IoT-Enabled 2-Axis Gimbal System for Camera Stabilization

Sai Preeth Aduwala*, Alekhya Marini†, Uday Kumar Reddy Pesala‡
*Department of General Engineering, San Jose State University
Email: saipreeth.aduwala@sjsu.edu
†Department of General Engineering, San Jose State University
Email: alekhya.marini@sjsu.edu
‡Department of Computer Engineering, San Jose State University
Email: uday.pesala@sjsu.edu

***Abstract*—This paper presents the design and implementation of a cost-effective, IoT-enabled 2-axis gimbal stabilizer for camera stabilization using an Arduino Nano ESP32, MPU6050 motion sensor, and servo motors. The system measures real-time pitch and roll angles, corrects instability using servos, and transmits data to the Arduino IoT Cloud. A custom 3D-printed chassis enhances stability and alignment. The proposed system offers an accessible and extensible solution for hobbyists and educators in embedded systems and control applications.**

***Index Terms*—Gimbal, Arduino Nano ESP32, MPU6050, Servo Motor, IoT, Cloud, Stabilization, Complementary Filter**

I. Introduction

In today's world of technology, the demand for precision and stability is growing across various domains—particularly in drones, photography, robotics, and industrial automation. One of the most crucial components in ensuring this stability is the gimbal system, which compensates for unwanted motion and keeps the mounted object—usually a camera—steady. Traditional gimbal systems are often expensive and lack flexibility for customization or integration with cloud platforms. This project presents a cost-effective and IoT-enabled 2-axis gimbal stabilizer that uses an Arduino Nano ESP32, MPU6050 motion sensor, and servo motors. The system detects the device's orientation in real-time (pitch and roll), corrects it using servo motors, and uploads this data to the Arduino IoT Cloud, where it can be monitored from anywhere. This solution combines mechanical stabilization with remote data visibility, making it highly relevant in educational, hobbyist, and industrial environments. The design features a real-time feedback system and was complemented by a custom 3D-printed chassis, built using a Bambu Labs A1 printer, to provide mechanical support and precision alignment.

II. Problem Statement

In environments where electronic devices or cameras are exposed to constant motion—such as drones, vehicles, or robots—maintaining a steady orientation becomes a challenge. Even small tilts or vibrations can lead to blurry images, incorrect readings, or unstable footage. Commercially available stabilizers can be expensive and are often not customizable or connectable to online platforms. The specific problems addressed in this project are:

- How to measure real-time orientation changes in pitch and roll?
- How to correct these changes automatically using servo motors?
- How to remotely monitor orientation and servo activity over the internet?

The project aims to build a low-cost, accurate, and connected gimbal stabilization system that addresses these questions using open-source tools and readily available components.

III. Proposed Solution

To solve the problem of dynamic instability and remote monitoring, this project proposes the use of:

- Arduino Nano ESP32: Acts as the controller, processing data and managing the servos.
- MPU6050 Sensor: Captures raw acceleration and gyroscope data for calculating pitch and roll.
- Servo Motors: Adjust the platform position in real-time to counteract tilt.
- Arduino IoT Cloud: Allows real-time monitoring of pitch, roll, and servo angles through a dashboard accessible from any device.

Using a complementary filter, the system combines accelerometer and gyroscope data to generate smooth, reliable orientation readings. These angles are mapped to the servo motors, which react to maintain the desired orientation. The pitch and roll data, along with servo positions, are sent to the cloud for visualization.

IV. Components Used

Here's a detailed look at the components used in the system:

- Arduino Nano ESP32: A powerful microcontroller with built-in Wi-Fi and Bluetooth. It is compact and ideal for IoT projects.
- MPU6050: A 6-axis motion tracking sensor (3-axis accelerometer + 3-axis gyroscope), communicating via I2C.
- 2x Servo Motors (SG90 or MG90): Small, lightweight motors capable of rotating from 0° to 180°, used for controlling pitch and roll.

