Telemetry & Real-Time Analytics – POC

Scope: Battery Temperature, Speed Monitoring with Throttling, and Vehicle Tilt.

1. Objective of the Expanded POC

This project demonstrates an intelligent edge-computing system for Electric Vehicles (EVs). It captures critical environmental and performance metrics, applies local decision-making (Edge Filtering), and executes **active safety measures** (Throttling) before transmitting actionable data to the Firebase cloud.

2. Expanded Problem Statement

Basic telemetry systems only "report" data. A sophisticated automotive system must "react."

- **The Latency Gap:** Waiting for the cloud to tell a vehicle to slow down is dangerous.
- **The Noise Issue:** Constant speed and tilt data clog bandwidth.
- **The Safety Risk:** Over-speeding and high-angle tilts (rollover risks) require immediate local intervention.

3. System Components & Sensors

Feature	Sensor	Edge Action
Battery Health	MPU6050 (6-Axis)	Filter data if below 28°C.
Speed Monitoring	Potentiometer(0-5k ohm)	Throttling if > Limit.
Vehicle Tilt	MPU6050 (6-Axis)	Emergency Alert if angle > 45°.

4. Logic & Edge Computing Design

4.1 Speed Monitoring & Active Throttling

The ESP32 monitors the vehicle's velocity. To ensure safety, we implement a **Throttling Layer**:

- **Threshold:** 60 km/h.

- **Mechanism:** If `current_speed` exceeds the threshold, the ESP32 ignores the user's full throttle input and caps the Motor Controller's PWM signal to 50% capacity.
- **Telemetry:** Only "Over-speed" events and "Throttling Active" states are pushed to Firebase to save data.

4.2 Vehicle Tilt (Rollover Detection)

Using the MPU6050 Accelerometer, the system calculates the Roll and Pitch:

- **Logic:** If $Tiltangle > 45^{\circ}$, the system flags a high rollover risk.
- **Cloud Action:** Immediate push to `safety/alerts` node in Firebase for fleet management notification.

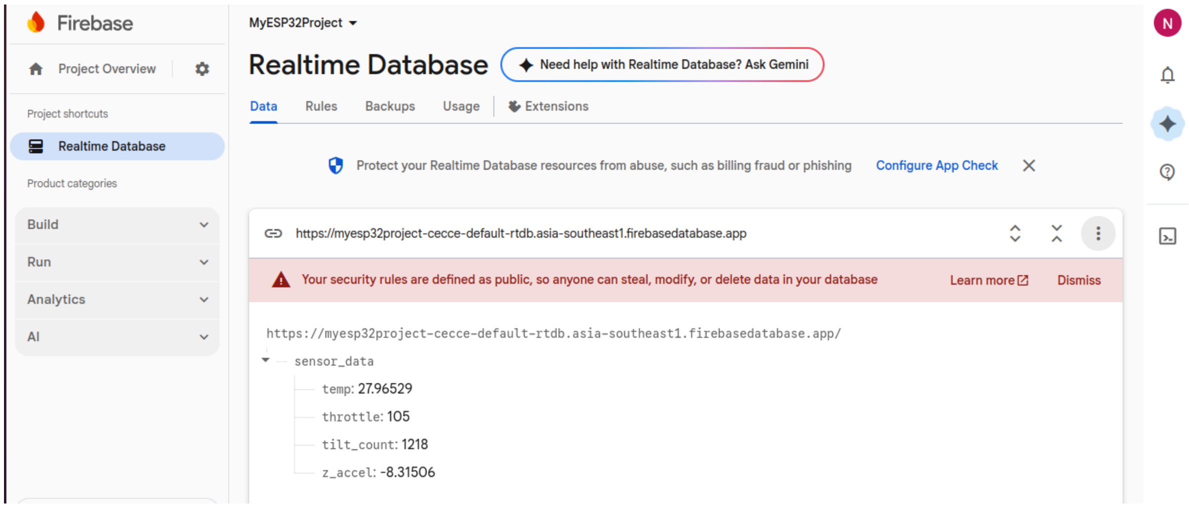
---

5. Technical Architecture

1. **Sensing Layer:** Continuous polling of Temp, Speed, and Tilt.
2. **Edge Logic Layer (ESP32):** * Compares inputs against preset safety constants.
    - Applies PWM reduction (Throttling) if speed is excessive.
3. **Communication Layer:** * Secure Wi-Fi transmission using Firebase Arduino Client.
    - Event-driven updates (only sends data when thresholds are breached).
4. **Visualization Layer:** OKT507-C (QT Environment) displays real-time gauges and a "Throttling Active" warning light.

