AA216: Flight Mechanics and Classical Control

Ball Balancing Bot

Group 5

Atharva Chavan - 240003021

Dhruv Bhardwaj - 240003026

Pratul Pan - 240005035

Sanyuja Kharwandikar - *240005042*

Tirth Vinodrai Gohil - 240003079

The Problem

The essential problem is to balance a ball on the top of a platform, otherwise known as a Stewart Platform.

To achieve this, the platform must be controlled precisely and according to the location and motion of the ball, in order to always balance it on the centre, irrespective of its state.

Our step-by-step approach to the problem

01 : Locate

The Location of the ball is found out using the Pi Cam connected to the Raspberry Pi.

Ball is identified using image processing.

02 : Compute

The Tilt of the Platform requiered to balance the ball is calculated from the PID closed loop controller

03 : Actuate

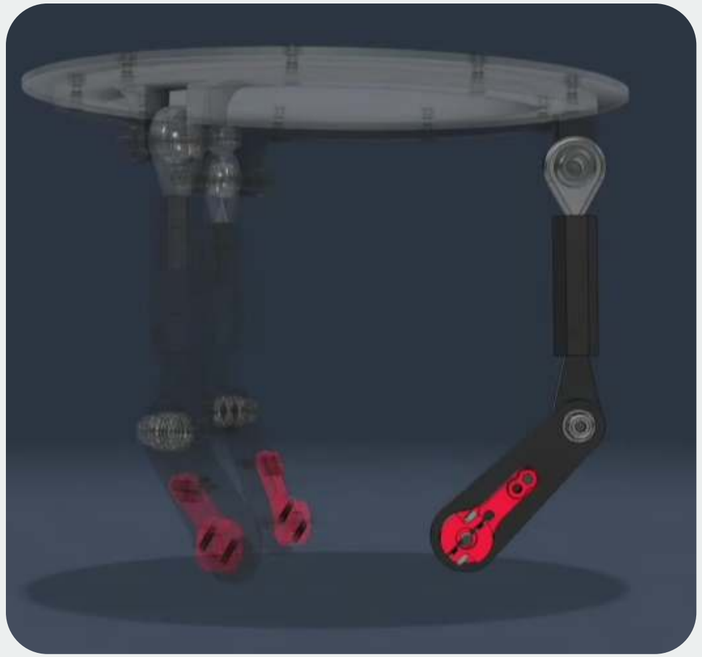
The Raspberry Pi, via the ESP32, issues commands to the Servo motors to rotate to specific angles found out using inverse kinematics equations.

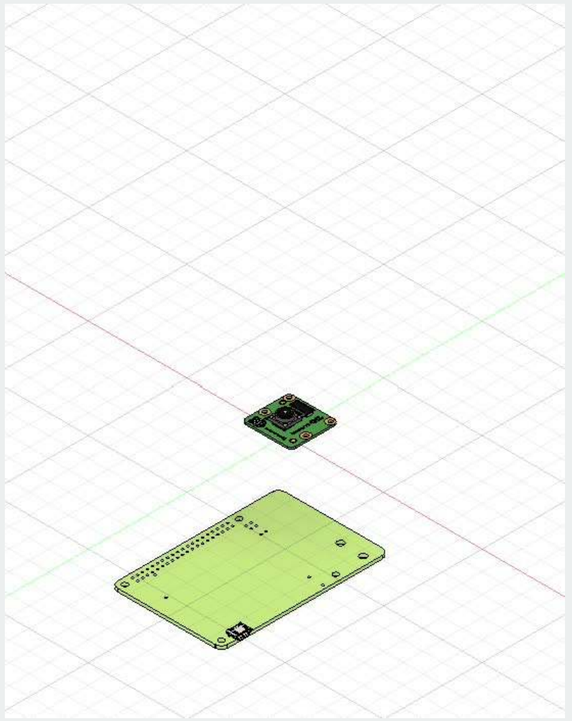
04 : Repeat

The Loop repeats at a high frequency (limited by the computational power of the microprocessor, Raspberry Pi) so that the system adapts to the updated position of the ball.

