

# Cloud-based driver monitoring and vehicle diagnostic with OBD2 telematics

Shashika Muramudalige, Malintha Amarasinghe, Afkham Azeez


2015 IEEE International Conference on Electro/Information Technology (EIT)

## Cite this paper

Downloaded from [Academia.edu](#) 

[Get the citation in MLA, APA, or Chicago styles](#)

## Related papers

[Download a PDF Pack](#) of the best related papers 



[Advanced Automotive Fault Diagnosis, 2nd Ed. - \(Malestrom\)](#)

ernesto perez

[advanced automotive fault diagnosis.pdf](#)

azis zunanto

[Experimental Measurement of the Environmental Impact of a Euro IV Vehicle in its Urban Use](#)

Fernando Ortenzi

# Cloud-Based Driver Monitoring and Vehicle Diagnostic with OBD2 Telematics

Malintha Amarasinghe<sup>#1</sup>, Sasikala Kottegoda<sup>#2</sup>, Asiri Liyana Arachchi<sup>#3</sup>, Shashika Muramudalige<sup>#4</sup>, H. M. N. Dilum Bandara<sup>#5</sup>, and Afkham Azeez<sup>\*6</sup>

<sup>#</sup>*Department of Computer Science and Engineering, University of Moratuwa,  
Katubedda, Sri Lanka, 10400*

<sup>1-5</sup>{malintha.10, sasikala.10, asiri.10, shashika.10, dilumb}@cse.mrt.ac.lk

<sup>\*</sup>*WSO2 Lanka Inc.*

*Colombo 03, Sri Lanka.*

<sup>6</sup>azeez@wso2.com

**Abstract**— We present a cloud-based vehicular data acquisition and analytics system for real-time driver behavior monitoring, trip analysis, and vehicle diagnostics. Our system consists of an On Board Diagnostics (OBD) port to Bluetooth dongle, a mobile app running on a smart phone, and a cloud-based backend. We use a Complex Event Processor (CEP) at both the smart phone and the backend to detect and notify unsafe and anomalous events in real time. For example, CEP engine at the smart phone can alert the driver about rising coolant temperature and rapid fuel drops. It also provides a trip log and filter out what messages to be send to the backend, saving both the bandwidth and power. CEP on the cloud detects reckless driving in real time based on the sensor data provided through the OBD port. Historical data is also used by the backend CEP engine to detect driving anomalies and to predict impeding sensor failures. The mobile app visualizes both real-time data from sensors and alerts. A web-based interface is provided to access the backend information. We tested the system on actual vehicles and demonstrated that the computing, bandwidth, and power consumption of the smart phone is reasonable. App is currently available in Google Play.

**Keywords**— Driver Monitoring, Internet of Things, OBD2, Vehicle Diagnostics

## I. INTRODUCTION

A car is no more a luxurious belonging of a person. It has rather become an integral part of a modern family. 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 [1]. This rapid increase of vehicles has led to many concerns for a range of people and organizations. For example, all parties (i.e., drivers, insurance companies, fleet vehicle managers, and law enforcement authorities) are concerned about reckless driving and driver anomalies. Moreover, people who are willing to purchase and sell cars are also concerned about the condition of the vehicle and its maintenance.

OBD, which stands for On Board Diagnostics could simply be described as a standard which allows accessing the status of sensors attached to a vehicle via a port referred to as the OBD port. Some of the frequently used sensors include speed, engine rpm, coolant temperature, fuel rate and oxygen. OBD2 [2], is the latest version of OBD and is implemented in most of the vehicles which are manufactured lately. Several adapters are commercially available to read data from the OBD2 port. ELM-327, which is used in the proposed system, is one such adapter where the data read from the OBD2 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. The vehicle diagnosis program proposed by Kim et al. [3] provides diagnosis of different kinds of vehicle malfunctions within the navigation system. It displays the data collected through the OBD port in a human readable manner. 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 distracts the driver. A lot of research work has been carried out in the area of vehicle monitoring through a server. While the main focus of most of the researches is on tracking vehicles [4, 5, 6], fault detection has also gained considerable attention [4, 6]. 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 proposed systems, there has been either simple or no processing of the data gathered from the Engine Control Unit (ECU) prior to display. Hence, it is hard to predict any undesired outcomes, such as an accident or failure of a sensor as they require both 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 OBD2 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 on real time data as well on archived data that are collected over 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 backend. The results of these analyses are made visible through two interfaces. The drivers themselves are able to get the results through the mobile app in the form of notifications. Alerts generated both in the app and backend due to undesirable situations are notified to the drivers. Also, results of long term analyses are displayed through a web interface. The web interface enables stakeholders such as vehicle owners, fleet vehicle managers, insurance companies, and authorities to analyze various cases of interest and initiate necessary process changes to enhance service quality, efficiency, cost, and promote responsible driving.

