**Biomimicry Robotic Snake**

**Seminar Report submitted in the partial fulfillment**

**of**

##### **Term Work**

##### In

##### **Electro-Mechanical Workshop**

By

**MAYANK SHARMA (H045)**

**PRATIK SHIVESHWAR (H049)**

**SAHIL SONAIYA (H050)**

Under the supervision of

**VINOD JAIN**

**(HOD, Mechatronics, MPSTME)**

**SVKM’s NMIMS University**

(Deemed-to-be University)

****

**MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING**

**Vile Parle (W), Mumbai-56**

**2020-2021**

### **CERTIFICATE**

****

This is to certify that the seminar report entitled **“Biomimicry Snake Robot”**, has been done by **Mayank Sharma, Pratik Shiveshwar, Sahil Sonaiya** under my guidance and supervision & has been submitted for term work evaluation for the subject **Electro-Mechanical Workshop** in semester **5** for thedegree of Bachelor of Technology in Mechatronics of MPSTME, SVKM’s NMIMS (Deemed-to-be University), Mumbai, India.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Vinod Jain**

**(HOD, Mechatronics, MPSTME)**

**Date:**

**Place: Mumbai**

**ABSTRACT**

As we see life around us, there is so much we discover just by observing the organisms,

Be it about their unique appearance or their behavior or about their special ability to survive in nature.

In the past, many scientists have taken inspiration from various life forms present on Earth, like spiders, bees, birds, fish, dogs, etc to replicate one or more of its mechanisms.

In the project, a snake bot is developed that mimics the behavior and movement patterns of a snake when an RC signal will be given to it so that the robot moves as required and besides, with few modifications or updates it can also provide motion in water.

The skeleton design is inspired by the vertebrae of snakes and is modified into the form of elliptical plates which are 3-D printed,

At present in our model, the skeleton of the snake consists of these elliptical plates and eight MG90 micro servo motors to give the movement to the snake's vertebrate.

Also after going through most of the research papers related to most slithering animals, and trying with graphical simulations, the mathematical expression for the slithering movement of snakes was achieved. This mathematical expression was later verified by the movement produced by our 3-D CAD model on simulation software.

**List of the figures**

| **Sr.No** | **Fig. No.** | **Name of the figure** | **Page No.** |
| --- | --- | --- | --- |
| 1 | 2.1 | ***Joints Between two vertebrates and within a vertebrate*.** | **3** |
| 2 | 2.2 | ***Skeletal Imitation of a Snake*** | **3** |
| 3 | 4.1 | ***The output of Python-1 Code*** | **5** |
| 4 | 4.2 | ***The output of Python-2 Code.*** | **10** |
| 5 | 4.3 | ***Electronic connections of Servo motors to Arduino*** | **11** |
| 6 | 4.4 | ***The oscilloscope shows the input given to the first 4 servo motors.*** | **12** |
| 7 | 4.5 | ***Virtual terminal showing Angles given to the first 8 servos*** | **12** |
| 8 | 4.6 | ***Shows Von Mises Stress of ABS material*** | **19** |
| 9 | 4.7 | ***Shows equivalent strain of ABS mater*** | **19** |
| 10 | 4.8 | ***Shows displacement of ABS material*** | **20** |
| 11 | 4.9 | ***Shows Von Mises Stress of Nylon 6/10 material*** | **20** |
| 12 | 5.0 | ***Shows equivalent strain of Nylon 6/10 material*** | **21** |
| 13 | 5.1 | ***Shows displacement of Nylon 6/10 material*** | **21** |
| 14 | 5.2 | ***Final snake robot with EPDM rubber skin*** | **22** |

**Contents**

|  | | | | Page No |
| --- | --- | --- | --- | --- |
| **Chapter 1 Introduction** ( of the seminar, and the flow of different chapters) | | | | |
| **1.1** | Background of Snake Robot | | | **1** |
| **1.2** | Inspiration for Snake robot | | | **1** |
| **1.3** | Contribution | | | **1** |
| **1.4** | Organization of the report | | | **2** |
| **Chapter 2 Features** | | | | |
| **2.1** | Aesthetics | | | **3** |
| **2.2** | Motion | | | **3** |
| **2.x** | Control | | | **3** |
| **Chapter 3 Applications** | | | | |
| **3.1** | Scientific-Exploration | | | **4** |
| **3.2** | Military Application | | | **4** |
| **Chapter 4 Methods** | | | |  |
| **4.1** | Mathematical expression for the motion | | | **5** |
| **4.2** | Code to simulate the motion | | | **6** |
| **4.3** | Electronics Simulation | | | **11** |
| **4.4** | Snake Robot Simulation | | | **17** |
| **Chapter 5 Conclusion** | | | | **18** |
| **Chapter 6 Summary** | | | | **19** |
| **Resource Material** | | | | **20** |
|  | | | |  |

**Chapter 1**

**Introduction**

**1.1. Background of the seminar topic**

Our project aims to build a robot that mimics the behavior of a snake in terms of its motion and looks. This project consists of a snake built using servo motors and 3-D printed parts which is lastly covered by an EPDM rubber sheet to replicate its skin and features like grip. This snake carries the potential to imitate movements of its natural form on both land and in water.

**1.2. Motivation and scope of the topic**

Biomimicry is learning from and then emulating natural forms, processes, and ecosystems to create more sustainable designs. Designs that can directly or indirectly serve humanity.

The core idea is that nature has already solved many of the problems we are grappling with: energy, food production, climate control, benign chemistry, transportation, collaboration, and more. This shift from learning about nature to learning from nature requires a new method of inquiry, a new set of lenses, and above all, a new humility.

Biomimicry is like studying a leaf to invent a better solar cell. Replicating the behavior or design or mechanisms of these earth-savvy designs can help humans to further sophisticate their technologies.

Similarly, we have tried to replicate a snake which carries similar behavior in terms of motion and aesthetics. Other than these features we plan to add more features in the future to make it fulfill the requirements as per the tasks assigned.

**1.3. Salient contribution/study (in point form)**

* We have successfully expressed snakes motion in a mathematical expression.
* The body of the snake is 3d printed and stress analysis is done for different materials.
* The movement of the snake is emulated to achieve near to perfect motion.

