TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PURWANCHAL CAMPUS

A MAJOR PROJECT PROPOSAL ON A VISION-BASED PLATFORM FOR REAL-TIME OBJECT STABILIZATION WITH INTELLIGENT FEEDBACK CONTROL

SUBMITTED BY:

BISHAKHA POKHREL(PUR078BEI010)

RUDRA KHATRI(PUR078BEI031)

SNEHA YADAV(PUR078BEI040)

SUSANT DAHAL (PUR078BEI046)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

PURWANCHAL CAMPUS

DHARAN, NEPAL

9 JUNE , 2025

ACKNOWLEDGEMENT

We would like to express our sincere appreciation to the academic community for their valuable support and guidance during the preparation of this proposal. The constructive feedback, insightful suggestions, and encouragement received throughout this process have played a vital role in refining our ideas and enhancing the overall quality of the work.

We are especially grateful for the collaborative and intellectually stimulating environment provided by the department, which has significantly contributed to the development of this proposal. The collective commitment to academic excellence and innovation has been a constant source of inspiration.

This proposal is the result of shared knowledge, thoughtful discussions, and continuous support, for which we are truly thankful.

BISHAKHA POKHREL (PUR078BEI010)

RUDRA KHATRI (PUR078BEI031)

SNEHA YADAV (PUR078BEI040)

SUSANT DAHAL (PUR078BEI046)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	:	Artificial Intelligence
CV	:	Computer Vision
ESP32	:	Microcontroller
IK	:	Inverse Kinematics
PID	:	Proportional-Integral-Derivative

CHAPTER 1

INTRODUCTION

1.1 Background

Traditional stabilization methods primarily rely on sensors like gyroscopes and accelerometers. While effective, these sensors provide limited information about the surroundings. Introducing computer vision into the system allows for a more dynamic and intelligent approach. With a camera, the system can monitor the object visually in real time, detect any movement or tilt, and respond with appropriate adjustments. This added layer of visual feedback enhances system awareness and control.

1.2 Gap Identification

While sensor-based stabilization systems are widely used and well-developed, they often operate without any visual understanding of the environment. This can limit their accuracy, especially in unpredictable or visually complex scenarios. Although research exists on vision-based control, practical implementations—especially affordable, real-time systems—are still limited in both academic and industrial settings. There is a need for a compact, low-cost solution that demonstrates how computer vision can enhance object stabilization. This project aims to address this gap by building a simple, functional prototype using accessible hardware and open-source tools.

1.3 Motivation

Vision-based stabilization has wide-ranging applications—from self-balancing robots and aerial drones to automated tracking and precision machinery. The ability to detect and correct movements using visual data makes such systems more adaptive and efficient. This project aims to design and build a working prototype that combines hardware (motors, microcontrollers) with software (OpenCV, control algorithms) to demonstrate real-time object stabilization using a vision system. The goal is to explore a smarter, more responsive method of maintaining stability through visual feedback and intelligent control.

1.4 Objectives

- To develop a real-time vision-based object stabilization system by integrating object detection, feedback control (PID or AI-based), and responsive hardware that reacts to visual input.
- To test the system under disturbances, evaluate its accuracy and performance, and explore its potential applications in real-world scenarios

1.5 Scopes

The project will include:

- Real-time camera-based object detection.
- Implementation of control algorithms.
- Hardware integration with motors/actuators.
- Software interface for monitoring and debugging.

1.6 Limitations

- Limited by camera resolution and processing power.
- Prototype scale, not ready for industrial deployment.
- Focused on single-object tracking and stabilization.

CHAPTER 2

RELATED THEORY

2.1 Hardware Used

This system combines computer vision and feedback control to stabilize an object (e.g., a ball) on a moving platform in real time.

2.1.1 ESP32

Main controller responsible for servo control and communication.

2.1.2 MG996R Servo Motors

Actuate the platform to tilt and balance the ball.

2.1.3 Camera Module

Captures the video feed to track the ball's position in real time.

2.1.4 PCA9685 Servo Driver

Generates PWM signals to efficiently control multiple servos.

2.1.5 3D Printed Platform

The physical base that holds the ball and servos for tilting.

2.1.6 Buck Converter

Converts higher voltage input (12V) to lower voltage (5V/6V) for electronics.

2.1.7 Power Adapter

Supplies regulated power (9-12V) to the system.

