A PROJECT REPORT ON

Snake Robot

Submitted to

The Innovation Cell,
IIT Bombay,
summer internship 2014

By the team of 7 members of B.Tech second year mentored by Mr. Pranjal Jain.

Name	Branch	College
Aditya Deole	Mechanical engineering	VNIT,Nagpur
Aditya Vidolkar	Electrical and Electronic Eng.	VNIT,Nagpur
Akash Singh	Electronics and communication Eng.	VNIT,Nagpur
Anubhav Paras	Electrical and Electronic Eng.	VNIT,Nagpur
Manish Saroya	Electrical and Electronic Eng.	VNIT,Nagpur
Omey Manyar	Mechanical Engineering	VNIT,Nagpur
Shivraj Dalu	Electrical and Electronic Eng.	VNIT,Nagpur

ACKNOWLEDGMENT

I hereby express my sincere thanks to respected Dr Abhishek Gupta Sir, Dr. S. N. Merchant and Dr. R K Singh for giving me the chance to be the part of this project.

I also take an opportunity to express my deep sense gratitude to Innovation Cell, IIT Bombay for providing all the required facilities and inspiration in bringing out this project work.

I am indebted to my mentor Ms. Pranjal Jain who always guided and helped me whenever needed.

INDEX

SNAKE ROBOT

ABSTRACT

Snake Robots have many degrees of freedom, which makes them extremely versatile and complex to control. This report presents modular snake robot, its electronic architecture and control. Inspired by biological snake, snake robot moves using cyclic motions called gaits. These cyclic motions directly control the snake robot's internal degrees of freedom which causes a net motion. Each mode of the robot is controlled by a sinusoidal oscillator with four parameters: amplitude, frequency, phase, and offset.

INTRODUCTION

Snake like robots have structural characteristics such as multi-degrees of freedom, multi-joints and modular structure, which allows them to move using various methods and good adaptabilities. Compared to wheeled and legged mobile mechanisms, the snake robot offers high stability. It can travel through almost all types of terrains. So it can be used for many applications such as rescue missions, fire-fighting and maintenance where it may either be too narrow or too dangerous for personnel to operate.

We have made a prototype which can basically perform various types of snake movements. Our robot is made up of 10 modules which are connected one after the other. Among them half of the modules move in vertical direction and remaining in horizontal direction.

Technical Background

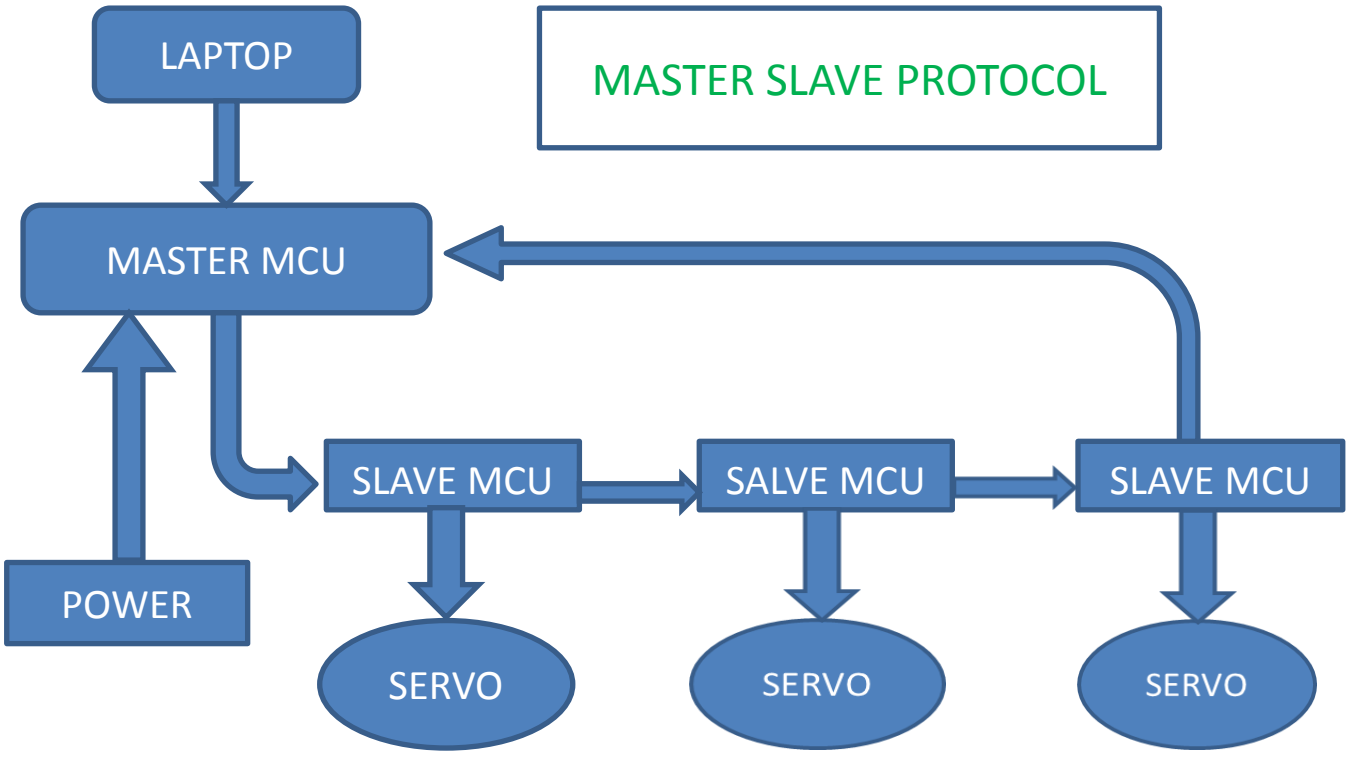
People had tried to make similar technology which can co-ordinate with their internal degree of freedom to perform various locomotive GAITS. But our robot is highly articulated and robust which can pass through compact places where the legged robots can't reach. We also made one graphic user interface which make our device easy to use.

PROPOSED SOLUTION

First Prototype

Revival of servo actuated snakebot. The older bot designed last year was revived and all basic gates were performed on this bot. The modular links for this snake were rectangular so that the center of mass was not shifted out of the base easily. But this design was not symmetric about center and stress distribution was not uniform. Thus it could not facilitate climbing and rolling operations.

Another versatile design was proposed using different approach on electronic and software level. Our main purpose of this type of approach was to incorporate more numbers of links and also to eliminate dynamixel for cheaper and easily available servos.
Each module of the bot had its own microcontroller to control the servos and the commands were sent by a master microcontroller. This is called master-slave protocol , here the master communicates between all the slaves one by one sending data bit by bit. The clock is set the master only and it can also select the slave it wants to communicate with. The slave receives these commands and processes them accordingly and gives feedback if needed.



This block diagram the electronic design of servo actuated snakebot
The two pursued communication protocols were I2C and SPI. Both the master and slave used were ATMEGA8 and clock set was 8 MHz . The microcontroller was programmed in c/c++ using AVRstudio and Winavr libraries.

