

A PROJECT REPORT ON STEWART PLATFORM

By

Ajinkya Bhole
Kanishka Bansal

2012A4PS063P
2012A4PS324P

Prepared in partial fulfillment of

ME G511

Mechanisms and Robotics



BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

ABSTRACT

This report documents the kinematic analysis, simulation and implementation of 6 DOF Stewart platform. The report also discusses the manufacturing process flow that was followed to fabricate the prototype of the robot. Processing, an open source language, was used for inverse kinematic study of the robot as well as to program the microcontroller with the help of compatible libraries. The development of Graphical User Interface (GUI), to manually control the platform in Processing is also briefly discussed. The last section of the report discusses a application of the Stewart platform as an experimental device to minimize the forced convective heat loss from the solar panel mounted on roofs.

Table of Contents

<i>Title Page</i>	1
<i>Abstract</i>	2
1. Introduction	4
2. Platform Design	5
2.1 Selection of parts	5
2.2 Cost Report	6
2.3 Platform Modelling	7
3. Platform Construction	8
3.1 Workshop	8
3.2 Control board	10
4. Platform Control	12
5. GUI and Simulation	15
5.1 Program structure	17
6. Proposed Application	18
6.1 Background	18
6.2 Harnessing the Energy	18
6.2 Application of Stewart platform	19
7. Conclusions and Future/Further work	20
8. Appendix	21
9. Bibliography	28

1. INTRODUCTION

A Stewart platform is a closed loop parallel manipulator whose Six DOF end effector is connected to the base platform by linear actuators. The first name is in honor of D.Stewart who was the first to introduce such a concept in 1965. It contains six linear actuators which enables the platform connected to base to have 6 degrees of freedom. Length of the actuators are varied by the rotating servo motors by specific angles to achieve different positions and orientation of the end effector – three rotational and three translational degrees.

It is widely used as flight simulator for pilot training, CNC machining and as surgical device. It is also used as a mounting station for large telescopes to orient it at any point on the sky. Further applications in industries include the flexible positioning of work pieces at specific angles and position. Good payload to net weight ratio, high precision & high flexibility are its major advantages.

The aim of the project is to document the development of the prototype of the platform and its application in the industry. This includes designing, construction, simulation and GUI to orient the end effector in desired location and orientation. This report also includes a brief description of the computer software and their results that were utilized in the completion of the project. CAD model of the platform was prepared in SolidWorks. In addition, the simulation was conducted using Processing language for a given orientation and position of the platform. The microcontroller was also programmed using the FIRMATA and Arduino library in Processing.

2. PLATFORM DESIGN

Platform Design consisted of selection of components and 3D modelling of the platform in SolidWorks. A brief description about each task with the factors that influenced the design decisions are discussed below.

2.1 SELECTION OF PARTS

The selection of material and the specifications of components involved weighing the available choices on the basis of economic feasibility and availability.

Major components

1. Base Platform & Moving Platform

The platforms are fabricated from Acrylic sheet.

Features - Light weight, easily available and high aesthetic value

2. Rotary Actuators

Rotary actuators were used in place of linear actuators because of their easy manufacturability and economic feasibility. Actuators are implemented by Servo Motor rather than Stepper Motor. The cost and easy control of the Servos lead to their use in the final design.

Features - Precision, high torque and low cost.

3. Links and Joints

Bicycle Spokes for the links and spherical ball socket for joints

Features – Cheap and easy availability

4. Microcontroller

ATmega328 microcontroller and its development board

Features - Preinstalled Arduino Bootloader & easily programmable

5. Battery

Lead acid battery – 6 V 4.5 Ah

Features - Satisfies servo requirements

2.2 COST REPORT

The cost report for each component used is given in the table below.

S. No.	Items	Quantity	Total Price (Rs)
1.	Avionic Servo 9g	6	1230
2.	Hausler 450 V2 and V3 Pro-plastic ball link set	1	98
3.	Hausler 450V2 and V3 Pro-plastic ball set	1	160
4.	Bicycle Spokes	6	12
5.	Acrylic Sheet	1 (200mm*200mm)	100
6.	Allen Bolt and Nut	1 (pack of 20)	40
7.	ATmega 328 Microcontroller Development Board	1	350
8.	Perforated Sheet	1	10
9.	Screw Terminal	1	5
10.	1N4007 Diode	1	2
11.	220uF 25V Capacitors	2	10
12.	Resistor - 100 ohm	1	0.5
13.	Resistor - 470 ohm	6	3
14.	Male Header Pins	1(pack of 50)	10
15.	Single Stranded Wire	1(5m)	10
16.	Battery 6V 4.5Ah	1	350
Total			2390.5

2.3 PLATFORM MODELLING

The modelling of the platform was done with SolidWorks. Procurement of the components was done after detailed analysis on CAD model of each part. The modelling ensured precise dimensioning of the parts and helped to determine the required torque of the servo motors.

This Motion Analysis toolbox of the model provided an estimate of the extent of translatory and rotatory motion of the platform. The model was simulated for different link lengths and a final model was proposed to meet the required specifications.

The Rendering of the final Model was made using PhotoView 360, a built in toolbox of SolidWorks. The model is shown in figure.

