# Providing Accident Detection in Vehicular Networks Through OBD-II Devices and Android-based Smartphones

Jorge Zaldivar, Carlos T. Calafate, Juan Carlos Cano, Pietro Manzoni
Department of Computer Engineering
Universitat Politècnica de València
Camino de Vera S/N, 46022, Spain
koke.zaldivar@gmail.com, {calafate, jucano, pmanzoni}@disca.upv.es

***Abstract*—The increasing activity in the Intelligent Transportation Systems (ITS) area faces a strong limitation: the slow pace at which the automotive industry is making cars "smarter". On the contrary, the smartphone industry is advancing quickly. Existing smartphones are endowed with multiple wireless interfaces and high computational power, being able to perform a wide variety of tasks. By combining smartphones with existing vehicles through an appropriate interface we are able to move closer to the smart vehicle paradigm, offering the user new functionalities and services when driving. In this paper we propose an Android-based application that monitors the vehicle through an On Board Diagnostics (OBD-II) interface, being able to detect accidents. Our proposed application estimates the G force experienced by the passengers in case of a frontal collision, which is used together with airbag triggers to detect accidents. The application reacts to positive detection by sending details about the accident through either e-mail or SMS to pre-defined destinations, immediately followed by an automatic phone call to the emergency services. Experimental results using a real vehicle show that the application is able to react to accident events in less than 3 seconds, a very low time, validating the feasibility of smartphone based solutions for improving safety on the road.**

***Keywords*-OBD-II; Android smartphone; testbed; performance analysis; vehicular security.**

## I. INTRODUCTION

The widespread adoption of mobile telephony has remarkably improved interpersonal communications in our society. Nowadays mobile phones resemble small multifunction computers, being characterized by a CPU power and RAM size similar to that of laptop computers available only a few years ago. The future trend is that more and more users own these intelligent mobile terminals, and that their main use gradually shifts towards functionalities including web surfing, social networking, multimedia streaming, online games, domotic applications, and so on. Under these premises, it is possible to introduce novel services using smartphones in many different contexts.

At the same time, the automobile industry is undergoing a major strategical shift towards more ecological and secure vehicles, introducing also new vehicular services such as eco-driving support and Internet access. However, it will take many years until most of the vehicles in our streets are equipped with these novel functionalities. In the particular case of supporting safety and emergency services, any anticipation is clearly desirable since it may help at saving lives.

Currently, several European projects within the 7th framework program, like for example ASSET-ROAD [1], CyberCars2 [2] and SAFETRIP [3], address safety on the road through vehicular networking. In the USA and in Asia we can find similar research activities and ITS projects, and several vehicle manufacturers are already working in this direction.

In this paper we propose combining existing vehicles with smartphones to achieve a solution able to improve security on the road. In our solution, smartphones are used as an alternative On-Board-Unit (OBU) within the vehicle, accessing the information in the vehicle's internal bus wirelessly. The only requirement to achieve this goal is that the vehicle supports the OBD-II standard [4]. Since this standard is mandatory since 2001, the solution is applicable to all vehicles aged 10 years or less (as of 2011). In this work, a specialized smartphone application was developed to provide support for emergency services based on the information available in the communications bus of the vehicle. In particular, the proposed application monitors the vehicle's speed and airbag triggers to detect when an accident has occurred. Positive accident detection is followed by any sequence of actions defined by the user, such as sending accident details via SMS or e-mail, or making an automated phone call to the emergency services.

Through different experiments made in a vehicle testbed we determine the most adequate techniques to detect an accident. In particular, we compare the accuracy of the acceleration estimation using either: (i) GPS information, (ii) accelerometer information, and (iii) vehicular speed provided by an OBD-II interface. We also determine the total time required to detect an accident, prepare the warning message to be delivered, and the actual delivery of such message through different communication channels. Results have shown that the application developed is able to correctly fulfill its purpose within a short time period, opening new research opportunities for the integration of smartphones and vehicular networks.

This paper is organized as follows: in the next section we refer to some related works in this field. In section III we briefly introduce the OBD-II standard since it plays a major role in the proposed application. Our proposal for vehicular security enhancement through smartphones is presented and discussed in section IV. Section V presents some performance evaluation results in a vehicular testbed. Finally, section VI concludes the paper.