- Voltage Regulators (e.g., 7805): Provide a stable 5V output from a higher-voltage source to power the servos safely.
- Capacitors (100nF, 1000nF): Used for power stabilization and filtering.
- Resistors (240 ohm, 390 ohm): Used in voltage regulation circuits
- Power Source (Battery or USB): Powers the system. Must be sufficient to drive both ESP32 and servo motors.
- Breadboard and Jump Wires: For prototyping and making all electrical connections.

V. System Architecture

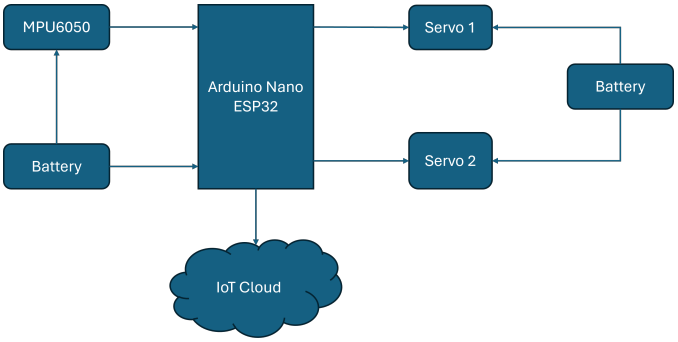

Fig. 1. System Architecture

The system architecture follows a simple flow: Sensor (MPU6050) → Microcontroller (Arduino Nano ESP32) → Actuator (Servo Motors) → Cloud (Arduino IoT)

- The MPU6050 continuously sends motion data to the ESP32 via I2C.
- MPU6050: A 6-axis motion tracking sensor (3-axis accelerometer + 3-axis gyroscope), communicating via I2C.
- The ESP32 calculates the pitch and roll angles using a complementary filter.
- These angles are then mapped to corresponding servo motor positions.
- Servo motors adjust their rotation accordingly to keep the platform level.
- Real-time values of pitch, roll, and servo angles are sent to the cloud

VI. Circuit Diagram

Below is the schematic of the system, clearly showing how all the components are wired together: Key Connections:

- MPU6050: VCC → 3.3V, GND → GND, SDA → GPIO 21, SCL → GPIO 22
- Servo Motors: GPIO 5 and GPIO 6
- Power Supply: Uses voltage regulation to protect the ESP32 from high-current servo spikes

VII. Software and Libraries

The project is programmed using the Arduino IDE and leverages multiple libraries to handle hardware and cloud communication.