Section II describes the architecture of the proposed system along with the key components. Section III describes driver monitoring features whereas vehicle diagnostic features are described in Section IV. Section V presents the experimental results. Conclusion and future enhancements are presented in Section VI.

## II. SOLUTION ARCHITECTURE

The proposed system is capable of collecting, storing, and analyzing vehicular data for a long period of time. As shown in Fig. 1 the mobile app pulls the vehicular data using an OBD2 to Bluetooth interface. The collected data is then preprocessed at the smartphone to detect interesting events. The preprocessed data is then sent to backend cloud servers using the smartphone’s 3G/4G connection.

Fig. 2 shows the architecture of the proposed solution. The solution 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, sensors, and servers as needed. The mobile app is one of the key elements of the system, as it is responsible for the data transmission between the vehicle and the backend servers while also performing the task of the view layer. The app plays three major roles throughout the process:

1) *Receiving data from OBD2 adaptor:* The app is capable of connecting to the ELM-327 adaptor via Bluetooth and communicating with the vehicle using OBD2 Parameter IDs (PIDs). Each PID provides a certain information about the vehicle, e.g., speed, engine rpm, fuel consumption, and error codes. The received data are logged inside the app and/or displayed in the user interface in real time. The architecture of the app is also extensible where the PIDs can be added dynamically.

2) *Monitoring vehicle:* The mobile app consists of a Complex Event Processor (CEP). 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. Hence, CEP has the capability to detect relevant events in incoming data streams according to a predefined set of queries. For example, queries can be added to alert the driver, if the vehicle is running with a high rate of fuel consumption or high coolant temperature for a considerably long period of time. CEP queries can be also used to summarize trip details in real time, e.g., average fuel consumption for the ongoing trip. We use Android implementation of the WSO2 Siddhi CEP engine as it is lightweight and outperforms many other CEP engines in terms of throughput and latency [7]. Moreover, Siddhi supports adding queries dynamically hence provides extensibility to support future monitoring tasks as well. More specifically, OBD2 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 CEP 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. Alerts are generated when unusual behaviors are detected in the incoming data (i.e., data matching a given CEP query is found)



Fig. 1. Overview of the proposed system.

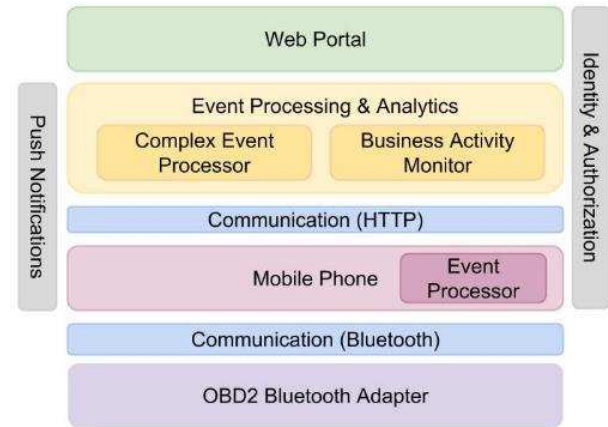


Fig. 2. Architecture of the proposed system.

and the results are shown/notified to the driver via the mobile app.

