

TED UNIVERSITY

**CMPE 491 / SENG 491 Senior Project**

**Center of Gravity Detection, Information, and Balancing System in Aircraft**

**Analysis Report**

**28.11.2025**

**Team Members:**

Yasin Anber, Computer Engineering

Sude İpekci, Computer Engineering

Selin Göç, Computer Engineering

Elif Sude Memiş, Software Engineering

**Supervisor:**

Prof. Dr. Hakkı Gökhan İLK

**Jury Members:**

Dr. Ali BERKOL

Dr. Ayşe Yasemin SEYDİM

**metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.**

1. **Introduction**

This report presents a detailed examination of the project titled "Center of Gravity Detection, Information, and Balancing System in Aircraft". The primary goal of this project is to develop a software system that processes real-time data from load cells. Using an HX711 (ADC) to convert analog signals into digital data for the microcontroller, the system employs an algorithm to determine the ideal center of gravity. Additionally, the system uses servo motors to move the balancing weight to the ideal coordinate in order to return the aircraft's center of gravity to its optimal location when a deviation from the optimal center of gravity is detected. In keeping with this goal, the project entails creating, programming, designing, and putting into practice a system that determines the center of gravity and continuously keeps it at its ideal location. Throughout the project, we will go through phases of system design, hardware and software integration, and testing in order to develop a reliable and efficient CG balancing system that enhances flight safety, performance, and fuel efficiency.

1. **Current system**

Currently, the majority of methods used to determine an aircraft's center of gravity (CG) are manual; fully automated CG measuring or balancing devices are rarely used outside of research purposes [4]. During operation, users place the aircraft on a mechanical balancing platform or use ruler-like reference points to determine the CG position [1][2]. While these methods have been used for a long time, they have some critical limitations.

First, manual balancing faces some limitations in measurement accuracy because it relies on human interpretation of pitch, alignment, and symmetry. According to the FAA, even small CG deviations can significantly alter longitudinal balance, but these deviations are difficult to manually detect due to their coarse resolution and user-dependent handling [1].

Secondly, there is no mechanism for dynamic or automatic correction, which means users must repeatedly reposition loads or weights during preliminary aircraft balancing operations; Raymer describes this process as iterative and error-prone during preliminary aircraft balancing operations [2].

Thirdly, each adjustment requires lifting, repositioning, and re-measuring the aircraft, increasing the repetitive workload and making the overall process time-consuming. Fourth, manual CG methods lack real-time monitoring capabilities; Any change in payload, fuel, sensor package, or modular structure requires repeating the entire CG control from beginning to end, which is particularly problematic for small UAVs, whose mass distributions frequently change during flight [3].

Finally, measurements are highly variable because the manual approach can be different for each user: two different users may obtain different CG estimates under the same conditions. The reasons for this discrepancy are differences in judgment, visual alignment, and applied technique[1].

In general, the lack of automatic detection and computational support makes the current CG determination process uncertain, repetitive, inefficient, and difficult to standardize. For applications where precision and repeatability are critical, such as experimental aircraft, small-scale UAV testing, and model aircraft prototyping, the need for an automated CG measurement and compensation system becomes evident [4].

1. **Proposed system**
   1. **Overview**

The Aircraft Center of Gravity (CG) Balancing System is an automated solution for model aircraft, designed for precise measurement, real-time monitoring, and correction of the CG location. This reliable system achieves CG determination and adjustment by integrating load-cell sensors, an HX711 signal amplifier, a microcontroller, and an X–Y servo-driven balancing mechanism. aircraft

