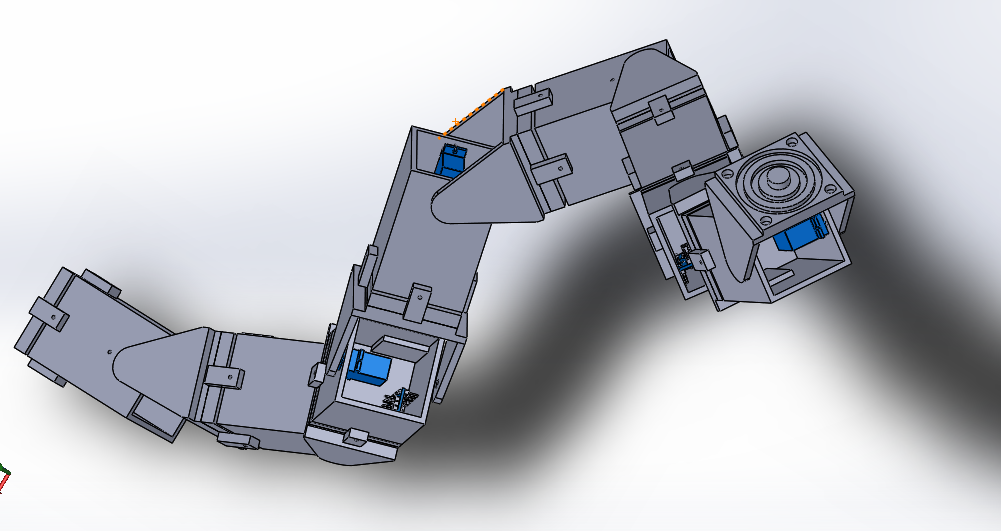
**SURVEILLANCE AND DATA ACQUISITION MODULAR SNAKE ROBOT**

PROJECT MEDUSA

(DRDO: DARE TO DREAM 2019)