3) *Selective transmission of data to backend:* A user has to pay for each megabyte that is consumed over the 3G/4G network. Therefore, the network bandwidth usage of the app is an important fact to consider. Hence, the CEP queries are used in the mobile app to filter out uninteresting events and only the useful data are transmitted over the 3G/4G network. CEP queries are also used to collect and send only the aggregated data for certain types of sensor readings. Such reduction in data not only reduces the bandwidth requirement, but also minimizes power consumption of the smartphone. Data are forwarded to the backend using HTTP messages.

Complex event processing is also utilized at the backend. Backend is also based on the Siddhi CEP engine; however, with a lot more computing capabilities and with the freedom to work with a large database. The CEP at the backend is responsible for detecting more complex driver and engine anomalies (e.g., reckless driving) and generating alerts in real time. Some of these alerts are pushed back (as push notifications) to the mobile app and notified to the driver. For example, warn the driver for reckless behavior and speed alters based on known GIS (Geographic Information System) information.

The system is capable of performing long term analyses too. By identifying patterns, it should be able to predict undesirable outcomes such as potential failures of sensors. This is enabled by using a Business Activity Monitor as the long term analyzer at the backend. BAM is a solution primarily intended to provide a real-time summary of business activities and is capable of collecting, storing, and analyzing data. In our implementation BAM receives events published by the mobile

app and stores them in a NoSQL database where analyses are performed regularly to identify gradual changes in data. We use WSO<sub>2</sub> BAM [8] due to its performance and better integration with the Siddhi CEP engine. WSO<sub>2</sub> BAM supports distributed processing, integrated with Apache Cassandra, a highly-scalable NoSQL database. This enables the distributed processing of datasets of a large number of vehicles and their owners across clusters of computers. While some of the results of analyses are notified to the user in the form of alerts (e.g., popups, e-mail, and SMS), all the results of the analyses performed in the backend are displayed on the web portal. This way, drivers themselves could monitor their driving behaviors and changes in the vehicle condition. Similarly, organizations could monitor their customers' behaviors and conditions of the vehicles that are owned by them. To support multiple users, roles, and devices the propose system also provide necessary identity and authorization services at the backend (see Fig. 2).

### III. DRIVER MONITORING

Driver monitoring includes two aspects. First is reckless driver monitoring, which is mostly useful to vehicle owners and organizations such as insurance companies and law enforcement authorities. The recklessness of the driver's driving pattern can be measured within a certain period of time such as 20 hours, one week, one month, and three months. Second is driver anomaly detection. While anomaly detection is important for the above organizations, it is more important to the drivers themselves to get alerted when they deviate from their usual driving pattern due to stress, drunk, distractions, etc. The two use cases are described in the following subsections.

#### A. Reckless Driving

Reckless driving causes a serious danger to the driver as well as general public. If a reckless driving detection methodology can be implemented, it will be beneficial for other drivers, vehicle owners, general public, insurance companies, and many other stakeholders who would not want to risk time and money for the consequences of reckless driving.

Bhoyar et al. [9] proposed a system for reckless driving detection, which is a mobile phone based rash driving detection system. There, reckless driving behavior was detected using the lateral acceleration and longitudinal acceleration. Many of the literature demonstrated that the sudden variation of the longitudinal acceleration is a good metric to detect reckless driving. Therefore, we also used longitudinal acceleration to detect reckless driving. The proposed solution for detecting reckless driving include following two steps:

1) *Preprocessing within app*: The speed of the vehicle can be read in real time from the OBD2 adapter. Because it is the acceleration/deceleration that is of concern, the Siddhi CEP engine transforms the speed streams into acceleration/deceleration streams by considering consecutive speed readings using the following Siddhi query:

```
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 predefined threshold to detect whether it is reckless or not. We use a threshold of 4.5 ms<sup>-2</sup>, as per the recommendation by the

American Association of State Highway and Transportation Officials [10]. Binary values of 1 are assigned to the acceleration/deceleration values above the threshold and binary 0 is assigned for values below the threshold. The total number of 1's is then counted over a predefined time interval (e.g., 2 minutes). The counts are then classified according to the driving cycle (i.e., traffic, normal, and highway) of the trip. A *driving cycle* is a series of data points representing the speed of a vehicle versus time. Driving cycle is determined by the average speed of the vehicle throughout 10 minutes. The count, together with the detected driving cycle are sent to the backend server periodically. Acceleration and deceleration values are sent as two separate streams.

#### 2) *Processing at Backend*

At the backend, data received from the app are summarized by the WSO<sub>2</sub> BAM as hourly, weekly, monthly, and three months aggregates and stored in an SQL database ready to be read by the web portal. Moreover, when the aggregate for a given time window is above a predefined threshold, alerts can be pushed to both the drivers and other stakeholders.

#### B. Detection of Driving Anomalies

Identifying driver patterns have become a principal research area due to the high concern about fuel efficiency and safe driving. Liaw [11] proposed a solution to the problem of identifying driving patterns using fuzzy logic. 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. Wahab et al. [12] suggests that the driving pattern can uniquely define a driver, thus enabling us to use it as a biometric. There, it is explained that, what is unique to each driver is the pressure distribution with time on the accelerator and brake pedals. Also, it was found that it is not the speed pattern, but the acceleration pattern that is unique to each driver.

We use a hidden Markov model to determine the difference between the current driving pattern and the past patterns of a (driver, vehicle) combination. Identifying driving anomalies happens in the following three stages:

1) *Preprocessing*: As in the use case of reckless driving detection, it is the acceleration that is of concern. Siddhi CEP engine transforms the speed streams into acceleration streams. These acceleration streams are then converted to acceleration transition streams which consist of the timestamp, previous acceleration, and the current acceleration. These acceleration streams are transmitted to the backend. These streams are used by the CEP in real time for calculating whether the readings are in accordance with the normal driving pattern. The same stream is also used by BAM to update the model daily.

2) *Model Creation*: The Markov Model is shown in Fig. 3 is stored in an SQL database in the form of a transition table of accelerations. The BAM processes the events (i.e., acceleration transition streams) received from the app, and then updates its transition table consisting of daily transition probabilities. Therefore, the more acceleration readings the BAM receive, the more accurate the model becomes.

