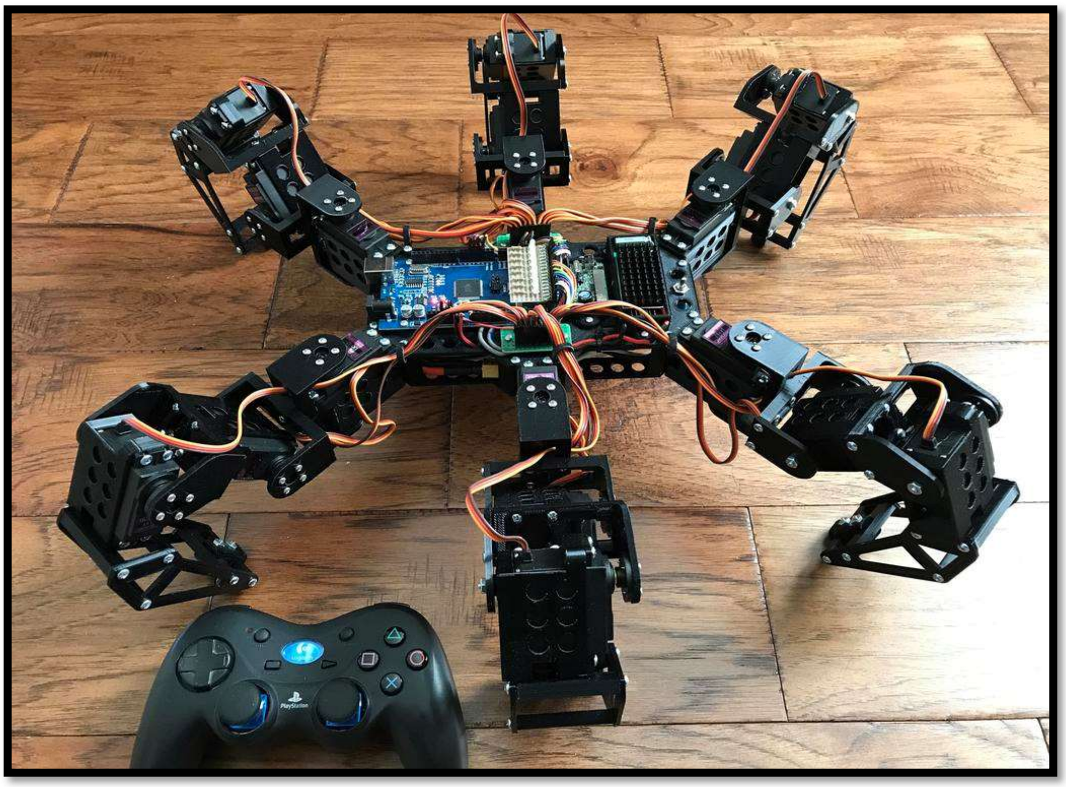
PS2 Controlled Hexapod Robot

Team Pheonix,Department of Automation and Robotics, Amrutvahini College of Engineering,Sangamner,Ahamednagar.

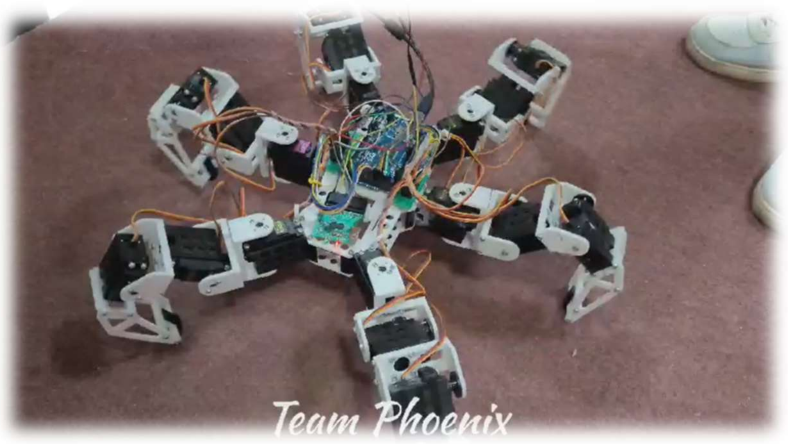
Email:-Kawadeganesh81@gmail.com ,punamkhilari4@gmail.com ,pragati7219@gmail.com



Abstract

This project outlines aspects of building a six legged walking robot that is capable of basic mobility tasks such as walking forward, backward, rotating in place. The legs will be of a modular design and will have one degree of freedom each. This robot will serve as a platform onto which additional sensory components could be added, or which could be programmed to perform increasingly complex motions. In this design we used three stepper motors for motion and one PIC 16F877A microcontroller. The microcontroller stores the program for walking, controls the three stepper motors, and reads the two sensor-switches in front. The walking program contains subroutines for walking forward and backward, turning right, and turning left. The two switch-sensors positioned in the front of the walker inform the microcontroller of any obstacles in the walker’s path. Based on the feedback from these switch-sensors, the walker will turn or reverse to avoid obstacles placed in its path. This report discusses the components that make up our final design. The

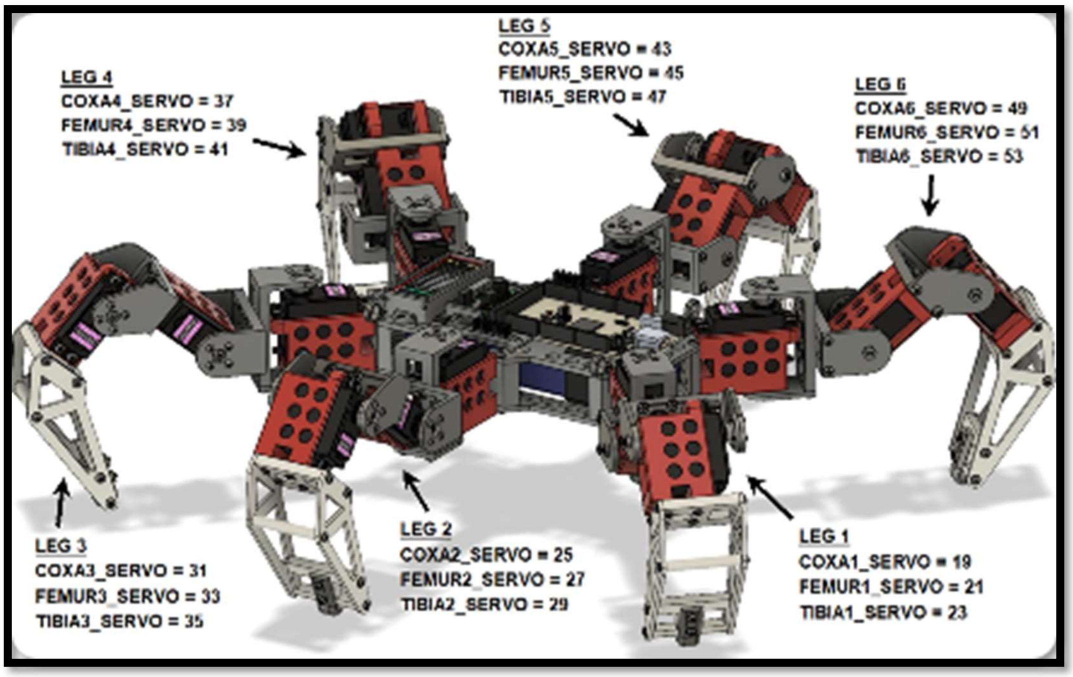
designed six legged-tripod gait able to perform various industrial tasks as the program is changeable as per requirement



Now a- days, robots are made for just walking on flat surface and are unable to walk on sand, hills & rough surfaces but this type of problem can be solved by using new technology of robots called Hexapods. A hexapod is capable of fulfilling all these needs & also has high accuracy in its motion because of the use of servo motors rather than DC motors. A hexapod has multiple legs which makes it an ideal choice not only for climbing purpose but also as dancing robot. This paper proposes the concept of mechanical/structural design for a hexapod walking robot. The hexapod walking robot design is inspired by the insects around us. The main feature of the insects is their stability and their style of movement. Taking these factors into consideration, a novel design is proposed in the form of "Hexapod" that walks on 6 legs & move in at least 3 directions. The hexapod prototype can thus be utilized in applications such as defense sector as a spy bot, for health monitoring of patients in remote areas, etc.

Introduction

“Robots will play an important role in providing physical assistance and even companionship for the elderly.”
-Bill Gates

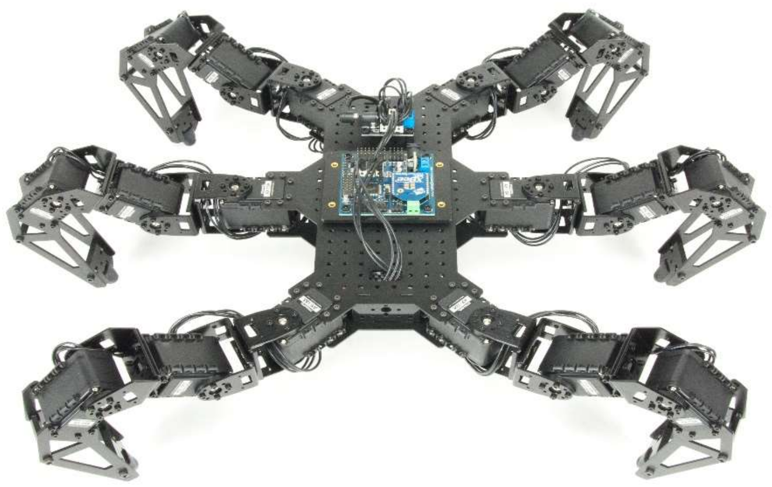


The engineers of today always strive to build new gadgets that can address the different problems faced by humans in their day to day life. Keeping this perspective in mind, the need of advanced robots, capable of reaching places that humans cannot, has emerged. The biomimetric bots serve the same purpose. Insects are often used for inspiration while designing walking robots because of their ability to easily navigate through all type of surfaces. The insects have a symmetric structure, low weight, small size and most important: their joint legs which are at least 6 in number. Due to such a body structure, they are able not only to move freely around but also survive in the changing environment conditions.There are many types of robots like industrial, agricultural, tele-robots, service robots, mobile robots, Hexapods etc. The Hexapod robots, have the ability to move in an environment unlike stationary industrial robots. And they can be classified by their travel environment; Water, air, ground, ice or space according to the application and control by the device they use to move; Legs, wheels or tracks. Thus, through an extensive and exhaustive research keeping all these characteristics of the insects at hand, it then became possible to come up with the Hexapod Walking robot. The hexapod which is an insect inspired design, is highly stable in its movements because its body is supported by 6 legs with as many as up to 12 servomotors fitted in the joints and the interconnects between the main body & legs. The robot thus designed showcases more flexible movements on all kinds of surfaces & does not require any other additional balance mechanism. It however requires a positive feedback to ensure more precise movements. This type of robot finds applications in real life, such as search and rescue applications, environment exploration, and also as a CNC machine.