The main goal of the system is to reduce the mistakes, inefficiencies, and time-consuming trial-and-error process that come with manual CG measurement. By collecting precise weight data from three load cells placed beneath the aircraft platform, the system determines whether the aircraft is within the allowed stability region and computes the precise CG coordinates. The most critical design aspect of the project is to maintain the aircraft's center of gravity at the optimum point. To balance the center of gravity, a system design will be established that enables the precise movement of a counterweight. The system comprises a counterweight, two 360° rotating servo motors, and two lead screw mechanisms. Servo motors are precision control elements capable of rotating at specific angles; each converts rotational motion into linear motion by turning the lead screw to which it is connected[5]. In this way, the counterweight is shifted in a controlled manner along the X and Y axes, allowing the center of gravity to be adjusted with millimetric precision. The screw pitch determines the distance the weight advances when the screw is rotated one full turn, and the system maintains the weight's position by preventing reverse movement under external forces. The servo motor on the X-axis moves the weight in the horizontal direction, while the servo motor on the Y-axis moves it in the vertical direction. The system detects deviations in the center of gravity and aims to maintain the aircraft's center of gravity at the optimum point by moving the counterweight to the calculated (X,Y) coordinates. Thereby, the objective is for the aircraft to return to a stable equilibrium position. All in all, the recommended approach provides an automated, precise, and user-friendly platform that significantly reduces manual labor while ensuring consistent CG measurements.

* 1. **Functional Requirements**

FR1: The system shall receive weight data from the load cells and convert them into digital form using the HX711 module.

FR2: The system shall calculate the CG coordinates according to the input from multiple loadcell sensors in real time.

FR3: To maintain equilibrium, the system shall control servo motors so that they move a balancing weight along X and Y axes.

FR4: To provide real-time feedback to the user, the system shall display important information including the current total weight, individual load readings, and the calculated CG position on a graphical interface such as a computer dashboard or serial plotter.

FR5: The system shall store a preset optimal CG position and continuously compare it to the current CG. Then, if the deviation rises above a predetermined threshold, the system shall start corrective balancing.

FR6: Before operating, the system shall undergo a preliminary calibration procedure to zero-load cells and create precise baseline measurements.

* 1. **Nonfunctional Requirements**

NFR1: The system shall process and update CG data every 50 milliseconds or less to ensure responsive real-time balancing.

NFR2: The system shall detect and correct CG deviations within ±2% of the actual balance point, preserving stable equilibrium under normal operating conditions.

NFR3: The system shall continue to function consistently under variable load conditions without data loss.

NFR4: The system shall run at a voltage lower than 12V DC, and all moving parts must have mechanical limits to prevent damage or injury. In this prototype, the operating voltage is limited to 5V DC to ensure safe and stable operation of the Arduino and HX711 module.

NFR5: The graphical interface shall provide visual indicators for sensor data, balance status, and system alerts.

NFR6: The system shall have modular software and hardware components so that individual parts can be replaced or updated without requiring a complete system redesign.

NFR7: The design shall facilitate integration with autonomous flight control systems or expansion to larger aircraft platforms in later versions.

NFR8: The system shall operate reliably under indoor laboratory conditions (temperature between 15–35°C and humidity ≤70%), minimizing drift caused by ambient environmental changes.

* 1. **Pseudo requirements**

### PR1: The system must be implemented using an Arduino-based microcontroller (Arduino UNO or an equivalent ATmega328P-based board), as required by the laboratory hardware available for the project.

### PR2: Load measurement must rely exclusively on strain-gauge load cells coupled with the HX711 24-bit ADC module, since this is the designated sensor interface supported by the project environment.

### PR3: All electronic components must operate within a 5V DC supply except servo motors, consistent with Arduino’s voltage limitations and the available laboratory power infrastructure.

### PR4: Software development must be carried out using the Arduino IDE and written in Arduino C/C++, as required by the course tools and microcontroller compatibility.

### PR5: The system's Center of Gravity (CG) output data, along with all diagnostic and status information, shall be communicated to the Operator through a dedicated Graphical User Interface (GUI) for real-time visualization and monitoring, ensuring dynamic data display takes precedence over raw numerical logging.

### PR6: The system shall be implemented and tested on a 3D-printed prototype aircraft.

### PR7: Only components available in the laboratory inventory or low-cost hobby-grade parts may be used, restricting actuator power, sensor quality, and platform size.

