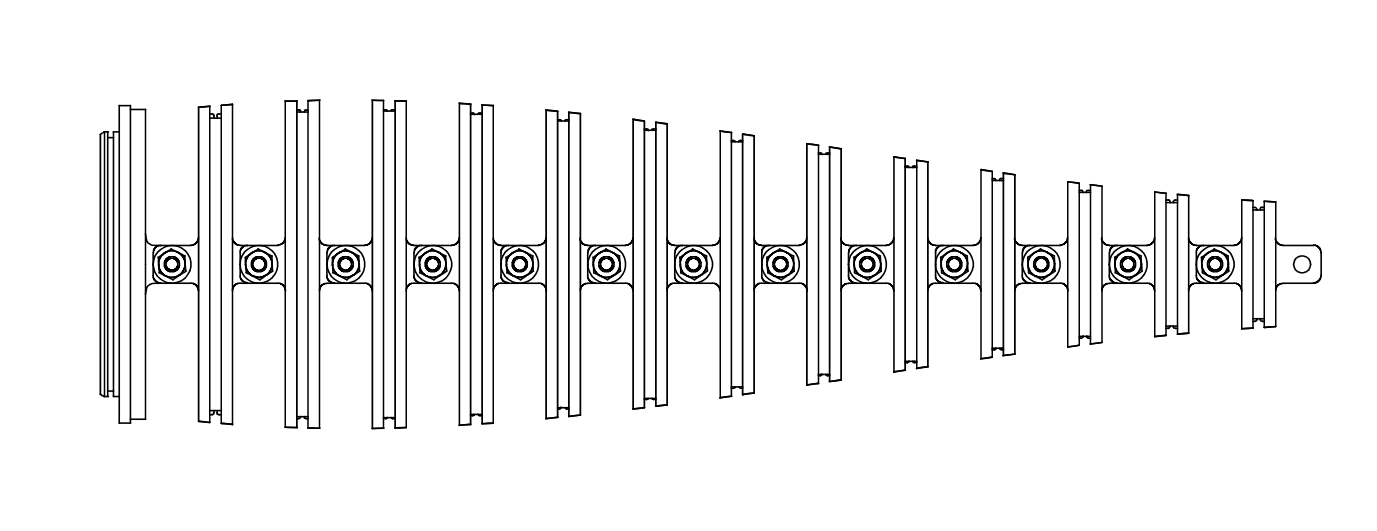
**Proposed mechatronics configuration to achieve *piscine* locomotion (*to be verified by physical testing*)**

Original configuration concept carried forward from PDR:

Each stepper motor is responsible for producing its own numbered sinusoidal piscine harmonic – e.g. for the third harmonic, the third stepper motor controls the cords that cross sides twice up to the end length of the fish vertebrae which in theory produces a sine wave with three peaks.

