University of Moratuwa

Department of Computer Science and Engineering

CS 4202 –Research and Development Project

Final Year Project Report

Cloud-Based Driver Monitoring and Vehicle Diagnostic with OBD2 Telematics

External Supervisor

Mr. Afkham Azeez

Internal Supervisor

Dr. H. M. N. Dilum Bandara

Group Members:

Malintha Amarasinghe	100029X
Sasikala Kottegoda	100266U
Asiri Liyana Arachchi	100297N
Shashika Muramudalige	100346P

ABSTRACT

The usage of vehicles all over the worlds has drastically increased during the last decade. Over 60 million passenger cars have been manufactured in the year of 2012. This rapid increase of vehicles has led to many concerns for a range of people and organizations where most of these concerns are common to all parties. For example, all parties (i.e., drivers, insurance companies, fleet vehicle managements, low enforcements authorities, etc.) are concerned about reckless driving and driver anomalies whereas drivers as individuals and people who are willing to purchase and sell cars are concerned also about the condition of the vehicle.

OBD (On Board diagnostics) is a standard which allows accessing the status of sensors of a vehicle via a port referred to as the OBD port. Few of those sensors can be stated as speed, engine rpm, coolant temperature, fuel rate and oxygen. OBD-II, is latest version of OBD and is implemented in almost all high-end vehicles which are manufactured lately. Several adapters are commercially available to read data from the OBD-II port. ELM-327, which is used in the project, is one such adapter where the data which are read from the port are transmitted via Bluetooth upon pairing.

Given the potential benefits of vehicular data analysis and the availability of technologies such as OBD, several vehicle monitoring and intelligent transport systems have been proposed. But, in almost all the systems proposed, there has been either simple or no processing of the data gathered from the Engine Control Unit (ECU) prior to displaying and are restricted to monitoring. Also they, hardly include a backend; therefore, are limited by the computational power of the smart phone. Hence, it is hard to predict any undesired outcome, such as an accident or a failure of sensor since they require real time and long-term analysis of data regarding the driving habits and the vehicle condition.

The proposed system is similar to the above described systems from the aspect that it uses OBD-II protocol and an Android app as the device of mediation. In addition, it comes with a set of complex analyses to perform reckless driving detection, driving anomaly detection, vehicle sensor failure prediction, high-fuel consumption and high-coolant temperature alert generation and trip detail summarization. The analyses are performed both in real time as well throughout a long period of time. While some of these analyses are performed within the app, more complex and resource consuming ones are performed in the back end. The results of these analyses are made visible through two interfaces. The drivers themselves are able to get the

results through the Android app in the form of notifications. Alerts generated both in the back end as well as the app due to undesirable situations are sent to the drivers. Also results of long term analyses are displayed through a web interface. The web interface enables stake holders such as organizations insurance companies and fleet vehicle management systems and authorities to view results of a desired set of people.

ACKNOLEDGEMENT

A successful project is never something that could be achieved alone. Kampana would have been a dream for a group of four final year students if not for the help and guidance received from many other parties.

The inception is the most important part of and project. Therefore, we would first like to thank Mr. Afkham Azeez, Director of Architecture, WSO2 Inc., for providing us the idea for this project and for guiding us all along. Next we would like to express our sincere gratitude to Dr. H. M. N. Dilum Bandara for giving us a big hand from the first day of the project right until the end. There are not enough words to how thankful we are, for giving us an immense support by bearing our mistakes, correcting us when we were wrong and guiding us along the correct path. We would also like to convey our hearties gratitude to Dr. Sanjiva Weerawarana, Founder, Chairman & CEO; WSO2, Inc., for his valuable ideas and for taking time to advise us despite his busy schedule. Also our sincere thanks goes to Dr. Chathura De Silva for lending us his OBD2 adaptor and for providing instructions as to how to continue with the work.

We would further be grateful to all the academic staff members, especially Dr. Malaka Walpola for providing us the opportunity to conduct the project and for encouraging towards this.

It is also necessary to thank all the people who helped us in collecting data, since without them, the project could not proceed.

Finally, we would like to thank all our friends, family members and others who have helped us even by a single word, since sometimes, even those little words encouraged us and motivated us a lot.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Background

A motor vehicle has become an integral part of a modern family. The usage of vehicles in Sri Lanka has drastically increased during the last decade. Approximately 30,000 new motor cars are being registered each year and the number keeps increasing [1]. This rapid increase of vehicles has led to many concerns not only for the drivers but also to the authorities responsible for law enforcement, insurance, and transportation infrastructure planning and maintenance. Thus, if we could provide a way to collect vehicle data and analyze them, it could provide many benefits such as the following to all stakeholders:

- For drivers
    - Self-evaluation of ones driving habits, e.g., how green the driver is?
    - Reduced insurance premium for drivers with good driving habits
    - Ability to monitor health status of the vehicle such as engine faults and emission status
    - Identify unusual driving behaviors
- For cooperates
    - Insurance companies could reward good drivers by reducing their premium and introduce pay-as-you-drive in Sri Lanka
    - Monitoring and management of fleet vehicles by tracking them and notifying unusual behaviors of drivers
- Authorities
    - Law enforcement such as police could better control the traffic and detect unusual driving behaviors
    - Organizations responsible for transportation infrastructure planning and maintenance can get long-term statistics related to vehicle flow, maintenance requirements, and planning for new and upgraded infrastructure

Given the potential benefits of vehicular data analysis, several vehicle monitoring and intelligent transport systems have been proposed [2, 3, 4]. The vehicle diagnosis program proposed by Kim et al. [2] provides a diagnosis of different kinds of vehicle malfunctions within the navigation

system. The information collected through the OBD-II port [5] are displayed in a human readable way. To see this information the driver has to select the 'vehicle information' menu of the navigation pane. Much of this information is displayed on the dashboard by default. Besides, a driver cannot be staring into the navigation pane while driving since it will distract the driver. Quite a lot of research work has been in the area of vehicle monitoring through a server [3, 4, 6, 7]. The main focus of most of the researches is on tracking the vehicles [4, 6, 7] whereas fault detection has also been a concern many times [3, 7]. But there has not been a single system which considers all the aspects that would be of concern to people who are dealing with vehicles. Also, in almost all the systems proposed, there has been either simple or no processing of the data gathered from the Engine Control Unit (ECU), but plainly displaying. Hence, it is hard to predict any undesired outcome, such as accidents through dangerous driving and failure of components through long-term analysis of sensors depending on the present situation and long-term behavior of the driver and the vehicle.

1.2. Problem Statement

The rapid increase of vehicle usage has led to many concerns not only for the drivers but also to the authorities responsible for law enforcement, insurance, and transportation infrastructure planning and maintenance. Though many researches have been carried out to address some of the problems of concern, not many have actually considered the major aspect, which is to collect data related to the driver and the vehicle for a very long time and to predict the future behaviors such as potential for accidents and failure of components and to reason certain behaviors such as an accident or a fuel inefficiency.

1.3. Proposed Solution

To address these limitations, we propose a cloud-based system, which is capable of collecting, storing and analyzing vehicular data for a long period of time. Vehicular Data will be collected using an OBD2 to Bluetooth interface and will be sent to backend cloud servers using a smartphone via 3G/4G connection. The backend of the proposed vehicular data analysis system is based on Complex Event Processing (CEP) and Business Activity Monitoring (BAM). Through this solution the stakeholders will be able to get discounts on insurance premiums through better

driving habits and manage and monitor their vehicles and drivers thus ultimately saving money and time.

Among the important individual vehicular data that needs to be collected, are the driving speed, engine RPM, fuel consumption, emission status, and coolant temperature. There is a standard interface in vehicles manufactured after 1996 [8] for this purpose, which is called OBD-II (On Board Diagnostics) [9], that sends data from the Engine Control Unit (ECU).

An OBD-II to Bluetooth adaptor is plugged into the OBD2 port so that the data coming from ECU can be sent to a mobile app via Bluetooth, from which the gathered data will be sent to the server through a 3G/4G network. The mobile app will also gather data from its inbuilt GPS and accelerometer sensors. The GPS data will be useful for tracking vehicles and traffic prediction. For detecting rapid brakes, accelerometer could be more accurate than using the derivative of speed parameter of the OBD2 port. Afterwards, data will be processed by a Complex Event Processor (CEP) and a Business Activity Monitor (BAM).

1.4 Outline

Section 2 contains a detailed literature survey on this project. It discusses about the fundamentals, related work, existing products and communication methods related to OBD2. The survey also includes a discussion about complex event processing and business activity monitoring. Further, it contains a survey on the main use cases which are driver behavior and vehicle condition analysis.

Section 3 focuses more on the implementation details of the main use cases. It describes how data collection and fabrication was done and how they were simulated. This section also contains the main architecture of the system highlighting the main components which are the OBD2 adapter, android app, CEP, BAM and the web portal. Further it discusses how they have been used to detect reckless driving, driver anomalies and vehicle sensor failures.

Section 4 consists of main outcomes of the project including screenshots of the android app and the web portal. Section 5 comprises of the conclusion and future work of the project.

2. LITERATURE SURVEY

Due to the advancements in the automotive industry and the concern about driver safety and traffic, several researches, studies and projects have been carried out in this area. A number of related tools and technologies have been identified as required for developing the proposed solution are explained in the following sections. The first is the OBD-II adapter which is discussed under Section 2.1 that is required to read data from the ECU. In Section 2.2, a brief description of similar work related to OBD-II that has been carried out is included. Next, in Section 2.3 data communication methods are discussed since it is the primary way of getting data into a smartphone and sending them to the backend. In Section 2.4 an introduction about GPS is included which is the primary way of obtaining the location of the vehicle. The backend processors, CEP and BAM are then explained in Section 2.5 and 2.6, respectively. Section 2.7 includes several researches that have been done in order to identify facts that can represent driving patterns. In Section 2.8, two common impending vehicle sensor failures are discussed which are impending oxygen sensor failure and impending MAF sensor failure. Finally, in Section 2.9 several existing methods of vehicular traffic estimation are discussed.

2.1. On-Board Diagnostics (OBD)

Comparing to the other technologies that are used in our day-to-day life, automotive engines have not changed a lot. The same basic principles that drove the first car engines are still used today. However, modern automotive engines have evolved to meet today's power, efficiency and emission standards. To achieve that, sensors are being used in each sub-system of the vehicle, which continuously monitor the states and are used as a feedback to control the inputs for the engine to gain the maximum in means of power, efficiency and emission standards. All the sensors are connected to the Powertrain Control Module (PCM); an electronic control unit which is considered as the "brain" of a modern vehicle.

There is a special circuit integrated to the PCM, which is designed to monitor emission control systems and key engine components. It is called On-Board Diagnostics (OBD) [8, 9] which refers to a vehicle's self-diagnostic and reporting capability. The diagnostic data can be read through a

port which is located in the passenger compartment. In modern vehicles, the port is called "OBD2 port".

If the OBD system detects a problem, the system turns on a warning lamp (MIL) on the vehicle's dashboard while storing related important information about the detected malfunction in the memory. The stored data can be retrieved via a standardized series of Diagnostic Trouble Codes (DTCs) [10] through the OBD2 port and will be very effective in accurately finding and fixing the problem.

Another key reason for the technology to become prominent is that it enables modern vehicles to integrate with various other systems. It allows an external device to access the vehicle's Electronic Controller Unit (ECU) via a standardized digital communications port to receive real-time vehicle data such as vehicle speed and engine RPM (i.e., number of revolutions of the crankshaft per minute) which are called OBD2 PIDs (On-Board Diagnostics Parameter IDs) [11]. Before OBD2 was introduced, there were several predecessors, which are ALDL (Assembly Line Diagnostic Link) and OBD1. OBD2 is an improvement over OBD1 in both capability and standardization. In 1996, the OBD2 specification was made mandatory for all cars manufactured in the United States to be sold in the United States [12]. After that it has been become popular among most vehicle manufacturers and almost all new vehicles supports OBD2 [13].

2.1.1. OBD2 scanners

There are various types of OBD2 scanners available for automotive technicians.

Figure 2.1. A handheld OBD2 scanner (Actron PocketScan) [14].

Figure 2.2. An OBD2 data logger (TEXA OBD log) [15].

Handheld scanners (see Figure 2.1) are the most common because of their ease of use. The terminal of the device can be connected to the OBD2 port and then both real-time data (Parameter IDs) and stored data (Diagnostic Trouble Codes) can be read through the device's screen. Data loggers (see Figure 2.2) are designed to be semi-permanently connected to the vehicle. These smaller and often screenless devices stay with the vehicle silently logging all of the information that the vehicle's OBD2 port provides. Those devices are used to monitor vehicles for a long period of time.

Data can also be read using a PC or a mobile phone through an OBD2 adaptor (see Figure 2.3). It is the cheapest among the included ways, if one already has a PC or a mobile phone. There are various types of OBD2 adaptors and software tools available. Those will be discussed in Sections 2.1.6 and 2.1.7 in detail.

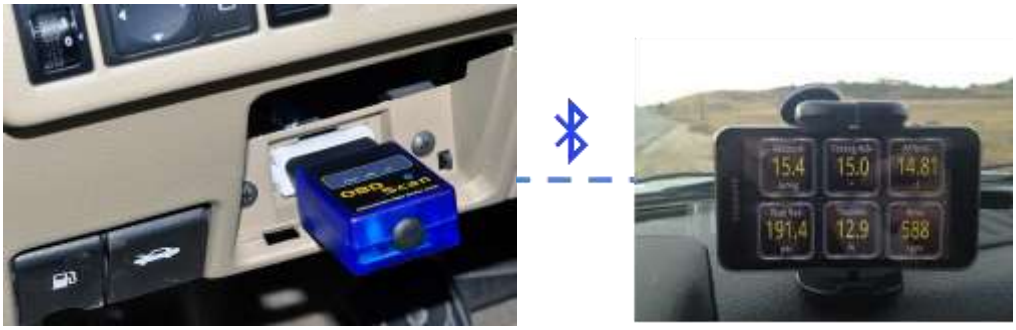

Figure 2.3. Reading data using an OBD2 Bluetooth adaptor and a mobile phone [16].

Torque [16] is a popular mobile app for Android devices which can be connected to an OBD2 Bluetooth adaptor (see Figure 2.3) via Bluetooth. The adaptor should be plugged into the OBD2 port and paired to the mobile phone via Bluetooth. After a successful connection, many of the supported real time engine parameters by the PCM can be seen via the app's GUI.

2.1.2. OBD2 PIDs

OBD-II PIDs (On-Board Diagnostics Parameter IDs) are codes used to request data from a vehicle. There is a standard set of PIDs which is defined by SAE standard J/1979. Other than that, manufacturers define their own PIDs. Figure 2.4 illustrates the bit combination of a PID.

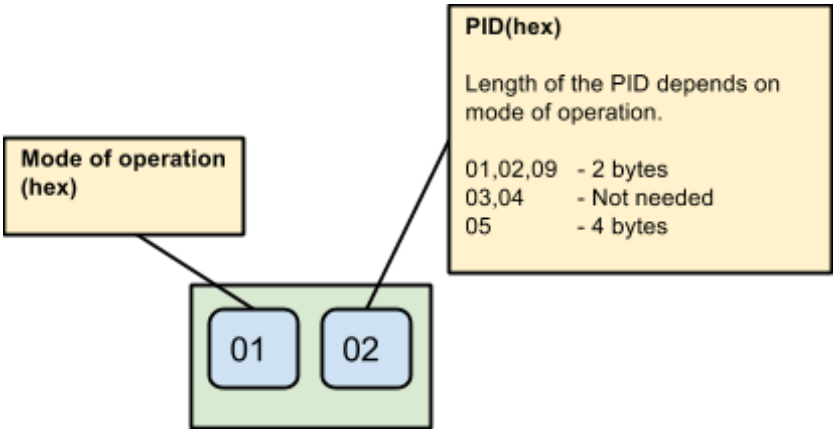

Figure 2.4. Bit combination of a PID [5].

According to Figure 2.4, mode of operation is followed by the PID [5]. PIDs operate in several modes of operations. The latest OBD-II standard, SAE J1979, defines ten such modes of operations [11] which are given in Table 2.1.

Table 2.1. Modes of operations of OBD2 PIDs [11].

Mode (hex)	**Description**
01	Show current data
02	Show freeze frame data
03	Show stored Diagnostic Trouble Codes
04	Clear Diagnostic Trouble Codes and stored values
05	Test results, oxygen sensor monitoring (non CAN only)
06	Test results, other component/system monitoring (Test results, oxygen sensor monitoring for CAN only)
07	Show pending Diagnostic Trouble Codes (detected during current or last driving cycle)

08	Control operation of On-Board component/system
09	Request vehicle information
0A	Permanent Diagnostic Trouble Codes (DTCs) (Cleared DTCs)

To receive real-time vehicle data, Mode 01 should be used. Table 2.2 includes a set of PIDs which can be used in Mode 01.

Table 2.2. A list of standard OBD2 PIDs [11].

PID (hex)	**Description**	**Output bytes**	**Equation**	**Range & Unit**
00	PIDs supported (01 - 20)	4	-	-
04	Engine Load	1	A*100/255	0-100 %
05	Engine Coolant Temperature	1	A - 40	-40 - 215 °C
0C	Engine RPM	2	((A*256)+B)/4	0 - 16,383.75 rpm
0D	Vehicle Speed	1	A	0 - 255 km/h
52	Ethanol fuel percentage	1	A*100/255	0 - 100 %
5C	Engine oil temperature	1	A - 40	-40 - 210 °C
5E	Engine Fuel Rate	2	((A*256)+B)*0.05	0 - 3212.75 L/h

'A' denotes the least significant byte of the output. 'B' denotes the next least significant byte.

2.1.3. OBD2 diagnostic trouble codes

OBD2 Diagnostic Trouble Codes (DTCs) are stored by the On-Board Diagnostic system in response to a problem found in the vehicle. These codes identify a particular problem area and are intended to provide a guide to locate where a fault might be occurring and rapidly identify the cause of the malfunctions within the vehicle. DTCs consist of a five-digit alphanumeric code. They were earlier interpreted through the blinking pattern of the "Check Engine" (MIL) light. Now the system supports gaining DTCs through OBD2 ports using modes of operations. Mode 03 and 04 can be used to show and clear stored DTCs respectively (see Table 2.1) [11].

Following diagram illustrates the bit combination of a DTC [10].

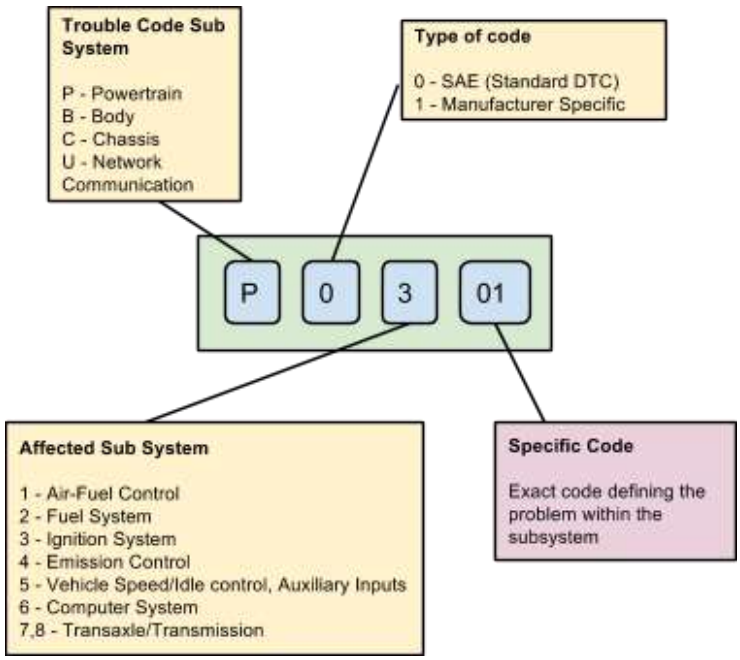

Figure 2.5. Bit combination of a DTC [5].