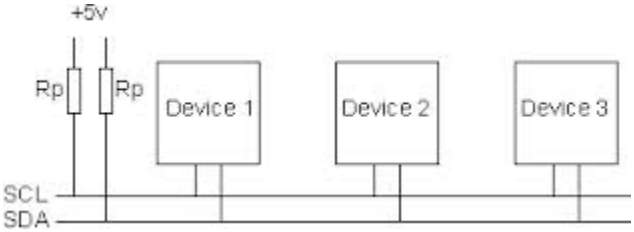
I2c:I²C

Inter-Integrated Circuit is a multimaster serial single-ended computer bus invented by the Philips semiconductor division.I²C uses only two bidirectional open-drain lines, Serial Data Line (SDA) and Serial Clock Line (SCL), pulled up with resistors. Typical voltages used are +5 V or +3.3 V although systems with other voltages are permitted. The I²C reference design has a 7-bit or a 10-bit (depending on the device used) address space. Common I²C bus speeds are the 100 Kbit/s standard mode and the 10 Kbit/s low-speed mode, but arbitrarily low clock frequencies are also allowed.
Why to use i2c:

- The most important advantage of i2c system is that the whole data transmission is done using only two wires. One line is for the data transmission and the other is for the clock. The data transmission is synchronized with the clock line.
- The other advantage of i2c is that the data is never lost during transmission. The slave acknowledges each and every bit of data it receives. Hence the data transmission is very effective.
- Around 127 slaves can be connected in series in i2c having different addresses.
- I2C includes a "protocol" for addressing, acknowledging, multi-master. While in other transmission protocols one has to do his own self.

I2C multiple servo control: The most difficult part faced during the servo control was to setting up multiple motors simultaneously using only two wires. Problem faced during this is that the TWI line is kept busy every time any one of the servos work and the microcontroller associated with it performs an infinite loop. Due to this the TWI wire is held by the first slave and no data can be passed to the second slave. Hence the data that is to be transferred to different slaves having different addresses can’t be received by the slaves except the first. Hence all the slaves don’t work simultaneously. So the best trick is to give all the data serially through the same first address. And using only the respective data in the whole sequential data dedicated to the particular slave and leaves the rest of the data

for other slaves to utilize their own data.



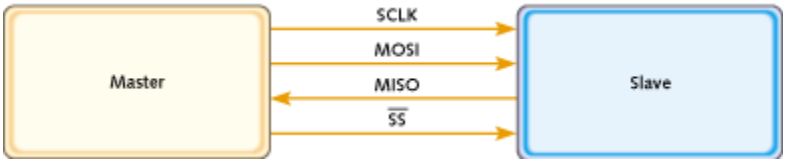
I2c has different modes of communication such as:

- Master in transmitter and slave as receiver and vice versa.
- Master in read/write mode and slave in read/write mode.

We have used the single master in write mode and the slaves in multiple receive mode.
The link for the multiple servo control mode is

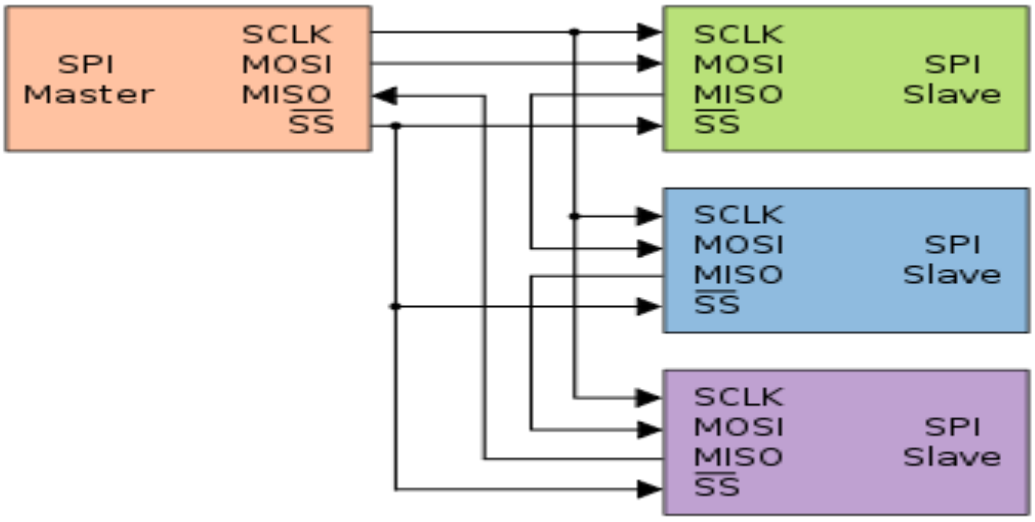
SPI

The Serial Peripheral Interface or SPI bus is a synchronous serial data link developed by Motorola. Devices communicate in master/slave mode where the master device initiates the data frame. Multiple slave devices are allowed with individual slave select lines. Sometimes SPI is called a four-wire serial bus, contrasting with three-, two-, and one-wire serial buses. SPI is often referred to as SSI (Synchronous Serial Interface).



Master selects the slave by holding its Slave Select line to low. Only the slave which has low input on SS gets activated.
In the snake robot we used SPI to form a daisy chain and the data is transferred by the master to the first slave and the slave transfers the data to the second slave and similarly the data transmission continues from the first slave to the second slave. The daisy chain link continues. The master receives the data from the last slave forming the last link of the daisy chain. Hence the daisy chain forms a complete cyclic loop and the data transmission is done. Hence if the data has to be transferred from the master to the last slave the data gets transferred from the master to the first slave and the same data gets transferred from the first slave to the second slave and hence the process continues till the data is received by the last slave.
As seen here daisy chaining reduces number of wires and as the master does not