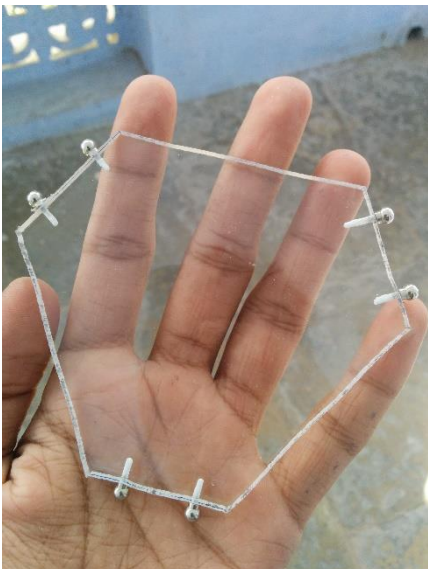


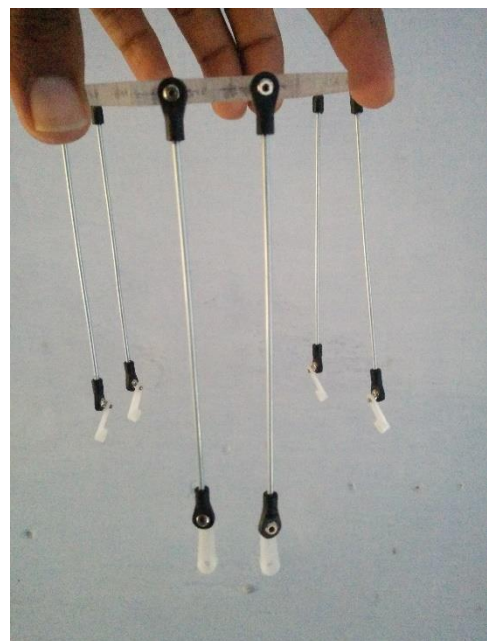
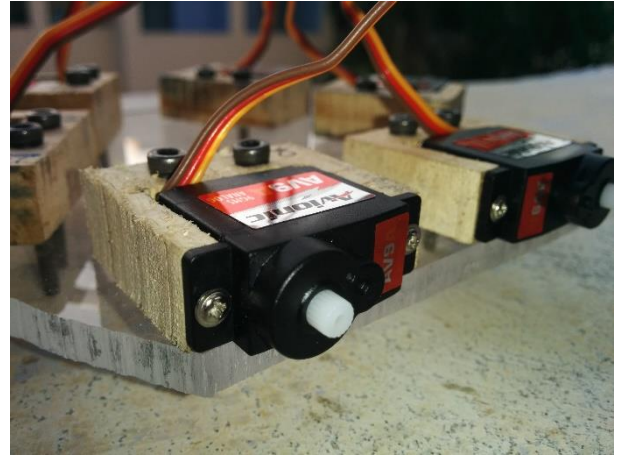
3. PLATFORM CONSTRUCTION

The CAD modelling of the platform was followed by initiation of fabrication phase of the prototype. All parts were manufactured and assembled in the mechanical workshop.

3.1 Workshop

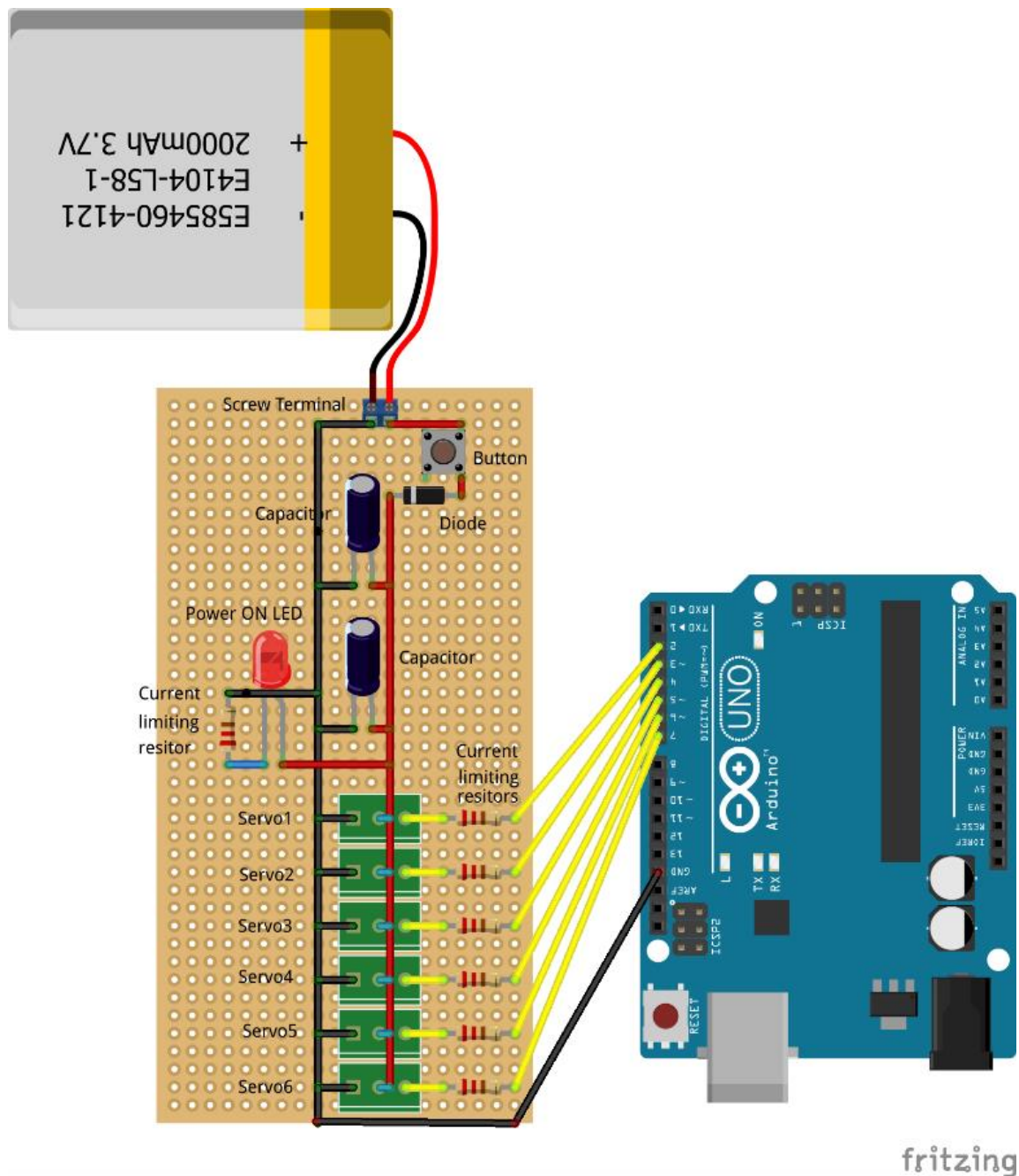
The acrylic sheet was cut in the desired shape and dimensions on the Do-All cutting machine. The servo mounts were made out of wood. Drilling and tapping holes on the acrylic Sheet and servo horn for Hausler Ball Sets posed manufacturing difficulties due to very less diameter of the ball sets nuts.