## II. Related Work

In the literature we can find some works that adopt Android based smartphones to support all sorts of in-vehicle services.

Hernandez et al. [5] developed a prototype of an on-board unit that allows the driver to communicate with his vehicle, as well as with other available devices (PDAs, cellular, sensor networks, and so on) and with the road infrastructure in order to consume intelligent transport services. In their work, they proposed two new services based on vehicle-to-infrastructure communications: a) real-time traffic reports, and b) eco-driving support.

Chen et al. [6] proposed a vehicular Android/OSGi platform that allows diagnosing or managing the system status of a vehicular platform remotely, and also to use visual intelligence to continually update their application services based on context-awareness without user intervention. Experiments made in a vehicular testbed showed that the proposed Android/OSGi platform has lighter applications and higher performance than a pure Android platform when performing complicated operations.

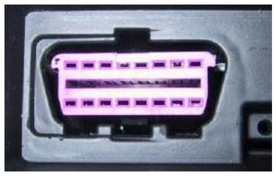
Al-Ani et al. [7] proposed using an Android-based terminal as a replacement to the vehicle's stereo system to provide a High Fidelity In-Vehicle Infotainment System (IVI). Such system provides informative and interactive material that will extend the driver experience by providing capabilities from a mobile-based operating system.

Cheng et al. [8] proposed an Android-based mobile device platform which provides network management functionalities to guarantee the communication quality. In order to achieve ubiquitous computing, the proposed algorithm supports seamless handover via effective resource and handover management between heterogeneous networks.

Spelta et al. [9] implemented a system for a vehicle-to-driver and vehicle-to-environment communication, based on a smartphone core and Bluetooth communication. Since authors focus on motorcycles, they equip them with an embedded CAN-Bluetooth converter that is interfaced with the smartphone, which acts as a gateway toward an audio helmet and a web server.

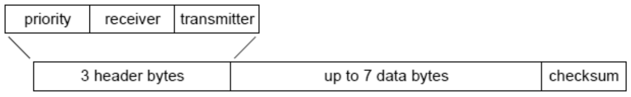
When focusing specifically on the support for accidents and emergency services using mobile phones, very few works are available. Thompson et al. [10] describe solutions to key issues associated with detecting traffic accidents, and detail how smartphone-based accident detection can reduce overall traffic congestion and improve the response time of emergency services. They also develop a smartphone-based accident detection system, and empirically analyze its ability to detect false positives, as well as its capabilities for accident reconstruction. However, they do not take advantage of any information from the vehicle's internal bus.

In this work we develop an application that communicates with the vehicle's internal bus using the OBD-II interface and Bluetooth communication to determine whether accidents have occurred, and to estimate the severity of such accidents. The information retrieved is then submitted to emergency services, and an emergency phone call is automatically established.

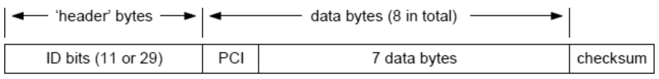

(a) Female connector

(b) Male connector.

Figure 1: Example of a) an in-vehicle OBD-II female connector, and b) a Bluetooth-enabled OBD-II device with male connector.



(a) Frame format adopted by the SAE J1850, ISO 9141-2 and ISO 14230-4 standards.



(b) Frame format adopted by the ISO 15765-4 (CAN) standard.

Figure 2: OBD frame formats.

## III. The OBD-II standard

The On-board Diagnostic (OBD) standards [4] were developed in the USA to detect car engine problems that can provoke an increase of the gas emission levels beyond acceptable limits. To achieve this purpose, the system is constantly monitoring the different elements related to gas emissions, including engine management functions, being a powerful tool to diagnose problems on vehicles' electrical systems. When a failure is detected, the system must store it in memory so that technicians may analyze it later on.