**1.4. Organization of the report**

The following pages summarize the project's discussion of construction methods, features, and application of snake robots. **Chapter 2** describes the features of snake robots. **Chapter 3** provides the application of snake robots in various situations. **Chapter 4** explains the making of the snake robot or the methods implemented to construct the snake robot. Chapter 5 considers the limitations and concludes the description. **Chapter 5**

**Chapter 2**

**Features**

**2.1 Aesthetics**

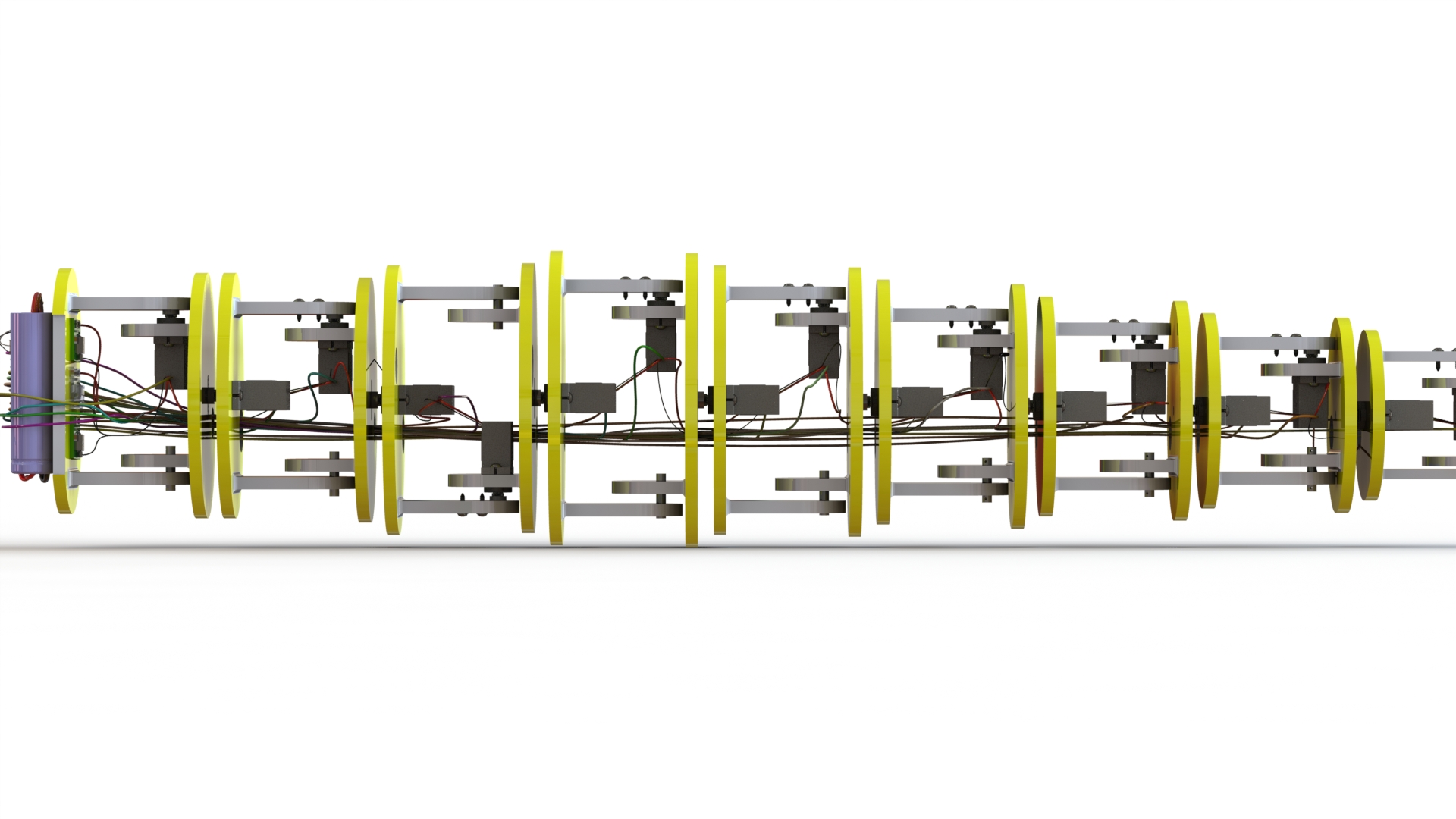
1. Covered with an EPDM rubber sheet to provide protection from foreign materials from entering inside and give an aesthetic look to the robot

**2.2 Motion**

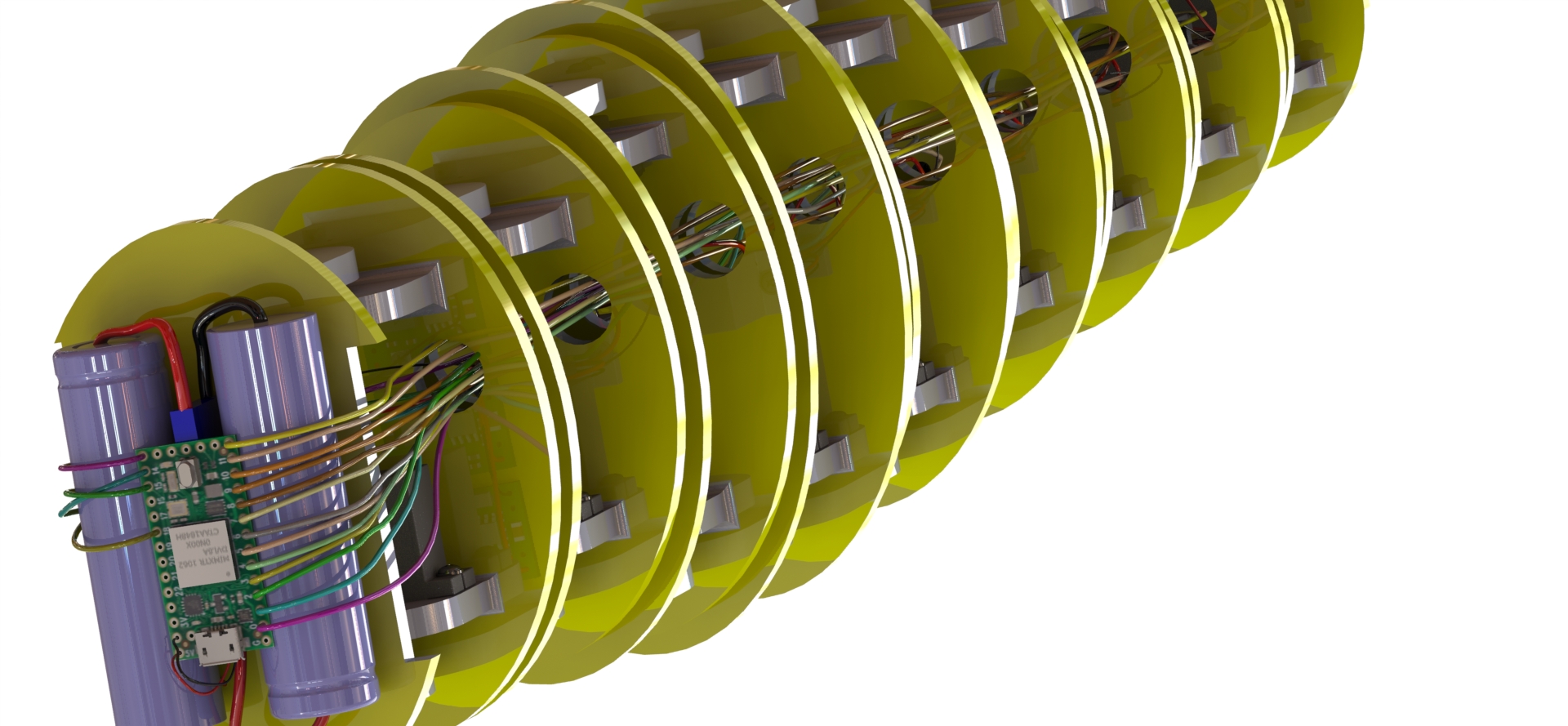
1. Locomotive: Can perform serpentine motion along all the standard 3-axis i.e multiple degrees of freedom.
2. Ability to imitate the motion of a snake