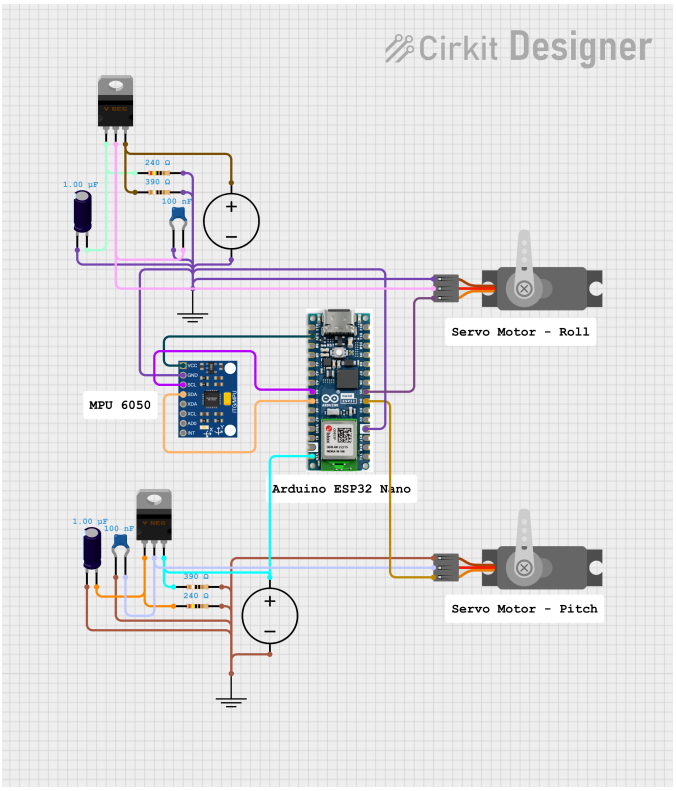

Fig. 2. Circuit Diagram

A. Libraries Used

- `Wire.h` For I2C communication between ESP32 and MPU6050
- `MPU6050.h` Simplifies sensor setup and data acquisition
- `ESP32Servo.h` Enables PWM control of servos with ESP32
- `ArduinoIoTCloud.h` Used to send data to the Arduino IoT Cloud platform
- `Arduino_ConnectionHandler.h` Manages Wi-Fi credentials and network reconnection

This combination of libraries allows seamless integration between hardware control and cloud connectivity.

B. Arduino IoT Cloud

The Arduino IoT Cloud is a platform developed by Arduino to easily create, deploy, and monitor Internet of Things (IoT) applications. It enables seamless integration between hardware and cloud services, allowing users to remotely monitor sensors, control actuators, and visualize data through customizable dashboards. In this project, the Arduino IoT Cloud plays a crucial role in enabling real-time remote monitoring of the gimbal system. It allows the user to:

- View Pitch and Roll Angles: The orientation data collected from the MPU6050 sensor is displayed live on the cloud dashboard.
- Track Servo Positions: The angles at which the servos are currently set (corresponding to pitch and roll compensation) are also uploaded and visualized.

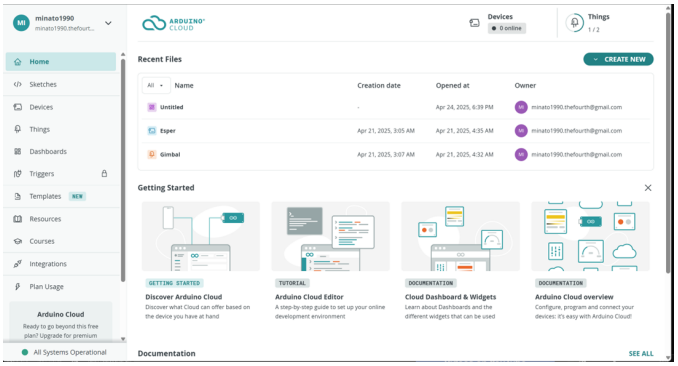

Fig. 3. Arduino IoT Cloud Homepage

- Access Data Remotely: Since the cloud dashboard can be accessed from any web browser or mobile app, it offers convenience and mobility to monitor the system from anywhere.
- Ensure Persistent Connectivity: Thanks to the built-in libraries like ArduinoIoTCloud.h, the ESP32 Nano maintains a stable connection to the cloud and automatically reconnects when needed.

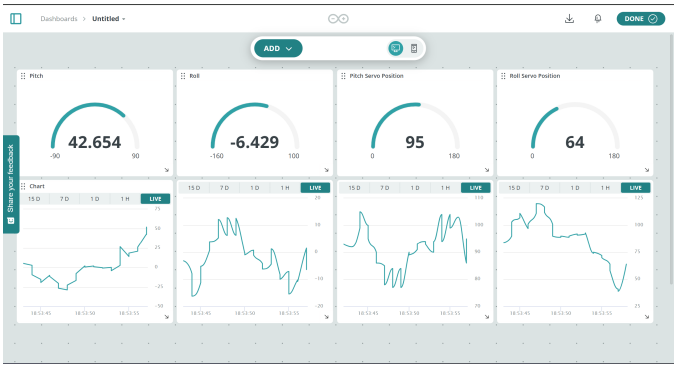

Fig. 4. Arduino IoT Cloud Dashboard

Setting up the IoT cloud involved:

- Creating a thing on the Arduino Cloud platform.
- Defining cloud variables like pitch_IoT, roll_IoT, servo1_IoT, and servo2_IoT.
- Linking the variables in the sketch using the automatically generated thingProperties.h file.
- Using the ArduinoCloud.update() function in the loop() to push data regularly.