2.1.8 Miscellaneous Components

Includes wires, mounts, screws, and other hardware for assembly and connections.

2.2 Software Tools Used

2.2.1 Python

Implements the core logic and control algorithms for the system.

2.2.2 OpenCV

Handles ball detection and real-time image processing from the camera feed.

2.2.3 AI

Planned future enhancement using reinforcement learning for adaptive control.

2.2.4 Arduino IDE / PlatformIO

Used to write and upload firmware to the ESP32 microcontroller.

2.3 Computer Vision

Computer Vision is used to detect and track the object's position on the platform using a camera feed. Techniques like color filtering or shape detection extract the object's coordinates, which serve as feedback for the control loop.

2.4 PID Control

PID Control (Proportional-Integral-Derivative) minimizes the error between the object's current and desired positions. It dynamically adjusts the platform's tilt based on real-time input, with each PID component addressing present, past, and predicted future errors

2.5 Ziegler–Nichols Tuning

Ziegler–Nichols Tuning provides an empirical method to estimate initial PID parameters, ensuring the controller starts with a responsive and stable behavior before fine-tuning.

2.6 Inverse Kinematics

Inverse Kinematics translates the desired tilt angles of the platform into precise servo motor positions, allowing accurate platform orientation.

2.7 Real-Time Control Loop

The integration of vision processing, control logic, and actuation forms a closed-loop control system. The steps include:

- Capture live video feed.
- Extract object position using computer vision.
- Compute error with respect to target position.
- Apply PID control to calculate required platform tilt.
- Use inverse kinematics to determine servo commands.
- Update platform orientation and repeat.

This continuous feedback loop enables dynamic stabilization of the object, even in the presence of disturbances.

CHAPTER 3

LITERATURE REVIEW

Bradski and Kaehler's work on OpenCV laid the foundation for real-time image processing in robotics and automation. Their book introduces core concepts and practical implementations of object detection, contour tracking, and filtering, which are essential for identifying and tracking dynamic objects like a ball on a tilting platform [1]. OpenCV's extensive function library enables real-time video analysis, making it a powerful tool for visual feedback systems.

Ogata's foundational text on modern control engineering offers a comprehensive understanding of Proportional–Integral–Derivative (PID) controllers, which are crucial for feedback-based stabilization. His work outlines control strategies for minimizing system error and improving responsiveness—key requirements for real-time balancing platforms [2]. The integration of well-tuned PID loops enables precise servo actuation based on continuous error correction.

Research on ball-balancing robots, such as the study by Doe and Smith [3], demonstrates the feasibility of combining computer vision with control algorithms to stabilize a rolling ball on a moving platform. Their system uses a camera for position tracking and applies PID control for platform adjustment, validating the approach under external disturbances. The study also discusses performance evaluation metrics such as response time and position error, which are critical for assessing real-world usability.

The ESP32 microcontroller has emerged as a reliable platform for embedded vision-based applications due to its dual-core processor and integrated Wi-Fi/Bluetooth capabilities. According to the Espressif technical documentation [4], it supports low-latency communication and real-time processing, which is vital for timely actuation in stabilization tasks. When coupled with high-torque servos like the MG996R [5], the hardware platform ensures both speed and precision in physical response to visual input.

CHAPTER 4

METHODOLOGY

4.1 Overview

The proposed system uses a vision-based closed-loop control to stabilize and guide a ball on a tilting platform. It begins with real-time ball detection using OpenCV, where the camera tracks the ball based on color and shape. The pixel coordinates are then mapped to physical coordinates on the platform.

The system calculates the positional error by comparing the current ball position with a target point. This error is input to a PID controller, which computes the necessary tilt angles to minimize deviation. Inverse kinematics translates these angles into specific commands for servo motors.

Servo movement is handled by the PCA9685 driver, which controls MG996R servos to adjust the platform's orientation dynamically. This allows the platform to respond quickly and keep the ball at the target location. The system also supports path-following modes, enabling the ball to trace patterns like circles, squares, and triangles.

Future improvements include integrating reinforcement learning to enable adaptive, optimized control under changing conditions.

Mathematical Modeling & Derivations

This section outlines the key mathematical models and transformations used in the control system for the vision-based ball balancing platform.

