# Multimodal PDA Interfaces to Assist Drivers in Monitoring Their Vehicles

Giuseppe Ghiani, Fabio Paternò

ISTI-CNR, Via Moruzzi 1
56124 Pisa, Italy
{Giuseppe.Ghiani, Fabio.Paterno}@isti.cnr.it

**Abstract.** In this paper we present a new hardware/software solution, which allows users to easily interact with their cars' components through the OBD-II system. We propose a multimodal interface for PDAs supporting vocal and graphical commands. Our aim is to provide a safe and usable way to access the sensed engine data and vehicular status while driving. The retrieved information, which can be presented through different modalities, is used to alert the driver about some events, such as surpassing the speed limit.

**Keywords:** Car interfaces, mobile devices, multimodal interfaces.

## 1 Introduction

Vehicle driving is a daily task for most people and today's cars have a great number of high-tech devices integrated. A typical electronic assistant is the GPS navigator, which can be considered an example of assistive technology thanks to the vocal output that allows the driver to follow the route and look at the road simultaneously. Another example is the speakerphone with Bluetooth connectivity. Many car drivers use their PDAs as mobile phones and navigators because of their low weight and small size and their benefit/cost ratio. The technology has also penetrated our cars, which actually include multi-processor systems able to monitor and control the car functionalities. However, the potential benefits of such pervasive technologies are still not available for most users because of the lack of interactive solutions able to allow those with little familiarity with the technical details of car engines and computers to exploit them.

Modern vehicles are equipped with some kind of OBD (On Board Diagnosis) system to electronically read the cause of an engine failure. OBD systems were initially designed to reduce air pollution caused by road vehicles and became compulsory in California for all new cars sold since 1987. The first implementations of these systems, known as OBD-I, provided the location of a failure (which sensors or actuators were not working) and the generic cause of malfunction (typically a "circuit open" or "short circuit" problem). Since 1996 all cars sold in the United States have been required to meet the OBD-II specification [1], adopting several additional features for injection monitoring and advanced failure detection. Many other countries have since adopted these recommendations: in the European Union, EOBD (European OBD) is mandatory for all 2001 and newer vehicles, while in Japan the JOBD system is used.

Most recent cars are OBD-II compliant and thus provide a standard interface to collect data from engine sensors. In fact, sensors continuously monitor the engine status and report their data to the Electronic Control Unit (ECU) that collects them. Reading and managing these parameters with a computer requires a hardware interface (usually called a scan tool) that is an intermediate layer between the car's OBD system and the computer. So far, this type of functionality has only been used for car servicing operators in order to facilitate their work.

In this paper we describe a new hardware/software solution for allowing users to easily exploit the OBD-II system services through a multimodal PDA interface. Our aim is to provide a usable way to access the sensed engine data and engine status while driving the car. Thus, for safety reasons, using the system on board should not require excessive attention from the driver. OBD-based monitoring is very practical if a handheld computer is used: PDAs with touch screens can be quickly installed on board (those equipped with GPS are often used as portable navigators). In addition, recent PDAs have enough memory capacity and computing speed to properly manage Automatic Speech Recognition (ASR) and Text To Speech (TTS) processing. Since any distraction to the driver can increase the risk of accidents, the ability for vocal input is a fundamental aspect for on-board devices: auditory information eliminates visual distraction while vocal input reduces mechanical one.

In the paper, after discussion of related work we first describe the architecture of the interactive system that we propose. We then illustrate the tasks supported and the corresponding multimodal interface designed and implemented. Lastly, we report on an early evaluation and provide indications for future work.

## 2 Related Work

Various works about driving support services have been presented. In [2] there is a discussion of the automotive software engineering state of practice and the newest software-driven functions, which also highlights the importance of the user interface to address the new challenges. TrafficView [3] focuses on safety and describes a PDA-based structure for inter-vehicle communication of local data between neighbouring cars (GPS coordinates and OBD gathered values). Another framework for collision avoidance is proposed in [4] where the hardware components (including a mini-PC) are integrated into the car and linked to the dashboard display. In [5] a querying model for collecting and sharing information about vehicles and roads status is described. A possible use of mobile devices and OBD systems for providing failure assistance to the driver is presented in [6]. However, none of such proposals have addressed the possibility of innovative and dynamic services to provide better support for the drivers in their daily tasks, such as indicating when the speed exceeds a given limit or the car pollution is particularly high.