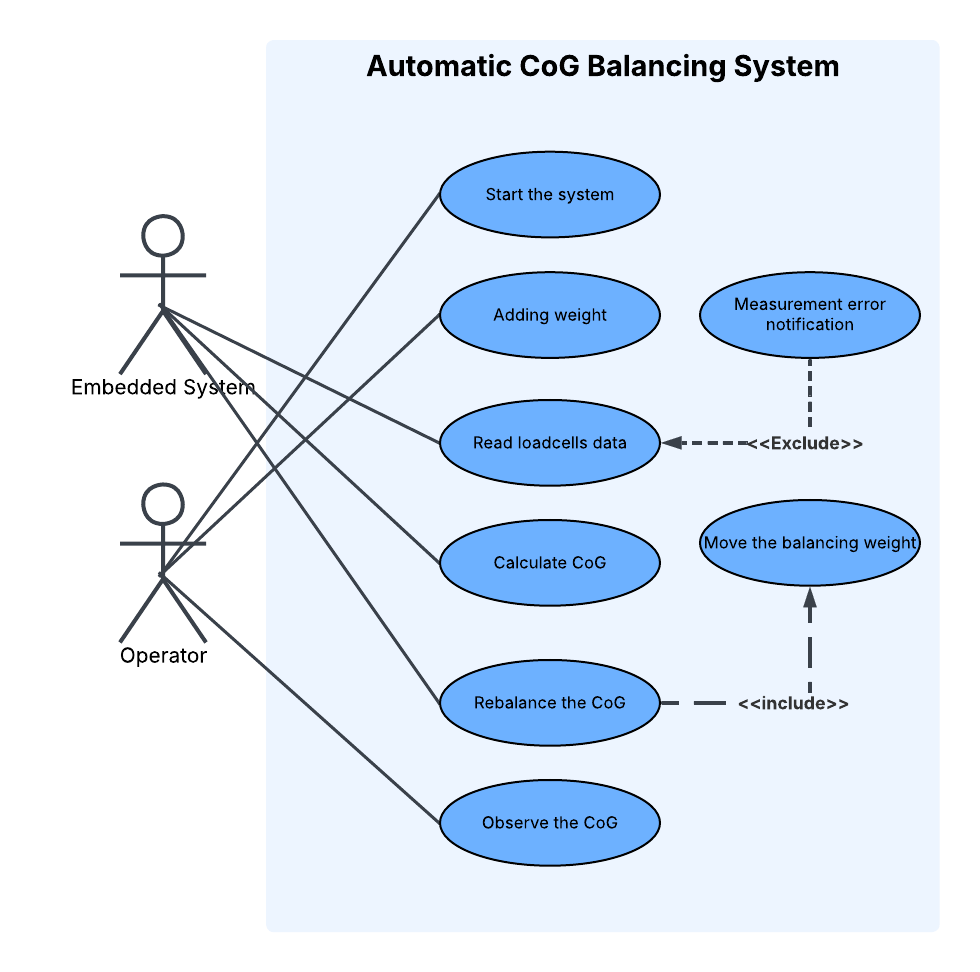
* 1. **System models**
     1. **Scenario**

**Scenario Name:** Pre-Flight Balancing After New Loading

**User:** Technical Personnel (Aircraft Operator)

**Description:** Technical personnel load new weights onto the aircraft. The center of gravity must be checked for flight safety. The aircraft is placed on the balancing platform. The system operates automatically, reading data from the load cells. The Arduino-based embedded system calculates the instantaneous center of gravity (CG\_current). The calculated CG deviates from the predefined optimum point (CG\_opt). Using the balancing algorithm, the system calculates the distance (x,y) the balancing counterweight must move to correct this deviation. With this command, the servo motors drive the worm gear mechanism. The counterweight moves, restoring the center of gravity to the optimum point with millimeter precision. The display interface displays the Balance Achieved message and the stable CG coordinates. The technical personnel are ready for flight.

* + 1. **Use case model**



The provided Use Case Diagram illustrates the functional requirements and the interactions between the actors and the **Automatic Center of Gravity (CoG) Balancing System** for an aerospace application.