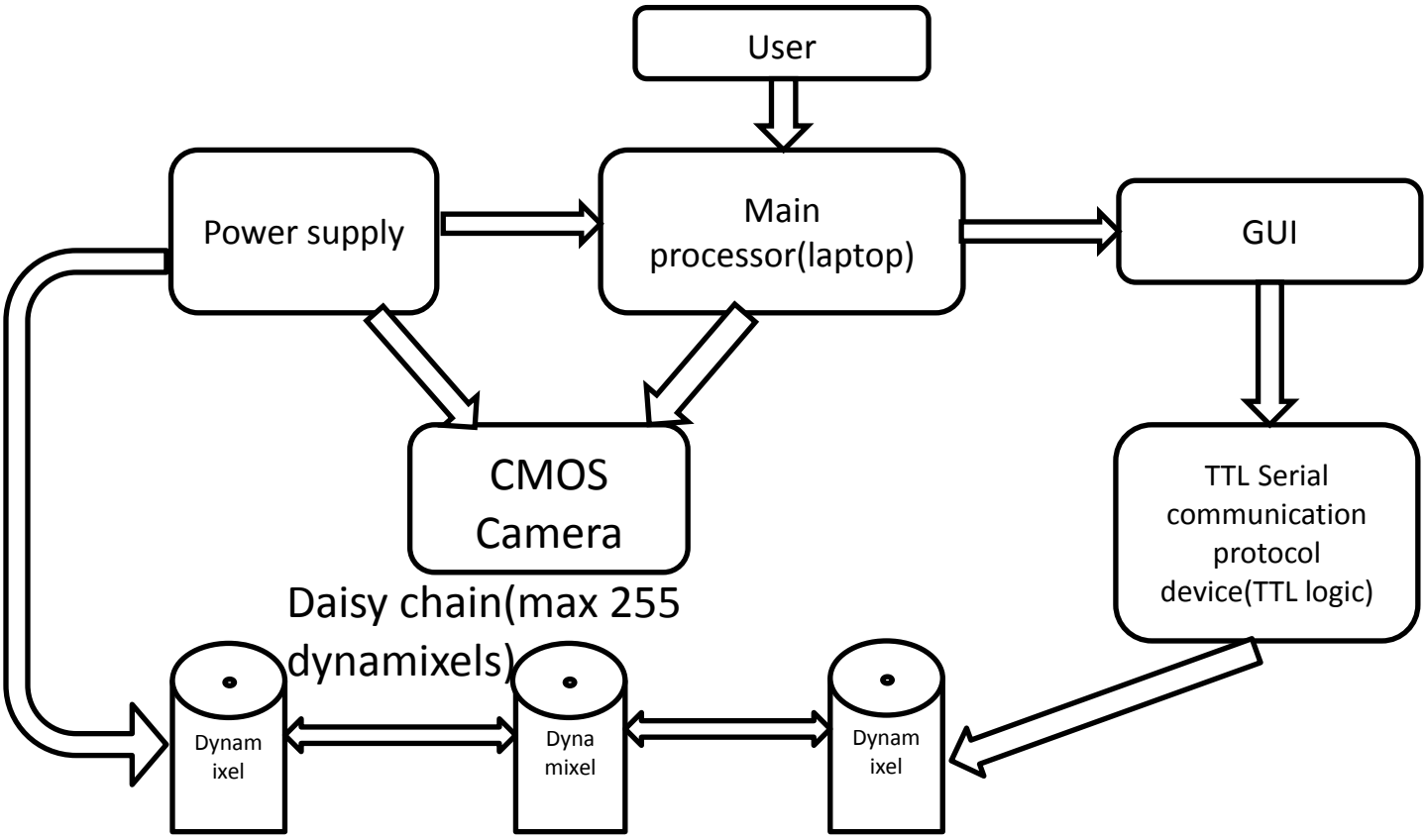
have to hold a particular slave low so the SS pins of all slaves directly grounded.



Advantages of SPI:

- It's a full duplex communication i.e. simultaneous transfer and receiving of data from master and slave.
- Higher throughput than I^2C.
- Complete protocol flexibility for the bits transferred
- It's quite simple to understand and the hardware connections are also very easy to implement.
- Addressing of individual slaves is not required

Block diagram system level showing input and outputs of the systems.



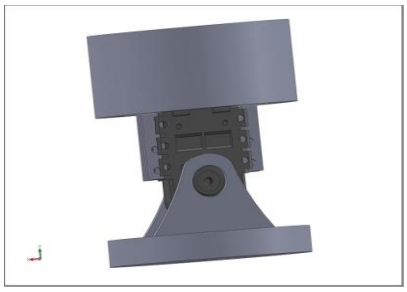
In the arrow shows the respective input and output of the systems in blocks. In snake we have made a GUI that can be used by the user to control the dynamixels with a single click of a button without wasting any time in writing commands in the terminal of Ubuntu. There is a camera on snake so that it can be used for surveillance purpose.

IMPLEMENTATION

A. Hardware implementation

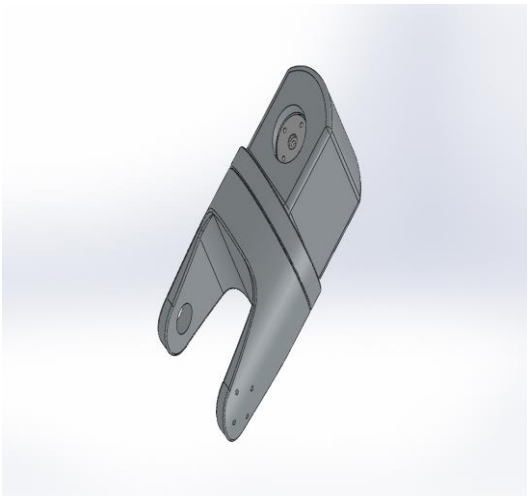
- Mechanical Design:
  - The Snake Robot being a class of hyper redundant robots makes them versatile and their high degree of freedom makes them very helpful in various applications.
  - To achieve these applications various design parameters have to be taken into consideration.
  - Various factors had to be taken in mind for the mechanical design such as:
    1) The different types of gaits needed to be implemented
    2) The weight of the robot
    3) The Degrees Of Freedom of the robot
    4) Type of actuator used
    5) Type of power supply
    6) Method of communication(Wired or Wireless)
  - After taking into consideration the above factors the process of design was initiated and a single modular link was designed.

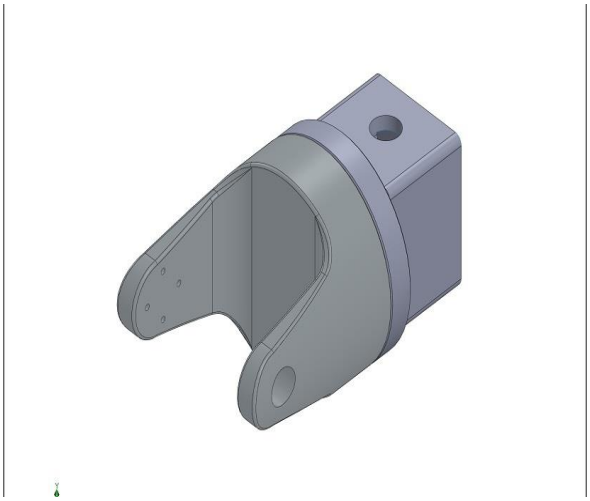
- The Design Of Links:
  - The gaits which we decided to implement were rolling, side winding, lateral undulation, climbing, caterpillar and swimming.
  - So we decided to make a snake robot which can work underwater also, hence we decided to go with closed link approach.
  - We used DYNAMIXEL AX-12A as our actuator and we came up with our first design of the link using dynamixels.



MODEL OF INITIAL LINK CAD

- But there were certain problems in this link with respect to the manufacturing aspects as producing a hollow casing cannot be done by manual milling, more about manufacturing problems is dealt in the following sub sections.
- So after reconsidering all the factors we came up with a better design for the robot and this link was finalized.
- All the designs and simulations were done using Dassault Systems SOLIDWORKS 2013 software.





LINK BEFORE WEIGHT REDUCTION **LINK AFTER WEIGHT REDUCTION**

❖ IMAGES OF THE LINKS:

➢ THE MAIN LINK:

- THE HEAD OF THE SNAKE ROBOT:

- THE TAIL OF THE SNAKE ROBOT:

B. Material used:

- Light weight and Good tensile strength were the primary criteria for selection of materials.

- Light weight was one of the most important factors for the design of the link.
- Aluminum and nylon were the ones to be shortlisted at the beginning.
- Finally we decided to go with a new type of material called DELRIN.(polyoxymethylene plastic $(CH_2O)n$)
- Its density being 1.41 g/cm^3 almost half of that of aluminum.
- This material was cost effective and had good machinability and hence was selected as the material for manufacturing the links.

