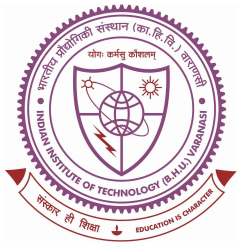
Ball-Plate Balancing System with Stewart Platform

B.Tech Project Presentation (Sem-VII)



Supervisor: Dr. Shyam Kamal
Department of Electrical Engineering
Indian Institute of Technology (BHU) Varanasi, India

Problem Statement

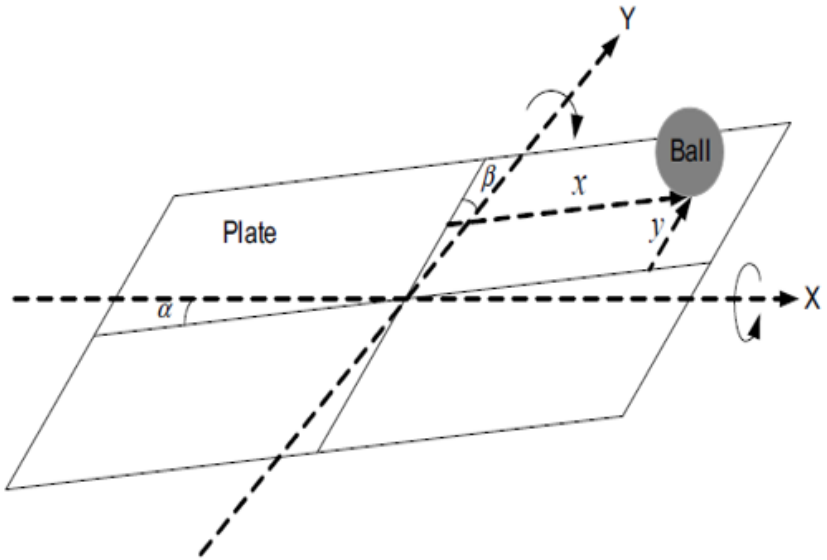
Fixed Position Control of a Ball on a Plate using Stewart Platform as an Actuator.

To Implement this Model, we have used Two Major Components:

- Ball-Plate System
- Stewart Platform

Ball Plate System

Mathematical Modelling



- Euler Lagrange equation : $\frac{d}{dt}\frac{\partial T}{\partial q_i} - \frac{\partial T}{\partial q_i} + \frac{\partial V}{\partial q_i} = Q_i$

T - Kinetic energy of the system
V - Potential energy of the system
Q - Composite force on the system

Mathematical Modelling

The kinetic energy of ball consists of its both rotational with respect to its centre of mass and transnational energy:

$$T_b = \frac{1}{2} m_b (\dot{x}_b^2 + \dot{y}_b^2) + \frac{1}{2} I_b (\omega_x^2 + \omega_y^2) \tag{1}$$

$$\dot{x}_b = r_b \omega_y, \dot{y}_b = r_b \omega_x \tag{2}$$

$$\begin{aligned} T_b &= \frac{1}{2} [m_b (\dot{x}_b^2 + \dot{y}_b^2) + \frac{I_b}{r_b^2} (\dot{x}_b^2 + \dot{y}_b^2)] \\ &= \frac{1}{2} (m_b + \frac{I_b}{r_b^2}) (\dot{x}_b^2 + \dot{y}_b^2) \end{aligned} \tag{3}$$

Mathematical Modelling

- The plate's kinematic energy is given be the following relation:

$$T_p = \frac{1}{2}(I_p + I_b)\left(\dot{\alpha}^2 + \dot{\beta}^2\right) + \frac{1}{2}m_b\left(x_b\dot{\alpha} + y_b\dot{\beta}\right)^2$$
$$= \frac{1}{2}(I_p + I_b)\left(\dot{\alpha}^2 + \dot{\beta}^2\right) + \frac{1}{2}m_b\left(x_b^2\dot{\alpha}^2 + 2x_b\dot{\alpha}y_b\dot{\beta} + y_b^2\dot{\beta}^2\right)$$

- The total kinetic energy of the system can be expressed as follows:

$$T = T_b + T_p$$
$$= \frac{1}{2}\left(m_b + \frac{I_b}{r_b^2}\right)(\dot{x}_b^2 + \dot{y}_b^2) + \frac{1}{2}(I_p + I_b)\left(\dot{\alpha}^2 + \dot{\beta}^2\right)$$
$$+ \frac{1}{2}m_b\left(x_b^2\dot{\alpha}^2 + 2x_b\dot{\alpha}y_b\dot{\beta} + y_b^2\dot{\beta}^2\right)$$

Mathematical Modelling

- The ball's potential energy can be expressed as:

$$V_b = m_b g h = m_b g (x_b sin\alpha + y_b sin\beta) \tag{4}$$

- The Lagrangian function of the system is represented as:

$$L = T_b + T_p - V_b$$

- With Euler Lagrange equation, we get the non-linear differential equations for the ball-plate-system as :

$$\left(m_b + \frac{I_b}{r_b^2}\right) \ddot{x}_b - m_b \left(x_b \dot{\alpha}^2 + y_b \dot{\alpha}\dot{\beta}\right) + m_b g \sin\alpha = 0$$

$$\left(m_b + \frac{I_b}{r_b^2}\right) \ddot{y}_b - m_b \left(y_b \dot{\beta}^2 + x_b \dot{\alpha}\dot{\beta}\right) + m_b g \sin\beta = 0$$

Model Linearisation

- Moment of Inertia of a solid ball is given as $I_b = \frac{2}{5} m_b r_b^2$

$$m_b \left[\frac{5}{7} \ddot{x}_b - \left(x_b \dot{\alpha}^2 + y_b \dot{\alpha} \dot{\beta} \right) + g \sin \alpha \right] = 0$$

$$m_b \left[\frac{5}{7} \ddot{y}_b - \left(y_b \dot{\beta}^2 + x_b \dot{\alpha} \dot{\beta} \right) + g \sin \beta \right] = 0$$

- We linearize the non-linear ball-plate mathematical model by making the following assumptions:
  - The inclination of the plate would be small (up to 5 degrees): $\alpha \ll 1$ and $\beta \ll 1$, therefore $\sin \alpha \approx \alpha$ and $\sin \beta \approx \beta$.
  - The rate of change of inclination would be slow, therefore $\dot{\alpha}^2 \approx 0$, $\dot{\beta}^2 \approx 0$, and $\dot{\beta}\dot{\alpha} \approx 0$.

Model Linearisation

- So, the final linearised equations are :

$$\frac{5}{7}\ddot{x}_b + g\alpha = 0 \quad \& \quad \frac{5}{7}\ddot{y}_b + g\beta = 0$$

- By assuming $\alpha(s)$ and $\beta(s)$ as inputs to ball-on-plate system, we find the transfer functions:

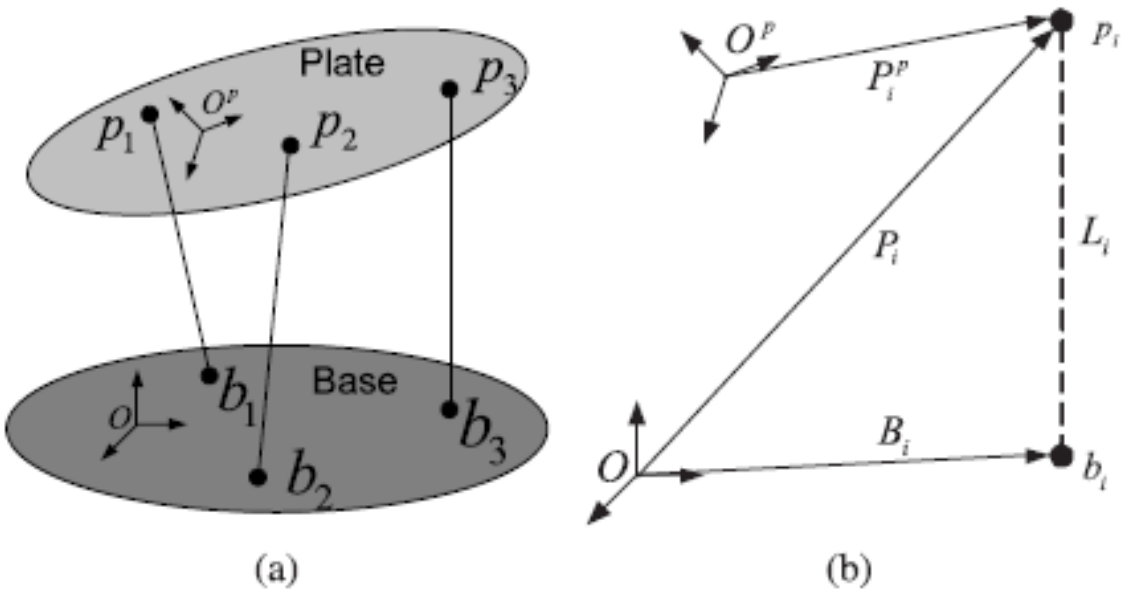
$$P_x(s) = \frac{X_b(s)}{\alpha(s)} = \frac{g}{\frac{5}{7}s^2}$$

$$P_y(s) = \frac{Y_b(s)}{\beta(s)} = \frac{g}{\frac{5}{7}s^2}$$

Stewart Platform

Stewart Platform is a type of mechanical system having 6 DOF that consists of six adjustable legs actuated with servo motors, connected to a base plate and a top plate.