3) *Determining Anomalies*: This is a task that happens almost in real time. An anomaly is detected by calculating the probabilities of the Markov Chain resulted by the recent accelerations. There is a need to multiply each probability to determine the probability of a chain. However, a single

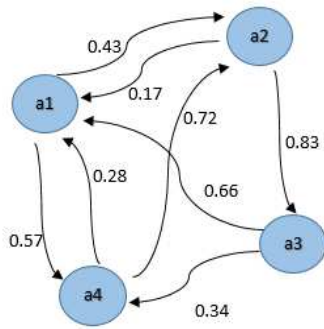


Fig. 3. Acceleration transition diagram.

transition that has zero probability during the measured time interval results in a zero probability of the entire chain. Therefore, to avoid this problem we simplified the calculation by considering the property emphasized by Eqn. 1.

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

Whenever the CEP engine receives an 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 averaged probability of each chain of accelerations is calculated using the property highlighted in Eqn. 1. Once averaged probability of each chain drops below a predefined threshold, an alert is sent to the driver and other stakeholders.

#### IV. VEHICLE DIAGNOSTIC

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 (MAF) sensor and Oxygen (O<sub>2</sub>) sensor are two of the most important sensors that crucially determine the engine performance, emissions, and other important functions. OBD2 system usually takes some time to identify malfunctions of these two sensors and indicate that on the dashboard. However, in the meantime, lots of fuel wastage can result. Sensors do not fail suddenly, but gradually. Therefore, we came up with a solution to detect impending failures so that they can be identified early. Next, we discuss how our solution predicts O<sub>2</sub> and MAF sensor failures. We also briefly describe how engine coolant temperature, engine oil temperature, fuel economy, and battery voltage monitoring are performed.

##### A. Oxygen Sensor Failure

O<sub>2</sub> sensor measures the amount of Oxygen left in the exhaust, which in turn is used to balance the air/fuel ratio. A typical oxygen sensor's operation range should be between 0.1 V and 0.9 V in lean and rich conditions of the vehicle, respectively. It will deviate when the sensor is failing. When any of the bounds reaches 0.45 V sensor is considered as failed [13].

In our solution, the maximum and minimum voltages are read by the app for 15 minutes at the start of each trip and transmitted to the BAM by which the data are stored in the database. BAM performs two regression analyses periodically for both minimum and maximum values w.r.t the time as shown in Fig. 4. Once regression is complete, the system predicts the potential date of sensor failure and the driver will be warned, if the predicted date is closer to the current date.

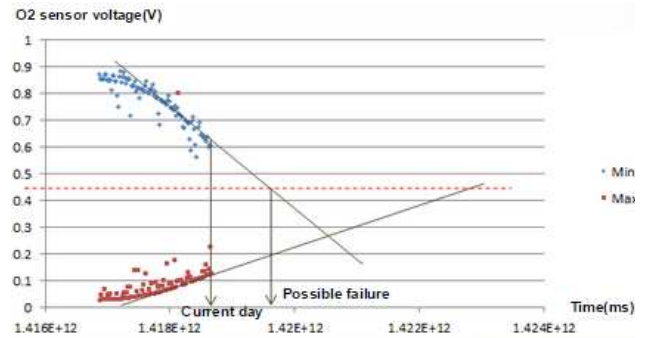
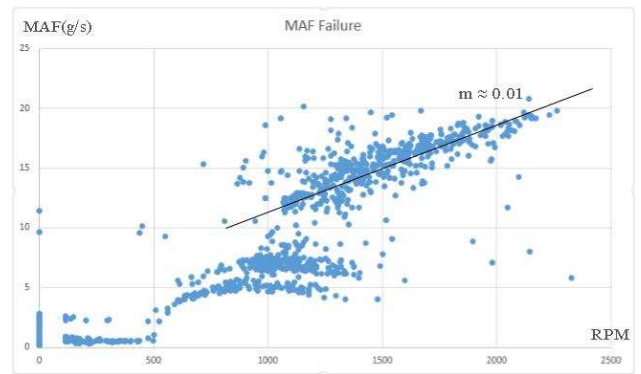
Fig. 4. Regression of max and min values of the O<sub>2</sub> sensor.

Fig. 5. Flow rate vs. rpm in higher rpms for a normal sensor.

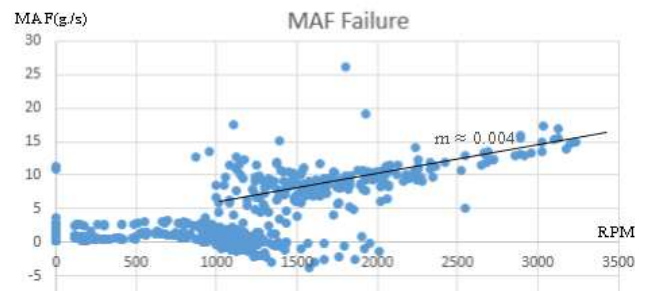


Fig. 6. Flow rate vs. rpm for a faulty sensor.

##### B. Mass Air Flow Sensor Failure

MAF is an air flow sensor, which measures the mass flow rate of air entering the engine which in turn is used to calculate fuel delivery and spark timing. OBD2 system gives a MAF value in grams per second (gs<sup>-1</sup>) depending on the engine model and the capacity; hence, MAF value cannot be directly used to detect a failure. As illustrated in Fig. 5, higher MAF values have a linear relationship with the corresponding engine rpm values. When the MAF sensor is failing, the gradient will be gradually reduced as shown in Fig. 6.