**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR CAMPUS**

**Guided by**

Aman Malhotra

Amrit Das

**SRM TEAM HUMANOID**

**Topical Breakdown**

* Introduction
* Overview of Complete Snake Bot Design
* **Structural Design**

> Casing

> Swivel Clamp

> Modular Contact Surfaces

* **Electronic and Communication Design**

> The Main Components within Unit Module

> Factors of Communication between Modules

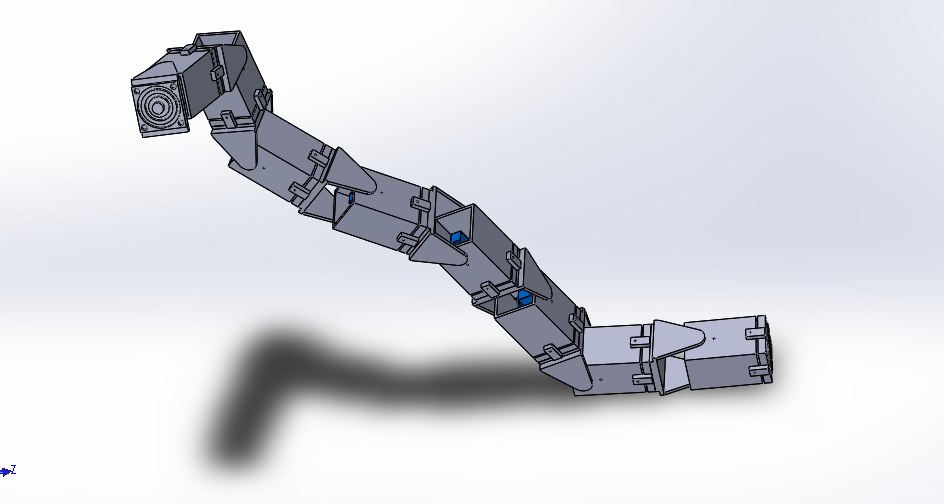
* **Algorithm and Coding**

> Communication code between Master Arduino and Slave Arduino

> Communication code between Master Arduino and Radio Controller

> Seven achievable Gaits

* Challenges
* Limitations
* Additional Components
* Conclusion
* Authors

****

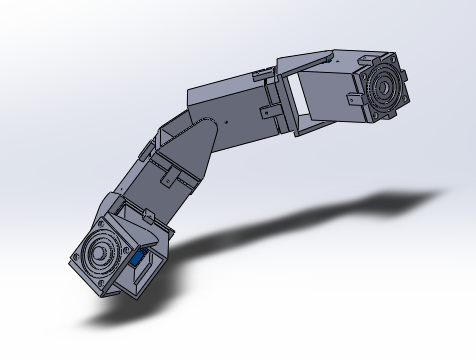
**Introduction**

Snake Bots are robots that closely resemble the structural features of biological snakes. Due to their unique method of propulsion and movement, Snake Bots can be used in a wide range of fields in urban regions such as buildings to climb pipelines, hostage situations and other sensitive tasks that cannot be risked with the involvement of an individual.

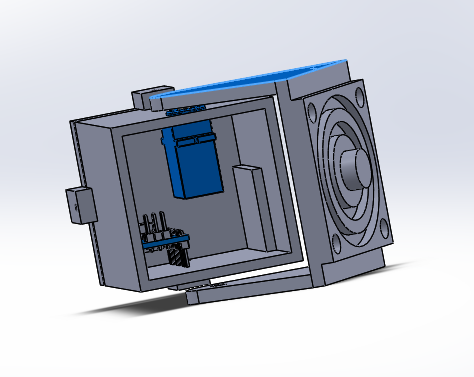
They can especially be applied in the field of Military applications and assist Anti-Terrorist Organisations and Indian Intelligence Organisations for many tasks that are currently not possible using conventional service bots.

For such purposes, the snake structure requires a well defined and modular structure, with accurate electronics architecture and sound, functional directions through coding and algorithm.

Therefore, the modular architecture that is being presented, enables the user to move the bot to a specified location, detach a module/unit from the main modular structure for positional surveillance of sensitive targets and continue to the next position.



**Overview of the Complete Snake Bot Design:**

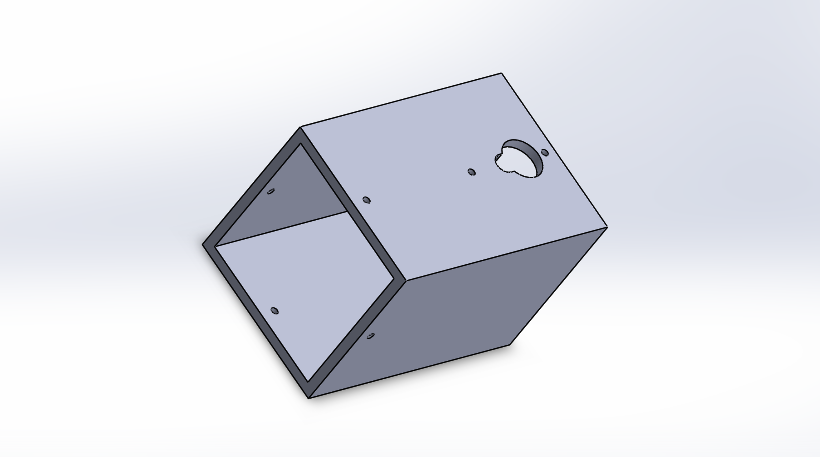


The structure is formed with **8 repeating units or modules**, each containing a power unit or battery, a microcontroller, a servo, a casing and a clamp, providing 2 degrees of freedom, uniformly across every module. The casing and clamps have contact surfaces in which, magnets are placed in order to enable modularity and presence of contact (copper) coils for communication between the said modules. Further details on the structure enabling modularity are mentioned below.

**Structural Design**

The snake bot consists of modules with identical mechanical elements. Therefore, each module consists of:

1. A Casing
2. A Swivel Clamp
3. Modular Contact Surfaces
4. The Casing:



The dimensions of the Casing made is (50 x 50 x 75) mm, with the thickness of the Casing being 5mm.

The Casing is an essential component that ensures the safety of the control systems within, from external features and elements.

It is constructed using the material, ABS or Acrylonitrile Butadiene Styrene, through the process of 3D printing.