- Stewart Platform representation

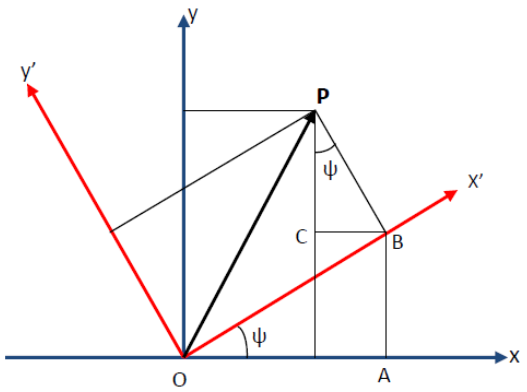


Inverse Kinematics Solution of Stewart Platform

Calculation of Leg Length

- $R = R_z(\psi)R_y(\phi)R_x(\theta)$
  where $R_x(\theta)$, $R_y(\phi)$, and $R_z(\psi)$ are the rotation matrices for rotations around the x, y, and z axes, respectively.

- $$R(\phi,\theta,\psi) = \begin{bmatrix} C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi s_\psi \\ C_\theta S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix}$$
  where S and C here means sin() and cos() of their respective subscript angles.



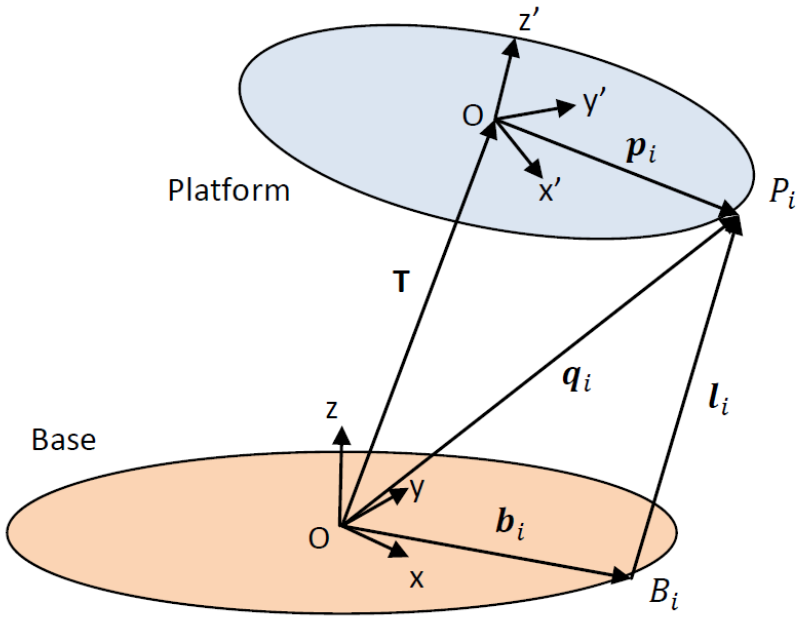
Calculation of Leg-Length

$$q_i = Rp_i + T$$

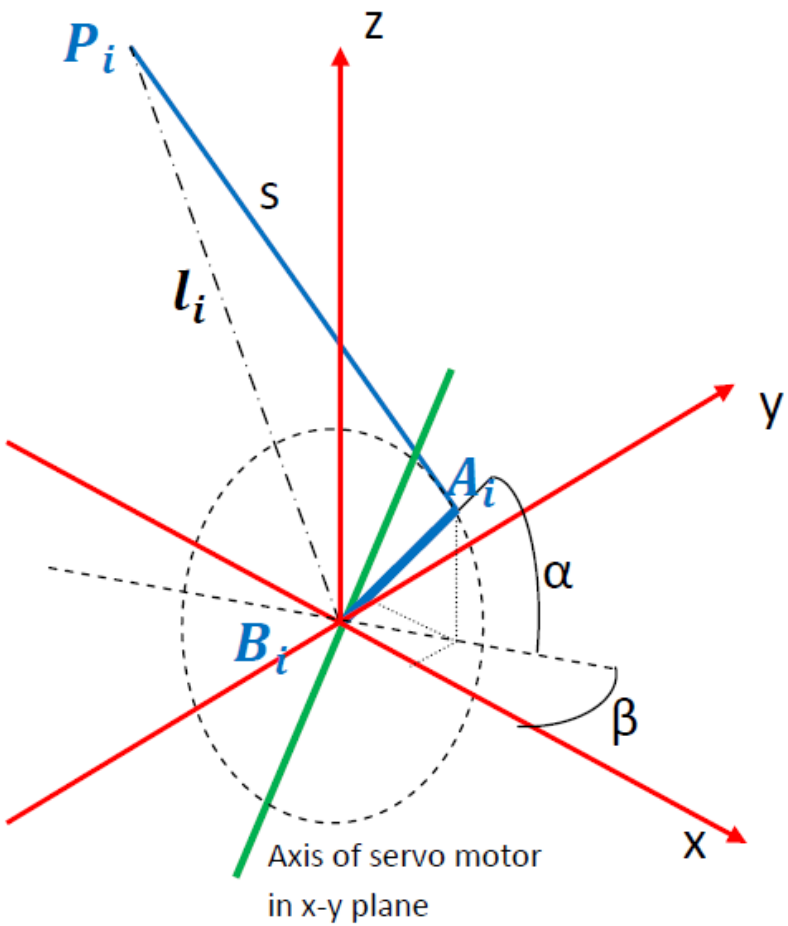
$$b_i + l_i = q_i$$

$$l_i = q_i - b_i$$

$$l_i = Rp_i + T - b_i$$



Calculation of Servo Motor Angle from Leg Length



- Let P_i be $[x_p, y_p, z_p]$, B_i be $[x_b, y_b, z_b]$, and the point where the connecting rod is attached to the servo motor arm be $[x_a, y_a, z_a]$.
- $a^2 = (x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2$
- $l^2 = (x_p - x_b)^2 + (y_p - y_b)^2 + (z_p - z_b)^2$
- $s^2 = (x_p - x_a)^2 + (y_p - y_a)^2 + (z_p - z_a)^2$

Calculation of Servo Motor Angle from Leg Length

$$x_a = R\cos\alpha\cos\beta + x_b$$
$$y_a = R\cos\alpha\sin\beta + y_b$$
$$z_a = R\sin\alpha + z_b$$

- $l^2-(s^2-a^2) = 2Rsin\alpha(z_p-z_b)+2Rcos\alpha[cos\beta(x_p-x_b)+sin\beta(y_p-y_b)]$
- The equation is of the following form:

$$S = Tsin\alpha + Ucos\alpha \tag{5}$$

Using the following trigonometric identity:

$$a\sin x + b\cos x = c\sin(x+v) \tag{6}$$

where

$$c = \sqrt{a^2+b^2} \tag{7}$$

$$\tan v = \frac{b}{a} \tag{8}$$

Calculation of Servo Motor Angle from Leg Length

Using the above trigonometric identity, equation can be written as:

$$S = \sqrt{T^2 + U^2}\sin(\alpha + \delta) \quad \text{where} \quad \delta = \arctan\frac{U}{T} \tag{9}$$

$$sin(\alpha + \delta) = \frac{S}{\sqrt{T^2 + U^2}} \tag{10}$$

$$\alpha = \arcsin\frac{S}{\sqrt{T^2 + U^2}} - \arctan\frac{U}{T} \tag{11}$$

where

$$S = l^2 - (s^2 - a^2) \tag{12}$$

$$T = 2R(z_p - z_b) \tag{13}$$

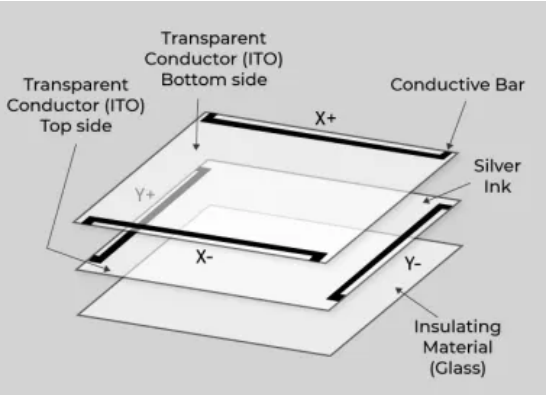
$$U = 2R(cos\beta(x_p - x_b) + sin\beta(y_p - y_b)) \tag{14}$$

Apparatus Used

Hardware

- Resistive Touch Plate
- Rotary Actuators : DS3218 6V 20Kg Digital Servo
- IMU (Inertial Measurement Unit)
- Arduino UNO

Resistive Touch Plate



It is a Position-Sensitive Detector. When the Plate is touched at any point, there is a change in the Resistance of the Material of the surrounding area. The Circuit detects the change in the Resistance and identifies the Point of Contact.