Many OBD scan tools are available today. Most of them are compatible with applications that have been written for both PCs/laptops and handhelds. Some complete hardware and software solutions for PDA-based OBD monitoring already exist. [8] provides cheap scan tools and parts to build them for both PC and PDA; a freeware diagnostic application only for PC is also available for download. Software for using these kinds of scan tools with Pocket PC and Palm OS handhelds is freely available at [9]. Professional solutions for diagnostic and performance monitoring are offered in [10]. However, most scan tools are not designed to be permanently installed on the car because are promoted mainly as diagnostic devices for occasionally reading and clearing the trouble codes in case of engine failure. Several scan tools consist of an OBD cable, the interpreter circuitry and a serial cable. The most compact versions have a tiny interpreter board in the OBD connector, but do not have a switch to turn off the circuitry. Thus, when the car is not used the scan tool must be physically disconnected, and this is unpractical due to the typical position of the vehicle's OBD connector (below the driving wheel block). For this reason, we have also developed a custom wireless scan tool (see Figure 1) suitable for everyday use together with the mobile application.

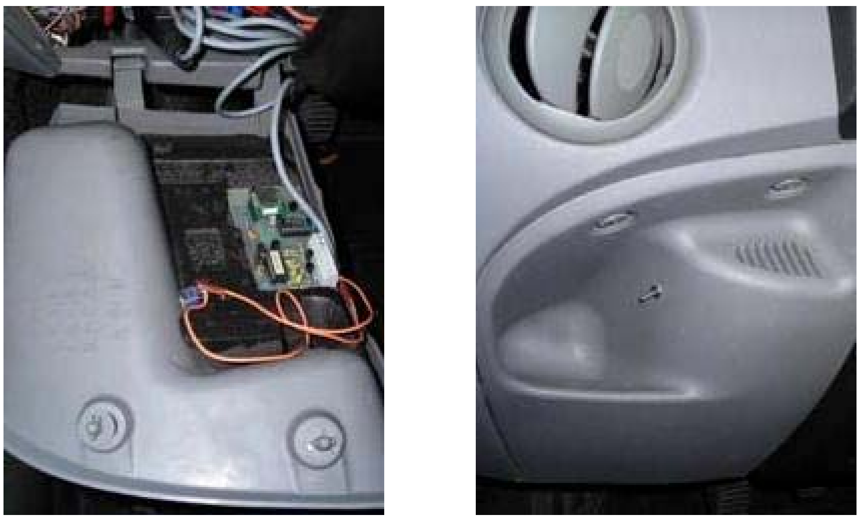


**Fig. 1.** Our scan tool has been fixed to the door of the fuse box (left). The power switch is accessible from the driver's seat (right).

## 3 System Architecture

The system architecture includes the car's OBD and we do not make any modification on the car engine neither on the ECU. Our system is composed of the scan tool hardware and the application running on a PDA (see Figure 2).

We considered that normally a general purpose computer cannot be directly connected to a car's OBD system. The main reason is that the voltage levels on the car OBD connector differs from the RS232 ones. Thus, a direct connection may damage a serial port. Another issue is the type of OBD protocol supported by the ECU, that usually requires the bus initialization before “talking” to the car. For these reasons, we needed a scan tool peripheral to link the PDA to the OBD connector. Unfortunately, connecting a PDA to a peripheral is often difficult. An adapter cable, a special card or even an expansion device are needed to provide a PDA with the RS232 capability. Cables and adapters consume space inside the car and take time to connect whenever the driver gets on board and wants to use his PDA. For this reason, a scan tool with Bluetooth connectivity has been designed. This prototype complies with the OBD ISO9141 [14] protocol and has an integrated Bluetooth module which does not need any external power source.

The limited dimensions make this scan tool extremely unobtrusive because it can be hidden under the dashboard. The only visible component is the power switch for turning off the scan tool when it is not needed, avoiding unnecessary vehicle battery load (see Figure 1).