DERLIN

- But using delryn wasn't enough as we weren't getting enough friction to execute climbing gait.
- There was slipping between the surface of the tree and the surface of the robot and the torque wasn't enough for the robot to successfully climb the tree.
- So we came up with an idea of covering the outer chassis with a material that can provide necessary friction.
- Initially we went forward with a Bat Grip as a covering but it was a failure, so finally we decided to go with anti skid tapes which provided us with the necessary friction for the robot to climb.

❖ ANTI SKID TAPES:

❖ OUR ROBOT COVERED WITH THE ANTI SKID TAPE:

C. Manufacturing:

- The most widely used manufacturing processes are milling, lathe machining, molding, water jet cutting, etc. Milling being the most important among these.
- Manual milling was ruled out as a machining process as the design parameters had certain constraints.
- The rectangular cavity couldn't be done using manual milling machine, hence we opted for CNC milling process.
- Once the outer skeleton was ready we made the holes for the horn of the motor to fit and also drilled the remaining holes that will be required for fastening purposes.
- In the later stage the extra material was removed using a 3 axis universal milling machine.

UNIVERSAL MILLING MACHINE

B. Software implementation

We were in the need of a distributive platform which could communicate with the actuators. This need was full-filled by ROS which is open source operating system. The Robot Operating System (ROS) is a set of software libraries and tools which is useful for building software applications. ROS makes our software system truly robust and it can fully function on Ubuntu.

At the lowest level, ROS can be used to build a message passing interface that provides inter-process communication. A communication system is often one of the first needs to arise when implementing a new robot application. ROS's message passing can be used for communication between distributed nodes via the anonymous publish/subscribe mechanism. Another benefit of using a message passing system is that it forces you to implement clear interfaces between the nodes in your system, thereby improving encapsulation and promoting code reuse.

Setting up environment in ROS which will make our robot run.

Firstly in ROS we created a workspace named cobra within it we created a package named transformer. In src of transformer package we created rosnodes.

Some for generating message information and publishing them on a topic named gait_topic and the some others for subscribing information from the topic.

Explanation of the files included in src folder of transformer package:-

1. **Climbing.cpp**: For publishing climbing gait information

2. **Dynamix.cpp**: This node is to publish the gait information as prescribed in the gui. It can set the amplitude, angle and speed of individual motors as entered in gui.

3. **Sidewinding.cpp**: This node publishes the gait information of sidewinding motion.

4. **Moveforward.cpp**: This node generate the information of forward moving gait. This gate resembles the caterpillar motion. The snake robot can also be made to move backward such that the phase of the vertical motors is set exactly opposite and the robot moves backward according to the amplitude and the speed given through gui.
5. **Mamba.cpp**: This node is to publish the information about the mamba motion gait.

6. **Feedback.cpp**: This node takes feedback (position, speed, error) from a particular dynamixel as prescribed and give it to another dynamixel so that the second dynamixel can replicate the motion of the first dynamixel. The feedback.cpp consists of two node operating simultaneously. This node subscribes to two publishers. The node subscribes to the first publisher node known as the feedback manager node of dynamixel tutorials package. After receiving the data the same data is published to the second set of dynamixel motors through the communicator node. The data transmission occurs normally through the UART. The data given to the second set is related to that of the goal position of the first dynamixel rather than the present location in order to avoid the lag in the movement of the second dynamixel. Hence the second set of dynamixels actually mimic the first set without any time lag.

7. **Rotateright.cpp**: This node publishes the information of gait which will rotate the snake in right direction around a particular point.

8. **Rotateleft.cpp**: This node publishes the information of gait which will rotate the snake in left direction around a particular point.

9. **Rolling.cpp**: This node publishes the data of rolling gait.

10. **Dynamixel.h**: This is the header file which contains the function used for serial transmission of data to the dynamixel.

11. **Communicator.cpp**: this is actually a transmitter and subscriber. This subscribes the data from the topic named gait_topic(all the above nodes publish on this node) and transmit it through UART to the usb2dynamixel. Further this data is transferred to the dynamixel through TTL serial communication by the USB2Dynamixel.

12. **Gait.cpp**: This node is similar to the other nodes but it has a precious advantage over the others. This node is used specifically to generate the gaits without any torque constraint on the motors. The angles are fed on the n(no. of motors) 1d arrays and these angles are regained depending upon the time duration from the time the node was started on the ros master. The speeds of the angles are attained as a function of the available time and this system is really useful to simply generate any gait by simply providing the angles.

    The type of information which we talked about is in form of set of array of values viz. header, motor_id, angle_value, torque_off . And this set of values is kept in a file named Angle.msg which is include in msg folder.

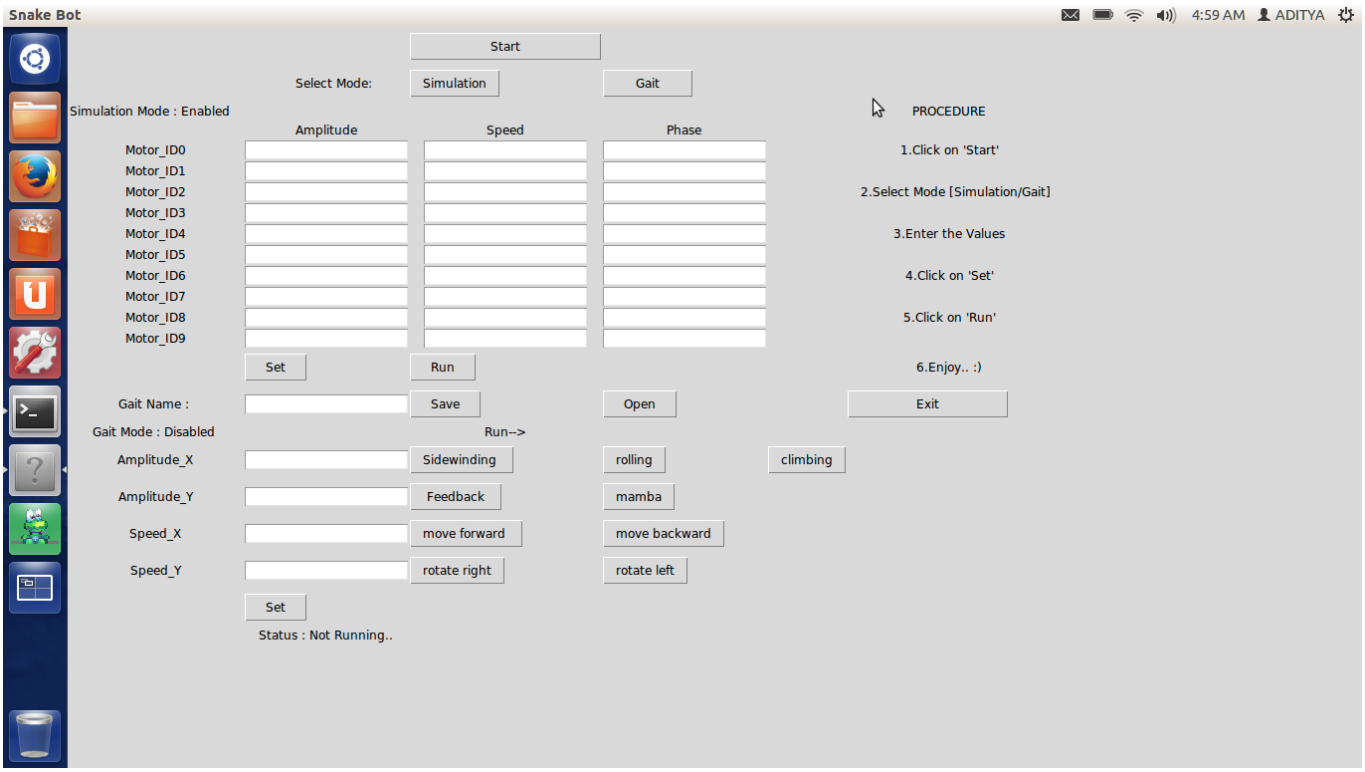
Connecting bus and joint controller for dynamixel: This topic includes the rosbuild packages scripted in python. The python package dynamixel_tutorials consisting of the nodes such as the controller_manager and controller_spawner helped controlling the dynamixel and by a single roslaunch command in which the angle scaled within -2.7 to 2.7 could be given in order to control the position of the dynamixel.