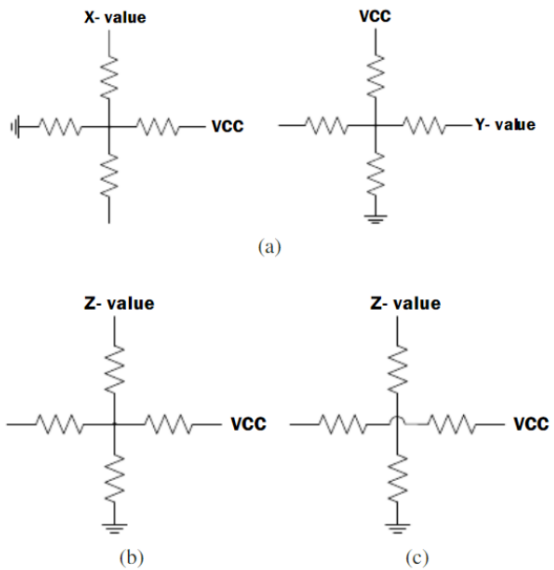
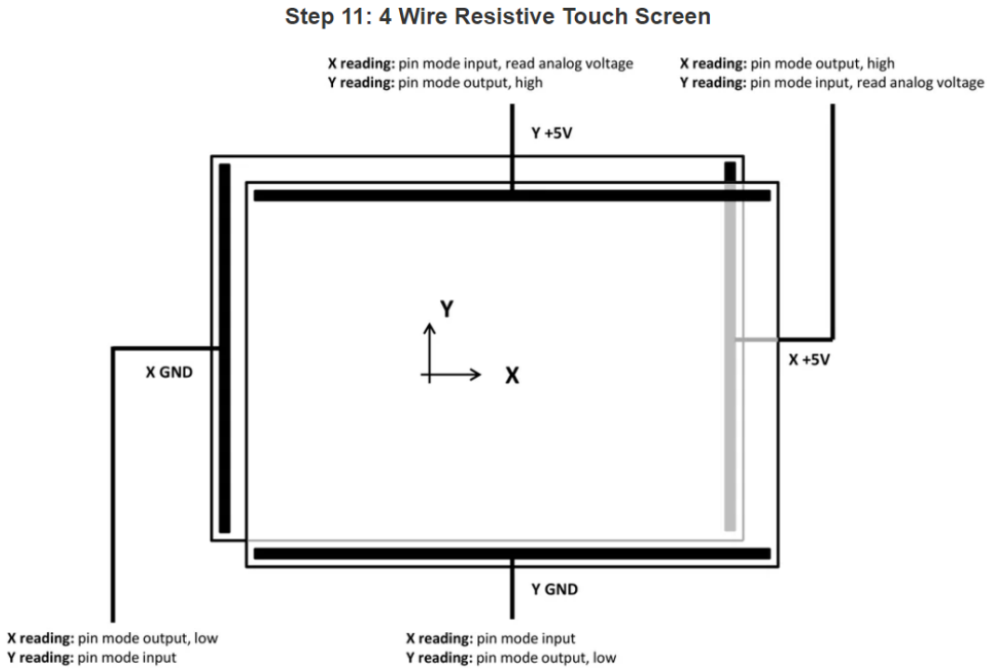


Fig. 5: Diagram of reading ADC values: (a) x- and y- values, (b) z- value when panel is touched, (c) z- value when panel is not touched.

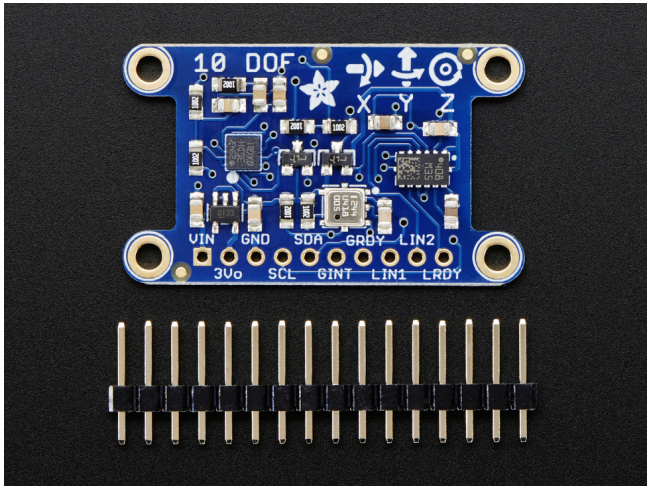


Rotary Actuators : DS3218 6V 20Kg Digital Servo

A Rotary Actuator is a Type of Motor that converts an Electrical Signal into Rotational Motion. The DS3218 6V 20Kg Digital Servo is a High-Torque Servo Motor that can rotate up to 360 degrees.

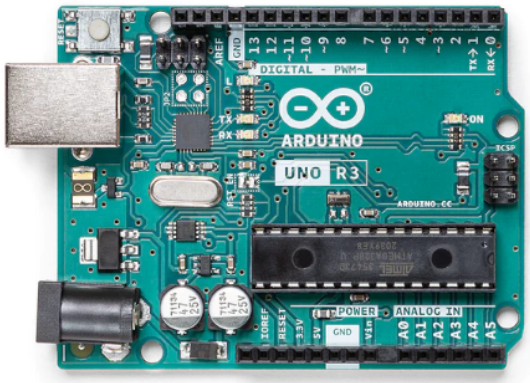
- Servo Motors have Three Wires: (Wire - color - Pin Connection)
  Power - Red - 5V Pin
  Ground - Black or Brown - GND Pin
  Signal - Yellow - PWM $(\sim)Pin$.

IMU



- We are using a 10 DOF IMU. It is a combination of Accelerometer, Gyroscope, Magnetometer and Barometer.
- Here IMU is used to verify the Roll, Pitch, and Yaw of the Plate after each Iteration.

Arduino UNO



- Micro-Controller Board comprising 14 Digital Input/Output Pins, 6 Analog Input Pins, a 16 MHz Quartz Crystal, a USB Connection, a Power Jack, and a Reset button.
- The Sensed Parameters are sent to Arduino and they are used in the Code for Evaluating Servo Angles.

