**Real-Time 3D Aircraft Visualization Project**

**Project Title**

**B2 Bomber Real-Time 3D Simulator: Sensor-Controlled Visualization Using ESP8266 and MPU6050**

**Project Overview**

This project demonstrates a real-time 3D visualization of a B2 Bomber aircraft model, controlled by physical movements detected from an MPU6050 inertial measurement unit (IMU) connected to an ESP8266 microcontroller. The sensor data (position, roll, pitch, and yaw) is transmitted via serial communication in JSON format and rendered in a Processing 3 graphical user interface (GUI). The system allows users to simulate aircraft maneuvers with adjustable parameters through UI sliders, making it ideal for interactive demos, educational tools, or portfolio showcases.

The visualization supports fullscreen mode on high-resolution displays (e.g., 3840x1080), model centering, smoothing for stable motion, and real-time trimming for calibration.

**Key Features**

* **Sensor Integration**: MPU6050 provides acceleration and gyroscopic data, processed with a complementary filter for accurate orientation and position estimation.
* **Wireless Data Transmission**: ESP8266 sends compact JSON packets over USB serial at 460800 baud for low-latency updates.
* **3D Rendering**: Processing 3D (P3D) engine loads and manipulates an OBJ model of the B2 Bomber, applying transformations based on sensor input.
* **User Interface**: ControlP5 library provides sliders for trimming position, rotation, zoom, and scale in real-time.
* **Smoothing and Constraints**: Lerp-based interpolation ensures smooth movements, with constraints to keep the model within view.
* **Fullscreen Support**: Utilizes Processing's fullScreen(P3D, SPAN) for immersive viewing on wide screens.

**Technologies Used**

* **Hardware**: ESP8266 microcontroller, MPU6050 IMU sensor.
* **Software**:
  + Arduino IDE for ESP8266 firmware (C++).
  + Processing 3 for GUI and 3D rendering (Java-based).
  + Libraries: Adafruit MPU6050 (Arduino), ControlP5 (Processing).
* **Data Format**: JSON for serial communication.
* **Development Environment**: Windows (tested on 3840x1080 resolution).

**How It Works**

1. **Sensor Data Acquisition**: The MPU6050 reads acceleration and gyro values, which are integrated to compute velocity, position, roll, pitch, and yaw using a complementary filter and damping.
2. **Serial Transmission**: ESP8266 formats the data as JSON (e.g., {"posX":0.123,"posY":-0.456,"posZ":0.789,"roll":12.34,"pitch":-5.67,"yaw":8.90}) and sends it over serial at ~100 Hz.
3. **Processing Visualization**:
   * Parses incoming JSON in serialEvent().
   * Applies smoothing (lerp) to sensor values for stable animation.
   * Centers the OBJ model using bounding box calculations.
   * Transforms the model with sensor-driven translation and rotation.
   * UI sliders allow real-time adjustments for calibration and viewing.
4. **Fullscreen Mode**: The sketch runs in fullscreen, spanning multiple monitors if needed, with dynamic camera positioning.

**Setup and Installation**

**Hardware Requirements**

* ESP8266 board (e.g., NodeMCU).
* MPU6050 sensor module.
* USB cable for serial connection.

**Software Setup**

1. **Arduino IDE**:
   * Install ESP8266 board support via Boards Manager.
   * Install Adafruit MPU6050 and Adafruit Sensor libraries.
   * Upload the provided Arduino sketch to ESP8266.
2. **Processing 3**:
   * Download from [processing.org](https://processing.org/).
   * Install ControlP5 library via Tools > Manage Tools > Libraries.
   * Place your b2.obj model file in the sketch folder.
   * Run the Processing sketch.

**Running the Project**

1. Connect ESP8266 to your PC via USB.
2. Upload the Arduino code.
3. Close Arduino Serial Monitor.
4. Run the Processing sketch.
5. Move the MPU6050 sensor to control the 3D aircraft model.
6. Use UI sliders to trim and adjust visualization.

**Code Snippets**

**Arduino Sketch (ESP8266 + MPU6050)**

(Attached as sketch\_aug30a.ino in query – full code for sensor reading and JSON output.)

**Processing Sketch (3D Visualization)**

(Attached as B2\_bomber\_veiw.pde in query – full code with UI, serial parsing, and model rendering.)

**Challenges and Solutions**

* **Model Centering**: Calculated bounding box center to ensure rotations pivot correctly.
* **Smooth Movement**: Applied lerp smoothing to prevent jitter from raw sensor data.
* **Fullscreen on Wide Screen**: Used fullScreen(P3D, SPAN) for immersive 3840x1080 view.
* **Serial Reliability**: High baud rate (460800) and JSON parsing for low-latency data transfer.

**Potential Improvements**

* Add wireless UDP transmission instead of USB serial.
* Integrate more advanced sensor fusion (e.g., Kalman filter).
* Export as standalone executable for demos.
* Add VR/AR support for immersive experiences.

**Links**

* GitHub Repository: [github.com/yourusername/b2-bomber-simulator](https://github.com/yourusername/b2-bomber-simulator) (Upload your code here).
* LinkedIn Post: Share this report with screenshots/videos of the running demo.

This project showcases embedded systems, sensor fusion, and interactive 3D graphics – perfect for IoT, robotics, or visualization portfolios. If you'd like to expand or modify, let me know!

1. <https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/84080527/f04508df-c018-4a61-bbc5-bc6a456d8ff2/sketch_aug30a.ino>
2. <https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/84080527/deedc42b-48ea-47b8-b7e5-90a80a9ef101/B2_bomber_veiw.pde>