The first OBD standard, known as OBD-I, defined only a few parameters to monitor, and did not establish a specific emission level for vehicles. Thus, failures resulted in just a visual warning to the driver and the storage of the error. The second generation of OBD, known as OBD-II, standardizes

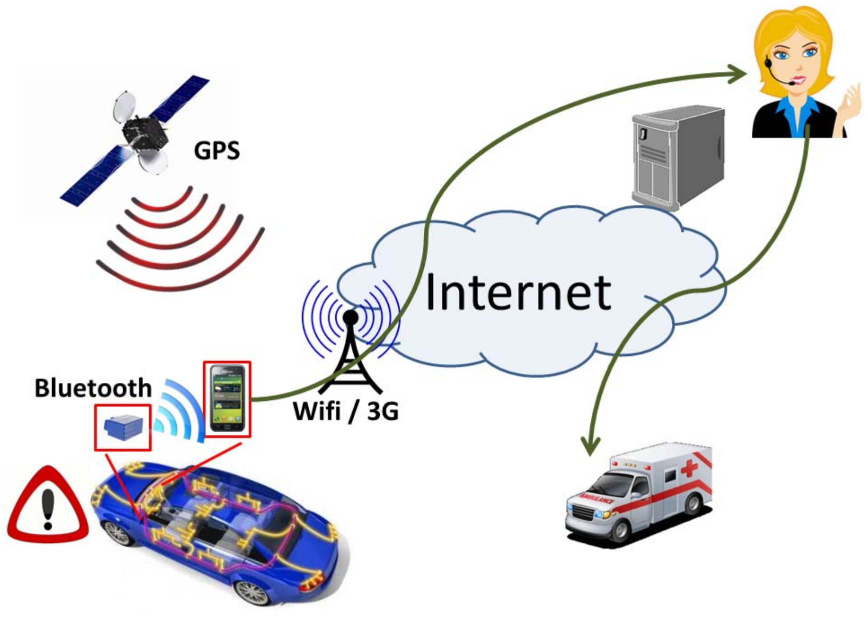


Figure 3: Illustration of the different interacting elements in the scope of the proposed application.

different elements such as the connector used for diagnostic, the electrical signaling protocols, and the message format. Additionally, it defines a list of parameters that can be monitored, assigning a code to each parameter. A detailed list of DTCs (Diagnostic Trouble Codes) is also defined in the standard.

Several operating modes are defined by the OBD-II standard to allow for an easier interaction with the system, and defining the desired functionality. Most automobile manufacturers have introduced additional operation modes that are specific to their vehicles, thus offering a full control of the available functionality.

The European version of the OBD-II standard, known as EOBD, is mandatory for all gasoline and diesel vehicles since 2001 and 2003, respectively. Despite it introduces small improvements, EOBD strongly resembles OBD-II, sharing the same connectors and interfaces.

Figure 1 shows an example of both male and female OBD-II connectors. In particular, the male connector shown in the figure is part of a Bluetooth-enabled OBD-II device that offers a bridge between the vehicle's internal bus and smartphone using a Bluetooth connection.

### A. Communication protocols

Although the physical interface is well defined, the communications protocol varies depending on the manufacturer. The available protocols are: (i) SAE J1850 PWM (Pulse-Width Modulation) [11], (ii) SAE J1850 VPW (Variable Pulse Width) [11], (iii) ISO 9141-2 [12], (iv) ISO 14230 KWP2000 (Keyword Protocol 2000) [13], and (v) ISO 15765 CAN [4]; these protocols present significant differences between them in terms of the electrical pin assignments. Notice that most vehicles implement only one of these protocols. For instance, Chrysler uses the ISO 9141-2 protocol, General Motors uses SAE J1850 VPW, and Ford uses SAE J1850 PWM.

### B. Diagnostic Trouble Codes (DTCs)

Diagnostic Trouble Codes were standardized in document ISO 15031-6 [14], and allows engine technicians to easily determine why a vehicle is malfunctioning using generic scanners. The proposed format assigns alphanumeric codes to the different causes of failure, although extensions to the standard are allowed to support manufacturer-specific failures.

### C. OBD message formats

The OBD system was designed to offer a flexible communications system. Message delivery among different devices requires defining the type of message to be delivered, along with the transmitter and the receiver devices. The adoption of different message priorities is also supported in order to make sure that critical information is processed first.

However, depending on the protocol used, the format of this message may vary slightly (see figure 2). Notice that both frame formats allow up to 7 data bytes, and they include a checksum field in order to detect any transmission errors.