### **Actors**

1. **Operator (Technical Personnel):** The human user responsible for initiating the process, loading the aircraft, and monitoring the system's status and final output.
2. **Embedded System (Arduino):** The microcontroller platform running the control algorithm (C/C++). It is the core intelligence that performs calculations, takes measurements, and executes the autonomous balancing action.

## **Use Cases and Flow Descriptions**

### **1. Start the system**

* **Purpose:** To activate the CoG balancing platform, initialize the embedded system to a predetermined datum position.
* **Actors:** Initiated by the Operator and executed by the Embedded System.

### **2. Adding weight**

* **Purpose:** To represent the physical change in the aircraft's load configuration, such as adding cargo, fuel, or new equipment, which necessitates a CoG check and potential rebalancing.
* **Actors:** Performed by the Operator.

### **3. Read loadcells data**

* **Purpose:** To acquire the real-time weight data from the Full-Bridge strain gauge load cells placed under each wheel of the aircraft. This weak analog signal is amplified and converted to 24-bit digital data by the HX711 ADC module before being sent to the Arduino.
* **Actors:** Performed by the Embedded System.

### **4. Calculate CoG**

* **Purpose:** To use the collected loadcell weight data and the known coordinates of the load cells to compute the aircraft's current Center of Gravity coordinates (CG\_x, CG\_y) using the moment principle.
* **Actors:** Performed by the Embedded System.

### **5. Rebalance the CoG**

* **Purpose:** This is the core function of the system. It detects the deviation (Delta x, Delta y) between the calculated current CoG (CG\_current) and the optimal target CoG (CG\_opt). It then calculates the required displacement for the counterweight to create an opposing moment and bring the CoG back to the ideal setpoint.
* **Actors:** Managed and executed by the Embedded System.
* **Relationship:** This use case includes the *Move the balancing weight* scenario.

### **6. Observe the CoG**

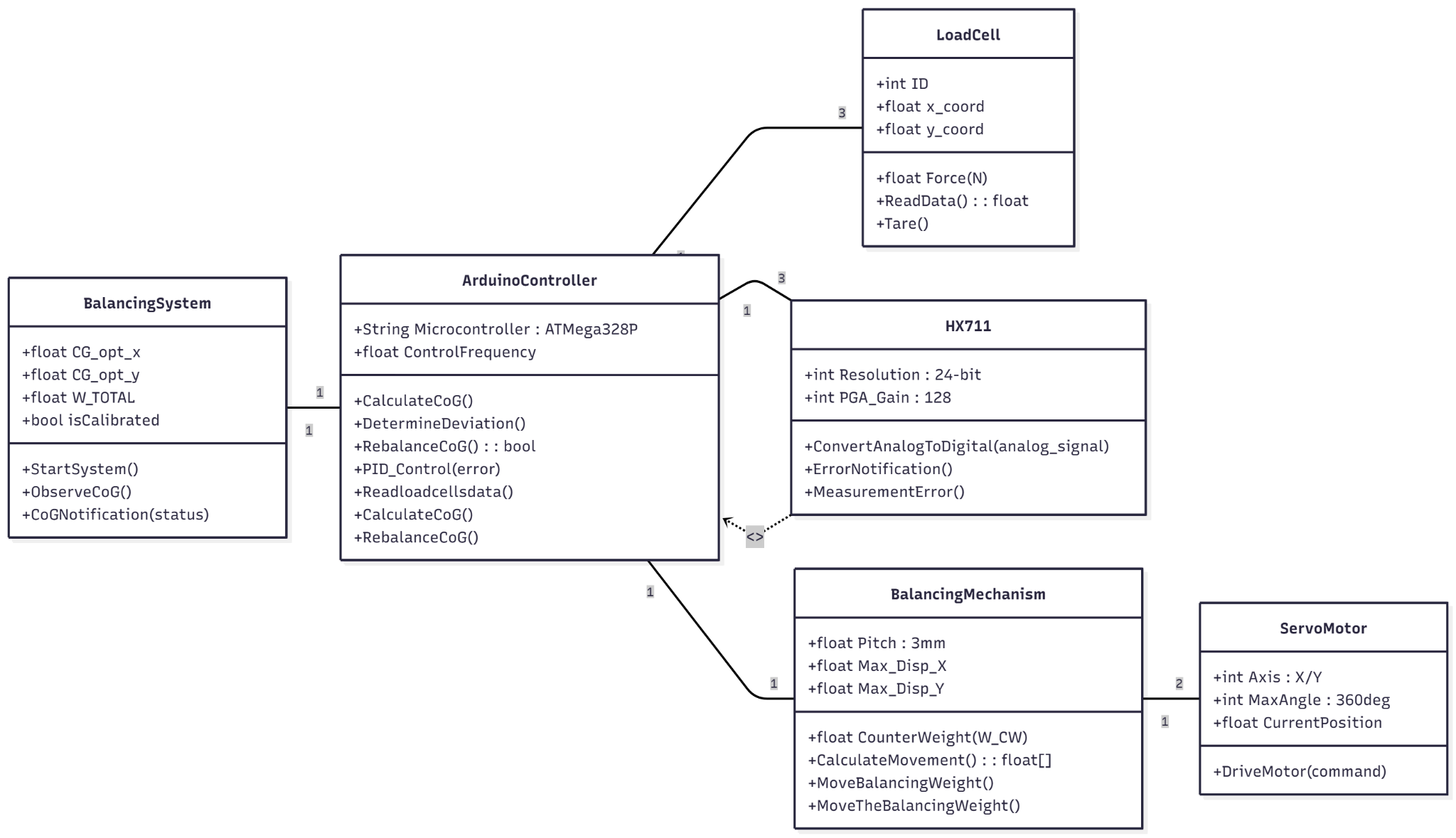
* **Purpose:** To display the vital operational data, including the instantaneous CoG location, total weight (W\_total), and the stabilization status, to the user via a computer interface (e.g., Serial Plotter).
* **Actors:** The Operator receives the value; the Embedded System provides the data.