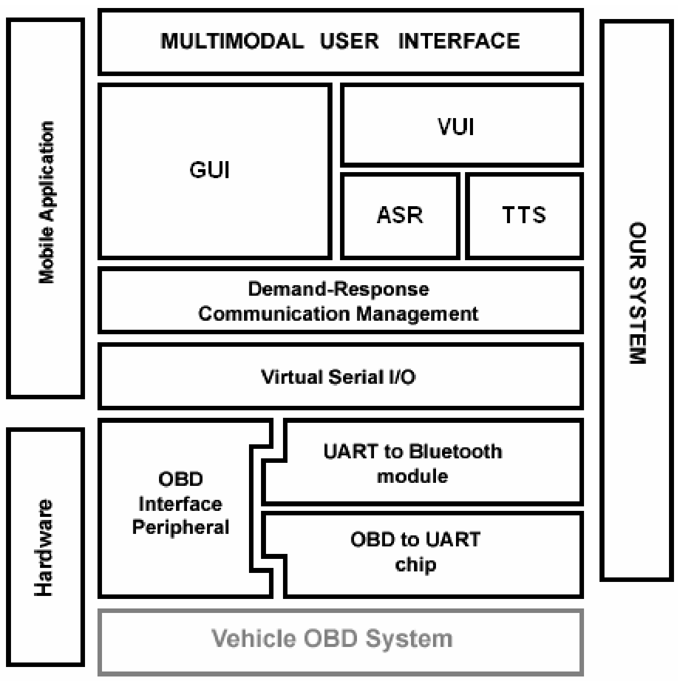


**Fig. 2.** System architecture.

## 3.1 Hardware: the OBD Interface Board

The OBD Interface Board is a 100 x 50 mm circuit board. The “car side” part is the OBD to UART (Universal Asynchronous Receiver/Transmitter) interpreter, which provides a way to directly exchange ASCII bytes with the car’s ECU managing the ISO9141 features [14]. Such a chip manages the OBD bus initialization and many other manners of ISO9141 features. The part communicating with the computer is a stand-alone UART to Bluetooth integrated module with Serial Port Profile (SPP). For the Bluetooth module technical details refer to [11]. It can be discovered and registered by any computer with Bluetooth capability. The computer can communicate transparently with the OBD chip through a virtual serial port. This is done by the operating system, which binds a COM port name (e.g. COM6 or COM8) to the discovered Bluetooth device; the association between port name and device is stored in the registry file, which can also be edited manually.

The designed scan tool is a very simple device: inexpensive, common components have been used to build it. Providing compatibility with non-ISO OBD-II protocols would simply require replacing the OBD interpreter with another specific version or with a multi-protocol chip. Thus, the general circuitry structure would be maintained.

## 3.2 Software: the Mobile Application

The OBD interpreter is quite recent and is still considered an experimental chip. No software component library is available to simplify its use. For this reason, we had to develop the communication routines following the interpreter chip specifications contained in its datasheet. Chip information and datasheet are available at [12].

The application communicates with the car in a demand-response manner. Requests are OBD “mode 1” commands. To read a parameter value, the command consists of the string “01” followed by the desired Parameter ID (PID). Parameter IDs are two-character, OBD-II compliant keywords and our application can read about 20 parameters. Note that not all of them are retrievable from every car: in the next section we will discuss how the user is informed about the PIDs supported by the vehicle. A typical PID is the “0D”, which is the keyword associated to the speed sensor. Therefore, to request the current vehicle speed, the “010D” string must be sent to the scan tool. After each request, the application waits for the answer message, which contains a header and a data field. Writing and reading on the I/O stream are made through the (virtual) COM port used, specified by the user on the settings form.
Since wireless connections are likely to be broken sometimes, an intermediate layer has been defined and placed between the standard serial communication library and the rest of the application, which detects the disconnection and provides the reconnection.