## IV. PROPOSED VEHICULAR MONITORING AND SAFETY APPLICATION

The proposed application basically combines two elements - vehicle and smartphone - to achieve a symbiosis between both that is able to improve the effectiveness of emergency services by making accident detection fully automatic. Accident detection is based on the parameters provided by the OBD-II interface, such as airbag triggering detection, and is complemented with the information gathered by the mobile phone itself, such as GPS information. This is illustrated

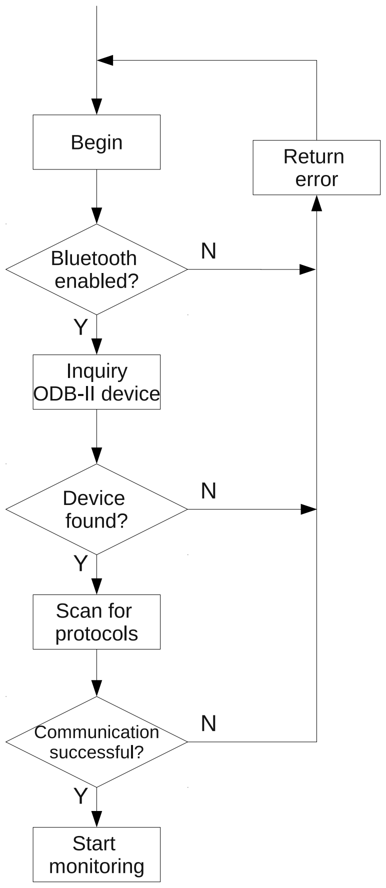


Figure 4: Flow diagram for the initial application procedures.

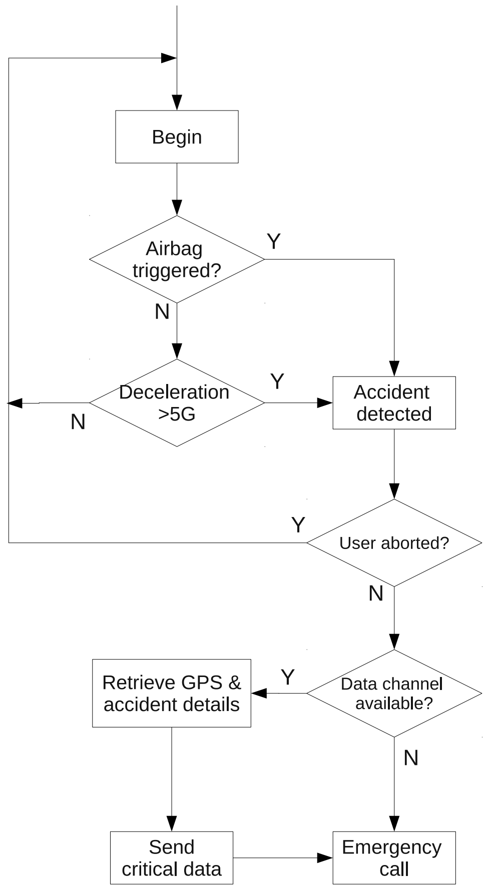


Figure 5: Flow diagram for the accident detection strategy.

in figure 3. Initially the smartphone connects to an OBD-II device via Bluetooth to retrieve data from the vehicle's bus. The information gained, together with data from other sources (e.g. GPS system) is packed and sent to an emergency services database or to other third parties defined by the user if an accident is detected. This procedure is followed by an automatic call to an operator, which will send an ambulance or other rescue services to the accident location.