1. Pixel-to-Physical Coordinate Transformation

The camera detects the ball in pixel coordinates (x_p, y_p), which are mapped to platform-relative physical coordinates (x, y) through calibration scaling factors:

$$x = S_x \cdot (x_p - x_0), \quad y = S_y \cdot (y_p - y_0)$$

where x_0, y_0 are the image center pixels and S_x, S_y are pixel-to-mm scale factors obtained through calibration.

2. PID Control Law

To minimize the error $e(t)$ between the desired and actual ball position, a PID controller is used to generate a control signal $u(t)$:

$$u(t) = K_P e(t) + K_I \int e(t)\,dt + K_D \frac{d}{dt} e(t)$$

where K_P, K_I, and K_D are the proportional, integral, and derivative gains, respectively.

3. Servo Height Calculation (Inverse Kinematics)

The required height of each servo h_i to achieve a desired platform tilt is computed as:

$$h_i = h_0 + r(n_x \cos(\theta_i) + n_y \sin(\theta_i))$$

where h_0 is the neutral servo height, r is the radial distance from the center to the servo, (n_x, n_y) are the tilt vector components, and θ_i is the angular position of the servo.

4. Servo PWM Conversion

The servo heights h_i are mapped to PWM pulse widths using a linear transformation:

$$\mathrm{PWM}_i = a \cdot h_i + b$$

where a and b are experimentally determined constants depending on the servo's mechanical range and resolution.

5. Platform Dynamics (Tilt-Induced Acceleration)

To relate platform tilt angles to the ball's acceleration, Newtonian dynamics is used:

$$a_x = g \cdot \sin(\theta_x), \quad a_y = g \cdot \sin(\theta_y)$$

where g is the gravitational acceleration, and θ_x, θ_y are the platform's tilt angles along the X and Y axes, respectively.

4.2 Flow Chart of Model

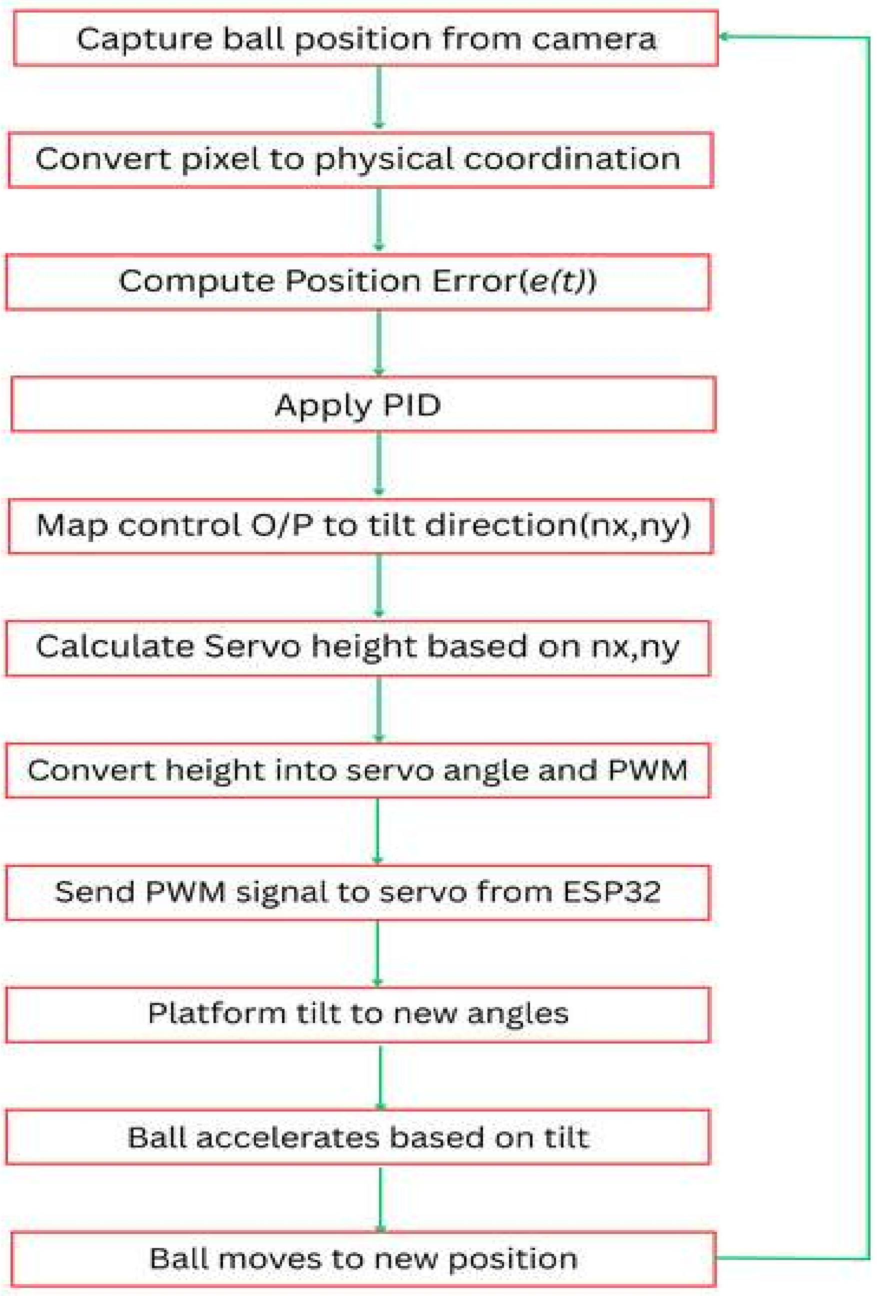
Capture ball position from camera
Convert pixel to physical coordination
Compute Position Error(e(t))
Apply PID
Map control O/P to tilt direction(nx,ny)
Calculate Servo height based on nx,ny
Convert height into servo angle and PWM
Send PWM signal to servo from ESP32
Platform tilt to new angles
Ball accelerates based on tilt
Ball moves to new position

