2 = -0.5*C(1)*(xs(i)-Lseg)^2;
    else
        add2=0;
    end
    if(x(i)>2*Lseg)
        add3 = -0.5*C(2)*(xs(i)-2*Lseg)^2;
    else
        add3=0;
    end
    ys(i) = 1/(E*I) * (add1 + add2 + add3);
end

%-------------------------------------------------------------------------
%experimental proof of concept
%-------------------------------------------------------------------------
xpoc = [0,0.0127,0.0254,0.0381,0.0508,0.0635,0.0762,0.0889,0.1016,0.1143...
     ,0.127,0.1397,0.1524,0.1651,168275];
```

```
ypoc = [0,0.0047625,0.0079375,0.0127,0.0206375,0.0301625,0.041275,0.060325...
     0.0777875,0.104775,0.136525,0.17145,0.212725,0.276225,0.3063875];

%-------------------------------------------------------------------------
%plot the elastic curves
%-------------------------------------------------------------------------
%   blue line = non-linear result
%   red line  = linear, small displacement result
%   x = experimental POC

plot(x,y,'b')
hold on
plot(xs,ys,'r')
plot(xpoc,ypoc,'kx')
legend('non-linear','small displacement','experimental','location','Best')
legend boxoff
ylabel('vertical displacement')
xlabel('x-location on beam')
axis([0 0.4 0 0.4])

%as a check, add up the lengths of the segments
Lsum=0.;
for i=1:cnt-1
    Lsum=Lsum + sqrt( (x(i+1)-x(i))^2 + (y(i+1)-y(i))^2 );
end
Lsum
```

Appendix C: Fin Cross-section Drawings



Appendix D: Motor Sizing Calculations for Oscillations

Oscillations

Force Required (N)	Diameter of spool (m)	Required Torque (N-m)	Torque (lbm-in)	Torque (g-cm)	T (oz-in)
25	0.018	0.225	1.99	2294	31.9
Time per Cycle (s)	Cable distance (m)	Angle Radians	Angle Degrees	Angular Speed (rad/s)	
1.04	0.0254	2.82	161.70	5.43	
Required Power (W)	Power (lbm-in/min)				
1.22	**103.12**				

Selected Motor

Tstall (oz-in)	Tstall (g-cm)	Tstall (lbm-in)	Wnl (RPM)
33	2376.26	2.06	290
Tm/(0.5*Tstall)	1.25*Wnl*Tstall (lbm-in/min)		
1.93	746.98		

Appendix E: Arduino Code used to Control Fin Prototype

```
//Mark Marzotto
//2017-2018 Senior Project
//Mechanical Stingray Control Code

//Needed for motor driver
#include "DualMC33926MotorShield.h"
DualMC33926MotorShield md;

//Creating global variables

//Variables used to keep track of encoder/motor position
long count1=0;
long count2=0;
long deg1=0;
long deg2=0;

//Variables to control motor voltage input
long speed1=0;
long speed2=0;

//controller values to improve voltage error signal
long control1=85;
long control2=70;
long control1_2=-45;
long control2_2=-85;
long control2_reset=-70;
long control2_reset2=45;

//variables used to control logic
long direction1=0;
long direction2=0;
long directionA=0;
long directionB=0;
long reset1=0;



void setup() {
  //Attaching interupts to pins.
  //Pin 2 connects to encoder1 signalA, Pin3 connects
  //to encoder2 signal B
  attachInterrupt(0,AChanged,CHANGE);
  attachInterrupt(1,BChanged,CHANGE);

  //Starting serial communication
  Serial.begin(115200);
```

```
  //For motor driver
  md.init();
}

void loop(){
  // The main loop reads encoder values and spins the 2 motors to produce the
  // fin motion.
  // loop() is interrupted every time an input changes on pin 2 or pin 3
  // The motion consists of 4 stages:
  //1. Fin twists back and goes up
  //2. Fin levels twist to neutral position
  //3. Fin twists forward and goes down
  //4. Fin levels twist to neutral position


  //Convert the encoder value from counts to degrees
  //979.2 counts per revolution per encoder specs
  deg1=((count1)/979.62)*360;
  deg2=((count2)/979.62)*360;

  //Record position and speed of motors to serial port
  Serial.print("Degrees1: ");
  Serial.print(deg1);
  Serial.print(", Degrees2: ");
  Serial.print(deg2);
  Serial.print(", Speed1: ");
  Serial.print(speed1);
  Serial.print(", Speed2: ");
  Serial.print(speed2);
  Serial.println();


