



TESTLAB

AI-Supported Flight Systems Test Automation Platform for the System Integration Laboratory

Project Team: Sinem Ünsal, Melike Demir, Adahan Evrensel, Baran Ural Çavdar

Academic Advisor: Asst. Prof. Dr. Talha Karadeniz

Project Partner: TUSAŞ - Lift Up Program

www.tusas.com



1. About the Organization

This project was developed for the System Integration Laboratory (SEL) of Turkish Aerospace (TUSAŞ) under the TUSAŞ Lift Up graduation program. TUSAŞ is Turkey's leading aerospace company, designing and producing helicopters, fixed-wing aircraft, and avionics systems for both national and international programs.

Within the SEL environment, verifying that flight system displays respond correctly is an essential part of avionics testing. TestLab is designed to extend and strengthen this process by adding a simulation-based layer of automated, visual, and logical validation, making display verification faster, more repeatable, and easier to trace across many parameters and fault combinations.

2. Project Subject & Objective

Context: A modern helicopter cockpit display presents dozens of safety-critical parameters - engine torque, turbine temperature, NG/NP speeds, oil pressures, fuel, hydraulic, electrical, environmental data, and WCA messages - on a single unified screen. Each parameter must be shown with the correct value, unit, color, and warning behavior.

Problem: Manually validating the many critical parameters on a flight display is time-consuming, repetitive, and error-prone, and is difficult to trace visually - especially as coverage must extend to numerous fault and multi-system failure combinations.

Objective: To develop a simulation-based platform that reproduces helicopter system behavior, visualizes it on a cockpit-style display, and automatically validates the display's response:

- Real-time system simulation
- Cockpit-style unified display
- Scenario-based fault injection
- Warning / Caution / Advisory (WCA) validation
- Screenshot-based evidence and automated reporting

3. Work Done - I

System architecture and core components:

- **Real-time cockpit simulation (PyQt5)** - eight parameter groups covering twin engines, fuel, hydraulic, electrical, environmental, rotor/transmission, and system status, each shown with value, unit, color-coded state, and a proportional bar gauge.
- **Deterministic catalog of 20 fault scenarios** - across ENGINE, FUEL, ELECTRICAL, ENVIRONMENTAL, HYDRAULIC, ROTOR, CASCADE, and NOMINAL categories, each with defined injected values and expected outcomes.
- **WCAStore** - priority-based WCA management (WARNING > CAUTION > ADVISORY) with in-place severity escalation, so the most critical condition is always shown first.
- **FaultGate** - a time gate that raises a WCA only after a parameter stays abnormal for a minimum duration, preventing false alarms caused by transient spikes.
- **Color-coded state** - Green: normal, Yellow/Orange: caution, Red: warning.

4. Work Done - II

Visual validation pipeline - checks both the internal logic and what the operator sees on screen:

- **Deterministic core** - logic checks (FaultGate timing, WCAStore priority) plus OpenCV HSV color analysis (~5 ms, offline); the fraction of red/yellow/green pixels is compared against fixed thresholds.
- **ML-supported layer** - YOLOv8 ML screenshot classification (~10 ms); predictions below 0.6 confidence are flagged for manual review.
- **Optional semantic review** - AI Vision (~2 s) answers scenario-specific questions for additional inspection, but does not replace the deterministic checks.

Additional capabilities:

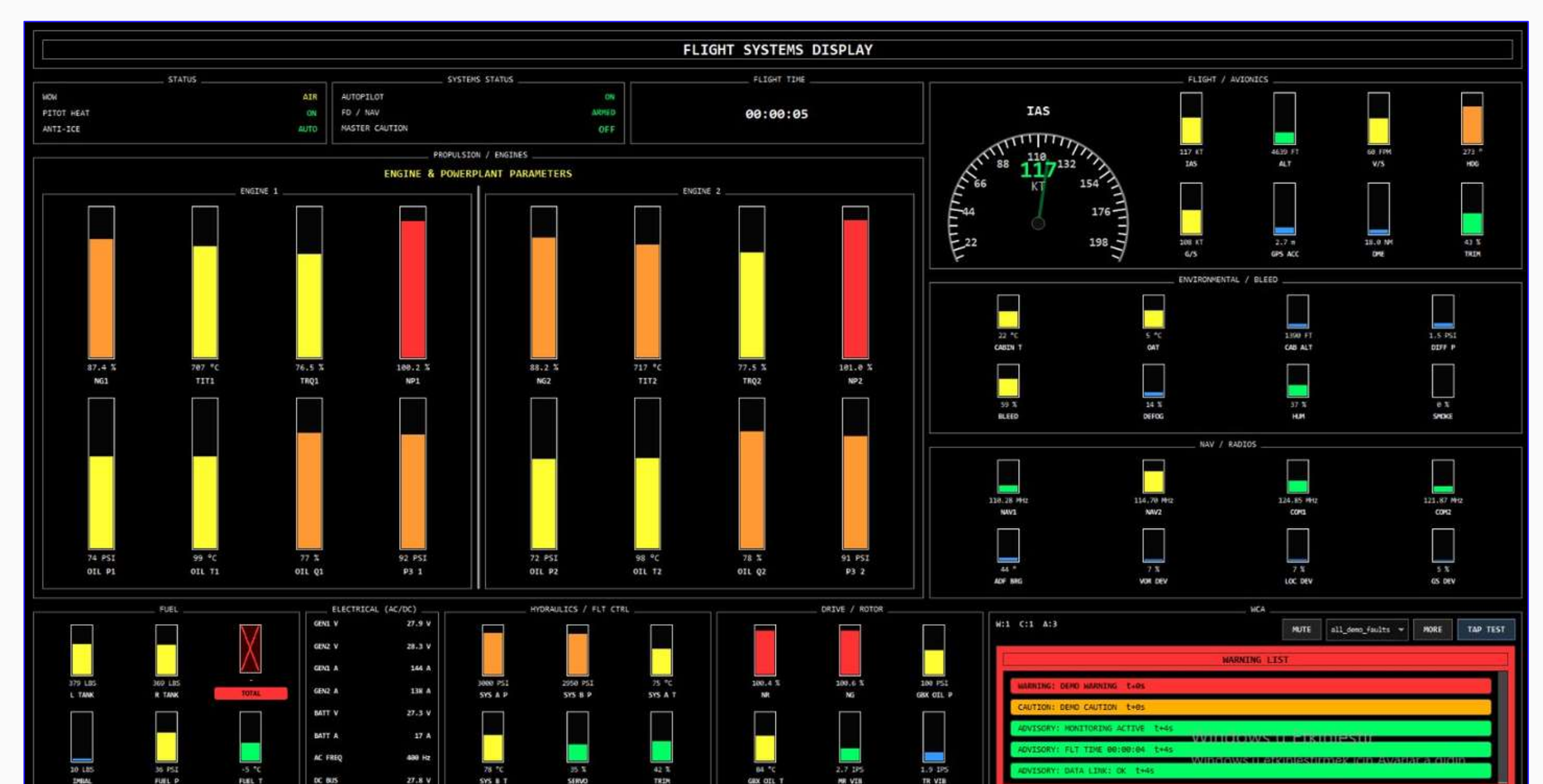
- Automatic screenshot evidence for every scenario, linked to its scenario ID for traceability
- Automated HTML reports (scenario, expected/actual result, PASS/FAIL, captured screenshots, detected mismatches)
- ML training pipeline: 4× data augmentation, 80/20 split (seed=42), early stopping, versioned model output
- Real-time voice assistant; announces active WCA conditions in a non-blocking subprocess, mutable from the UI

Technologies: Python, PyQt5, pytest, OpenCV, YOLOv8 (Ultralytics)



5. Results

- All 20 defined fault scenarios passed functional validation.
- WCA entries and the Master Caution indicator behaved correctly in every scenario, with screenshot evidence captured automatically.
- OpenCV HSV (~5 ms) and YOLOv8 (~10 ms) both stayed below their performance targets.
- FaultGate produced no false WCA alerts from single-tick transient values.
- Cascading scenarios produced multiple simultaneous WCA entries, correctly prioritized by WcaStore.
- The result is an end-to-end, repeatable, and traceable validation workflow.



Developed interface - Flight Systems Display

PROJECT TEAM



Sinem Ünsal



Melike Demir



Adahan Evrensel



Baran Ural Çavdar