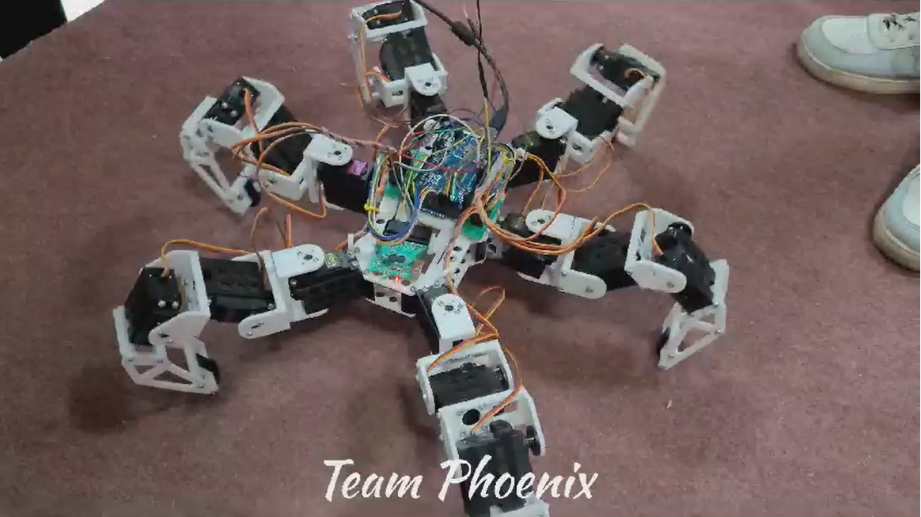
**PS2 Controlled Hexapod Robot**

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

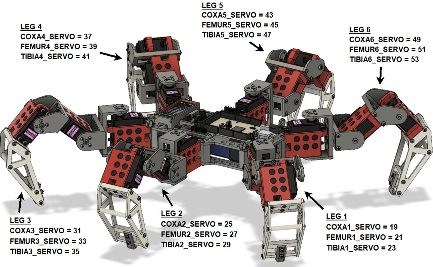
****

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:-

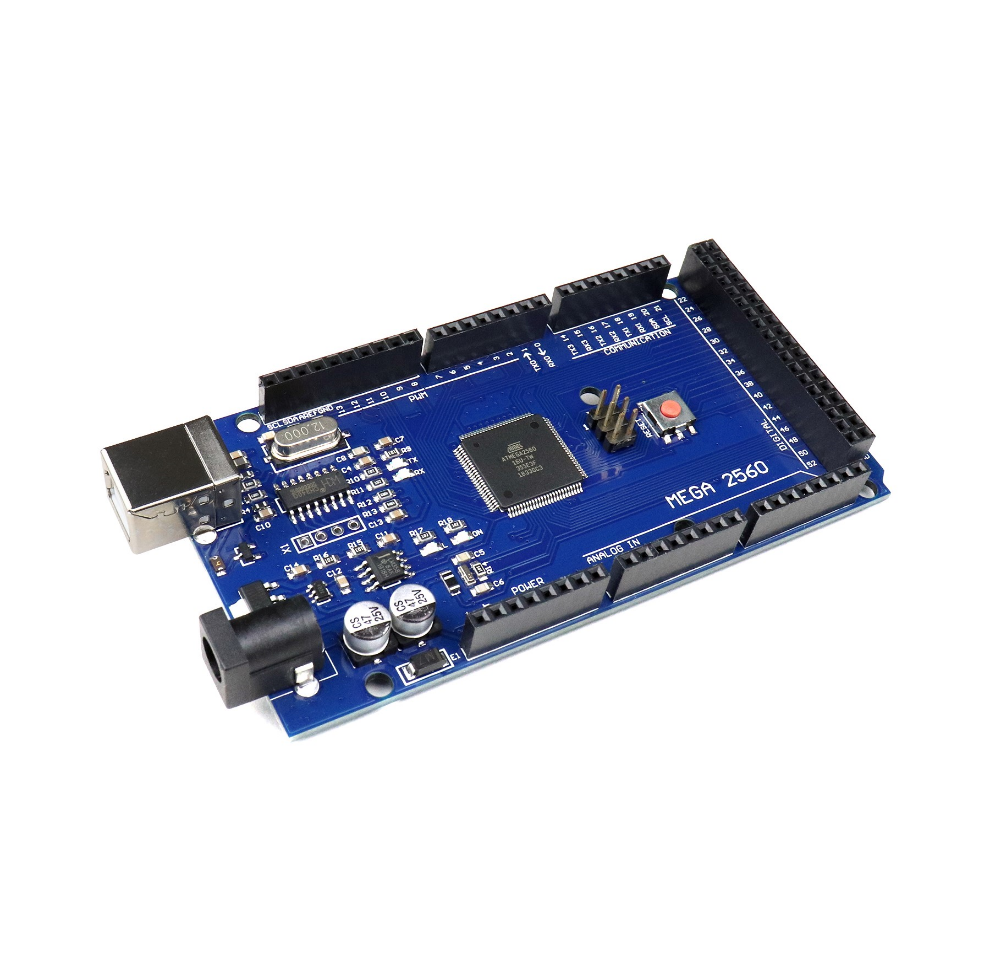
1. To study the movement and dynamics of the Hexapod robot.
2. Designing the model of Hexapod robot on CAD software Solid Works.
3. To design the Hexapod basing on the market needs and making it available for selling in the market.
4. For modifying the design based on requirements.
5. Analysis and simulation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