The controller_manager node: The node is very helpful in giving the feedback of all the dynamixels connected in daisy chain. The feedback message consists of the temperature, angles and other information of its goal position.

GUI (GRAPHICS USER INTERFACE)

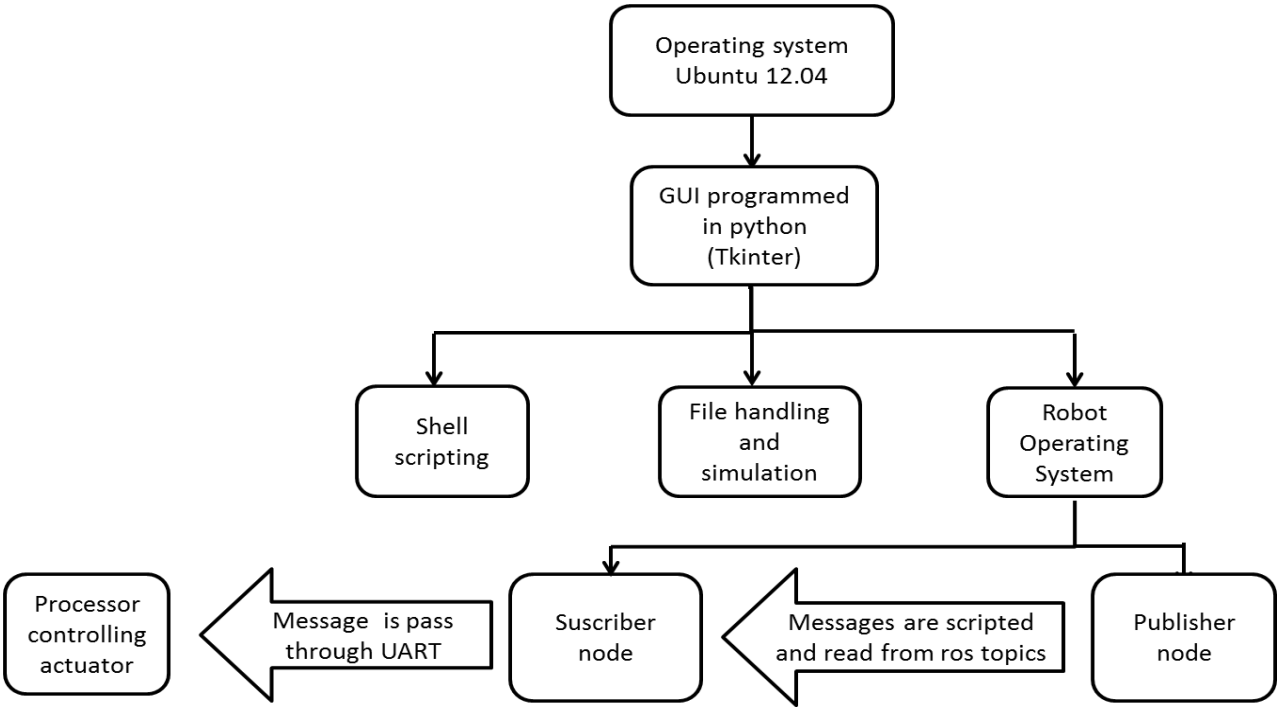
To make our robot user friendly we came up with an idea of making GUI (Graphics user interface). The GUI was programming in python using packages of Tkinter a cross platform. Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. It is a thin object-oriented layer. One can easily use this GUI as we have provided procedure within it. It saves time as we don't have to type commands in the terminal again and again. With the help of this GUI we can also give different angles, speeds and phases to different dynamixels in simulation mode, so attaining specific shape becomes very easy for any user. There is also a provision to save these simulation data in a file. We have also made different buttons for different gaits like sidewinding, rolling, forward ,backward, left and right motion in gait mode so that on one click the bot starts to perform a particular type of gait with different speeds and amplitudes. We can also take feedback of motor conditions like temperature, torque, speed, position and error with the help of this GUI. The above procedures are performed under the gait mode which is switched on by pressing the respective gait button on the top of the gui. Each button click corresponds to its own shell script file which is a file consisting of different commands that are performed on the terminal background simultaneously. The important feature of file handling has also been included to replace the respective values in the respective cpp files in the transformer package. The user has to put minimal effort to set the required gaits he wants on the snake robot.

Below is the snapshot of the GUI.



GUI snapshot

Below is the block diagram for functions of GUI.



C. Electronics

- Actuators

The actuator used in snake robot is AX_12A servo motor(dynamixel). The AX-12A robot servo has the ability to track its speed, temperature, shaft position, voltage, and load. As if this weren't enough, the control algorithm used to maintain shaft position on the ax-12 actuator can be adjusted individually for each servo, allowing you to control the speed and strength of the motor's response. All of the sensor management and position control is handled by the servo's built-in microcontroller. This distributed approach leaves the main controller free to perform other functions. These servos can be connected in daisy chain. A daisy chain is a wiring scheme in which multiple devices are wired together in sequence or in a ring. We can also set there baud rate, id, goal position before use as per the requirement.

✓ USB2Dynamixel

USB2Dynamixel is a device used to operate Dynamixel directly from PC. USB2Dynamixel is connected to USB port of PC, and 3P and 4P connectors are installed so that various Dynamixels can be connected. Also, USB2Dynamixel can be used to change from USB port to Serial port on the PC without serial port such as notebook computer, etc. The communication mode can be selected by changing the switch of USB2Dynamixel. We used TTL communication for Dynamixels using 3-pin port. The best facility provided by the usb2dynamixel is that it is very simple to use and it avoid the logic level shifter circuit which is prepared to involve the communications in a daisy chain. Its can help the user to shift to any kind of data transmission which he can be compatible with.

USB2Dynamixel

✓ Power supply

USB2Dynamixel does not supply power to Dynamixel. Therefore, the power must be supplied separately to operate Dynamixel. We supplied power to the actuators with a three cell lipo battery which is kept off board. The smps of Texas instrument shown couldn't be used because the ampere rating of the smps is 1.5amp which could drive only a limited number of motors. Hence lithium-polymer battery was used which could supply infinite current.

GAITS AND ROBOT KINAMATICS

Gait is the pattern of movement of the limbs of animals during locomotion. Snake bot gaits are often designed by investigating periodic changes to the shape of the robot. They can move by adapting their shape to different periodic functions.