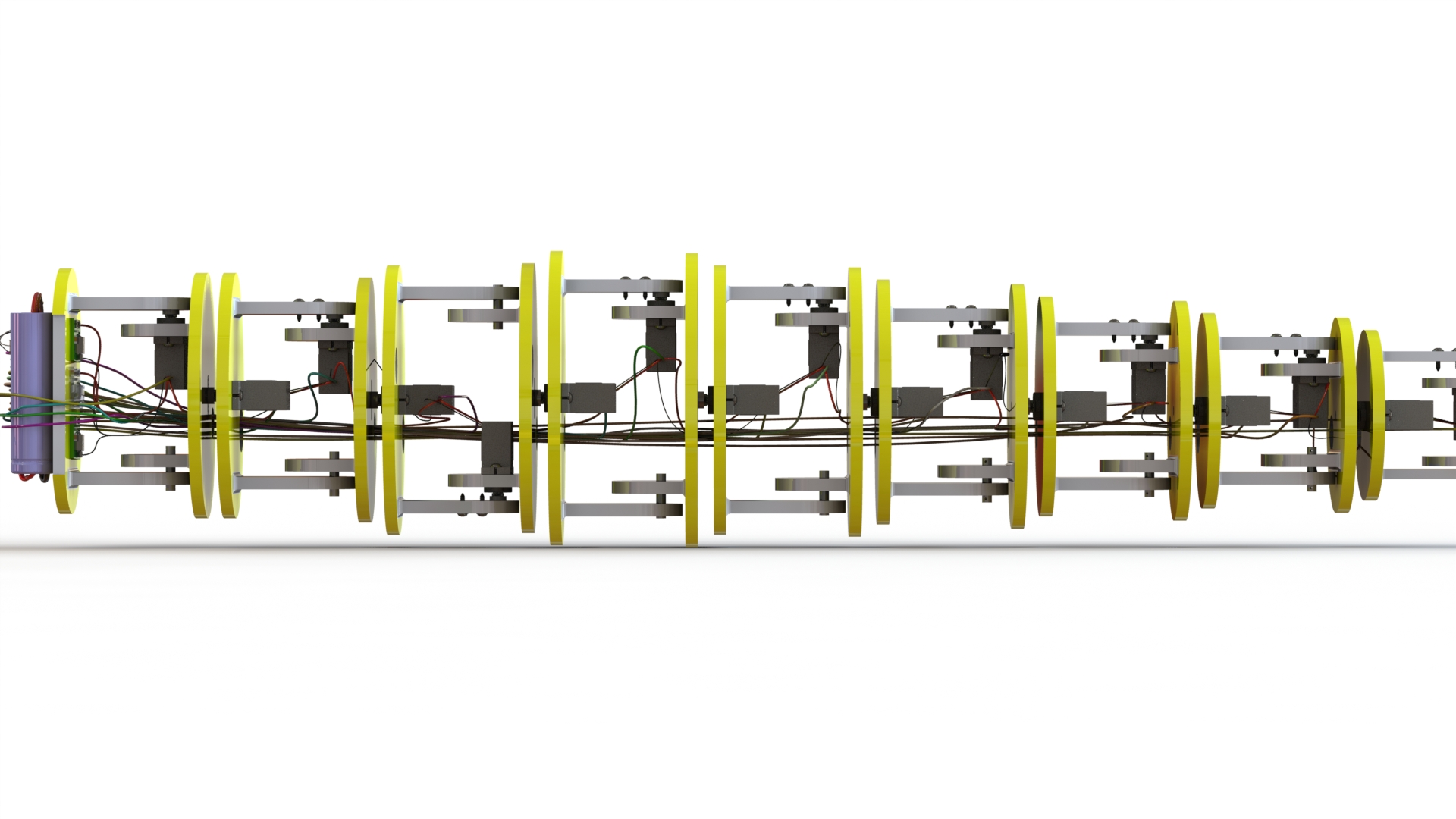
**2.3** **Structure**

1. A novel structural design which aides the motion of the robotic snake
2. This structure consists of many mechanical links just like vertebrates in the snake and these links clubbed together to become a robotic snake.
3. The entire length of the snake is about 500mm, having 16servos

****

**Fig 2.1: Joints Between two vertebrates and within a vertebrate.**





**Fig2.2: Skeletal Imitation of a Snake**

**2.4 Control**

1. It is controlled by a 2.4GHz communication frequency.

**Chapter 3**

**Applications**

**3.1 Scientific-Exploration**

1. For research purposes, where it can enter grasslands or into the environment where usually snakes reside so that we can monitor a snake or group of snakes more closely for any scientific purpose.
2. If modified to locomote in water, it can swim to explore the underwater.

**3.2 Military Application**

1. As it is completely covered in EPDM rubber, it can move through the debris of collapsed buildings for search and rescue work.
2. Can be used in military applications, where an underwater robot with thrusters can be detected on sonar.

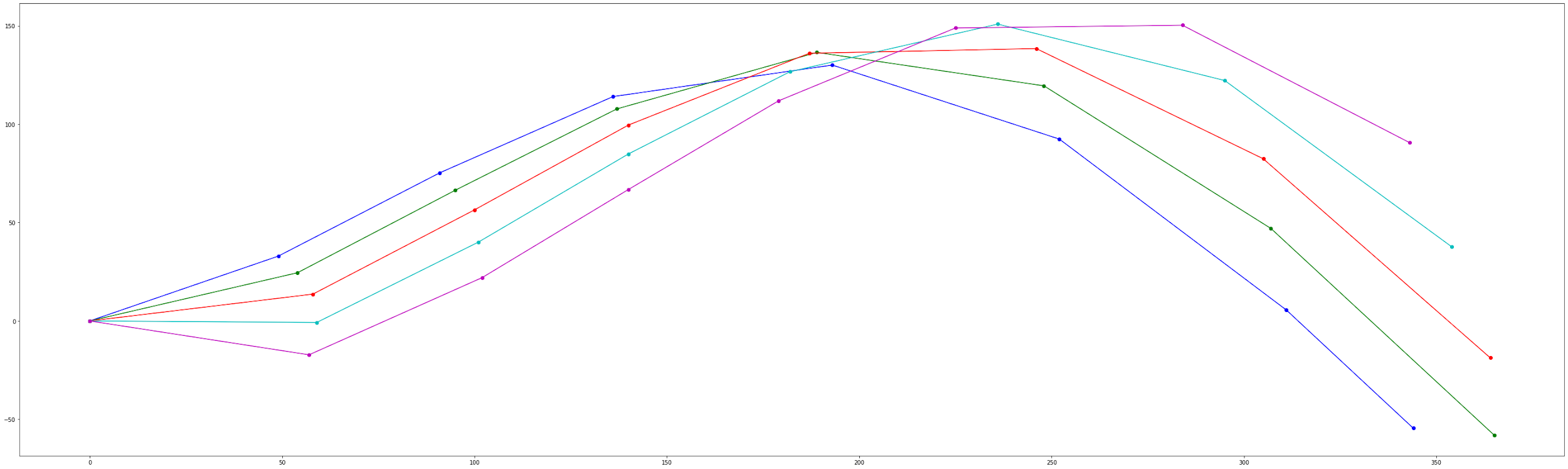
**Chapter 4**

**Methods**

The entire bot is controlled by a 24GHz transmitter. The receiver is

**4.1] Motion Expression:**

Expressing the motion of snakes through mathematics.