The list of all the standard DTCs is well defined in SAE J2012 [10]. Some of the common DTCs which occur in vehicles can be listed as:

Table 2.3. Commonly available DTCs [10].

DTC	**Meaning**
P0101	MAF Sensor Performance Issue

P0138, P0139	O2 Sensor Circuit High Voltage/Slow Response (Bank 1 Sensor 2)
P0171, P0174, P0175	Rich or Lean Conditions of Air/Fuel ratio
P0301-P0308	Cylinder 1-8 Misfire Detected
P0420	Catalyst System Efficiency Below Threshold (Bank 1)

2.1.4. Pin diagram

The OBD-II specification provides for a standardized hardware interface—the female 16-pin (2x8) J1962 connector. Unlike the OBD-I connector, which may have been placed anywhere the vehicle, the OBD-II connector is located on the driver's side of the passenger compartment and required to be within two feet (0.61 m) of the steering wheel (unless an exemption is applied for by the manufacturer, in which case it is still somewhere within reach of the driver).

Figure 2.6 shows the pin-out of the connector as defined in SAE J1962 [17]. Pin numbers and their roles are listed in Table 2.4.

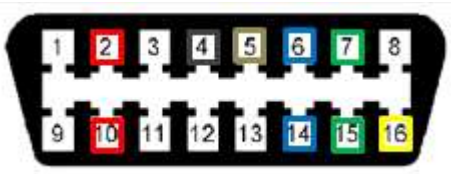

Figure 2.6. Pin diagram of OBD2 port [17].

Table 2.4. Pin numbers and meanings of OBD2 port [17].

Pin No	**Description**		**Pin No**	**Description**
1	Vendor option		9	Vendor option
2	J1850 Bus+		10	J1850 Bus-
3	Vendor option		11	Vendor option

4	Chassis Ground		12	Vendor option
5	Signal Ground		13	Vendor option
6	CAN High J-2284		14	CAN Low J-2284
7	ISO 9141-2 K-LINE Tx/Rx		15	ISO 9141-2 L-LINE Tx/Rx
8	Vendor option		16	+12v Battery power

2.1.5. Protocol types

There are five protocols in use with the modern OBD-II interface which are:

- ISO15765 (CAN-BUS)
- ISO14230 (KWP2000)
- ISO9141-2
- SAE J1850 VPW
- SAE J1850 PWM

Often it is possible to confirm the protocol in use based on which pins are present on the J1962 connector. Many vehicles implement CAN protocol after 2006 [18]. By model year 2008, all vehicles sold in the US must use the CAN bus [19].

Table 2.5. Standard OBD2 protocols [19].

Name	**Data rate**	**Data pins**	**Companies who use it**
ISO 15765 CAN	250/500 KBaud	6, 14	US Vehicles
ISO 14230 KWP2000	1.2 - 10.4 kBaud	7, 15	European 2003+

ISO 9141-2	10.4 kBaud	7, 15	European, Chrysler(2000-2004)
SAE J1850 VPW	10/41.6 KB/s	2	General Motors
SAE J1850 PWM	41.6KB/s	2, 10	Ford

2.1.6. Adaptor types

One of the key reasons that OBD2 has recently become an enabling technology for in-vehicle applications is due to the appearance of different types of OBD-II adaptors. These connectors enable an easier way of connecting between external devices and the vehicle's Electronic Control Unit (ECU). Following are some of the commercially available adaptors.

2.1.6.1. Automatic link

The adaptor costs about $100 and comes along with first party apps for Android and iOS based phone and tablets.

2.1.6.2. Mojio

Mojio [20] costs about $150. It is a cellular and GPS device that connects to the OBD2 port. It has the capability to directly transmit ECU data along with GPS data to the server. So there is no need of intermediate device such as a mobile phone to transmit data. The users are provided with a proprietary app which can also communicate with the device and read data such as DTCs.

2.1.6.3. Craven OBDII connector

Unlike Automatic, the Craven adapter [24] utilizes the third-party apps such as Torque (Android) or DashCommand (iOS) and less expensive. It is available in both Bluetooth and WiFi models. But this does not provide other built in modules such as GPS modules or Accelerometers.

2.1.6.4. ELM327 based adaptors

This is the least expensive option available which is in the price range $7-20 depending on the manufacturer and the terminal type.

Figure 2.7. ELM327 Bluetooth, USB and WiFi adaptors [22].

The device is based on ELM327 microcontroller, which has a low power CMOS design and RS232 serial interface. Based on the terminal type, it includes an RS232 to Bluetooth, USB or WiFi converter module inside the adaptor.

The microcontroller has the ability to detect all the 5 standard OBD2 protocols which is one of the most important features of the adaptor. Because of that, the App developer does not need to worry about the underlying complexity of the OBD2 protocol of the vehicle.

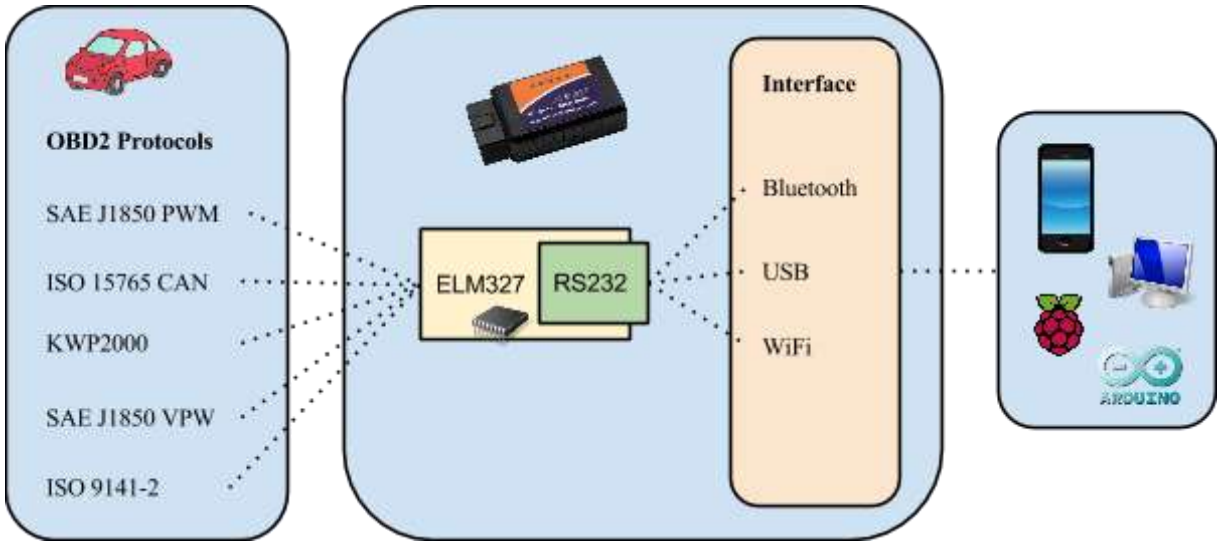

Figure 2.8. Connectivity of the ELM327 adaptor with the vehicle and applications [23].

Since it need to be connected to a mobile phone, only applicable types are Bluetooth and WiFi adaptors. Among them Bluetooth adaptor was chosen since it is less expensive and typically has a lower power consumption than the WiFi adaptor.

ELM327 device basically supports two types of commands. They are AT commands and OBD2 commands. AT commands are used configure the ELM327 device and will never be sent to ECU. Any other command which does not start with AT will be sent to the ECU without any

modification. Those OBD commands to the vehicle are only allowed to contain the ASCII codes for hexadecimal digits (0 to 9 and A to F).

Table 2.6. Example list of AT commands available for ELM327 [24].

AT Command(s)	**Description**
ATZ	Reset All
AT@1,AT@2, ATI	Display device description, ID and version, .g., "ELM327 v1.3a OBDGPSLogger"
ATh	Set default protocol to h, where h is the protocol number (see Table 2.7)
ATAh	Set default protocol to h. If it was not successful search through all protocols and automatically identify the relevant protocol.
ATDP	Describe the current protocol
ATTh	Try protocol h
ATTAh	Try protocol h with auto search

Table 2.7. Available protocol numbers in ELM327 [24].

Protocol	Description
0	Automatic
1	SAE J1850 PWM (41.6 kbaud)
2	SAE J1850 VPW (10.4 kbaud)
3	ISO 9141-2 (5 baud init)
4	ISO 14230-4 KWP (5 baud init)
5	ISO 14230-4 KWP (fast init)
6	ISO 15765-4 CAN (11 bit ID, 500 kbaud)
7	ISO 15765-4 CAN (29 bit ID, 500 kbaud)
8	ISO 15765-4 CAN (11 bit ID, 250 kbaud)
9	ISO 15765-4 CAN (29 bit ID, 250 kbaud)
A	SAE J1939 CAN (29 bit ID, 250* kbaud)
B	User1 CAN (11* bit ID, 125* kbaud)
C	User2 CAN (11* bit ID, 50* kbaud)

*user adjustable

Other than that, ELM327 supports protocol-specific commands such as setting protocol specific baud rate and setting timer multiplier.

2.1.7. OBD2 readable apps

The OBD2 has been becoming popular among drivers since it gives a certain level of transparency for the vehicle. Hence, there are several apps popular among users, which are available for various platforms.

For Android, there are several third-party apps like Torque [16] and ELM327 Terminal [25] (see Figures 2.9 and 2.10). The Terminal app is very primitive and is limited only to interactively test various AT and OBD2 commands via ELM327 adaptor and it does not do any further processing. Torque app comes along with an attractive GUI which is capable of showing OBD2 PID values in real time using its GUI components like graphs and digital/dial meters (see Figure 2.9). It is also capable of viewing DTCs and clear them. Similar apps are available for iOS and Windows platforms too.

Figure 2.9. Torque app [16].

Figure 2.10. ELM 327 Terminal app [25].

There are also PC-based software such as ProScan [26] and OBD GPS Logger [27]. Both are more focused on improving fuel efficiency. Moreover, ProScan tool supports real-time plotting of data, data logging, clearing DTCs, performing fuel system and emission tests.

Main features that are lacking in the above apps are:

- Mobile apps are easier to connect and drive with but lack of computational power and capacity to do comprehensive tests on sensors.
- PC apps do have that capability, but it is not always viable for a driver to take a PC (or a laptop) with the vehicle.
- Do not have the capability to identify impending sensor failures.

2.1.8. Simulators

It is not always a viable option to test OBD2 readable apps in a real environment with a vehicle. As a solution, existing apps which can simulate a vehicle with an ELM327 device plugged in, can be used.

OBDSim [28] is one such simulator. It supports multiple platforms including Windows and Linux, and is able to simulate multiple terminal types which are serial USB and Bluetooth connections. The tool can be configured to replay a previous OBD2 log file dumped by OBD GPS Logger which is gained from running a real vehicle or user can specify values for OBD2 PIDs using the GUI tool (see Figure 2.12). The tool supports a number of AT commands which are available to configure an ELM327 device [29].

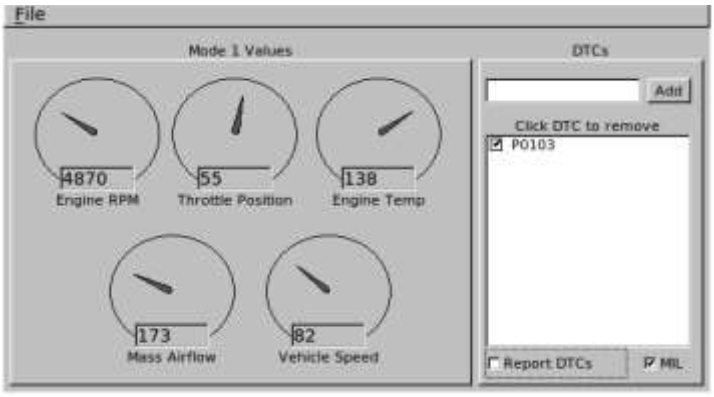

Figure 2.11. OBDSim GUI plugin [28].

2.2 Related Work using OBD-II

Kim et al. [2] proposed a vehicle diagnosis program within the navigation system using the OBD-II technology. Vehicle data that is collected through an OBD-II adapter. The data are then wirelessly transmitted to a Bluetooth dongle from which services are provided to the driver by displaying them in the vehicle information menu of the navigation system.

Several researches have been carried out for monitoring vehicle data collected by an OBD-II adapter, through a server [3, 4, 6, 7]. Y. Yang et al. [3] proposed a system, consisting an ELM327, and Android smartphone and a remote monitoring center for Hybrid vehicles. The ELM327 communicates with the ECU to gather data from which it is sent to the smartphone via Bluetooth. The smartphone acts as the client to complete data acquisition and transmission to the server in which data are monitored. This mainly focuses on vehicle tracking and displaying other data on a server. W. D. Huang [4] proposed a similar solution for general vehicles.

Also, some insurance companies have implemented a policy to provide a discount on the insurance premium for good drivers [24]. The customers get a device to be plugged into the OBD-II port, from which all driver related data is transmitted. By monitoring these data for a fixed period of time, the insurance companies can decide whether the usual driving pattern of a driver is risky or not. Depending on the goodness of their driving, customers get a discount for the insurance premium.

2.3. Communication Methods

2.3.1 Bluetooth

Bluetooth is a wireless communications system which is intended to replace cables connecting many types of different components [30]. Ubiquitousness, low power and low cost are three significant features of Bluetooth. Devices that are connected through Bluetooth (referred to as pairing) are allowed to communicate wirelessly through short-range, ad hoc networks known as piconets.

The most important factor of any communication method when deciding the usability is its data rate. Bluetooth, has come across a number of versions in order to fulfill this requirement by increasing the data rate in major proportions. It has started from 1 Mbps (Approximately 0.7Mbps in practice) and evolved constantly to transmit in 3Mbps in the second version. Many devices still

run on this version since it is fast enough for most of the applications. It has been improved much as to give 24 Mbps in its third version and a mode selection (out of the previous three) in the latest version [31].

Bluetooth technology operates in the unlicensed Industrial, Scientific and Medical (ISM) band at 2.4 to 2.485 GHz, using a spread spectrum, frequency hopping, full-duplex signal at a nominal rate of 1600 hops/sec [30]. The 2.4 GHz ISM band is available and unlicensed in most countries [30]. Though interferences are of major concern in other wireless communication techniques, Bluetooth is said to have a higher degree of interference immunity compared to them. This is enabled by the adaptive frequency hopping approach.

The range a device can communicate via Bluetooth depends on the vendor of the device although a minimum range is required by the core specification. Devices are of three types depending on their class of radio:

1. Class 3 radios – range of up to 1 meter or 3 feet
2. Class 2 radios – range of 10 meters or 33 feet – most commonly found in mobile devices
3. Class 1 radios – range of 100 meters or 300 feet – used primarily in industrial use cases

Specifications of devices of the type of class two radios are of importance in this project since, the communication is mediated via a mobile phone.

Bluetooth technology is designed to have a very low power consumption. Since smartphone that are coming each day to the market are capable of doing large processing, the power consumption dedicated to Bluetooth should be considerably low. Mobile phones use radio class 2 and 2.5 mW of power for the Bluetooth module.

2.3.2. 3G/4G

3G is an ITU specification for the third generation of wide-area mobile communications technology. 3G provides increased bandwidth, up to 384 Kbps when a device is stationary or moving at pedestrian speed, 128 Kbps in a car, and 2 Mbps in fixed applications. 3G works over wireless air interfaces such as GSM, TDMA, and CDMA [32]. 4G is the fourth generation of mobile communication standard. It enables data and streaming multimedia at higher speeds and offers at least 100 Mbit/s with high mobility and up to 1 Gbit/s with low mobility (nomadic) [33].

2.4. GPS

GPS is satellite navigation or sat navigation system is a system of satellites that provide autonomous Geo-spatial positioning with global coverage. GPS allows small electronic receivers to determine their location (longitude, latitude, and altitude) very accurately within a few meters using time signals transmitted along a line-of sight by radio transmitters from satellites. GPS satellites frequently broadcast, precisely timed messages, including following two parameters [34].

- Time, the message was transmitted
- Satellite position at time of message transmission

Once a message received, the receiver calculates the transmit time of the message (using time difference) and then using speed of light it calculates the distance to the satellite. The satellites are located medium earth orbit where between 24 and 32 satellites are located. Receiver device need signals from at least three satellites to compute its two dimensional (latitude and longitude) position distance or at least four satellites to compute its three dimensional (latitude, longitude and altitude) position.

GPS is one of the data fields that cannot be collected from the OBD-II port. But it is extremely useful for some use cases such as for vehicle tracking and estimation of vehicular traffic. There are two ways that GPS data could be collected:

1. By using a dedicated GPS module
2. By using a smartphone

After doing some research, it was found out that a GPS module costs around Rs. 3,500.00. This would increase the cost to implement the system. Thus, it was concluded that the solution lies within smartphones since modern smartphones come with the facility to locate the mobile phone using GPS. But it is a known fact that GPS drains the battery of the mobile much faster. The faster you are travelling the greater the energy consumption [35]. Hence, a simple and a cost effective way should be found out in-order to gather GPS data.

2.5. Real-Time Processing

“Today's IT environments generate continuous streams of data for everything from monitoring financial markets and network performance, to business process execution and tracking RFID

tagged assets" [36]. Hence, fast analyzing of large data volumes has become inevitable. Complex Event Processing (CEP) has emerged as a method for analyzing such data where data are produced by events [37].

2.5.1. Databases or CEP?

A white paper published by Sybase [37] states that the technology of CEP delivers the data analysis tools traditionally provided by relational databases or even spreadsheets, but in a real-time event-driven implementation that is capable of processing incoming data at very high rates and producing results with near-zero latency.

If we consider a scenario of storing a large amount of data in a relational database in a high frequency and getting them analyzed in real time, there are two approaches that could be taken. According to the approach that is taken, databases are of two types as follows.

1. Passive Databases
2. Active databases

Passive databases are the ones that which store, manage and process data in a timely (scheduled) manner [38]. In passive databases, queries which are run periodically will have to be written for analyzing. These queries may read from various tables and produce a new tuple which may be inserted into a resultant table. Hence, the analysis will be done offline but not in response to the incoming events [37].

In spite of having to write the triggers manually, active databases allow analyses to be done in response to incoming events [38]. The complexity added in writing the triggers is a major drawback of this approach. Also the vulnerabilities caused by the added complexity, and the high time complexity caused by inefficient code add up reasons for not considering this approach as a method for real-time analyses.