Our bot is made up of 10 links where single degree of freedom rotary joints are alternatively oriented in vertical and horizontal plane of robot. Because of this design our gait equations consist of two separate equations which propagate through vertical and horizontal joints. The equations are as follow

$$\theta_{vi} = A_{ver} \sin(\omega_{ver} t + (i-1) \delta_{ver}) + O_{ver}$$

$$\theta_{hi} = A_{hor} \sin(\omega_{hor} t + (i-1)\delta_{hor} + \delta_o) + O_{hor}$$

θvi = Angle of i th vertical link
θhi = Angle of i th horizontal link
Aver = Amplitude of i th vertical link
Ahor = Amplitude of i th horizontal link
ωver = Frequency of vertical sine wave
ωhor = Frequency of horizontal sine wave
δver = Phase difference between two consecutive vertical links

δhor = Phase difference between two consecutive horizontal links

δo = Phase difference between two adjacent links

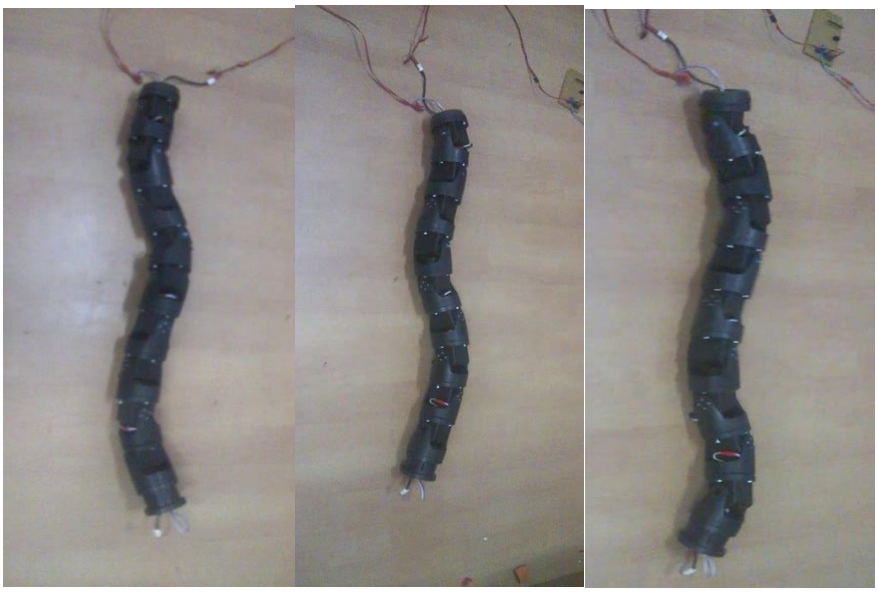
Over = default orientation vertical links

Ohor = default orientation horizontal links

✓ *SIDEWINDING*

Sidewinding is a unique type of locomotion used by some snakes to move across loose or slippery substrates. In sidewinding, the ground contact segments are in static contact with the ground. At their leading edges, the snake is continually lowering itself to the ground, while simultaneously peeling itself up at the trailing edge. The raised portions of the snake's body are arch segments. As the snake sidewinds, the ground contact segments leave ground contact tracks against the terrain. This gait can be achieved by setting the values of variables in the above equations as follows:-

$A_{ver}=pi/6$, $A_{hor}=pi/6$, $\omega_{ver}=5pi/6$, $\omega_{hor}=5pi/6$,
$\delta_{ver}=2pi/3$, $\delta_{hor}=2pi/3$, $\delta_{o}=0$, $O_{ver}=0$, $O_{hor}=0$

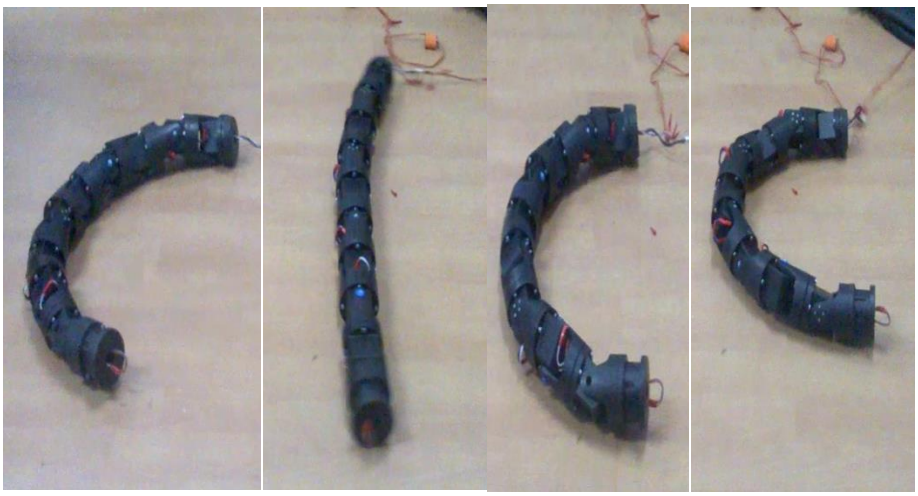


✓ *ROLLING*

In basic rolling gait, the robot forms a static backbone shape. It forms a shape like an arc of constant curvature and then rolls within the shape. Robot remains level on the ground during this motion and as it cycles through the gait, it rolls either towards or away from the center of the arc. This gait can be achieved by setting the values of variables in the above equations as follows:-

$A_{ver}=pi/3$, $A_{hor}=pi/3$, $\omega_{ver}=5pi/6$, $\omega_{hor}=5pi/6$,

$\delta_{ver}=0$, $\delta_{hor}=0$, $\delta_{o}=pi/6$, $O_{ver}=0$, $O_{hor}=0$

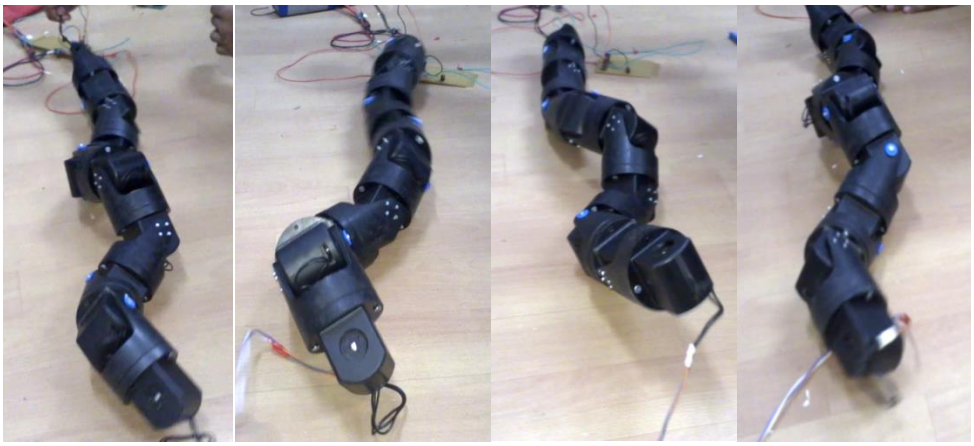


`

✓ *CORKSCREW MOTION*

A unique gait, corkscrew motion, causes the snake robot to spiral its body, and propagating these spirals bask through its body, propelling it forward. It is useful for travelling forward or backward in the presence of obstacles, when linear progression cannot easily propel the robot. Corkscrewing is also valuable when maneuvering through a small hole in a wall or a chain link fence. This gait can be achieved by setting the values of variables in the above equations as follows:-