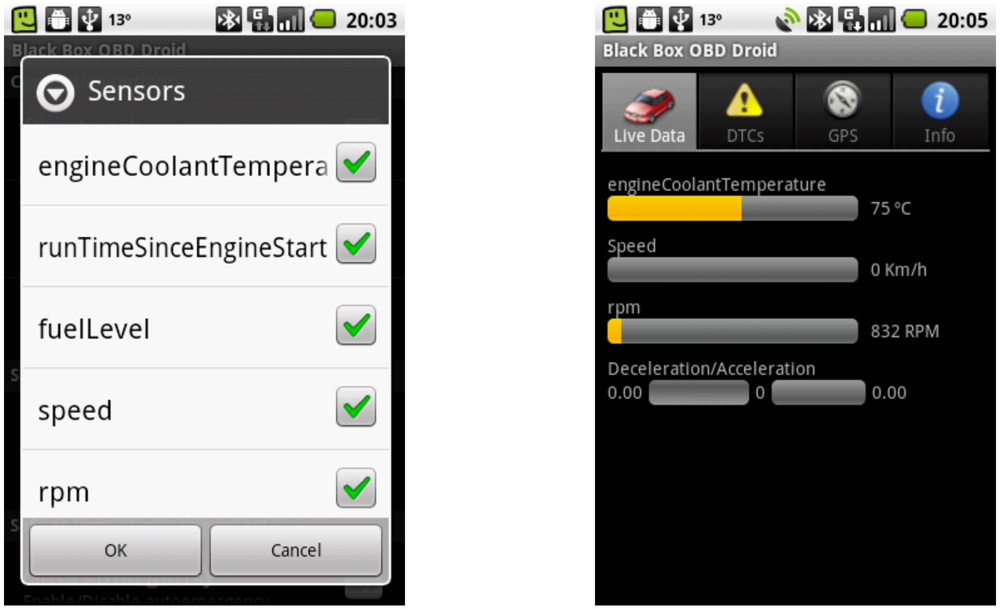
The application also offers general purpose information to the driver, including gas levels, detection of failures in mechanical elements, extensive engine feedback data, etc.

To understand why the proposed solution is feasible, it is worth noticing that, despite initial OBD-II connectors relied on RS-232 (serial port) or USB connections, nowadays we can already find in the market OBD-II connectors that provide Bluetooth or even Wifi connectivity, enabling seamless connectivity with smartphones and all sorts of mobile devices. Also, the price of such connectors is not high (typically below \$50), meaning that a large-scale adoption of such devices is possible. In terms of application deployment, the availability of application distribution services (i.e Market places) for the major operating systems available – Android/Google, iOS/Apple, Windows Mobile 7/Microsoft, BlackBerry OS/RIM - allows for a quick dissemination of such applications to users on a worldwide basis.

### A. *Implementation details*

Our application was developed for the Android platform, nowadays available in most smartphones in both Europe and the USA. Android-based smartphones typically include different wireless interfaces, such as Bluetooth, Wifi, GPS and 3G, making them ideal for our purposes. In particular, our solution will rely on the Bluetooth technology to establish a data link between the smartphone and a Bluetooth-enabled OBD-II interface. This approach removes the need for any sort of cable, thus making it more robust against car crashes. Since a data communications channel between the smartphone and an online server is required, it can be established using either the Wifi or the 3G interface. Typically, mobile telephony services, such as voice calls and SMS generation, can also be used. For instance, the system can be configured to send an SMS to our family, establish a voice channel with the emergency services, and send detailed accident information, including impact speed and current GPS position, to a special purpose server. This way, all the entities involved in the process may obtain all the information considered relevant.

Figure 4 shows the flow diagram for the startup process: first the application checks whether Bluetooth is enabled, returning an error otherwise; in a second step it attempts to contact the OBD-II device defined. In case it is found, the different protocols supported are checked to determine which



engineCoolantTempera ✓
runTimeSinceEngineStart ✓
fuelLevel ✓
speed ✓
rpm ✓

engineCoolantTemperature 75 °C
Speed 0 Km/h
rpm 832 RPM
Deceleration/Acceleration 0.00 0 0.00

Figure 6: Snapshots of the proposed application: parameter selection menu (left) and real-time parameter visualization (right).

one is valid for the current vehicle. Finally, if bidirectional communication is established successfully, the application will start the system monitoring process.

Since the most novel component of the proposed application is automatic detection and reaction upon accidents, we now address how accidents are handled. Figure 5 shows the flow diagram for the accident detection process. Basically, if either the airbag is triggered or the deceleration detected is greater than 5 Gs, we consider that an accident occurred. Notice that severe accidents are usually associated with forces above 20 Gs, and death conditions take place in impacts above 50 Gs.

To avoid false alarm situations, the user is given 1 minute to abort the actions that follow. In case the user does not abort, the accident warning procedure acts by retrieving GPS and accident details, which are automatically sent through e-mail and/or SMS to a pre-specified address. Afterward, an automatic emergency call to the emergency services is made. In case no data channel is available, the call takes place immediately.