---

6. Firebase Data Structure



Protect your Realtime Database resources from abuse, such as billing fraud or phishing

https://myesp32project-cecce-default-rtdb.asia-southeast1.firebasedatabase.app

Your security rules are defined as public, so anyone can steal, modify, or delete data in your database

```
https://myesp32project-cecce-default-rtdb.asia-southeast1.firebasedatabase.app/
▾ sensor_data
    temp: 27.96529
    throttle: 105
    tilt_count: 1218
    z_accel: -8.31506
```

Esp 32 code

```
#include <WiFi.h>
#include <FirebaseESP32.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

// Include helper headers for the Mobizt library
#include <addons/TokenHelper.h>
#include <addons/RTDBHelper.h>

// --- WiFi & Firebase Credentials ---
#define WIFI_SSID "naveen"
#define WIFI_PASSWORD "12345678"
#define DATABASE_URL
"myesp32project-cecce-default-rtdb.asia-southeast1.firebasedatabase.app"
#define DATABASE_SECRET "lPB66kuttHkMmRbY59px5JFT9wow3CWvQxhiRL5T"

// --- Pins ---
#define THROTTLE_PIN 34
#define THROTTLE_MIN 0
#define THROTTLE_MAX 4095

// Objects
Adafruit_MPU6050 mpu;
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

float lastZ = 0;
int tiltCount = 0;
bool mpuReady = false;
unsigned long sendDataPrevMillis = 0;

void setup() {
  Serial.begin(115200);
  delay(1000);
  Serial.println("\n--- Initializing System ---");

  WiFi.mode(WIFI_STA);
  WiFi.setSleep(false);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to WiFi");
  int timeout_counter = 0;
  while (WiFi.status() != WL_CONNECTED && timeout_counter < 40) {
        delay(500);
        Serial.print(".");
        timeout_counter++;
  }

  if (WiFi.status() == WL_CONNECTED) {
        Serial.println("\n[SUCCESS] WiFi Connected!");
  }

  // --- Firebase Config Fix ---
  config.database_url = DATABASE_URL;
  config.signer.tokens.legacy_token = DATABASE_SECRET;

  // Assign callback for token generation
  config.token_status_callback = tokenStatusCallback;

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

  // --- MPU6050 Init ---
  Wire.begin(21, 22);
  if (mpu.begin()) {
        mpuReady = true;
        sensors_event_t a, g, temp;
        mpu.getEvent(&a, &g, &temp);
        lastZ = a.acceleration.z;
        Serial.println("MPU6050 Ready!");
  }

  analogReadResolution(12);
}

void loop() {
  // 1. Read Sensors
  int rawValue = analogRead(THROTTLE_PIN);
  int throttlePercent = map(rawValue, THROTTLE_MIN, THROTTLE_MAX, 0, 100);

  float currentZ = 0;
  float tempVal = 0;

  if (mpuReady) {
        sensors_event_t a, g, temp;
        mpu.getEvent(&a, &g, &temp);
        currentZ = a.acceleration.z;
        tempVal = temp.temperature;

        if (abs(currentZ - lastZ) > 1.2) {
        tiltCount++;
        Serial.println("\n>>> MOTION DETECTED <<<");
        }
        lastZ = currentZ;
  }

  // 2. USB Serial Output (Local)
  Serial.print("DATA,");
  Serial.print(throttlePercent); Serial.print(",");
  Serial.print(currentZ); Serial.print(",");
  Serial.print(tiltCount); Serial.print(",");
  Serial.println(tempVal);

  // 3. Firebase Update (Fixed Syntax)
  // Check Firebase.ready() instead of WiFi status
  if (Firebase.ready() && (millis() - sendDataPrevMillis > 500)) {
        sendDataPrevMillis = millis();

        FirebaseJson json;
        json.set("throttle", throttlePercent);
        json.set("tilt_count", tiltCount);
        json.set("z_accel", currentZ);
        json.set("temp", tempVal);

        // FIXED LINE: Removed ".RTDB" and used pointer for json
        if (Firebase.updateNode(fbdo, "/sensor_data", json)) {
        Serial.println(">> Firebase Updated Successfully");
        } else {
        Serial.print(">> Firebase Error: ");
        Serial.println(fbdo.errorReason());
        }
  }

  delay(100);
}
```

Hardware Image :