  //beginning of motion control
  //Step 1: Twisting motor to 60° and pulling motor to -115°

  if(direction1==0 || direction2==0){
    if(deg2<60){
      //decelerating voltage signal with controller
      speed2=((60.0-(deg2))/60.0)*300.0+control2;
      md.setM2Speed(speed2);
      direction2=0;
    }
    else{
      md.setM2Speed(0);
      direction2=1; //this motor has reached its desired value
      deg2=60; //Set to desired value in case of overshoot
    }
```

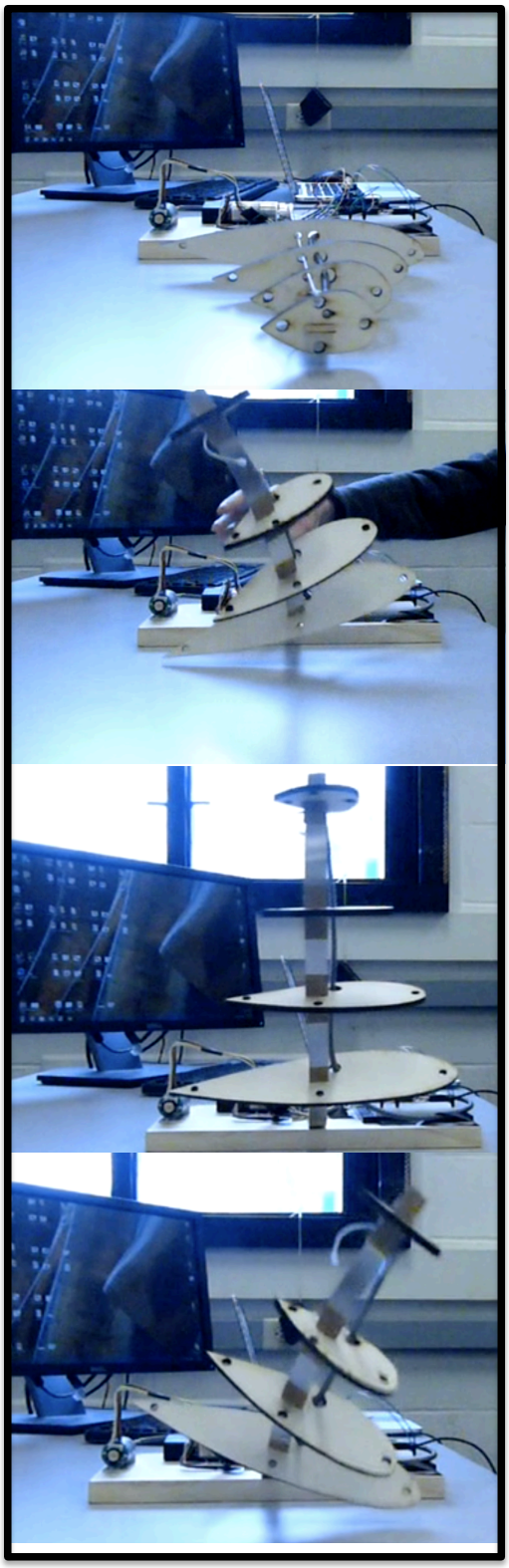
```
    if(deg1>-115){
      //decelerating voltage signal with controller
      speed1=((-115.0-(deg1))/-115.0)*300.0+control1;
      md.setM1Speed(speed1);
      direction1=0;
    }
    else{
      md.setM1Speed(0);
      direction1=1; //this motor has reached its desired value
      deg1=-115; //Set to desired value in case of overshoot
    }
  }

  else{

  //Step 2: Level twisting Motor to 0 degrees

    if(reset1==0){
      if(deg2>0){
        //decelerating voltage signal with controller
        speed2=((deg2)/60.0)*-200.0+control2_reset;
        md.setM2Speed(speed2);
      }
      else{
        md.setM2Speed(0);
        reset1=1; //this motor has reached its desired value
        deg2=0; //Set to desired value in case of overshoot
      }
    }

//Step 3: Twisting motor to -50 degrees and pulling motor to initial //level
position (0 degrees)

    else if(directionA==0 && reset1==1 || directionB==0 && reset1==1){
      if(deg2>-50){
        //decelerating voltage signal with controller
        speed2=((-50.0-(deg2))/-50.0)*-300.0+control2_2;
        md.setM2Speed(speed2);
      }
      else{
        md.setM2Speed(0);
        directionA=1; //this motor has reached its desired value
        deg2=-60; //Set to desired value in case of overshoot
      }

      if(deg1<0){
        //decelerating voltage signal with controller
        speed1=((deg1)/-115.0)*-100.0+control1_2;
```