Robots can be found everywhere. One of the most important part of a robot is its chassis. There are several basic chassis types: wheeled, tracked and legged. Wheeled are fast, but not suitable for rough terrain. Tracked are slower, but more suitable to rugged terrain. Legged are quite slow and more difficult to control, but extremely robust in rough terrain. Legged are capable to cross large holes and can operate even after losing a leg.Many researches were performed in this field in past few years, because of its large potential. Legged are especially ideal for space missions.There are also several projects in military research

"A major goal in this field is in developing capabilities for robots to autonomously decide how, when and where to move."

Project Motivation



This is my hexapod robot inspired by the Phantom X AX Trossen Robotics. The Phantom X costs $1300, which is way too expensive for my taste, so I decided to design a robot using cheap 180 degree MG996R 11kg-cm servos. Before starting, I added up the approximate weights of the servos, 3D printed parts, hardware, and electronics and computed the worst-case required torques for the various leg joints, which I found were within the capacity of these servos.

Problem Statement

Hexapod walking robots have been one of the robots that has changed the pace in technology through several years. Many studies have been carried out in the prospect of their development in research centers, universities. However, only in the recent past have efficient walking machines been conceived, designed and built with performances that can be suitable for practical applications. This project gives an overview of the state of the art on hexapod walking robots and its limitations. Careful attention is given to the main design issues and constraints that influence the technical feasibility and operation performance. A design procedure is outlined in order to systematically design a hexa- pod walking robot. In particular, the proposed design procedure takes into account the main features, such as mechanical structure and leg configuration, actuating and driving systems, payload, motion conditions, and walking gaints in general legged robots 12 servos, 2 in each leg cannot stabilize the hexapod and efficiently is largely decreased. Hence 18 servos are used.

As the number of servomotors used for the hexapod are 18, and due to the lack of 18 PWM pins in the arduino and external source or a method must be required to run all the 18 servo.

Objective:

The main objective of this project can be stated as follows:-

1) To study the movement and dynamics of the Hexapod robot.
2) Designing the model of Hexapod robot on CAD software Solid Works.
3) To design the Hexapod basing on the market needs and making it available for selling in the market.
4) For modifying the design based on requirements.
5) Analysis and simulation of the hexapod.
6) Fabrication of hexapod.
7) Automation and controlling.
8) Testing

Methodology

3.1 Required tools for the project:-

1) Hardware Requirements:-

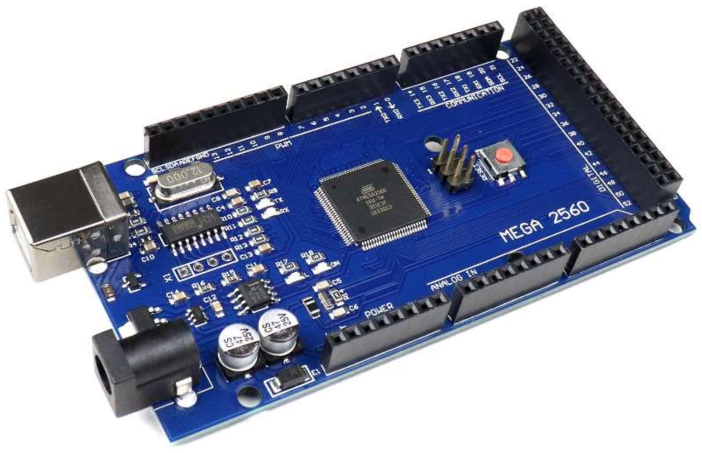
    a) Arduino Mega2560 R3
    b) Servo Motor
    c) 3D printed Parts
    d) PS2 Controller
    e) 5500 MAH Lipo Battery
    f) Jumping Wires
    g) 12-5v Buck Convertor
    h) 624Z bearing
    i) Screws

2) Software Requirements-:

    a) Ardino Software
    b) Solid Works

❖ Tools

1) Arduino Mega2560 R3:-



Arduino has become very popular in the world in recent times. In this project the arduino mega is used to communicate with processing and control the arm robot servos.

The causes of the spread of Arduino at such a rapid rate are:

1) it can be used on all platforms due to the simplicity of the development.

2) With the help of the advanced library, even complex operations can be easily solved.

3) There is a lot of hardware support that is compatible with Arduino and can work together.

4) Communication with the environment is easy because it is open source.

Microcontroller	**ATmega328P – 8 bit AVR family microcontroller**
Operating Voltage	**5V**
Recommended Input Voltage	**7-12V**
Input Voltage Limits	**6-20V**
Analog Input Pins	**6 (A0 – A5)**
Digital I/O Pins	**14 (Out of which 6 provide PWM output)**
DC Current on I/O Pins	**40 mA**
DC Current on 3.3V Pin	**50 mA**
Flash Memory	**32 KB (0.5 KB is used for Bootloader)**
SRAM	**2 KB**
EEPROM	**1 KB**
Frequency (Clock Speed)	**16 MHz**

2) MG996 Servo Motor:-

Servo motors are the kinds of motors that can fulfill the commands we want. They can operate steadily even at very small or very large speeds. In these motors, the large moment can be obtained from the small size.

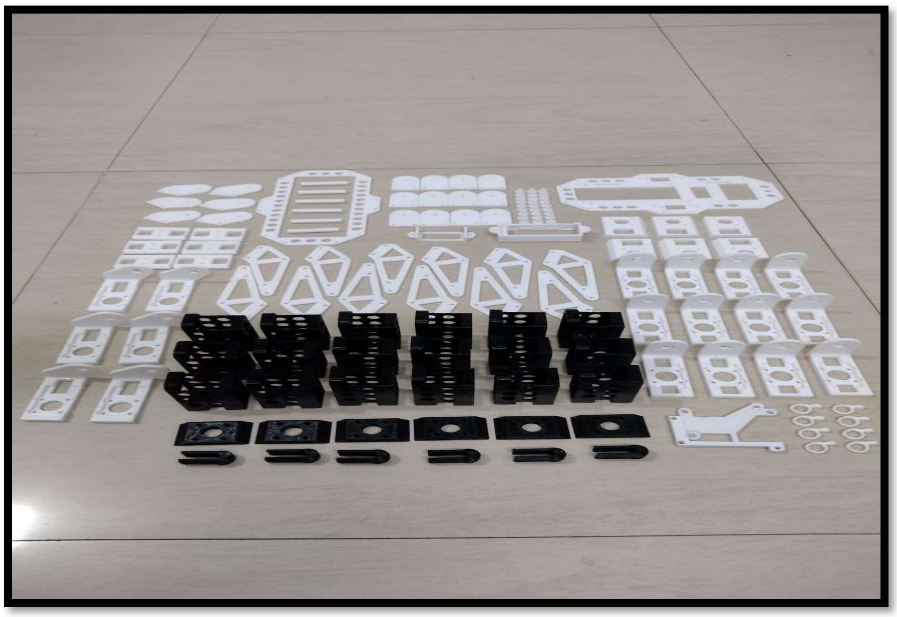
Primarily, servomotors are geared dc motors with a positional feedback control that allows the rotor (shaft) to be positioned accurately. The position control signal is a single variable width pulse. The pulse can be varied from 1 to 2 ms.

Operating Voltage:	**+5V typically**
Current:	**2.5A (6V)**
Stall Torque:	**9.4 kg/cm (at 4.8V)**
Maximum Stall Torque:	**I I kg/cm (6V)**
Operating speed:	**0.17**

Rotation :	**360**
Weight Of Motor:	**55gm**

3) 3D Printed Parts:-

We designed the robot in Solid Works and PTC Creo6.0 Software and printed the parts on Creality CR 10S 3D printer. The parts were designed with low weight in mind (thin walls, lightening holes, etc.) to maximize the chances of this working with the servos I had chosen. The Black PLA parts were printed a bit on the hot side (210° C) for better layer-to-layer adhesion and part strength. All parts were printed with a 0.4mm nozzle at 0.2mm layer height.



4) PS2 Wireless controller with Reciever:=

The PS2 wireless controller is a standard controller for the PlayStation 2 and is identical to the original DualShock controller for the PlayStation console. It features twelve analog (pressure-sensitive) buttons (X, O, Π, Δ, L1, R1, L2, R2, Up, Down, Left and Right), five digital button (L3, R3 Start, Select and the analog mode button) and two analog sticks. The controller also features two vibration motors, the left one being larger and more powerful than the one on the right. It is powered by two AAA batteries. It communicates with the console using 2.4 GHz RF protocol.

5) 5500 MAH Lithium-polymer Battery:-

ROARINGTOP
HIGH DISCHARGE LI-PO BATTERY
5500mAh
70C
PLEASE READ SAFETY INSTRUCTIONS PRIOR TO USE

Lithium polymer battery Pack (LiPo) batteries are known for performance and reliability. Lithium polymer battery Pack (LiPo) batteries deliver the full rated capacity at a price everyone can afford. They are equipped with heavy-duty discharge leads to minimize resistance and sustain high current loads and stand up to the punishing extremes of aerobatic flight and RC vehicles.