CAD Design of the Robot

The Design consists of the main platform connected to the housing of motors and controllers via **3RRS (Revolute-Revolute-Spherical) Joints**. Hence the system can be termed as a **3RRS parallel manipulator.**



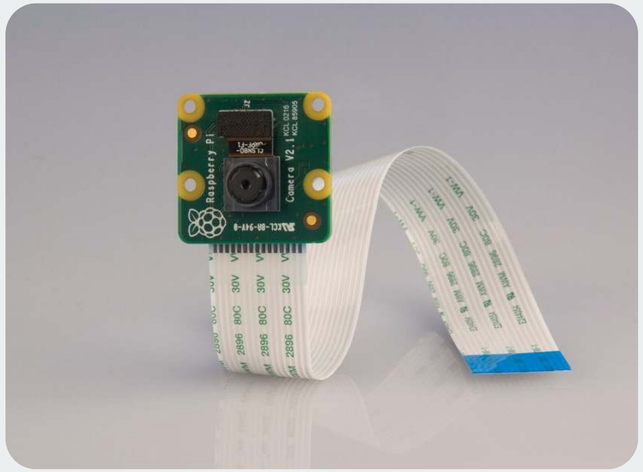


Hardware Components used

- The system uses **Raspberry Pi 5** as the main microprocessor unit to handle image processing and Inverse Kinematics.
- **Pi Cam** provides visual input to the Raspberry Pi 5 which is used to locate the ball.
- The **Servo Motors** tilt the platform in the desired orientation, which are connected to **ESP32 Microcontroller,** which recieves commands from the RPi for actuating motors.
- The System is powered by **DC Voltage supply.**

RPi 5

Servo Motor

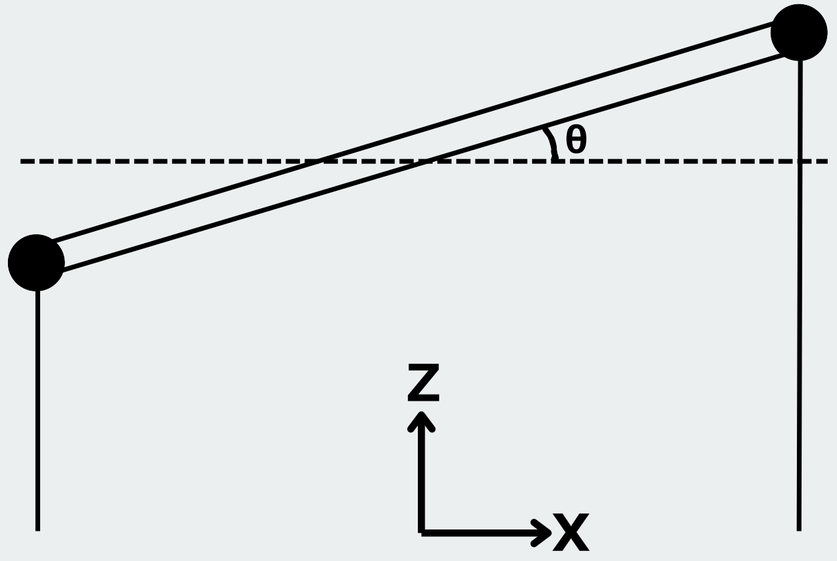


Pi Cam

ESP32

Ball Dynamics

- Forces Along Inclined Platform:
  - **F = mgsinθ**
- Equation of Motion:
  - **mx¨ = mgsinθ**
- Small Angle Approximation:
  - **sinθ ≈ θ**
- Linearized System:
  - **x¨ = gθ**
- Platform tilt (θ) directly controls ball acceleration which is the basis for controller design