### **7. Move the balancing weight**

* **Purpose:** To physically shift the counterweight using a mechanical system consisting of two servo motors and lead screw mechanisms. This movement is precisely controlled in the X and Y axes based on the distance calculated by the rebalancing algorithm. The movement is governed by a control algorithm to ensure high speed and minimal oscillation.
* **Actors:** Performed by the Embedded System.

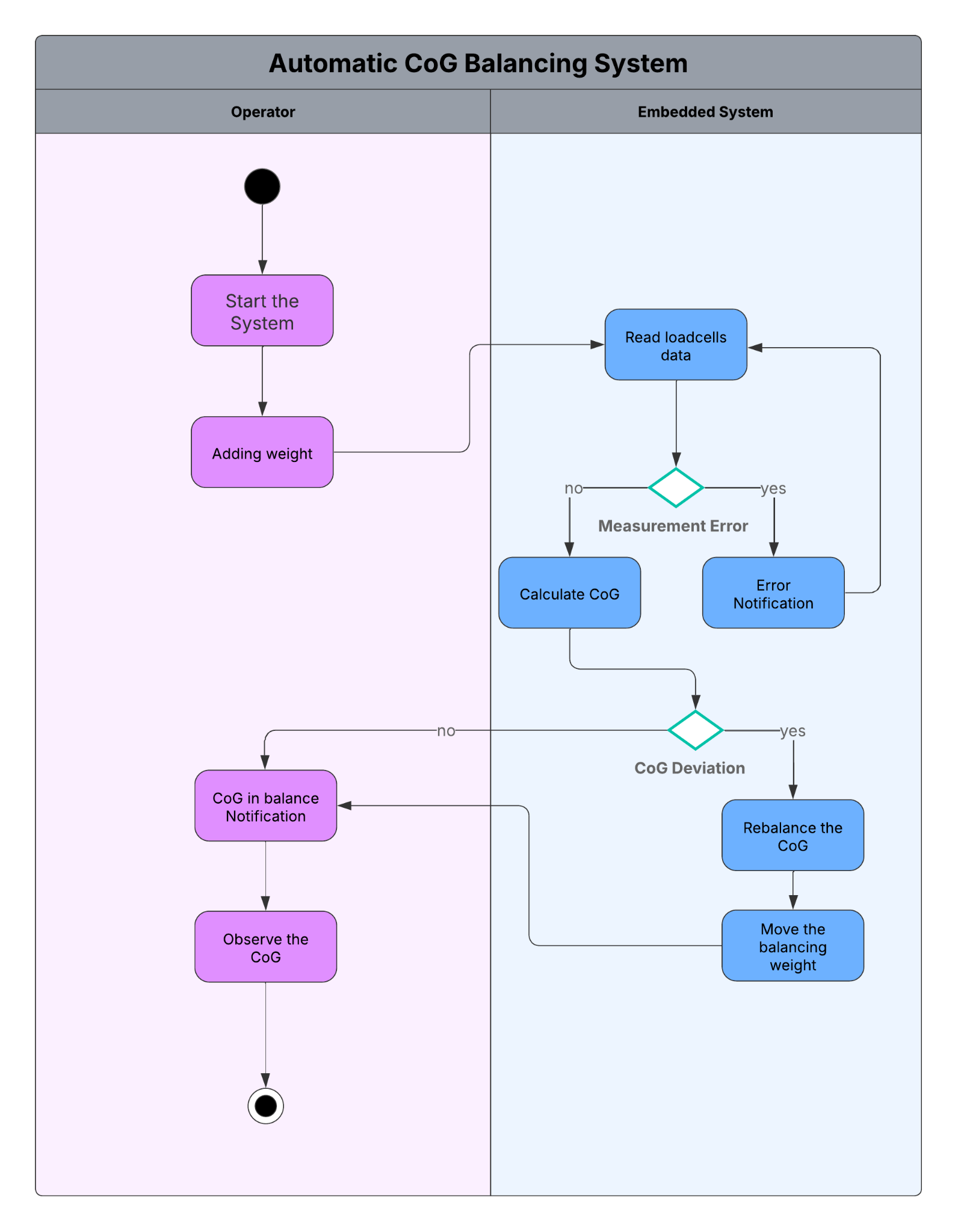
### **8. Measurement error notification**

* **Purpose:** To interrupt the normal flow and notify the operator if an unexpected error, excessive noise, or malfunction is detected in the load cells, HX711, or data reading process.