6) 624Z Bearings:-

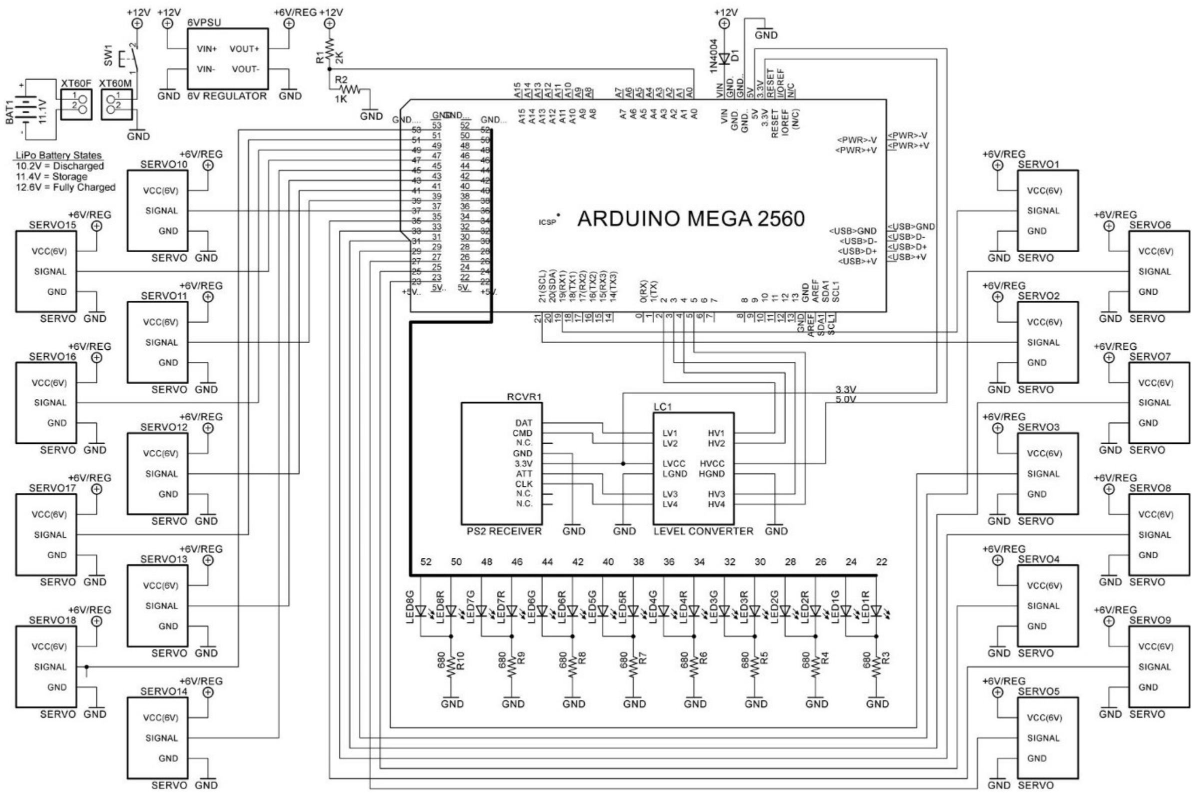
This radial ball bearing 624zz for 3d printer,robot is a 4x13x5 mm double shielded ball bearing with 624zz abec-3 quality, 624zz bearings are the popular item that could be used in many application that uses this size 4x13x5 mm bearings, each 624zz bearings are closed with 2 metal shields, one shield from each side, to protect the bearing from dust or any possible contamination.

bearings come in various qualities. We have carefully selected the best quality bearings from numerous suppliers. This means our bearings are meant for continuous use and they'll last longer than other cheap bearings. Ideal for use in your 3d printer, cnc project or robot.

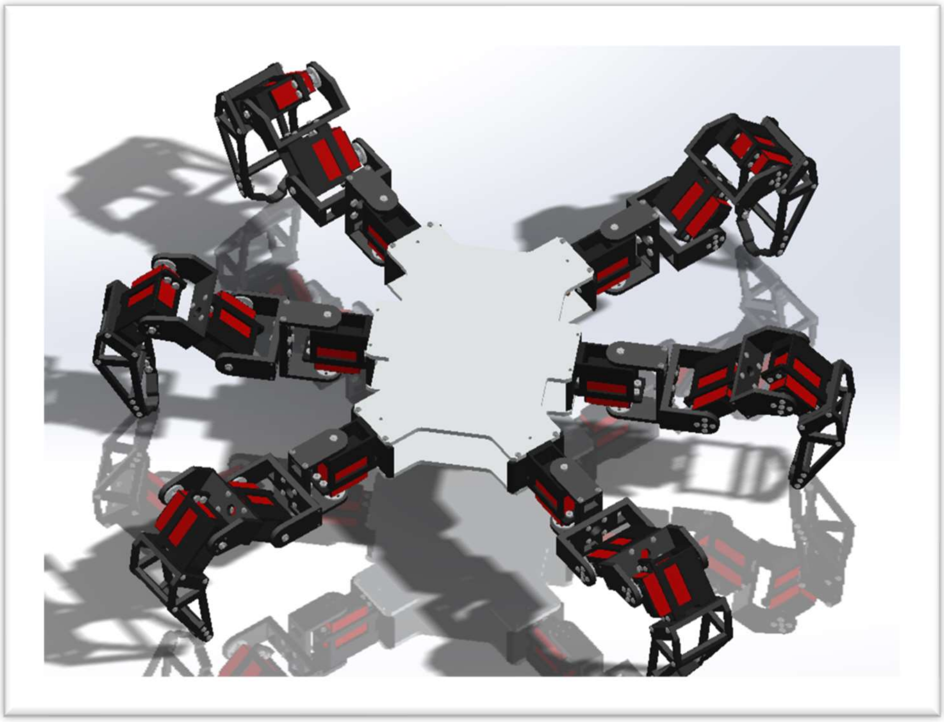
7) Jumping Wires:-

A jumper wire (also known as jumper, jumper wire, DuPont wire) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

1.2 Design of Project:-

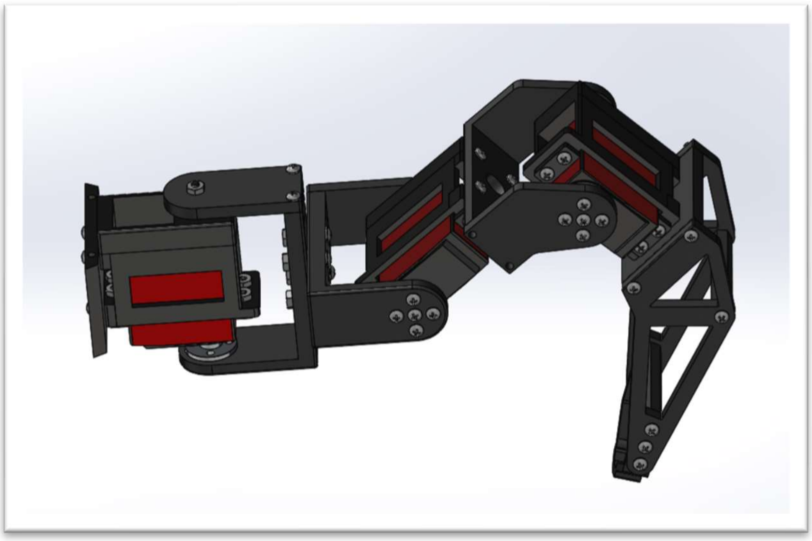


A) Body:-



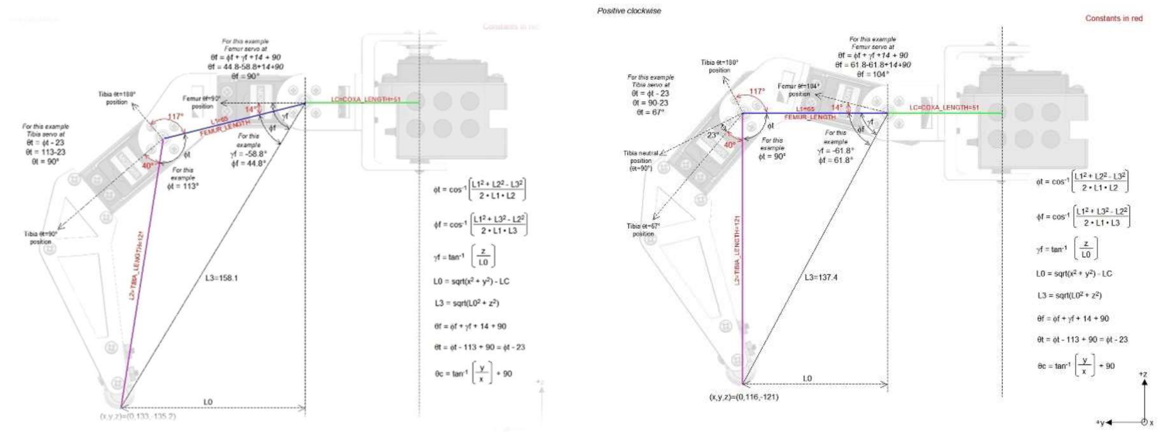
The main body constitutes of light-weight composite materials. It makes the whole project cost effective. To keep the center of gravity at the mid-point of the body, the legs are fixed at the opposite ends so that the robot can move continuously and balance it-self. The arduino is fixed in the back portion of the body and the servo-controller is kept in the front portion of the body.