### B. Graphical user interface (GUI)

In addition to the accident detection system, the application offers additional information to users. The first step is to establish communication with the vehicle using a Bluetooth connection to the OBD-II device. Once communication is established, the user may select the sensors considered of interest for monitoring, as shown in figure 6, left. The monitored information is then shown in the application main window (see figure 6, right), and refreshed periodically to provide the user with a feedback about the vehicle in terms of performance, problems detected, and other information of interest.

## V. Performance Evaluation

In this section we present the results of our experiments when evaluating the performance of our application at monitoring a real vehicle while moving. The first issue we address is the sampling rate performance when varying the number of sensors monitored. Notice that the CAN bus used for in-vehicle communications is shared, and different data sources will compete for bandwidth. This means that the number of samples received per sensor will vary depending on the number of sensors monitored.

In our application, both vehicle speed and airbag sensor data are continuously being retrieved. Thus, at least two sensors are always monitored. In case the user wishes to monitor other sensors as well, the extra traffic in the network associated with this additional sensing will cause the sample rate of all sensors to degrade. Figure 7 shows this behavior based on real experiments using our application. In fact, we find that not only does the individual sensor sampling rate decrease, but the total sampling rate also diminishes due to contention on the bus. Thus, in order to maintain the per-sensor sampling rate above one sample per second, we should avoid monitoring more that four sensors overall. Since our application allows defining exactly which sensors are monitored, this requirement is easy to fulfill.

For our system, acceleration information is a critical parameter. In fact, besides airbag triggering detection, acceleration values offer an alternative means to determine whether an accident has occurred. We considered retrieving acceleration information through three different sources: (i) smartphone internal sensors, (ii) GPS information, and (iii) vehicle sensors. We discarded the acceleration information based on the smartphones' G-force sensors since the range of values supported is too low to allow distinguishing between severe accidents

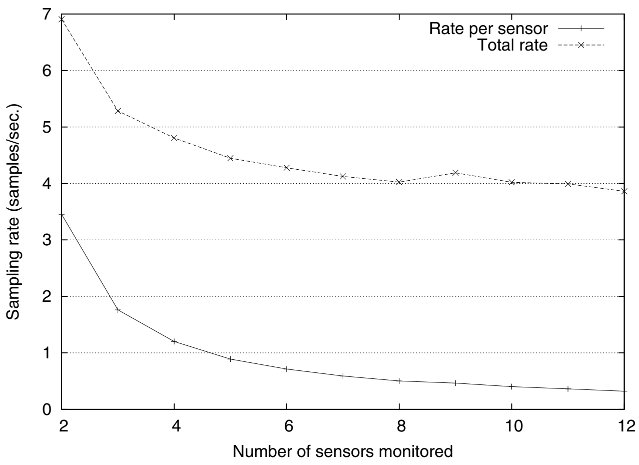


Figure 7: Sampling rate variation when varying the number of monitored items.

and other types of events (e.g. the smartphone falling on the floor). Instead, we opted for estimating acceleration values based on the derivative of speed. When obtaining acceleration estimations from GPS readings made by the smartphones GPS interface, we found that the information made available was not rigorous enough to make hard decisions about whether an accident had occurred. Basically, even small position estimation errors have a negative influence on the estimated speed, which in turn has a further negative influence on acceleration estimations derived from it, meaning that the system would not be able to discriminate correctly between normal breaking and a car crash. Notice that differences of just a few meters may be associated with quite different G-force values. Thus, we finally opted for retrieving speed information from the vehicle itself, and derive acceleration based on that information.

Figure 8 (top) shows the real speed trace retrieved from a vehicle's bus while moving. Strong acceleration and deceleration events took place during the test period to allow obtaining higher acceleration values.

Figure 8 (bottom) shows the estimated acceleration obtained from the derivative of speed for the values shown. We can observe that the speed sampling rate is high enough to allow for an accurate acceleration estimation. In this case the G forces experienced by the vehicle are always below 1G, even when applying maximum break force. This is expected since only top sports cars are able to break with a deceleration above 1 G. In fact, very high G values may only take place when real accidents occur since not even the existing Formula 1 cars can break at more than 5 Gs. This confirms the adequacy of our strategy, which relies on G force levels along with airbag triggers to detect accidents.