Side View

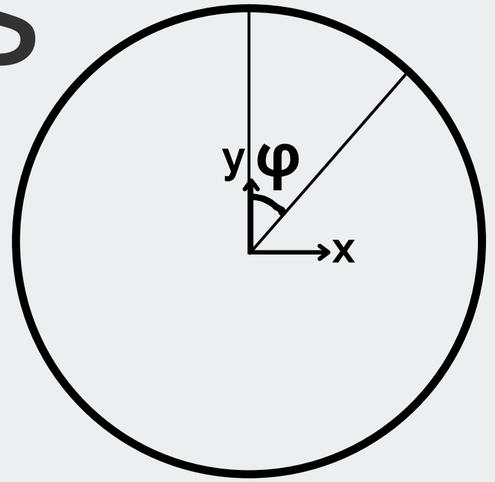
AA216 : *Flight Mechanics and Classical Control*

Inverse Kinematics

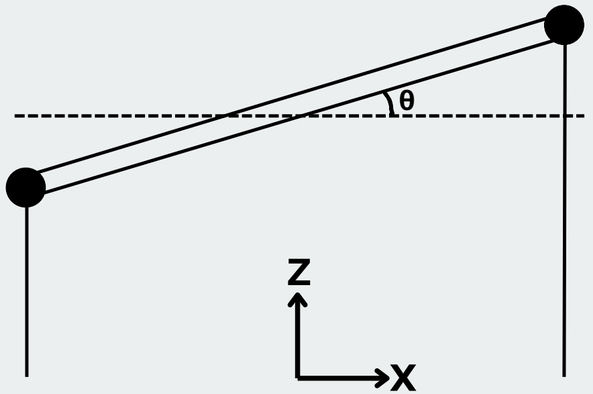
Mathematical relations and equations are used the find the angle of each of the servo motors corresponding to the Tilt magnitude and Direction required.

- Input: Platform tilt → θ (tilt angle), φ (direction angle)
- Output: Motor angles → θ_1, θ_2, θ_3

- θ (Theta) – Tilt Magnitude:
  - Represents how much the platform is tilted
  - θ = 0° means flat surface
  - Higher θ means greater inclination
- φ (Phi) – Tilt Direction:
  - Represents the direction of tilt in the horizontal plane
  - φ = 0° → tilt along x-axis
  - φ = 90° → tilt along y-axis
  - φ defines orientation of tilt



Top View



Side View

Inverse Kinematics

Rotation Matrix : *R = Rz(ϕ) Rx(θ)*

$$R_z(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

- For any point on platform:
  - Pi′=R·Pi
  - Pi: original point
  - Pi′: new position after tilt
- For each leg:
  - Li=Pi′−Bi
  - Bi: fixed base joint
  - Li: direction of ith leg

- Servo angle depends on projection of Li

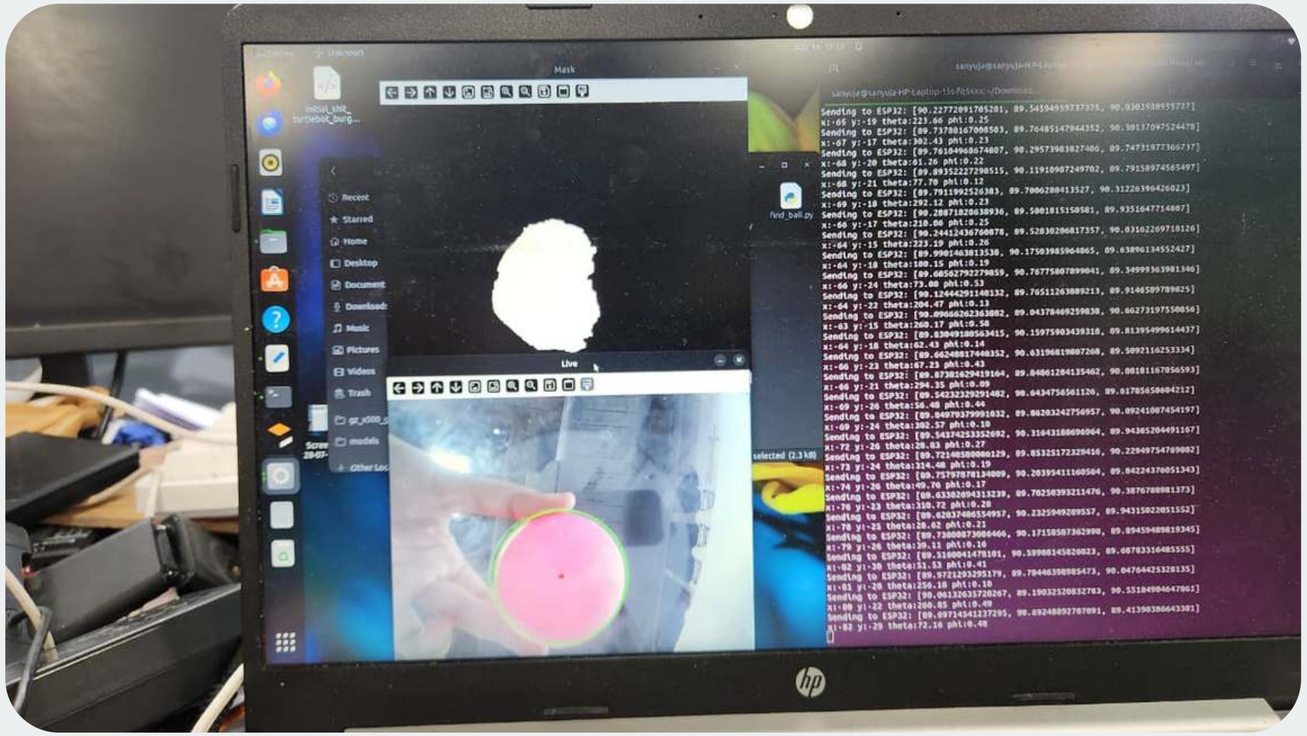
Typical relation:

$$\theta_i = \tan^{-1}\left(\frac{y_i}{x_i}\right)$$

- Uses geometry of links and constraints

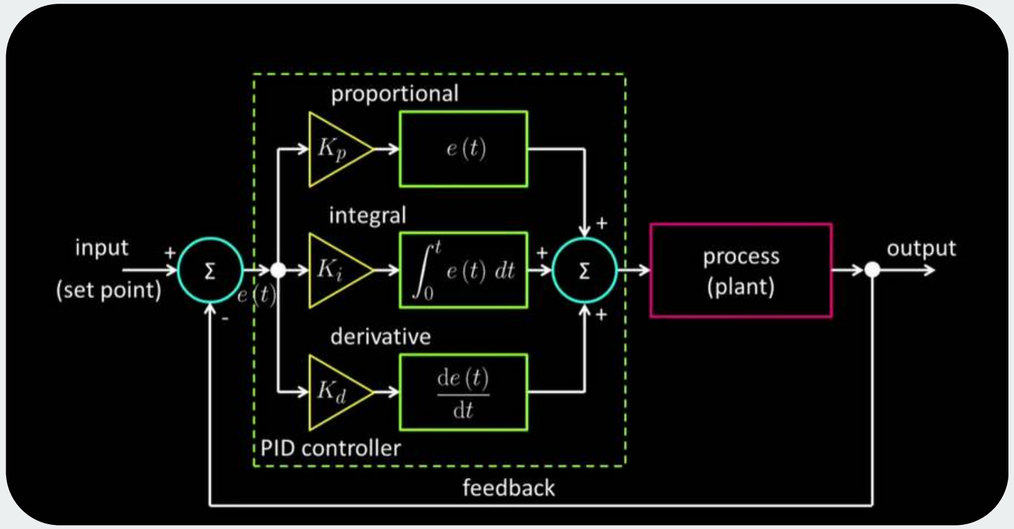
Tracking Ball's position

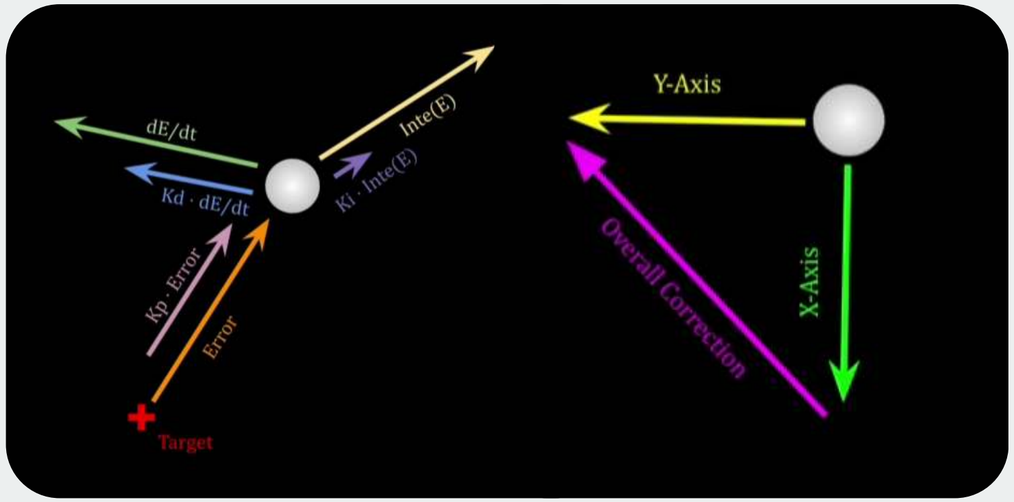
- **Image Acquisition & Preprocessing :** Capture frames from overhead camera; convert to HSV and apply thresholding + filtering for segmentation
- **Object Detection & Localization :** Perform contour/blob detection; compute centroid to obtain ball position (x, y)
- **State Feedback Generation :** Map pixel coordinates to platform frame and provide real-time position (and velocity) to controller



Control using PID

- **Error Computation :** Error = Desired position – Measured position (x, y)
- **Control Law (PID):**
  - **u(t) = Kp e(t) + Ki ∫e(t) dt + Kd d(e(t))/dt**
- **Control Action :** Generates corrective tilt commands to minimize error and stabilize the ball
- PID is applied **separately in X and Y directions** and combined to get final Tilt (in terms of θ & φ)