This integration adds a modern and smart touch to the project by transforming it from a local system into an IoT-enabled solution, capable of being monitored and potentially controlled from anywhere in the world.

VIII. Code Overview

The code begins by initializing all components in the setup() function. It checks the MPU6050 connection, attaches the servos, and establishes cloud connectivity. In the loop() function:

- Sensor readings are taken via mpu.getMotion6()
- Raw values are used to calculate roll and pitch angles using trigonometric formulas
- A complementary filter blends gyroscope and accelerometer data to produce stable orientation angles
- These angles are mapped to servo positions using map() and written to servos
- Data is uploaded to the cloud every cycle

A small delay ensures consistent timing between updates.

IX. Working Principles

The core principle behind the system is inertial stabilization using feedback control. The MPU6050 sensor outputs acceleration and angular velocity data. While the accelerometer helps in determining orientation relative to gravity, the gyroscope helps in detecting fast movements. To combine their strengths and cancel out their weaknesses (noise and drift), a complementary filter is used:

$$\text{angle} = \alpha \cdot (\text{angle} + \text{gyro} \cdot dt) + (1 - \alpha) \cdot \text{acc\_angle} \quad (1)$$

with $\alpha = 0.98$. gyro * dt integrates angular velocity to estimate angle change. acc_angle is the angle calculated from acceleration data. This filtered angle is converted to a 0–180° range for the servo using the map() function, ensuring the servo moves in proportion to the tilt. Meanwhile, real-time values are pushed to the Arduino IoT Cloud, where users can visualize motion and servo response via a dashboard.

X. Mechanical Design and 3D Printed Chassis

To house and support the two-axis gimbal system, several custom chassis components were designed and 3D printed using the Bambu Labs A1 printer. The use of 3D printing offered a precise, customizable, and lightweight solution to the mechanical design challenges of the gimbal. The printed parts include the base frame, servo mounts, and rotation arms for both pitch and roll axes. These components were modeled to accommodate the physical dimensions of the ESP32 Nano board, the MPU6050 module, and the servo motors, ensuring structural integrity while maintaining smooth and unrestricted movement. Using PLA material, the 3D prints provided a good balance between strength and ease of fabrication. The precision of the Bambu Labs A1 printer allowed for tight tolerances and snap-fit joints, reducing the need for additional fasteners or adhesives. Advantages of using 3D printing in this project included:

- Rapid prototyping and iteration
- Customization of parts to fit unique geometries
- Cost-effective production of one-off or small-batch components
- Clean and professional-looking enclosures

This mechanical platform not only gave the project a polished, robust finish but also supported the dynamic motion required for stabilization effectively, helping the servos operate smoothly without wobble or vibration.