* **Actors:** Performed by the Embedded System.
* **Relationship:** This use case excludes the Read loadcells *data* scenario, meaning a data reading error is an exceptional condition that prevents the system from proceeding with a normal measurement and calculation cycle.
  + 1. **Object and class model**



* + 1. **Dynamic models**

**1. Activity diagram**



**2. Sequence diagram**

metin, ekran görüntüsü, yazı tipi, diyagram içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

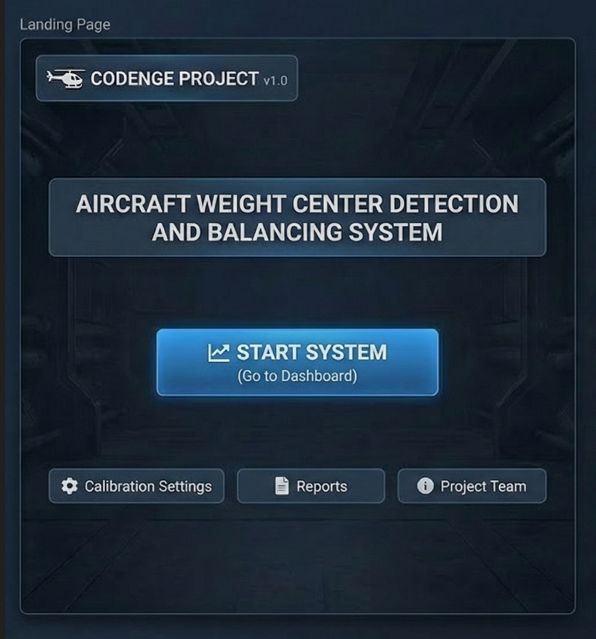
metin, diyagram, ekran görüntüsü, paralel içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**3.5.5 User interface - navigational paths and screen mock-ups**