Cord configuration for the 1st harmonic



Cord configuration for the 2nd harmonic

Cord configuration for the 3rd harmonic

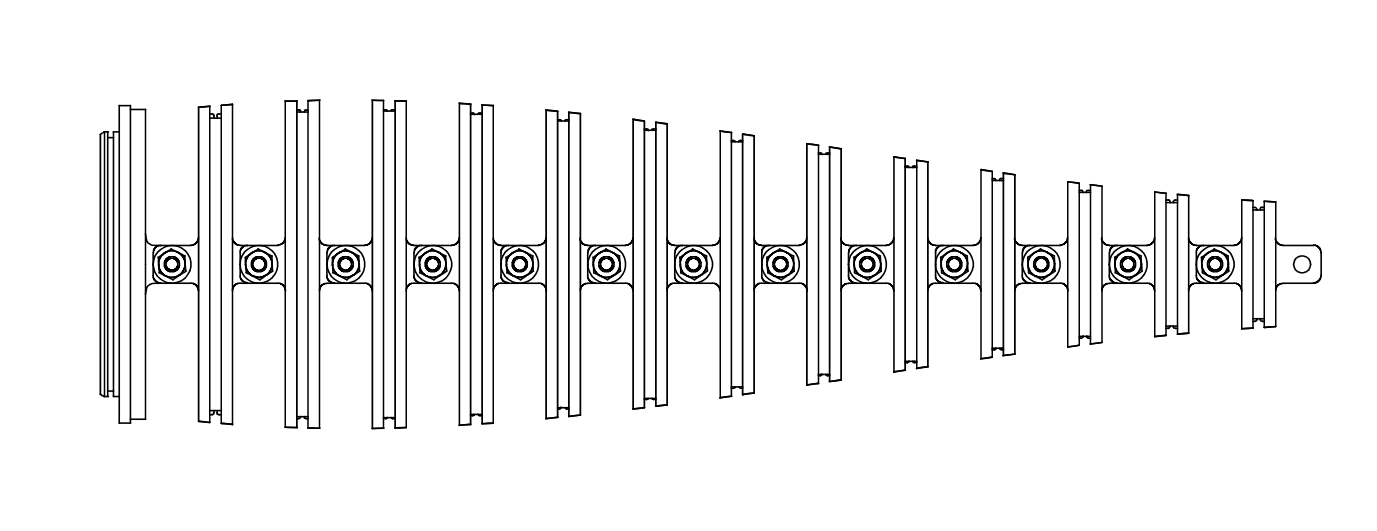
Three Stepper Motors

Preliminary testing:

Due to the nature of the project (i.e. the mathematics of the fish involves solving complicated non-homogenous partial differential equations), physical testing through iterative methods was found to be the best solution. Hence using the second vertebrae model (produced by rapid prototyping) with a standard sized NEMA 17 stepper motor, the initial concept had the unforeseen problems of having too much bending towards the front end with little flexing towards the end. It also had the occasional problem of the cord getting ‘stuck’ as in crosses over the links which had the unintended result of desynchronising the stepper. To avoid these problems, two cord configurations are proposed.

Configuration Concept 2 – Antagonizing Stepper Motors

Rather than just one stepper motor per harmonic, multiple steppers (*n*) will be used depending on the *nth* harmonic (where *n* is one to three). Each cord connection from each motor will only reach up to a certain length, and when trying to do higher harmonics, such as the third, all three will be oscillating in opposite direction to the adjacent motors to produce the harmonic. This produced the piscine motion when the prototype model was tested in air, however, will have to be verified by testing submerged in water.