B) Legs:-



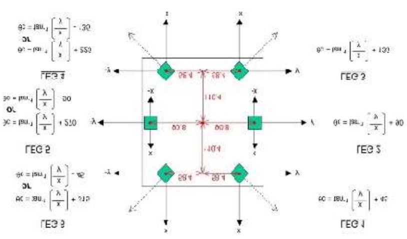
Each leg of the robot has the same functions and six degrees of freedom. The only difference between the legs is the motion algorithm. The legs are designed in such a way that it can walk on irregular platforms and can climb terrains. The movement access of one of its legs,

1.3 Mechanism:-



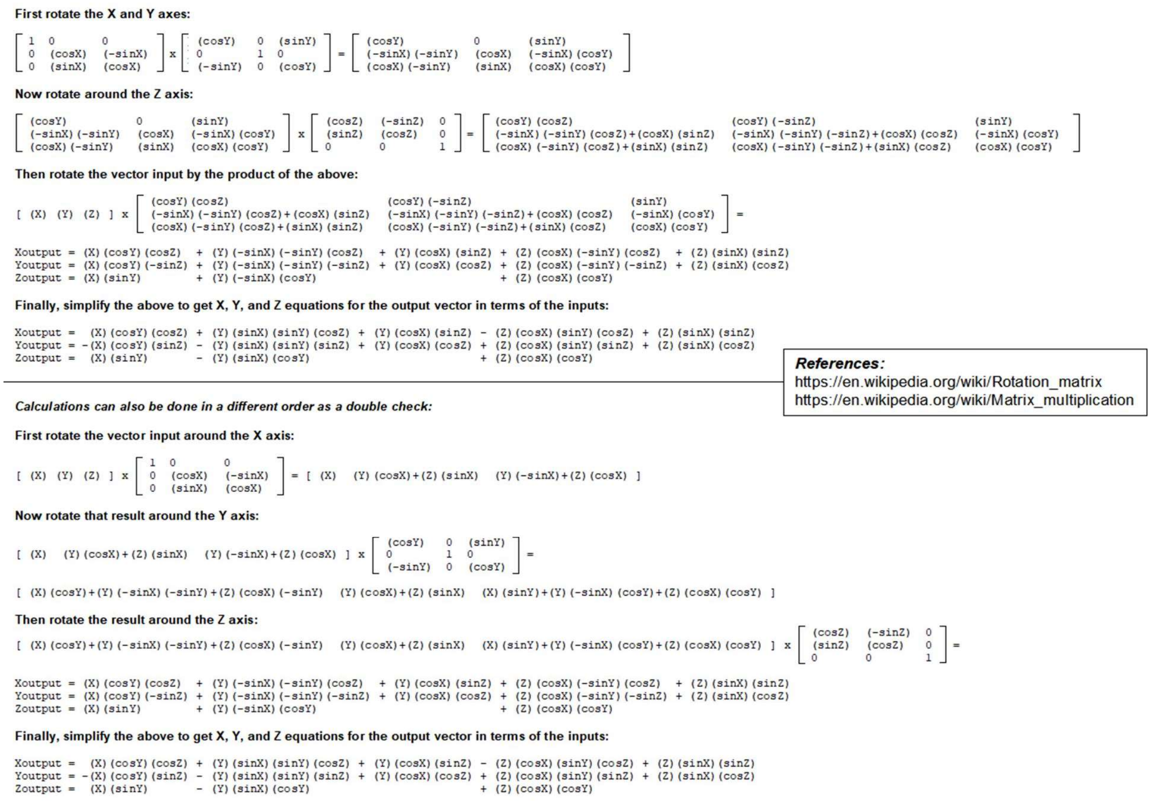
The robot works using inverse kinematic (IK) calculations:-

the servo positions are computed based on the desired x,y,z coordinates of the tip of each foot. Knowing a few constants – the tibia, femur, and coxa lengths and the offset angles between the axis of the servos and the structure permits the desired servo angles for all possible x,y,z coordinates to be calculated. Each leg is mounted at a different angle and the offsets of each leg from the center of the body is different, so the computed coxa angle varies for each leg.



When walking, the robot uses the remote joystick inputs – x and y from the right joystick and rotation from the left joystick – to tell the body which way to move. The gait engines compute the individual foot tip positions required to accomplish the desired movement. Then the IK calculation is applied to each leg’s foot tip coordinate to tell the individual servos where to go to position the legs as requested by the gait engine.

There are also modes programmed to stand in-place and translate or rotate the body. The translation is just a simple addition of an x, y, or z offset to the coordinate system. Body rotation is more involved, but is derived as shown in below where the x, y, and z offsets are computed using matrix multiplication.



First rotate the X and Y axes:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & (\cos X) & (-\sin X) \\ 0 & (\sin X) & (\cos X) \end{bmatrix} x \begin{bmatrix} (\cos Y) & 0 & (\sin Y) \\ 0 & 1 & 0 \\ (-\sin Y) & 0 & (\cos Y) \end{bmatrix} = \begin{bmatrix} (\cos Y) & 0 & (\sin Y) \\ (-\sin X)(-\sin Y) & (\cos X) & (-\sin X)(\cos Y) \\ (\cos X)(-\sin Y) & (\sin X) & (\cos X)(\cos Y) \end{bmatrix}$$

Now rotate around the Z axis:

$$\begin{bmatrix} (\cos Y) & 0 & (\sin Y) \\ (-\sin X)(-\sin Y) & (\cos X) & (-\sin X)(\cos Y) \\ (\cos X)(-\sin Y) & (\sin X) & (\cos X)(\cos Y) \end{bmatrix} x \begin{bmatrix} (\cos Z) & (-\sin Z) & 0 \\ (\sin Z) & (\cos Z) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} (\cos Y)(\cos Z) & (\cos Y)(-\sin Z) & (\sin Y) \\ (-\sin X)(-\sin Y)(\cos Z)+(\cos X)(\sin Z) & (-\sin X)(-\sin Y)(-\sin Z)+(\cos X)(\cos Z) & (-\sin X)(\cos Y) \\ (\cos X)(-\sin Y)(\cos Z)+(\sin X)(\sin Z) & (\cos X)(-\sin Y)(-\sin Z)+(\sin X)(\cos Z) & (\cos X)(\cos Y) \end{bmatrix}$$

Then rotate the vector input by the product of the above:

$$[\ (X)\ (Y)\ (Z)\ ]\ x \begin{bmatrix} (\cos Y)(\cos Z) & (\cos Y)(-\sin Z) & (\sin Y) \\ (-\sin X)(-\sin Y)(\cos Z)+(\cos X)(\sin Z) & (-\sin X)(-\sin Y)(-\sin Z)+(\cos X)(\cos Z) & (-\sin X)(\cos Y) \\ (\cos X)(-\sin Y)(\cos Z)+(\sin X)(\sin Z) & (\cos X)(-\sin Y)(-\sin Z)+(\sin X)(\cos Z) & (\cos X)(\cos Y) \end{bmatrix} =$$

$$\begin{aligned} Xoutput &= (X)(\cos Y)(\cos Z) + (Y)(-\sin X)(-\sin Y)(\cos Z) + (Y)(\cos X)(\sin Z) + (Z)(\cos X)(-\sin Y)(\cos Z) + (Z)(\sin X)(\sin Z) \\ Youtput &= (X)(\cos Y)(-\sin Z) + (Y)(-\sin X)(-\sin Y)(-\sin Z) + (Y)(\cos X)(\cos Z) + (Z)(\cos X)(-\sin Y)(-\sin Z) + (Z)(\sin X)(\cos Z) \\ Zoutput &= (X)(\sin Y) + (Y)(-\sin X)(\cos Y) + (Z)(\cos X)(\cos Y) \end{aligned}$$

Finally, simplify the above to get X, Y, and Z equations for the output vector in terms of the inputs:

$$\begin{aligned} Xoutput &= (X)(\cos Y)(\cos Z) + (Y)(\sin X)(\sin Y)(\cos Z) + (Y)(\cos X)(\sin Z) - (Z)(\cos X)(\sin Y)(\cos Z) + (Z)(\sin X)(\sin Z) \\ Youtput &= -(X)(\cos Y)(\sin Z) - (Y)(\sin X)(\sin Y)(\sin Z) + (Y)(\cos X)(\cos Z) + (Z)(\cos X)(\sin Y)(\sin Z) + (Z)(\sin X)(\cos Z) \\ Zoutput &= (X)(\sin Y) - (Y)(\sin X)(\cos Y) + (Z)(\cos X)(\cos Y) \end{aligned}$$

References:
https://en.wikipedia.org/wiki/Rotation_matrix
https://en.wikipedia.org/wiki/Matrix_multiplication

Calculations can also be done in a different order as a double check:

First rotate the vector input around the X axis:

$$[\ (X)\ (Y)\ (Z)\ ]\ x \begin{bmatrix} 1 & 0 & 0 \\ 0 & (\cos X) & (-\sin X) \\ 0 & (\sin X) & (\cos X) \end{bmatrix} = [\ (X)\quad (Y)(\cos X)+(Z)(\sin X)\quad (Y)(-\sin X)+(Z)(\cos X)\ ]$$

Now rotate that result around the Y axis:

$$[\ (X)\quad (Y)(\cos X)+(Z)(\sin X)\quad (Y)(-\sin X)+(Z)(\cos X)\ ]\ x \begin{bmatrix} (\cos Y) & 0 & (\sin Y) \\ 0 & 1 & 0 \\ (-\sin Y) & 0 & (\cos Y) \end{bmatrix} =$$

$$[\ (X)(\cos Y)+(Y)(-\sin X)(-\sin Y)+(Z)(\cos X)(-\sin Y)\quad (Y)(\cos X)+(Z)(\sin X)\quad (X)(\sin Y)+(Y)(-\sin X)(\cos Y)+(Z)(\cos X)(\cos Y)\ ]$$

Then rotate the result around the Z axis:

$$[\ (X)(\cos Y)+(Y)(-\sin X)(-\sin Y)+(Z)(\cos X)(-\sin Y)\quad (Y)(\cos X)+(Z)(\sin X)\quad (X)(\sin Y)+(Y)(-\sin X)(\cos Y)+(Z)(\cos X)(\cos Y)\ ]\ x \begin{bmatrix} (\cos Z) & (-\sin Z) & 0 \\ (\sin Z) & (\cos Z) & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{aligned} Xoutput &= (X)(\cos Y)(\cos Z) + (Y)(-\sin X)(-\sin Y)(\cos Z) + (Y)(\cos X)(\sin Z) + (Z)(\cos X)(-\sin Y)(\cos Z) + (Z)(\sin X)(\sin Z) \\ Youtput &= (X)(\cos Y)(-\sin Z) + (Y)(-\sin X)(-\sin Y)(-\sin Z) + (Y)(\cos X)(\cos Z) + (Z)(\cos X)(-\sin Y)(-\sin Z) + (Z)(\sin X)(\cos Z) \\ Zoutput &= (X)(\sin Y) + (Y)(-\sin X)(\cos Y) + (Z)(\cos X)(\cos Y) \end{aligned}$$