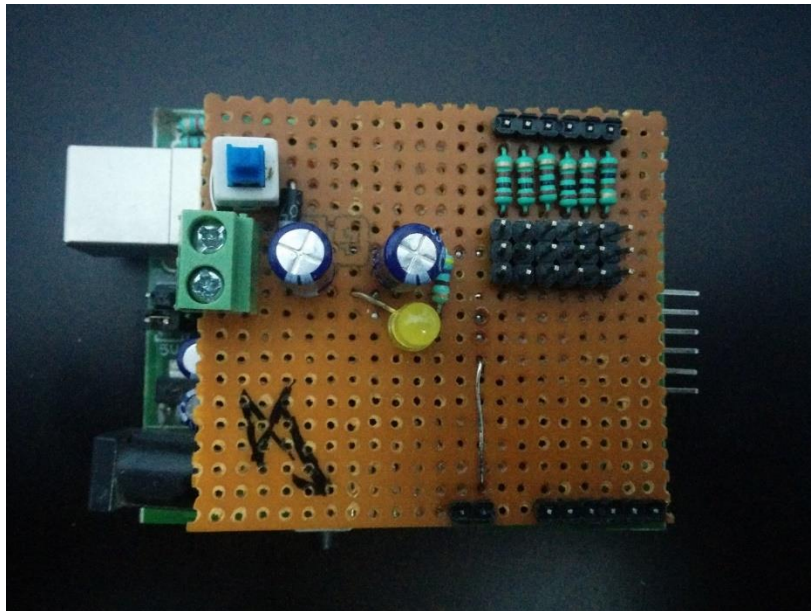
3.2 Control Board

A shield for the ATmega development board was prepared to counter the problems of wire routing of the servos and the battery. This also ensured a compact design of the board. A rough design for the Control Board was sketched in Fritzing, an open source software. The figure below shows the sketch.



Screw Terminal	To connect battery voltage input
Push button	Switching the control board On and Off.
Diode	Ensures unidirectional current flow
Electrolytic Capacitors (220uF 25V)	Stabilize voltage fluctuation
On/Off LED	Visual feedback of proper board functioning

The figure below show the manufactured control board.



4. PLATFORM CONTROL

The Stewart platform is a parallel robot with six rotatory actuators. The platform achieves variable translation and rotation motion by varying the lengths of the legs. The servos connected to the legs have to be driven simultaneously to achieve a desired position and orientation.

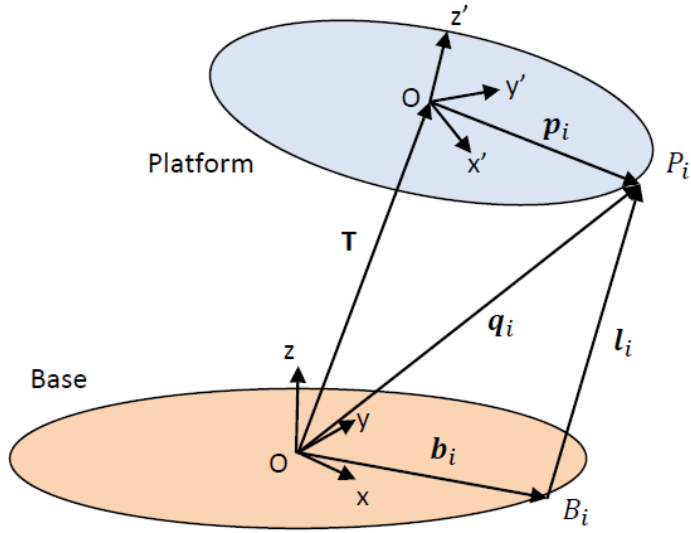
Stewart platform, a parallel robot has much simpler inverse kinematics model rather than forward kinematics unlike any other articulated robot. It requires to represent 18 simultaneous nonlinear equations in 6 unknowns (Angular displacement of each servo) to determine the end effector's position.

Thus, calculation of the leg lengths corresponding to servo's angular displacements for the desired position and orientation is much simpler than to determine the spatial parameters of the platform for corresponding rotations of the motor.

