**Pawel Jaglarz**

**20097569**

**BSc (Hons) in Software Systems Development**

**Final Year Project Report**

[**20097569@mail.wit.ie**](mailto:20097569@mail.wit.ie)

**Table of Contents**

Introduction

Functional Summary

Assumptions

Functional Requirements

* System Context
* Detailed Requirements List
* Data Requirements

Technical Investigations

Interface Specifications

Non-Functional Requirements

Quality Assurance Provisions

* Software Test Procedures
* Software Validation Procedures

Methodology Overview

Project Plan

Risks and Contingencies

System Architecture

Prototype Implementation

**Introduction**

This project aims to develop a real-time OBD-II (On-Board Diagnostics II) data monitoring systems for most vehicles on the market. I would like to interface a live data monitoring screen into the vehicle so the user receives data on the go this data would later be used for example for preventative maintenance or find cause of an issue that arises with the vehicle. OBD – II uses in car sensors to receive data that may not naturally show up on the cars on board display unit. My project would allow users to more closely monitor those sensors such as engine RPM, coolant temperature, boost pressure etc… I decided to create this project because of my passion of cars and wanting to take care of my car better through using this device to save money time and prevent stress not knowing what is exactly wrong with my car.

**Functional Summary**

This project's primary objective is to enhance user vehicle management by offering a real-time OBD-II (On-Board Diagnostics II) data monitoring system. Customers may get the information they need about the health and performance of their vehicle through the system's live data monitoring interface. The main characteristics of the system are:

**Real-Time Monitoring:** This function gives customers instant access to data from several in-car sensors, including as engine RPM, coolant temperature, and boost pressure, giving them a quick picture of how well the car is performing.

**Preventative Maintenance**: By allowing users to monitor critical data over time, the technology makes it possible to identify possible problems early. By scheduling maintenance before issues worsen, this proactive strategy helps save time and money.

**Diagnostic Support:** Users can use the data gathered to determine the underlying cause of a car problem, streamlining the troubleshooting process and decreasing the need for mechanics.   
**User-Friendly Interface:** The live data display's intuitive navigation makes it possible for users to swiftly obtain the information they require while driving without being distracted.   
**Improved Vehicle Care:** By developing a deeper comprehension of the systems in their vehicles, owners may take better care of them, which will increase their longevity and performance.

**Assumptions**

**Vehicle Compatibility**: It is assumed that most vehicles on the market are equipped with the OBD-II port and sensors generally used due to regulations. The systems are expected to work as effectively and efficiently as possible with most petrol and diesel vehicles manufactured after the year 1996

**Users Technical Proficiency**: Users are assumed to have basic understanding of their vehicles systems and technology. They should be comfortable the device to their vehicle and navigating the user interface. (Plenty of video tutorials on majority of cars and their OBD-II connection point location)

**Data Accuracy**: It is assumed that the data retrieved from the vehicle’s OBD-II system is accurate and reliable. The system does not account for potential sensor malfunctions or discrepancies in data reporting.

**User Environment**: The system assumes that users will maintain a safe driving environment and not get distracted by the live screen instead focusing on the road.

**Functional Requirements**

**System Context:**

A diagram of a computer system

Description automatically generated

**Components:**

1. **Users**
   * Interacts with the system via Raspberry Pi interface.
2. **Raspberry Pi (Monitoring Screen)**
   * Central processing unit for live diagnostics.
3. **OBD-II Connection**
   * Connects to the vehicle's ECU to collect diagnostic data.
4. **Vehicle ECU**
   * Provides sensor data and diagnostic trouble codes (DTCs).
5. **Mobile Device (Hotspot)**
   * Used as a mobile hotspot to facilitate data uploads.
6. **AWS Cloud Server**
   * Stores historical data and allows for further processing and analysis.
7. **Data from Sensors**
   * Source of real-time diagnostic information.

**Data Flow:**

* **User to Raspberry Pi:**
  + Input/output interactions for accessing diagnostics.
* **Raspberry Pi to OBD-II Connection:**
  + USB/Bluetooth communication to retrieve data from the vehicle.
* **Raspberry Pi to Vehicle ECU:**
  + CAN Network for accessing diagnostic data.
* **Raspberry Pi to Mobile Device:**
  + Data uploads for cloud synchronization.
* **Mobile Device to AWS Cloud Server:**
  + Uses mobile hotspot to upload data to the cloud.
* **Vehicle ECU to Data from Sensors:**
  + Diagnostic data flow from sensors to ECU.

**Requirements List**

**1. User Interface (UI)**

* 1.1 To display live diagnostics, the system must have an intuitive user interface. 1.2 Users will be able to enter vehicle identifying numbers (VINs) into the system. 1.3 Real-time data, such as engine temperature, RPM, and vehicle speed, will be shown by the system. 1.4 Data should be represented graphically by the system (e.g., graphs, gauges). 1.5 Users should be able to view and clear diagnostic trouble codes (DTCs) on the system.

**2. Information Gathering**

* 2.1 The OBD-II port on the car will be connected to the system. 2.2 The ECU's diagnostic trouble codes (DTCs) will be retrieved by the system. 2.3 The system will obtain real-time sensor information from the car, such as:
* 2.3.1 Engine temperature
* 2.3.2 Fuel level
* 2.3.3 Vehicle speed
* 2.3.4 Engine Speed
* 2.3.5 Oil Temperature

3. **Data Processing**

3.1 The system shall process retrieved data to generate diagnostic reports. 3.2 The system shall identify potential issues based on DTCs. 3.3 The system shall provide recommended actions based on diagnostics.

4. **Mobile Connectivity**

4.1 The system shall utilize a mobile device as a hotspot for cloud connectivity. 4.2 The system shall upload diagnostic data to the AWS Cloud Server. 4.3 The system shall allow users to manually trigger data uploads to the cloud. 4.4 The system shall support secure data transmission to the cloud.

5. **Cloud Integration**

5.1 The system shall store historical diagnostic data on the AWS Cloud Server. 5.2 The system shall allow users to retrieve past diagnostics from the cloud. 5.3 The system shall provide users with a summary of past issues and resolutions.

**Data Requirements**

**1. External Data Sources**

* **OBD-II Data**
  + Vehicle diagnostic trouble codes (DTCs)
  + Live sensor data (e.g., speed, temperature, fuel level)
* **Manufacturer Database**
  + Detailed descriptions of DTCs and suggested repairs

**2. Local Data Stores**

* Diagnostic Data Store
  + Stores historical diagnostics and user interactions
* **User Data Store**
  + Stores user profiles, preferences, and vehicle information

**A screenshot of a computer

Description automatically generated**

**Class Descriptions**

1. **User**
   * Contains user details and methods for creating and updating user profiles.
2. **Vehicle**
   * Holds information about the vehicle, including make, model, and year.
3. **DiagnosticData**
   * Stores diagnostic reports, including DTCs and associated sensor data.
4. **SensorData**
   * Represents individual sensor readings (e.g., temperature, speed) with their type and units.
5. **ManufacturerDB**
   * Represents external data from the manufacturer, detailing DTCs and suggested repairs**.**