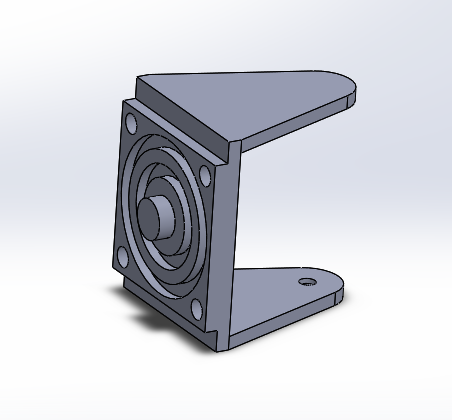
ABS was chosen as the construction material as it ensures Heat Resistance up to 110°C, with a Melting Temperature of 200°C, with the ability to maintain flexural strength of 75.84 MPa (11,000 psi).

According to the volume present inside the casing, the components maximize the space made available, but still maintaining enough separation between them to avoid any unnecessary interaction.

Internal space is also provided for additional components such as camera units, signal jammers, sensor arrays, and so on to give a unique function and purpose to each module, therefore providing the user of our Snake Bot a wide range of field functionality and flexibility.

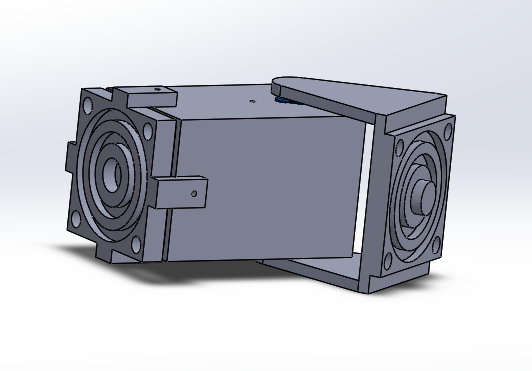
The presence of circular cutouts on one of the sides of the casing is to provide easy servo integration into the module, also liberating the corn (shaft) to be attached to the Swivel Clamp and two narrow holes to hold the servo firmly in place using screws.

2) The Swivel Clamp:

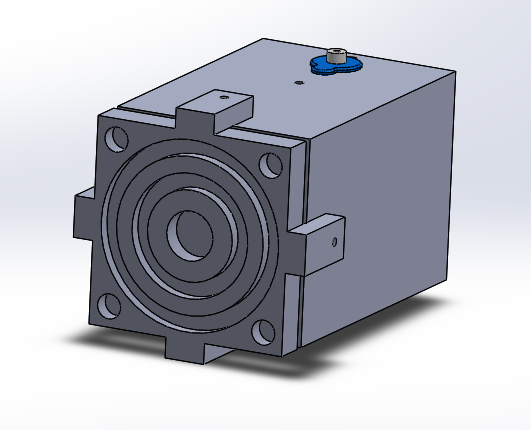


It is a triangular swivel clamp that connects to the servo, and transfers the torque produced by it, to the next module, through the connection made by the contact surfaces between them.

3) The Modular Contact Surfaces and its Mechanism:



Firstly, an individual part, with grooves made for 2 contact coils and magnets on the four corners, is connected to the main casing structure using four extensions from the part, each housing a hole, that are aligned to the holes on the Casing through which the screws will be inserted to hold the structure firmly in place.



Now, on this said part, there is a 5mm protrusion in the centre of the contact surface from the Clamp, in order to fit within the depression of the contact surface present on the part affixed to the Casing. Contact (copper) coils, the ones to be placed in the concentric grooves on the contact surfaces, will also be placed on the surfaces of the protrusion and depression that make contact when assembled. Therefore, there shall be 3 contact (copper) coils on each of the contact surfaces. Thus, this establishes the swivel motion between the modules, also allowing the flow of communication in between the modules without the use of cables, that may hinder the modular design.

The contact surface on the Clamp houses identical concentric grooves for contact coils and certain grooves for magnets on the corners of the part to align with the ones on the Casing structure.

This combinational design is essential for holding the module to the rest of the Snake Bot and detach the same, when necessary.