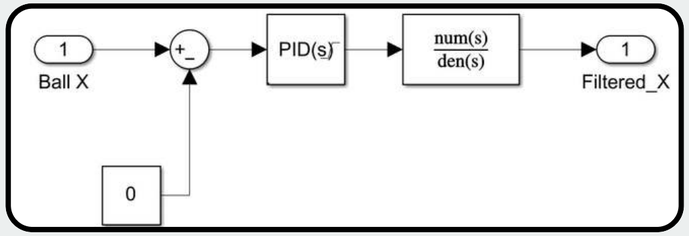


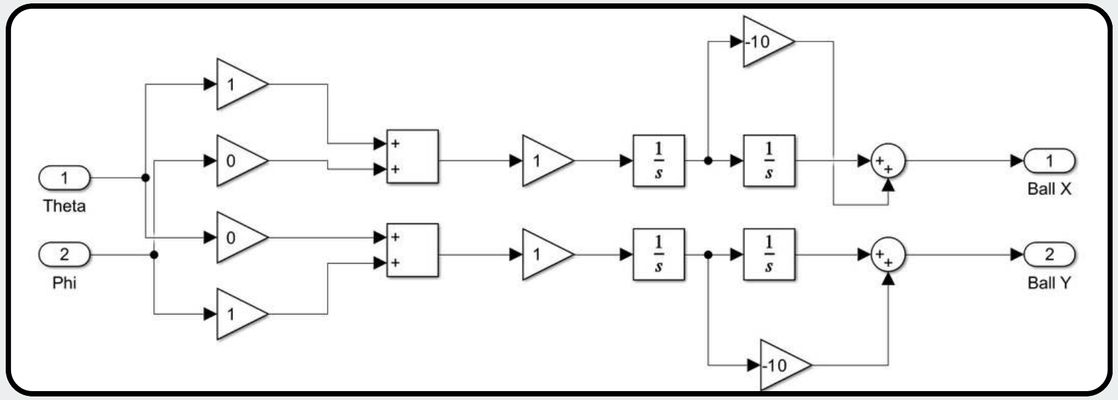


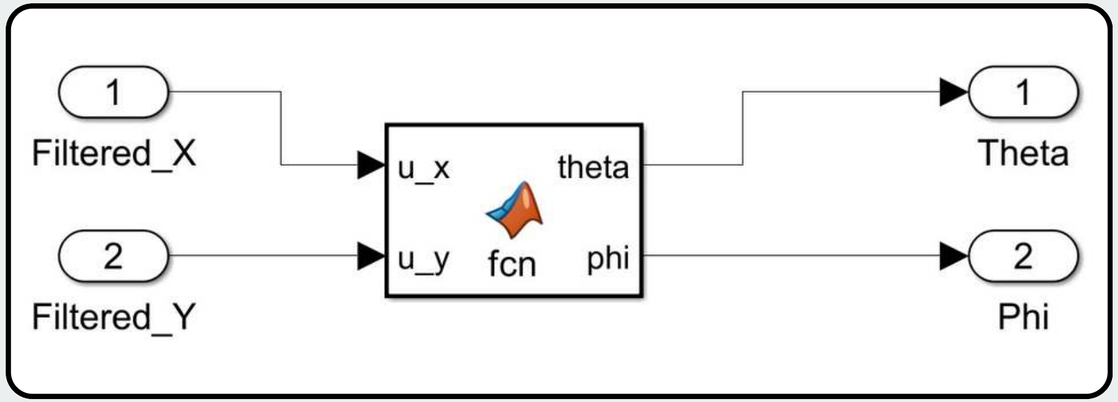
AA216 : *Flight Mechanics and Classical Control*

Simulations

```
1   function [theta, phi] = fcn(u_x, u_y)
2
3   theta = u_x;
4   phi   = u_y;
5
6   end
```

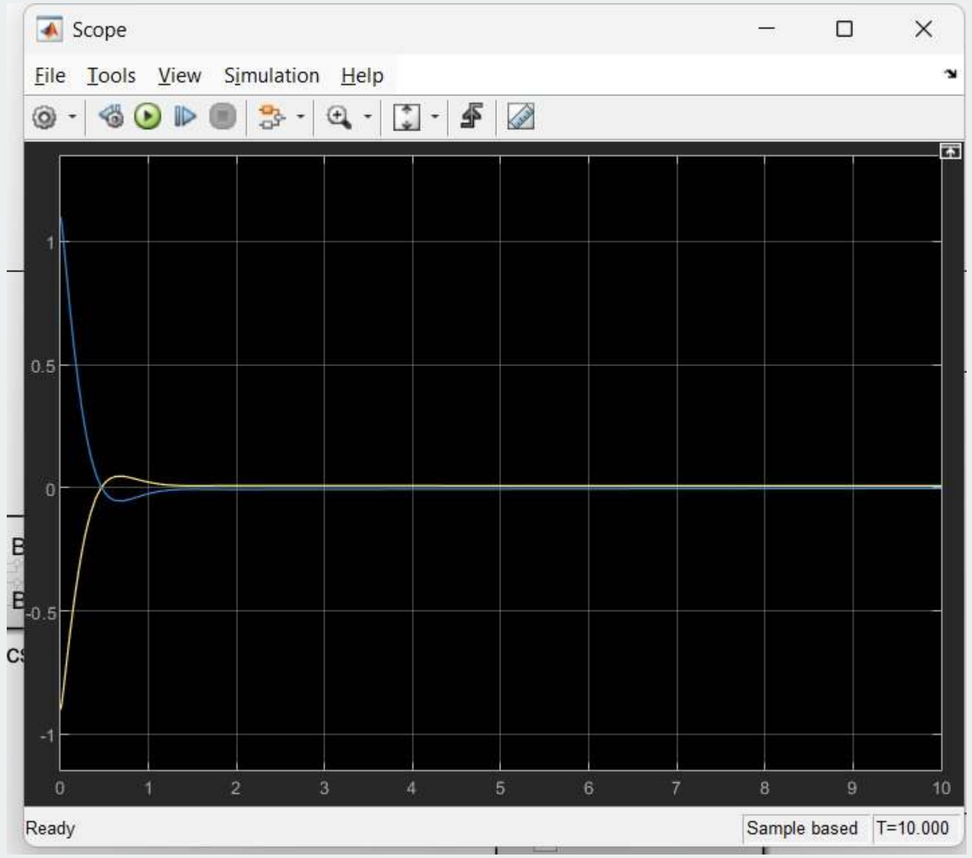






- **Model Development :** Dynamic model of ball and platform implemented (in MATLAB/Simulink)
  - Includes system kinematics and control logic
- **Controller Testing :** PID controller evaluated under different initial conditions
  - Performance analyzed for stability, response time, and overshoot
- **Results & Validation :** Simulation verifies system behavior before hardware implementation
  - Helps in tuning controller gains for optimal performance