In contrast, Complex Event Processors (CEPs) are ones that are already built and optimized for analyzing large volumes of data in response to events [39]. This reduces the time consumed for implementing the analyses by preventing us from rewriting the triggers. Being optimized for time and space consumption and being secure are the added benefits of using a CEP.

2.5.2. Complex event processing

According to Luckham who is considered as the first person to have directed the first project to pave the way to a generic CEP language and execution model [41], Event processing is the technology of processing a set of events in-order to derive useful information in the events [40]. CEP is the processing of events when, events come from a large number of sources and are if complex inter-relations while surfing through them for patterns that create opportunities or threats in real time.

According to Leavitt [41], CEP systems collect data from numerous sources about raw events within a company's operations on an ongoing basis (see Figure 2.6). Then use algorithms and rules are used to determine in real time the interconnected trends and patterns that combine them into complex events. They then send the findings to the appropriate business user.

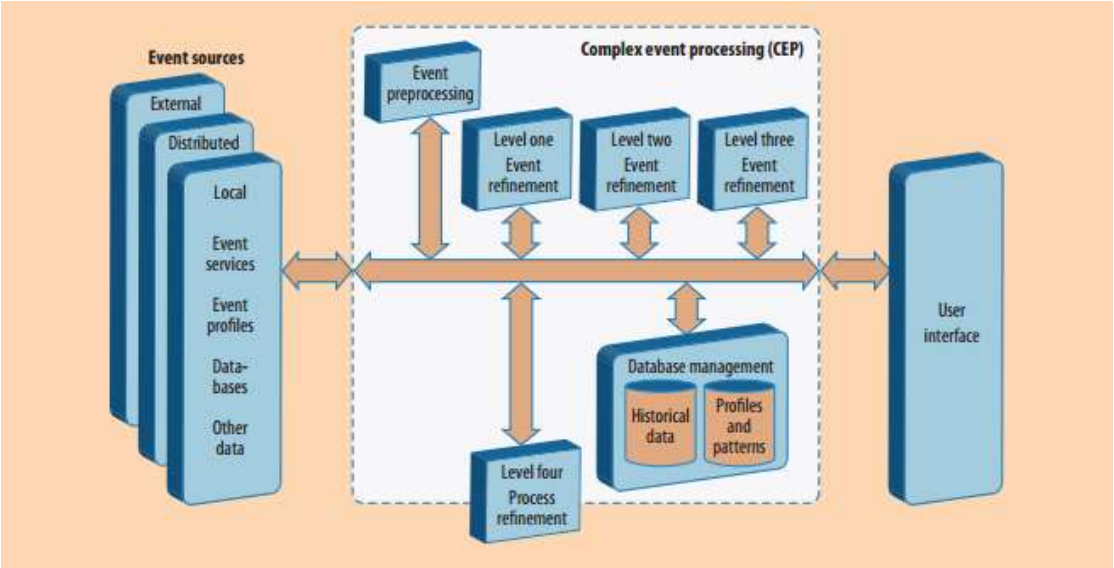

Figure 2.12. The process of CEP [41].

The most important fact about CEP is that it is an event driven model [40]. Rizvi [38] states that an 'Occurrence of Significance' is considered an event. Thus the generation of data from the OBD2 adapter can be seen as a generation of an event. Each event may carry only a little piece of information, but when each event is aggregated to its context, useful patterns, which derive greater knowledge may occur [40]. For example, though each speed reading might not provide any useful information about a driver, a speed pattern derived by the collection of speed data for a long period of time might define the driver's behavior and might even uniquely define a driver.

2.5.3. Complex event processors

Following are some of the complex event processors that are used widely.

2.5.3.1. TIBCO

The TIBCO CEP platform is a high-performance system for rapidly building applications that analyze and act on real-time streaming data following the stream-base architecture [42]. The company claims that the product possesses the following capabilities [42]:

- Rapid time from development to deployment via the industry's first and only graphical event-flow language
- Extreme performance with a low-latency, high-throughput event server and the broadest connectivity to real-time and historical data
- Rapid, on-the-fly processing of complex data streams and fast time to prototype, test, and deploy real-time applications

2.5.3.2. Oracle CEP

Oracle CEP, is a low latency, Java-based middleware framework for event-driven applications. It is a light-weight application server which connects to high volume data feeds and has a CEP engine to match events based on user defined rules [43]. Oracle CEP has the capability of deploying user Java code (POJOs) which contain the business logic.

2.5.3.3. WSO2 CEP

WSO2 CEP is built with the aim of recognizing the events with significant importance within a large set of events and to perform certain actions on them in real time. It is also claimed that the CEP is highly efficient in performance and is massively scalable [44].

2.5.4. Comparison of complex event processors

Since all the CEPs appeared to be performing in the way that is needed in the research, it was needed to compare between them to identify the most suitable CEP.

Suhothayan et al. provides a brief description of a comparison of five CEP engines: TelegraphCQ, SASE, Cayuga, Esper and Siddhi [45] which are used as the core processing engines inside the

commercially available complex event processors. In brief, it explains how the technology of CEP evolved.

TelegraphCQ can be considered the first in the family. It was built as a system to process continuous queries by extending the architecture of PostgreSQL [46], and later, Truviso did some more modifications in order to support historical queries as well. However, none of these methods were able to suffice the expected speed [45]. Later products, i.e., SASE, Cayuga and Esper came up by using variants of Nondeterministic finite automata on which, later, the presently commercially available products were built upon.

Suhothayan et al. [45] states that, in a study carried out by Diao et al., SASE performs much better that TelegraphCQ due to its NFA based architecture and algorithms that are used to yield a higher scalability. But SASE had a major limitation of not being able to handle complex event types where the outputs of some queries should be used as inputs to some others. Cayuga was a system which addressed this problem. It also supported higher scalability and concurrency. Esper was a much more advanced complex event processing engine, thus ended up being the most widely used open source CEP engine and is used as the core of many more CEP solutions [45].

In the study carried out by Suhothayan et al. [45], Esper and Siddhi were compared by providing exactly the same conditions. In the cases of simple filtering and averaging over time windows Siddhi performs 20-30% better (faster) than Esper, and in the case of pattern matching, Siddhi performs 10 times faster than Esper. Since Siddhi was performing extraordinarily, it was concluded to use WSO2 CEP which uses Siddhi as its inner core.

2.5.5. WSO2 CEP in detail

WSO2 CEP is an open source product developed by WSO2 Inc. in order to cater the need to listen to events and to detect patterns in real time without having to store them [44]. It runs WSO2 Siddhi at the core which does all the processing such as filtering, averaging and pattern matching. WSO2 CEP comes with the following features:

- Extremely high performance
- Powerful and Extensible Query Language for processing
- Support for rich event model (Support for metadata and various data types)
- Massive Scalability

- High Availability
- Tight integration with WSO2 Business Activity Monitor
- Many more

There are three main aspects that should be of concern when working with WSO2 CEP [44].

- Event Publishing
- Event Processing
- Output

Figure 2.13 shows the flow of events through the server.

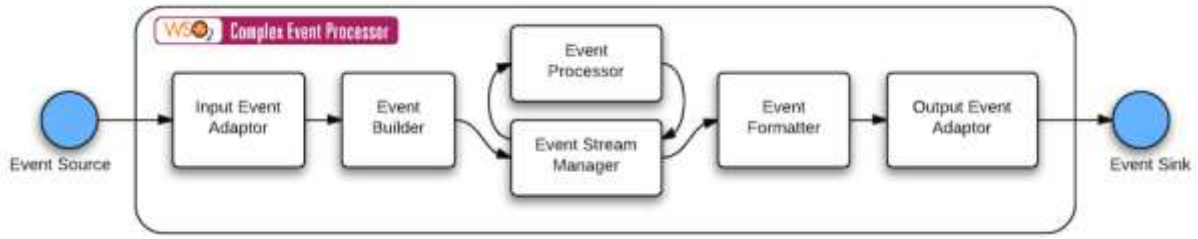

Figure 2.13. Event flow of WSO2 CEP [44].

In order to publish data to the CEP correctly, 'Input Event Adapters' and 'Event Builders' should be created and configured properly. Additionally, Event Streams, that define which attributes and elements need to be captured from the incoming message (via various transport protocols such as Http, jms, Thrift, etc.) should be created.

Mainly, the processing of WSO2 CEP happens through execution plans. An execution plan is a set of queries and related input and output streams. It holds the processing logic.

WSO2 CEP uses 'Output Adapters' to send events to sinks. Implementations, such as to send as a SOAP message and to store in a database that are needed most of the time are by default provided.

2.6. Business Activity Monitoring

Business activity monitoring refers to aggregating, analyzing and presenting information about business activities. This definition is paramount when designing a solution to address a business activity monitoring use case. Aggregation refers to the collection of data, analysis to the

manipulation of data to extract information, and presentation refers to representing this data visually or in other ways such as alerts.

In the use case of Vehicular Data Analysis and Real Time Processing, the information which is the vehicular data generated by the OBD2 adapter should be aggregated, analyzed and presented in order to meet the project requirements. Also with the enormous amount of data received from the adapter it requires a way to handle data quickly and efficiently. For example, to determine the reckless driving behavior of a driver, calculated acceleration of the vehicle is used. The system should be capable of stating whether the driving pattern is reckless or not and how frequently the driver tends to perform reckless driving just by looking at the summarized data over a period of time. This involves executing a number of processes over big data to get an outcome which exactly what BAM is all about. Thus, a tool which can perform BAM functionalities is an ideal approach for the task.

An evaluation of business activity monitoring tools is given in [47]. In the study, authors have considered available BAM tools in the category of both proprietary and open source. Specifically regarding proprietary tools, Oracle and IBM and, regarding open source tools WSO2BAM and Chroniker.

In order to evaluate the products they have defined six main key aspects.

1. Installation: so it is possible to conclude about the difficulties of this process.
2. Integration: so it is possible to conclude how easy it is to access existent and heterogeneous data.
3. Dashboards: so it is possible to conclude the existent variety and difficulty in the creation of dashboards and in what way they can help managers.
4. Alerts: so it is possible to conclude and difficulty in the creation of alerts and in what way they can help preventing some problems.
5. Cost: so it is possible to conclude if a higher cost means more functionalities, flexibility and versatility.
6. Support: so it is possible to conclude if documentation and help are specific of a given type of tool (proprietary or open source).

Table 2.8 shows a summary of the main aspects regarding the evaluation of BAM tools.

Table 2.8. A comparison of WSO2 and Oracle BAMs [47].

Characteristic	**Oracle BAM**	**WSO2 BAM**	**IBM Business Monitor**	**Chroniker**
Level of difficulty of installation	Medium	Medium	Difficult	Medium
Integration with existing data / Configuration data sources	Good and easy	Good and easy	Difficult	Not Available
Variety and difficulty of creation of dashboards	Much variety Easy creation	Much variety Difficult creation	Much variety Difficult creation	Much variety Difficult creation
Variety and difficulty of creation of alerts	Much variety Easy creation	Some variety Difficult creation	Not Available	Not much variety Difficult creation
Cost	High	Low or none (Open Source)	Higher than Oracle BAM	Low or none (Open Source)
Support	Very good	Almost inexistent	Not Available	Low

Since the project depends a lot on how the analyzed data are presented, other than using BAM inbuilt dashboards it is proposed to go for more advanced tool for that particular task. There are a lot of tools which provide functionality to rapidly create visually appealing and engaging web components such as dashboards, microsites, gadgets and also which provide server-side capabilities out-of-the-box. Thus the above mentioned aspect of difficulty level in the creation of dashboards does not affect much on choosing a BAM.

Opposed to WSO2 BAM, using Oracle BAM leads to lower performance of other applications after the installation of oracle environment. Therefore, they strongly advise to use computers with high resources for receiving effective results.

With all this being said, the best choice for an organization depend on several factors, of which we point out the availability for investment in the product, the necessity for training and the need for support. For example, smaller organizations or projects might prefer open source solutions as it includes low-cost, high control which is considered to be the main benefit for some customers. They also give the ability to users to change the source code.

According to above all facts WSO2 BAM can be considered as the most viable selection for the project not only because, it is promising in producing desired results but also being open source and free.

2.6.1. WSO2 BAM

The WSO2 BAM architecture reflects the natural flow of aggregating, analyzing and presenting information about business activities in its very design.

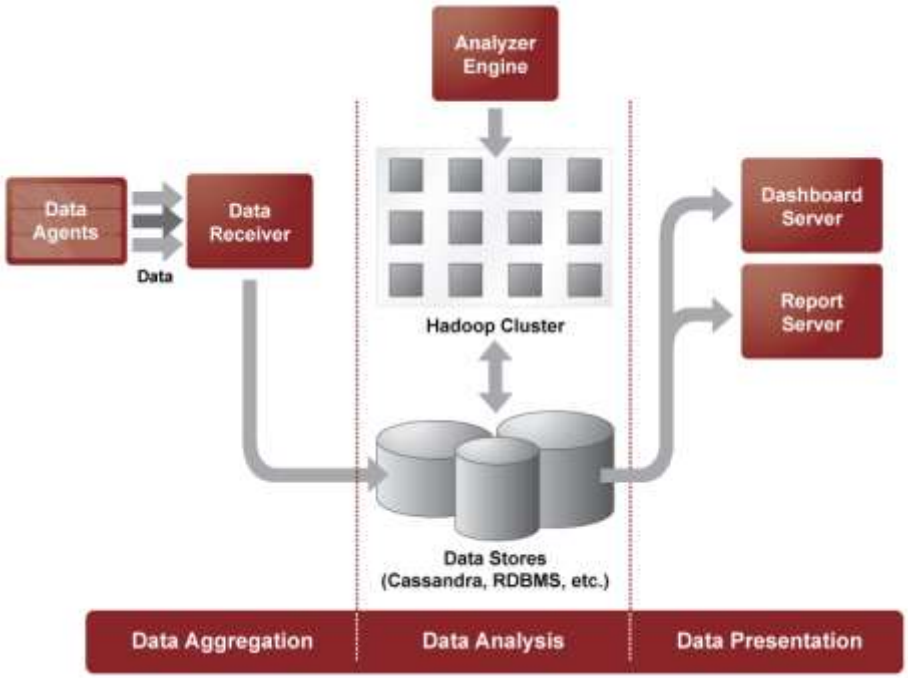

Figure 2.14. WSO2 BAM Architecture [48].

WSO2 BAM architecture can be broken down into four main modules:

1. Data Agents
2. The Data Receiver
3. The Analyzer Engine
4. The Dashboard and Reports

Data that need to be monitored goes through these modules in order. The data flow is as follows;

1. Data will be sent from the data agent to the BAM server,
2. The Data Receiver will process and store the received data in the Cassandra data store.
3. Then, the Analyzer Engine will start to analyze this data according to defined analytic queries. This will usually follow a pattern of retrieving the data from the data store, performing some sort of data operation such as an addition and storing it back in a data store. This data store can be different from the Cassandra data store. The data operations will happen locally if the Analyzer Engine is not pointed to a Hadoop cluster.
4. Finally, the dashboard or reports will query the data store for the analyzed data and show it in the UI.

2.7. Identification of Driver Patterns

Due to the high concern of fuel efficiency and safe driving, identifying driver patterns has been a principal research area. Liaw [49] proposed a solution to the problem of identifying driving patterns by using fuzzy logic as the base technique. There, the main focus has been on classifying the region a driver usually drives in where regions are considered to be of five main categories such as stop-n-go, urban, suburban, rural and highway. The analysis starts off with breaking up a trip into several Driving Pulses (DPs) where a driving pulse represents an active driving period between contiguous stops. Each of these DPs are characterized by its average speed and the running distance. A mature membership function as shown in Figure 2.15 defines driving patterns according to a set of trained fuzzy rules, therefore allowing each input DP to be classified into a region as a degree of association.

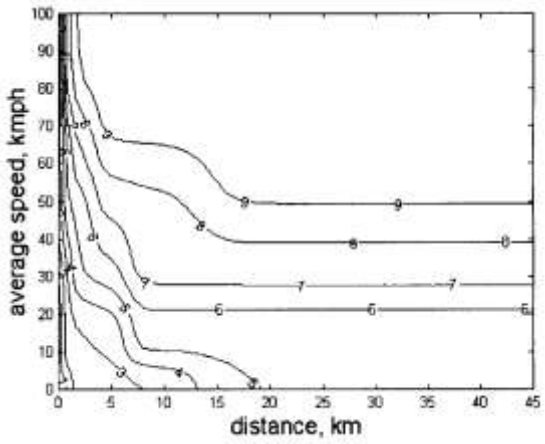

Figure 2.15. A membership function - The contours show the map of boundaries of fuzzy output number in a scale of 0-10, which is used to infer the driving patterns from stop-n-go to highway driving [49].

Finally the results of the sequential DPs created by a trip are aggregated to produce an overall driving pattern.

Research carried out by Wahab et al. [50] suggests that the driving pattern can uniquely define a driver, thus enabling us to use it as a biometric. Biometric authentication is an area consisted of a number of technologies that uses one's physiological and behavioural attributes to uniquely identify a person. Physiological attributes are considered to be stable features such as the fingerprint and iris. Behavioural attributes, though considered as a biometric (hence, unique to each person), can change depending on the psychological status of a person. In their research, two main characteristics, namely the accelerator pedal pressure and the brake pedal pressure have been used as the biometrics. Similar to the previous study, the

researchers started with partitioning the trip data into chunks of data that are within two stops (referred to as a stop&go region) since only an insignificant amount of information is present while a vehicle is not on the move. Figures 2.16 - 2.18 show the associated vehicle speed, accelerator pedal pressure and brake pedal pressure signals of a vehicle for a particular stop&go region.

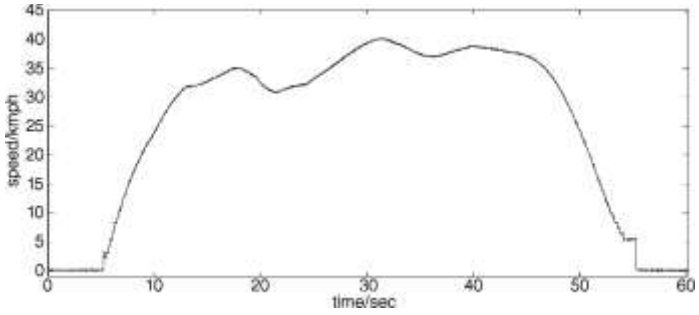

Figure 2.16. Signal trajectory of a stop&go region for the vehicle speed [50].

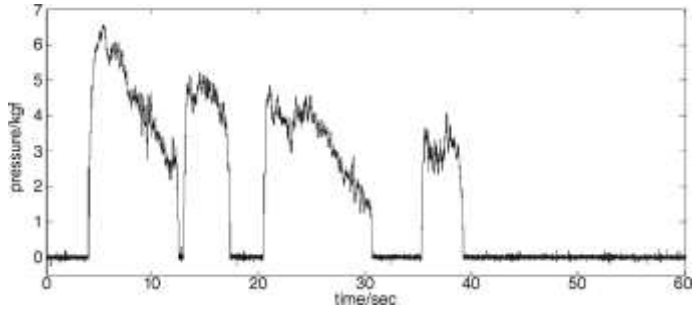

Figure 2.17. Signal trajectory of a stop&go region for the accelerator pedal pressure [50].