The graphical user interface has been designed to optimize the limited space available on the screen. Large character fonts have been used to improve readability. Input components, such as buttons, have been made wide enough to be used with a finger. The interaction is easier when the PDA pen is not required; this is true both on board if the PDA is fixed to the holder, and outboard during an engine inspection: a tiresome situation would result if the pen fell inside the engine compartment. The multimodal interface further simplifies the driver’s operations: it supports vocal commands and provides vocal output so that the driver can keep his eyes on the road. ASR technology used on this application is speaker-independent and available in many languages. Moreover it is extremely tolerant of environmental noise and background speech.

## 4 Tasks Supported

The range of parameters that can be read from an OBD-II system depends on the vehicle’s type. Every parameter is related to a sensor located on the engine compartment. Some of the sensed data are presented on the dashboard, very often by analogue instruments. However, the digital values monitoring can be useful even for the information already available on the dashboard. Reading an exact value from the PDA’s display can show a bad calibration of the analogue instruments. Depending on the driver’s preference a different measure unit may be used to translate the value. Vehicle speed and air/coolant temperature can be displayed in Metric or Imperial/US units. The largest possible set of values should be readable, while the user may select just a subset of them. This is because not all drivers have the same automotive/motor skills, and each user is mainly interested in knowing parameters that s/he can understand. The next section describes the functionality currently supported.

### 4.1 Real Time Parameters Reading

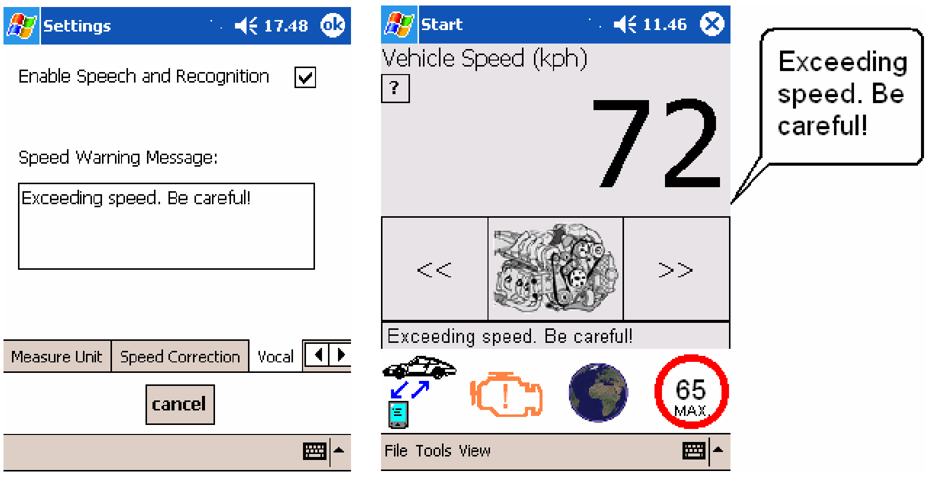
After the OBD bus initialization, the application asks the car’s ECU for the set of supported Parameter IDs (PIDs). Every PID is a keyword referring to a certain parameter that can be retrieved from the ECU. For each PID the user is provided with the flag indicating if it is supported or not by the vehicle. An unsupported PID usually means that the related sensor is not present on the car. Users may customize the parameter list depending on the sensors present in their cars and, of course, their knowledge. The upload frequency provided by the car’s ECU and the scan tool is quite low. For this reason the default presentation for read values is single-parameter oriented: if just one parameter is monitored at a time its refresh rate is the highest. Another reason is that when a single parameter is presented it can be printed with the biggest font and it can be more clearly readable. Reading all the enabled parameters at the same time is also possible by the multi-scan form: the user should give a priority to every parameter. The priority affects of course the upload frequency and allows to update more often those values that changes frequently (as the engine RPM). Since the PDA screen is small, little character fonts are used to display such amount of information. Otherwise, the user can select the desired parameters from the custom list, then the parameter values are shown on a label that is updated about 2-3 times per second.

Some drivers may find it useful to correct the speedometer readout based on tyre size: this function computes a correction factor for the vehicle speed from the tyres and wheels measurements. This can be done if the user enters the original tyres size and the current tyres size for his car. When the correction is explicitly requested, the application automatically provides a vehicle speed closer to the real one. This feature may be useful when the GPS signal is not available such as into road tunnels (where speed is often controlled by police). Otherwise a simple navigator software could, provide the real car speed.