During literary survey, we came across a documentation by Wokingham U3A Math Group on mathematics of the inverse kinematics of the platform. There were two parameters to be determined to solve for inverse model.

1. The distance between the joint on the platform and the joint on the base for a given position and orientation.
2. The angular displacement of each servo that satisfies the distance constraint of each pair of platform and base joints.

The paper provided a closed-form solution for inverse kinematic transformation to convert Cartesian variables into required lengths of the leg and then corresponding servo angles. The paper is attached in Appendix for reference.



The full rotation matrix of the Platform relative to the Base is then given by:

$$\begin{aligned}
 {}^P\mathbf{R}_B &= \mathbf{R}_z(\psi) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{R}_x(\varphi) \\
 &= \begin{pmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & -\sin \varphi \\ 0 & \sin \varphi & \cos \varphi \end{pmatrix} \\
 &= \begin{pmatrix} \cos \psi \cos \theta & -\sin \psi & \cos \psi \sin \theta \\ \sin \psi \cos \theta & \cos \psi & \sin \psi \sin \theta \\ -\sin \theta & 0 & \cos \theta \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & -\sin \varphi \\ 0 & \sin \varphi & \cos \varphi \end{pmatrix} \\
 &= \begin{pmatrix} \cos \psi \cos \theta & -\sin \psi \cos \varphi + \cos \psi \sin \theta \sin \varphi & \sin \psi \sin \varphi + \cos \psi \sin \theta \cos \varphi \\ \sin \psi \cos \theta & \cos \psi \cos \varphi + \sin \psi \sin \theta \sin \varphi & -\cos \psi \sin \varphi + \sin \psi \sin \theta \cos \varphi \\ -\sin \theta & \cos \theta \sin \varphi & \cos \theta \cos \varphi \end{pmatrix}
 \end{aligned}$$

$$l_i = T + {}^P\mathbf{R}_B \cdot p_i - b_i$$

A_i are the points of the arm/leg joint on the i^{th} servo with coordinates

$$\mathbf{a} = [x_a \quad y_a \quad z_a]^T \text{ in the base framework.}$$

B_i are the points of rotation of the servo arms with the coordinates

$$\mathbf{b} = [x_b \quad y_b \quad z_b]^T \text{ in the base framework.}$$

P_i are the points the joints between the operating rods and the platform,
with coordinates $\mathbf{p} = [x_p \quad y_p \quad z_p]^T$ in the platform framework

After much simplification of the inverse kinematic equations, the corresponding angles of the servo motors in closed form are:

$$\alpha = \sin^{-1} \frac{L}{\sqrt{M^2 + N^2}} - \tan^{-1} \frac{N}{M}$$

$$\text{where } L = l^2 - (s^2 - a^2)$$

$$M = 2a(z_p - z_b)$$

$$N = 2a[\cos \beta (x_p - x_b) + \sin \beta (y_p - y_b)]$$

5. GUI AND SIMULATION

The inverse kinematic model of the platform involves complex trigonometric and algebraic functions. The determination of sets of joint variables which would achieve the desired position and orientation of the manipulator requires huge amount of processing memory and cannot be computed on a microcontroller. Therefore, all the processing of the equations to find the solutions is done on the computer and later the platform is controlled through serial communication to the microcontroller.

Processing, an open source language with built in graphic libraries for 2D and 3D output was used for computation of the inverse kinematic equations. Computer simulation of the platform was also conducted to evaluate the efficiency of the inverse kinematic model with the help of following graphics libraries of processing.

1. ControlP5

ControlP5 is a GUI and controller library for processing that provides controllers such as Sliders, Buttons, Toggles, Knobs, Text fields & Radio Buttons in applet mode,

2. Peasycam

Peasycam is an open source library of processing which provides mouse driven camera rendering.

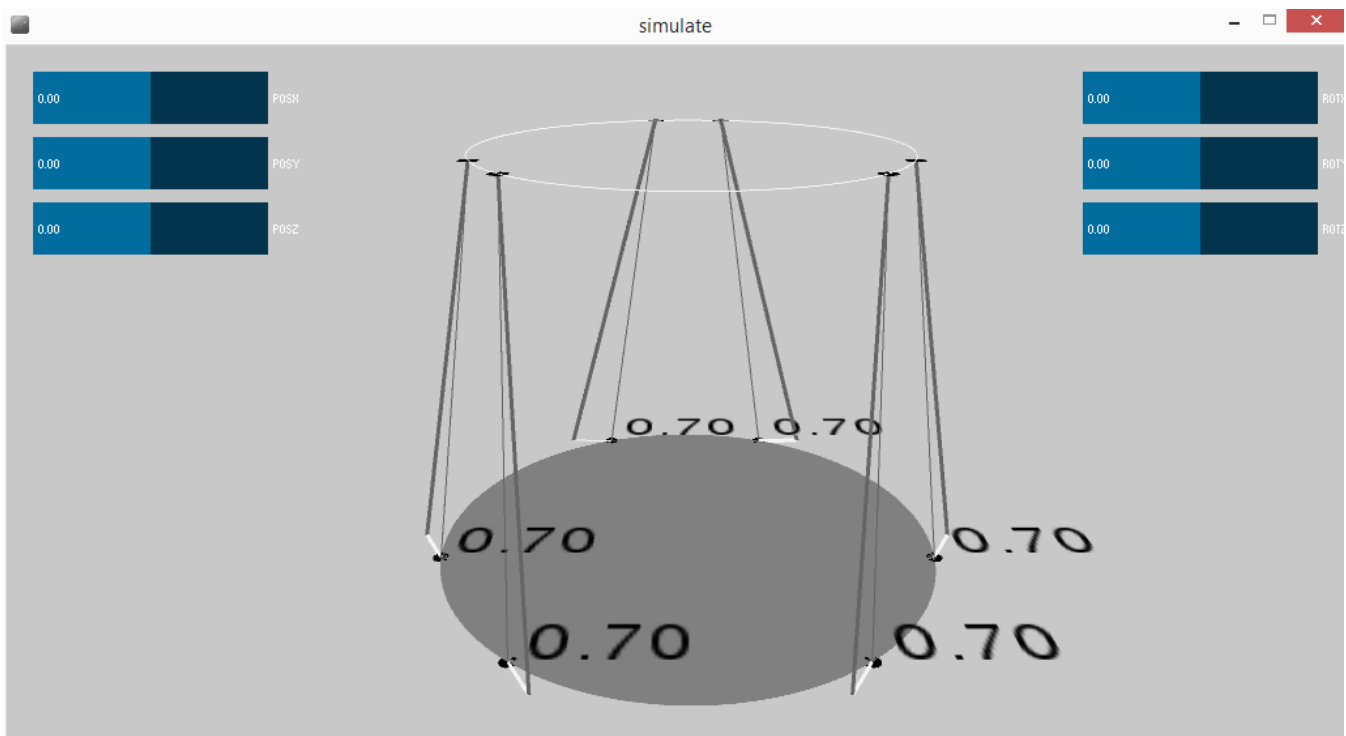
The computer runs two processes, the Graphical User Interface, GUI, which provides feedback to the operator the actual state of the task, and the serial communication from the computer to the microcontroller which controls the Stewart platform. The platform is controlled via a graphical user interface at the PC with the help of Serial and Arduino library in processing.