Simulations : Result



The system successfully stabilizes the ball at the desired position with minimal steady-state error, demonstrating effective closed-loop control. It responds quickly to disturbances with acceptable overshoot and settling time, indicating good dynamic performance. The tuned PID controller ensures smooth and stable operation across varying initial conditions. Additionally, the experimental results closely match simulation outcomes, validating the accuracy and reliability of the overall system design.

Controlled motion of Platform

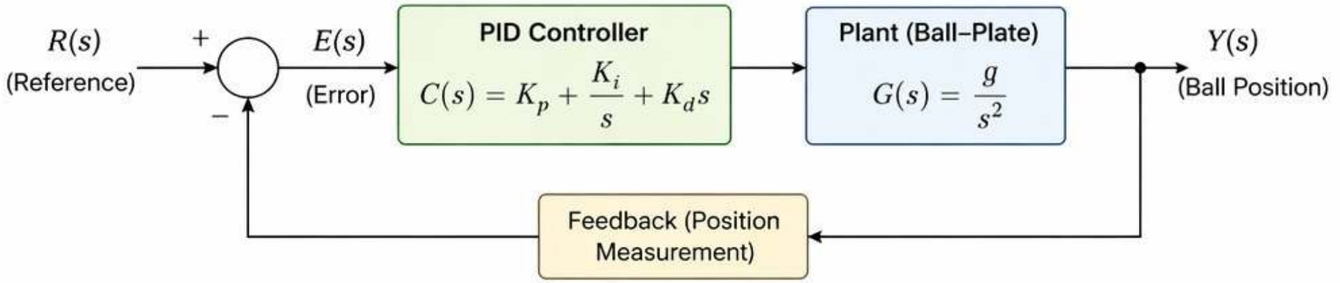
PID CONTROLLER DESIGN – BALL ON PLATE SYSTEM

1. SYSTEM MODEL

For small angles, the ball–plate system can be approximated as a double integrator.

$$G(s) = \frac{g}{s^2} \qquad (g = 9.81\ m/s^2)$$

2. BLOCK DIAGRAM



3. OPEN LOOP TRANSFER FUNCTION

$$G_{ol}(s) = C(s)G(s) = \left(K_p + \frac{K_i}{s} + K_d s\right)\frac{g}{s^2} = \frac{g(K_d s^2 + K_p s + K_i)}{s^3}$$

4. CLOSED LOOP TRANSFER FUNCTION

$$T(s) = \frac{C(s)G(s)}{1 + C(s)G(s)} = \frac{g(K_d s^2 + K_p s + K_i)}{s^3 + g(K_d s^2 + K_p s + K_i)}$$

5. CHARACTERISTIC EQUATION

Denominator = 0

$$s^3 + gK_d s^2 + gK_p s + gK_i = 0$$

6. CRITICALLY DAMPED DESIGN

For a critically damped third-order system, the desired characteristic equation is:

$$(s + a)^3 = s^3 + 3as^2 + 3a^2 s + a^3 = 0$$

Match coefficients with the characteristic equation:

$$s^3 + gK_d s^2 + gK_p s + gK_i = s^3 + 3as^2 + 3a^2 s + a^3$$

Therefore,

$$gK_d = 3a \Rightarrow K_d = \frac{3a}{g} \qquad gK_p = 3a^2 \Rightarrow K_p = \frac{3a^2}{g} \qquad gK_i = a^3 \Rightarrow K_i = \frac{a^3}{g}$$

7. DESIGN SUMMARY

- Choose a > 0 (design parameter) to set the speed of response.
- Compute PID gains:

$$K_d = \frac{3a}{g}, \quad K_p = \frac{3a^2}{g}, \quad K_i = \frac{a^3}{g}$$

- These gains give a critically damped closed-loop response (no overshoot, fastest without oscillation for this model).
- In practice, due to unmodeled dynamics and delays, the response may become underdamped.