A typical OBD-retrievable parameter is the "Estimated Engine Load", computed by ECU in function of the engine status (intake air pressure, throttle position, engine RPM, etc.). As default, the user is provided with the percentage engine load value; otherwise, if the user sets the maximum Horse Power (HP) of the vehicle, the current estimated output power can be displayed.

## 4.2 Interesting Events Notifications (Warnings)

The driver can be alerted about some events related to the current values of the monitored parameters. When the vehicle speed is monitored, the desired speed limit can be set. The application will provide a vocal warning when the limit is exceeded. Users may also custom the warning message by entering the exact sentence that it should communicate (see Figure 3). If speech and recognition are not enabled, when the speed limit is exceeded an audio clip is played.


Enable Speech and Recognition ✓

Speed Warning Message:

Exceeding speed. Be careful!

Vehicle Speed (kph)

72

Exceeding speed. Be careful!

**Fig. 3.** Customizable speed warning message (left) and vocal notification (right).

The "head lamps reminder" informs the driver to turn on the headlights when a certain speed is reached. This function may be useful in those countries where the headlights must be always kept on in suburban areas.

Diagnostic Trouble Codes (DTCs) are error codes caused by engine malfunctions. When a malfunction is detected the ECU allocates a DTC instance on the embedded flash memory and turns on the Malfunctioning Engine Lamp (MIL) on the dashboard. If the problem is not serious the MIL indicator is soon turned off, but the DTC remains stored on the ECU. While reading the current parameter, the application periodically queries the ECU for stored DTCs. The automatic querying for DTCs provides the driver with the information about any previous failures.

Modern vehicles have a single MIL lamp that can be simply on or off. The MIL icon of this application can be in several states, supplying much more information to the user (see Figure 4). If there is no car connection the "?" caption is shown on a grey icon. When communicating with the car, if the dashboard MIL lamp is off and there are no stored DTCs, then the MIL icon caption shown is "OK". If there are serious DTCs on the ECU memory the application shows a red MIL icon. In this case, the dashboard MIL is also on, and the user should read the DTC(s) description. The orange MIL icon means that, although the vehicle MIL is off, at least one DTC is still stored in the ECU, and the user may open the DTCs form to display the previous trouble list.

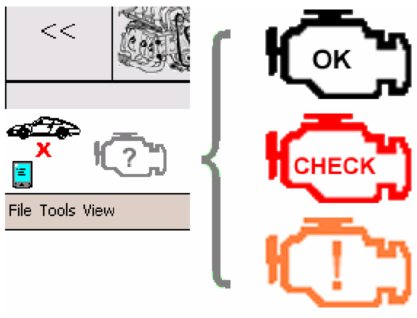
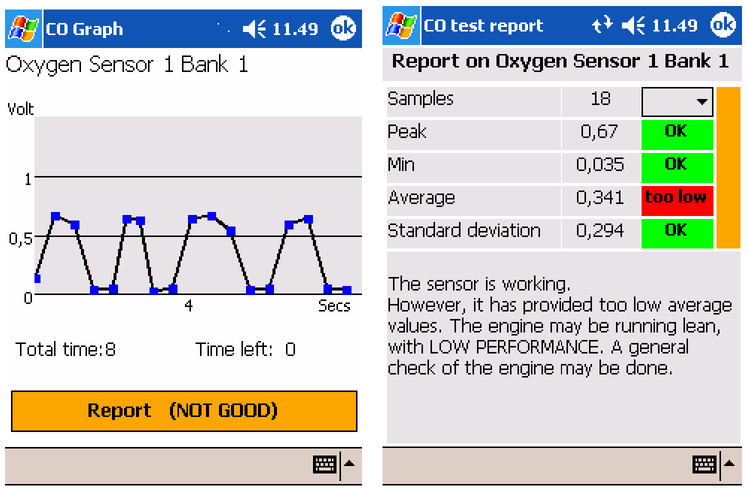


**Fig. 4.** The different MIL icons used by the application.

## 4.3 Quick Efficiency Checks

Semantic analysis of monitored values may highlight an engine problem or a poorly working sensor. In the "Oxygen Test", the average and standard deviation of the Oxygen Sensor sampled data are calculated. These values allow users to estimate whether the sensor is working correctly and the engine is running well. The test result is graphically presented and a report is given in a few short sentences (Figure 5). The evaluation report is selected by following a simple decision chart (see Figure 6).