The Arduino Library for Processing

This library allows you to control an Arduino board from Processing without writing code for the Arduino. Firmata (a firmware) has to be previously uploaded on the board which is available in the arduino package. After installing the firmware, Serial and Arduino library enables the user to simultaneously do computation and communicate with the microcontroller through single sketch in Processing.

A sample code for the simulation of Stewart platform was provided on *Github* by user *thiagoresan*. Modifications after careful understanding and analysis on the sample code were done to ensure compatibility with our prototype of the Stewart platform.

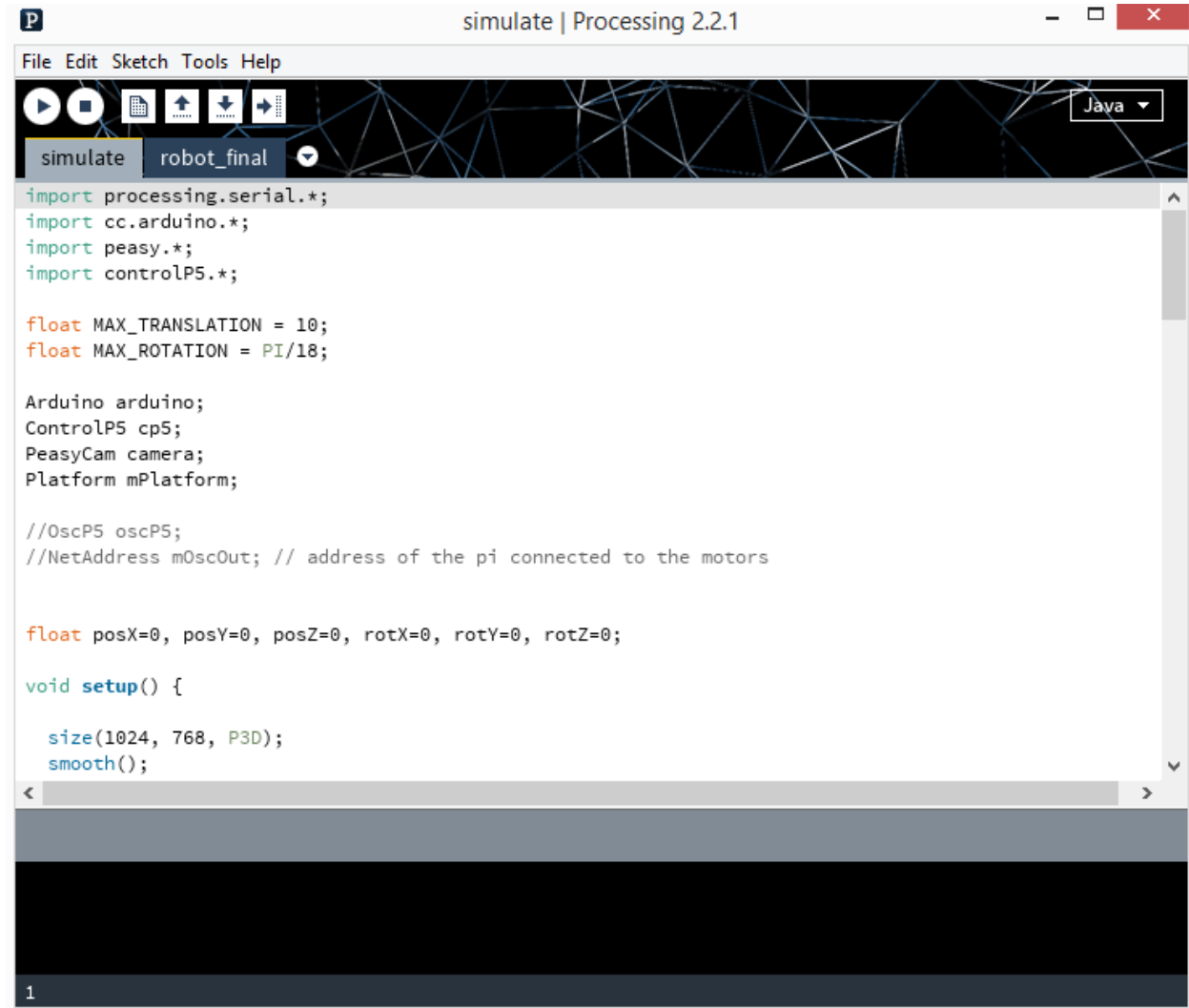
The feature of Graphical User Interface to control the platform was added using the above mentioned libraries.