**Electronics and Communication Design**

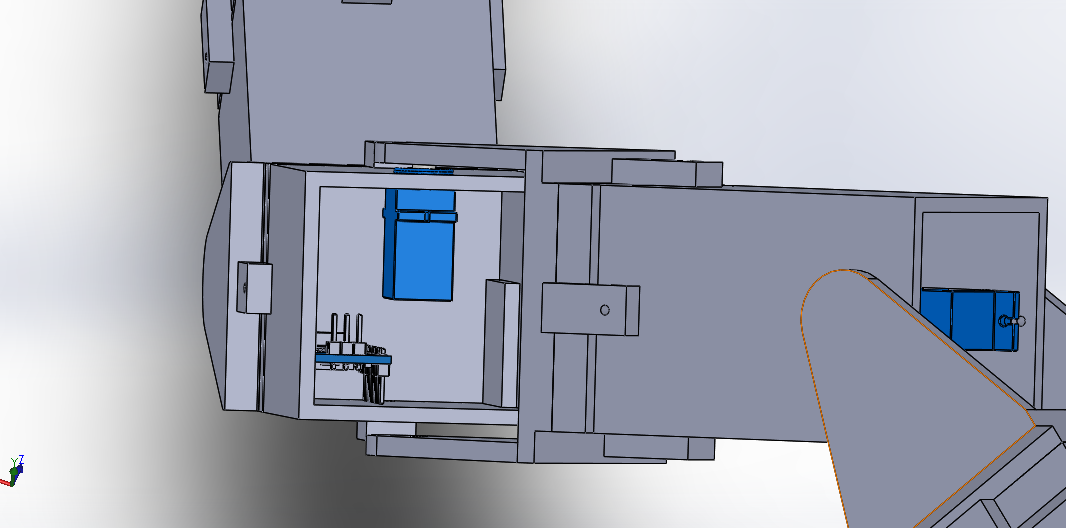
With the mechanical structure of the entire bot being independent modules, the electronics of this bot is also independent i.e each module have its own microcontroller, servo and each module are communicating with the other modules with the help of I2C communication between the microcontrollers of each module.

All the modules share a common ground and each module is connected to the forward and backward module by SCL and SDA data line using I2C protocol i.e. each module is assigned with a specific unique address so that the master controller will direct all the slave controllers to perform their specific task.

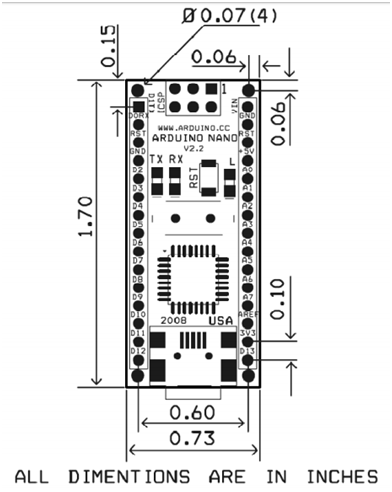
There will be a master microcontroller which will be the master for all the other (slave) microcontroller. This master controller will also receive the signals from the receiver of the RC (radio controller). And hence this entire bot can be controlled using RC.

For the modular design, each module has an independent control system. The main components used are:

1. Microcontroller - Arduino Nano
2. Power Source - Lithium Ion Battery
3. Servo Motor - Tower Pro 9G
4. Radio Controller - FS-CT6B



1. Microcontroller - Arduino Nano:



Arduino Nano is a small, compatible, flexible and breadboard friendly Microcontroller board, based on ATmega328p (Arduino Nano V3.x) / Atmega168 (Arduino Nano V3.x). It comes with exactly the same functionality as in Arduino UNO but quite in small size. It comes with an operating voltage of 5V, however, the input voltage can vary from 7 to 12V. Arduino Nano Pinout contains 14 digital pins, 8 analog Pins, 2 Reset Pins & 6 Power Pins. Each of these Digital & Analog Pins is assigned with multiple functions but their main function is to be configured as input or output.