**Methodology**

**3.1 Required tools for the project:-**

1. Hardware Requirements:-
2. Arduino Mega2560 R3
3. Servo Motor
4. 3D printed Parts
5. PS2 Controller
6. 5500 MAH Lipo Battery
7. Jumping Wires
8. 12-5v Buck Convertor
9. 624Z bearing
10. Screws
11. Software Requirements-:
12. Ardino Software
13. 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** |

1. **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** |

1. **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.

1. **PS2 Wireless controller with Reciever:=**

The PS2 wireless controller is a standard controller for the PlayStation 2 and is identical to the original [DualShock](https://en.wikipedia.org/wiki/DualShock#DualShock_3) controller for the PlayStation console. It features twelve analog (pressure-sensitive) buttons ( Χ, 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.



1. **5500 MAH Lithium-polymer Battery:-**

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.

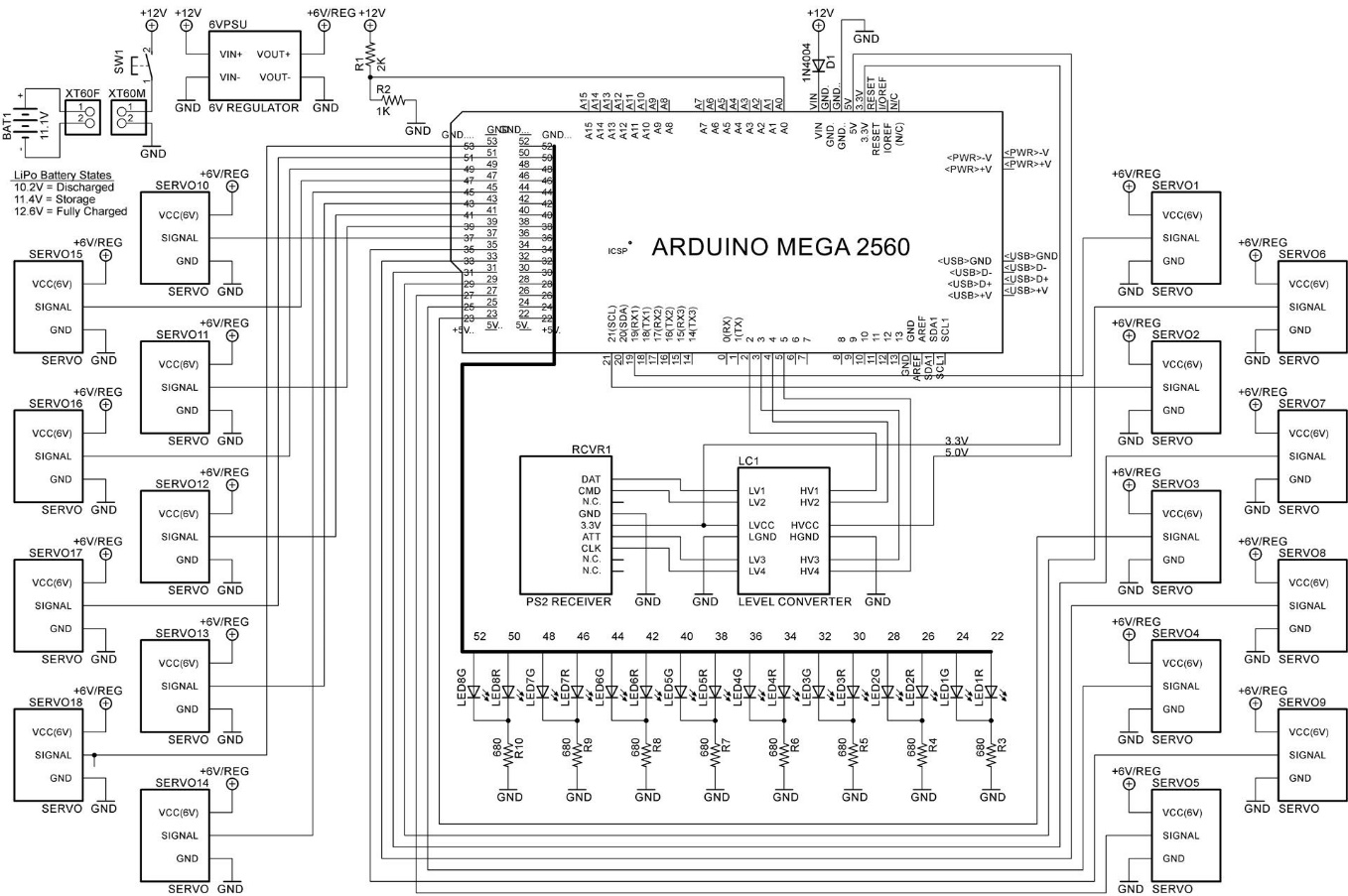
1. **624Z Bearings:-**

 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.

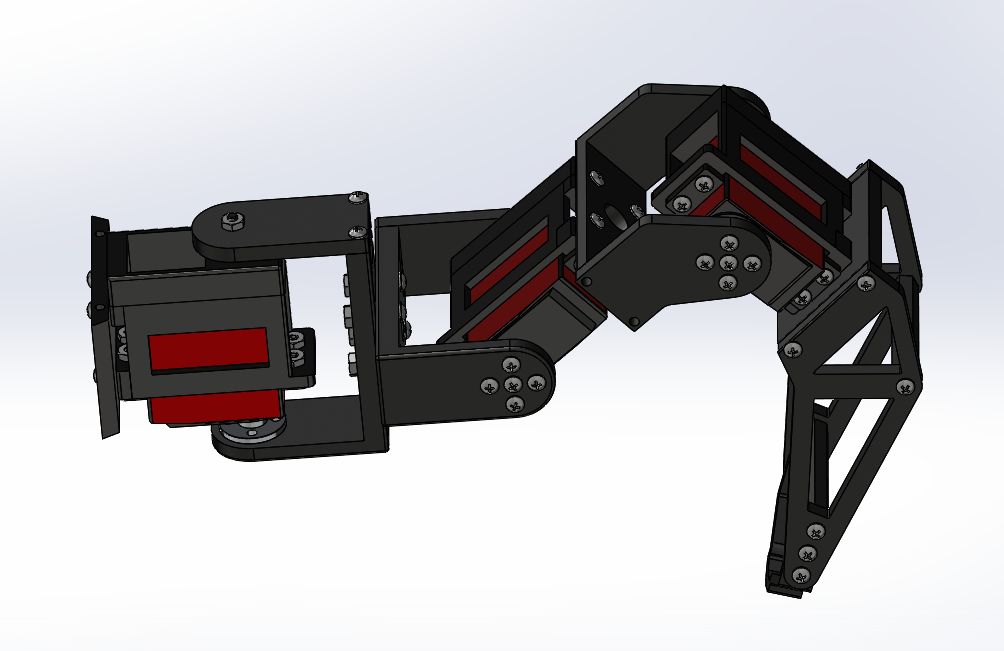
1. **Jumping Wires:-**

A jumper wire (also known as jumper, jumper wire, DuPont wire) is an [electrical wire](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Breadboard) or other prototype or test circuit, internally or with other equipment or components, without soldering.

