Software Requirements Specification (SRS)

Networked Onboard Vehicle Analytics (NOVA)

Submitted to:
National Telecommunication Institute (NTI)
Internship Program

Submission Date: December 28, 2025

Proposed by:

Names:
Anas Mohamed Gouda
Abdelrahman Okasha
Mohamed Esam
Wael Kahlwi
Aya Ramadan
Asma Maher

Mentor:
SEI TECH – Industry Mentor

Document Information
Version: 1.0
Date: December 28, 2025
Status: Draft

Table of Contents

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document describes the functional and non-functional requirements for the AIoT system running on an ESP32 microcontroller for a vehicle monitoring and safety system. The document is intended for submission with the final project as part of an internship provided by the National Telecommunication Institute (NTI). It is targeted at developers, testers, mentors, and stakeholders to understand the expected behavior and constraints of the software. The team will be responsible for training the machine learning models used in this project for engine health monitoring and driver state detection.

1.2 Scope

The software manages multiple sensors and actuators in a vehicle to monitor location, cabin conditions, engine health, driver presence and status, vehicle lighting, and driver authentication. It processes data using sensor fusion and machine learning inference on the edge, communicates alerts via SMS in critical situations, and publishes telemetry data to a remote MQTT broker. The system runs on FreeRTOS for task management and concurrency, integrating AI capabilities for intelligent decision-making in an IoT context.

1.3 References

- ESP32 Technical Reference Manual
- FreeRTOS API Reference
- MQTT Protocol Specification (v3.1.1 or v5)
- Relevant sensor datasheets (e.g., DHT22, MPU-6050 or similar IMU, etc.)
- TinyML / Edge Impulse documentation for ML model deployment
- NTI Internship Guidelines for Final Project Submission

1.4 Overview

The remainder of this document describes the product functions, constraints, interfaces, detailed requirements, and system architecture.

2. Overall Description

2.1 Product Perspective

The system is an AIoT-based vehicle monitoring solution aimed at enhancing vehicle safety, security, and predictive maintenance. It integrates multiple sensors with an ESP32 microcontroller as the central processing unit, leveraging AI for on-device inference and IoT for remote connectivity.

2.2 Product Functions

The main functions are:
- Real-time vehicle location tracking using GPS and IMU sensor fusion
- Cabin environment monitoring (temperature and humidity)
- Engine health monitoring using vibration and temperature data with on-device ML inference
- Critical event alerting via SMS
- Driver door proximity detection
- Front headlight status monitoring
- Driver authentication using RFID
- Driver state monitoring using camera and ML inference
- Telemetry publishing to a remote MQTT broker

2.3 User Classes and Characteristics

- Driver: Authenticated via RFID, monitored for state (drowsiness/fatigue)
- Fleet Manager / Remote Monitor: Receives telemetry and alerts via MQTT broker and SMS
- Maintenance Personnel: Uses engine health inference results
- NTI Internship Team: Develops, trains ML models, and submits the project

2.4 Operating Environment

- Microcontroller: ESP32 (dual-core, Wi-Fi capable)
- Operating System: FreeRTOS
- Power: Vehicle 12V supply with appropriate regulation
- Temperature: Automotive grade (-40°C to +85°C)

2.5 Design and Implementation Constraints

- Programming language: C/C++ (ESP-IDF framework)
- Real-time operating system: FreeRTOS for task scheduling and synchronization
- Communication protocols: MQTT over Wi-Fi, I2C, UART
- Machine learning: Lightweight models suitable for edge inference TensorFlow Lite Micro trained by the team
- Limited RAM and flash memory on ESP32

2.6 Assumptions and Dependencies

- Stable Wi-Fi connectivity for MQTT communication
- Cellular network coverage for SMS alerts
- ML models for engine fault detection and driver state monitoring will be trained by the team and deployed on the device
- Hardware connections as specified are reliable
- Compliance with NTI internship requirements for project documentation and submission

2.7 System Architecture Diagram

The system architecture is depicted below in a high-level block diagram. It illustrates the key components, including sensors, the ESP32 microcontroller, communication interfaces, ML inference modules, and external systems.