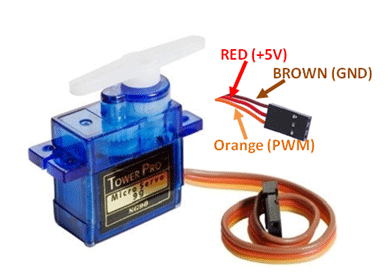
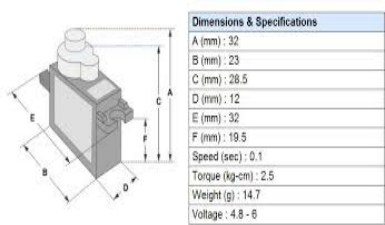
2. Power Source - Lithium Ion Battery:



Dimensions - 40\*25\*3 mm; Output voltage - 3.7V; Capacity - 500 mAh.

The Battery’s dimensional size and its capacity are well within our requirements for the components of the bot. It has the vital role of providing power to its respective module when attached and also detached from the main chain structure.

3. Servo Motor - Tower Pro 9G:



Operating Voltage is +5V typically; Torque: 2.5kg/cm; Operating speed is 0.1s/60°; Gear Type: Plastic; Rotation : 0°-180°; Weight of motor : 9gm; Package includes gear horns and screws.

One of the, if not the most important component that is responsible for the motion, propulsion and functionality of the modules and therefore the entire Snake Bot, is the Servo. It is affixed to the Casing of the module, and provides horizontal and vertical motion, depending upon the alternate positioning of the module.

4. Radio Controller - CT6B



FS-CT6B

6 channels; RF power : less than 20db; Modulation : GFSK; Code type : 2.4Ghz no interference; Bandwidth: 500KHZ; Band: 140; Antenna length: 26mm; Power: 12v 1.5AA\*8(for transmitter), 4.0-6.5v(for receiver)

As in the above sample figure, this package includes a transmitter, a receiver & a programming cable. In this project one of the modules especially the head/master module contains the receiver of the RF controller of dimension 45\*23\*9mm with the weight of 13g, so the user can control the bot within its range.

Electronic factors of communication through Contact Surfaces:

On each modular surface, there are three contact channels i.e. one for the ground link between each Arduino module the second and third one is for SCL(serial clock) and SDA(serial data) for I2C communication between the modules. As these contacts between the modules are only contact connections instead of wired connections, it provides better flexibility between the modules and also makes the modules easily detachable.

**Algorithm and Coding**

Following up the Electronic and Mechanical designs, the bot would be programmed to be controlled by FS-CT6B Radio Controller.

The FS-CT6B can be configured by connecting it to the system and using the Flysky software.

The bot has one Master Arduino Nano board and the remaining modules each have a slave Arduino Nano.

The communication between Master Arduino and any Slave Arduino is done using the Wire library.

Sample code for Master Arduino in communication between the same and a Slave Arduino is as follows:

#include<Wire.h>

void main()

{

Serial.begin(9600); //Serial communication begins

Wire.begin(); //Begins I2C communication

}

void loop()

{

Wire.beginTransmission(8); // Start transmission to the slave arduino

Wire.write(“1”);

Wire.endTransmission(); // Stops transmission

}

The Slave to the Master Arduino will have a sample code as follows:

#include<Wire.h>

void setup()

{

Wire.begin(8);

Wire.onReceive(receiveEvent);

Serial.begin(9600);

}

void loop()

{

delay(100);

}

void receiveEvent(int x)

{

Serial.print(x); }

While the communication between the Radio Controller and Master Arduino is done using the EnableInterrupt library.

Implementing the EnableInterrupt library:

Controlling a servo :

#include<EnableInterrupt.h>

#include<Servo.h>

Servo s1;

#define SERIAL\_PORT\_SPEED 9600

#define RC\_NUM\_CHANNELS 2

#define SERVO\_PIN 5

#define RC\_CH1 0

#define RC\_CH2 1

#define RC\_CH1\_INPUT A0

#define RC\_CH2\_INPUT A1

void main()

{

pinMode(RC\_CH1\_INPUT, INPUT);

pinMode(RC\_CH2\_INPUT, INPUT);

pinMode(SERVO\_PIN,OUTPUT);

enableInterrupt(RC\_CH1\_INPUT, calc\_ch1, CHANGE);

enableInterrupt(RC\_CH2\_INPUT, calc\_ch2, CHANGE);

Serial.begin(9600);