5.1 PROGRAM STRUCTURE

The program consists mainly of one platform class which holds class which holds the basic parameters of the platform. The draw method of the platform class renders the 3D graphic of the platform which is called continuously by the main function to simulate the movement of the platform. Furthermore there is a method (`getMapAngle()`) in the platform class which returns the values of joint variables to the main function which then implements the communication and movement to the corresponding servos.

Corresponding code for the platform class, simulation and GUI to control the platform has been attached in the appendix.



```
simulate | Processing 2.2.1
File Edit Sketch Tools Help
simulate robot_final
import processing.serial.*;
import cc.arduino.*;
import peasy.*;
import controlP5.*;

float MAX_TRANSLATION = 10;
float MAX_ROTATION = PI/18;

Arduino arduino;
ControlP5 cp5;
PeasyCam camera;
Platform mPlatform;

//OscP5 oscP5;
//NetAddress mOscOut; // address of the pi connected to the motors

float posX=0, posY=0, posZ=0, rotX=0, rotY=0, rotZ=0;

void setup() {
  size(1024, 768, P3D);
  smooth();
}
```

6. PROPOSED APPLICATION

Experimental analysis of heat transfer by forced convection on inclined windward roof by Stewart platform.

6.1 BACKGROUND

The 70 to 80% of the world's total energy consumption is derived from the combustion of fossil fuels in the earth's crust. Geopolitical implications often arises due to concentration of fossil fuel reserves in very few countries. Combining with the ever increasing consumption rate and very low replenishment rate, it is very crucial to research into the development and application of renewable energy sources.

Among all the renewable energy sources, solar energy is available everywhere in the world and the total solar energy received in the Earth's atmosphere in a day is 24 times greater than the Earth's annual energy consumption. It is clean and can power from minuscule devices like watches and calculator to entire home.

6.2 HARNESSING THE ENERGY

A Building Integrated Photovoltaic/Thermal system (BIPV/T) consists of a photovoltaic array with channels underneath. A fluid flows through the channel and recovers incident solar energy as thermal energy which can later be used as water heating or heating homes.

The PV surface absorbs the incoming solar radiation converting a small portion (6 – 10%) to electrical energy and the remaining heat is carried by the fluid flowing in the channels by convection. A thesis paper on convective heat transfer (see Appendix.) represents the energy balance equation of the system as

$$\alpha = E + Q_{C\ in} + Q_{C\ out} + Q_{R\ in} + Q_{R\ out}$$

α	Solar radiation absorbed by the PV panel
E	Electrical energy produced
$Q_{C\ in}$	Convective heat absorbed by fluid in the channel
$Q_{C\ out}$	Convective heat removed by the wind flowing over the panel
$Q_{R\ in}$	Heat removed by the radiation heat transfer from the bottom of the panel
$Q_{R\ out}$	Radiation heat loss from the top surface of the PV panel

6.3 APPLICATION OF THE PLATFORM

The wind-induced convective heat loss over a BIPV/T system mounted on an inclined roof of a low rise building is about 30-50% of the absorbed solar energy (Chen, 2009; Candanedo *et al.*, 2010).

A wind induced Convective Heat Transfer Coefficient (CHTC) for an external surface depends on the wind speed, wind direction, free stream turbulence intensity and integral length scale of the turbulence of the wind, size of the surface and surface roughness since these factors dictate the surface to- air temperature difference and the amount of heat convected away.

(Extract from the thesis report)

We propose the utilization of platform to simulate as a flat plate inclined and yawed at different angles to determine CHTC for different orientations of the platform. The platform could be also used to obtain the most optimum orientation of the solar panels to maximize the incident solar radiation and minimize convective heat loss to the surrounding by the wind.

7. CONCLUSIONS AND FUTURE/FURTHER WORK

A GUI controlled Stewart platform was designed, fabricated and simulated as part of this project. The project began with literary survey of the presently available knowledge on the subject from web blogs, research papers and books. It proceeded by finalizing the design of the platform through CAD model in SolidWorks and decide on the specification of the components. Procurement of the required parts was followed by fabrication and programming. The need of simulation to verify the kinematic analysis resulted in getting familiar with Processing and its libraries. Some problems regarding the amount of data that is exchanged via serial communication between Processing and Arduino were faced but were rectified later by using a single sketch in Processing for computation, simulation and GUI to control the servos was sought. The project on the whole provided a unique opportunity to learn, create and innovate to implement the proposed design in the assigned timeframe.

The proposed plan of work for the project was met with satisfactory results. Further fine refinement in the movement of the platform can be done in future with the help of more accurate servos and rigid links. Platform may be used as an autonomous device which can be controlled by mounting the required sensors and modifying the code. As developers, we wish to see more development in parallel robotics and research to find more feasible applications of the same.