```
        md.setM1Speed(speed1);
      }
      else{
        md.setM1Speed(0);
        directionB=1; //this motor has reached its desired value
        deg1=0; //Set to desired value in case of overshoot
      }
    }

  //Step 4: Level twisting motor to 0 degrees

    else{
      if(deg2<0){
        //decelerating voltage signal with controller
        speed2=((deg2)/-60.0)*200.0+control2_reset2;
        md.setM2Speed(speed2);
      }
      else{
        md.setM2Speed(0);
        deg2=0; //Set to desired value in case of overshoot

        //reset logic variables to initial values to make 1st IF
        //statement true again
        direction1=0;
        direction2=0;
        reset1=0;
        directionA=0;
        directionB=0;
      }
    }
   }
}

//When the Motor 1 encoder changes position need to find out which direction
it is moving and either add or subtract 1 count
void AChanged(){
  int A;
  int B;
  A=digitalRead(2);
  B=digitalRead(5);
  if(A==HIGH){
    //A is high, check B
    if(B==HIGH){
      //A and B are both high
      count1=count1-1;
    }
    else{
      //A is high, B is Low
```

```
      count1=count1+1;
    }
  }
  else{
  //A is low, check B
      if(B==HIGH){
      //A is low and B is high
      count1=count1+1;
    }
    else{
      //A is low, B is Low
       count1=count1-1;
    }
  }
}  //end AChanged

//When the Motor 2 encoder changes position need to find out which
//direction it is moving and either add or subtract 1 count
void BChanged(){
  int A;
  int B;
  A=digitalRead(3);
  B=digitalRead(6);
  if(A==HIGH){
    //A is high, check B
    if(B==HIGH){
      //A and B are both high
      count2=count2-1;
    }
    else{
      //A is high, B is Low
      count2=count2+1;
    }
  }
  else{
  //A is low, check B
      if(B==HIGH){
      //A is low and B is high
      count2=count2+1;
    }
    else{
      //A is low, B is Low
       count2=count2-1;
    }
  }
}  //end BChanged
```

Appendix F: Progression of Swimming Motion from Video Footage

Left: Upward progression of fin oscillation
Right: Undulation progression through full swimming cycle



References

[1] Chen Z, Um TI, Zhu J, Bart-Smith H. Bio-Inspired Robotic Cownose Ray Propelled by Electroactive Polymer Pectoral Fin. ASME. ASME International Mechanical Engineering Congress and Exposition, Volume 2: Biomedical and Biotechnology Engineering; Nanoengineering for Medicine and Biology ():817-824. doi:10.1115/IMECE2011-64174.

[2] Rosenberger, LJ. "Pectoral Fin Locomotion in Batoid Fishes: Undulation Versus Oscillation." Journal of Experimental Biology, vol. 204, no. 2, 2001, pp. 379-394.

[3] Ma, HW, et al. "A Biomimetic Cownose Ray Robot Fish with Oscillating and Chordwise Twisting Flexible Pectoral Fins." Industrial Robot: An International Journal, vol. 42, no. 3, 2015, pp. 214-221.

[4] "Cownose Ray." Monterey Bay Aquarium, 2017, www.montereybayaquarium.org/animal-guide/fishes/cownose-ray.

[5] OchreOblivion. "Cownose Rays Foraging - New England Aquarium." YouTube, YouTube, 7 Nov. 2011, www.youtube.com/watch?v=wEROmZ8JnSE.

[6] Yang, Shao-bo, Jing, Qiu, Xiao-yun, Han, Kinematics Modeling and Experiments of Pectoral Oscillation Propulsion Robotic Fish, In Journal of Bionic Engineering, Volume 6, Issue 2, 2009, Pages 174-179, ISSN 1672-6529, https://doi.org/10.1016/S1672-6529(08)60114-6.

[7] Bejgerowski W, Ananthanarayanan A, Mueller D, Gupta SK. Integrated Product and Process Design for a Flapping Wing Drive Mechanism. ASME. *J. Mech. Des.* 2009;131(6):061006-061006-9. doi:10.1115/1.3116258.

[8] Polulu, "Pololu Aluminum Scooter Wheel Adapter for 4mm Shaft," 2018, from https://www.pololu.com/product/2672

[9] Polulu, "20.4:1 Metal Gearmotor 25Dx50L mm LP 6V with 48 CPR Encoder," 2018, from https://www.pololu.com/product/2283

[10] Polulu, "Pololu Dual MC33926 Motor Driver Shield for Arduino," 2018, from https://www.pololu.com/product/2503

[11] Creative Robotics, "What Are Quadrature Encoders," from http://www.creative-robotics.com/quadrature-intro