In case an accident occurs, it is also important to determine how long it takes for the system to react upon that event. Thus, we made tests in order to measure this and other tasks performed by our application. Figure 9 shows the average time overhead associated with the most important tasks performed. Each value represents the mean of 10 different experiments.

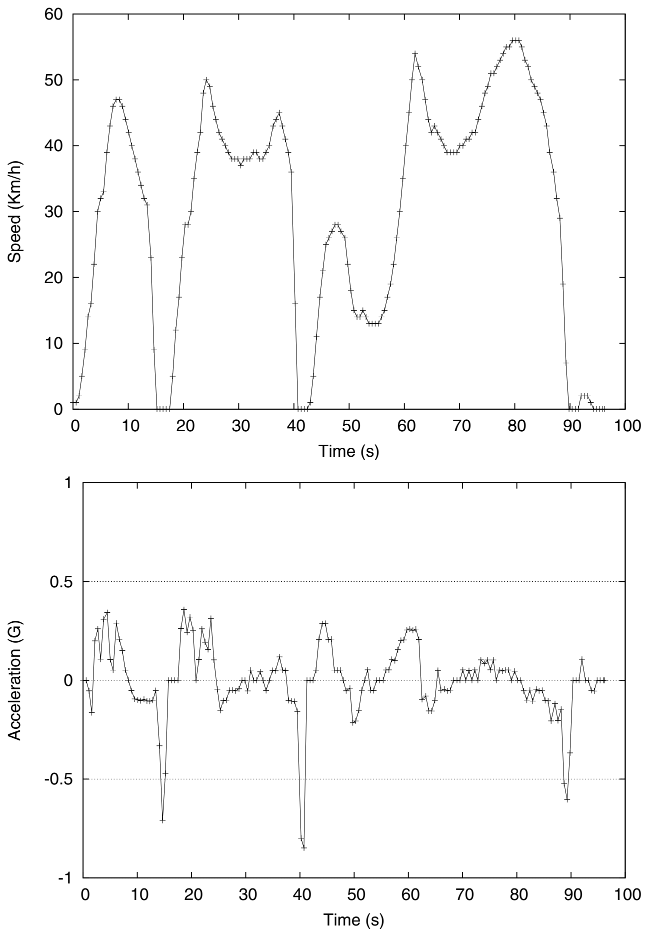


Figure 8: Speed values obtained from the OBD-II interface (top) and the estimated acceleration values (bottom).

We can see that accident initial startup procedures, which encompass Bluetooth Connection, Reset Device, Search for supported protocols, Protocol selection, and retrieving and configuring the ELM327 device, take about 9 seconds overall. Although this time overhead is relatively large, this tasks take place only once upon startup. Thus, this overhead does not represent any significant drawback to the proposed system. When the application is running, the Live Data Tab is continuously being updated with information about the different sensors monitored. This updating process introduces an almost insignificant overhead, as shown in the figure. Finally, if an accident is detected, the procedures specified in figure 5 take place. Our results show that the total time required to perform all the tasks, including the delivery of an e-mail and SMS with the accident details, followed by initiating the emergency phone call, is below 3 seconds on average, although the performance of the wireless communications channel may differ, causing this value to increase up to 6 seconds.

## VI. Conclusions and future work

Current research efforts aim at developing solutions to enable a future where the networked vehicle plays a central

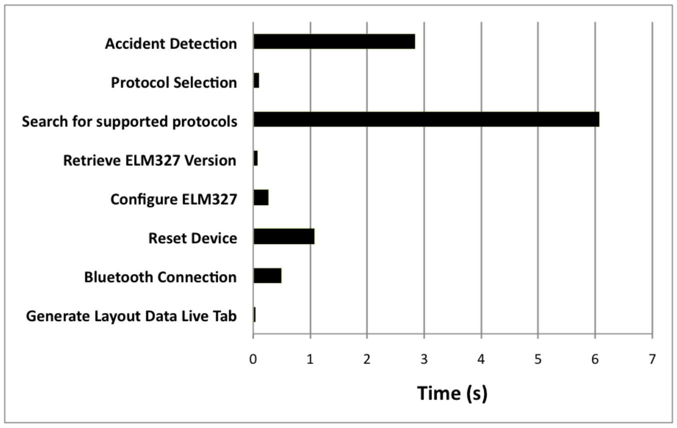


Figure 9: Overhead associated with different tasks.

roles in our lives, offering countless possibilities in sectors such as security, publicity, entertainment and commerce.