***Fig 4.1 Output in Python environment***

**Fig 4.1** displays the movement of the snake with respect to time. Sequence:dark blue,green,red, light blue, purple.Servo angles are calculated in real-time according to the current position of the snake. These servo angles are then given to the servo motor by the microcontroller.

**4.2] Code to Simulate the motion [Python] :**

**Python Code:1 -- Gives an animation**

import numpy as np

import matplotlib.pyplot as plt

import matplotlib

matplotlib.use("Agg")

import math

import matplotlib

matplotlib.rcParams['animation.embed\_limit'] = 2\*\*128

from matplotlib import animation, rc

from IPython.display import HTML

# Set up formatting for the movie files

Writer = animation.writers['ffmpeg']

writer = Writer(fps=100, metadata=dict(artist='Me'), bitrate=1800)

# First set up the figure, the axis, and the plot element we want to animate

fig\_dims = (25, 12)

fig, ax = plt.subplots(figsize=fig\_dims)

plt.close()

ax.set\_xlim(( 0, 700))

ax.set\_ylim((-160, 160))

line, = ax.plot([], [], lw=2,marker='o')

line1, = ax.plot([], [], lw=1)

# initialization function: plot the background of each frame

def init():

line.set\_data([], [])

return (line,)

def wave\_position(t,x,dir\_x):

f = 0.013

theta\_Max = 10

y = theta\_Max\*np.sin(f\*(x- t))\*np.sqrt(x) + dir\_x\*np.sqrt(x)

return y

def circle\_y\_value(a,b,r,x\_value,t,dir\_x): #a-> x\_curr, b-> y\_curr

if wave\_position(t,x\_value,dir\_x)>b:

y\_value = np.sqrt((r\*\*2)-((a-x\_value)\*\*2)) + b

elif wave\_position(t,x\_value,dir\_x)<b:

y\_value = -np.sqrt((r\*\*2)-((a-x\_value)\*\*2)) + b

elif wave\_position(t,x\_value,dir\_x)==b:

if wave\_position(t,x\_value,dir\_x)>=0:

y\_value = np.sqrt((r\*\*2)-((a-x\_value)\*\*2)) + b

else:

y\_value = -np.sqrt((r\*\*2)-((a-x\_value)\*\*2)) + b

return y\_value

def intersection\_x\_value(x,t,l,dir\_x):

i = 0

delta = 0.5

for i in np.arange(x+l,x,-delta):

wave1 = wave\_position(t,i-delta,dir\_x)

wave2 = wave\_position(t,i,dir\_x)

circle1 = circle\_y\_value(x,wave\_position(t,x,dir\_x),l,i-delta,t,dir\_x)

circle2 = circle\_y\_value(x,wave\_position(t,x,dir\_x),l,i,t,dir\_x)

d1 = wave1 - circle1

d2 = wave2 - circle2

if d1==0 or d2==0 or (d1\*d2<=0 and (d1<0 or d2<0)):

return i

Break

return x+l

def animate(t):

dir\_x = 0

x\_data = []

y\_data = []

x\_value = 0

for i in range(8):

x\_data.append(x\_value)

y\_data.append(wave\_position(t,x\_value,dir\_x))

x\_value = intersection\_x\_value(x\_value,t,60,dir\_x)

x\_axis = np.linspace(0,500,500)

line.set\_data(x\_data, y\_data)

return (line,line1,)

anim = animation.FuncAnimation(fig, animate, init\_func=init,frames=1000, interval=1, blit=True)

rc('animation', html='jshtml')

anim

# Uncomment the next line to save the output as video

#anim.save('im.mp4', writer=writer)

[***The output of Python-1 Code.***](https://www.youtube.com/watch?v=rU_yN3grEaA)

**Python Code:2 -- Outputs how each servo angle changes**

import numpy as np

import matplotlib.pyplot as plt

import math

def wave\_position(t,x,dir\_x):

f = 0.013

theta\_Max = 10

y = theta\_Max\*np.sin(f\*(x- t))\*np.sqrt(x) + dir\_x\*np.sqrt(x)

return y

def circle\_y\_value(a,b,r,x\_value,t,dir\_x): #a-> x\_curr, b-> y\_curr

y\_val = 0

if wave\_position(t,x\_value,dir\_x)>b:

y\_val = np.sqrt((r\*\*2)-((a-x\_value)\*\*2)) + b

elif wave\_position(t,x\_value,dir\_x)<b:

y\_val = -np.sqrt((r\*\*2)-((a-x\_value)\*\*2)) + b

elif wave\_position(t,x\_value,dir\_x)==b:

if wave\_position(t,x\_value,dir\_x)>=0:

y\_val = np.sqrt((r\*\*2)-((a-x\_value)\*\*2)) + b

else:

y\_val = -np.sqrt((r\*\*2)-((a-x\_value)\*\*2)) + b

return y\_val

def intersection\_x\_value(x,t,l,dir\_x):

i = 0

delta = 0.5

for i in np.arange(x+l,x,-delta):

wave1 = wave\_position(t,i-delta,dir\_x)

wave2 = wave\_position(t,i,dir\_x)

circle1 = circle\_y\_value(x,wave\_position(t,x,dir\_x),l,i-delta,t,dir\_x)

circle2 = circle\_y\_value(x,wave\_position(t,x,dir\_x),l,i,t,dir\_x)

d1 = wave1 - circle1

d2 = wave2 - circle2

if d1==0 or d2==0 or (d1\*d2<=0 and (d1<0 or d2<0)):

return i

break

return x+l

def angle\_lines(m1,m2):

angle = math.atan((m1-m2)/(1+(m1\*m2)))

return angle

def angle(x1,x2,x3,y1,y2,y3):

m1 = (y1-y2)/(x1-x2)

m2 = (y2-y3)/(x2-x3)

angle = angle\_lines(m1,m2)

return angle

#init()

x\_data = []

y\_data = []

x\_value = 0

servo0 = []

servo1 = []

servo2 = []

servo3 = []

servo4 = []

servo5 = []

servo6 = []

servo7 = []

time = np.linspace(0,1000,1000)

for t in time:

x\_data = []

y\_data = []

x\_value = 0

dir\_x = 0

for i in range(9):

x\_data.append(x\_value)

y\_data.append(wave\_position(t,x\_value,dir\_x))

x\_value = intersection\_x\_value(x\_value,t,60,dir\_x)

if x\_data[1] == 0:

servo0.append(0)

else:

servo0.append(int(angle\_lines(0,y\_data[1]/x\_data[1])\*180\*7/22))

if math.isnan(angle(x\_data[0],x\_data[1],x\_data[2],y\_data[0],y\_data[1],y\_data[2])):

servo1.append(0)

else:

servo1.append(int(angle(x\_data[0],x\_data[1],x\_data[2],y\_data[0],y\_data[1],y\_data[2])\*180\*7/22))

if math.isnan(angle(x\_data[1],x\_data[2],x\_data[3],y\_data[1],y\_data[2],y\_data[3])):

servo2.append(0)

else:

servo2.append(int(angle(x\_data[1],x\_data[2],x\_data[3],y\_data[1],y\_data[2],y\_data[3])\*180\*7/22))

if math.isnan(angle(x\_data[2],x\_data[3],x\_data[4],y\_data[2],y\_data[3],y\_data[4])):

servo3.append(0)

else:

servo3.append(int(angle(x\_data[2],x\_data[3],x\_data[4],y\_data[2],y\_data[3],y\_data[4])\*180\*7/22))

if math.isnan(angle(x\_data[3],x\_data[4],x\_data[5],y\_data[3],y\_data[4],y\_data[5])):

servo4.append(0)

else:

servo4.append(int(angle(x\_data[3],x\_data[4],x\_data[5],y\_data[3],y\_data[4],y\_data[5])\*180\*7/22))

if math.isnan(angle(x\_data[4],x\_data[5],x\_data[6],y\_data[4],y\_data[5],y\_data[6])):

servo5.append(0)

else:

servo5.append(int(angle(x\_data[4],x\_data[5],x\_data[6],y\_data[4],y\_data[5],y\_data[6])\*180\*7/22))

if math.isnan(angle(x\_data[5],x\_data[6],x\_data[7],y\_data[5],y\_data[6],y\_data[7])):

servo6.append(0)

else:

servo6.append(int(angle(x\_data[5],x\_data[6],x\_data[7],y\_data[5],y\_data[6],y\_data[7])\*180\*7/22))

if math.isnan(angle(x\_data[6],x\_data[7],x\_data[8],y\_data[6],y\_data[7],y\_data[8])):

servo7.append(0)

else:

servo7.append(int(angle(x\_data[6],x\_data[7],x\_data[8],y\_data[6],y\_data[7],y\_data[8])\*180\*7/22))

fig, ax = plt.subplots(figsize=(20,5))

plt.xlabel("Time (Seconds)")

plt.ylabel("Servo angles (Degrees)")

ax.plot(servo0,'b',linewidth=0.8,label='Servo 1')

ax.plot(servo1,'g',linewidth=0.8,label='Servo 2')

ax.plot(servo2,'r',linewidth=0.8,label='Servo 3')

ax.plot(servo3,'c',linewidth=0.8,label='Servo 4')

ax.plot(servo4,'m',linewidth=0.8,label='Servo 5')

ax.plot(servo5,'k',linewidth=0.8,label='Servo 6')

#plt.show()

legend = ax.legend(loc='upper right', shadow=True, fontsize='large')

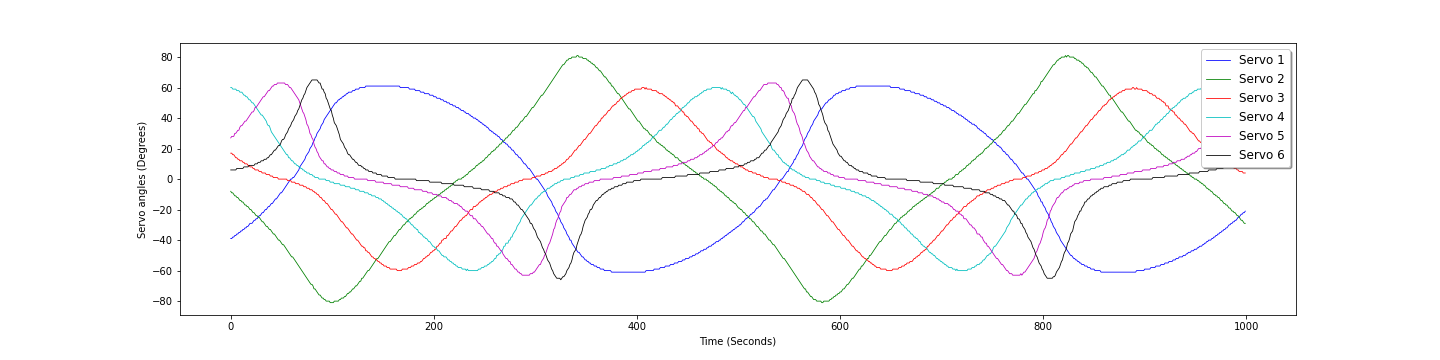
# Put a nicer background color on the legend.

legend.get\_frame().set\_facecolor('white')

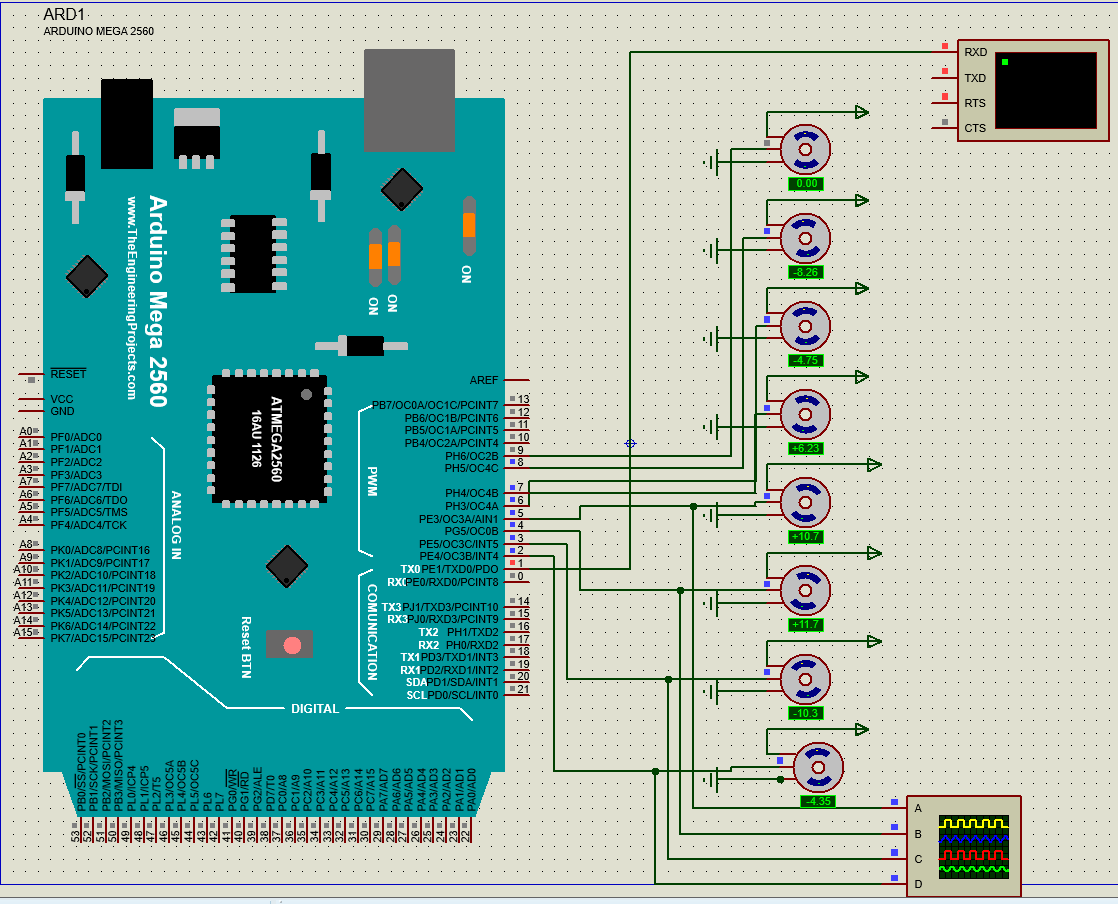
plt.show()