A regression analysis is done periodically and corresponding gradients are stored in the database. Another regression analysis is done with a lesser frequency for the gradient values. The prediction is given by calculating the time it takes for the gradient to acquire a certain threshold. When the expected time to fail is lesser than one month, an alert is generated. As the time gets closer and closer, alerts are generated at a higher frequency.

##### C. Engine Coolant Temperature Monitoring

Coolant temperature, is queried from the OBD2 adapter periodically and is forwarded to the CEP engine on the mobile app. Received temperature readings are averaged over

2 minutes using a time window. If the averaged value is greater than the given temperature threshold (we used 104°C), the driver is alerted immediately about the possible overheating.

**D. Engine Oil Temperature Monitoring**

Engine oil temperature can be retrieved using OBD2 Mode 01 PID 5C. If the oil temperature is too low while the engine rpm is high, the produced water and Sulfur as by-products of the combustion process can form acids and damage the engine bearings. The app is also capable of generating alerts, if the engine oil temperature is not within the desired operating range when the average engine rpm is greater than a certain threshold. As the operating range and the rpm threshold differ from vehicle to vehicle it is configurable within the app along with the time window for the average rpm.

**E. Fuel Economy Monitoring**

The fuel rate, along with the speed can be used to calculate the fuel economy. Using two separate time windows for the fuel rate and speed, the average fuel rate and the average speed are calculated for a certain time duration. Then the average fuel economy can be calculated using Eqn. 2.

$$Fuel\ Economy = (Avg.\ fuel\ consumption)/(Avg.\ speed) \quad (2)$$

If the average fuel economy becomes lower than a certain threshold, an alert is generated by the mobile app.

**F. Vehicle Battery Voltage Monitoring**

Battery voltage can be retrieved from the OBD2 adapter using the AT command “AT RV” [2]. If this goes below a certain threshold, the driver is alerted that there is a possible battery charging failure.

**V. EVALUATION**

The main deliverables of the project are the mobile app for Android devices and the web interface which displays monitored data and analyzed results. The app was built in such a way that it would display monitored data in real time using dashboards as shown in Fig. 7. The app also shows notifications (alerts).

In the use case of reckless driving detection, speed classes were categorized as 20 kmph or below (Class A), 20 - 80 kmph (Class B), 80+ kmph (Class C). Once the summarization of data happens in the backend, it is able to display results as seen in Fig. 8. Rapid acceleration counts (blue bars in the figure) and rapid deceleration counts (yellow bars) were displayed in the same graph. A user can get an idea of the recklessness of a driver just by seeing that the height of the bars being great. Especially, if the deceleration count is too high, it indicates a higher recklessness due to rapid braking.

Driving anomaly test was carried out using two users where a lot of data could be collected from one of them (*user 1*). The model was created for *user 1* using half of the data collected. Then data from both the users were used in real time to validate the current user. First, data from the other user (*user 2*) was validated against the model and then data from *user 1* was sent to the CEP to be validated. The two datasets resulted in a significant difference of average transition probabilities. The resultant probabilities as well as the given speed patterns as the input are shown in Fig. 9 and Fig. 10, respectively.