Procedure

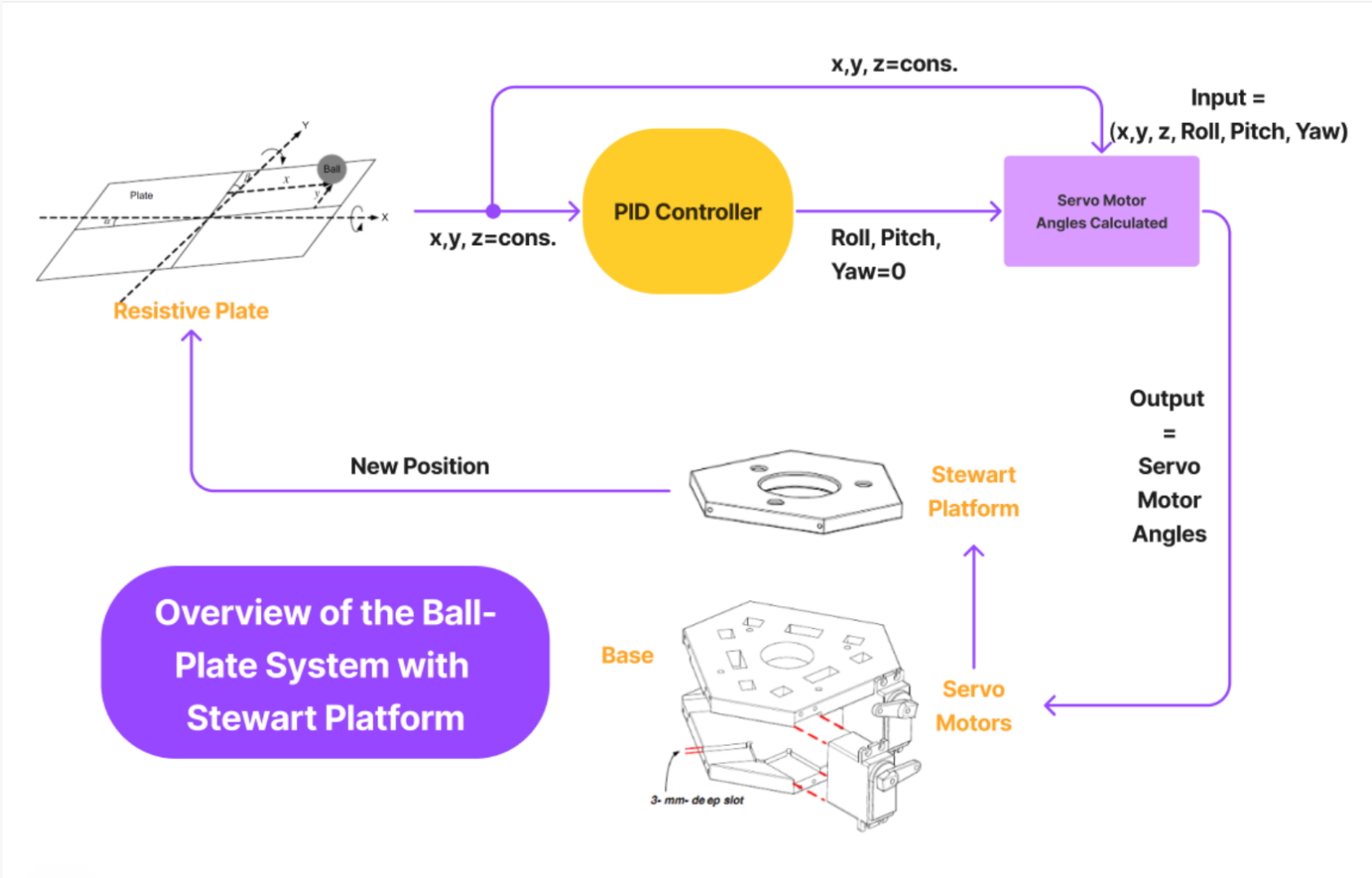

x,y, z=cons.
Input =
(x,y, z, Roll, Pitch, Yaw)
Plate
Ball
Resistive Plate
x,y, z=cons.
PID Controller
Roll, Pitch,
Yaw=0
Servo Motor
Angles Calculated
Output
=
Servo
Motor
Angles
New Position
Stewart
Platform
Overview of the Ball-Plate System with Stewart Platform
Base
Servo
Motors
3- mm- de ep slot

Plate Coordinates (x,y) from Resistive Touch Plate

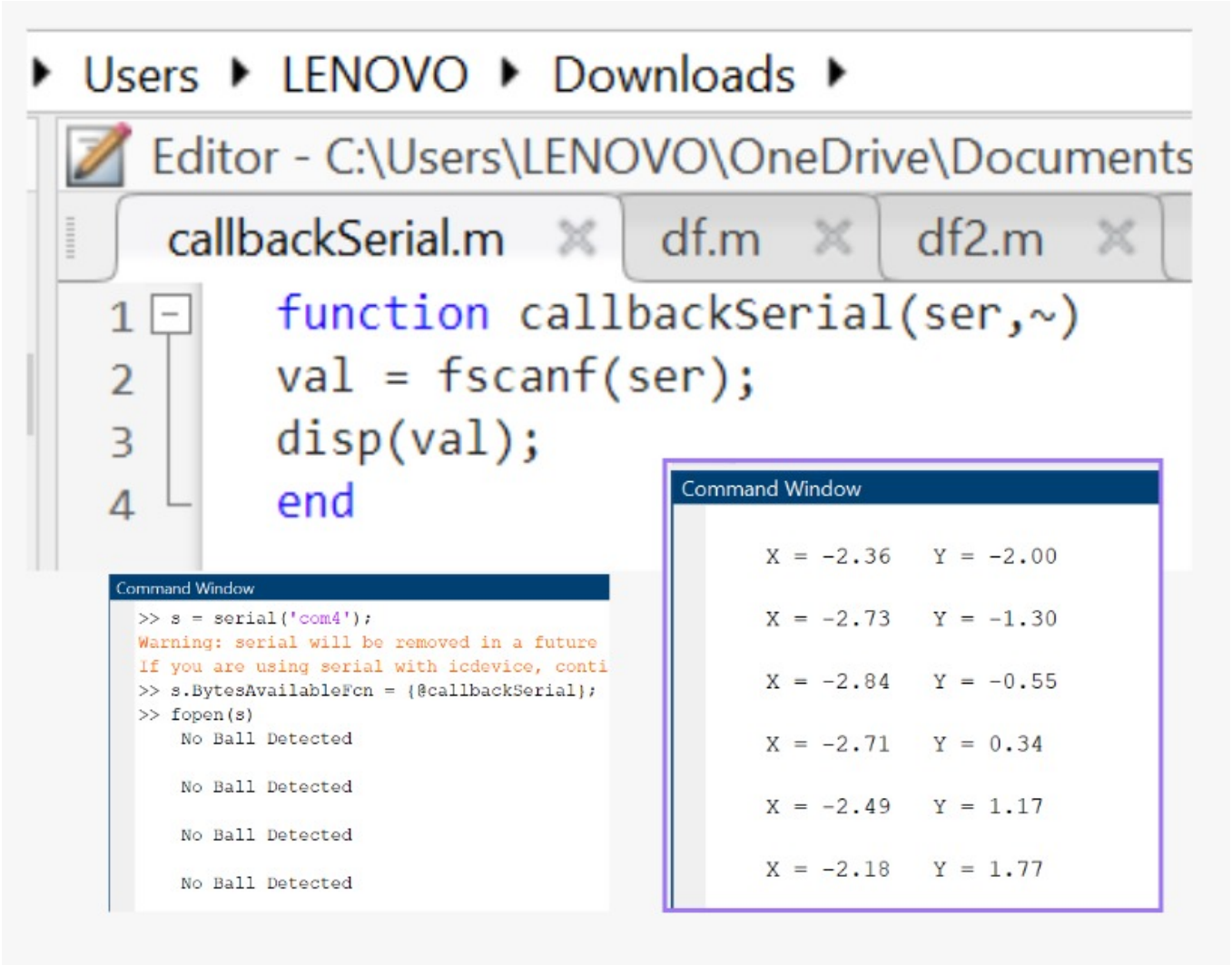
```
TouchScreen ts = TouchScreen(XP, YP, XM, YM,1);

void setup() {
  Serial.begin(9600);
}

void loop() {
  // a point object holds x y and z coordinates
  TSPoint p = ts.getPoint();
```

```
X = -0.16       Y = -1.30
X = 0.33        Y = -1.43
X = 0.56        Y = -1.48
X = 0.47        Y = -1.38
X = 0.27        Y = -1.27
X = 0.22        Y = -1.40
No Ball Detected
No Ball Detected
No Ball Detected
No Ball Detected
No Ball Detected
No Ball Detected
No Ball Detected
```

```
double XX = 11-(p.y/(900.000/20.000));
double YY = 13-(p.x/(1000.000/26.000));

if ((XX<-4.0) && (XX>-6.5) && (YY>12.5) && (YY<13.5)) {
  Serial.print("\tNo Ball Detected");
  Serial.print('\n');
}

else {
 Serial.print("\tX = "); Serial.print(XX);
 Serial.print("\tY = "); Serial.print(YY);
 Serial.print('\n');
```

```
X = -0.11       Y = 3.82
X = -1.07       Y = 3.28
X = -2.04       Y = 2.60
X = -2.78       Y = 1.87
X = -3.31       Y = 1.01
X = -3.60       Y = 0.03
X = -3.69       Y = -0.96
X = -3.49       Y = -1.74
X = -3.09       Y = -2.42
X = -2.53       Y = -2.99
X = -1.91       Y = -3.51
X = -1.29       Y = -3.95
```