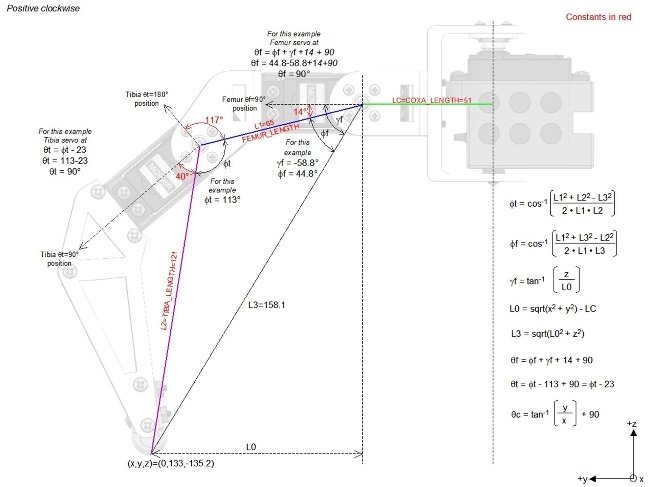
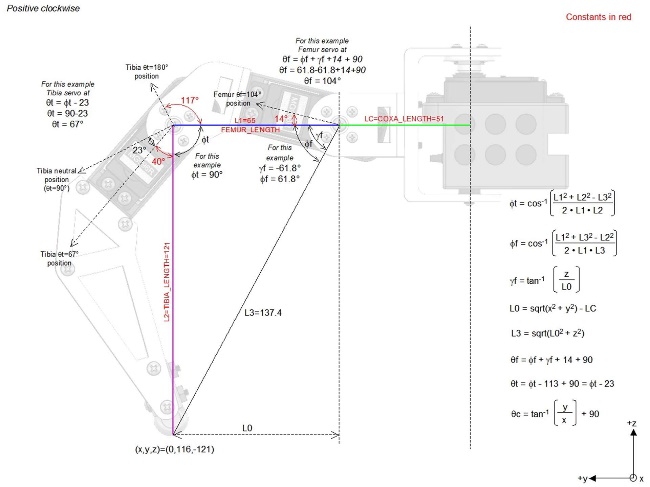
* 1. **Design of Project:-**

1. **Body:-**

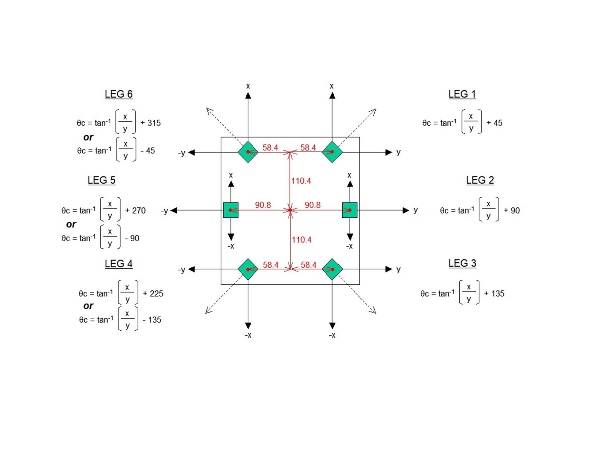
The main body constitutes of light-weight composite materials. It makes the whole project cost effective. To keep the center of gravity at the mid-point of the body, the legs are fixed at the opposite ends so that the robot can move continuously and balance it-self. The arduino is fixed in the back portion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