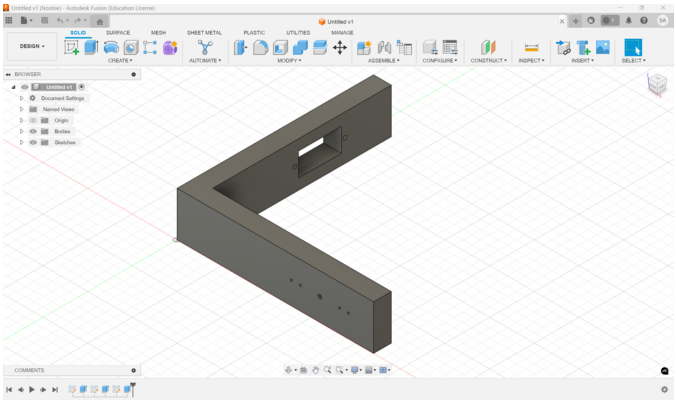

Fig. 5. Mechanical Chassis - Servo 1

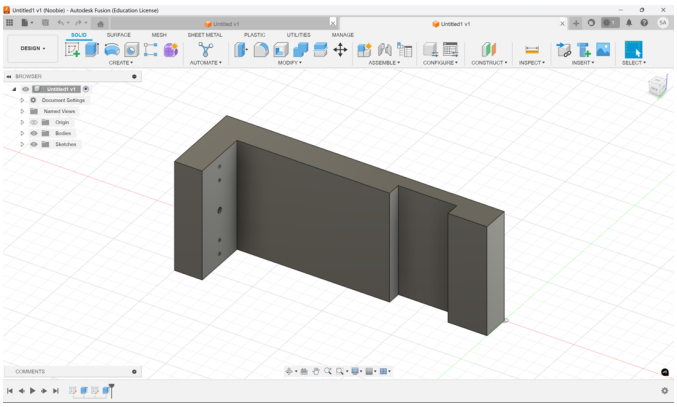

Fig. 6. Mechanical Chassis - Servo 2

XI. Challenges and Limitations

During development, several challenges were encountered:

- Sensor Noise: Raw accelerometer data is noisy, requiring filtering for accuracy.
- Servo Jitter: Servo motors sometimes jitter due to power fluctuations or minor sensor changes.
- Wi-Fi Stability: Cloud connection occasionally drops, especially in areas with poor connectivity.
- Power Supply Issues: Servos can draw more current than USB or onboard regulators can safely supply, requiring an external regulated power source.
- Limited to Two Axes: The system handles only pitch and roll. Adding yaw would require an additional servo and more complex math.

Despite these limitations, the system works reliably under normal conditions and offers room for improvements.

XII. Future Improvements

There are several enhancements that can be made to increase the performance and scope of this project:

- Add Yaw Axis: Introduce a third motor for complete 3-axis stabilization.
- Use Kalman Filter: Replace the complementary filter with a Kalman filter for more accurate data fusion.
- Add Manual Control: Integrate a joystick or mobile app for manual gimbal movement.
- Battery Monitoring: Implement voltage sensors to monitor battery health.
- 3D-Printed Enclosure: Design a stable mechanical structure for mounting all components.
- Use Brushless Motors: Upgrade from servo motors to brushless motors with encoders for smoother motion.
- Data Logging: Store orientation and servo data on an SD card or cloud database for analysis.

XIII. Conclusion

This project successfully demonstrates the design and implementation of a 2-axis gimbal stabilizer using an Arduino Nano ESP32, MPU6050 sensor, and servo motors, with added IoT functionality through the Arduino Cloud. It serves as a practical example of how low-cost components can be used to build an effective stabilization system that can also be monitored remotely. The fusion of mechanical stabilization with cloud-based visualization makes this project both educational and practical. It opens the door to more complex applications, such as drone camera systems, autonomous robots, and real-time remote monitoring tools. With the addition of a custom 3D-printed chassis and integration with Arduino IoT Cloud, the gimbal stands as a practical and extensible solution for real-world stabilization tasks. This project is an excellent starting point for anyone interested in embedded systems, control theory, and IoT integration.

References

[1] Arduino, "Arduino IoT Cloud," [Online]. Available: https://www.arduino.cc/en/IoT/HomePage

[2] InvenSense, "MPU-6050 Datasheet," [Online]. Available: https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/

[3] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer, 2017, ch. 13.

[4] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[5] Espressif Systems, "ESP32 Technical Reference Manual," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

[6] T. Igoe, "Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists," McGraw-Hill, 2011.

[7] A. Bahga and V. Madisetti, *Internet of Things: A Hands-On Approach*. Universities Press, 2014.

[8] J. Stucker, "Additive manufacturing technologies: Technology introduction and business implications," *Journal of Technology Management & Innovation*, vol. 7, no. 3, pp. 1–7, 2012.