8. EXAMPLE (Choose a = 2 rad/s)

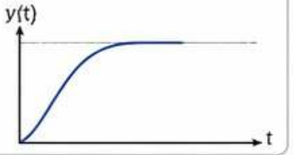
Given $g = 9.81$ m/s²

$$K_d = \frac{3(2)}{9.81} = 0.612\ ,\quad K_p = \frac{3(2)^2}{9.81} = 1.224\ ,\quad K_i = \frac{(2)^3}{9.81} = 0.816$$

9. RESULT (IDEAL)

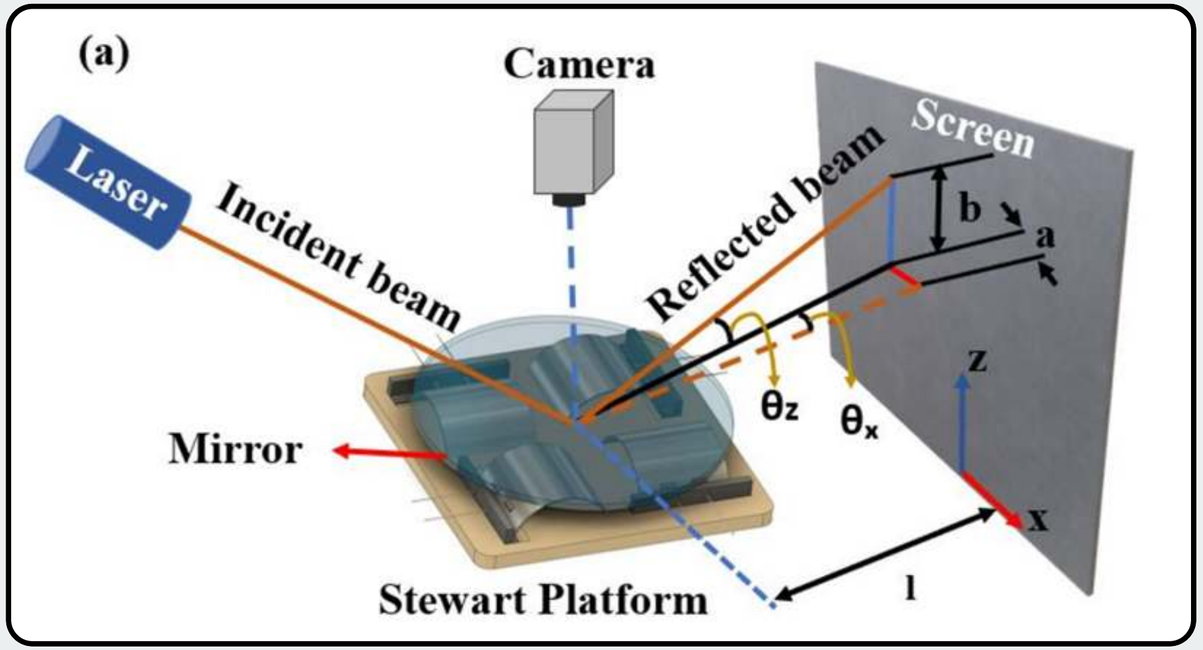
Closed-loop poles: $(s + a)^3 = 0$ → triple real pole at $s = -a$

⇒ Critically damped response (no oscillation, fastest settling).



Future Scope and Applications

- **Beam Steering Systems :** In advanced systems, Stewart platforms provide precise multi-axis control and are used in applications like beam steering. For instance, mirror-based Stewart platforms can steer laser beams by adjusting orientation
- **Precision Surgical Platforms :** Enables stable, accurate positioning by compensating disturbances and tremors
- **UAV & Payload Stabilization :** Applicable in drone landing on moving platforms and maintaining payload stability



Thank you very much for your attention!