The Oxygen Sensor is probably the most important part of modern engines. It measures the concentration of oxygen in the exhaust gas. The ECU uses the Oxygen Sensor output to maintain the best air-fuel mixture ratio. If the sensor is not working or its output is flawed, the engine may run badly with high fuel consumption and polluting emissions. By doing the simple oxygen test provided by our application the user can quickly check if the injection is working well and he is informed if a suspected malfunctioning exists. If so, the user can decide to bring the vehicle to a mechanic or to do a check himself. We are now considering creating other simple checks, such as the “idle RPM” and the “idle accelerator position” ones.



Report on Oxygen Sensor 1 Bank 1

| | | |
|---|---|---|
| Samples | 18 | |
| Peak | 0,67 | OK |
| Min | 0,035 | OK |
| Average | 0,341 | too low |
| Standard deviation | 0,294 | OK |

The sensor is working.
However, it has provided too low average values. The engine may be running lean, with LOW PERFORMANCE. A general check of the engine may be done.

**Fig. 5.** Oxygen sensor test result (left) and test report (right).

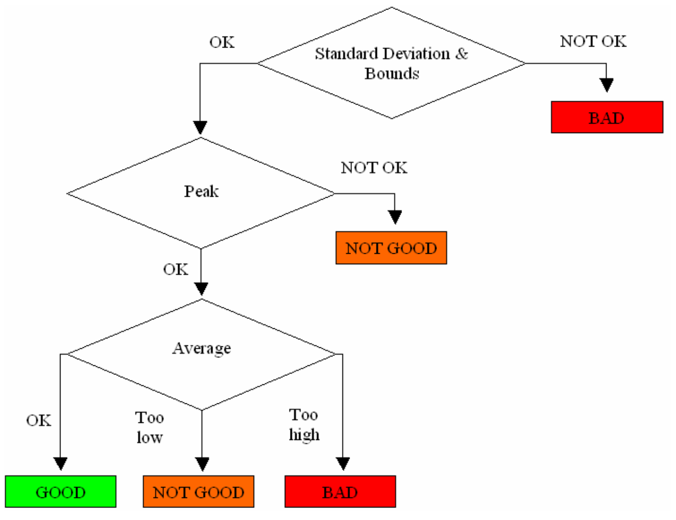
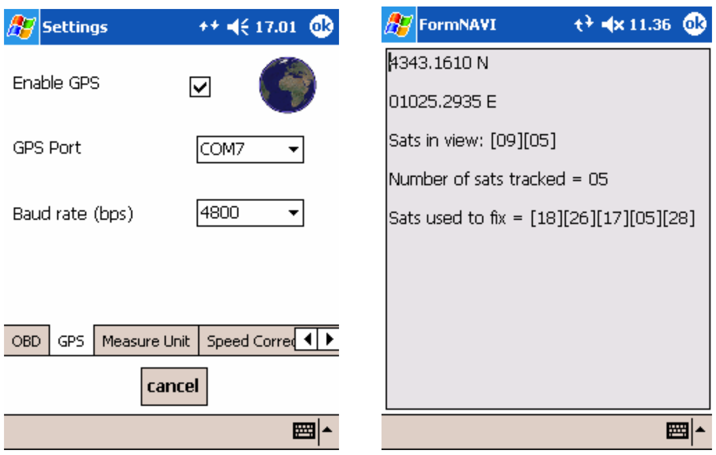


**Fig. 6.** Oxygen Test evaluation flow chart.

### 4.4 Location-aware functions

The association between values (e.g.: speed, engine load/temperature) and vehicle terrestrial coordinates is basic to develop services based on vehicle-to-vehicle (v2v) and vehicle-to-infrastructure (v2i) data exchange [4]. We considered the location-awareness as a complementary aspect for the car status monitoring. To facilitate future data logging enhancements, a GPS decoding routine based on National Marine Electronic Association (NMEA) protocol has been developed. Such a function connects to the GPS port and gets the downstream bytes. Coordinates values and satellites information are parsed from GGA, GSA, GSV and RMC sentences and presented on a specific form (see Figure 7).