$A_{ver}=pi/6$, $A_{hor}=pi/3$, $\omega_{ver}=5pi/12$, $\omega_{hor}=5pi/6$,
$\delta_{ver}=2pi/3$, $\delta_{hor}=2pi/3$, $\delta_{o}=0$, $O_{ver}=0$, $O_{hor}=0$



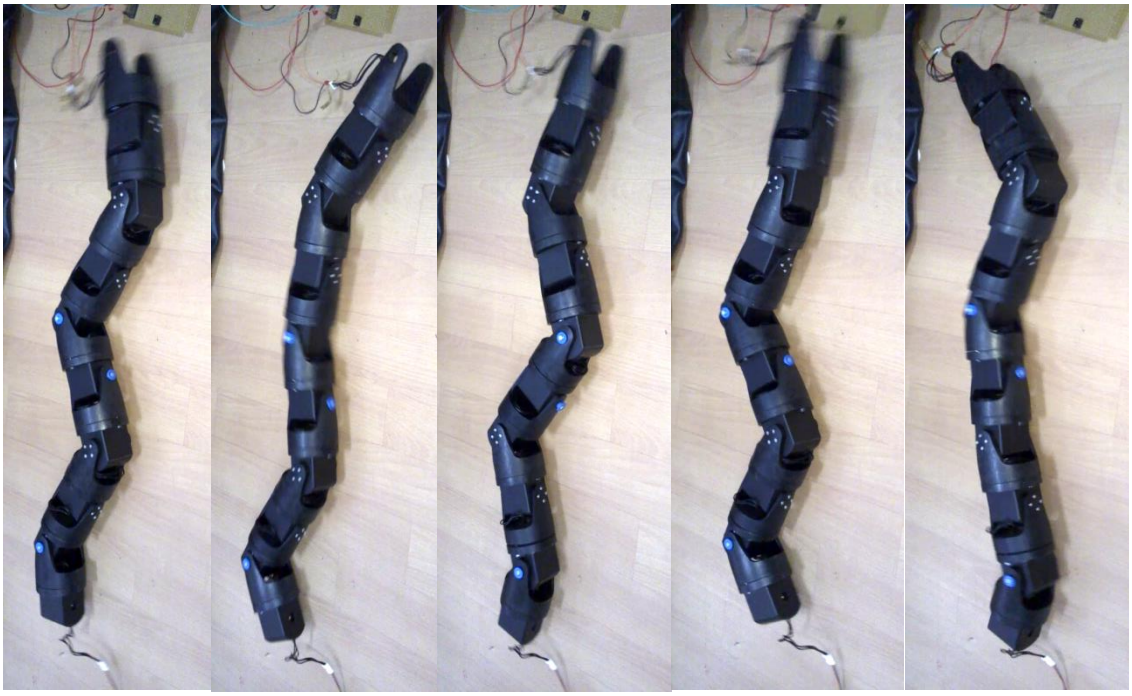
▪ LATERAL UNDULATION

Lateral undulation (also denoted serpentine crawling) is a continuous movement of the entire body of the snake relative to the ground. This locomotion is obtained by propagating waves from the front to the rear of the snake while exploiting roughness in the terrain. Every part of the body passes the same part of the ground ideally leaving a single sinus-like track. All the contact points with the ground constitute possible push-points for

the snake and the snake needs at least three push-points to obtain a continuous forward motion. Two points are needed to generate forces and the third point is used to balance the forces such that they act forward. This gait can be achieved by setting the values of variables in the above equations as follows:-

$A_{ver}=\pi/3$, $A_{hor}=0$, $\omega_{ver}=5\pi/6$, $\omega_{hor}=0$,

$\delta_{ver}=2\pi/3$, $\delta_{hor}=0$, $\delta_{o}=\pi/6$, $O_{ver}=0$, $O_{hor}=0$

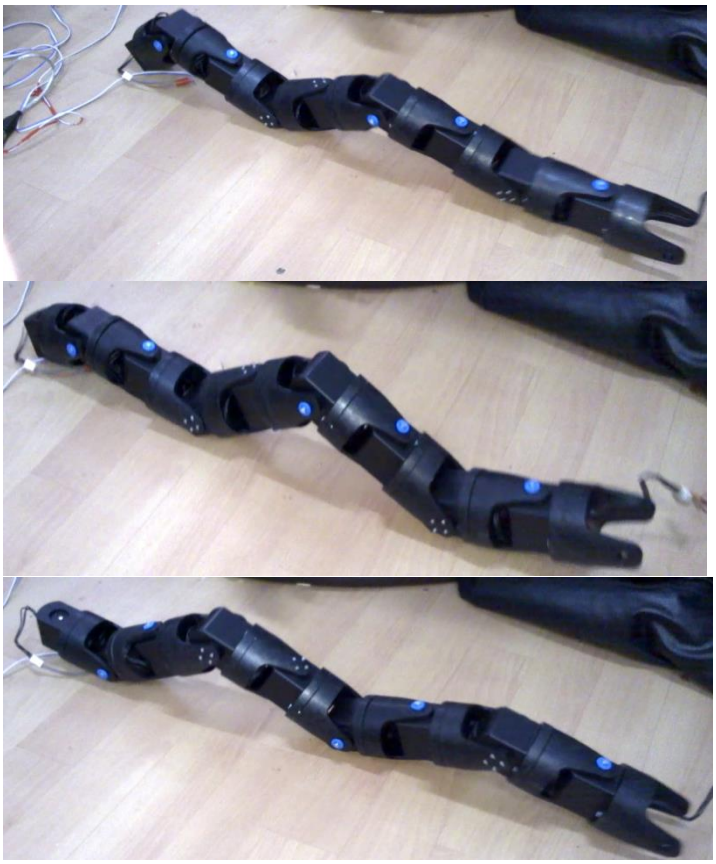


- ## CATERPILLER MOTION

Caterpillars send a vertical travelling wave through their body from the end to the front in order to move forward. Caterpillar motion is same as lateral undulation only the difference is sine wave propagates in horizontal plane in lateral undulation whereas it propagates in vertical plane in caterpillar motion. This gait can be achieved by setting the values of variables in the above equations as follows:-

$A_{ver}=0$, $A_{hor}=\pi/3$, $\omega_{ver}=0$, $\omega_{hor}=5\pi/6$,

$\delta_{ver}=0$, $\delta_{hor}=2\pi/3$, $\delta_{o}=\pi/6$, $O_{ver}=0$, $O_{hor}=0$



- ## CLIMBING

Pole climbing uses a different range of parameterizations of the helix gait which are more appropriate for climbing on the outsides of poles and pipe. The base shape of the robot is still a helix, but one in which the diameter is much wider and the pitch is less steep. The main constraints for climbing are weight of the robot, length of one link and no. of links. The diameter of the pole to be climbed should be chosen such that the snake surrounds the pole at least twice. This gait can be achieved by setting the values of variables in the above equations as follows:-