Finally, simplify the above to get X, Y, and Z equations for the output vector in terms of the inputs:

$$\begin{aligned} Xoutput &= (X)(\cos Y)(\cos Z) + (Y)(\sin X)(\sin Y)(\cos Z) + (Y)(\cos X)(\sin Z) - (Z)(\cos X)(\sin Y)(\cos Z) + (Z)(\sin X)(\sin Z) \\ Youtput &= -(X)(\cos Y)(\sin Z) - (Y)(\sin X)(\sin Y)(\sin Z) + (Y)(\cos X)(\cos Z) + (Z)(\cos X)(\sin Y)(\sin Z) + (Z)(\sin X)(\cos Z) \\ Zoutput &= (X)(\sin Y) - (Y)(\sin X)(\cos Y) + (Z)(\cos X)(\cos Y) \end{aligned}$$

1.4 Program Interface:-

Arduino Code to run the Robotic Mechanism Using PS2 Controller:-

```
#include <PS2X_lib.h>
#include <Servo.h>
#include <math.h>



const int BATT_VOLTAGE = 0;           //12V Battery analog voltage input port

const int PS2_DAT = 2;                //gamepad port definitions
const int PS2_ATT = 3;
const int PS2_CMD = 4;
const int PS2_CLK = 5;
const int RUMBLE = true;
const int PRESSURES = false;

const int COXA1_SERVO  = 19;          //servo port definitions
const int FEMUR1_SERVO = 21;
const int TIBIA1_SERVO = 23;
const int COXA2_SERVO  = 25;
const int FEMUR2_SERVO = 27;
const int TIBIA2_SERVO = 29;
const int COXA3_SERVO  = 31;
const int FEMUR3_SERVO = 33;
const int TIBIA3_SERVO = 35;
const int COXA4_SERVO  = 37;
const int FEMUR4_SERVO = 39;
const int TIBIA4_SERVO = 41;
const int COXA5_SERVO  = 43;
const int FEMUR5_SERVO = 45;
const int TIBIA5_SERVO = 47;
const int COXA6_SERVO  = 49;
const int FEMUR6_SERVO = 51;
const int TIBIA6_SERVO = 53;
const int COXA_LENGTH = 51;
const int FEMUR_LENGTH = 65;
const int TIBIA_LENGTH = 121;

const int TRAVEL = 30;
const long A12DEG = 209440;
const long A30DEG = 523599;

const int FRAME_TIME_MS = 20;
const float HOME_X[6] = {  82.0,   0.0, -82.0,  -82.0,    0.0,  82.0};
const float HOME_Y[6] = {  82.0, 116.0,  82.0,  -82.0, -116.0, -82.0};
const float HOME_Z[6] = { -80.0, -80.0, -80.0,  -80.0,  -80.0, -80.0};

const float BODY_X[6] = { 110.4,  0.0, -110.4, -110.4,    0.0, 110.4};
const float BODY_Y[6] = {  58.4, 90.8,   58.4,  -58.4,  -90.8, -58.4};
const float BODY_Z[6] = {   0.0,  0.0,    0.0,    0.0,    0.0,   0.0};

const int COXA_CAL[6]  = {2, -1, -1, -3, -2, -3};
const int FEMUR_CAL[6] = {4, -2,  0, -1,  0,  0};
const int TIBIA_CAL[6] = {0, -3, -3, -2, -3, -1};

int gamepad_error;
byte gamepad_vibrate;
unsigned long currentTime;
unsigned long previousTime;

int temp;
int mode;
int gait;
int gait_speed;
int gait_LED_color;
int reset_position;
int capture_offsets;
int batt_LEDs;
int batt_voltage;
int batt_voltage_index;
int batt_voltage_array[50];
long batt_voltage_sum;

float L0, L3;
float gamma_femur;
float phi_tibia, phi_femur;
float theta_tibia, theta_femur, theta_coxa;

int leg1_IK_control, leg6_IK_control;
float leg1_coxa, leg1_femur, leg1_tibia;
float leg6_coxa, leg6_femur, leg6_tibia;

int leg_num;
int z_height_LED_color;
int totalX, totalY, totalZ;
int tick, duration, numTicks;
int z_height_left, z_height_right;
int commandedX, commandedY, commandedR;
int translateX, translateY, translateZ;
```

```
float step_height_multiplier;
float strideX, strideY, strideR;
float sinRotX, sinRotY, sinRotZ;
float cosRotX, cosRotY, cosRotZ;
float rotOffsetX, rotOffsetY, rotOffsetZ;
float amplitudeX, amplitudeY, amplitudeZ;
float offset_X[6], offset_Y[6], offset_Z[6];
float current_X[6], current_Y[6], current_Z[6];

int tripod_case[6]   = {1,2,1,2,1,2};
int ripple_case[6]   = {2,6,4,1,3,5};
int wave_case[6]     = {1,2,3,4,5,6};
int tetrapod_case[6] = {1,3,2,1,2,3};

PS2X ps2x;
Servo coxa1_servo;      //18 servos
Servo femur1_servo;
Servo tibia1_servo;
Servo coxa2_servo;
Servo femur2_servo;
Servo tibia2_servo;
Servo coxa3_servo;
Servo femur3_servo;
Servo tibia3_servo;
Servo coxa4_servo;
Servo femur4_servo;
Servo tibia4_servo;
Servo coxa5_servo;
Servo femur5_servo;
Servo tibia5_servo;
Servo coxa6_servo;
Servo femur6_servo;
Servo tibia6_servo;

void setup()
{
  Serial.begin(115200);

  coxa1_servo.attach(COXA1_SERVO,610,2400);
  femur1_servo.attach(FEMUR1_SERVO,610,2400);
  tibia1_servo.attach(TIBIA1_SERVO,610,2400);
  coxa2_servo.attach(COXA2_SERVO,610,2400);
  femur2_servo.attach(FEMUR2_SERVO,610,2400);
  tibia2_servo.attach(TIBIA2_SERVO,610,2400);
  coxa3_servo.attach(COXA3_SERVO,610,2400);
  femur3_servo.attach(FEMUR3_SERVO,610,2400);
  tibia3_servo.attach(TIBIA3_SERVO,610,2400);
  coxa4_servo.attach(COXA4_SERVO,610,2400);
  femur4_servo.attach(FEMUR4_SERVO,610,2400);
  tibia4_servo.attach(TIBIA4_SERVO,610,2400);
  coxa5_servo.attach(COXA5_SERVO,610,2400);
  femur5_servo.attach(FEMUR5_SERVO,610,2400);
  tibia5_servo.attach(TIBIA5_SERVO,610,2400);
  coxa6_servo.attach(COXA6_SERVO,610,2400);
  femur6_servo.attach(FEMUR6_SERVO,610,2400);
  tibia6_servo.attach(TIBIA6_SERVO,610,2400);

  //connect the gamepad
  gamepad_error = ps2x.config_gamepad(PS2_CLK, PS2_CMD, PS2_ATT, PS2_DAT,
PRESSURES, RUMBLE);
  if(gamepad_error == 0)      Serial.println("Controller attached");
  else if(gamepad_error == 1) Serial.println("No controller found");
  else if(gamepad_error == 2) Serial.println("Controller found but not
accepting commands");
  else if(gamepad_error == 3) Serial.println("Controller refusing to enter
Pressures mode");

  //verify the gamepad type
  gamepad_type = ps2x.readType();
  if(gamepad_type == 0)      Serial.println("Unknown Controller type found");
  else if(gamepad_type == 1) Serial.println("DualShock Controller found");
  else if(gamepad_type == 2) Serial.println("GuitarHero Controller found");
  else if(gamepad_type == 3) Serial.println("Wireless Sony DualShock
Controller found");

  gamepad_vibrate = 0;

  for(int i=0; i<8; i++)
  {
    pinMode((RED_LED1+(4*i)),OUTPUT);
    pinMode((GREEN_LED1+(4*i)),OUTPUT);
  }

  for(batt_voltage_index=0; batt_voltage_index<50; batt_voltage_index++)
    batt_voltage_array[batt_voltage_index] = 0;
  batt_voltage_sum = 0;
  batt_voltage_index = 0;

  for(leg_num=0; leg_num<6; leg_num++)
  {
    offset_X[leg_num] = 0.0;
    offset_Y[leg_num] = 0.0;
    offset_Z[leg_num] = 0.0;
  }
  capture_offsets = false;
  step_height_multiplier = 1.0;

  mode = 0;
  gait = 0;
  gait_speed = 0;
  reset_position = true;
  leg1_IK_control = true;
  leg6_IK_control = true;
}

void loop()
{
  if((gamepad_error == 1) || (gamepad_type == 2))
  {
    Serial.println("Invalid Controller!");
    return;
  }

  currentTime = millis();
  if((currentTime - previousTime) > FRAME_TIME_MS)
  {
    previousTime = currentTime;

    ps2x.read_gamepad(false, gamepad_vibrate);
    process_gamepad();
    if(reset_position == true)
    {
      for(leg_num=0; leg_num<6; leg_num++)
      {
        current_X[leg_num] = HOME_X[leg_num];
        current_Y[leg_num] = HOME_Y[leg_num];
        current_Z[leg_num] = HOME_Z[leg_num];
      }
      reset_position = false;
    }
    if(mode < 99)
    {
      for(leg_num=0; leg_num<6; leg_num++)

leg_IK(leg_num,current_X[leg_num]+offset_X[leg_num],current_Y[leg_num]+offset_
Y[leg_num],current_Z[leg_num]+offset_Z[leg_num]);
    }

    //reset leg lift first pass flags if needed
    if(mode != 4)
    {
      leg1_IK_control = true;
      leg6_IK_control = true;
    }

    battery_monitor();
    print_debug();

    //process modes (mode 0 is default 'home idle' do-nothing mode)
    if(mode == 1)                           //walking mode
    {
      if(gait == 0) tripod_gait();          //walk using gait 0
      if(gait == 1) wave_gait();            //walk using gait 1
      if(gait == 2) ripple_gait();          //walk using gait 2
      if(gait == 3) tetrapod_gait();        //walk using gait 3
    }
    if(mode == 2) translate_control();      //joystick control x-y-z mode
    if(mode == 3) rotate_control();         //joystick control y-p-r mode
    if(mode == 4) one_leg_lift();           //one leg lift mode
    if(mode == 99) set_all_90();            //set all servos to 90 degrees
mode
  }
}
void process_gamepad()
{
  if(ps2x.ButtonPressed(PSB_PAD_DOWN))    //stop & select gait 0
  {
    mode = 0;
    gait = 0;
    reset_position = true;
  }
  if(ps2x.ButtonPressed(PSB_PAD_LEFT))    //stop & select gait 1
  {
    mode = 0;
    gait = 1;
    reset_position = true;
  }
  if(ps2x.ButtonPressed(PSB_PAD_UP))      //stop & select gait 2
  {
    mode = 0;
    gait = 2;
    reset_position = true;
  }
  if(ps2x.ButtonPressed(PSB_PAD_RIGHT))   //stop & select gait 3
  {
    mode = 0;
    gait = 3;
    reset_position = true;
  }
  if(mode == 0)
  {
    if(batt_LEDs > 3) gait_LED_color=0;
    else gait_LED_color=1;
    if(ps2x.Button(PSB_PAD_DOWN))  LED_Bar(gait_LED_color,1);    //display
gait 0
    if(ps2x.Button(PSB_PAD_LEFT))  LED_Bar(gait_LED_color,2);    //display
gait 1
    if(ps2x.Button(PSB_PAD_UP))    LED_Bar(gait_LED_color,3);    //display
gait 2
    if(ps2x.Button(PSB_PAD_RIGHT)) LED_Bar(gait_LED_color,4);    //display
gait 3
  }
  if(ps2x.ButtonPressed(PSB_TRIANGLE))    //select walk mode
  {
    mode = 1;
    reset_position = true;
  }
  if(ps2x.Button(PSB_TRIANGLE))           //vibrate controller if walk button
held
    gamepad_vibrate = 64;
  else
    gamepad_vibrate = 0;
  if(ps2x.ButtonPressed(PSB_SQUARE))      //control x-y-z with joysticks mode
  {
    mode = 2;
    reset_position = true;
  }
  if(ps2x.ButtonPressed(PSB_CIRCLE))      //control y-p-r with joysticks mode
  {
    mode = 3;
    reset_position = true;
  }
  if(ps2x.ButtonPressed(PSB_CROSS))       //one leg lift mode
  {
    mode = 4;
    reset_position = true;
  }
  if(ps2x.ButtonPressed(PSB_START))       //change gait speed
  {
    if(gait_speed == 0)
      gait_speed = 1;
    else
      gait_speed = 0;
  }
  if(ps2x.Button(PSB_START))              //display gait speed on LEDs if
button held
  {
    if(gait_speed == 0) LED_Bar(1,8);     //use green LEDs for fast
    else LED_Bar(0,8);                    //use red LEDs for slow
  }
  if(ps2x.ButtonPressed(PSB_SELECT))      //set all servos to 90 degrees for
calibration
  {
    mode = 99;
  }
  if((ps2x.ButtonPressed(PSB_L1)) || (ps2x.ButtonPressed(PSB_R1)))
  {
    //capture offsets in translate, rotate, and translate/rotate modes
    capture_offsets = true;
  }
  if((ps2x.ButtonPressed(PSB_L2)) || (ps2x.ButtonPressed(PSB_R2)))
  {
    for(leg_num=0; leg_num<6; leg_num++)  //clear offsets
    {
      offset_X[leg_num] = 0;
      offset_Y[leg_num] = 0;
      offset_Z[leg_num] = 0;
    }
    leg1_IK_control = true;               //reset leg lift first pass flags
    leg6_IK_control = true;
    step_height_multiplier = 1.0;         //reset step height multiplier
  }
}
void leg_IK(int leg_number,float X,float Y,float Z)
{
  //compute target femur-to-toe (L3) length
  L0 = sqrt(sq(X) + sq(Y)) - COXA_LENGTH;
  L3 = sqrt(sq(L0) + sq(Z));
  if((L3 < (TIBIA_LENGTH+FEMUR_LENGTH)) && (L3 > (TIBIA_LENGTH-FEMUR_LENGTH)))

  {
    //compute tibia angle
    phi_tibia = acos((sq(FEMUR_LENGTH) + sq(TIBIA_LENGTH) -
sq(L3))/(2*FEMUR_LENGTH*TIBIA_LENGTH));
    theta_tibia = phi_tibia*RAD_TO_DEG - 23.0 + TIBIA_CAL[leg_number];
    theta_tibia = constrain(theta_tibia,0.0,180.0);

    //compute femur angle
    gamma_femur = atan2(Z,L0);
    phi_femur = acos((sq(FEMUR_LENGTH) + sq(L3) -
sq(TIBIA_LENGTH))/(2*FEMUR_LENGTH*L3));
```