}

void loop(unit8\_t channel, unit8\_t\_input\_pin)

{

if(digitalRead(RC\_CH1\_INPUT)==HIGH)

{

Servo.write(90);

}

else

{

Servo.write(0);}

}

Using the above-mentioned procedures along with the following algorithm, the linear progression for the bot using strafing is achieved:

#include <Servo.h>

#include <wire.h>

// Define servo objects for the snake segments

Servo s1;

Servo s2;

Servo s3;

Servo s4;

Servo s5;

Servo s6;

Servo s7;

Servo s8;

// Define variables

int forwardPin = 14; // Remote control movement pins

int reversePin = 15;

int rightPin = 16;

int leftPin = 17;

int forwardVal = 0; // Remote control variables

int reverseVal = 0;

int rightVal = 0;

int leftVal = 0;

int counter = 0; // Loop counter variable

float lag = .5712; // Phase lag between segments

int frequency = 1; // Oscillation frequency of segments.

int amplitude = 40; // Amplitude of the serpentine motion of the snake

int rightOffset = 5; // Right turn offset

int leftOffset = -5; // Left turn offset

int offset = 6; // Variable to correct servos that are not exactly centered

int delayTime = 7; // Delay between limb movements

int startPause = 5000; // Delay time to position robot

int test = -3; // Test variable takes values from -6 to +5

void setup()

{

Wire.begin();

// Set movement pins as inputs

pinMode(forwardPin, INPUT);

pinMode(reversePin, INPUT);

pinMode(rightPin, INPUT);

pinMode(leftPin, INPUT);

// Set movement pins to low

digitalWrite(forwardPin, LOW);

digitalWrite(reversePin, LOW);

digitalWrite(rightPin, LOW);

digitalWrite(leftPin, LOW);

// Attach segments to pins

s1.attach(2);

s2.attach(3);

s3.attach(4);

s4.attach(5);

s5.attach(6);

s6.attach(7);

s7.attach(8);

s8.attach(9);

// Put snake in starting position

s1.write(90+offset+amplitude\*cos(5\*lag));

s2.write(90+offset+amplitude\*cos(4\*lag));

s3.write(90+offset+amplitude\*cos(3\*lag));

s4.write(90+amplitude\*cos(2\*lag));

s5.write(90+amplitude\*cos(1\*lag));

s6.write(90+amplitude\*cos(0\*lag));

s7.write(90+amplitude\*cos(-1\*lag));

s8.write(90+amplitude\*cos(-2\*lag));

delay(startPause); // Pause to position robot

}

void loop()

{

// Read movement pins

forwardVal = digitalRead(forwardPin);

reverseVal = digitalRead(reversePin);

rightVal = digitalRead(rightPin);

leftVal = digitalRead(leftPin);

// Forward motion

if (forwardVal == HIGH){

for(counter = 0; counter < 360; counter += 1)

{

delay(delayTime);

s1.write(90+offset+amplitude\*cos(frequency\*counter\*3.14159/180+5\*lag));

s2.write(90+offset+amplitude\*cos(frequency\*counter\*3.14159/180+4\*lag));

s3.write(90+offset+amplitude\*cos(frequency\*counter\*3.14159/180+3\*lag));

s4.write(90+amplitude\*cos(frequency\*counter\*3.14159/180+2\*lag));

s5.write(90+amplitude\*cos(frequency\*counter\*3.14159/180+1\*lag));

s6.write(90+amplitude\*cos(frequency\*counter\*3.14159/180+0\*lag));

s7.write(90+amplitude\*cos(frequency\*counter\*3.14159/180-1\*lag));

s8.write(90+amplitude\*cos(frequency\*counter\*3.14159/180-2\*lag));