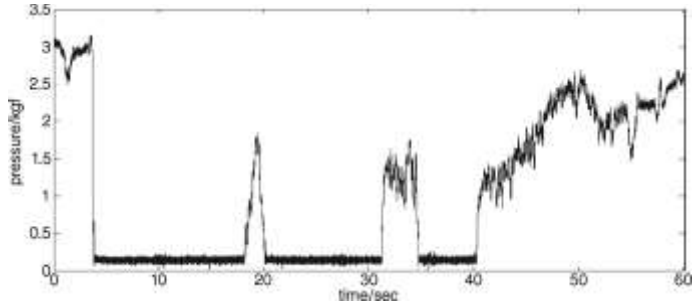

Figure 2.18. Signal trajectory of a stop&go region for the brake pedal pressure [50].

Driver characteristics have a greater impact on the dynamics of the data (acceleration and brake pedal pressures) rather than the single data items. Thus, the features were further broken down as the acceleration pedal pressure, brake pedal pressure, rate of change of the pressure of the acceleration pedal and the rate of change of the pressure on the brake pedal. Then the driver

recognition task was carried out using Gaussian Mixture Model (GMM) and Evolving Fuzzy Neural Network (EFuNN) where EFuNN performed better than GMM.

2.8. Identification of Impending Sensor Failures

Modern computerized engine control systems rely on inputs from a variety of sensors to regulate engine performance, emissions and other important functions.

The Powertrain Control Module (PCM) has the ability to control the emissions while delivering the torque to the vehicle requested by the driver. When a driver pushes the accelerator, the throttle valve opens, which regulates the airflow into the engine. The intake manifold is the main air passage from the throttle valve to the engine cylinders. The amount of fuel needed for the combustion in each engine cylinder is a direct function of the throttle position and the mass of air through the intake manifold. This mass of air is measured by the Mass Air Flow (MAF) sensor. The Manifold Absolute Pressure (MAP) sensor measures the intake manifold pressure, which is also used to measure the amount of air going into the cylinder as a second method. After the combustion happens, oxygen sensors can measure the amount of oxygen left in the exhaust gas, which in turn can be used to calculate the air/fuel ratio. To minimize the emissions, this ratio should be around 14.7 which is called the stoichiometric ratio [51]. Exhaust gases are then passed through the catalytic converters in which most of the remaining hydrocarbons (HC), carbon monoxide (CO) and nitrogen oxides (NOx) are oxidized to nitrogen (N2), carbon dioxide (CO2) and water (H2O).

To keep the consistency of the flow, the sensors must provide accurate information. Otherwise problems in driving, increased fuel consumption and emission failures can result. OBD2 system already has the capability to detect malfunctions of the vehicle sensors and turns the “MIL” light on to indicate to the driver. It is indicated after the emissions are exceeded 1.5 times the applicable standard [51]. The corresponding sub-system(s) and the specific sensor(s) for the failure are available via a standardized series of DTCs.

However, there are several limitations of existing OBD systems:

- The DTCs are set only after the particular sensor is failed.

- OBD2 systems can detect when components are functioning within their operating range, but are limited with the ability to determine whether they are functioning accurately within the range [51].

In other words, the existing OBD systems cannot detect impending sensor failures. An oxygen sensor which begins to fail, starts giving incorrect readings which cannot be identified by the OBD2 system. This would result in an incorrect air/fuel ratio and hence would reduce the efficiency of the engine. The combustion would also produce a significant amount of pollutants which will go through the catalytic converter. Since the amount of pollutants is high, the converter will drastically degrade which is very expensive to replace.

In this project the following main impending sensor failures will be focused:

- Impending oxygen sensor failure
- Impending MAF sensor failure

2.8.1. Impending Oxygen Sensor Failure

The Oxygen sensor is used to monitor the residual oxygen in the exhaust gases whose output is calibrated to measure the air/fuel ratio in the engine cylinders [51]. It is proportional to the amount of oxygen in the exhaust gases. As seen in Figure 2.19 Oxygen sensors are placed before and after the catalytic converter.

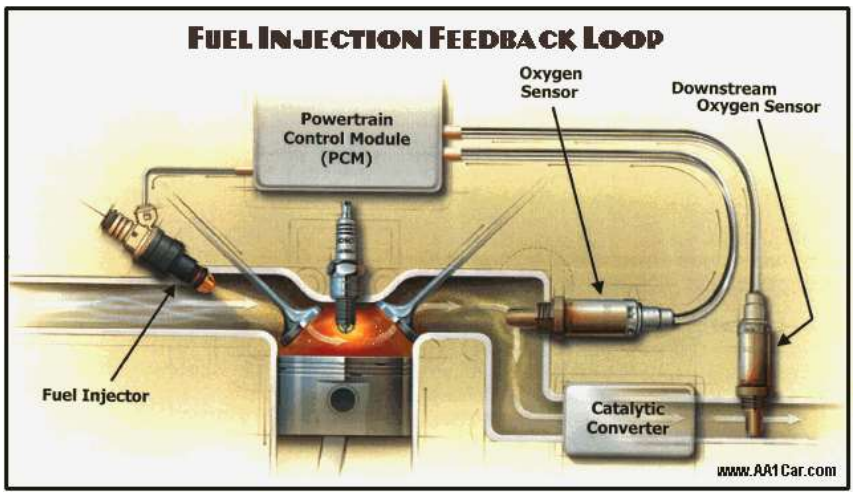

Figure 2.19. Locations of the oxygen sensors [52].

Upstream Pre-Cat sensor is used for regulating the fuel supply. Downstream Post-Cat sensor monitors the efficiency of the catalytic converter.

As illustrated in Figure 2.20, an oxygen sensor will typically generate up to about 0.9 volts when the fuel mixture is "rich" (i.e., the air/fuel ratio is less than the stoichiometric ratio) and there is little unburned oxygen in the exhaust. When the mixture is "lean" (i.e., the air/fuel ratio is greater than the stoichiometric ratio), the sensor output voltage will drop down to about 0.2 volts or less. When the air/fuel mixture is balanced or at the equilibrium point of about 14.7 to one, the sensor will read around 0.45 volts [52].

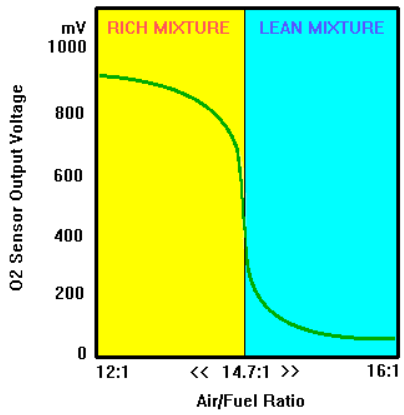

Figure 2.20. Oxygen sensor switching voltage [52].

When the PCM receives a rich signal (high voltage) from the O2 sensor, it leans the fuel mixture to reduce the sensor's feedback voltage. When the O2 sensor reading goes lean (low voltage), the computer reverses again making the fuel mixture go rich. This constant flip-flopping back and forth of the fuel mixture occurs with different speeds depending on the fuel system. It keeps the air/fuel ratio steady at the stoichiometric ratio [52].

Graphs shown in Figure 2.21 indicate oxygen sensor waveforms operated under different conditions.

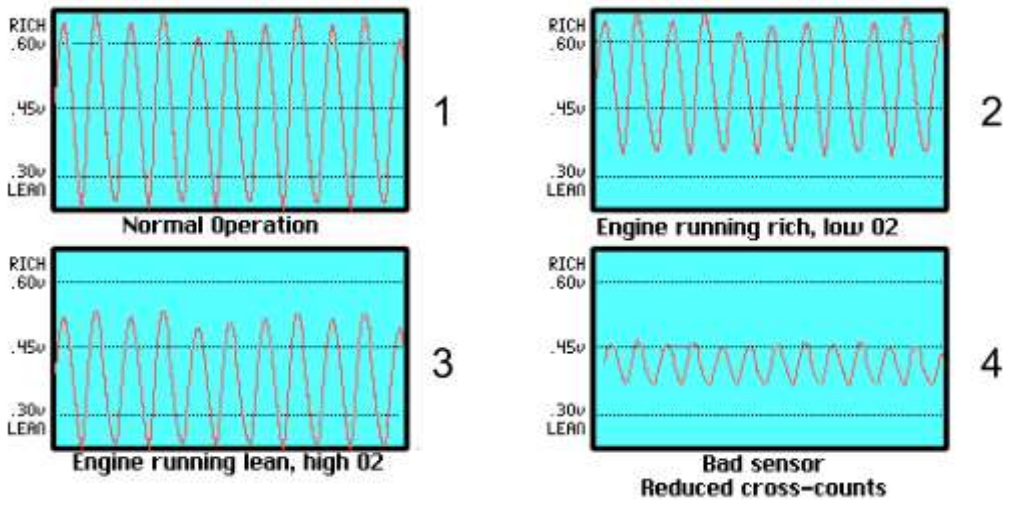

Figure 2.21. Oxygen sensor waveforms [52].

Graph 1 indicates a good O2 sensor which produces an oscillating waveform that makes voltage transitions from near minimum (0.1V) to near maximum (0.9V). This performance of the O2 sensor gradually tends to degrade with age as contaminants of the exhaust gas such as lead and silicon accumulate on the sensor tip which will decrease the output voltage range. Graph 4 indicates the waveform of a degraded oxygen sensor.

The PCM monitors the cross counts (the number of times the oxygen sensor crosses the center line) and the response time as shown in Figure 2.22 [5]. The reduced cross counts, and longer response time beyond a certain threshold would essentially cause the PCM to enable a fault code and illuminate the "MIL" light (e.g., graph 4 of Figure 2.21).

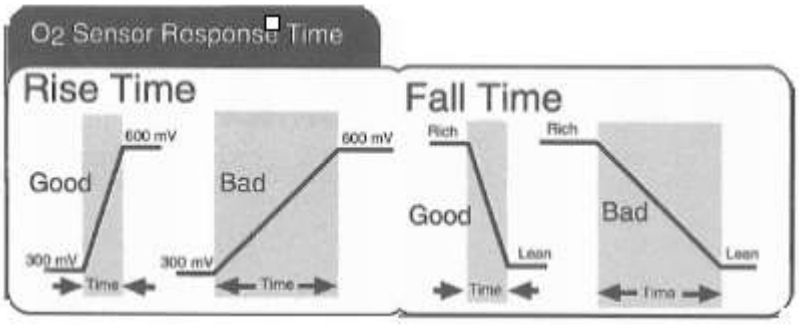

Figure 2.22. Response time comparison of good and bad oxygen sensors [5].

Before the PCM identifies, the sensor may have operated for a long time being slightly biased to rich or lean outputs as indicated in graphs 2 and 3 respectively. It would increase emissions and

fuel consumption of the vehicle unnoticingly for a long period of time. Those rich or lean bias conditions can be monitored by reading the oxygen sensor values via the OBD2 port. PIDs shown in Table 2.9 can be used [11] for this purpose.

Table 2.9. PIDs that can be used for reading oxygen sensor [5].

PID (hex)	Description	Output bytes	Equation	Range & Unit
13	Oxygen sensors present	1	-	-
14 - 17	Bank 1, Sensor 1:4 Oxygen sensor voltages	2	A/200	0-1.275 (Volts)
18 - 1B	Bank 2, Sensor 1:4 Oxygen sensor voltage	2	A/200	0-1.275 (Volts)

2.8.2. Impending MAF sensor failure

Mass Air Flow Meter (MAF) is an air flow sensor, also known as a MAF sensor, and is an integral component of the computer controlled engine system found on most modern cars. Generally located in the plastic housing between the engine and the air filter, the MAF sensor measures the volume and density of air entering the engine [53]. It is an electronic sensor that sends a signal to the PCM, which then uses the measurements to help calculate fuel delivery and spark timing. In most applications, the MAF sensor utilizes an electrically charged wire to determine the amount of air flow entering the air intake. This wire produces an electrical voltage, based on airflow, and sends it to the PCM, which then uses the information to adjust spark timing, fuel delivery and spark advance. A fault in the MAF sensor can cause a rough idle, poor fuel economy and possibly even stalling.

One of the more commonly overlooked maintenance items is the air filter, which can have a direct effect on the MAF sensor. The air filter is engineered to filter out impurities in the air before it enters the engine, thus reducing excess wear and tear on the engine. A dirty air filter will not only allow impurities to enter the engine, but it can also allow dirt to accumulate in the MAF sensor, which can reduce its effectiveness.

Engines require a precisely controlled air and fuel mixture, and spark timing in order to function properly, in which the MAF sensor plays an important part. Keeping a car and all its related components in excellent working condition is necessary in order to maximize fuel economy and maintain power, performance and reliability

As mentioned previously, MAF sensors will malfunction over time, because a layer of dirt and grime builds up on the sensor wire. The grime coating insulates like a sweater, then the ECU gets it as a less air going than the standard. So it will lead a lean condition and illuminate the Service Engine Soon or Check Engine light to indicate to the driver that the car requires service.

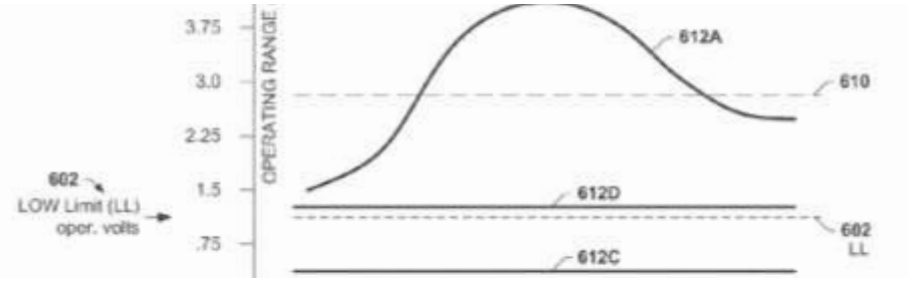

Figure 2.23. Mass Air Flow [53].

Figure 2.23 illustrates Mass Air Flow (MAF) sensor and several possible voltage outputs vs. time. LL and HL are respectively the Low Limit and the High Limit of the output of the sensor that is operated under normal conditions. LL may be established at about 1.1 Volts, and HL at about 4.5 Volts, and a typical or average voltage level at about 2.8 Volts. For example, sensor output plot 612A, illustrates that initially, at time0, the MAF sensor is indicating that the engine is aspirating a lower level of airflow such as at idle or low engine RPMs. Thereafter, the MAF sensor indicates an airflow that peaks around time 4-5 (e.g., indicating a higher engine RPM), and then finally reduces down just below a median or average level and final level at time 9 (e.g., indicating just under a mid-range RPM).

Sensor output plot 612B and 612C illustrate possible failed MAF sensor outputs not remaining between exemplary High limit HL and Low limit LL. Sensor output plot 612D illustrates a possible MAF sensor output which remains steady at a low level, while remaining between exemplary high and low limits, HL and LL respectively.

3. IMPLEMENTATION

3.1 Use Cases

The project was carried to suffice a number of use cases which could mainly be divided into two categories: analyzing driver behavior and analyzing vehicle condition. Each of these use cases are described in the following sections.

3.1.1 Analysis of driver behavior

Analyzing driver behavior is a much needed use cases for organizations such as Insurance companies, Fleet vehicle management systems and law enforcement authorities. Driver behaviors analyzes were done in two major aspects which are namely, reckless driving detection and driver anomaly detection.

3.1.1.1 Reckless Driving Detection

Reckless Driving detection is certainly something that is needed not only by some organizations but also by the whole society. It is a fact that reckless driving causes hundreds of deaths all over the year each minute. Law enforcement authorities need reckless driving detection methods to implement new laws. To prevent these road accidents. Fleet vehicle management systems need is detected to provide a better service to their customers through being capable of guarantying the safety of the passenger by being able to identify and recognize reckless drivers. Insurance companies pay several millions of dollars for damages caused by reckless driving. Therefore, a system which could detect would be beneficial for them to minimize payments in such cases. Even, most of the drivers do not want be driving recklessly. It happens because there are no systems for detecting reckless driving. Thus it was decided that reckless driving detection is a feature the system could provide.

3.1.1.2 Driving Anomaly Detection

The driving pattern is a something that is unique for a person. Therefore, an anomaly in the driving pattern most of the time indicates that either the driver has changed. Also it could indicate a significant change in the mental/physical status of the driver where, being drunk is the most common reason. Therefore, driving anomaly detection is required by insurance companies to identify whether the owner of the vehicle is the person who drove the vehicle that claims for a

damage. Therefore, the use case of detecting anomalous driving has become one of the most essential use cases.

3.1.2 Analysis of vehicle condition

Once a vehicle is bought, it is sold to many people at many times. Since all these users might not use the vehicle in a good way, it is best if a person could know the condition of a vehicle before purchasing. Currently available systems are only capable of detecting faulty conditions once the fault arises. Hence, a system which could perform a detailed analysis to predict such failures is much more beneficial for a buyer. Also, it is beneficial for the sellers from the sense that they could emphasize and prove the good condition of the vehicle they are planning to sell. Also, for the owner knowing the condition of the vehicle is important since it makes him feel comfortable if a failures is not to occur in the near future. The following sub sections discuss each of the use cases which are related to detecting the vehicle status and that the system addresses.

3.1.2.1 Sensor Failure Detection

Since all the modern vehicle models are computerized, the ECU completely relies on the outputs of various sensors such as the speed, rpm, coolant temperature, mass air flow rate, oxygen, etc. A malfunction in one of these sensors could have a huge impact on the ECU. Even though there exist systems that scan the OBD port to check whether there is a malfunction in any sensor, there is no system to predict the failure date of a sensor before it fails. The system needed to have this capability so that the vehicle owners could be prepared beforehand and act accordingly.

Among the factors that decide the fuel consumption of a vehicle, are the oxygen and the mass air flow. Also it is a known fact that every person wants his/ her vehicle to uses fuel efficiently. Therefore, it was decided to analyze and predict the failure of oxygen and mass flow rate sensors.

3.1.2.2 Alerting of High Fuel Consumption

The fuel consumption of a vehicle is one of the biggest concerns of an owner. The fuel consumption mostly rises up due to bad driving habits which happens without the knowledge of the driver. Therefore, a system that could alert the driver in high fuel consumption situations is needed by people.

3.1.2.3 Alerting of High Coolant Temperature

The coolant temperature of a vehicle, is one of the sensors that indicates the engine temperature. A higher coolant temperature indicates a higher engine temperature, therefore should be monitored frequently. Even though the coolant temperature is displayed in the dashboard of the vehicle, it is ignored by people due to many reasons where the most common can be seen as not paying special attention for the value of the dashboard, forgetting even the value is observed and not knowing the optimum range of operation. Therefore, a system which could alert the driver when the coolant temperature rises above its optimum range is of high importance for vehicle owners.

3.1.3. Summarization of trip details

The condition of a vehicle is highly dependent on the distance and the duration it runs, the type of the roads it runs on and the surrounding it is in. For example, once, a vehicle is driven in a dusty area, the air filters of the vehicle will have to be serviced earlier than what was expected. Also, the fuel consumption on a trip that was run in countryside is typically higher compared to others. Hence, providing a trip summary for each trip including the distance travelled, start and the destination, distance travelled, average fuel rate throughout the trip, the time of the day at which the trip started, etc. is important.