1. **Legs:-**

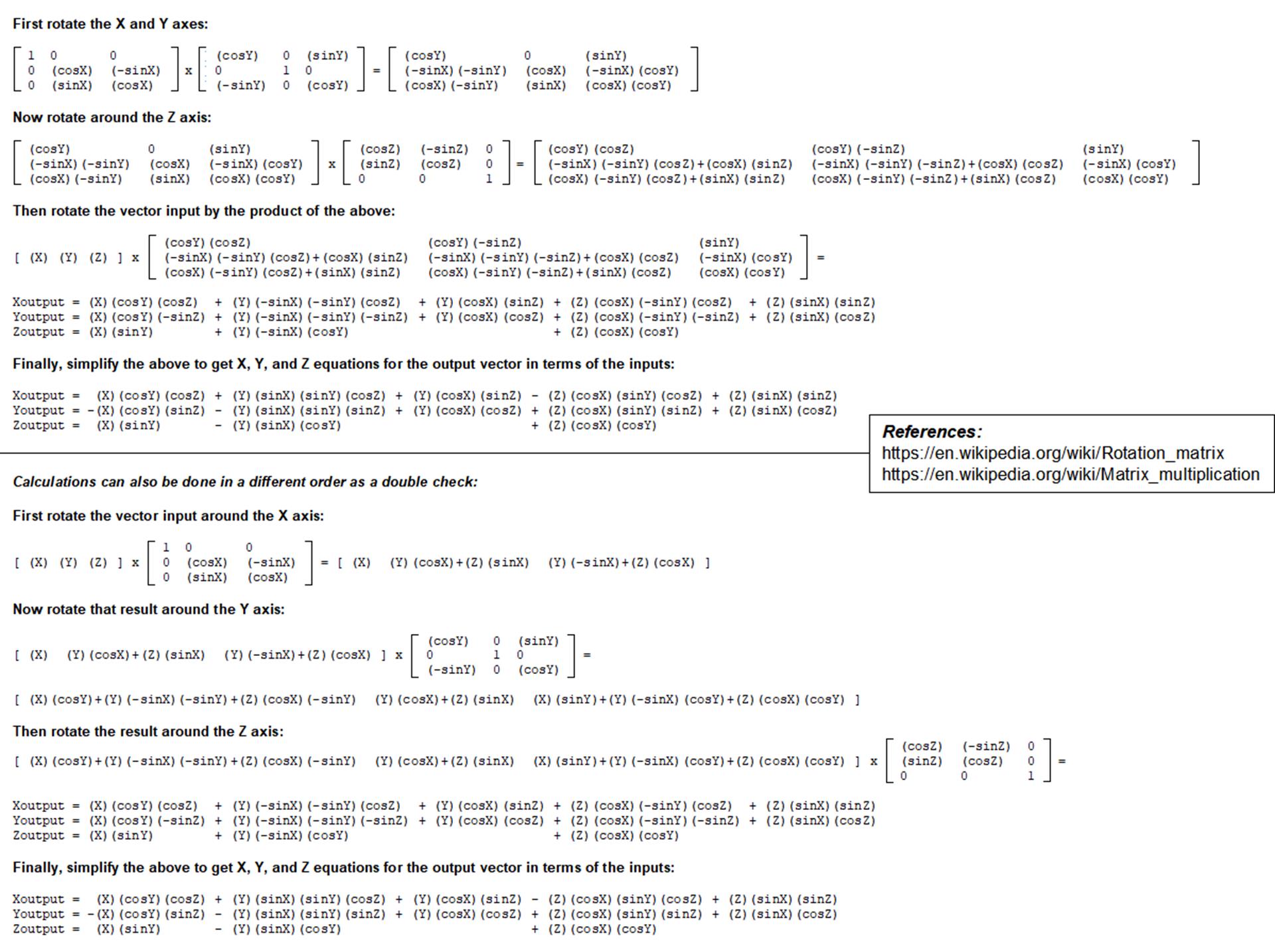
Each leg of the robot has the same functions and six degrees of freedom. The only difference between the legs is the motion algorithm. The legs are designed in such a way that it can walk on irregular platforms and can climb terrains. The movement access of one of its legs,

* 1. **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.

* 1. **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();

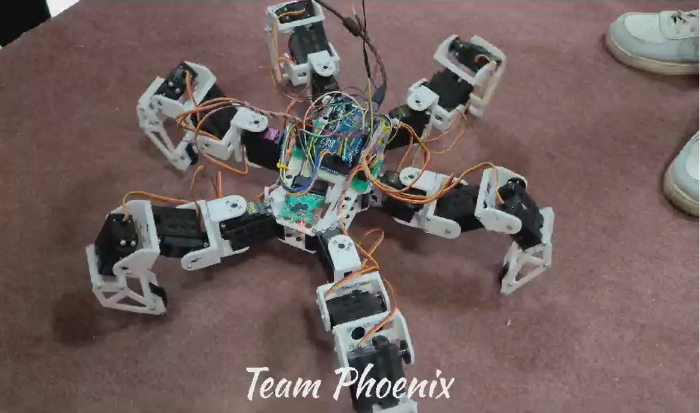
  Serial.print(currentTime-previousTime);

  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.

1. **Redundancy:-**

If one leg gets damaged, the hexapod robot can still function and adapt its gait to maintain stability.

1. **Maneuverable:-**

The agile design enables the robot to move in any direction, making it highly maneuverable and capable of intricate movements.

1. **Load Capacity:-**

Hexapod robots can carry heavy payloads, making them suitable for various applications, including rescue missions and industrial tasks.

1. 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.

1. **Human Interaction:-**

Advancements in artificial intelligence and human-robot interaction will enable hexapod robots to assist humans in various domestic and professional settings.

1. **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, movement