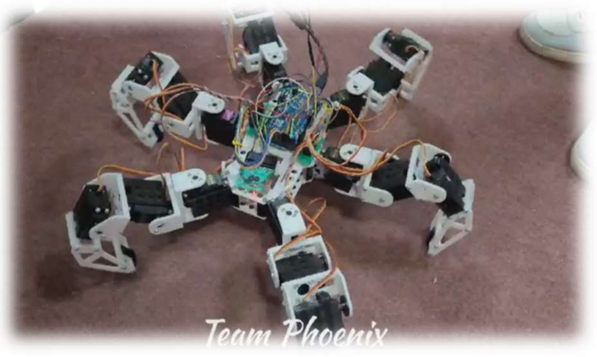
```
    theta_femur = (phi_femur + gamma_femur)*RAD_TO_DEG + 14.0 + 90.0 +
FEMUR_CAL[leg_number];
    theta_femur = constrain(theta_femur,0.0,180.0);

    //compute coxa angle
    theta_coxa = atan2(X,Y)*RAD_TO_DEG + COXA_CAL[leg_number];

    //output to the appropriate leg
    switch(leg_number)
    {
      case 0:
        if(leg1_IK_control == true)                       //flag for IK or
manual control of leg
        {
          theta_coxa = theta_coxa + 45.0;                 //compensate for leg
mounting
          theta_coxa = constrain(theta_coxa,0.0,180.0);
          coxa1_servo.write(int(theta_coxa));
          femur1_servo.write(int(theta_femur));
          tibia1_servo.write(int(theta_tibia));
        }
        break;
      case 1:
        theta_coxa = theta_coxa + 90.0;                  //compensate for leg
mounting
        theta_coxa = constrain(theta_coxa,0.0,180.0);
        coxa2_servo.write(int(theta_coxa));
        femur2_servo.write(int(theta_femur));
        tibia2_servo.write(int(theta_tibia));
        break;
      case 2:
        theta_coxa = theta_coxa + 135.0;
        theta_coxa = constrain(theta_coxa,0.0,180.0);
        coxa3_servo.write(int(theta_coxa));
        femur3_servo.write(int(theta_femur));
        tibia3_servo.write(int(theta_tibia));
        break;
      case 3:
        if(theta_coxa < 0)
          theta_coxa = theta_coxa + 225.0;                // (need to use
different
        else                                              //  positive and
negative offsets
          theta_coxa = theta_coxa - 135.0;                //  due to atan2
results above!)
        theta_coxa = constrain(theta_coxa,0.0,180.0);
        coxa4_servo.write(int(theta_coxa));
        femur4_servo.write(int(theta_femur));
        tibia4_servo.write(int(theta_tibia));
        break;
      case 4:
        if(theta_coxa < 0)                                 //compensate for leg
          theta_coxa = theta_coxa + 270.0;                 // (need to use
        else                                               //  positive and
          theta_coxa = theta_coxa - 90.0;                  //  due to atan2
        theta_coxa = constrain(theta_coxa,0.0,180.0);
        coxa5_servo.write(int(theta_coxa));
        femur5_servo.write(int(theta_femur));
        tibia5_servo.write(int(theta_tibia));
        break;
      case 5:
        if(leg6_IK_control == true)                        //flag for IK or
          if(theta_coxa < 0)                               //compensate for leg
            theta_coxa = theta_coxa + 315.0;               // (need to use
            theta_coxa else                                             //
 positive and
= theta_coxa - 45.0;                //  due to atan2 results above!)
          theta_coxa = constrain(theta_coxa,0.0,180.0);
          coxa6_servo.write(int(theta_coxa));
          femur6_servo.write(int(theta_femur));
          tibia6_servo.write(int(theta_tibia));
        }
        break;
    }
  }
}
void tripod_gait()
{
  //read commanded values from controller
  commandedX = map(ps2x.Analog(PSS_RY),0,255,127,-127);
  commandedY = map(ps2x.Analog(PSS_RX),0,255,-127,127);
  commandedR = map(ps2x.Analog(PSS_LX),0,255,127,-127);
  if((abs(commandedX) > 15) || (abs(commandedY) > 15) || (abs(commandedR) >
15) || (tick>0))
  {
    compute_strides();
    numTicks = round(duration / FRAME_TIME_MS / 2.0); //total ticks divided
into the two cases
    for(leg_num=0; leg_num<6; leg_num++)
    {
      compute_amplitudes();
      switch(tripod_case[leg_num])
      {
        case 1:                                  forward (raise and lower)
```

```
          current_X[leg_num] = HOME_X[leg_num] -
amplitudeX*cos(M_PI*tick/numTicks);
          current_Y[leg_num] = HOME_Y[leg_num] -
amplitudeY*cos(M_PI*tick/numTicks);
          current_Z[leg_num] = HOME_Z[leg_num] +
abs(amplitudeZ)*sin(M_PI*tick/numTicks);
          if(tick >= numTicks-1) tripod_case[leg_num] = 2;
          break;
        case 2:                                 back (on the ground)
          current_X[leg_num] = HOME_X[leg_num] +
amplitudeX*cos(M_PI*tick/numTicks);
          current_Y[leg_num] = HOME_Y[leg_num] +
amplitudeY*cos(M_PI*tick/numTicks);
          current_Z[leg_num] = HOME_Z[leg_num];
          if(tick >= numTicks-1) tripod_case[leg_num] = 1;
          break;
      }
    }
    //increment tick
    if(tick < numTicks-1) tick++;
    else tick = 0;
  }
}
void wave_gait()
{
  commandedX = map(ps2x.Analog(PSS_RY),0,255,127,-127);
  commandedY = map(ps2x.Analog(PSS_RX),0,255,-127,127);
  commandedR = map(ps2x.Analog(PSS_LX),0,255,127,-127);

  //if commands more than deadband then process
  if((abs(commandedX) > 15) || (abs(commandedY) > 15) || (abs(commandedR) >
15) || (tick>0))
  {
    compute_strides();
    numTicks = round(duration / FRAME_TIME_MS / 6.0); //total ticks divided
into the six cases
    for(leg_num=0; leg_num<6; leg_num++)
    {
      compute_amplitudes();
      switch(wave_case[leg_num])
      {
        case 1:                                 forward (raise and lower)
          current_X[leg_num] = HOME_X[leg_num] -
amplitudeX*cos(M_PI*tick/numTicks);
          current_Y[leg_num] = HOME_Y[leg_num] -
amplitudeY*cos(M_PI*tick/numTicks);
```

```
          current_Z[leg_num] = HOME_Z[leg_num] +
abs(amplitudeZ)*sin(M_PI*tick/numTicks);
          if(tick >= numTicks-1) wave_case[leg_num] = 6;
          break;
        case 2:                                   back one-fifth (on the ground)
          current_X[leg_num] = current_X[leg_num] - amplitudeX/numTicks/2.5;
          current_Y[leg_num] = current_Y[leg_num] - amplitudeY/numTicks/2.5;
          current_Z[leg_num] = HOME_Z[leg_num];
          if(tick >= numTicks-1) wave_case[leg_num] = 1;
          break;
        case 3:                                   back one-fifth (on the ground)
          current_X[leg_num] = current_X[leg_num] - amplitudeX/numTicks/2.5;
          current_Y[leg_num] = current_Y[leg_num] - amplitudeY/numTicks/2.5;
          current_Z[leg_num] = HOME_Z[leg_num];
          if(tick >= numTicks-1) wave_case[leg_num] = 2;
          break;
        case 4:                                   back one-fifth (on the ground)
          current_X[leg_num] = current_X[leg_num] - amplitudeX/numTicks/2.5;
          current_Y[leg_num] = current_Y[leg_num] - amplitudeY/numTicks/2.5;
          current_Z[leg_num] = HOME_Z[leg_num];
          if(tick >= numTicks-1)
            wave_case[leg_num] = 3;
          break;
        case 5:                                   back one-fifth (on the ground)
          current_X[leg_num] = current_X[leg_num] - amplitudeX/numTicks/2.5;
          current_Y[leg_num] = current_Y[leg_num] - amplitudeY/numTicks/2.5;
          current_Z[leg_num] = HOME_Z[leg_num];
          if(tick >= numTicks-1) wave_case[leg_num] = 4;
          break;
        case 6:                                   back one-fifth (on the ground)
          current_X[leg_num] = current_X[leg_num] - amplitudeX/numTicks/2.5;
          current_Y[leg_num] = current_Y[leg_num] - amplitudeY/numTicks/2.5;
          current_Z[leg_num] = HOME_Z[leg_num];
          if(tick >= numTicks-1) wave_case[leg_num] = 5;
          break;
      }
    }
    //increment tick
    if(tick < numTicks-1) tick++;
    else tick = 0;
  }
}
void ripple_gait()
{
  //read commanded values from controller
  commandedX = map(ps2x.Analog(PSS_RY),0,255,127,-127);
  commandedY = map(ps2x.Analog(PSS_RX),0,255,-127,127);
  commandedR = map(ps2x.Analog(PSS_LX),0,255,127,-127);

  //if commands more than deadband then process
  if((abs(commandedX) > 15) || (abs(commandedY) > 15) || (abs(commandedR) >
15) || (tick>0))
  {
    compute_strides();
    numTicks = round(duration / FRAME_TIME_MS / 6.0); //total ticks divided
into the six cases
    for(leg_num=0; leg_num<6; leg_num++)
    {
      compute_amplitudes();
      switch(ripple_case[leg_num])
      {
        case 1:                               forward (raise)
          current_X[leg_num] = HOME_X[leg_num] -
amplitudeX*cos(M_PI*tick/(numTicks*2));
          current_Y[leg_num] = HOME_Y[leg_num] -
amplitudeY*cos(M_PI*tick/(numTicks*2));
          current_Z[leg_num] = HOME_Z[leg_num] +
abs(amplitudeZ)*sin(M_PI*tick/(numTicks*2));
          if(tick >= numTicks-1) ripple_case[leg_num] = 2;
          break;
        case 2:                               forward (lower)
          current_X[leg_num] = HOME_X[leg_num] -
amplitudeX*cos(M_PI*(numTicks+tick)/(numTicks*2));
          current_Y[leg_num] = HOME_Y[leg_num] -
amplitudeY*cos(M_PI*(numTicks+tick)/(numTicks*2));
          current_Z[leg_num] = HOME_Z[leg_num] +
abs(amplitudeZ)*sin(M_PI*(numTicks+tick)/(numTicks*2));
          if(tick >= numTicks-1) ripple_case[leg_num] = 3;
          break;
        case 3:
          current_X[leg_num] = current_X[leg_num] - amplitudeX/numTicks/2.0;
          current_Y[leg_num] = current_Y[leg_num] - amplitudeY/numTicks/2.0;
          current_Z[leg_num] = HOME_Z[leg_num];
          if(tick >= numTicks-1) ripple_case[leg_num] = 4;
          break;
        case 4:
          current_X[leg_num] = current_X[leg_num] - amplitudeX/numTicks/2.0;
          current_Y[leg_num] = current_Y[leg_num] - amplitudeY/numTicks/2.0;
          current_Z[leg_num] = HOME_Z[leg_num];
          if(tick >= numTicks-1) ripple_case[leg_num] = 5;
          break;
        case 5:
          current_X[leg_num] = current_X[leg_num] - amplitudeX/numTicks/2.0;
          current_Y[leg_num] = current_Y[leg_num] - amplitudeY/numTicks/2.0;
          current_Z[leg_num] = HOME_Z[leg_num];
          if(tick >= numTicks-1) ripple_case[leg_num] = 6;
          break;
        case 6:
          current_X[leg_num] = current_X[leg_num] - amplitudeX/numTicks/2.0;
          current_Y[leg_num] = current_Y[leg_num] - amplitudeY/numTicks/2.0;
          current_Z[leg_num] = HOME_Z[leg_num];
          if(tick >= numTicks-1) ripple_case[leg_num] = 1;
