Project Specification: Live FSAE EV Calculation Suite

Real-Time Engineering Dashboard for ITU Racing

Development Team Specification

January 22, 2026

Contents

1 System Overview **2**

2 Module A: Pre-Charge & Discharge Safety Calculator **2**
- 2.1 A.1 Math & Logic . . . 2
- 2.2 A.2 User Inputs (Controls) . . . 2
- 2.3 A.3 Real-Time Visuals (Outputs) . . . 3

3 Module B: Battery Endurance & Discharge Simulator **4**
- 3.1 B.1 Math & Logic . . . 4
- 3.2 B.2 User Inputs (Controls) . . . 4
- 3.3 B.3 Real-Time Visuals (Outputs) . . . 4

4 Module C: Wheatstone Bridge Balancer **5**
- 4.1 C.1 Math & Logic . . . 5
- 4.2 C.2 User Inputs (Controls) . . . 5
- 4.3 C.3 Real-Time Visuals (Outputs) . . . 5

5 Module D: Filter Designer (Signal Processing) **6**
- 5.1 D.1 Math & Logic . . . 6
- 5.2 D.2 User Inputs (Controls) . . . 6
- 5.3 D.3 Real-Time Visuals (Outputs) . . . 6

System Overview

Goal: Create a modular, reactive web application for validating electrical designs in real-time. This system is designed to replace static spreadsheets with interactive tools that allow for immediate design verification and "what-if" analysis during design judging.

Key UX Principle: *"Zero-Latency Feedback."* Users must see charts update instantly as they adjust parameters (sliders/inputs). There should be no "Calculate" buttons; the system is always live.

Recommended Tech Stack:

- **Backend/Logic:** Python (NumPy/SciPy for engineering math).
- **Frontend/UI:** Streamlit (preferred for speed) or React + Plotly.js.

—

Module A: Pre-Charge & Discharge Safety Calculator

Objective: Visualize the voltage ramp-up (Pre-charge) and ramp-down (Discharge) to ensure compliance with FSAE rules (specifically TSAL logic and HV disconnect safety).

A.1 Math & Logic

The core physics relies on the RC Circuit Time Constant (τ).

- **Time Constant:**
$$\tau = R \cdot C$$
- **Charging Equation (Pre-charge):**
$$V(t) = V_{bus} \cdot \left(1 - e^{-\frac{t}{\tau}}\right)$$
- **Discharging Equation:**
$$V(t) = V_{bus} \cdot e^{-\frac{t}{\tau}}$$
- **Energy Stored:**
$$E = \frac{1}{2}CV^2$$

A.2 User Inputs (Controls)

1. **Bus Voltage (V_{bus}):** Slider [0V – 600V]. *Default: 400V*
2. **Total Capacitance (C_{bus}):** Input Field (μF). *Default: 1000μF*
3. **Pre-Charge Resistor (R_{pre}):** Slider [100Ω – 5kΩ].
4. **Discharge Resistor (R_{dis}):** Slider [1kΩ – 50kΩ].

A.3 Real-Time Visuals (Outputs)

- **Dynamic Graph:** Plot Voltage (y) vs. Time (x).
  - **Trace A (Green):** Charging Curve.
  - **Trace B (Red):** Discharging Curve.
  - **Overlays:** Horizontal dashed line at **95% Voltage** (Pre-charge complete) and **60V** (Safe voltage threshold).
- **Safety Check Component:**
  - If discharge time to 60V $>$ 5.0 seconds $\rightarrow$ **DISPLAY WARNING:** "FAIL: Discharge too slow per EV.5.5."
  - If Pre-charge peak power (I^2R) $>$ Resistor Power Rating $\rightarrow$ **DISPLAY WARNING:** "Resistor thermal limit exceeded."

Module B: Battery Endurance & Discharge Simulator

Objective: Estimate vehicle runtime and thermal load during an Endurance event to optimize strategy.