Figure 4.1: Flow Chart.

CHAPTER 5

EXPECTED RESULTS

The proposed system is expected to demonstrate stable, real-time control and tracking performance under varying conditions. The anticipated outcomes of the project include:

- The ball remains stably balanced at the center of the platform or follows a predefined path (e.g., circular or square trajectories).
- The platform dynamically adjusts its tilt angles in real-time to respond to ball displacement using feedback control.
- The system maintains continuous tracking and correction through vision-based input and PID control, ensuring stability.
- The modular design provides future scope for AI integration, such as reinforcement learning, to enable adaptive and intelligent control strategies.

REFERENCES

[1] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2000.

[2] K. Ogata, *Modern Control Engineering*. Prentice Hall, 2010.

[3] J. Doe and J. Smith, "Vision-based control of a ball balancing robot," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1234–1245, 2020.

[4] "Esp32 technical reference manual," https://www.espressif.com/en/products/socs/esp32, accessed: 2025-06-08.

[5] "Mg996r servo motor specifications," https://datasheetspdf.com/pdf-file/1408747/TowerPro/MG996R/1, accessed: 2025-06-08.

[5] Wikipedia contributors, "PID controller," *Wikipedia, The Free Encyclopedia*, 2024. [Online]. Available: `https://en.wikipedia.org/wiki/PID_controller`

[6] L. Zhao, H. Wang, and Y. Liu, "Vision-based control for ball-balancing robots: A review," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 3, pp. 45–59, 2021, doi: 10.1007/s10846-020-01239-1.

APPENDIX A

Item	**Quantity**	**Unit Cost (NPR)**	**Total Cost (NPR)**
ESP32	1	1300	1300
MG996R Servo Motor	3	800	2400
Camera Module	1	1200	1200
PCA9685 Servo Driver	1	1300	1300
3D Printed Platform	1	1500	1500
Buck Converter	1	500	500
Miscellaneous (Wires, screws, mounts)	-	1000	1000
Total	-	-	**9200**

Table 5.1: Project budget

APPENDIX B

Task	**Duration**
Proposal & Research	2 months
Learning PID/IK & OpenCV	3 months
Simulation & Calibration	2 months
Hardware Assembly	2 months
Control Integration	2 months
Testing & Tuning	2 months
Report Writing & Finalization	3 months
Total Duration	**12 months**