The graph shown in Fig. 9 can be divided into two regions: one, where the probability is very low, and the other where the probability is relatively high. The threshold which separates

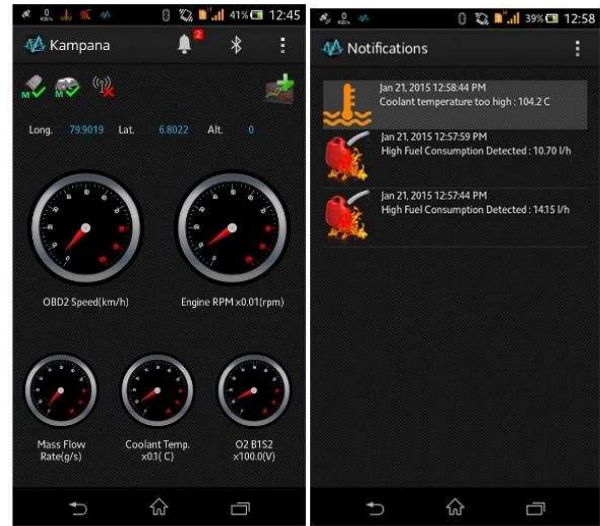


Fig. 7. Dashboards of the Android app.

**Class B**

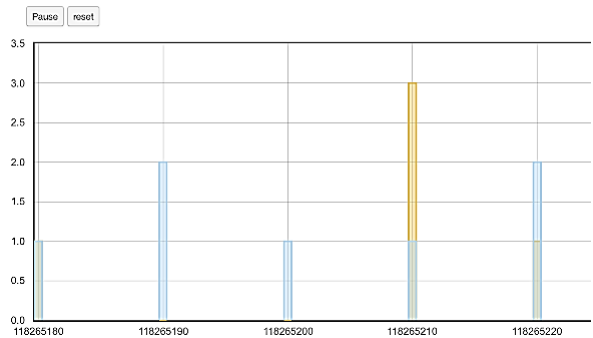


Fig. 8. Rapid accelerations/decelerations graph.

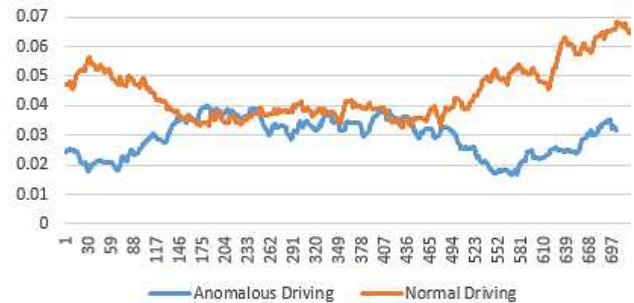


Fig. 9. Driving anomaly results based on probabilities.

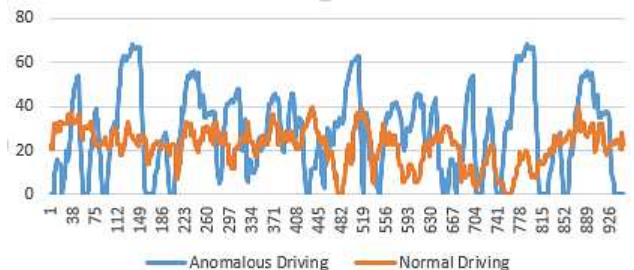


Fig. 10. Speed patterns used as inputs to driving anomaly detection.

the two regions, was used to generate the alert. It can be seen that the probability transition patterns of the two drivers are significantly different from each other. But with time, *user 2*'s pattern becomes a bit smoother and the pattern of *user 1* becomes a little wavy. Therefore, the acceleration transition probability for the anomalous driver get a little high.

TABLE I  
OXYGEN SENSOR ANALYSIS RESULTS

Timespan of Data	Predicted Failure Date	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%

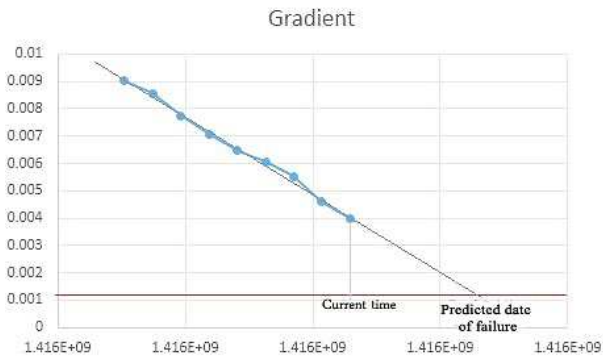


Fig. 11. Regression of the gradient of the line between MAF rate and rpm..

Alternatively, the acceleration transition probability of *user I* gets a little low. This way a little change in the driving pattern results in a big difference in the acceleration transition probability.

Long term (ranging from weeks to years) and known sensor failure data are required to demonstrate vehicle diagnostics. As such data were not available, we generated synthetic data to indicate sensor failures to test the detection and prediction of sensor failures. A set of fabricated data were generated that mimics failure of the Oxygen sensor in 3 months. We first collected a real dataset that consists of minimum and maximum voltage values with time. The dataset was then converted into a dataset as shown in Fig. 4. Table I summarizes the predicted failure date and the calculated wear level of the sensor using different time spans. According to the dataset actual failure happens on Feb 16, 2015. It could be observed that the percentage error reduces as the failure date becomes closer. Vehicle owner can see the current sensor status using the web portal.

Similar synthetic dataset was created to demonstrate MAF sensor failure. Data was fabricated in a way that the gradient of the regression line between the MAF value and the rpm is reduced with time. As Fig. 11 indicates, the user could view the web interface to see how long it will take for the sensor to fail.

## VI. SUMMARY AND FUTURE WORK

We propose a driver monitoring and vehicle diagnostic system using the telematics provided by OBD2 port available in most of the contemporary vehicles. The proposed solution is able to provide real-time alerts such as rising coolant temperature and rapid fuel drops at the vehicle using a CEP engine implemented on a mobile app. Moreover, the app filters out uninteresting events that are forwarded to the cloud-based

backend conserving both the bandwidth and energy. Furthermore, the app visualizes real-time readings from vehicular sensors and notifications pushed from the cloud backend. Given the data published to the backend, CEP on the backend detects reckless and anomalous driving in real time. Backend also use historical data to detect driving anomalies and predict impending sensor failures. App is currently available on Google Play and can be used without the backend features.

