✅ Final Algorithm (Step-by-Step Logic)

Step 1: Preprocessing

- Input: Temperature data from Internal & Ambient sensors every 5 minutes.
- Tasks:
  - Interpolate missing timestamps.
  - Mark `NaN` or `0` internal values as **Invalid**.
  - Ensure `ΔIA = Internal - Ambient` is computed.

Step 2: Rolling Calculations

- Rolling window = **1 hour** (12 readings).
- Compute:
  - `Rolling_Mean` of `ΔIA`
  - `Rolling_Std` of `ΔIA`
  - Use this to estimate local variance and system stability.

Step 3: Adaptive Thresholds (IEEE-Based)

Use dynamic, self-correcting limits:

```
Upper_Threshold = Ambient + min(Rolling_ΔIA + Rolling_Std, 65)
Lower_Threshold = Ambient + max(Rolling_ΔIA - Rolling_Std, 0)
```

- Cap threshold at **Ambient + 65°C** as per **IEEE 980 standard**.
- Adjust thresholds if ambient or ΔIA trend increases.

Step 4: Derivative Calculation

- 1st Derivative (`ΔT`) = Instantaneous rate of temp change.
- 2nd Derivative (`Δ²T`) = Acceleration (spike detection).
- Smooth (optional): apply rolling mean for stability.

Step 5: Long-Term Trend Watch

- Use **12-hour** and **24-hour** rolling mean of ΔIA:
  - If increasing trend: flag potential system heating.
  - Trigger "Trend Warning" if average ΔIA slope > threshold.

Step 6: Classification Logic

Every new reading is classified into one of these:

Condition	Classification
`NaN or 0`	**Invalid**
`ΔIA > 65`	**Critical Alarm**
`Temp > Upper_Threshold` and `Δ²T > average std`	**Spike Alarm**
`Temp > Upper_Threshold`	**Alarm**

`Temp > 90% of Upper_Threshold`	**Warning**
`ΔIA trend ↑ for 12+ hours or weekly mean ↑`	**Trend Warning**
`Temp < Lower_Threshold`	**Sensor Drift / Cooling Surge**
Otherwise	**Normal**

Step 7: False Positive Filter

- If a spike ($\Delta^2 T$ > threshold) is followed by 2 stable readings → **don't raise alarm**, just **log incident**.

Step 8: Logging

Log each event with timestamp and classification label:

plaintext
CopyEdit
```
2025-06-19 08:45 – Warning: Temperature approaching upper threshold.
```

🔁 Loop

- After each classification and log: proceed to **next reading** in real-time stream.