#Uncomment the next line to save the graph

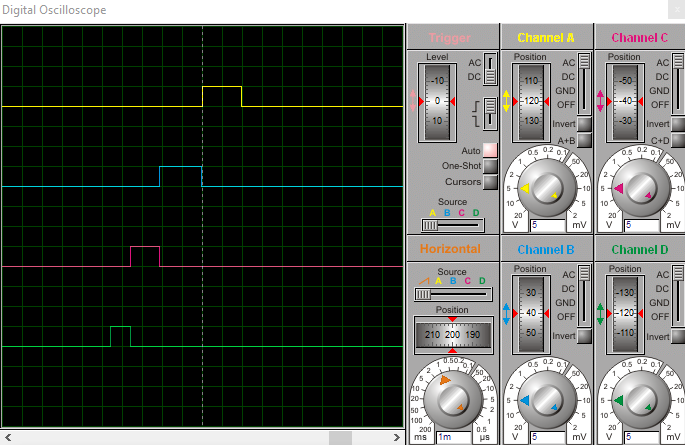
#fig.savefig('foo.png')

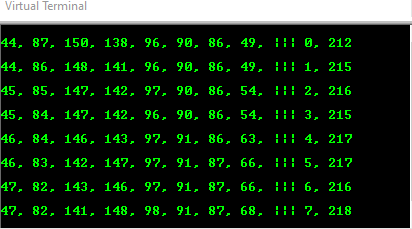
***Fig 4.2 The output of Python-2 Code.***

**4.3] Proteus Simulations**

****

***Fig 4.3: Electronic connections of Servo motors to Arduino***

***Fig 4.4: Oscilloscope shows the input given to the first 4 servo motors.***



***Fig 4.5: Virtual terminal showing Angles given to first 8 servos***

***Format: A1, A2, A3, A4, A5, A6, A7, A8, ||| Time-frame, Time-Delay***

***A- Angle in degrees***

**Teensy Code (Arduino IDE):**

**#include <math.h>**

**#define pi 22/7**

**#define servo\_numbers 8**

**double deg2rad = 22.000/(180.000\*7.000);**

**int t = 0;**

**#include <Servo.h>**

**Servo servo1, servo2, servo3, servo4, servo5, servo6, servo7, servo8, servo9, servo10, servo11, servo12, servo13, servo14, servo15; // create servo object to control a servo**

**double wave\_position(double t, double x)**

**{**

**double f = 1;**

**double theta\_Max = 10;**

**double y = theta\_Max\*sin(deg2rad\*(x-t)\*f)\*sqrt(x);**

**return y;**

**}**

**double circle\_y\_value(double a,double b,double r,double x\_value,double t)//: #a-> x\_curr, b-> y\_curr**

**{**

**double y\_value;**

**double wave\_pos = wave\_position(t,x\_value);**

**//Serial.println(a);**

**//Serial.print(", ");**

**//Serial.println(a);**

**if (wave\_pos>b)**

**{**

**y\_value = sqrt(sq(r)-(sq(a-x\_value))) + b;**

**}**

**else if (wave\_pos<b)**

**{**

**y\_value = -sqrt(sq(r)-(sq(a-x\_value))) + b;**

**}**

**else if (wave\_pos==b)**

**{**

**if (wave\_pos>=0)**

**{**

**y\_value = sqrt(sq(r)-(sq(a-x\_value))) + b;**

**}**

**else**

**{**

**y\_value = -sqrt(sq(r)-(sq(a-x\_value))) + b;**

**}**

**}**

**return y\_value;**

**}**

**double intersection\_x\_value(double x,double t,double l)**

**{**

**double i;**

**//Serial.println(x);**

**double flag = 0;**

**for (i=x+l-1; i>=x; i=i-1)**

**{**

**double wave1 = wave\_position(t,i);**

**double wave2 = wave\_position(t,i+1);**

**double circle1 = circle\_y\_value(x,wave\_position(t,x),l,i,t);**

**double circle2 = circle\_y\_value(x,wave\_position(t,x),l,i+1,t);**

**double d1 = wave1 - circle1;**

**double d2 = wave2 - circle2;**

**//Serial.println(x);**

**//Serial.print(", ");**

**//Serial.println(x,6);**

**if (d1==0 || d2==0 || (d1\*d2<0 && (d1<0 || d2<0)))**

**{**

**//Serial.println("\*");**

**flag = i;**

**break;**

**}**

**}**

**return flag;**

**}**

**double angle\_lines(double m1,double m2)**

**{**

**double servo\_angle = atan((m1-m2)/(1+(m1\*m2)));**

**return servo\_angle;**

**}**

**double angle(double x1,double x2,double x3,double y1,double y2,double y3)**

**{**

**double m1 = (y1-y2)/(x1-x2);**

**double m2 = (y2-y3)/(x2-x3);**

**double servo\_angle = angle\_lines(m1,m2);**

**return servo\_angle;**

**}**

**void setup() {**

**// put your setup code here, to run once:**

**servo1.attach(2); // attaches the servo on respective pins to the servo object**

**servo2.attach(3);**

**servo3.attach(4);**

**servo4.attach(5);**

**servo5.attach(6);**

**servo6.attach(7);**

**servo7.attach(8);**

**// servo8.attach(9);**

**// servo9.attach(10);**

**// servo10.attach(11);**

**// servo11.attach(12);**

**// servo12.attach(13);**

**// servo13.attach(44);**

**// servo14.attach(45);**

**// servo15.attach(46);**

**Serial.begin(115200);**

**t = 0;**

**}**

**void loop() {**

**unsigned long start = millis();**

**int angles[servo\_numbers];**

**// put your main code here, to run repeatedly:**

**double x\_data[servo\_numbers+1];**

**double y\_data[servo\_numbers+1];**

**double x\_value = 1;**

**for(int i = 0; i<servo\_numbers+1; i++)**

**{**

**x\_data[i] = x\_value;**

**y\_data[i] = wave\_position(t,x\_value);**

**//Serial.println(t);**

**x\_value = intersection\_x\_value(x\_value,t,60);**

**//Serial.println(x\_value);**

**}**

**/\*for(int i=0;i<servo\_numbers+1;i++)**

**{**

**Serial.print(x\_data[i]);**

**Serial.print(", ");**

**Serial.print(y\_data[i]);**

**}\*/**

**Serial.println();**

**angles[0] = (angle\_lines(0,y\_data[1]/x\_data[1])\*180\*7/22)+90;**

**for(int i = 1;i<servo\_numbers;i++)**

**{**

**angles[i] = (angle(x\_data[i-1],x\_data[i],x\_data[i+1],y\_data[i-1],y\_data[i],y\_data[i+1])\*180\*7/22)+90;**

**}**

**for(int i=0;i<servo\_numbers;i++)**

**{**

**Serial.print(angles[i]);**

**Serial.print(", ");**

**}**

**Serial.print("||| ");**

**Serial.print(t);**

**unsigned long ends = millis();**

**Serial.print(", ");**

**Serial.println(ends-start);**

**t = t+1;**

**int input1=angles[0];**

**int input2=angles[1];**

**int input3=angles[2];**

**int input4=angles[3];**

**int input5=angles[4];**

**int input6=angles[5];**

**int input7=angles[6];**

**servo1.write(input1); // tell servo to go to position in variable 'angles'**

**servo2.write(input2);**

**servo3.write(input3);**

**servo4.write(input4);**

**servo5.write(input5);**

**servo6.write(input6);**

**servo7.write(input7);**

**//delay(1000);**

**}**

***“Output for this code can be seen in final robot simulation below”***

**4.4] Snake Robot Simulation**

**1) Motion analysis:-**

We used SOLIDWORKS software for performing motion analysis of the snake robot. For performing motion analysis we used the servo angle values extracted from the code written in python for a serpentine curve.