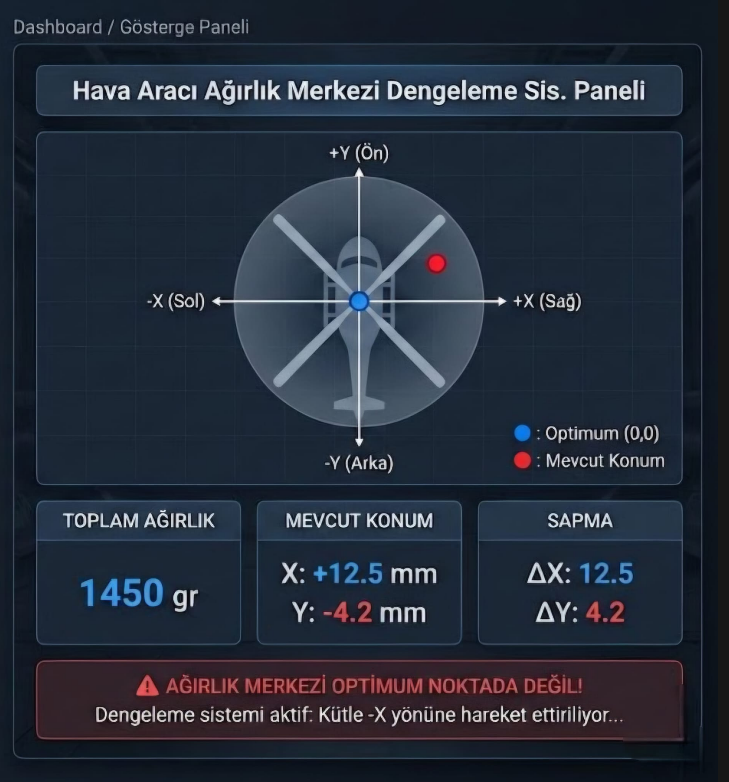
The proposed two-page architecture, featuring a Landing Page for initialization and a Dashboard for real-time monitoring, represents the optimal engineering approach for the Codenge project by strictly enforcing a functional separation between static configuration and dynamic operation. This "Single Pane of Glass" design philosophy minimizes cognitive load and navigation latency, ensuring that critical telemetry data such as center of gravity coordinates, total weight, and system status are instantly accessible in a centralized view—vital for operational safety and rapid decision-making. Furthermore, this streamlined architecture aligns perfectly with the scope of a Proof-of-Concept (PoC) prototype by prioritizing algorithmic performance and real-time visualization over unnecessary navigation complexity, thus demonstrating a clear focus on the system's core engineering objectives.

**1. Landing Page**

****

This interface serves as the secure initialization point for the system. It isolates operational controls from configuration settings to prevent accidental interference. The screen provides direct access to the active monitoring dashboard while offering secondary access to calibration, historical reports, and project information via modal pop-ups. The 'START SYSTEM' button initiates the operational process and directly redirects the user to the Dashboard for real-time data monitoring."

**2. Real-Time Center of Gravity Detection, Monitoring, and Balancing Panel (Dashboard)**

****

The core operational panel displays real-time telemetry data received from the load cells. It features a dynamic visual coordinate system that tracks the actual Center of Gravity (CoG) relative to the optimum origin (0,0). The dashboard includes numerical indicators for total weight and deviation vectors (ΔX, ΔY), alongside a status alert system that notifies the operator of ongoing balancing actions.