In this paper we propose combining existing vehicles with smartphones through wireless OBD-II interfaces to achieve a solution that allows monitoring the vehicle and trigger automated warning procedures in case an accident is detected. We consider that such technology allows accelerating the provision of medical services to injured people on the road through an instant emergency call, as well as pinpointing the exact position of the vehicle and offering an estimation of the severity of the impact through deceleration estimation.

We developed a prototype for the Android platform to validate our approach, and tested the performance of this prototype application in a real vehicle while moving. Experimental results show that, although the limited bandwidth of the CAN bus limits the amount of sensors that can be simultaneously monitored, critical data can still be retrieved accurately and timely since the number of critical sensors is low. Concerning accident detection, we find that the whole process takes less than three seconds, including reading critical data from the vehicle's bus, delivering information about the accident through both e-mail and SMS, and finally starting an emergency call.

As future work we plan to further validate our application in real crash tests, and to develop an enhanced emergency services server that is able to integrate the information automatically delivered by the vehicle with information manually introduced by the operator, thereby achieving a detailed accident characterization and management service.

## ACKNOWLEDGMENTS

This work was partially supported by the Ministerio de Ciencia e Innovación, Spain, under Grant TIN2008-06441-C02-01.

## REFERENCES

[1] FP7 SP1 Cooperation agreement 217643, "ASSET Road - Advanced Safety and Driver Support for Essential Road Transport," 2010.

[2] FP6 Project IST-2004-028062, "Close Communications for Cooperation between Cybercars," 2009.

[3] FP7 SCP8-GA-2009-233976, "SAFETRIP: Satellite Applications For Emergency handling, Traffic alerts, Road safety and Incident Prevention," 2010.

[4] International Organization for Standardization, "ISO 15765: Road vehicles, Diagnostics on Controller Area Networks (CAN)," 2004.

[5] U. Hernandez, A. Perallos, N. Sainz, and I. Angulo, "Vehicle on board platform: Communications test and prototyping," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pp. 967 –972, 2010.

[6] M.-C. Chen, J.-L. Chen, and T.-W. Chang, "Android/OSGi-based vehicular network management system," *Elsevier Computer Communications*, vol. 34, no. 2, pp. 169 – 183, 2011.

[7] Tarik Al-Ani, Tony Savarimuthu, and Maryam Purvis, "Android-based in-vehicle infotainment system (aivi)," in *Information Science Postgraduate Day 2010*, (Dunedin, New Zealand), pp. 51–52, October 2010.

[8] Y.-H. Cheng, W.-K. Kuo, and S.-L. Su, "An android system design and implementation for telematics services," in *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, vol. 2, pp. 206 –210, 2010.

[9] C. Spelta, V. Manzoni, A. Corti, A. Goggi, and S. Savaresi, "Smartphone-based vehicle-to-driver/environment interaction system for motorcycles," *Embedded Systems Letters, IEEE*, vol. 2, no. 2, pp. 39 – 42, 2010.

[10] C. Thompson, J. White, B. Dougherty, A. Albright, and D. C. Schmidt, "Using smartphones to detect car accidents and provide situational awareness to emergency responders.," in *MOBILWARE'10*, (Chicago, IL, USA), pp. 29–42, 2010.

[11] SAE International - Vehicle Architecture For Data Communications Standards, "Class B Data Communications Network Interface," 2006.

[12] International Organization for Standardization, "ISO 9141-2:1994/Amd 1:1996," 1996.

[13] International Organization for Standardization, "ISO 14230-1:1999: Road vehicles, Diagnostic systems, Keyword Protocol 2000," 1999.

[14] International Organization for Standardization, "ISO 15031-6: Road vehicles – Communication between vehicle and external equipment for emissions-related diagnostics – Part 6: Diagnostic trouble code definitions," 2010.