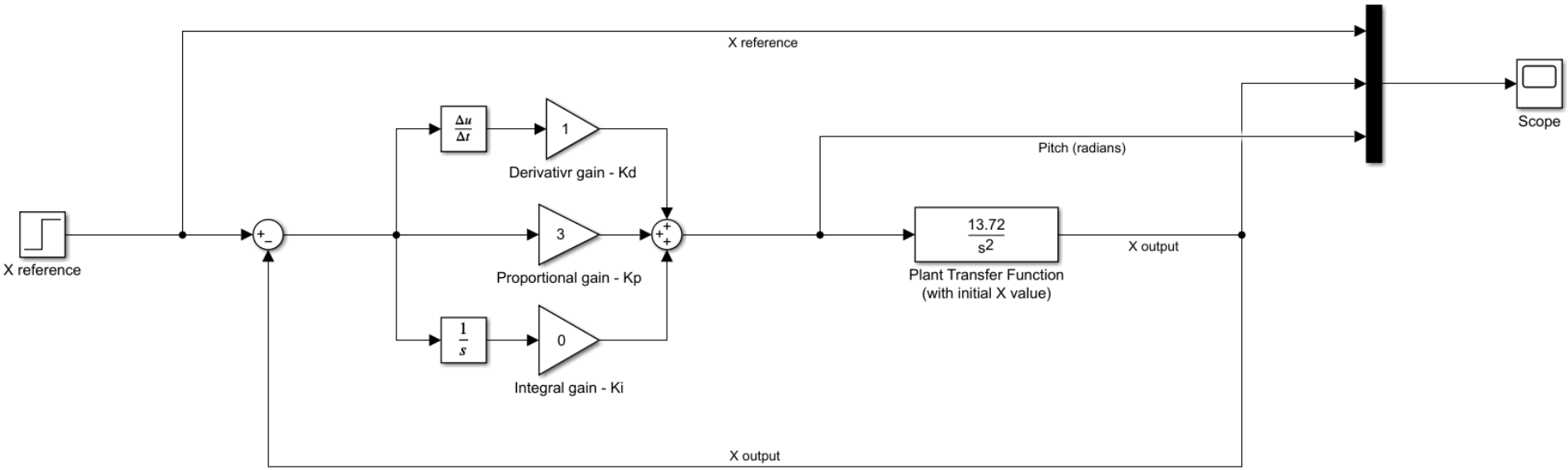
Arduino-MATLAB Serial Communication



Controller Implementation

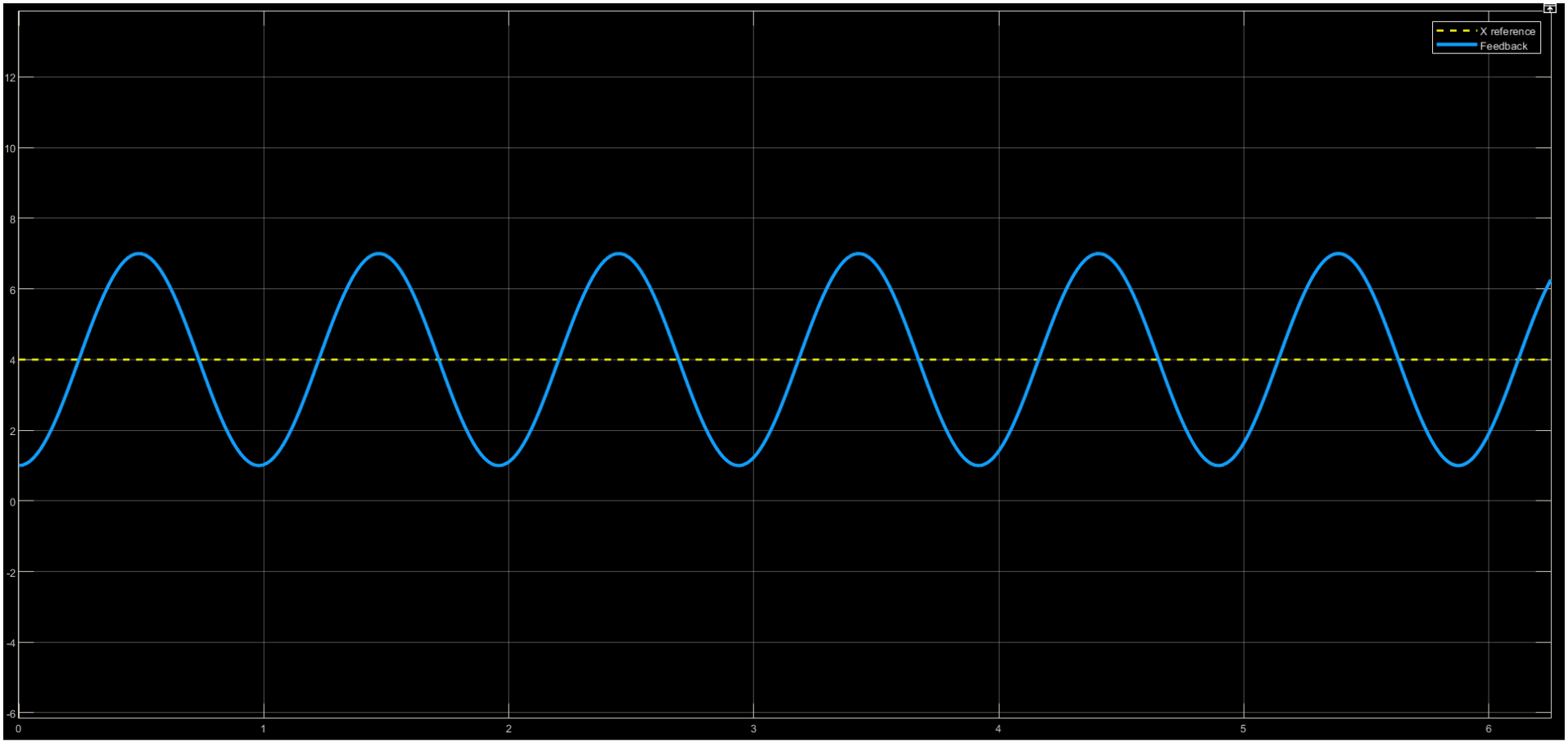
- PID controller is implemented for the tracking problem by considering the system transfer function as

$$P_x(s) = \frac{X_b(s)}{\alpha(s)} = \frac{g}{\frac{5}{7}s^2}$$



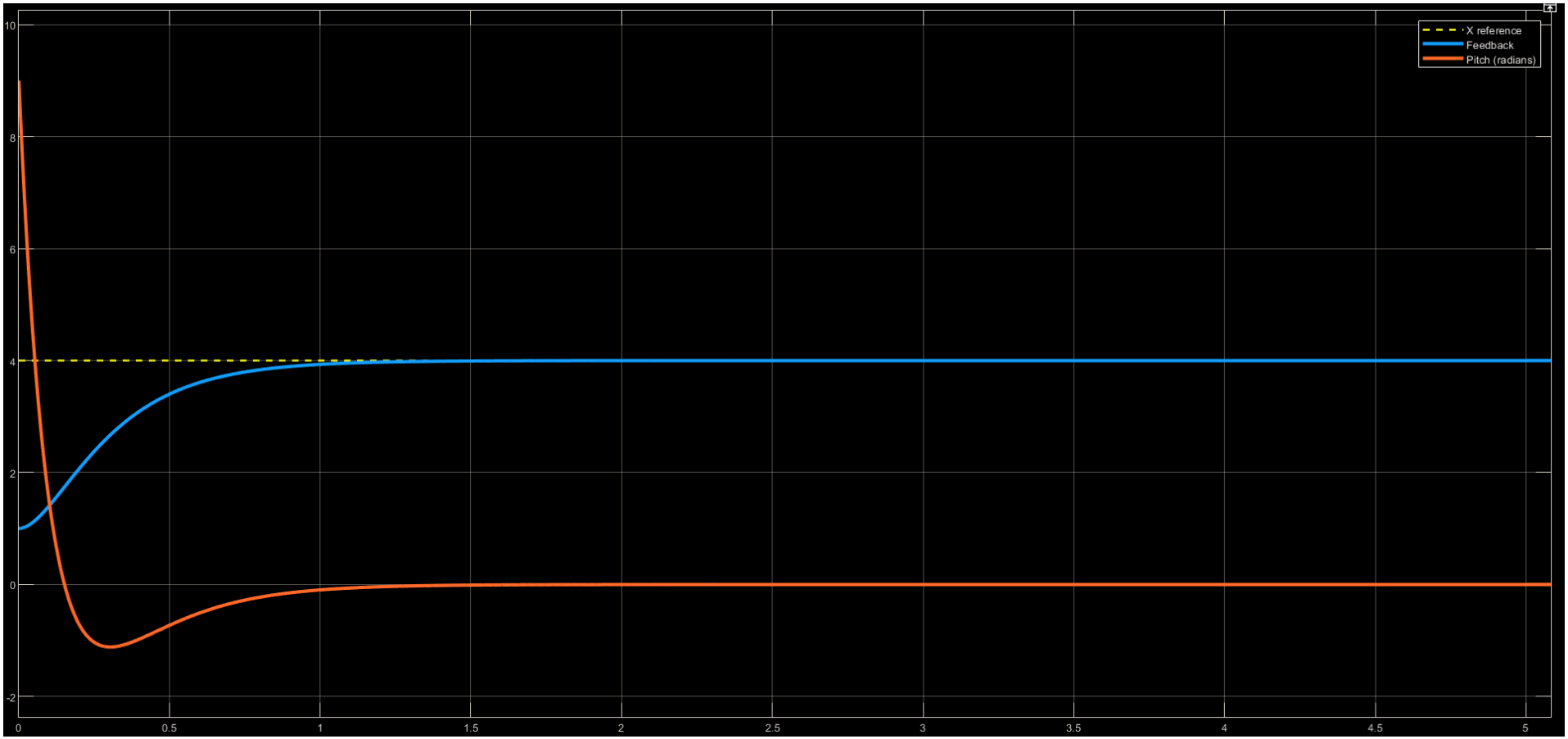
Simulation Results obtained by tuning parameters