3.2 Data Collection and Fabrication

Data Collection was a main task of the project. Data was collected from various users to get a variety of data. Since the original data did not have necessary characteristics for some use cases, data had to be fabricated.

3.2.1. Data collection

Data was collected from various users owning different vehicles. Since the use case of driver anomalies required a large set of data to be collected from a single user, data was continuously connected from one user. This large set of data was collected from a Toyota 'Axio' vehicle whereas other sets were collected from vehicles of models 'Toyota Aqua' and 'Honda Fit'. Initially, a commercially available named 'Torque Pro' developed by Ian Hawkins was used for data collection. Once our Android app, which is a part of the project was built, it was used for data collection.

3.2.2. Data fabrication

Data fabrication was specifically needed for depicting a data set with a failing sensor. Since analyzing and predicting sensor failures needed data with failure as the input and since such a dataset could not be found, a normal data should have been fabricated to indicate a failure. Each of the following section describes how data was fabricated depicting an Oxygen sensor failure and a MAF sensor failure respectively.

3.2.2.1 Oxygen Sensor Failure

In Section 2.8.1, it was described that an oxygen sensor will typically generate up to about 0.9 volts when the fuel mixture is "rich" and will drop down to about 0.2 volts or less that, when the mixture is "lean" whereas when the air/fuel mixture is balanced or at the equilibrium point of about 14.7 to one, the sensor will read around 0.45 volts [52]. Fig 3.1 depicts the min max variation of the Oxygen sensor that was present in the collected data set.

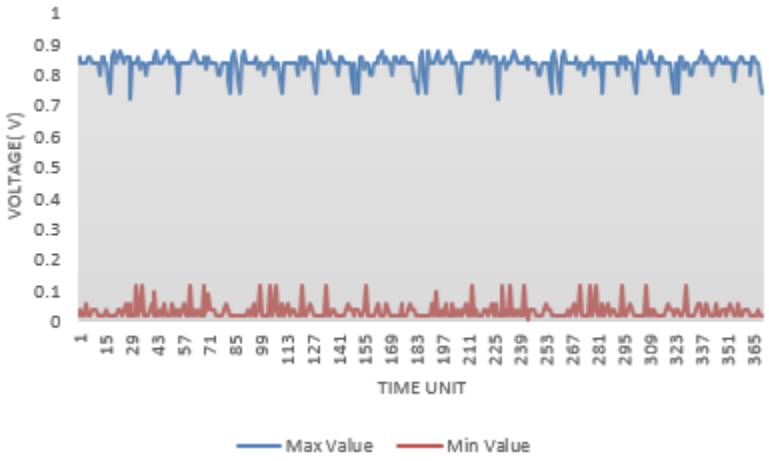

Fig. 3.1. Min max variation of a healthy O2 Sensor.

Once the sensor starts failing, the output voltage in rich state drops down and the output voltage in the lean state rises up (i.e., the amplitude reduces once the sensor starts failing). Therefore, the desired min max variation of the output voltages was as in Fig 3.2 and the original data set was superimposed with it to get the data set with failing Oxygen sensor voltages as shown in Fig 3.3.

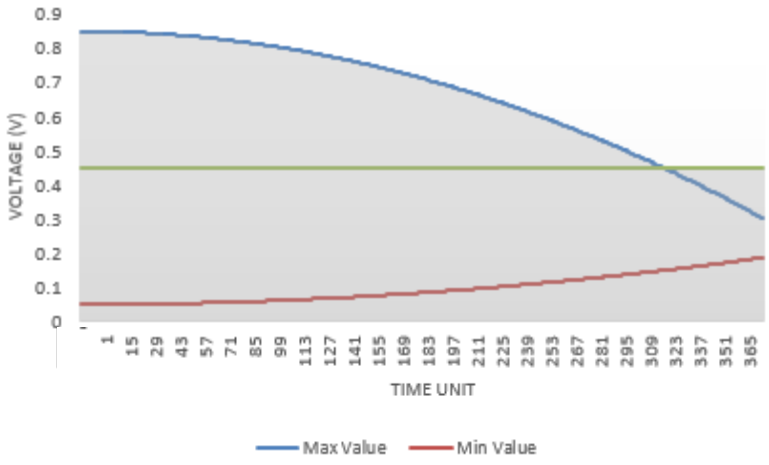

Fig. 3.2. The Desired change in voltages.

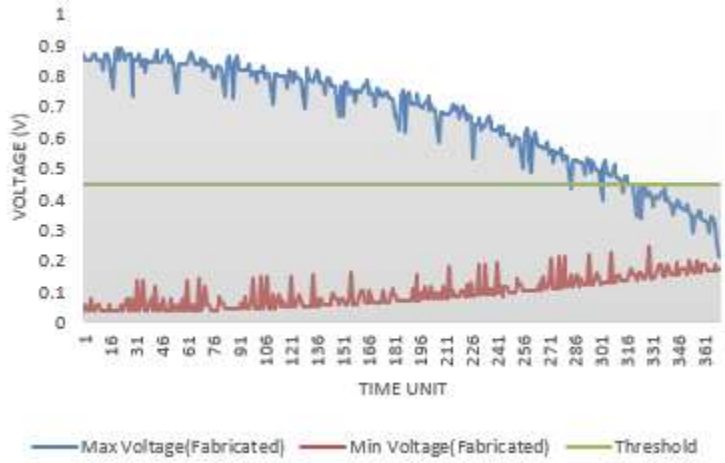

Fig. 3.3. Min max variation of a failing O2 Sensor

3.2.2.2 MAF Sensor Failure

For a known engine displacement, the engine airflow can be calculated at 100% VE (volumetric efficiency) in sea-level-standard-day cubic feet per minute (scfm) as in eqn. 1.

$$100\%\ \text{VE AIRFLOW (scfm)} = \text{DISPLACEMENT (ci)} \times \text{RPM} / 3456 \qquad (1)$$

Therefore it can be seen that the Mass Air Flow has a linear relationship with the engine rpm. Therefore, the output of a healthy MAF sensor changes linearly with the engine rpm. The relationship was seen in the collected data set in higher rpms is as seen in Fig. 3.4.

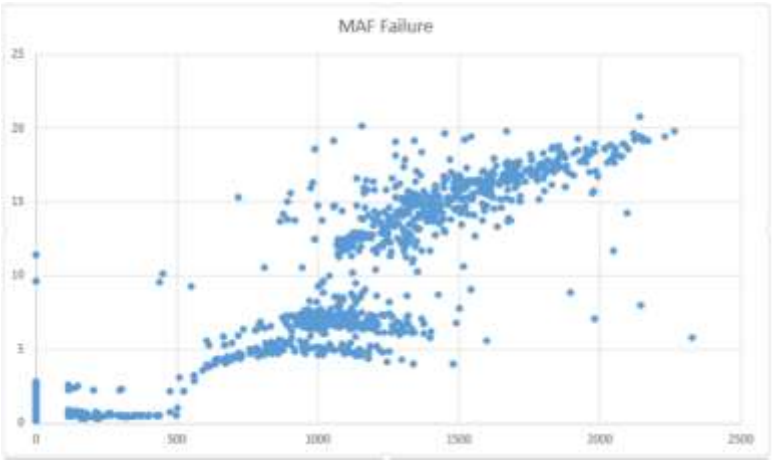

Fig. 3.4 The Engine RPM vs the MAF value of the collected dataset

The gradient of the line between the engine rpm and MAF value reduces with time once the sensor starts failing. Therefore, the dataset needed to be changed in order to depict this difference. The data changed to indicate a somewhat failed sensor is shown in fig. 3.5.

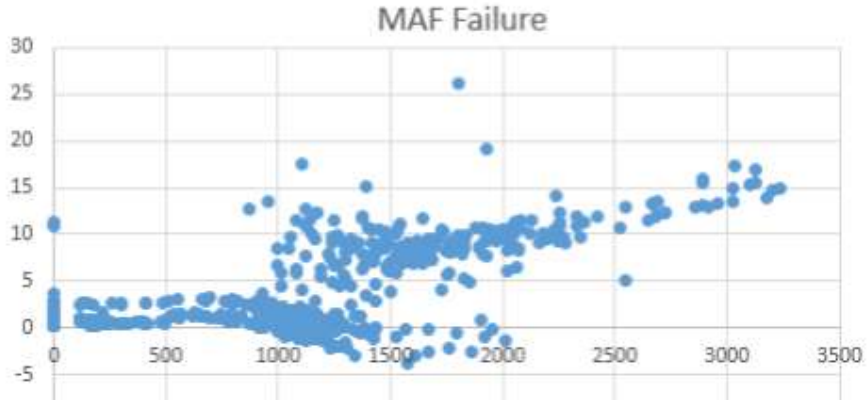

Fig 3.5. Engine rpm vs. MAF value when the sensor is somewhat failed

3.2.3. Vehicle simulation

The flow of data starts from the car engine to the smartphone. Therefore, the system needs to be connected and communicating with the PCM (Powertrain Control Module) throughout project. Since this is not a practical situation, the first component that was developed was a simulator to simulate the PCM.

The simulator consists of two components as follows.

1. CSV Reader

2. Communicator

3.2.3.1 CSV Reader

The actual datasets that were collected from various sources were stored in .csv files where each trip is stored as a separate file. The simulator is developed with the capability of reading a given .csv file. The user interface of the simulator is shown in fig 3.6.

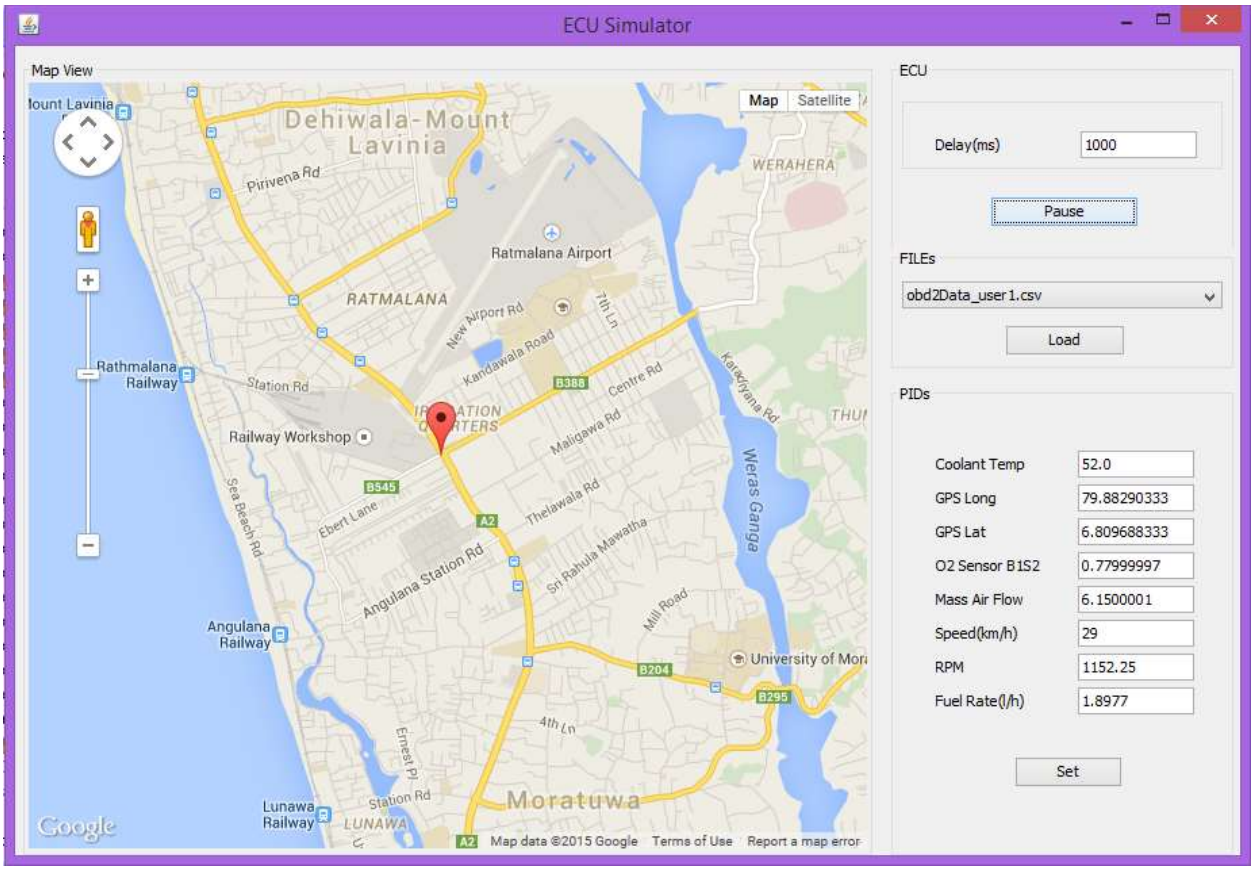

Fig 3.6. Data simulator

3.2.3.2 Communicator

Communicator acts as a vehicle where an OBD2 adapter is connected. Once the CSV Reader reads a line in the file, it is separated into single data values. Each of these data values are sent via Bluetooth to the Android App upon request using the OBD2 protocol. The communicator supports both OBD2 and AT command requests by the app.

3.3 Implementation

3.3.1 Architecture

The proposed system is capable of collecting, storing and analyzing vehicular data for a long period of time. Vehicular data are collected using an OBD-II to Bluetooth interface and are sent to backend cloud servers using a smartphone via 3G/4G connection as shown in Fig. 3.1.

Fig. 3.7. The overview of the proposed system

Fig. 3.2 shows the solution architecture which is designed in a scalable and an extensible manner so that the system can be extended to have a rich set of functionalities supporting numerous vehicles and servers as needed.

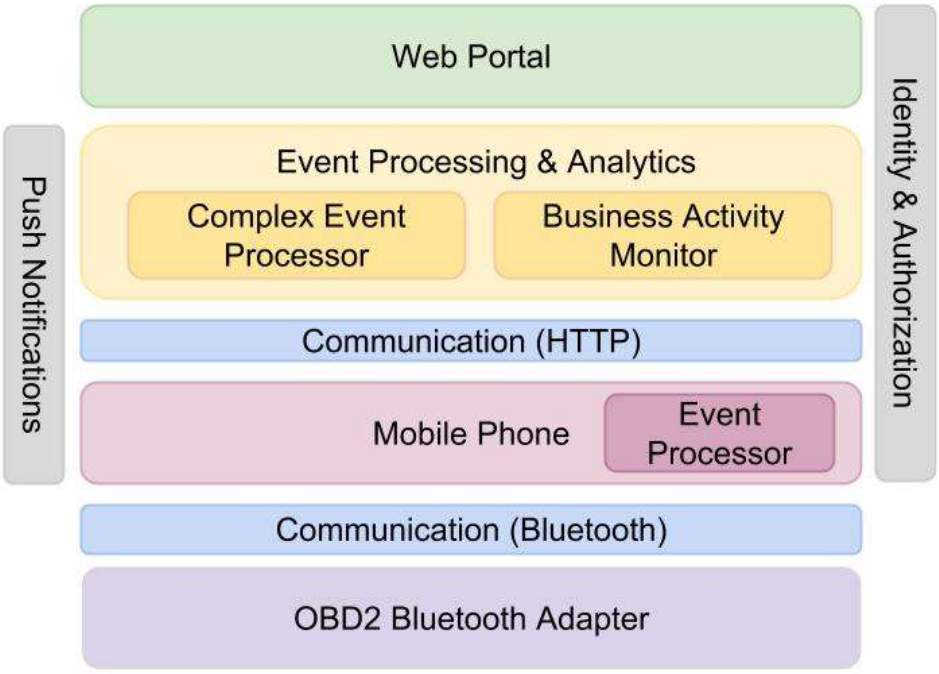

Fig. 3.8. The architecture of the system

The Android app is one of the key elements of the system since it does the duty of a data transmitter between the vehicle and the backend servers while performing the task of the view layer (see Fig. 3) as well. The three core tasks it is responsible of is as follows.

1. Receiving Data from the OBD2 Adapter
2. Monitoring Real Time Data
3. Selective Transmission of Data to the Backend

Each of these tasks are described in Section 3.3.2. The user is given the freedom of choosing whether data is transmitted to the backend or not. If it is not allowed, the analyses performed by the whole system are limited to the ones that are done within the app. For major analyses, the user needs to enable data transmission to the backend. Once it is allowed, data is transmitted to CEP and BAM through the data connection of the phone.

Fig. 3.9. The dashboard of the Android app

Complex event processing can be regarded as a service that receives and matches lower-level events and generates higher-level events in real time. Simply, it is a component that responds to event streams in an event driven manner. Complex event processing is required to happen at two stages within system depending on the weight of the processing. A portion of the processing is handled by the Android app before transmitting data to the backend server allowing less consumption of the bandwidth. The library, WSO2 Siddhi [4] which is the core processing engine of WSO2 CEP is used in the processing done in within the Android app. Here, data from the engine, are received as event streams (e.g., speed stream consisting of time stamp and speed, fuel stream, coolant temperature stream, etc.). The Siddhi engine decides the importance of the received streams and depending on the information they provide, it decides two things. For simple use cases such as coolant temperature monitoring it makes the app generate an alert. For complex use cases, it transmits the filtered streams to the backend servers. WSO2 CEP, the other portion of CEP, is run on the backend since it has the freedom to work with a large database performing complex processing. The CEP at the backend is responsible for detecting unusual patterns and generating alerts in real time.

The system is capable of performing long term analyses too. By identifying patterns, it should be able to predict undesirable outcomes such as failures of sensors in advance. This is allowed by using WSO2 BAM as the long term analyzer at the backend which is capable of collecting, storing

and analyzing data. It receives events published by the app and stores them in a Cassandra database where analyses are performed regularly to identify gradual changes in them.

Where some of the result of analyses are notified to the user in the form of alerts, all the results of the analyses performed in the backend are displayed in a web portal as shown in Fig. 4. This way, drivers themselves could monitor their driving behaviors and changes in the vehicle condition and other organizations could monitor their customers' behaviors and conditions of the vehicles that are owned.

3.3.2 Android app

The Android app, which is the key component which enable data transmission between the OBD-II adaptor and the backend servers, plays three major roles throughout the process.

1) Receiving Data from the OBD2 Adapter:

The app is capable of connecting to the ELM-327 adaptor via Bluetooth and communicating with the vehicle using OBD-II Parameter IDs (PIDs), each providing some information about the vehicle, in order to receive vehicular data. The received data are logged inside the app or displayed in the user interface using graphs and meters which can be added dynamically. The architecture of the app has also given extensibility for adding PIDs which are not in the app.

2) Monitoring the vehicle:

The app consists of a real time data processor which is based on WSO2 Siddhi [4] which has a capability to act according to a predefined set of queries and has been ultimately used to monitor the vehicle using OBD-II data. For example, queries are added to alert the driver if the vehicle is running with a high fuel consumption rate or a high coolant temperature for a considerably long period of time and to summarize trip details in real time. Siddhi supports adding queries dynamically hence giving extensibility to support future monitoring task as well. Alerts are generated when unusual behaviors are detected and the results are shown in the app to the driver.

3) Selective Transmission of Data to the Backend:

A user has to pay for each megabyte that is consumed. Therefore, the network bandwidth usage of the app is an important fact to consider. Hence, the app is implemented in such a way that

only useful data are transmitted over a 3G/4G network where irrelevant data are dropped. Also data streaming is reduced where possible in order to minimize battery consumption. The Siddhi module assists the app for both these tasks.

