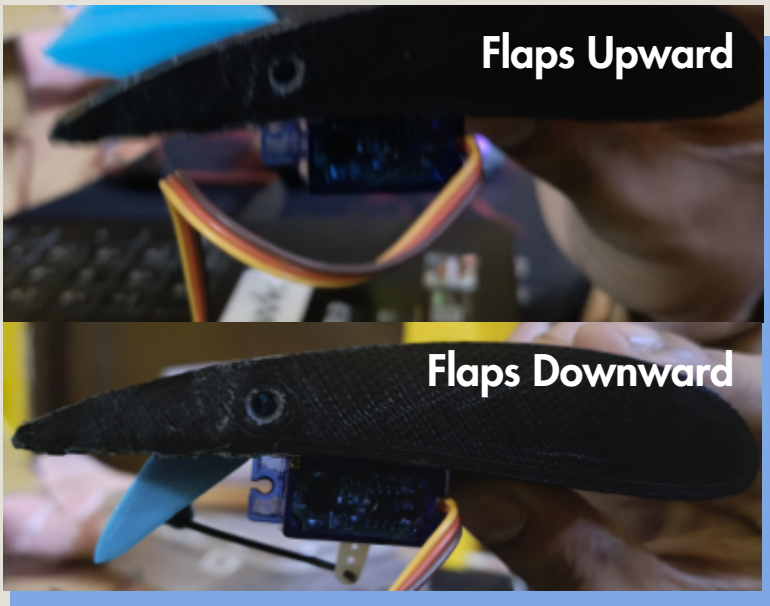
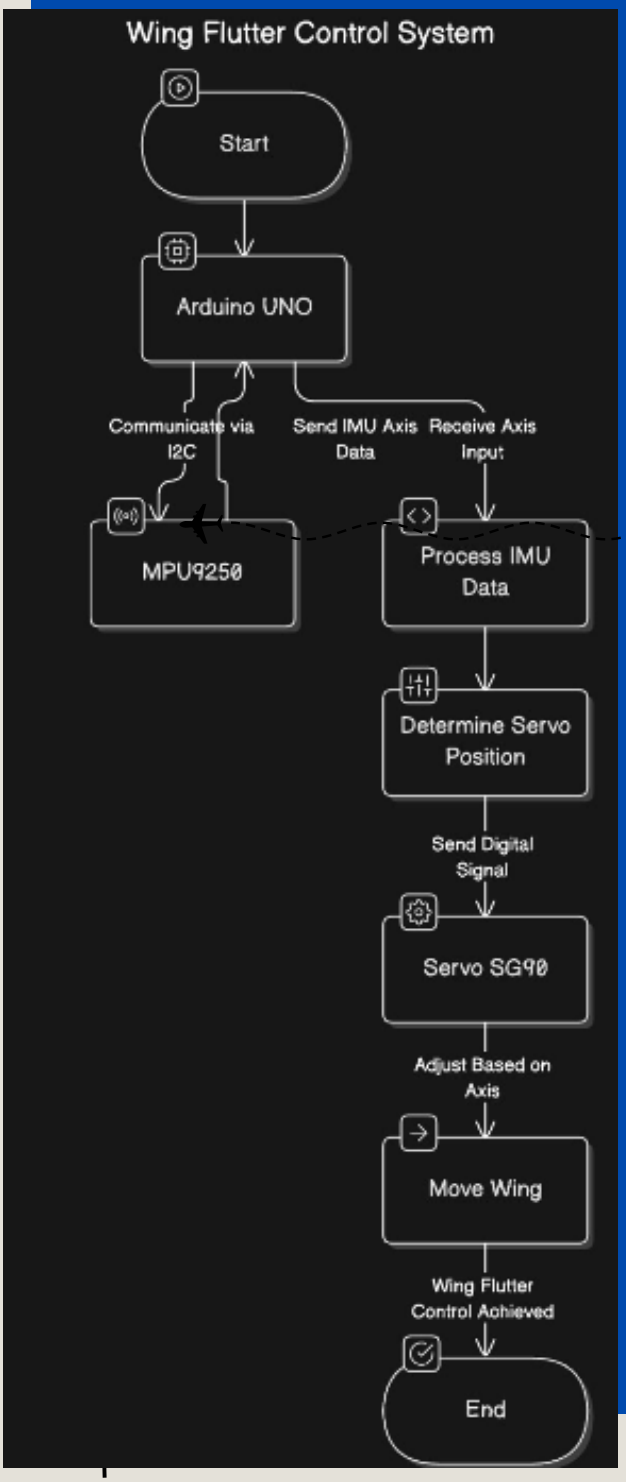
WING FLUTTER CONTROL

Wing flutter control is a vital area of research in aerospace engineering, aimed at addressing the self-excited oscillations that can occur in aircraft wings due to aerodynamic, structural, and inertial interactions. This phenomenon poses a significant threat to the structural integrity and safety of the aircraft, especially at high speeds or during extreme flight conditions. This project focuses on designing and implementing a control system to effectively suppress wing flutter, utilizing advanced techniques like sensor-actuator feedback and adaptive control algorithms. By simulating the dynamics of wing flutter and integrating real-time control strategies, this project seeks to enhance the stability, performance, and safety of modern aircraft.