The main drawback of the proposed solution is its complete dependency on the data communication of the smartphone. If the driver does not allow data transmission via the smartphone the system will not be useful. The proposed solution also assumes that a driver possesses a smartphone capable of running an Android app. A black box in the form of a dedicated hardware device can be built to overcome this problem where, once the device is plugged into the OBD2 port, data will be uploaded to the remote servers autonomously. We are already building a CEP engine for Arduino-based embedded systems, which we believe will be useful in building such a black box with the ability to do data filtering at the vehicle. The current feature set only consists of a limited number of features which were identified as crucial. The architecture is built in such a way that new functionalities can be added whenever needed. Another drawback of the system is that traffic conditions are not considered when training the Markov model, which can sometimes lead to wrong interpretation of driver behavior. In future we plan to strengthen the accuracy of event detection with fine-tuned algorithms and more real-world data.

## REFERENCES

- [1] "2012 Production Statistics," International Organization of Motor Vehicle Manufacturers, 2012. [Online]. Available: <http://www.oica.net/category/production-statistics/2012-statistics/>.
- [2] ELM Electronics, "On Board Diagnostics (OBD) ICs," 2015. [Online]. Available: <http://elmelectronics.com/DSheets/ELM327DS.pdf>.
- [3] M. J. Kim, J. W. Jang, and Y. S. Yu, "A study on in-vehicle system using OBD-II with navigation," *Int. Journal Computer Science and Network Security*, vol. 10, no. 9, Sept., 2010, pp. 136-140.
- [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," *Advance Materials Research*, vol. 605-607, 2013.
- [5] A. Aljaafreh et al., "Vehicular data acquisition system for fleet management automation," in *Proc. IEEE Intl. Conf. on Vehicular Electronics and Safety (ICVES)*, July 2011, pp.130-133.
- [6] P. S. Ganapati et al., "Android based universal vehicle diagnostics and tracking system," *Intl. Journal of Modern Engineering & Management Research*, vol. 2, no. 1, Mar. 2014, pp. 35-41.
- [7] S. Suhothayan et al., "Siddhi: A second look at complex event processing architectures," in *Proc. ACM workshop on Gateway Computing Environments (GCE '11)*, Nov. 2011, pp. 43-50.
- [8] WSO2 Inc., About BAM [Online]. Available: <https://docs.wso2.com/display/BAM241/About+BAM>
- [9] V. Bhojar, P. Lata, J. Katkar, A. Patil, and D. Javale, "Symbian based rash driving detection system," *Intl. Journal of Emerging Trends & Technology in Computer Science*, vol. 2, no. 2, 2013.
- [10] American Association of State Highway and Transportation Officials, in *A Policy on Geometric Design of Highways and Streets, AASHTO*, 2011, ch.3, pp. 3.
- [11] B. Y. Liaw, "Fuzzy logic based driving pattern recognition for driving cycle analysis," *Journal of Asian Electric Vehicles*, vol.2, no.1, Jun, 2004.
- [12] A. Wahab et al., "Driver recognition system using FNN and statistical methods," *Advances for In-Vehicle and Mobile Systems: Challenges for International Standards*, Springer, 2007, pp.011-023.
- [13] "Oxygen sensors: How to diagnose and replace." [Online]. Available: <http://www.aalcar.com/library/o2sensor.htm>