Link to watch the Motion analysis:-<https://youtu.be/fDz0RqOSchQ>

**2) Static structural analysis:-**

Structural analysis is necessary to make sure that the material used will be able to bear the forces and its effects on the mechanical link. We chose two materials ABS, Nylon 6/10 for comparison.

We have chosen one link from the whole robot for performing static-structural analysis. For a fair comparison between the two materials, we have kept some parameters constant:

1. Same mesh size:

Element size- 3.5mm

Tolerance- 0.175mm

1. Load:

Applied torque of 0.19 Nm

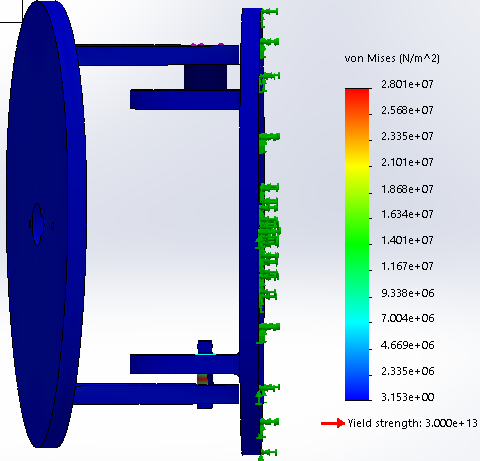
The results for both materials have been presented in the table below

| **S.No.** | **Material** | **Max Von Mises Stress (N/m^2)** | **Max equivalent Strain** | **Max Resultant Displacement (mm)** |
| --- | --- | --- | --- | --- |
| 1 | ABS | 2.801e+07 | 7.825e-03 | 1.897e+00 |
| 2 | Nylon 6/10 | 2.812e+07 | 2.091e-03 | 5.121e-01 |

**Snapshots from the structural analysis:**

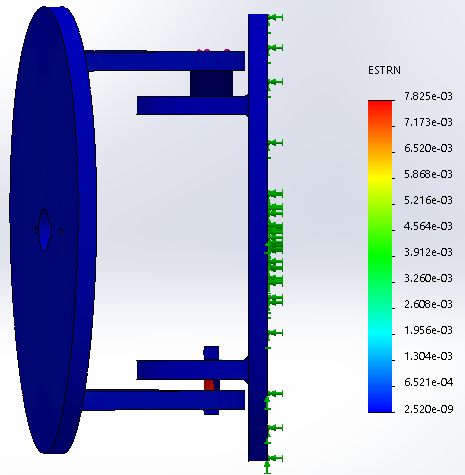
1. **ABS:**

Von Mises Stress:



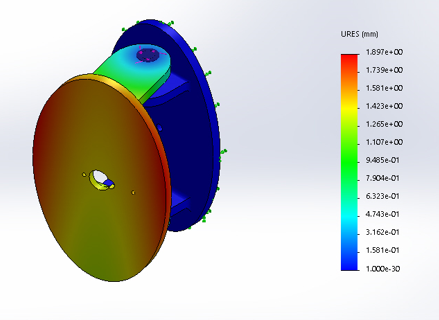
***Fig 4.6: Shows Von Mises Stress of ABS material***

Equivalent Strain:

****

***Fig 4.7: Shows equivalent strain of ABS material***

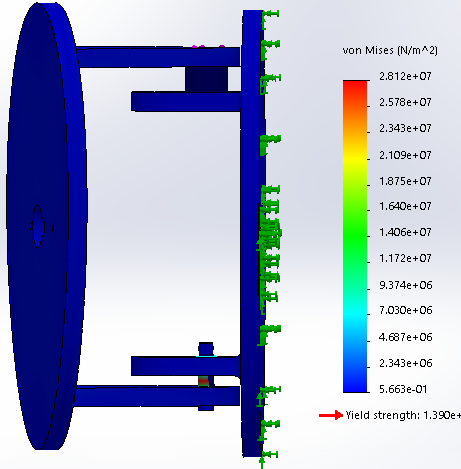
Displacement:

****

***Fig 4.8: Shows displacement of ABS material***

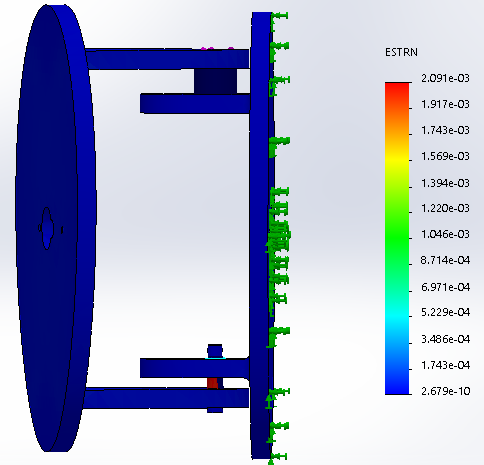
1. **Nylon 6/10:**

Von Mises Stress:

****

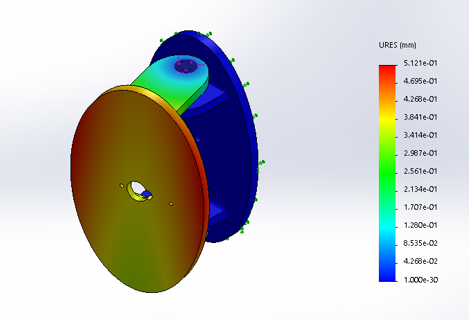
***Fig 4.9: Shows Von Mises Stress of Nylon 6/10 material***

Equivalent Strain:



***Fig 5.0: Shows equivalent strain of Nylon 6/10 material***

Displacement:

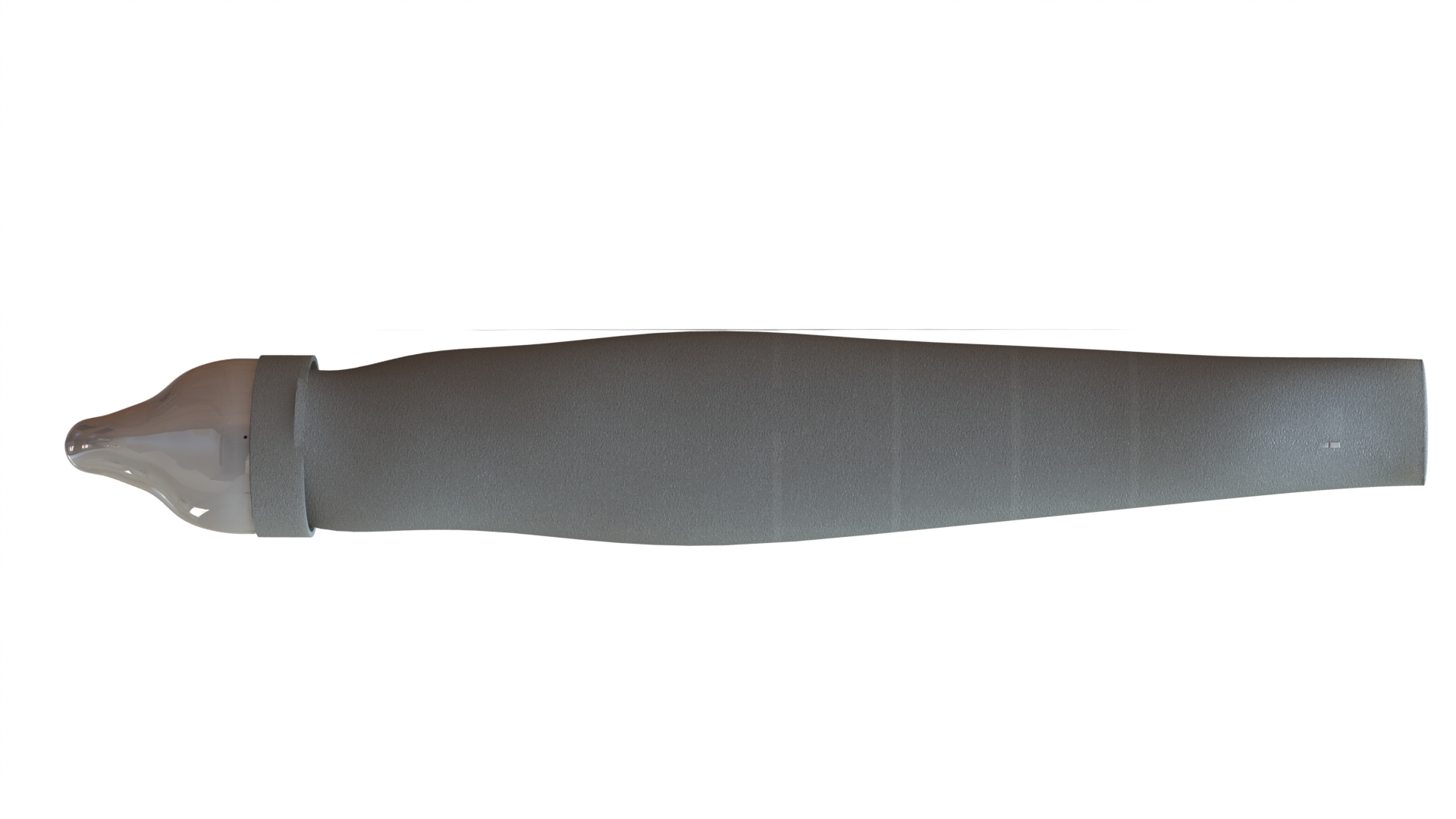


***Fig 5.1: Shows displacement of Nylon 6/10 material***

**Chapter 5**

**Conclusion**

By the use of the mathematical function we derived, we were able to determine the motion of the snake, and how each servo motor moves with respect to each other. The snake is powered by 2 Li-On batteries connected in series. They are also connected to a BMS(Battery Management System) to take care of over-discharge, over-current, and it also charges the battery without the need of a special charger. This is then stepped down to 6.0V for the servo motors. This ensures that the servo motors receive maximum power, and they work with maximum speed and torque. The entire robot is controlled by a Teensy 4.0 microcontroller, which receives signals from a 2.4GHz frequency receiver.



***Fig 5.2: Final snake robot with EPDM rubber skin***

**Chapter 6**

**Summary**

In this chapter, we are going to summarize the entire project. This project aimed to develop a biomimicry robotics snake which can be used in search-rescue operations on land as well as in water.

To keep the project low cost, the frame of the robot was 3d printed, and light servo motors are used. The skin is made of EPDM rubber, which is flexible and can resist harsh conditions(like water, sun, dust).

This snake is controlled by a transmitter communicating at 2.4GHz frequency.

#### 

#### 

#### 

#### 

#### **REFERENCES**

# [1] J. Gao, X. Gao, W. Zhu, J. Zhu, and B. Wei, **"Design and research of a new structure rescue snake robot with all body drive system,"** *2008 IEEE International Conference on Mechatronics and Automation*, Takamatsu, 2008, pp. 119-124, doi: 10.1109/ICMA.2008.4798737.

# [2] E. Kelasidi, K. Y. Pettersen and J. T. Gravdahl, **"Energy efficiency of underwater snake robot locomotion,"** *2015 23rd Mediterranean Conference on Control and Automation (MED)*, Torremolinos, 2015, pp. 1124-1131, doi: 10.1109/MED.2015.7158907.

# 

# [3] P. Chavan, M. Murugan, E. V. V. Unnikkannan, A. Singh and P. Phadatare, **"Modular Snake Robot with Mapping and Navigation: Urban Search and Rescue (USAR) Robot,"** *2015 International Conference on Computing Communication Control and Automation*, Pune, 2015, pp. 537-541, doi: 10.1109/ICCUBEA.2015.110.

# 

# [4] E. Kelasidi, A. M. Kohl, K. Y. Pettersen, and J. T. Gravdahl, **"Waypoint guidance control for underwater snake robots exposed to ocean currents,"** *2016 24th Mediterranean Conference on Control and Automation (MED)*, Athens, 2016, pp. 518-525, doi: 10.1109/MED.2016.7535893.

[5] E. Kelasidi, K. Y. Pettersen, P. Liljebäck, and J. T. Gravdahl, **"Locomotion efficiency of underwater snake robots with thrusters,"** *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Lausanne, 2016, pp. 174-181, doi: 10.1109/SSRR.2016.7784295.

[6] E. Kelasidi, P. Liljeback, K. Y. Pettersen and J. T. Gravdahl, **"Innovation in Underwater Robots: Biologically Inspired Swimming Snake Robots,"** in *IEEE Robotics & Automation Magazine*, vol. 23, no. 1, pp. 44-62, March 2016, doi: 10.1109/MRA.2015.2506121.

[7] S. Manzoor and Y. Choi, **"Modular design of snake robot for various motions implementation,"** *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Xi'an, 2016, pp. 211-213, doi: 10.1109/URAI.2016.7625739.

[8] Z. AnFan, M. ShuGen, L. Bin, W. MingHui, and W. YueChao, **"Modeling and Simulation of an Underwater Planar Eel Robot in Non-inertial Frame,"** *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, Honolulu, HI, 2017, pp. 840-845, doi: 10.1109/CYBER.2017.8446603.

[9] Y. A. Baysal and I. H. Altas, **"Modelling and Simulation of a Wheel-Less Snake Robot,"** *2020 7th International Conference on Electrical and Electronics Engineering (ICEEE)*, Antalya, Turkey, 2020, pp. 285-289, doi: 10.1109/ICEEE49618.2020.9102599.