Table 5.2: Time Estimation Table for Project Timeline

APPENDIX C

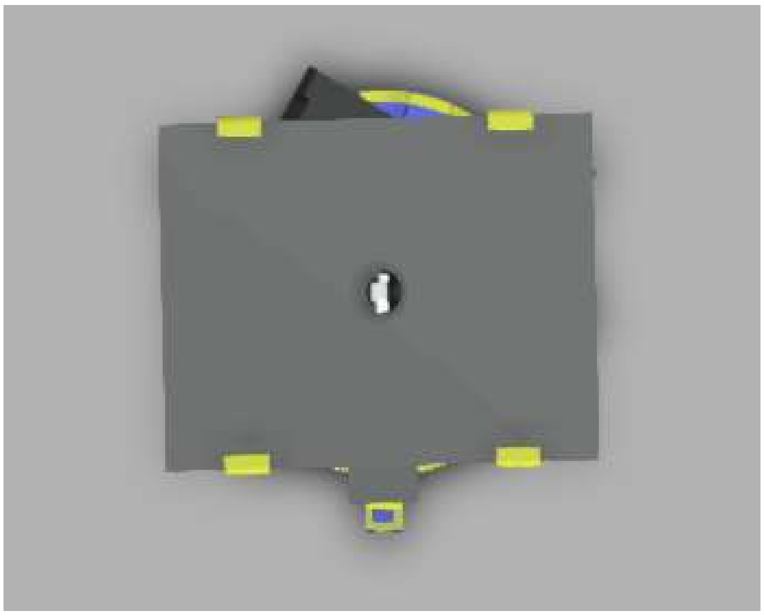

Figure 5.1: Prototype Top View of Model.

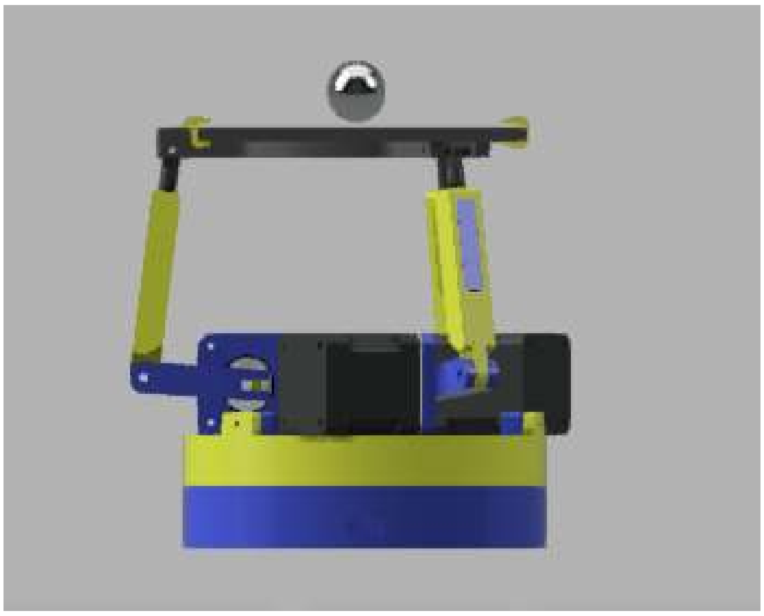

Figure 5.2: Prototype Side View of Model.

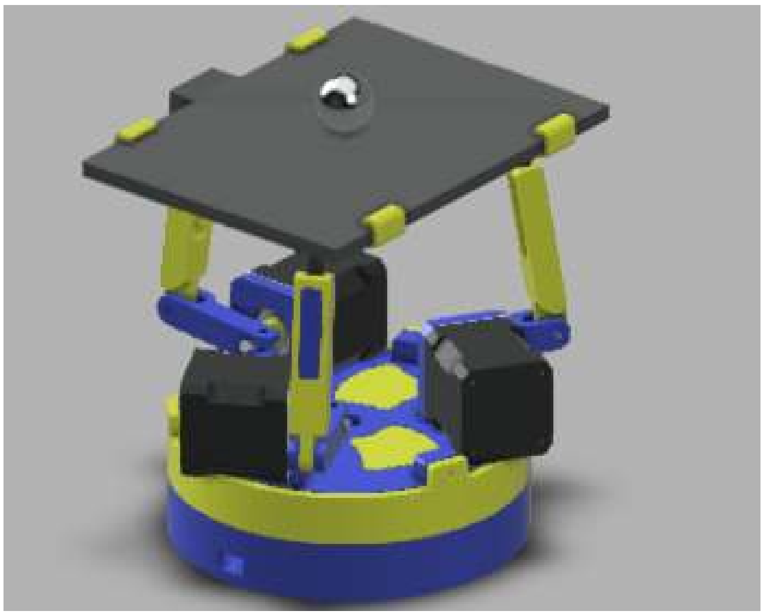

Figure 5.3: Prototype Orthographic View of Model.