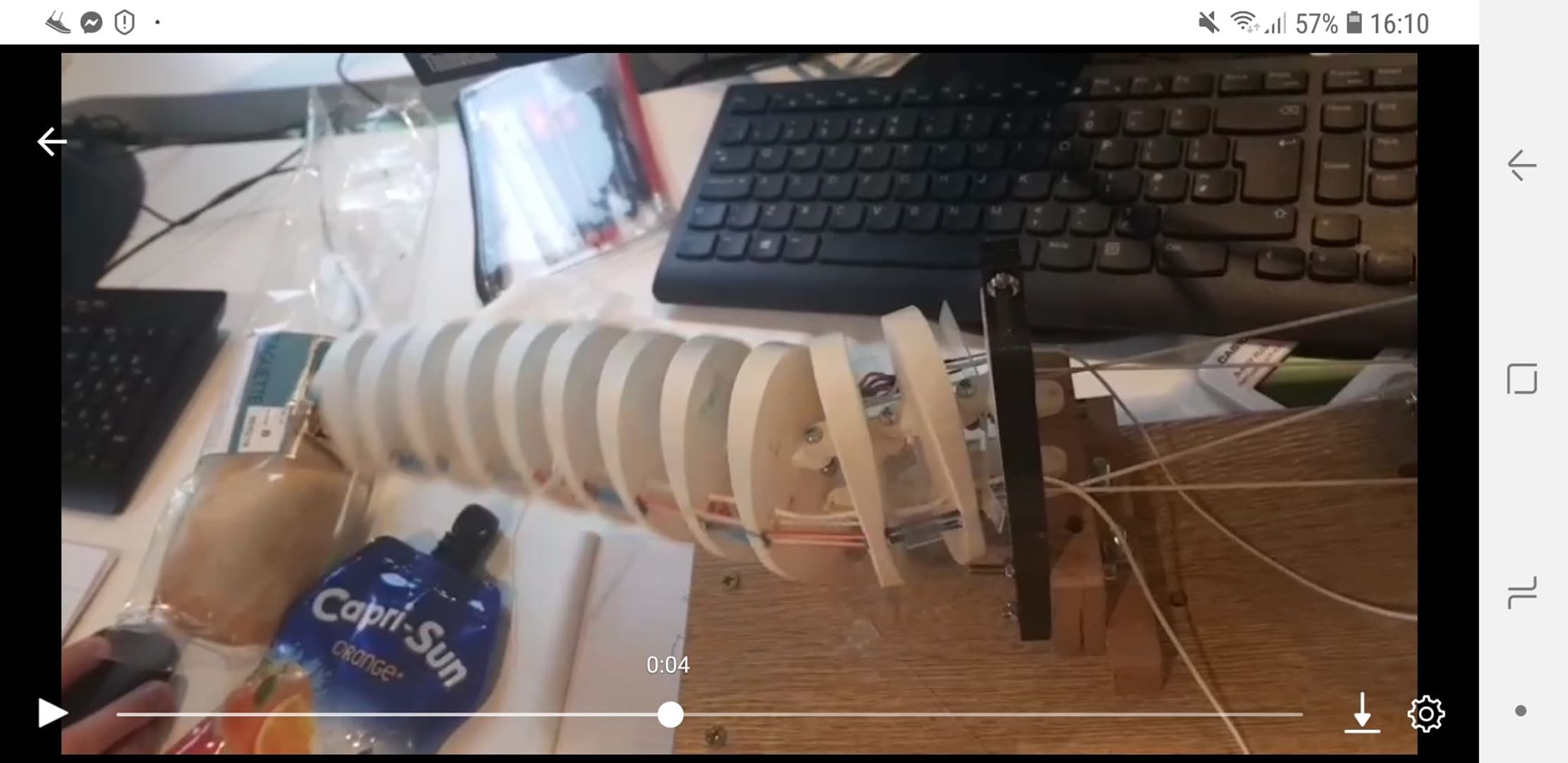
Cord for 3rd Stepper

Cord for 2nd Stepper

Cord for the 1st Stepper

Stepper Motors 1,2 and3

Figure \_\_ - Second Harmonic using two Stepper Motors - Tested in the air only



Configuration Concept 3 – Using the reactive forces of the water to form the harmonics

Same cord configuration as the antagonistic concept, however, rather than having *n* motors to create the *nth* harmonic, the higher harmonics are created by using a single motor and the reactive forces from the water to create the sinusoid shape. Test was done only with the reaction forces present only in the tail thus will also be verified by underwater testing.

Figure \_\_ - Second harmonic produced by using reactive force only and having the cord reach up to a certain length along the local axis of the body.



**Brief Summary the Arduino Code**

The finished code will accommodate for the two proposed concept. Each motor configuration state can be changed via the serial monitor using the keypad. For example, state 1 could be just one motor functioning, state 2 could be two antagonising motors to create the harmonic motion and state 3 could be a single motor creating second harmonic through reactive means etc. The actual values in the code (such as speed, acceleration and position to rotate to) will be refined through iterative testing to obtain the *piscine* locomotion.

There is also a default state which will set the stepper motors back to their defined zero position. This will be used in between state changes to ensure the motors do not become desynchronised with each other during the state change and will be done by entering the value zero through the serial monitor (for example this is state 0).

The code also incorporates the *AccelStepper* library. The step per time algorithm (by David Austin) used in the library allows for smoother acceleration and deceleration profiles (rather than linear acceleration which causes jerks in the motion of the stepper).

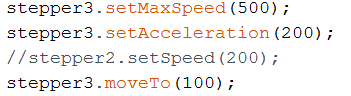
**Brief Explanation of the Arduino Code**

*A more in-depth explanation through comments will be provided in the Arduino source code which is in the DHF.*

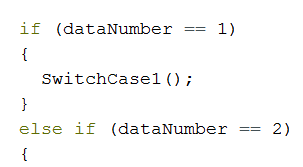
Each stepper motor used will have to be first created as an *AccelStepper* object:



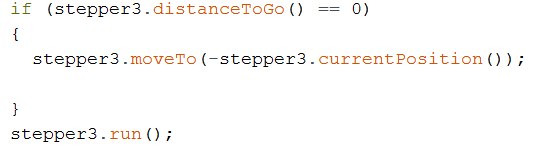
Then values for the allowable max speed, acceleration and the position have to be defined:



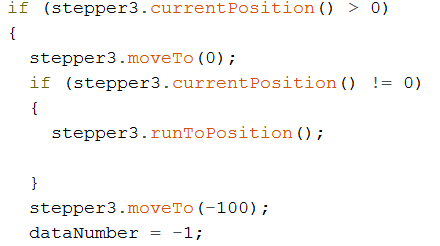
The motor states can be changed through a simple if loop:



The motor can produce the *flapping* or *piscine swimming* effect through bouncing from one position to the other which is achieved by having the value of the position reverse into the negatives when the motor has reached the to go to value:



And a reset state position is then achieved by essentially programming the steppers to return to the initial zero position whenever a value such as zero is entered through the serial monitor:



Functions to take in information via the serial monitor were taken from programs created by Dr. Jones.