The app consists of a Bluetooth command service which connects with the OBD2 Bluetooth adapter with the serial UUID. Once it is connected to adapter first it configures the adapter using the AT commands to return data in a more convenient format into the app using ATE0 (ECHO OFF), ATS0 (NO SPACE) and ATH0 (HEADERS OFF) commands sent by the app. Once it is done the adapter is also configured to auto select the protocol of the vehicle using the ATSP0 command. Once that command is sent the adapter starts searching for the protocol of vehicle from the known 5 standard OBD2 protocols. Then the app send the following OBD2 commands 0100, 0120, 0140 and 0160 to identify the supported PIDs by the vehicle among the PIDs from 0x00 to 0x80.

Since there are many standard PIDs, it is not practical to implement them separately. The app addresses that problem by keeping a configuration file for PIDs it supports and auto generates the logics to retrieve the PID data from the OBD2 adapter and it will be automatically available for the all the other functionality of the app including displaying and logging. A sample configuration file is as shown in fig 3.4. The app uses the Expr java library [54] to evaluate expressions dynamically and calculate the actual values from the raw bytes retrieved from the adapter. This method gives the extensibility to the app for adding a new PID with a less effort.

Fig. 3.10. PID configuration file

Once the actual PID values are calculated they are transformed to a common message format. Because of that there is a low coupling between the OBD2 client, PIDs and other components of the system which rely on PIDs. The major components that communicated via the common message format is illustrated in the architecture diagram of the App shown in fig 3.5.

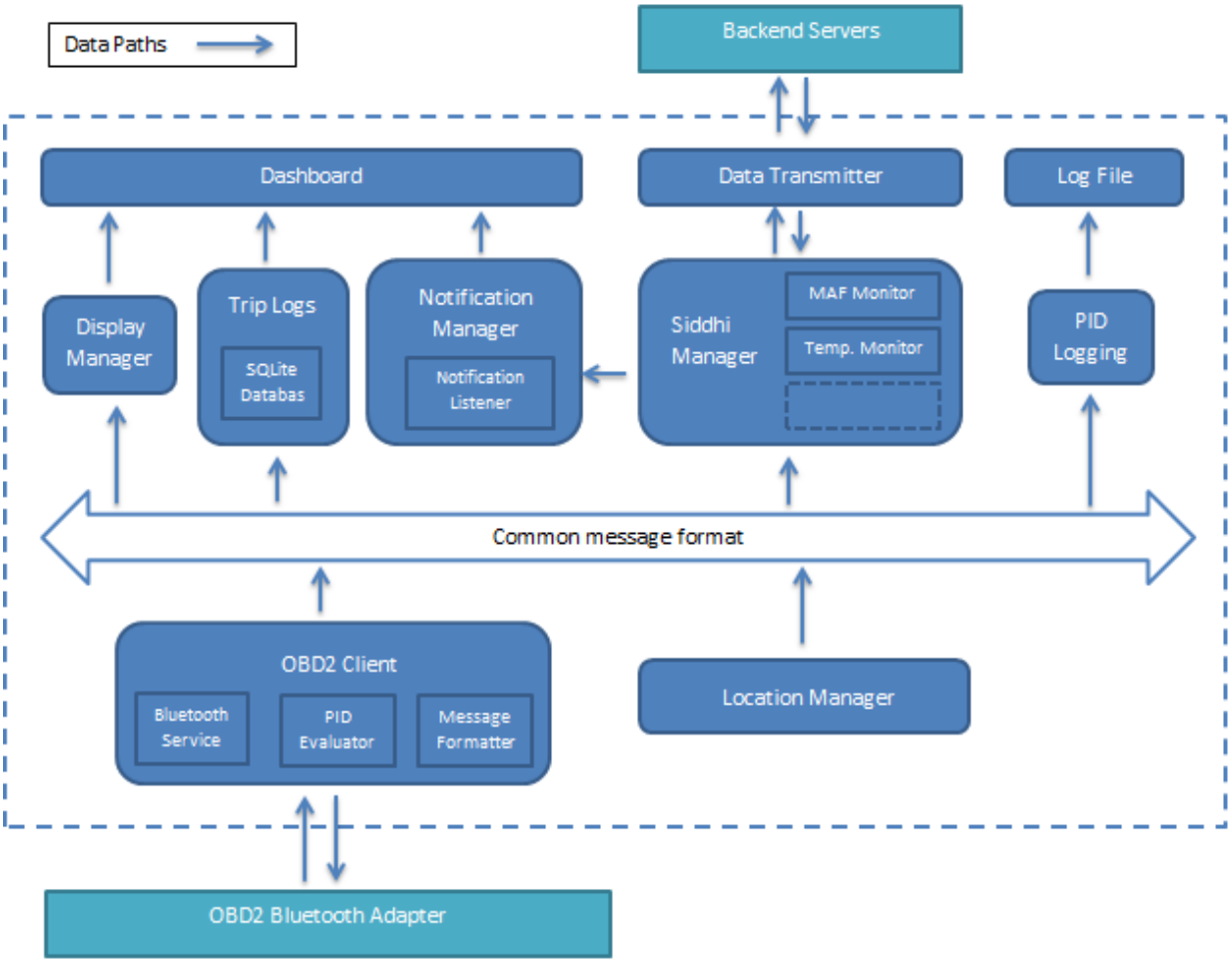

Fig. 3.11. The architecture of the Android App

Display manager is responsible for retrieve data through the common format and publishing them to the app's dashboard. The dashboard consists of graphs and dial meters which can be added dynamically. Those gadgets and the Display manager communicates though a pub-sub pattern. When a gadget is added to the dashboard, it gets registered to the display manager with the PID it needs to be displayed. When the required PID value is received, Display manager publishes it to the gadgets which are subscribed to the corresponding PID.

Trip logs component stores current trip details in once a trip is over. It includes a configured lightweight SQLite database. The stored trip log details include total distance, average fuel efficiency and total time.

For the notifications sent from the server side sensor analytics and app side monitoring tasks are connected with the Android system notifications. Because of that the alerts are readily available to the user and the user can be alerted even when the user is driving the vehicle using message tones and custom vibrations.

The app includes a Siddhi engine which does the real time sensor monitoring within the app and preprocessing data before sending to the backend servers. All the processing is done via the Siddhi queries defined in Siddhi Query Language. Siddhi Manager consists of the Siddhi engine and a set of Monitors that examines PIDs in real time. Implemented monitors includes fuel rate monitor, coolant temperature monitor, MAF and O2 sensor monitor. Once the pre-processing is done by the monitors, data is transmitted to the backend servers if further analysis is needed.

PIDs can also be logged within the app. The PIDs which needs to be logged, logging frequency and the path of the log file can be configured by the user.

3.3.3 Driver behavior

Driver monitoring was basically done in two forms. Reckless driving monitoring, which is mostly wanted by most of the organizations such as Insurance companies and low enforcement authorities is one of the displays the recklessness of the drivers driving pattern within a certain period of time such as 20 hours, one week/month and three months. The second, is driving anomaly detection which is important somewhat for the above organizations but more important to drivers themselves to get alerted when he's deviated due to stressed, drunk, distractions, etc. The implementation of these two use cases are described in the Sections 3.3.3.1 and 3.3.3.2 respectively.

3.3.3.1 Reckless Driving Detection

Since many of the literature stated that the sudden variation of the longitudinal acceleration is a good measurement to detect reckless driving, it was decided to be used in this use case.

The speed of the vehicle can be read in real time from the OBD2 adapter. Since it is the acceleration/deceleration that is in concern, the Siddhi engine transforms speed streams into acceleration/deceleration streams by considering consecutive speed readings using the following Siddhi query. A sample Siddhi query is shown below.

```
from a=obd_speed,b=obd_speed
select b.speed-a.speed as speedDifference,
b.time - a.time  as timeInterval, b.time as timeStamp
insert into obd_accele_calculation;
```

Calculated acceleration/deceleration is compared with a threshold and detected whether it is reckless or not. The threshold used in the system, which is 4.5m/s^2 is recommended by the American Association of State Highway and Transportation Officials. The binary number 1 and 0 are assigned to the points which have an acceleration/deceleration higher than the threshold and to normal points respectively. And the total number of 1's are counted over a predefined period of time (e.g., 2 minutes). The calculated number of counts are then classified according to the driving cycle (i.e., traffic, normal and highway) of the trip. The driving cycle is determined by the average speed of the vehicle throughout 10 minutes. The count, together with the class is sent to the backend server periodically as two separate streams for acceleration and deceleration.

Data are published to acceleration and deceleration streams via an http input adaptor. WSO2 BAM by default provides many different input and output adaptors which can be associated with event streams [47]. Http adaptor is one such adaptor which enables events to be published to streams through a rest call. Once data is published to these event streams, data is written into a Cassandra column family with the name of the event stream. A script is written in to read data in these column families and to write into either another column family or a SQL table with some modification. The script is written in HiveQl. Therefore, a hive table should be first created and data should be loaded into that from the Cassandra column space in order for performing analyzes. The Hive query which was written to create and load data into 'accelcounttable' from the column family for which the input events are published is shown below.

```
CREATE EXTERNAL TABLE IF NOT EXISTS accelecounttable (key STRING
,acceleCount BIGINT, decceleCount BIGINT, class STRING, time
BIGINT, vin STRING, username STRING ) STORED BY
'org.apache.hadoop.hive.cassandra.CassandraStorageHandler'
WITH SERDEPROPERTIES (
'wso2.carbon.datasource.name'='WSO2BAM_CASSANDRA_DATASOURCE',
"cassandra.cf.name" = "recklessDrivingStream",
"cassandra.columns.mapping" = ":key, payload_acceleration,
payload_deAcceleration ,payload_class , payload_timeStamp,
payload_vin , payload_username");
```

According to the driving cycle for which the input dataset belongs to, the acceleration count data should be written to separate tables. The query to find the total acceleration count within two minutes with the class label 'A' is shown below. Once the acceleration count is summed, it is written to a Hive separate table which corresponds to a summarized SQL table.

```
insert into table classacount select  (ROUND(time / 120000) *
120000) as time , vin , username , class,
sum(acceleCount) as acceleCount , sum(decceleCount) as
decceleCount
from accelecounttable
where class = 'A'
group by (ROUND(time / 120000) * 120000) ,class, vin, username;
```

As a measure of recklessness, the average high acceleration/deceleration count per hour is displayed. This summarization happens through the following query.

```
insert into table recklessaverage select   vin , username  , class ,
(sum(acceleCount)/count(acceleCount)*120) as acceleAverage , (sum
(decceleCount)/count(decceleCount)*120) as decceleAverage
from classacount
group by class, vin, username;
```

This way data received from the app are summarized by WSO2 BAM as hourly, weekly, monthly and every three months and stored in an SQL database.

3.3.3.2 Driving Anomaly Detection

It was found after some research that it is not the speed pattern but the acceleration pattern that is unique for each person but the acceleration pattern. A Markov Model was used to determine the difference between the current pattern and the existing pattern of a driver vehicle combination. Identifying driving anomalies happens in three stages.

1. Pre-processing
2. Creating and updating the model
3. Determining Anomalies

As in the use case of reckless driving detection, since it is the acceleration that is of concern, the Siddhi engine transforms the speed streams into acceleration streams. This happens through the following query.

```
from a=speed_stream,b=speed_stream select a.timestamp as
timestamp,(b.obd2_speed-a.obd2_speed)*5/(18.0*(b.timestamp-
a.timestamp)/1000) as accel insert into accel_stream;
```

Then, these acceleration streams are a converted to acceleration transition streams which consist of the timestamp, previous acceleration and the current acceleration. The Siddhi query which converts the acceleration streams into a streams with previous acceleration and current acceleration (known as the 'accel_trans_stream') is shown below.

```
from c=accel_stream, d=accel_stream select c.timestamp, c.accel
as accel1, d.accel as accel2 insert into accel_trans_stream;
```

These acceleration transition streams are transmitted to the backend twice: once to the CEP in real time for calculating whether the readings are in accordance with the normal pattern, and next to the BAM daily for updating the model.

A Markov Model has states and probabilities associated to transitions between these states. Since the model used for the use case of reckless driving is implemented as a Markov Model, acceleration values are considered as states and the probability of changing from one acceleration to another is defined as the transition probability. The transition diagram for this is shown in fig 3.6.

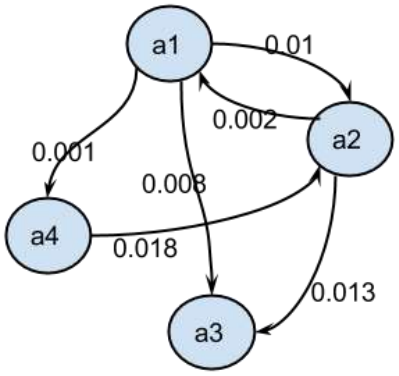

Fig. 3.12. Acceleration Transition Diagram

The BAM processes the events (acceleration transition streams) received from the app and stores the model in an SQL table using the following hive query.

```
INSERT OVERWRITE TABLE accelTransSql
SELECT username, vehicle, accel1, accel2, count(*) as accel_count
FROM AccelTransCas
GROUP BY username, vehicle, accel1, accel2;

INSERT OVERWRITE TABLE totalAccelSql
SELECT username, vehicle, count(*) as count_accel
FROM AccelTransCas
GROUP BY username, vehicle;
```

The first of the above two queries sum up the number of times a particular transition has occurred and the second calculates the number of total acceleration transitions that are available for a user. Therefore, a particular transition probability can be calculated by diving the number of times that transition has occurred by the total number of transitions. These two tables are updated daily with new data from the user. Therefore, the more accelerations the BAM receives, the more accurate the model becomes.

Determining anomalies happens almost real time. Anomaly is detected by calculating the probability of the Markov Chain resulted by the acceleration readings within the past five minutes. But a single transition that has zero probability results in zero probability of the chain. Also there is a need to multiply each probability. The above problem is avoided and the calculation is simplified by considering property emphasized by (1).

$$y = \sqrt[3]{a * b * c} \rightarrow \lg(y) = \frac{\lg(a)+\lg(b)+\lg(c)}{3} \quad (1)$$

Therefore, rather than calculating the probability of the Markov Chain, the average acceleration transition probability is calculated for the detection of anomalous driving. Whenever the CEP receives an event stream (acceleration transition stream), it calculates the average transition probability for 5 minutes using a sliding window on the timestamp with the aid of the transition table stored in the database by BAM. The script written inside WSO2 CEP is shown below.

```
define table acceltranssql(username string, vehicle string,
accel1 float, accel2 float, accel_count int) from
('datasource.name'='fyp_data_source',
'table.name'='acceltranssql');
  define table totalaccelsql(username string, vehicle string,
count_accel int) from ('datasource.name'='fyp_data_source',
'table.name'='totalaccelsql');
  define table accelProbTableDemo(timestamp long, username
string, vehicle string, avg_prob double) from
('datasource.name'='fyp_data_source',
'table.name'='accelProbTableDemo');
  from accel_trans#window.length(300) as t unidirectional join
acceltranssql as at
  on t.username==at.username and t.vehicle==at.vehicle and
t.accel1==at.accel1 and t.accel2==at.accel2
  select t.username, t.vehicle, sum(at.accel_count) as
sum_accel_count, t.timestamp
  group by t.username,t.vehicle
  insert into accel_prob_stream for expired-events;
  from accel_prob_stream as s unidirectional join totalaccelsql
as st
  on s.username==st.username and s.vehicle==st.vehicle
  select s.username as username, s.vehicle as vehicle , convert
(s.sum_accel_count,double)/(st.count_accel*300) as probability,
s.timestamp
  insert into prob_stream;
  from accel_prob_stream as s unidirectional join totalaccelsql
as st
  on s.username==st.username and s.vehicle==st.vehicle
  select s.timestamp, s.username, s.vehicle as vin, convert
(s.sum_accel_count,double)/(st.count_accel*300) as avg_prob
  insert into accelProbTableDemo;
  from prob_stream[probability < 0.00018]
  select username, vehicle, probability, timestamp output last
every 15 sec
  insert into anomaly_stream;
```

The averaged probability of each chain of accelerations is calculated and once it drops down a predefined threshold, an alert is sent to the driver.

3.3.4 Vehicle condition

Modern computerized engine control systems rely on inputs from a variety of sensors. Among the sensors which are present in modern computerized vehicles, Mass Air Flow sensor and Oxygen sensor are two most important sensors that crucially determine the engine performance, emissions and other important functions. OBD-II system may take some time to identify malfunctions so in

the meantime lots of wastage of fuel can result. Sensors do not fail suddenly but gradually. Therefore, the system was developed to detect impending failures so that they can be identified beforehand. Each of the vehicle diagnostic mechanisms that are handles in the system are described in the following sections.

3.3.4.1. Oxygen sensor impending failure detection

As explained in section 2.8.1, a healthy O2 sensor's voltage operates within the range 0.2 V and 0.9 V with a mid-level of 0.45 V. Once it goes totally above or below 0.45 V, it is considered as a failure. Since the sensor fails progressively a regression analysis can be performed to investigate the current status of the sensor.

Oxygen sensor voltage can be retrieved from the adapter using the PID 14. Once it is received by the app's OBD2 client, it is retrieved by the O2 Monitor in the Siddhi Manager. Within the O2 Monitor O2 voltage data are added to a moving window of 30 minutes defined in Siddhi and an aggregation is done once in 30 minutes for getting the min and max voltages. A sample Siddhi query is given below.

```
from O2DataStream#window.time(30 min) select timestamp,max
(obd2_o2sensorV_B1S2) as maxval,min(obd2_o2sensorV_B1S2) as minval insert
into O2MaxMinStream;

from O2MaxMinStream select timestamp,maxval,minval output last every 30 min
insert into O2DataOutStream;
```

The data are stored within the app while the vehicle's engine is started. Once the trip is over (the engine is turned off), the stored data are sent to BAM using the input http adapter defined in BAM. Once sent they are stored in a Cassandra database inside BAM.

For a healthy sensor, calculated min voltage should be always below 0.45 V threshold and calculated max voltages should be always above 0.45 V threshold. So a regression analysis can be done to min and max voltages separately using BAM analytic scripts. Using the two regression lines the failure date can be estimated as illustrated in the following graph.