- $K_p = 3$



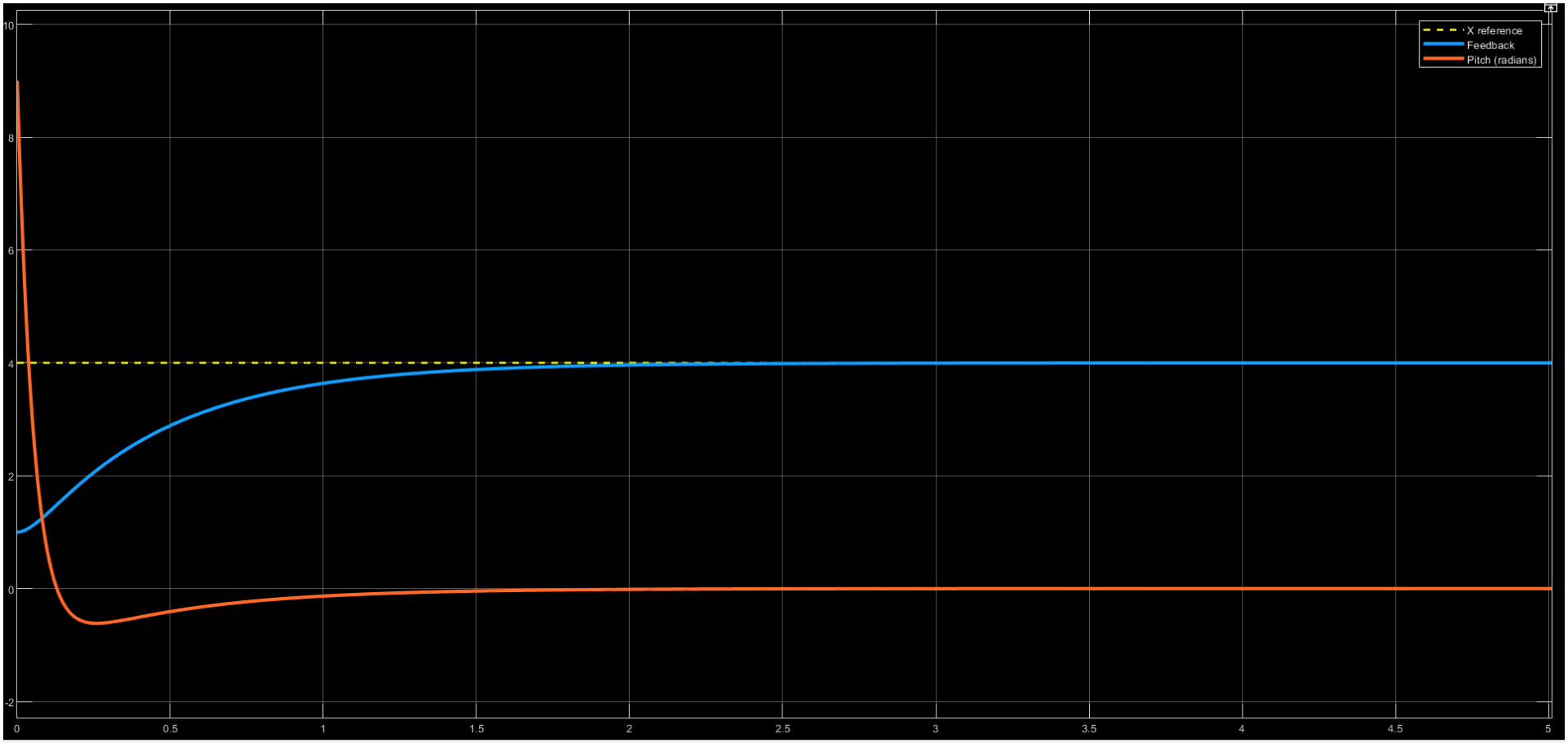
Simulation Results obtained by tuning parameters

- $K_p = 3, K_d = 1$



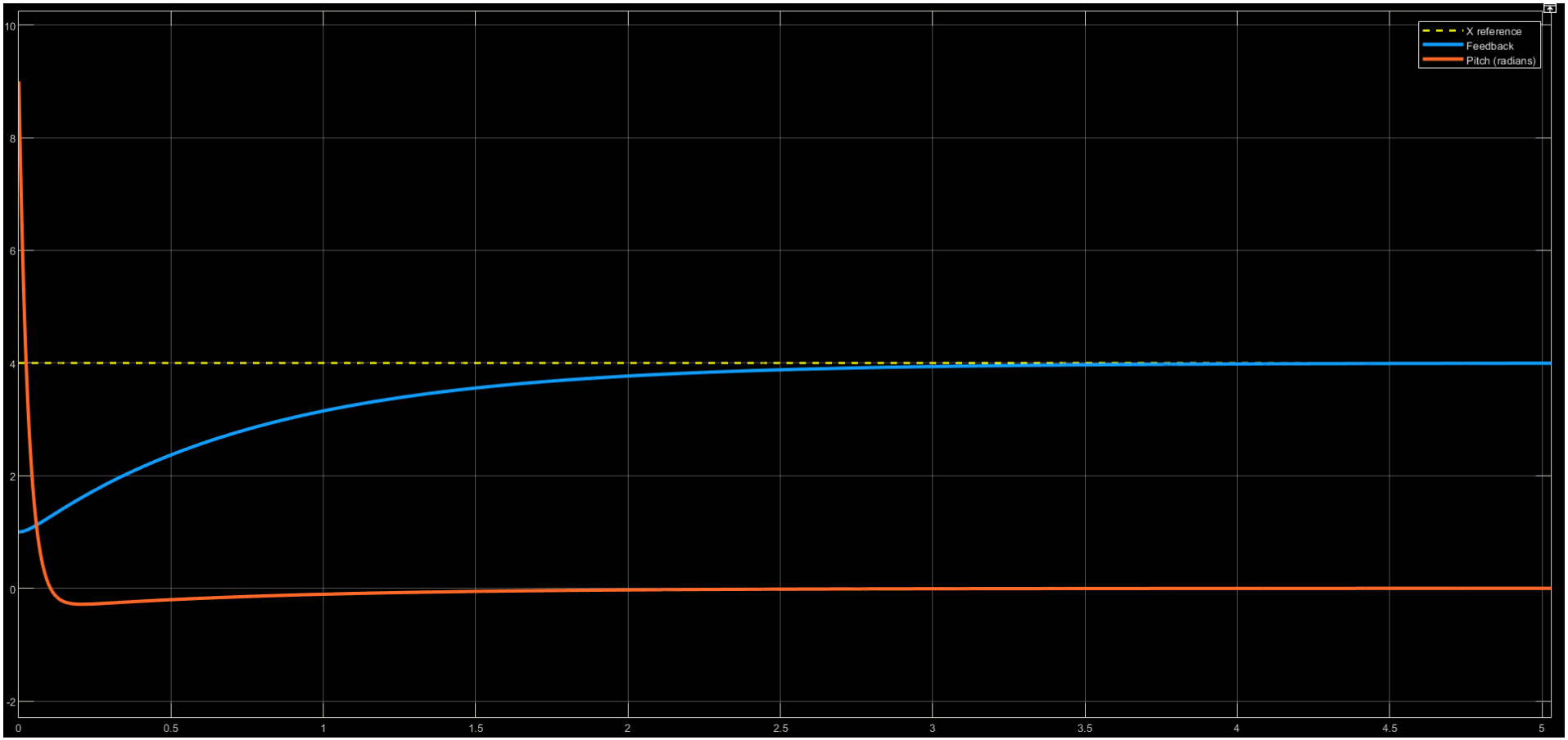
Simulation Results obtained by tuning parameters