Components needed:

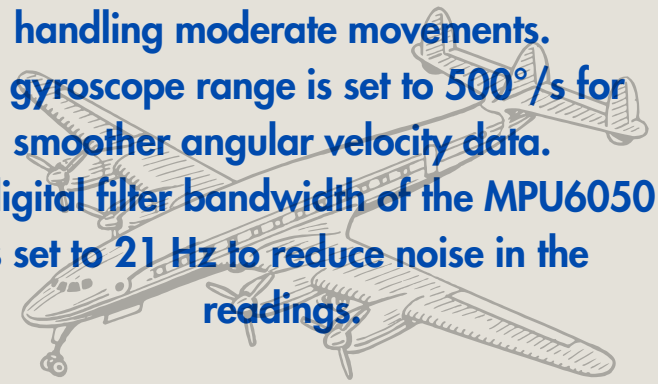
- Arduino Uno - Microcontroller to process data and control the servo.
- MPU6050 or MPU9250 - Inertial Measurement Unit (IMU) for reading 3-axis acceleration
- Servo 9G-Manually adjusts the wing
- NACA 2412 aerodynamic wings





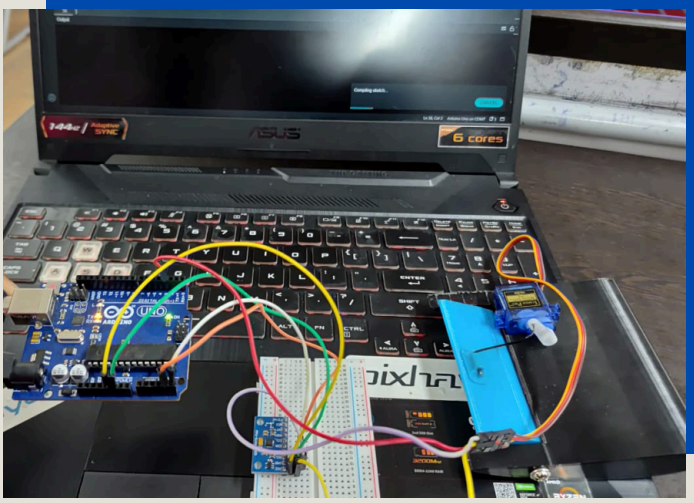
Setup Hardware Components:
Connect the MPU6050 (IMU) to the Arduino using I2C pins (SCL, SDA).
Attach the servo motor to pin 3 of the Arduino.
Provide regulated 5V power to both the Arduino and the MPU6050 sensor.

Initialize MPU6050:
The IMU (MPU6050) is initialized in the setup() function using the apple.begin() method.
The accelerometer range is set to ±8G for handling moderate movements.
The gyroscope range is set to 500° /s for smoother angular velocity data.
The digital filter bandwidth of the MPU6050 is set to 21 Hz to reduce noise in the readings.



Servo Initialization:
The servo is attached to pin 3 using the servo.attach(3) method.
The initial position of the servo is set to 0° using servo.write(0).

Sensor Data Reading:
In the loop(), the IMU readings are fetched using the apple.getEvent(&a, &g, &temp) function:
a: Accelerometer data (linear acceleration in x, y, and z axes).
g: Gyroscope data (angular velocity in x, y, and z axes).
temp: Temperature data from the IMU.



Processing Y-axis Data:

The y-axis acceleration (a.acceleration.y) is extracted.

The raw y-axis acceleration value (in units of m/s^2) is mapped to a servo angle using the map() function:

The range of acceleration is assumed to be -10 to 10 m/s^2.

This range is mapped to servo angles 0° to 180°.

Servo Control:

The calculated servo angle is written to the servo using servo.write(value).

The servo motor physically moves to the corresponding angle based on the mapped value.

Data Output:

The mapped servo angle is printed to the Serial Monitor using Serial.println(value) for debugging and observation

Working:

To set up an SG90 servo motor with an Arduino Uno and an MPU9250 sensor, you'll first connect the MPU9250 to the Arduino using I2C, linking SDA to A4 and SCL to A5, while powering it through the 3.3V pin. The signal pin of the SG90 servo goes to Digital Pin 9 on the Arduino, and you'll also connect its VCC and GND pins to the 5V and GND on the board. In the Arduino IDE, you'll need to include the libraries for both the MPU9250 and the servo. The code initializes the MPU9250 to gather pitch, roll, or yaw data, and then it maps the chosen axis data, like pitch, to a range that the servo can understand (from 0 to 180 degrees). This mapped value is then used to adjust the servo's angle, making it tilt according to the orientation detected by the MPU9250. Once everything is set up, upload the code to the Arduino Uno, enabling the servo to respond to the tilt of the MPU.