8. APPENDIX

A. Platform class

```
class Platform {
    private PVector translation, rotation, initialHeight;
    private PVector[] baseJoint, platformJoint, q, l, A;
    private float[] alpha;
    private float[] MapAngle;
    private float baseRadius, platformRadius, hornLength, legLength;

    // REAL ANGLES
    private final float baseAngles[] = {
        248.19, 291.81, 8.19, 51.81, 128.19, 171.81 };

    private final float platformAngles[] = {
        259.64, 280.36, 19.64, 40.36, 139.64, 160.36
    };

    private final float beta[] = {
        PI, 2*PI,-PI/3, 2*PI/3,PI/3,4*PI/3};

    // REAL MEASUREMENTS
    private final float SCALE_INITIAL_HEIGHT = 116;
    private final float SCALE_BASE_RADIUS = 78.05;
    private final float SCALE_PLATFORM_RADIUS = 55.62;
    private final float SCALE_HORN_LENGTH = 15;
    private final float SCALE_LEG_LENGTH = 122;

    public Platform(float s) {
        translation = new PVector();
        initialHeight = new PVector(0, 0, s*SCALE_INITIAL_HEIGHT);
        rotation = new PVector();
        baseJoint = new PVector[6];
        platformJoint = new PVector[6];
        alpha = new float[6];
        MapAngle = new float[6];
        q = new PVector[6];
        l = new PVector[6];
        A = new PVector[6];
        baseRadius = s*SCALE_BASE_RADIUS;
        platformRadius = s*SCALE_PLATFORM_RADIUS;
        hornLength = s*SCALE_HORN_LENGTH;
        legLength = s*SCALE_LEG_LENGTH;

        for (int i=0; i<6; i++) {
```

```

float mx = baseRadius*cos(radians(baseAngles[i]));
float my = baseRadius*sin(radians(baseAngles[i]));
baseJoint[i] = new PVector(mx, my, 0);
println(baseJoint[i]);
}

for (int i=0; i<6; i++) {
float mx = platformRadius*cos(radians(platformAngles[i]));
float my = platformRadius*sin(radians(platformAngles[i]));

platformJoint[i] = new PVector(mx, my, 0);
println(platformJoint[i]);
q[i] = new PVector(0, 0, 0);
l[i] = new PVector(0, 0, 0);
A[i] = new PVector(0, 0, 0);
}
calcQ();
}

public void applyTranslationAndRotation(PVector t, PVector r) {
rotation.set(r);
translation.set(t);
calcQ();
calcAlpha();
}

private void calcQ() {
for (int i=0; i<6; i++) {
// rotation
q[i].x = cos(rotation.z)*cos(rotation.y)*platformJoint[i].x +
(-sin(rotation.z)*cos(rotation.x)+cos(rotation.z)*sin(rotation.y)*sin(rotation.x))*platformJoint[i].y +
(sin(rotation.z)*sin(rotation.x)+cos(rotation.z)*sin(rotation.y)*cos(rotation.x))*platformJoint[i].z;

q[i].y = sin(rotation.z)*cos(rotation.y)*platformJoint[i].x +
(cos(rotation.z)*cos(rotation.x)+sin(rotation.z)*sin(rotation.y)*sin(rotation.x))*platformJoint[i].y +
(-cos(rotation.z)*sin(rotation.x)+sin(rotation.z)*sin(rotation.y)*cos(rotation.x))*platformJoint[i].z;

q[i].z = -sin(rotation.y)*platformJoint[i].x +
cos(rotation.y)*sin(rotation.x)*platformJoint[i].y +
cos(rotation.y)*cos(rotation.x)*platformJoint[i].z;

// translation
q[i].add(PVector.add(translation, initialHeight));
// println("Q" + i + q[i]);
l[i] = PVector.sub(q[i], baseJoint[i]);
//println("L" + i + l[i]);
}
}

```

```

}

private void calcAlpha() {
    for (int i=0; i<6; i++) {
        float L = l[i].magSq()-(legLength*legLength)+(hornLength*hornLength);
        float M = 2*hornLength*(q[i].z-baseJoint[i].z);
        float N = 2*hornLength*(cos(beta[i])*(q[i].x-baseJoint[i].x) + sin(beta[i])*(q[i].y-baseJoint[i].y));
        alpha[i] = (asin(L/sqrt(M*M+N*N)) - atan2(N, M));

        A[i].set(hornLength*cos(alpha[i])*cos(beta[i]) + baseJoint[i].x,
        hornLength*cos(alpha[i])*sin(beta[i]) + baseJoint[i].y,
        hornLength*sin(alpha[i]) + baseJoint[i].z);

        float xqxb = (q[i].x-baseJoint[i].x);
        float yqyb = (q[i].y-baseJoint[i].y);
        float h0 = sqrt((legLength*legLength)+(hornLength*hornLength)-(xqxb*xqxb)-(yqyb*yqyb)) - q[i].z;

        float L0 = 2*hornLength*hornLength;
        float M0 = 2*hornLength*(h0+q[i].z);
        float a0 = asin(L0/sqrt(M0*M0+N*N)) - atan2(N, M0);

        // println(i+": "+alpha[i]);
    }
    for(int m=0;m<6;m++)
    {
        if(m%2==0)
        {
            MapAngle[m]=degrees(alpha[m]+radians(80));
        }
        else
        {
            MapAngle[m]=degrees(-1*alpha[m]+radians(100));
        }
    }
}

public float[] getAlpha(){
    return alpha;
}

public float[] getMapAngle(){
    return MapAngle;
}

public void draw() {
    // draw Base
    noStroke();
    fill(128);