          break;
      }
    }
    //increment tick
    if(tick < numTicks-1) tick++;
    else tick = 0;
  }
}
void tetrapod_gait()
{
  //read commanded values from controller
  commandedX = map(ps2x.Analog(PSS_RY),0,255,127,-127);
  commandedY = map(ps2x.Analog(PSS_RX),0,255,-127,127);
  commandedR = map(ps2x.Analog(PSS_LX),0,255,127,-127);

  //if commands more than deadband then process
  if((abs(commandedX) > 15) || (abs(commandedY) > 15) || (abs(commandedR) >
15) || (tick>0))
  {
    compute_strides();
    numTicks = round(duration / FRAME_TIME_MS / 3.0); //total ticks divided
into the three cases
    for(leg_num=0; leg_num<6; leg_num++)
    {
      compute_amplitudes();
      switch(tetrapod_case[leg_num])
      {
        case 1:                               forward (raise and lower)
          current_X[leg_num] = HOME_X[leg_num] -
amplitudeX*cos(M_PI*tick/numTicks);
          current_Y[leg_num] = HOME_Y[leg_num] -
amplitudeY*cos(M_PI*tick/numTicks);
          current_Z[leg_num] = HOME_Z[leg_num] +
abs(amplitudeZ)*sin(M_PI*tick/numTicks);
          if(tick >= numTicks-1) tetrapod_case[leg_num] = 2;
          break;
        case 2:                               back one-half (on the ground)
          current_X[leg_num] = current_X[leg_num] - amplitudeX/numTicks;
          current_Y[leg_num] = current_Y[leg_num] - amplitudeY/numTicks;
          current_Z[leg_num] = HOME_Z[leg_num];
          if(tick >= numTicks-1) tetrapod_case[leg_num] = 3;
          break;
        case 3:                                 back one-half (on the ground)
          current_X[leg_num] = current_X[leg_num] - amplitudeX/numTicks;
          current_Y[leg_num] = current_Y[leg_num] - amplitudeY/numTicks;
          current_Z[leg_num] = HOME_Z[leg_num];
          if(tick >= numTicks-1) tetrapod_case[leg_num] = 1;
          break;
      }
    }
    //increment tick
    if(tick < numTicks-1) tick++;
    else tick = 0;
  }
}
void compute_strides()
{
  //compute stride lengths
  strideX = 90*commandedX/127;
  strideY = 90*commandedY/127;
  strideR = 35*commandedR/127;

  //compute rotation trig
  sinRotZ = sin(radians(strideR));
  cosRotZ = cos(radians(strideR));
  if(gait_speed == 0) duration = 1080;
  else duration = 3240;
}
void compute_amplitudes()
{
  //compute total distance from center of body to toe
  totalX = HOME_X[leg_num] + BODY_X[leg_num];
  totalY = HOME_Y[leg_num] + BODY_Y[leg_num];

  //compute rotational offset
  rotOffsetX = totalY*sinRotZ + totalX*cosRotZ - totalX;
  rotOffsetY = totalY*cosRotZ - totalX*sinRotZ - totalY;

  //compute X and Y amplitude and constrain to prevent legs from crashing into
each other
  amplitudeX = ((strideX + rotOffsetX)/2.0);
  amplitudeY = ((strideY + rotOffsetY)/2.0);
  amplitudeX = constrain(amplitudeX,-50,50);
  amplitudeY = constrain(amplitudeY,-50,50);

  //compute Z amplitude
  if(abs(strideX + rotOffsetX) > abs(strideY + rotOffsetY))
    amplitudeZ = step_height_multiplier * (strideX + rotOffsetX) /4.0;
  else
    amplitudeZ = step_height_multiplier * (strideY + rotOffsetY) / 4.0;
}
void translate_control()
{
  translateX = map(ps2x.Analog(PSS_RY),0,255,-2*TRAVEL,2*TRAVEL);
  for(leg_num=0; leg_num<6; leg_num++)
    current_X[leg_num] = HOME_X[leg_num] + translateX;

  //compute Y direction move
  translateY = map(ps2x.Analog(PSS_RX),0,255,2*TRAVEL,-2*TRAVEL);
  for(leg_num=0; leg_num<6; leg_num++)
    current_Y[leg_num] = HOME_Y[leg_num] + translateY;

  //compute Z direction move
  translateZ = ps2x.Analog(PSS_LY);
  if(translateZ > 127)
    translateZ = map(translateZ,128,255,0,TRAVEL);
  else
    translateZ = map(translateZ,0,127,-3*TRAVEL,0);
  for(leg_num=0; leg_num<6; leg_num++)
    current_Z[leg_num] = HOME_Z[leg_num] + translateZ;

  if(capture_offsets == true)
  {
    for(leg_num=0; leg_num<6; leg_num++)
    {
      offset_X[leg_num] = offset_X[leg_num] + translateX;
      offset_Y[leg_num] = offset_Y[leg_num] + translateY;
      offset_Z[leg_num] = offset_Z[leg_num] + translateZ;
      current_X[leg_num] = HOME_X[leg_num];
      current_Y[leg_num] = HOME_Y[leg_num];
      current_Z[leg_num] = HOME_Z[leg_num];
    }
  }
  if(capture_offsets == true)
  {
    capture_offsets = false;
    mode = 0;
  }
}
void rotate_control()
{
  //compute rotation sin/cos values using controller inputs
  sinRotX = sin((map(ps2x.Analog(PSS_RX),0,255,A12DEG,-A12DEG))/1000000.0);
  cosRotX = cos((map(ps2x.Analog(PSS_RX),0,255,A12DEG,-A12DEG))/1000000.0);
  sinRotY = sin((map(ps2x.Analog(PSS_RY),0,255,A12DEG,-A12DEG))/1000000.0);
  cosRotY = cos((map(ps2x.Analog(PSS_RY),0,255,A12DEG,-A12DEG))/1000000.0);
  sinRotZ = sin((map(ps2x.Analog(PSS_LX),0,255,-A30DEG,A30DEG))/1000000.0);
  cosRotZ = cos((map(ps2x.Analog(PSS_LX),0,255,-A30DEG,A30DEG))/1000000.0);

  //compute Z direction move
  translateZ = ps2x.Analog(PSS_LY);
  if(translateZ > 127)
    translateZ = map(translateZ,128,255,0,TRAVEL);
  else
    translateZ = map(translateZ,0,127,-3*TRAVEL,0);

  for(int leg_num=0; leg_num<6; leg_num++)
  {
    //compute total distance from center of body to toe
    totalX = HOME_X[leg_num] + BODY_X[leg_num];
    totalY = HOME_Y[leg_num] + BODY_Y[leg_num];
    totalZ = HOME_Z[leg_num] + BODY_Z[leg_num];

    //perform 3 axis rotations
    rotOffsetX =  totalX*cosRotY*cosRotZ + totalY*sinRotX*sinRotY*cosRotZ +
totalY*cosRotX*sinRotZ - totalZ*cosRotX*sinRotY*cosRotZ +
totalZ*sinRotX*sinRotZ - totalX;
    rotOffsetY = -totalX*cosRotY*sinRotZ - totalY*sinRotX*sinRotY*sinRotZ +
totalY*cosRotX*cosRotZ + totalZ*cosRotX*sinRotY*sinRotZ +
totalZ*sinRotX*cosRotZ - totalY;
    rotOffsetZ =  totalX*sinRotY          - totalY*sinRotX*cosRotY
                     + totalZ*cosRotX*cosRotY
 - totalZ;
    current_X[leg_num] = HOME_X[leg_num] + rotOffsetX;
    current_Y[leg_num] = HOME_Y[leg_num] + rotOffsetY;
    current_Z[leg_num] = HOME_Z[leg_num] + rotOffsetZ + translateZ;

    //lock in offsets if commanded
    if(capture_offsets == true)
    {
      offset_X[leg_num] = offset_X[leg_num] + rotOffsetX;
      offset_Y[leg_num] = offset_Y[leg_num] + rotOffsetY;
      offset_Z[leg_num] = offset_Z[leg_num] + rotOffsetZ + translateZ;
      current_X[leg_num] = HOME_X[leg_num];
      current_Y[leg_num] = HOME_Y[leg_num];
      current_Z[leg_num] = HOME_Z[leg_num];
    }
  }
  if(capture_offsets == true)
  {
    capture_offsets = false;
    mode = 0;
  }
}
void one_leg_lift()
{
  if(leg1_IK_control == true)
  {
    leg1_coxa  = coxa1_servo.read();
    leg1_femur = femur1_servo.read();
    leg1_tibia = tibia1_servo.read();
    leg1_IK_control = false;
  }
  if(leg6_IK_control == true)
  {
    leg6_coxa  = coxa6_servo.read();
    leg6_femur = femur6_servo.read();
    leg6_tibia = tibia6_servo.read();
    leg6_IK_control = false;
  }
  temp = ps2x.Analog(PSS_RX);
  temp = map(temp,0,255,45,-45);
  coxa1_servo.write(constrain(int(leg1_coxa+temp),45,135));
  temp = ps2x.Analog(PSS_RY);
  if(temp < 117)                                //if joystick moved up
  {
    temp = map(temp,116,0,0,24);                //move leg 1
    femur1_servo.write(constrain(int(leg1_femur+temp),0,170));
    tibia1_servo.write(constrain(int(leg1_tibia+4*temp),0,170));
  }
  else                                          //if joystick moved down
  {
    z_height_right = constrain(temp,140,255);   //set Z step height
    z_height_right = map(z_height_right,140,255,1,8);
  }
  temp = ps2x.Analog(PSS_LX);
  temp = map(temp,0,255,45,-45);
  coxa6_servo.write(constrain(int(leg6_coxa+temp),45,135));
  temp = ps2x.Analog(PSS_LY);
  if(temp < 117)                                //if joystick moved up
  {
    temp = map(temp,116,0,0,24);                //move leg 6
    femur6_servo.write(constrain(int(leg6_femur+temp),0,170));
    tibia6_servo.write(constrain(int(leg6_tibia+4*temp),0,170));
  }
  else                                          //if joystick moved down
  {
    z_height_left = constrain(temp,140,255);     //set Z step height
    z_height_left = map(z_height_left,140,255,1,8);
  }

  if(z_height_left>z_height_right)
    z_height_right = z_height_left;              //use max left or right value
  if(batt_LEDs > 3) z_height_LED_color=0;        //use red LEDs if battery
strong
  else z_height_LED_color=1
  LED_Bar(z_height_LED_color,z_height_right);    //display Z height
  if(capture_offsets == true)
  {
    step_height_multiplier = 1.0 + ((z_height_right - 1.0) / 3.0);
    capture_offsets = false;
  }
}
void set_all_90()
{
  coxa1_servo.write(90+COXA_CAL[0]);
  femur1_servo.write(90+FEMUR_CAL[0]);
  tibia1_servo.write(90+TIBIA_CAL[0]);

  coxa2_servo.write(90+COXA_CAL[1]);
  femur2_servo.write(90+FEMUR_CAL[1]);
  tibia2_servo.write(90+TIBIA_CAL[1]);

  coxa3_servo.write(90+COXA_CAL[2]);
  femur3_servo.write(90+FEMUR_CAL[2]);
  tibia3_servo.write(90+TIBIA_CAL[2]);

  coxa4_servo.write(90+COXA_CAL[3]);
  femur4_servo.write(90+FEMUR_CAL[3]);
  tibia4_servo.write(90+TIBIA_CAL[3]);

  coxa5_servo.write(90+COXA_CAL[4]);
  femur5_servo.write(90+FEMUR_CAL[4]);
  tibia5_servo.write(90+TIBIA_CAL[4]);

  coxa6_servo.write(90+COXA_CAL[5]);
  femur6_servo.write(90+FEMUR_CAL[5]);
  tibia6_servo.write(90+TIBIA_CAL[5]);
}

void battery_monitor()
{
  batt_voltage_sum = batt_voltage_sum -
batt_voltage_array[batt_voltage_index];
```

```
  batt_voltage_array[batt_voltage_index] =
map(analogRead(BATT_VOLTAGE),0,1023,0,1497);
  batt_voltage_sum = batt_voltage_sum +
batt_voltage_array[batt_voltage_index];
  batt_voltage_index = batt_voltage_index + 1;
  if(batt_voltage_index > 49) batt_voltage_index = 0;
  batt_voltage = batt_voltage_sum / 50;
  batt_LEDs = map(constrain(batt_voltage,1020,1230),1020,1230,1,8);
  if(batt_LEDs > 3) LED_Bar(1,batt_LEDs); /splay green if voltage >= 11.40V
  else LED_Bar(0,batt_LEDs);              //display red if voltage < 11.40V
}

void print_debug()
{
  currentTime = millis();
  Serial.print(currentTime-previousTime);
  Serial.print(",");
  Serial.print(float(batt_voltage)/100.0);
  Serial.print("\n");
}
```