Enable GPS ✓

GPS Port COM7

Baud rate (bps) 4800

4343.1610 N

01025.2935 E

Sats in view: [09][05]

Number of sats tracked = 05

Sats used to fix = [18][26][17][05][28]

**Fig. 7.** GPS settings (left) and coordinates display form (right).

On the GPS settings page the user should enter the COM port of the GPS module (see Figure 7). On the combo box just the valid ports are listed. This approach allows the use of a generic GPS. The decoding function has been successfully tested on a PDA with integrated GPS and on a PDA with a CF (Compact Flash) GPS Card.

## 5 Multimodal Interface

Most “on the road” tasks can be done without using the pen. However, while car passengers may physically interact using their hands, the driver should not for safety reasons. The risk caused by in-car usage of mobile devices is discussed in [7], where different types of driving distractions are described: it has been proven that every device, even the vehicle’s instrumentation, may distract the driver.

We considered the vocal input to be a possible solution to provide accessibility to the main functions, reducing mechanical and visual distraction. Several tasks have been made accessible through vocal keywords, and the feedback is provided both graphically and vocally. When the user is driving, s/he can set a new speed limit or a new parameter to check by just saying a few simple keywords.

Vocal processing is managed by the Loquendo Embedded ASR/TTS engines [13]. Loquendo functions have C/C++ interfaces. Since the main application is written in C#, two different DLL components were developed in C++ language and imported on the C# application:

- AsrMod.DLL is initialized with the grammar file from which a Recognition Object (RO) is created. The asrRecog( ) method listens to the user voice trying to recognize the sentence. The returned structure, basically a string, is then interpreted by the application.
- TtsMod.DLL defines the ttsRead( ... ) method, which accepts a string and produces a vocal audio file. The file is automatically played by the computer audio board.

The speech sessions are usually created at start up. Before saying a vocal command, the user must press a keypad button of the PDA and the application responds with an audio prompt. When ASR is successful, the new status is briefly described by a TTS vocal message (see Figure 8). We opted for this approach so that the speech recognition function is called only when the keypad is pressed. Using a continuous speech processing would have been difficult, because environmental noise would be processed uselessly. As a consequence, engine noise or passenger speech would cause undesired selections.

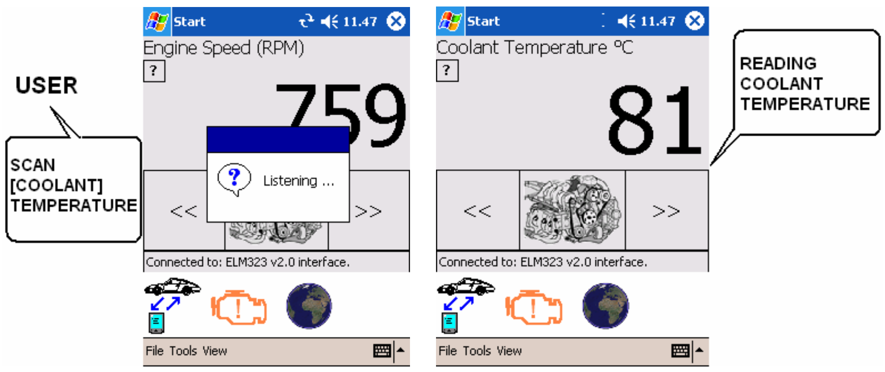

USER
SCAN [COOLANT] TEMPERATURE
Start 11.47
Engine Speed (RPM)
759
Listening ...
<< >>
Connected to: ELM323 v2.0 interface.
File Tools View
Start 11.47
Coolant Temperature °C
81
<< >>
Connected to: ELM323 v2.0 interface.
File Tools View
READING COOLANT TEMPERATURE

**Fig. 8.** After pressing the key pad, the user can say a vocal command (left) and the application provides a vocal output describing the new status (right).