- $K_p = 3, K_d = 1.5$



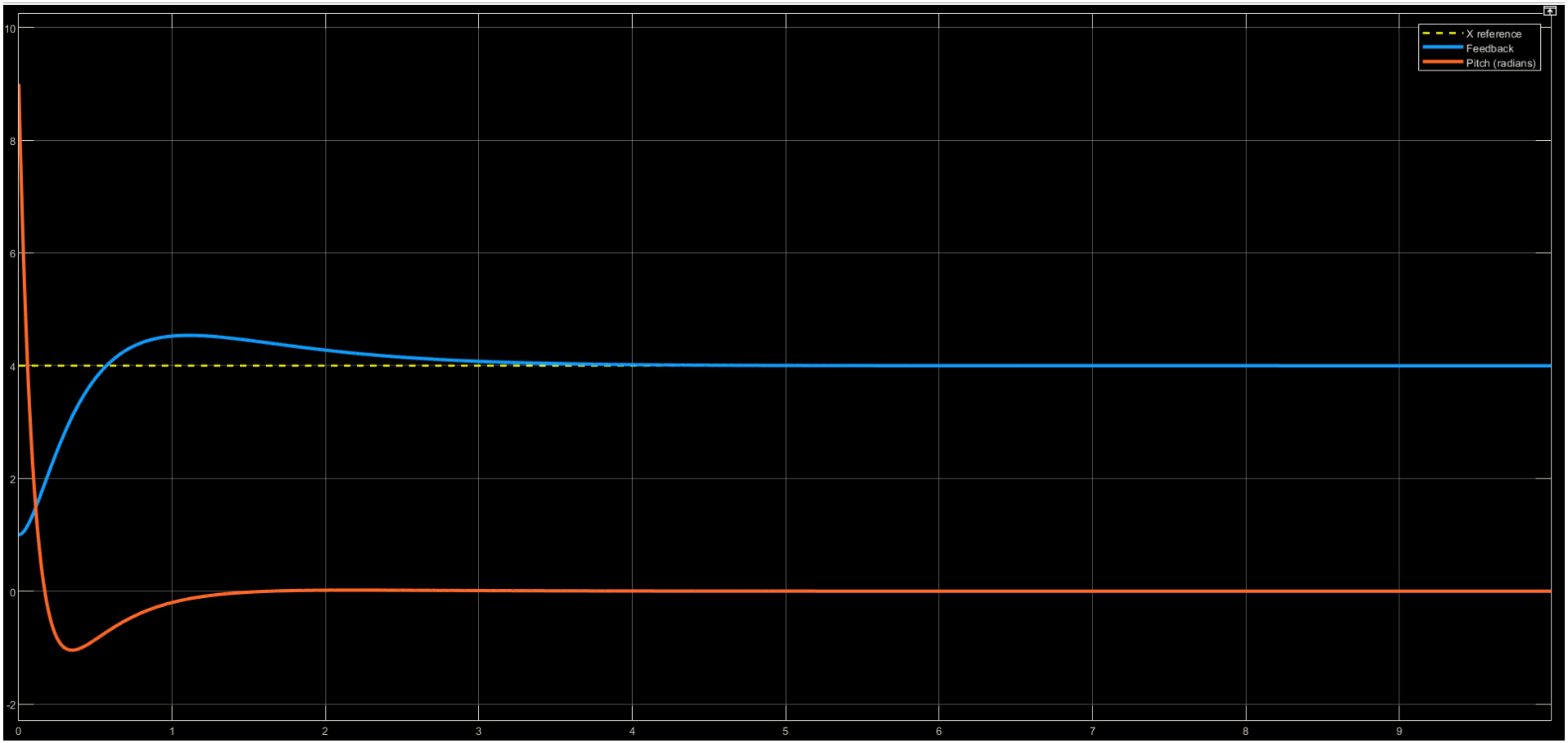
Simulation Results obtained by tuning parameters

- $K_p = 3, K_d = 2.4$



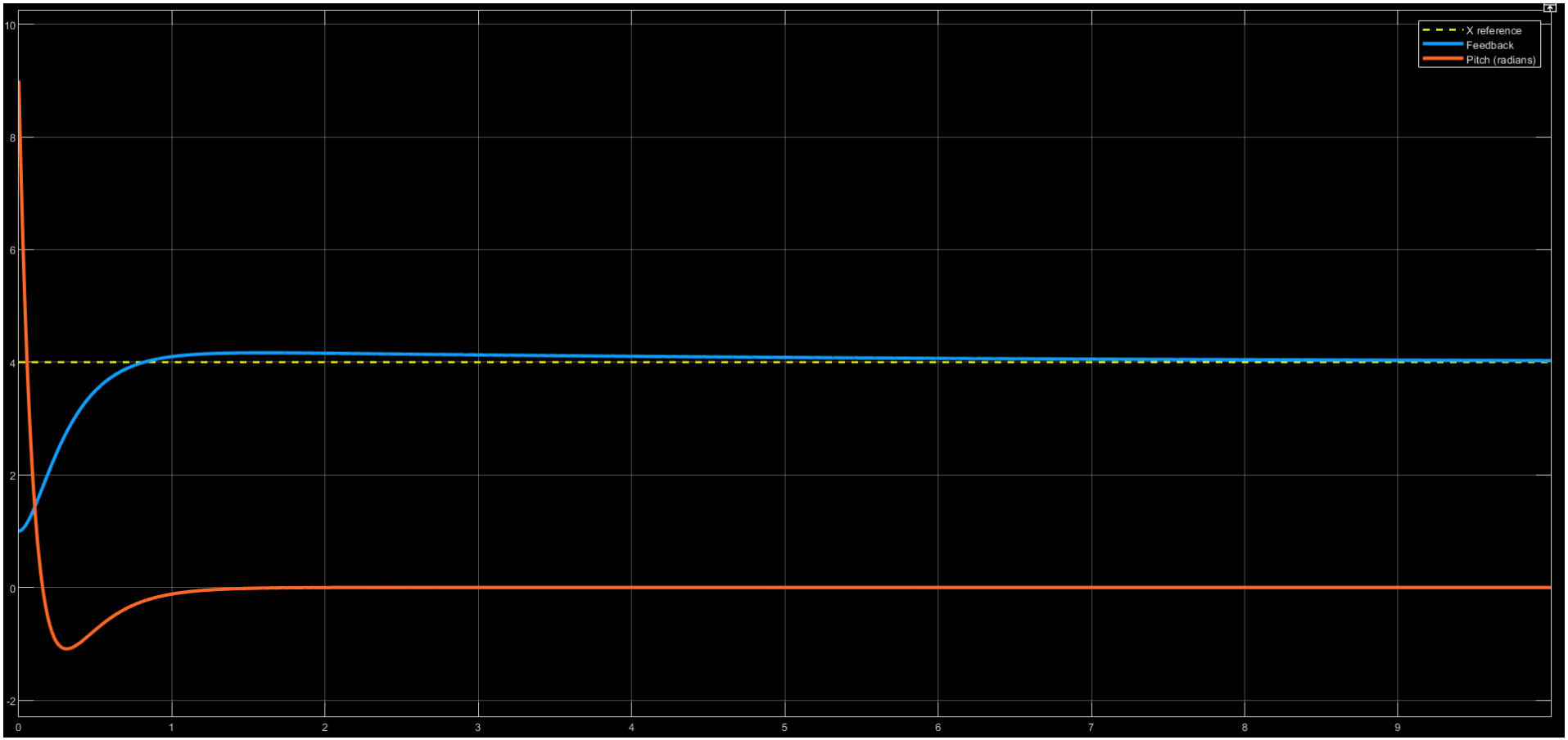
Simulation Results obtained by tuning parameters

- $K_p = 3, K_d = 1, K_i = 1$



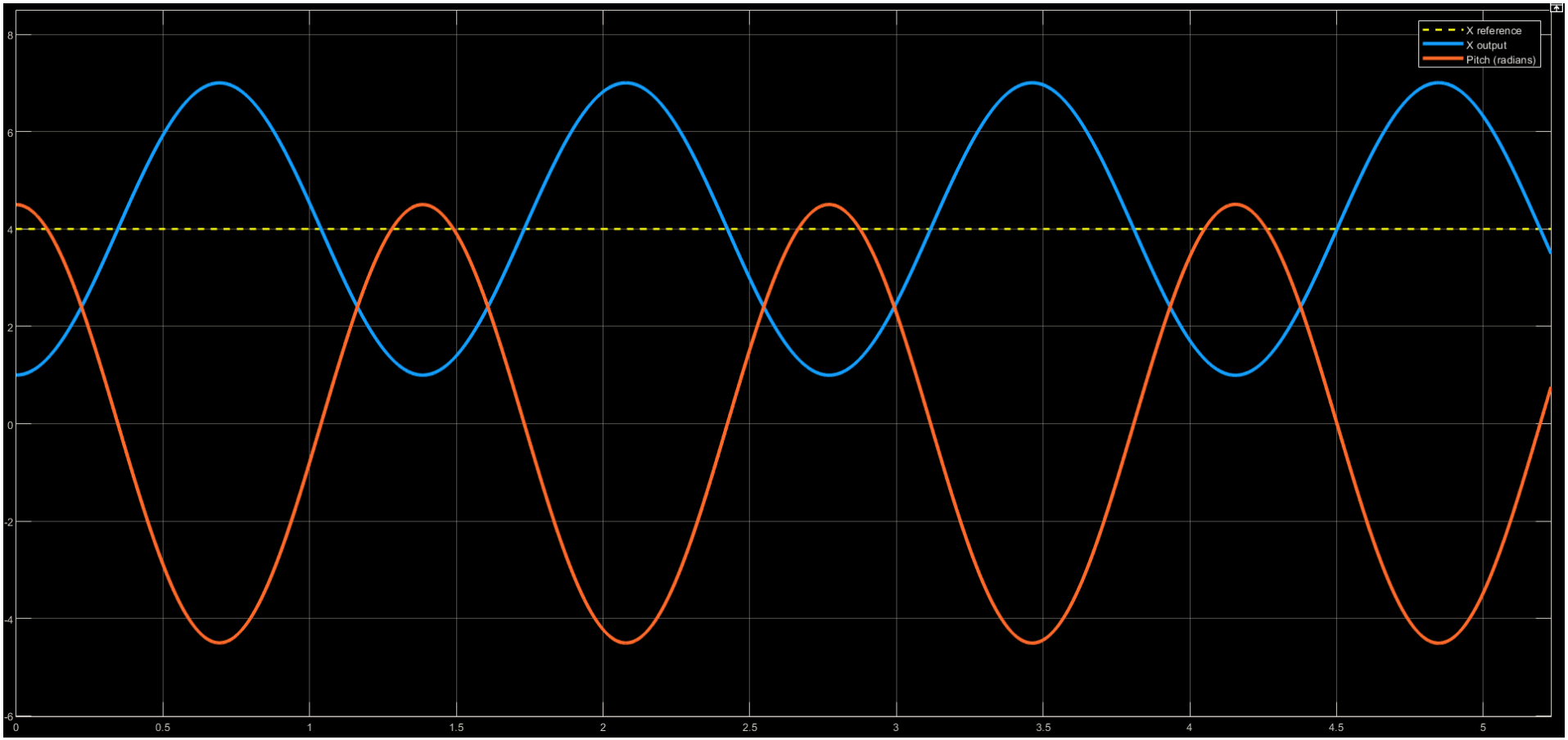
Simulation Results obtained by tuning parameters

