Analytics & Alert Logic

1. Introduction

This document describes the implementation of **Analytics & Alert Logic** in a Proof of Concept (POC) developed using an ESP32 edge device, a temperature sensor, Firebase Realtime Database, and an instrument cluster. The focus of this POC is on **edge-level analytics**, where raw sensor data is analyzed locally to generate meaningful events and alerts before being transmitted to the cloud.

The solution demonstrates how real-time analytics can improve system efficiency, reduce data overload, and enable reliable alert generation in automotive and IoT systems.

---

2. Objective of the POC (Analytics Focused)

The primary objective of this POC is to implement **real-time analytics at the edge** in order to:

- Analyze temperature sensor data locally on the ESP32
- Detect abnormal operating conditions using threshold-based analytics
- Convert raw sensor readings into event-based analytical outputs
- Transmit only meaningful and relevant analytics results to the cloud

By performing analytics at the edge, the system minimizes unnecessary data transmission and enables faster and smarter alert generation.

---

3. Problem Statement

Temperature sensors generate continuous streams of raw data that are:

- High in volume
- Mostly non-critical under normal conditions
- Inefficient to transmit entirely to the cloud

Sending all raw data to the cloud increases bandwidth usage, storage cost, and processing latency. The challenge was to design and implement an analytics mechanism that can **distinguish between normal and abnormal behavior in real time** and trigger alerts only when required.

---

4. Initial Approaches Attempted

In the initial approach:

- Continuous temperature values were considered for direct cloud transmission
- No analytics or filtering was applied at the edge

Limitations of the initial approach:

- Lack of data intelligence
- High bandwidth consumption
- No event classification
- Increased number of false alerts due to temporary temperature spikes

These limitations highlighted the need for edge-based analytics.

---

5. Finalized Approach (Implemented Analytics)

The finalized solution implements **Edge Analytics with Event Detection**:

- Analytics logic executes on the ESP32
- Temperature data is analyzed locally in real time
- Only analytical outcomes (events, counts, and alert flags) are sent to Firebase
- The cloud is used primarily for monitoring and alert propagation

This approach ensures efficient data handling and reliable alert generation.

---

6. Analytics-Driven Alert Logic

In this POC, alert generation is **analytics-driven** and not based on raw sensor data.

Service Alert Trigger Conditions:

- Temperature exceeds 28 °C
- Threshold violation occurs more than three distinct times

Analytics Outputs Used for Service Alerts:

- High-temperature event count
- Alert status flag
- Final service warning message

Only after analytics validation **are the service alerts triggered**.

---

7. Analytics Implementation

7.1 Edge Analytics (ESP32)

- Temperature is read every 5 seconds
- Threshold comparison is applied at 28 °C
- Rising-edge detection is used to identify valid threshold crossings
- Only valid crossing events are counted

Generated analytics data:

- Current temperature value
- High-temperature event count

- Alert status flag

7.2 Cluster-Side Analytics

- Reads analytical data from Firebase
- Validates repeated threshold violations
- Triggers the final service alert only after analytics confirmation

---

8. System Architecture (Analytics Flow)

The analytics flow of the system is as follows:

Sensor → ESP32 (Analytics Processing) → Firebase (Filtered Analytics Data) → Instrument Cluster (Decision Analytics) → Driver Alert

Analytics is performed at multiple levels, with **edge analytics as the primary processing layer**.

---

9. Firebase-Based Integration Rationale

Firebase Realtime Database was selected because it:

- Supports real-time synchronization of analytics data
- Instantly updates analytical values such as flags and counters
- Handles structured analytics data efficiently
- Is lightweight and suitable for IoT and POC implementations

Firebase acts as a real-time bridge between the edge device and the cluster.

---

10. Challenges Faced (Analytics Related)

- Preventing false event counts caused by short temperature spikes
- Designing accurate rising-edge detection logic
- Maintaining analytics accuracy over long runtimes
- Balancing analytics complexity with ESP32 resource constraints

---

11. Key Learnings

- Edge analytics significantly reduces cloud dependency
- Event-based analytics is more effective than raw data streaming
- Threshold- and count-based analytics improves alert reliability
- Multi-stage analytics validation avoids false alerts
- Analytics logic must be lightweight, deterministic, and robust

---

12. Current Status of the POC

- Edge analytics fully implemented on ESP32
- Threshold-based event detection functioning correctly
- Analytics data successfully stored in Firebase
- Instrument cluster consuming analytics data
- Alert logic validated using analytics outputs

The POC is fully functional and demonstrable.

---

13. Future Enhancements (Analytics Perspective)

- Adaptive or dynamic analytics thresholds
- Trend-based analytics over time
- Predictive analytics using historical data
- Multi-parameter analytics (temperature, voltage, current)
- Analytics dashboards and visualization
- Machine learning-based anomaly detection

---

14. Conclusion

This POC successfully demonstrates **real-time analytics implemented at the edge**, where raw sensor data is transformed into meaningful events, counts, and alerts before reaching the cloud. By executing analytics on the ESP32 and transmitting only filtered analytical results to Firebase, the system achieves reduced data transfer, faster alert response, improved accuracy, and enhanced safety and reliability.

The implementation clearly illustrates how **Analytics & Alert Logic** can be efficiently realized in modern automotive and IoT systems using edge computing principles.