$A_{ver}=pi/2$, $A_{hor}=pi/2$, $\omega_{ver}=pi$, $\omega_{hor}=pi$,

$\delta_{ver}=0$, $\delta_{hor}=pi/2$, $\delta_{o}=0$, $O_{ver}=0$, $O_{hor}=0$

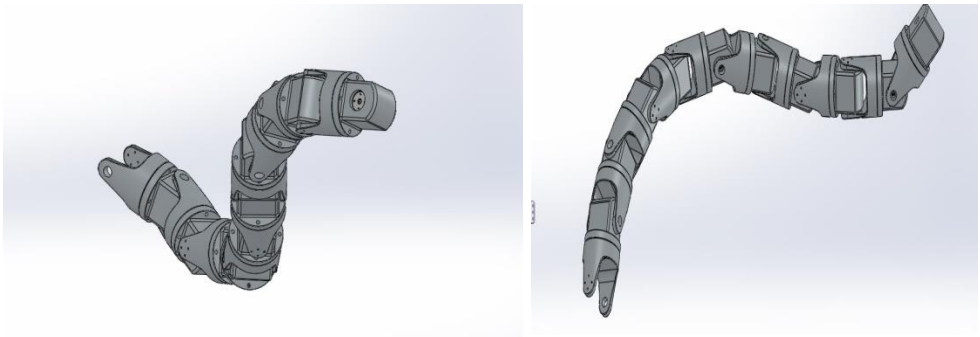
- ## MAMBA MOTION

  To make our snake a surveillance type robot we have attached a CMOS-Camera in the head of the snake, so that we can take feedback images from the camera. This will let us know where our robot exactly is. The Main problem we face was that the camera was not stable enough to catch stable video images. To overcome this problem we took inspiration from mamba snake who can move forward with ¾ of his body standing straight perpendicular to the ground. We tried to make the front part of the snake to stand straight in the air and move the rest of the body move such that the head remains stable and horizontal with respect to the ground. But in our robot our links having some significant weight the motor was in some constraint to lift fewer number of links. So hence we decided to simulate such a motion in which the fourth link will attain an angle of 90 degrees with the length of the snake and the third link will lift the first two links making it look as if it has lifted its head. The intermediate part of the snake will do a caterpillar motion such that the snake could make a forward motion. The last link will have a mean point of 90 degrees to make the snake balanced.

  Making the snake balanced while doing the caterpillar motion:
  To make the head stable the first link has a mean center at an angle of 80 degrees and giving it mere amplitude of 10 degrees making it synchronous with the wave motion of the fourth link. Note that the fourth link is a vertical link having a synchronous wave motion with the fourth link.

Similarly the last link that has an offset of 90 degrees also has amplitude of 10 degrees which is synchronous with the second last link.

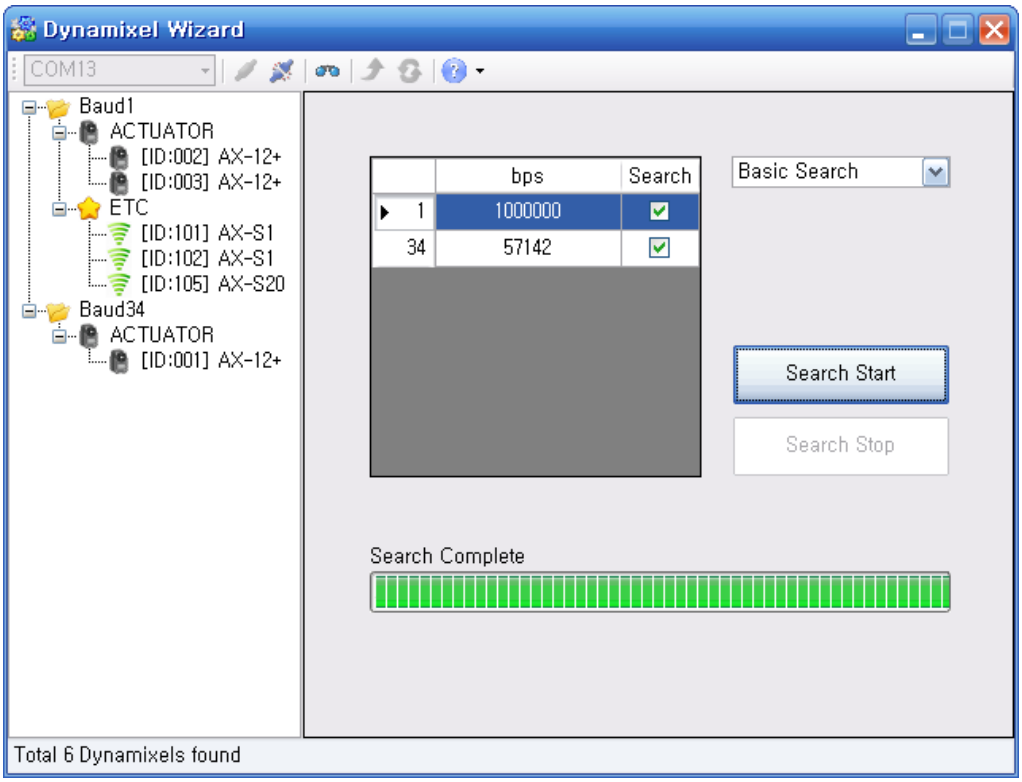
Sample simulation in solidworks for sidewinding gait.



Calibration and Repair of the dynamixels: The dynamixel AX12-A are quite fragile and have plastic gears and have a dc motor as the driving motor of the potentiometer.

1. The dc motors can get damaged due to overheating. This problem can be detected if the motor doesn't move even if the signal is given and the horn doesn't move even by the force given by hand. So what the user should do is to open the dynamixel case and replace the dc motor.
2. Another common problem faced is that the gears break and the motor inside rotates infinitely. Such problem could be solved if one opens the casing and replaces the broken gear.

3. An unusual error that can occur is that the dynamixel do not show any id and do not get recognized. Even if someone searches for that motor by pinging through ros, the dynamixel doesn't get recognized. So the only solution is to flash the memory back to the atmega8-l IC of the dynamixel circuit using the roboplus software of the robotis. This is done by attaching the single dynamixel to the usb2dynamixel and using the dynamixel firmware recovery wizard of the roboplus.



CONCLUSION

All the primary GAITS were implemented on the snake robot. The snake was robust and could perform in rocky terrains. It had the capability of using the friction of rough surfaces to move forward. We were able to control the robot with a GUI.

FUTURE PROSPECTS:

- The snake could climb trees and structures with variable radius. This could be achieved by making use of the feedback node.
- Other activities such as swimming and propulsion under deep water could be done using efficient gaits could be achieved.
- The snake can be made autonomous by making use of sensors, cameras and making it intelligent enough to attain gaits according to the terrain autonomously.

- Another very important future aspect could include designing own motors along with the mini controller making it more suitable to your own application.

REFERENCES

- S. Hirose, Biologically Inspired Robots (Snake-like Locomotor and Manipulator). Oxford University Press, 1993.
- http://wiki.ros.org/
- http://inobotics.blogspot.in/
- http://www.engineersgarage.com/
- http://answers.ros.org/question/82146/how-to-create-a-ros-dynamixel-controller/
- http://biorobotics.ri.cmu.edu/projects/modsnake/