}

}

// Reverse motion

if (reverseVal == HIGH){

for(counter = 360; counter > 0; counter -= 1) {

delay(delayTime);

s1.write(90+offset+amplitude\*cos(frequency\*counter\*3.14159/180+5\*lag));

s2.write(90+offset+amplitude\*cos(frequency\*counter\*3.14159/180+4\*lag));

s3.write(90+offset+amplitude\*cos(frequency\*counter\*3.14159/180+3\*lag));

s4.write(90+amplitude\*cos(frequency\*counter\*3.14159/180+2\*lag));

s5.write(90+amplitude\*cos(frequency\*counter\*3.14159/180+1\*lag));

s6.write(90+amplitude\*cos(frequency\*counter\*3.14159/180+0\*lag));

s7.write(90+amplitude\*cos(frequency\*counter\*3.14159/180-1\*lag));

s8.write(90+amplitude\*cos(frequency\*counter\*3.14159/180-2\*lag));

}

}

// Right turn

if (rightVal == HIGH){

// Ramp up turn offset

for(counter = 0; counter < 10; counter += 1) {

delay(delayTime); s1.write(90+offset+.1\*counter\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+5\*lag)); s2.write(90+offset+.1\*counter\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+4\*lag)); s3.write(90+offset+.1\*counter\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+3\*lag)); s4.write(90+.1\*counter\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+2\*lag)); s5.write(90+.1\*counter\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+1\*lag)); s6.write(90+.1\*counter\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+0\*lag)); s7.write(90+.1\*counter\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180-1\*lag)); s8.write(90+.1\*counter\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180-2\*lag));

}

// Continue right turn

for(counter = 11; counter < 350; counter += 1) {

delay(delayTime); s1.write(90+offset+rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+5\*lag));

s2.write(90+offset+rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+4\*lag));

s3.write(90+offset+rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+3\*lag));

s4.write(90+rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+2\*lag));

s5.write(90+rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+1\*lag));

s6.write(90+rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+0\*lag));

s7.write(90+rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180-1\*lag));

s8.write(90+rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180-2\*lag));

}

// Ramp down turn offset

for(counter = 350; counter < 360; counter += 1)

{

delay(delayTime); s1.write(90+offset+.1\*(360-counter)\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+5\*lag)); s2.write(90+offset+.1\*(360-counter)\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+4\*lag)); s3.write(90+offset+.1\*(360-counter)\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+3\*lag)); s4.write(90+.1\*(360-counter)\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+2\*lag)); s5.write(90+.1\*(360-counter)\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+1\*lag));

s6.write(90+.1\*(360-counter)\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180+0\*lag)); s7.write(90+.1\*(360-counter)\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180-1\*lag)); s8.write(90+.1\*(360-counter)\*rightOffset+amplitude\*cos(frequency\*counter\*3.14159/180-2\*lag));

}

}

// Left turn

if (leftVal == HIGH){

// Ramp up turn offset

for(counter = 0; counter < 10; counter += 1) {

delay(delayTime); s1.write(90+offset+.1\*counter\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+5\*lag)); s2.write(90+offset+.1\*counter\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+4\*lag)); s3.write(90+offset+.1\*counter\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+3\*lag)); s4.write(90+.1\*counter\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+2\*lag)); s5.write(90+.1\*counter\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+1\*lag)); s6.write(90+.1\*counter\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+0\*lag)); s7.write(90+.1\*counter\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180-1\*lag)); s8.write(90+.1\*counter\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180-2\*lag));

}

// Continue left turn

for(counter = 11; counter < 350; counter += 1)

{

delay(delayTime);

s1.write(90+offset+leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+5\*lag));

s2.write(90+offset+leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+4\*lag)); s3.write(90+offset+leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+3\*lag));

s4.write(90+leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+2\*lag));

s5.write(90+leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+1\*lag));

s6.write(90+leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+0\*lag));

s7.write(90+leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180-1\*lag));

s8.write(90+leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180-2\*lag));

}

// Ramp down turn offset

for(counter = 350; counter < 360; counter += 1)

{

delay(delayTime);

s1.write(90+offset+.1\*(360-counter)\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+5\*lag)); s2.write(90+offset+.1\*(360-counter)\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+4\*lag)); s3.write(90+offset+.1\*(360-counter)\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+3\*lag)); s4.write(90+.1\*(360-counter)\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+2\*lag)); s5.write(90+.1\*(360-counter)\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+1\*lag)); s6.write(90+.1\*(360-counter)\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180+0\*lag)); s7.write(90+.1\*(360-counter)\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180-1\*lag));

s8.write(90+.1\*(360-counter)\*leftOffset+amplitude\*cos(frequency\*counter\*3.14159/180-2\*lag));