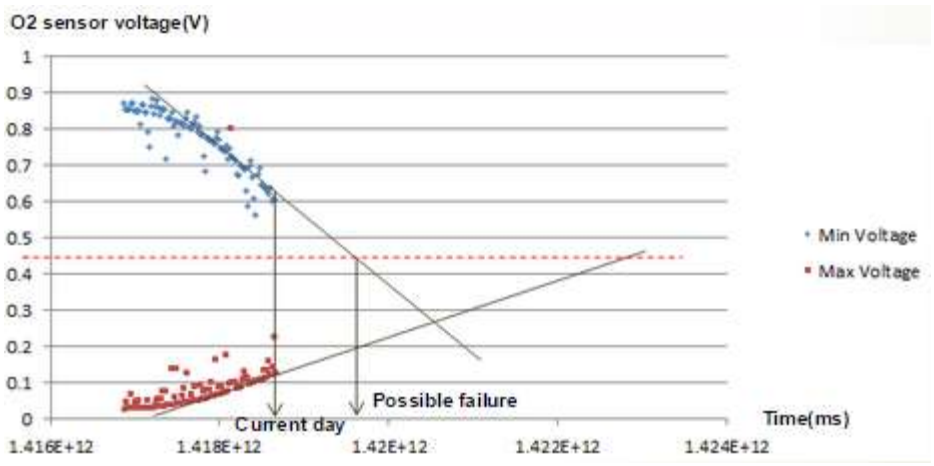

Fig. 3.13. Regression lines for min and max voltages for O2 sensor

The following BAM analytic script does the regression analysis for min voltage.

```
insert overwrite table O2StatTableMax select max(timemins) , vin , count
(timemins) , AVG(timemins) , SUM(timemins), SUM(timemins*timemins),
AVG(maxval) AS meanY , SUM(maxval) AS sumY, SUM(timemins*maxval) AS sumXY
FROM O2CasDataTable join var_maxtime where timemins>maxtime-
${hiveconf:timegap} and sensor='b1s1' GROUP BY vin;

insert overwrite table O2RegTableMaxTemp select timemins, vin , (n*sumXY -
sumX*sumY) / (n*sumXX - sumX*sumX), (meanY - ((n*sumXY - sumX*sumY) /
(n*sumXX - sumX*sumX))*meanX) from O2StatTableMax;
```

After the regression analysis is done, following equations can be used to estimate the failure date of the sensor and the current wear level.

$$\text{Failure Date Min V} = (0.45 - c_{min})/m_{min} \quad (2)$$

$$\text{Failure Date Max V} = (0.45 - c_{max})/m_{max} \quad (3)$$

$$\text{Wear Level Min V (\%)} = ((m_{min} * t + c_{min}) - 0.1) / 0.35 * 100 \quad (4)$$

$$\text{Wear Level Max V (\%)} = (0.9 - (m_{max} * t + c_{max})) / 0.45 * 100 \quad (5)$$

$$\text{Failure Date} = \text{Min(Failure Date Min V , Failure Date Max V)} \quad (6)$$

$$\text{Wear Level (\%)} = \text{Max(Wear Level Min V (\%),Wear Level Max V (\%))} \quad (7)$$

Where c_{min} = intercept of min V, m_{min} = gradient of min V, m_{max} = gradient of max V, c_{max} = intercept of max V

3.3.4.2. MAF sensor impending failure detection

RPM (rad/s) and MAF (gram/s) data is sent to the BAM within first 15 minutes in each trip. Then it calculates the gradient (m) of the each trip. MAF has a strong linear relationship between RPM. When the sensor is failing the gradient of the linear relationship reduces with the time. This reduction of the gradient is used to detect MAF sensor impending failure detection. The regression method is used to calculate the gradient.

$$m = \frac{n * \sum_0^n (RPM * MAF) - \sum_0^n (RPM) \sum_0^n (MAF)}{n * \sum_0^n (RPM * RPM) - \sum_0^n (RPM) \sum_0^n (RPM)}$$

Then calculated gradient values are stored with the time. The gradient graph w.r.t the time illustrates as follows.

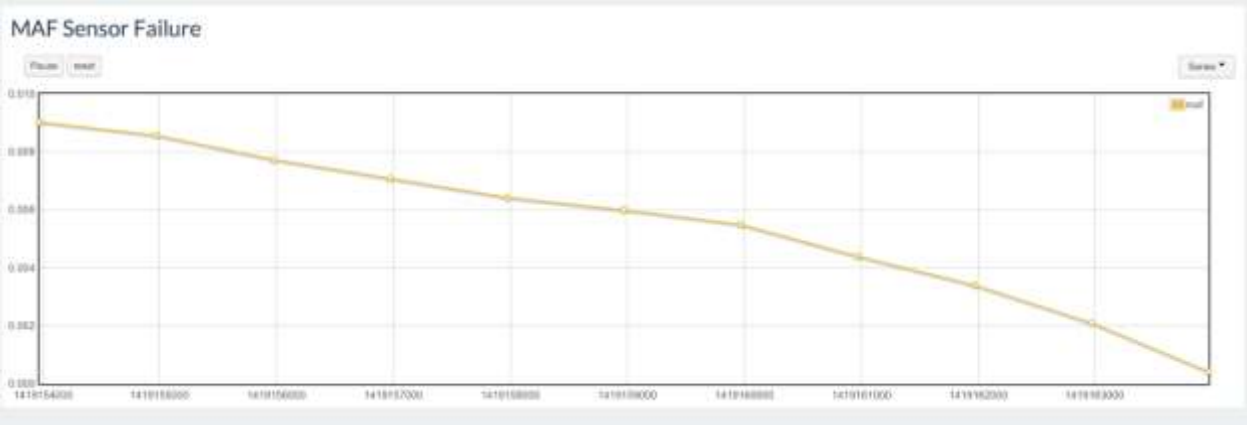

Fig. 3.14. Gradients of regressions performed for MAF and RPM

Then it is used the regression analysis again to those gradient values. Then it is calculated the wear level for the MAF sensor failure like below graph.

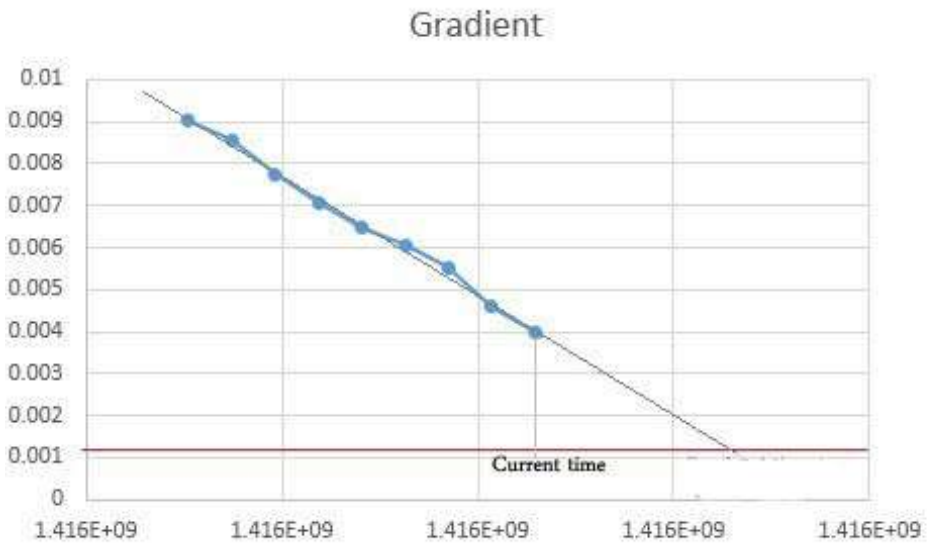

Fig. 3.15. Final regression line for gradients between MAF and RPM

According to the current situation, it will be evaluated the wear level of the MAF sensor.

3.3.4.3. Fuel Economy Monitoring

The app is capable of monitoring the fuel economy of the vehicle which done completely within the app using the fuel rate (PID 5E) and speed (PID 0D). The app calculates the average fuel consumption and the average speed for a given time window using the Siddhi engine. If the ratio of average fuel consumption and the speed exceeds a certain threshold it will alert the user immediately.

3.3.4.4. Engine Coolant Temperature Monitoring

The app retrieves the coolant temperature of the vehicle using the PID 05 periodically. The retrieved values are monitored continuously using Siddhi to check whether it exceeds the operating engine temperature. If it exceeds the user is alerted immediately by the app's notification system.

3.3.5 Other features

3.3.5.1 ***Trip Logs***

Android app is capable to provide trip details to the user. The trip duration is taken as the period when the app is start to close in one trip. It is able to provide the name of the start place and end place with start time and end time. Google geocoding is used to get name of the place which provide through Google map API. When the app starts, it will get the latitude and longitude through the smart phone and do reverse geocoding mechanism to get place of the name. Same operation is

done with the end of the trip. All trip data are saved to the in-built sqlite database in Android app. It is also capable to provide the trip distance and average fuel efficiency in each trip. There is a distance calculator which calculated the distance in real time. It gets the vehicle speed in real time and calculate the distance. There is also an average fuel efficiency calculator which gets the fuel efficiency in real time and get the average at the end of the trip.

4. RESULTS

4.1. Android App

Android app is capable of connecting to the OBD2 adapter via Bluetooth and communicating with it using the standard OBD2 protocol. It can communicate with the vehicle's ECU, obtain vehicle PIDs and display them to the user using dashboard gadgets. Once it is connected to the adapter, the first "adapter" indicator turns to green from red. The next "car" indicator will turn to green once the adapter identifies the OBD2 protocol of the vehicle and it is really to send data to the Android app. The gadgets includes graphs and dial meters which can be added dynamically. The app can also identify the parameters which is supported by the vehicle and indicating them to the user. As showed in figure 4.2, supported OBD2 parameters are indicated in green in the list. Any supported parameter can be added to the dashboard to see data in real time.

Fig. 4.1. App dashboards

The app also has the capability to log PIDs in real time. PIDs which needs to be logged, logging interval and the location of the log file can be configured by the user. In addition to the PIDs obtained from the vehicle, the app can show and log current location taken based on network based or GPS based which is also configurable as shown in figure 4.2.

The app monitors some of the engine parameters including fuel rate and coolant temperature and examines whether they operate within the proper range. They will be discussed further in later sections. Those monitoring tasks can be configured using the Android app as shown in figure 4.3.

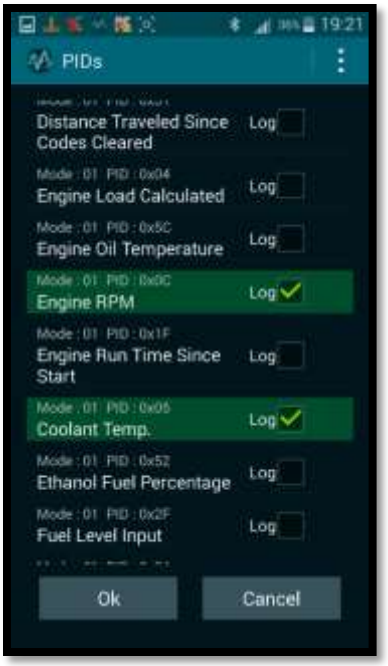

Fig. 4.2. App PID and log screen

Fig. 4.3. App enable feature screen

4.2. Driver Behavior

4.2.1 Reckless driving

The system is capable of calculating rapid acceleration counts of the driver in each speed class and displaying them in the web portal. It will also calculate and show the historical average rapid acceleration count per hour based on each speed class.

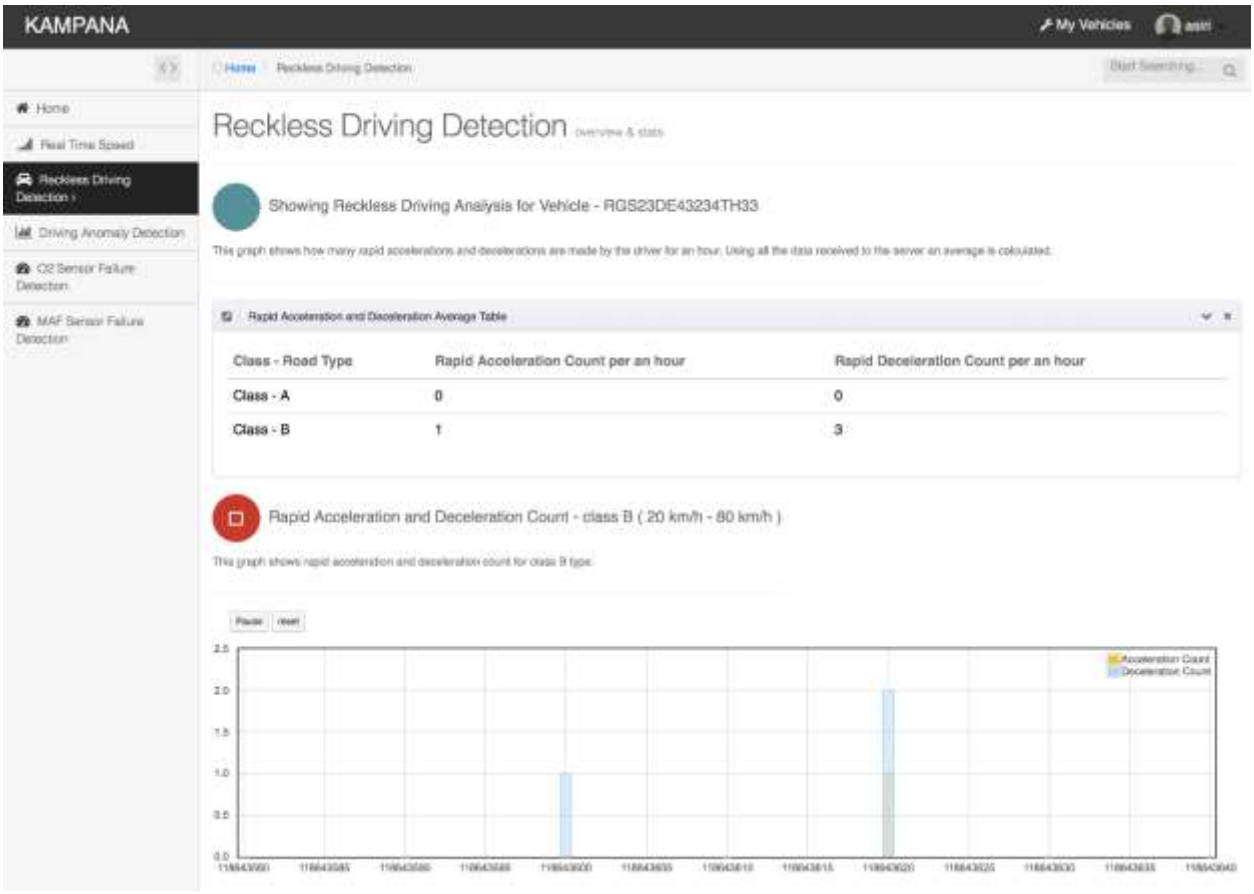

Fig. 4.4. Reckless driving web portal

4.2.2. Anomaly

The system is able to detect anomalies by calculating the probability of the Markov Chain resulted by the acceleration readings within the past five minutes. Once an anomaly is detected, it is notified by the server via the app immediately. As indicated by figure 4.5, the average probabilities which go below the red threshold line are identified as anomalous.

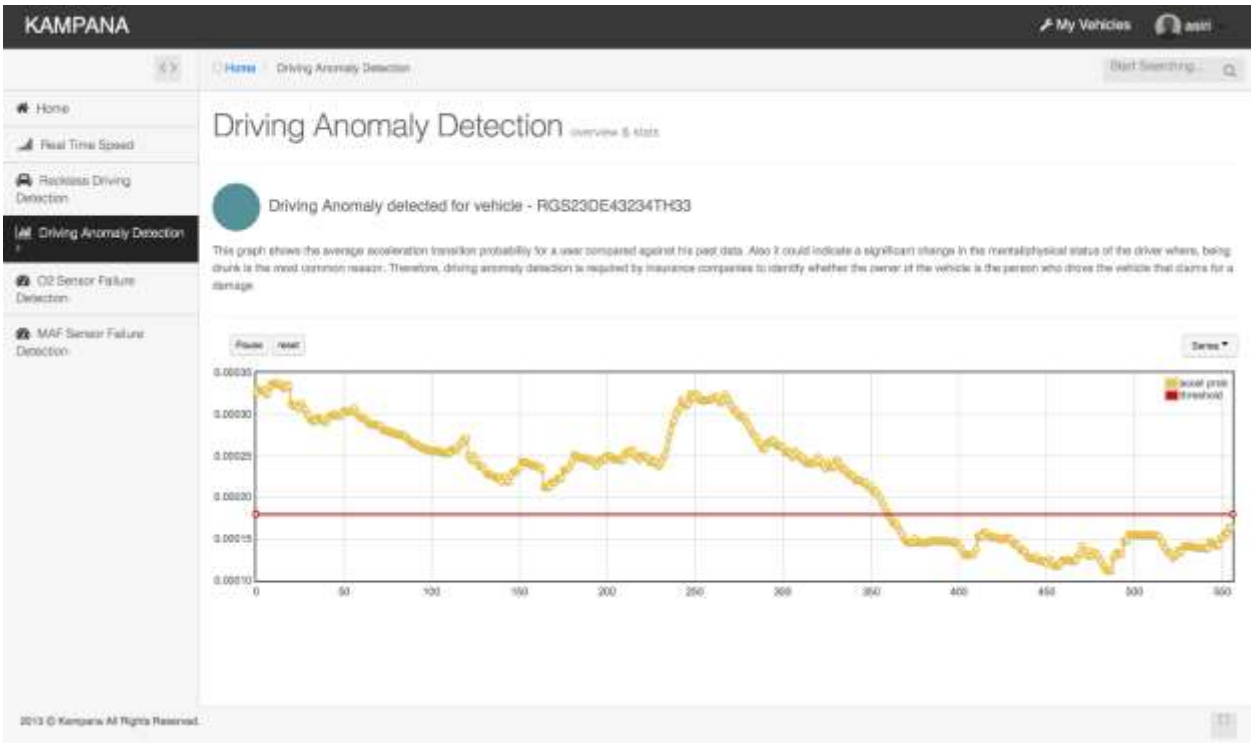

Fig. 4.5. Driving anomaly web portal

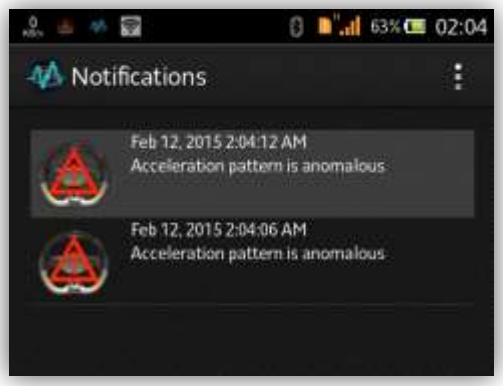

Fig. 4.6. Driving anomaly app notifications

4.3 Vehicle Condition

4.3.1. Oxygen sensor

BAM executes the analytic scripts once a day and performs the regression analytics to investigate the current status of the sensor to check whether there is an impending failure. It estimates the

wear level and if it exceeds 50%, it will also estimate the number of weeks it would take for the wear level to reach 0% and notifies the user immediately through the Android app's notification system. The user can also see the status of the sensor via the web portal at any time.