```

```

ellipse(0, 0, 2*baseRadius, 2*baseRadius);
for (int i=0; i<6; i++) {
  pushMatrix();
  translate(baseJoint[i].x, baseJoint[i].y, baseJoint[i].z);
  noStroke();
  fill(0);
  ellipse(0, 0, 5, 5);
  text(String.format("%.2f", degrees(alpha[i])), 5,5,5);
  popMatrix();

  stroke(245);
  line(baseJoint[i].x, baseJoint[i].y, baseJoint[i].z, A[i].x, A[i].y, A[i].z);

  PVector rod = PVector.sub(q[i], A[i]);
  rod.setMag(legLength);
  rod.add(A[i]);

  stroke(100);
  strokeWeight(3);
  line(A[i].x, A[i].y, A[i].z, rod.x, rod.y, rod.z);
}

// draw phone jointss and rods
for (int i=0; i<6; i++) {
  pushMatrix();
  translate(q[i].x, q[i].y, q[i].z);
  noStroke();
  fill(0);
  ellipse(0, 0, 5, 5);
  popMatrix();

  stroke(100);
  strokeWeight(1);
  line(baseJoint[i].x, baseJoint[i].y, baseJoint[i].z, q[i].x, q[i].y, q[i].z);
}

// sanity check
pushMatrix();
translate(initialHeight.x, initialHeight.y, initialHeight.z);
translate(translation.x, translation.y, translation.z);
rotateZ(rotation.z);
rotateY(rotation.y);
rotateX(rotation.x);
stroke(245);
noFill();
ellipse(0, 0, 2*platformRadius, 2*platformRadius);
popMatrix();}}

```


B. Simulation and GUI

```
import processing.serial.*;
import cc.arduino.*;
import peasy.*;
import controlP5.*;

float MAX_TRANSLATION = 15;
float MAX_ROTATION = PI/18;

Arduino arduino;
ControlP5 cp5;
PeasyCam camera;
Platform mPlatform;

float posX=0, posY=0, posZ=0, rotX=0, rotY=0, rotZ=0;

void setup() {
  arduino = new Arduino(this, Arduino.list()[0], 57600);
  arduino.pinMode(2, Arduino.SERVO);
  arduino.pinMode(3, Arduino.SERVO);
  arduino.pinMode(4, Arduino.SERVO);
  arduino.pinMode(5, Arduino.SERVO);
  arduino.pinMode(6, Arduino.SERVO);
  arduino.pinMode(7, Arduino.SERVO);*/
  size(1024, 768, P3D);
  smooth();

  textSize(20);

  camera = new PeasyCam(this, 666);
  camera.setRotations(-1.0, 0.0, 0.0);
  camera.lookAt(8.0, -50.0, 80.0);

  mPlatform = new Platform(1);
  mPlatform.applyTranslationAndRotation(new PVector(), new PVector());

  cp5 = new ControlP5(this);

  cp5.addSlider("posX")
    .setPosition(20, 20)
    .setSize(180, 40).setRange(-1, 1);
  cp5.addSlider("posY")
    .setPosition(20, 70)
    .setSize(180, 40).setRange(-1, 1);
```

```

cp5.addSlider("posZ")
    .setPosition(20, 120)
    .setSize(180, 40).setRange(-1, 1);

cp5.addSlider("rotX")
    .setPosition(width-200, 20)
    .setSize(180, 40).setRange(-1, 1);
cp5.addSlider("rotY")
    .setPosition(width-200, 70)
    .setSize(180, 40).setRange(-1, 1);
cp5.addSlider("rotZ")
    .setPosition(width-200, 120)
    .setSize(180, 40).setRange(-1, 1);

cp5.setAutoDraw(false);
camera.setActive(true);
}

void draw() {
    background(200);
    mPlatform.applyTranslationAndRotation(PVector.mult(new PVector(posX, posY, posZ),
MAX_TRANSLATION),
PVector.mult(new PVector(rotX, rotY, rotZ), MAX_ROTATION));
    mPlatform.draw();
    float[] map = mPlatform.getMapAngle();
    for(int j=0;j<6;j++)
    {
        arduino.servoWrite(j+2,int(map[j]));
        delay(10);
    }
    hint(DISABLE_DEPTH_TEST);
    camera.beginHUD();
    cp5.draw();
    camera.endHUD();
    hint(ENABLE_DEPTH_TEST);
}

void controlEvent(ControlEvent theEvent) {
    camera.setActive(false);

    //after a UI event send a OSC package
    float[] angles = mPlatform.getAlpha();

    for (float f : angles) {
        if(Float.isNaN(f)){
            return;
        }
    }
}

```

```
}

}
void mouseReleased() {
    camera.setActive(true);
}

void keyPressed() {
    if (key == &apos; &apos;) {
        camera.setRotations(-1.0, 0.0, 0.0);
        camera.lookAt(8.0, -50.0, 80.0);
        camera.setDistance(666);
    }
}
```

9. BIBLIOGRAPHY

1. Stewart platform with fixed rotary actuators: A low cost design study - Filip Szufnarowski
2. Detecting Singularities of Stewarts Platform - T. Charters, R. Enguica, P. Freitas
3. The Mathematics of Stewarts Platform Wokingham U3A Math Group
4. www.processing.org
5. www.arduino.cc
6. <https://github.com/ThomasKNR/RotaryStewartPlatform>
7. <https://github.com/thiagohersan/memememe/tree/master/Processing/StewartSimulator>