}

}

}

With further improvisation of the code and integration, many more functions can be added to the bot which will allow it to move in various ways. The gaits which can be achieved are:

1. Corkscrewing

A unique gait, corkscrewing, causes the snake robot to spiral its body, and propagation these spirals back through its body, propelling it forward. It is useful for travelling forwards or backwards in the presence of obstacles when linear progression cannot easily propel the robot. Corkscrewing is also valuable when manoeuvring through a small hole in a wall or a chain link fence. In this configuration the servos attached to the head and tail rotate in the vertical plane, all other servos rotate in the horizontal plane.

1. Side Winding

It is when the serpent wants to move sideways without taking a turn.

1. Rolling

Rolling is a useful feature to have to make the serpent move side waves quickly. The snake curves slightly into a 'C' shape and then rolls. Rolling is also one of our most energy efficient gaits since the snake's momentum is conserved. It is useful for uneven terrain and excels at climbing hills.

1. Channel/Pipe Climbing

Channel climbing is a variant of linear progression, where the amplitude and period of the sine wave are adjusted to fit the chosen channel or pipe. Channel climbing snake robots are often fitted with protective skin or rubber to provide additional friction and compliance. No adhesives are used to achieve snake robot channel climbing, only outward pressure provided by the sine waves.

1. Pole climbing

Pole climbing refers to moving upwards on cylinders whose perimeter is less than the length of the snake robot. The robot moves upwards without the aid of an adhesive by spiralling its body around the pole, gripping it, and using the rolling gait to travel up or down the pole. Snake robots are able to transition directly from another gait to pole climbing. It is the combination of rolling and corkscrewing.

1. Cornering

When it's in a place where it needs to blend in with the surrounding to drawing minimum attention but still needs to move on. This technique of cornering helps. A bend in the snake matching the bend of the corner or wall and the bend is maintained relative to the shape/corner as the snake moves forward.

1. Stair Climbing

Stair climbing is a composite behaviour of several simpler gaits and behaviours. The front section of the snake robot rises onto the stair, and the snake robot transfers as much weight as it needs to lift itself onto the next stair. The snake robot is able to climb multiple stairs in this manner.

**Challenges**

* Fabrication of such modules will introduce a layer of complication as 3D printing is not the most reliable procedure and is also time-consuming.
* Effective detachment and then attachment of modules is difficult as the alignment of the modules requires a high level of positional accuracy.
* The wireless communication using contact coils need proper physical contact between them to ensure no latency and missed input commands from the user.

**Limitations**

* The individual modules, after detaching from the main chain structure can only function according to the rated capacity of the battery, after which it will not be functional.
* After detaching from the main chain structure, a single module will only be left stationary as it cannot receive any more command inputs from the master module that houses the RC receiver.

**Additional Components**

Components that can be added easily to the unit module to greatly increase its functionality, include:

1. Infrared Camera
2. Proximity Sensor
3. Miniature Signal Jammer
4. Motion Detector
5. Air Sample Quantifier, and so on providing endless possibilities.

**Conclusion**

Project Medusa’s main aim is to create a snake bot that has the capabilities of traditional bots but provides a plethora of opportunities for the user according to the conditions, tasks and overall mission at hand. The modular function provided by our design is unique in its implementation and simple in its construction and application.

Softwares used:

* Solidworks 2018
* Arduino Program
* Microsoft Office

**Authors:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr.No. | Name | Email ID | Phone No. | Class |
| 1 | Sahil Harishchandra Sanil | sahil26sanil@gmail.com | 9867752500 | 1st Year |
| 2 | Prabhat Kumar | prabhatkumar\_ha@srmuniv.edu.in | 7367043412 | 1st Year |
| 3 | Soham Manoj Parlikar | parlikarsoham\_manoj@srmuniv.edu.in | 8097229916 | 1st Year |
| 4 | Kanishk Jain | kanpursha@gmail.com | 7550265627 | 1st Year |
| 5 | S Arut Prakash | arutprakash2000@gmail.com | 9566128058 | 1st Year |