- $K_p = 3, K_d = 1, K_i = 0.6$



Simulation Results obtained by tuning parameters

- $K_p = 3, K_i = 0.001$



Arduino Code that Calculates Servo Motor Angles from (x,y,z,Roll,Pitch,Yaw) Input

```
// ----------------------declaration of base points-----------------------------
 float  r_base = 0.0740;

 float base_point[6][3] = {
    {   -r_base * cos(28*deg_to_rad) ,   -r_base * sin(28*deg_to_rad) , 0.03},
    {   -r_base * cos(28*deg_to_rad) ,    r_base * sin(28*deg_to_rad) , 0.03},
    {    r_base * cos(88*deg_to_rad) ,    r_base * sin(88*deg_to_rad) , 0.03},
    {    r_base * cos(32*deg_to_rad) ,    r_base * sin(32*deg_to_rad) , 0.03},
    {    r_base * cos(32*deg_to_rad) ,   -r_base * sin(32*deg_to_rad) , 0.03},
    {    r_base * cos(88*deg_to_rad) ,   -r_base * sin(88*deg_to_rad) , 0.03}
};
//---------------------end of declaration of base points---------------------------

//----------------------declaration of platform points---------------------------
 float r_top = 0.0790;

float top_point[6][3] = {
    {   -r_top * cos(pi/4)          ,   -r_top * sin(pi/4)          ,   -0.005},
    {   -r_top * cos(pi/4)          ,    r_top * sin(pi/4)          ,   -0.005},
    {   -r_top * sin(15*deg_to_rad) ,    r_top * cos(15*deg_to_rad) ,   -0.005},
    {    r_top * cos(15*deg_to_rad) ,    r_top * sin(15*deg_to_rad) ,   -0.005},
    {    r_top * cos(15*deg_to_rad) ,   -r_top * sin(15*deg_to_rad) ,   -0.005},
    {   -r_top * sin(15*deg_to_rad) ,   -r_top * cos(15*deg_to_rad) ,   -0.005}
};
//-----------------------end of declaration of platform points ------------------
```

```
0.21    0.21    0.21    0.19    0.19    0.21
59.19
120.81
73.59
63.26
116.74
106.41
```

```
int zero_pos[6]={90,90,90,90,90,90};    // try to put this in degrees

//initial position of platform centre from base centre

   float trans_init[3] = {0   ,   0   ,  z_home  };       //translat

   float rot_init[3]   = {0   ,   0   ,   0     };         //rotati


float beta[6]=  { pi/2 , -pi/2 , -pi/6 , 5*pi/6 , -5*pi/6, pi/6 };
float Rm = 0.024;               // length of servo arm
float  D = 0.205;                // length of connecting rod

 float req_pos[6] ={0  ,  0  ,  0  ,  0  ,  10,  0 };

void getTrans_matrix() {

  Trans_matrix[0] = req_pos[0]+trans_init[0];
  Trans_matrix[1] = req_pos[1]+trans_init[1];
  Trans_matrix[2] = req_pos[2]+trans_init[2];
```

```
0.21    0.21    0.21    0.19    0.19    0.21
59.19
120.81
73.59
63.26
116.74
106.41
```

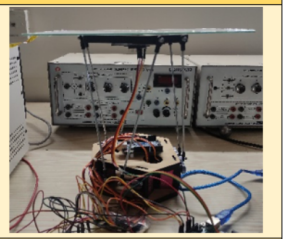
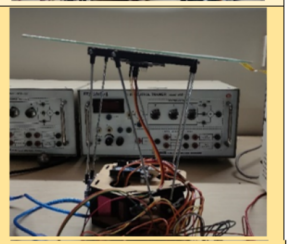
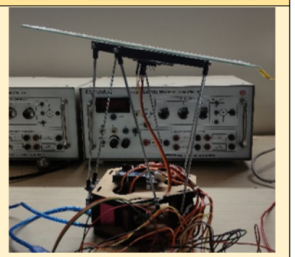
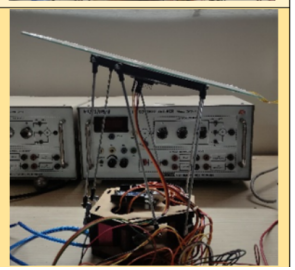
```
for(int i=0;i<6;i++){
   Serial.print(leg_length[i]);
   Serial.print('\t');
}
Serial.print('\n');

getservo_angle();

 for(int i=0;i<6;i++){
   if(i==0 || i==2  || i==4){
   Serial.print(90-servo_angle[i]);
   Serial.print('\n');
   }
   else{
   Serial.print(90+servo_angle[i]);
   Serial.print('\n');
   }
}

Serial.print('\n');

for(int i=0;i<6;i++){
   if(i==0 || i==2  || i==4)
   myservo[i].write(90-servo_angle[i]);
   else
  myservo[i].write(90+servo_angle[i]);
}
```

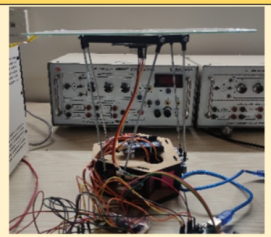
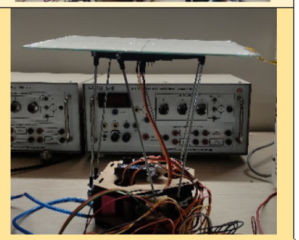
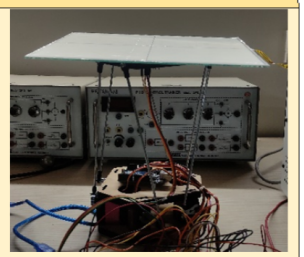
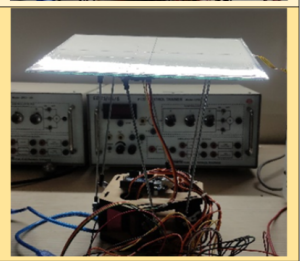
IMU Readings (Roll, Pitch) Cross-Check

```
243
244  sensors_event_t accel_event;
245  sensors_event_t mag_event;
246  sensors_event_t bmp_event;
247  sensors_vec_t   orientation;
248
249  /* Calculate pitch and roll from the raw accelerometer data */
250  accel.getEvent(&accel_event);
251  if (dof.accelGetOrientation(&accel_event, &orientation))
252  {
253      /* 'orientation' should have valid .roll and .pitch fields */
254      Serial.print(F("Roll: "));
255      Serial.print(orientation.roll);
256      Serial.print(F("; "));
257      Serial.print(F("Pitch: "));
258      Serial.print(orientation.pitch);
259      Serial.print(F("; "));
260  }
```

```
Roll: -58.09; Pitch: 9.16;
Roll: 30.54; Pitch: -5.39;
Roll: 57.67; Pitch: 13.84;
Roll: 11.40; Pitch: -2.63;
Roll: 5.64; Pitch: 9.05;
Roll: 3.87; Pitch: 11.69;
Roll: 11.98; Pitch: 10.21;
Roll: 7.54; Pitch: 5.58;
Roll: -1.67; Pitch: 8.40;
Roll: -0.89; Pitch: 12.24;
Roll: 4.28; Pitch: 7.14;
Roll: 4.61; Pitch: 8.96;
Roll: 2.57; Pitch: 10.35;
```

Results

Theoretical (Roll, Pitch)	Actual (Roll, Pitch)	Pictorial Depiction
(0, 0)	(0.27, -0.69)	
(0, 5)	(0.26, 5.22)	
(0, 10)	(-0.69, 10.71)	
(0, 15)	(0.27, 15.67)	

Theoretical (Roll, Pitch)	Actual (Roll, Pitch)	Pictorial Depiction
(0, 0)	(0.27, 0.69)	
(5, 0)	(6.73, 0.22)	
(10, 0)	(11.72, -0.51)	
(15, 0)	(16.83, -0.54)	

Future Prospects

- Position and trajectory control of ball will be achieved by combining the Arduino code and values with MATLAB environment.
- Testing of Stewart platform on ground vehicles moving in an inclined plane.