Diagram Description:
- Top Layer (External Systems): Represents remote connectivity via MQTT for telemetry and UART for SMS alerts.
- Middle Layer (ESP32 with FreeRTOS): Core processing unit handling ML inference (trained by the team), task scheduling, sensor fusion, and actuator control.
- Bottom Layer (Hardware Interfaces): Sensors and modules connected via protocols like UART, I2C, ADC, and GPIO.
This diagram can be refined using tools like Draw.io or Visio for the final submission to NTI.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 Hardware Interfaces

Interface	Protocol	Description
GPS module	UART	Receive NMEA sentences for positioning
IMU	I2C	Read accelerometer and gyroscope data
DHT22	One-wire (digital pin)	Read temperature and humidity
Engine vibration sensor + temperature	ADC (analog)	Read raw vibration and temperature values
SIM module (e.g., SIM800L)	UART	Send AT commands for SMS
Ultrasonic sensor (driver door)	Digital GPIO	Trigger and echo pins
Light sensor (headlights)	ADC or digital	Detect light intensity or current draw
Camera module	Parallel or SPI	Capture images for driver monitoring

3.1.2 Communication Interfaces

- MQTT over TCP/IP (Wi-Fi) to publish sensor data and status to a broker
- Topics to be defined (e.g., `/vehicle/<id>/location`, `/vehicle/<id>/engine`, etc.)

3.2 Functional Requirements

3.2.1 Location Tracking

- The system shall fuse GPS and IMU data to provide continuous and accurate vehicle position even during temporary GPS loss.
- Position data shall be published periodically to the MQTT broker.

3.2.2 Cabin Monitoring

- The system shall read temperature and humidity from the DHT22 sensor at configurable intervals.
- Data shall be published to the MQTT broker.

3.2.3 Engine Health Monitoring

- The system shall acquire vibration and engine temperature data via ADC.
- Data shall be fed to a team-trained ML model for binary classification (faulty / not faulty).
- Inference result and confidence shall be published to MQTT.
- In case of detected fault, an SMS alert shall be sent.

3.2.4 Critical Alerting

- The system shall send SMS alerts for:
  - Engine fault detection
  - Driver drowsiness/fatigue detection
  - Prolonged presence near driver door (potential theft attempt)
  - Headlight failure
  - Other configurable critical conditions

3.2.5 Driver Door Proximity Detection

- The system shall measure distance using the ultrasonic sensor.
- If an object is detected within a threshold distance for longer than a configurable time (e.g., 30 seconds), trigger an alert (SMS and MQTT).

3.2.6 Headlight Monitoring

- The system shall monitor light intensity or electrical parameters to detect headlight failure.
- On detection of damage/failure, publish status and send SMS alert.

3.2.7 Driver Authentication

- On RFID tag detection, the system shall read driver data and validate against stored whitelist.
- Authentication status shall be logged and published via MQTT.

3.2.8 Driver State Monitoring

- The system shall periodically capture images from the camera.
- Images shall be processed by a team-trained on-device ML model to detect drowsiness or distraction.
- Detection result shall trigger SMS alert and MQTT publication.

3.2.9 Data Telemetry

- All sensor readings, inference results, and events shall be published to appropriate MQTT topics with JSON payload.

3.2.10 Task Management

- The system shall use FreeRTOS tasks for:
  - Sensor reading and processing
  - ML inference
  - MQTT communication
  - Alert handling
  - Synchronization using queues, semaphores, and mutexes

3.3 Non-Functional Requirements

3.3.1 Performance

- Sensor sampling rates shall meet hardware specifications (e.g., GPS 1 Hz, IMU 100 Hz if needed).
- MQTT publish interval configurable (default 10–30 seconds).

3.3.2 Reliability

- System shall handle sensor failures gracefully (timeout and retry).
- Critical alerts shall have higher priority in task scheduling.

3.3.3 Security

- MQTT connection shall use TLS if configured.

3.3.4 Maintainability

- Code shall be modular with clear task separation.
- Configuration parameters (thresholds, intervals) shall be adjustable via compiled constants.