B.1 Math & Logic

- **Total Pack Capacity:**

$$C_{pack} = C_{cell} \cdot N_{parallel}$$

- **Runtime Estimate:**

$$t_{run} = \frac{C_{pack} \cdot (SoC_{start} - SoC_{end})}{I_{avg}}$$

- **Heat Generation (Joule Heating):**

$$P_{heat} = I_{rms}^2 \cdot R_{internal} \cdot N_{total\_cells}$$

B.2 User Inputs (Controls)

1. **Cell Configuration:** Inputs for Series (S) and Parallel (P).
2. **Cell Parameters:** Capacity (Ah), Internal Resistance ($m\Omega$).
3. **Load Profile:**
   - **Slider:** Average Current (A).
   - **Slider:** Peak Current (A) (used for heat spike estimation).
4. **Simulation Time:** Slider [0 – 60 mins].

B.3 Real-Time Visuals (Outputs)

- **Dual Y-Axis Graph:**
  - **Left Y-Axis:** Pack Voltage (V) dropping over time.
  - **Right Y-Axis:** Estimated Pack Temperature ($^\circ C$) rising over time.
- **KPI Cards:**
  - **"Time to Empty":** e.g., "24 mins" (Color code Red if < 22 mins).
  - **"Total Heat Waste":** e.g., "1.2 kW" (Critical for cooling system design).

Module C: Wheatstone Bridge Balancer

Objective: Design and visualize signal conditioning circuits for strain gauges or load cells.

C.1 Math & Logic

The bridge output voltage (V_g) is the difference between the two voltage dividers.

$$V_g = V_{source} \cdot \left(\frac{R_x}{R_3 + R_x} - \frac{R_2}{R_1 + R_2} \right)$$

Note: Linearity Error is defined as the deviation of V_g from a perfect straight line as R_x changes.

C.2 User Inputs (Controls)

1. **Source Voltage:** Input [3.3V or 5V].
2. **Known Resistors (R_1, R_3):** Input fields.
3. **Balancing Resistor (R_2): Fine-tune Knob/Slider** (Interactive element).
4. **Sensor Range (R_x):** Min Ω and Max Ω.

C.3 Real-Time Visuals (Outputs)

- **The "Needle" Gauge:** A visual representation of V_g. As the user moves the R_2 slider, the needle swings. The goal is to center it at 0V (Nulling the bridge).
- **Linearity Plot:** Graph of V_{out} vs. Change in Resistance (ΔR).
  - Feature: Highlight the "usable linear range" in green.

Module D: Filter Designer (Signal Processing)

Objective: Design passive Low-Pass Filters (RC) to clean noisy sensor data (e.g., wheel speed sensors).

D.1 Math & Logic

- **Cutoff Frequency (f_c):**
$$f_c = \frac{1}{2\pi RC}$$
- **Magnitude (Gain):**
$$|H(f)| = \frac{1}{\sqrt{1 + (f/f_c)^2}}$$
- **Phase Shift:**
$$\phi = -\arctan(2\pi f RC)$$

D.2 User Inputs (Controls)

1. **Target Frequency (f_c):** Slider [1 Hz – 20 kHz].
2. **Component Constraints:** Checkbox "Snap to Standard E24 Values" (Forces R/C to real-world resistor values).
3. **Noise Frequency:** Slider (Simulates a noise source, e.g., 50Hz mains or 10kHz inverter switching).

D.3 Real-Time Visuals (Outputs)

- **Bode Plot:**
  - Top Graph: Magnitude (dB) vs. Log Frequency.
  - Bottom Graph: Phase (ϕ) vs. Log Frequency.
- **Signal Preview (Time Domain):**
  - Overlay a "Noisy Sine Wave" (Input) and a "Clean Sine Wave" (Output).
  - *Effect:* As the user lowers the f_c slider, they visually see the jagged noise disappear from the output wave.