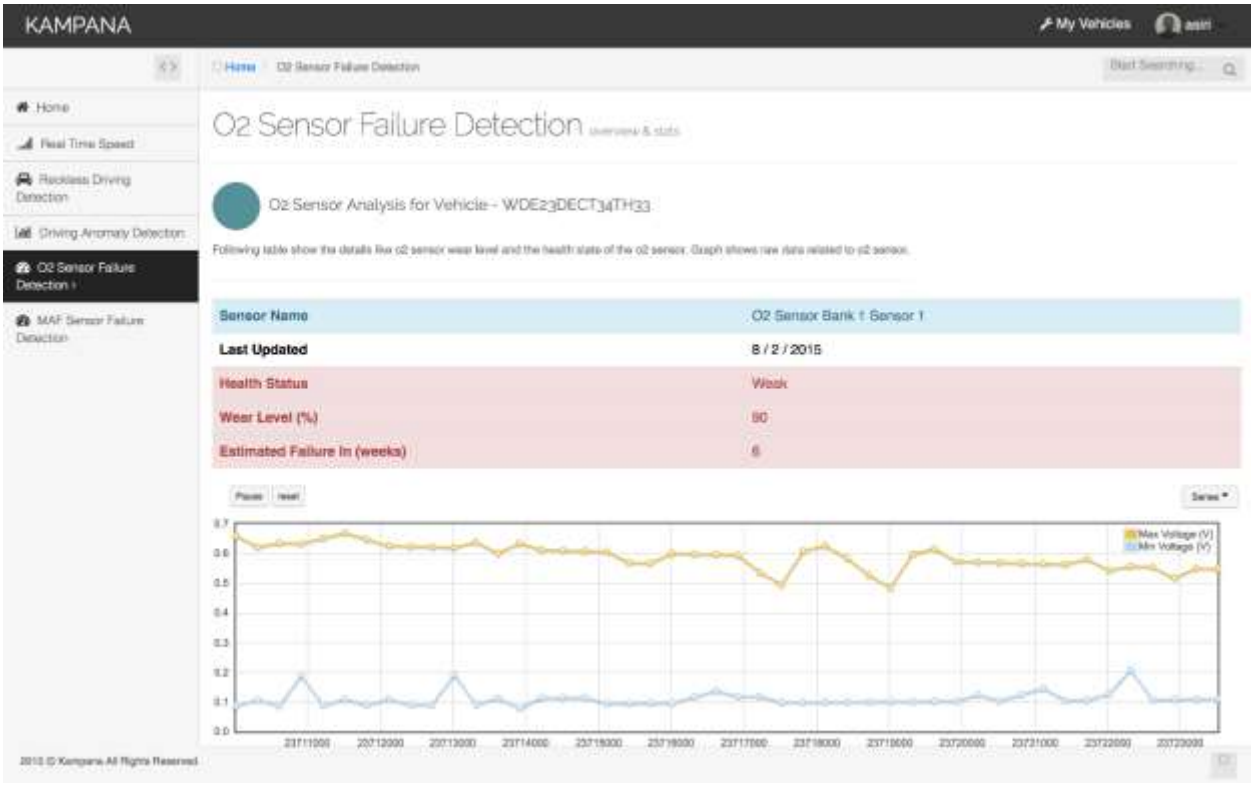

Fig. 4.7. Oxygen sensor web portal

Following table summarizes the predicted failure date and the calculated wear level of the sensor using different time spans. According to the dataset actual failure happens in 16th, Feb 2015. Using the results, it could be observed that the percentage error reduces as the failure date becomes closer.

Table 4.1. Results of prediction for o2 sensor.

Timespan of data	**Predicted Failure Date**	**Prediction Error%**	**Estimated Wear Level**
11/01/2015 – 20/01/2015	03/03/2015	57%	33.7%

20/01/2015 – 25/01/2015	20/02/2015	18%	43.1%
25/01/2015 – 30/01/2015	14/02/2015	11%	51.4%

4.3.2. MAF sensor

Through the BAM analytic scripts, a linear regression analysis is performed to estimate the wear level of the sensor. If it is greater than 50% the user is notified immediately through the Android app. The user can also check the current status of the sensor through the web portal any time.

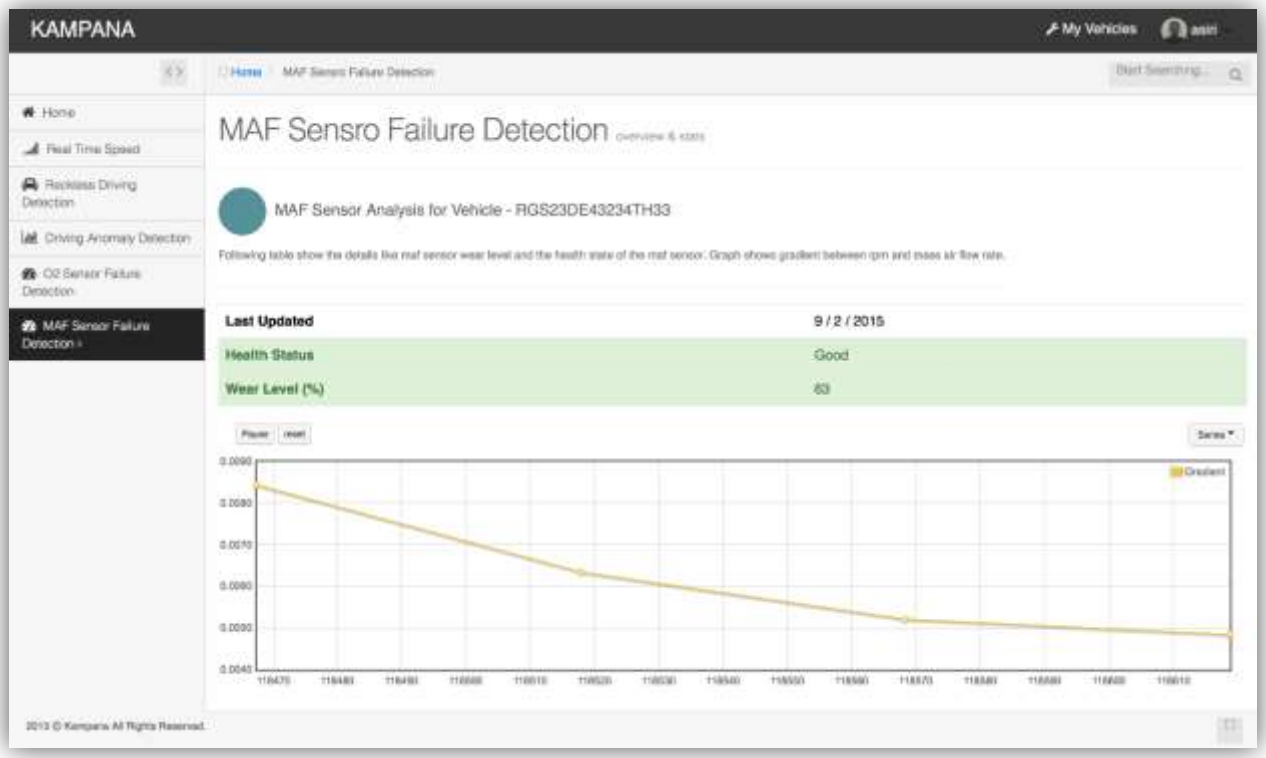

Fig. 4.8. MAF sensor web portal

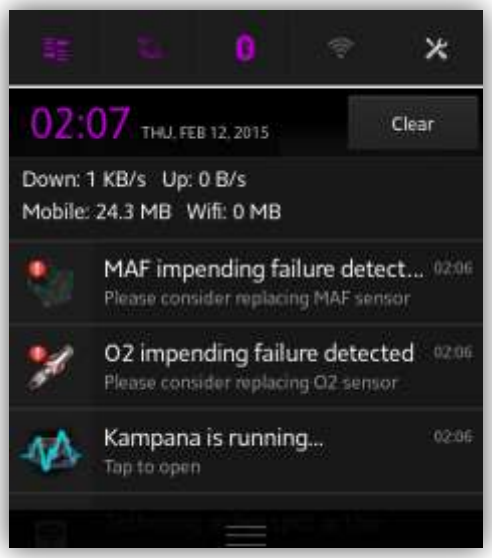

Fig. 4.9. MAF and O2 sensor notifications

4.3.3. Fuel economy and coolant temperature monitoring

Once the app detects an abnormal condition in fuel economy or coolant temperature, it automatically gives a notification to the user.

Fig. 4.10. App notifications for coolant and fuel rate

4.4. Other Features

4.4.1. Trip logs

This is one of the main key feature in Android app. When the user goes to the trip logs section, the user can see the trip logs list which sorted by the latest time. Trip log list directly shows the start place, end place and start time. The following screen shot shows the trip logs list.

Fig. 4.11. Trip log list

User can get more details by touching on the each trip log list item. Then it shows a new window with whole details of the selected trip. The detailed window shows start place name, start time, end place name, end time, trip distance and average fuel efficiency. The following screenshot shows the descriptive window of trip logs.

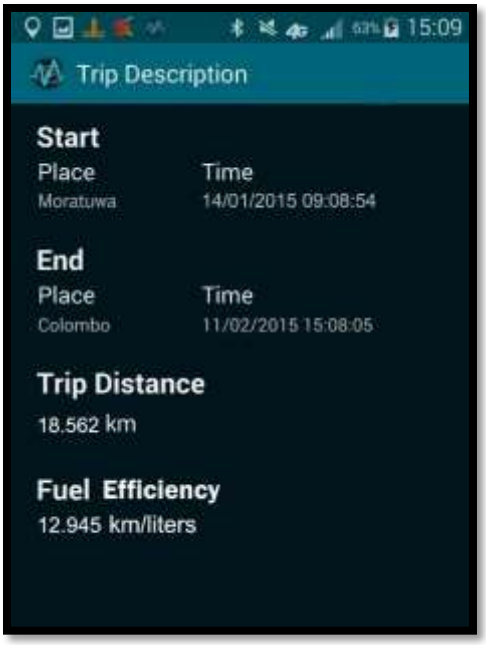

Fig. 4.12. Descriptive view of a single trip log

5. CONCLUSIONS AND FUTURE WORK

We implemented a platform for driver monitoring and vehicle diagnostics which integrates OBD2, Android and backend processing technologies such as Complex Event Processing and Business Activity Monitoring. The system is capable of identifying reckless driving habits and driver anomalous behavior which gives a major advantage for stakeholders such as insurance companies, law enforcement authorities and travel and transportation authorities. The system is also capable of detecting possible sensor failures earlier and alert the driver via the mobile phone which saves the cost of fuel and maintenance due to running the vehicle with a poor performance for a long time.

The Android application which is available as a free download in Google Play Store has currently achieved more than 100 downloads from different countries in just few weeks.

The main drawback of the proposed solution is its dependency on the data communication of the mobile phone to the backend. If the driver has not allowed data transmission on the mobile phone the system will not be useful for backend processing related use cases. The system assumes that a driver possesses a smart phone which is capable of running an Android app. In order to overcome this problem, a dedicated hardware device can be built, where, once the device is plugged into the OBD2 port, data will be uploaded to the remote servers autonomously. The current set of functionalities only consists of a limited number of feature which were identified as crucial. The architecture is built in such a way that new functionalities can be added whenever needed.

6. REFERENCES

[1] Department of Motor Traffic - Sri Lanka. *Statistics* [Online]. Available: http://www.motortraffic.gov.lk/web/index.php?option=com_content&view=article&id=84&Itemid=21&lang=en

[2] M.J. Kim, J. W. Jang and Y. S. Yu, "A Study on In-Vehicle System using OBD-II with Navigation," in *Int. Journal Computer Science and Network Security*, vol.10, no. 9, 136-140, Sept., 2010.

[3] Y. Yang et al., "Research and Development of Hybrid Electric Vehicles CAN-Bus Data Monitor and Diagnostic System through OBD-II and Android-Based Smartphones," in *Advances Mechanical Engineering*, Nov, 2013.

[4] W.D. Huang, Y. Zhang and J. H. Liu, "Research and Design of the Automobile OBD Data State Monitoring System Based on the Android Phones," in *Advance Materials Research*, vols. 605-607(2013), 2301-2305, 2013.

[5] NAPA Institute of Automotive Technology, *"*Introduction to OBDII*,"*, 1998 [Online]. Avaliable: http://www.lbcc.edu/attc/documents/OBD2.pdf

[6] A. Aljaafreh et al., "Vehicular Data Acquisition System for Fleet Management Automation," Electrical Engineering Department, Tafila Technical University, Tafila, 2011.

[7] P. S. Ganapati et al., "Android Based Universal Vehicle Diagnostics and Tracking System," in *International Journal of Modern Engineering & Management Research*, vol. 2(1), 35-41, Mar. 2014.

[8] *OBDII: Past, Present & Future.* [Online] Available: http://www.autotap.com/techlibrary/obdii_past_present_future.asp

[9] *OBD-II Background Information* [Online]. Available: http://www.obdii.com/background.html

[10] *Diagnostic Trouble Code Definitions*, SAE J2012, 2002.

[11] *E/E Diagnostic Test Modes,* SAE J1979, 2002.

[12] US Environmental Protection Agency. *Do you drive a 1996 or newer car or light truck?* [Online]. Available: www.epa.gov/obd/420f02016.pdf

[13] F. Schaffer. 2013.05.12. *List of OBD2 supported vehicles* [Online]. Available: http://www.blafusel.de/obd/obd2_scanned.php

[14] Actron. *PocketScan Code Reader CP9125* [Online]. Available: http://actron.com/product_detail.php?pid=16298

[15] TEXA S.p.A. *OBD Log for finding intermittent faults* [Online]. Available: http://www.texa.com/products/obd-log

[16] *Torque — OBD2 Performance and Diagnostics for your Vehicle* [Online]. Available: http://torque-bhp.com/

[17] *Diagnostic Connector Equivalent to ISO/DIS 15031*, SAE J1962, 2002.

[18] Auterra, LLC. *CAN Bus Equipped Vehicles - Complete Vehicle Make, Model and Year List* [Online]. Available: http://www.auterraweb.com/aboutcan.html

[19] *Vehicle OBD II Compatibility* [Online]. Avaliable: https://plxdevices.com/obd/

[20] *Mojio*, [Online]. Available: https://www.moj.io/

[21] *Bluetooth or WiFi OBDII Connector*, [Online]. Available: http://www.cravenspeed.com/obdii-connector/

[22] Torque. *Bluetooth Adapters* [Online]. Avaliable: https://torque-bhp.com/wiki/Bluetooth_Adapters

[23] ELM Electronics. (2014), *ELM327 OBD to RS232 Interpreter* [Online]. Available: http://elmelectronics.com/DSheets/ELM327DS.pdf

[24] Progressive Casualty Insurance Company, *Snapshot® | Progressive* [Online]. Avaliable: http://www.progressive.com/auto/snapshot/?version=default

[25] *Elm 327 Terminal - Android Apps on Google Play* [Online]. Available: https://play.google.com/store/apps/details?id=Scantech.Terminal&hl=en

[26] *OBD-II (OBD2) Scan Tool Software and Hardware - ProScan* [Online]. Available: http://www.myscantool.com/

[27] Icculus.org. *OBD GPS Logger for Linux and OSX* [Online]. Available: http://icculus.org/obdgpslogger/

[28] Icculus.org. *OBDSim*, [Online]. Available: http://icculus.org/obdgpslogger/obdsim.html

[29] OBDSim Reference Manual, v0.16, icculus.org, 2011.

[30] *What is Bluetooth Technology | Bluetooth Technology Website* [Online]. Available: http://www.Bluetooth.com/Pages/what-is-Bluetooth-technology.aspx

[31] *Bluetooth Basics - learn.sparkfun.com* [Online]. Available: https://learn.sparkfun.com/tutorials/bluetooth-basics/common-versions

[32] *What is 3G? Webopedia* [Online]. Available: http://www.webopedia.com/TERM/3/3G.html

[33] *What is 4G? Webopedia* [Online]. Avaliable: http://www.webopedia.com/TERM/4/4G.html

[34] G. Blewitt, "Basics of the GPS Technique: Observation Equations§", Dept. of Geomatics, Univ. Newcastle, 1997

[35] *What Drains The Smart-Phone Battery? | Europe News content from Electronic Design* [Online]. Available: http://electronicdesign.com/ed-europe/what-drains-smart-phone-battery

[36] *Oracle Complex Event Processing: High Availability*, white paper, Oracle Corporation, Nov, 2010

[37] *Analyze and Act on Fast Moving Data: An Introduction to Complex Event Processing,* white paper, Sybase. Inc., 2010

[38] S. Rizvi, "Complex event processing beyond active databases: Streams and uncertainties," Tech. Report, UCB/EECS-2005-26, Electrical Eng. and Comp. Sci., Univ. of California, Dec. 2005.

[39] G. Cugola and A. Margara, "Low Latency Complex Event Processing on Parallel Hardware," ACM Computing Surveys (CSUR), vol. 44 (3), no. 15, Jun. 2012.

[40] D. C. Luckham, "Event Processing and the Survival of Modern Enterprise,". in *Event Processing for Business*, Hoboken, New Jersey: John Wiley & Sons, Inc., 2012, ch. 1. sec. 3.

[41] N. Leavitt, *Complex-Event Processing Poised for Growth*, in *IEEE Computer Society*, vol. 42, no. 4, 17-20, April, 2009.

[42] *TIBCO* [Online]. Available: http://www.tibco.com/products/event-processing/complex-event-processing/streambase-complex-event-processing

[43] Oracle Corporation. *Overview of Oracle Complex Event Processing* [Online]. Available: http://docs.oracle.com/cd/E13157_01/wlevs/docs30/get_started/overview.html

[44] WSO2 Inc. (2014), *WSO2 Complex Event Processor Documentation - Complex Event Processor 3.1.0 - WSO2 Documentation,* [Online]. Available: https://docs.wso2.com/display/CEP310/WSO2+Complex+Event+Processor+Documentation

[45] S. Suhothayan et al., "Siddhi: A Second Look at Complex Event Processing Architectures," *GCE'11,* Seattle, Washington, USA, Nov. 2011.

[46] S. Sirish et al., "TelegraphCQ: an architectural status report," In *IEEE Data(base) Engineering Bulletin - DEBU* , vol. 26, no. 1, pp. 11-18, 2003.

[47] N. Sirbiladze, "An Evaluation of Business Activity Monitoring Tools," *Int. Conf. Computing, Commun. Syst. and Informatics Manage*, Bur Dubai, UAE, July, 2012, http://ijitcs.com/volume%204_No_1/Rui+Gomes.pdf

[48] WSO2 Inc. (2014), *About BAM* [Online]. Available: https://docs.wso2.com/display/BAM241/About+BAM

[49] B. Y. Liaw, "Fuzzy Logic Based Driving Pattern Recognition for Driving Cycle Analysis," in *Journal of Asian Electric Vehicles*, vol.2, no.1, Jun, 2004.

[50] A. Wahab et al., "Driver Recognition System Using FNN and Statistical Methods," in *Advances for In-Vehicle and Mobile Systems: Challenges for International Standards*, Springer, 2007, ch. 2, sec. X, pp.011-023.

[51] S. Ganesan, *Introduction to On Board Diagnostics (II)*, U.S. Army Vetronics Institute, Dec, 2002. http://groups.engin.umd.umich.edu/vi/w2_workshops/OBD_ganesan_w2.pdf

[52] *Oxygen Sensors: How to Diagnose and Replace* [Online] Available: http://www.aa1car.com/library/o2sensor.htm

[53] CARSDIRECT.COM, Inc., *What a Mass Airflow Meter Does and How to Maximize Your Car's Performance - CarsDirect* [Online]. Available: http://www.carsdirect.com/car-maintenance/what-a-mass-airflow-meter-does-and-how-to-maximize-your-cars-performance

[54] D. Bacon, *Evaluate mathematical expressions in Java* [Online]. Available: https://github.com/darius/expr