Result and Discussion



❖ Advantages:-

1) **Stability:-**
   The six legs provide excellent stability and balance, allowing the robot to navigate uneven terrain with ease.
2) **Redundancy:-**
   If one leg gets damaged, the hexapod robot can still function and adapt its gait to maintain stability.
3) **Maneuverable:-**
   The agile design enables the robot to move in any direction, making it highly maneuverable and capable of intricate movements.
4) **Load Capacity:-**
   Hexapod robots can carry heavy payloads, making them suitable for various applications, including rescue missions and industrial tasks.
5) Hexapods also show robustness.

❖ Future Scope:-

1) **Swarm Robotics:-**
   The integration of multiple hexapod robots working collaboratively will open new possibilities for complex tasks and swarm intelligence.
2) **Human Interaction:-**
   Advancements in artificial intelligence and human-robot interaction will enable hexapod robots to assist humans in various domestic and professional settings.
3) **Biomimicry:-**
   Hexapod robots will continue to draw inspiration from nature, aiming to replicate the efficiency, stability, and adaptability of real insects and other arthrop

Conclusion

This project emphasis the need for developing the legged robot rather than the wheeled robot. The model which is designed is basically based on the structure of six legged insects and its movements. This hexapod model is mainly designed for in places such as the after effects of the war and disaster zones which has an ability of obstacle avoidance, surveillance. It is designed in such a way that it has improved stability and performance compared to the other legged robots. Many experiments and tests are made to improve the overall performance of the robot and the future work will be concentrated on the energy consumption, movements