**3. System Calibration and Sensor Configuration Interface**

****

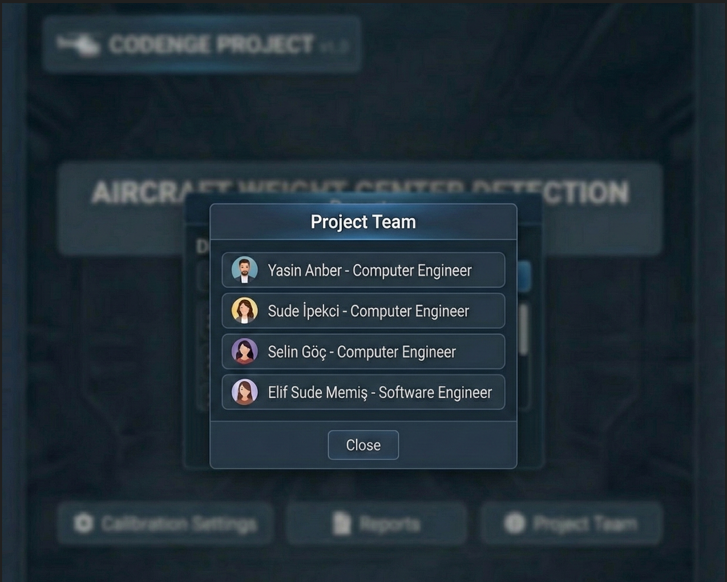
This configuration modal is designed for pre-operation setup. It allows the operator to perform essential maintenance tasks such as taring the load cells (zeroing), defining the geometric center of the aircraft, and adjusting the response sensitivity of the servo motors to ensure measurement accuracy.

**4. System Balancing History and Operational Reports Log**

****

A data retrieval interface that lists timestamped logs of past balancing events. It records critical data points such as weight changes, detected deviations, and the specific corrective actions taken by the automatic balancing system, enabling post-flight performance analysis.

**5. Project Credits and Developer Team Information Screen**

****

An informational pop-up displaying the project identity under the TUSAŞ LIFT-UP program. It lists the "Codenge" development team members highlighting their respective engineering roles.

1. **Glossary**

**Center of Gravity (CG):** The point where the aircraft’s weight is balanced. It determines stability and must be measured accurately.  
**Load Cell:** A sensor that measures weight using strain gauges. Four load cells are used to detect weight distribution.  
**HX711:** A high-precision ADC module that amplifies and converts load-cell signals into digital data for the microcontroller.  
**Arduino / Microcontroller:** The embedded controller (e.g., Arduino UNO) that reads sensor data, calculates the CG, and drives the servo motors.  
**Servo Motor:** A motor capable of precise rotation. It moves the balancing mass along the X and Y axes.  
**Balancing Mechanism:** The assembly of servos, threaded rods, and a movable weight used to correct the CG position.  
**Lead Screw:** A rod that converts motor rotation into linear movement for accurate positioning of the balancing mass.  
**Platform:** The surface holding the aircraft. It houses the load cells and supports CG measurement.  
**CG Threshold Region:** The acceptable range where the CG should remain for stable operation.  
**Calibration:** The process of zeroing and adjusting load-cell readings for accurate measurements.  
**Scenario:** A short narrative describing how the user interacts with the system in a given situation.  
**Actor:** Any external entity that interacts with the system. Here, the primary actor is the user.  
**Use Case:** A structured description of a goal-oriented interaction between the user and the system.  
**Dynamic Model:** A representation of system behavior over time, such as sensor reading and servo movement sequences.

1. **References**

[1] Federal Aviation Administration. *Aircraft Weight and Balance Handbook* (FAA-H-8083-1B). U.S. Department of Transportation, 2016.

[2] Raymer, D. P. *Aircraft Design: A Conceptual Approach*, 6th ed. American Institute of Aeronautics and Astronautics, 2018.

[3] Anderson, J. D. *Introduction to Flight*, 8th ed. McGraw-Hill Education, 2017.

[4] Abdelhafiz, A., & El-Sayed, A. “Automatic Center of Gravity Measurement System for Small Aircraft.” *International Journal of Aerospace Engineering*, 2020, pp. 1–8.

[5] Soneji, K. P., Patel, H. K., & Karkari, S. (2013). Vacuum linear feed through drive using stepper motor and lead screw. 2013 Nirma University International Conference on Engineering (NUiCONE). IEEE. <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6780185>