The grammar containing the vocal rules is compiled “on the fly” during the ASR initialization. The grammar is recursively defined in EBNF and stored into a text file that can be easily edited without modifying the entire application. For each subsequent call the recognition function produces a string with the words (and/or numbers) provided by the user; the string is a sentence whose structure complies with the grammar rules. The grammar has been made general enough to recognize different forms of the same command. For example, to set 65 as the maximum speed limit, the user can say “Maximum speed sixty-five”, or “Maximum sixty-five” or even “Sixty-five”. The resulting string is processed by the application, which properly interprets the relevant words. Such flexibility is useful especially for beginner users, that are not familiar with the commands. However, excessive flexibility means a high probability of misunderstanding of the commands: the wrong command may be chosen by the ASR recognition object if too many possible choices are available. We are now evaluating the grammar structure to achieve an optimal trade off between flexibility and manageability for the speech recognition.

## 6 Evaluation

A preliminary user test to evaluate the interface has been performed indoors Six volunteers were involved. Users ranged in age between 24 and 33 (average 27.8). Fifteen percent were female, and 50% had previously used a PDA. Users had good experience with PC-based interfaces, but very little with vocal ones.

Users were initially asked to execute 10 tasks through traditional touch screen interaction. They had to enter the settings for the connection to the scan tool hardware, to find all the parameters retrievable by the vehicle and to disable those that they were not interested in. The configuration steps are very important for the user to realize what information would be provided and must be done at least once during the first session. Common functions were also activated through the touch screen:

- select a specific parameter (it should be enabled beforehand),
- customize the available speed limits,
- select a speed limit,
- switch to another measurement unit,
- from the graphical-selection form select a specific parameter (it should be enabled beforehand),
- cancel the speed limit.

The users were observed while trying to accomplish the tasks without any interference. The observers took note of the task completion time and any comments made by the users. At the end of the first part, users were provided with the grammar structure and some examples of vocal commands. Then, they re-executed some typical on-the-road tasks through vocal commands:

- freely select a parameter,
- request the vocal description of the new parameter,
- set a speed limit,
- select a specific parameter,
- select the next parameter available,
- cancel the speed limit.

After the session, users were requested to fill in an evaluation questionnaire. The questionnaire asked users to indicate their educational level and to evaluate the system capabilities. User were also asked to rate the parameters shown in Table 1 on a 1 to 5 scale.

**Table 1.** User rating for the interaction attributes.

| **Parameters** | **Point score** | **Variance** |
|---|---|---|
| User interface clearness | 3.3 | 1.29 |
| Icons intuitiveness | 3.5 | 2.25 |
| Customization easiness | 3.3 | 1.75 |
| Vocal input efficacy | 4.16 | 0.81 |
| Vocal feed-back efficacy | 4 | 0.66 |
| On-drive safety | 3.3 | 1 |
| Benefits | 3.83 | 0.47 |

The test results highlighted that the GUI can be improved in terms of clarity. Users confirmed this, indicating that the menu structure and the icons should be more intuitive. Afterwards, we made several modifications to the GUI: menu captions and items have changed significantly and the main form icons are more intuitive.

Before starting the multimodal interaction, users received only a brief description of the vocal syntax. Nevertheless, most vocal commands were interpreted by the system surprisingly well. One reason seems to be the flexibility of the grammar, which defines different formats for the same command. Another reason is the reliability of the Recognition Object (RO) generated from the text grammar at ASR initialization time.

## 7 Conclusions and Future Work

We have presented a novel solution supporting a number of novel services for car drivers. Our application was developed for a typical handheld currently available in the market, the HP iPAQ hx2700. Even if the ASR and TTS functions require a lot of memory, they seem to be very efficient even while the application is communicating with the car ECU.

This aspect suggested the development of a new service: a sort of programmable scheduler for vocal notifications. The scheduler would periodically create vocal messages to describe the current value of the parameters of interest to the user. An advanced Oxygen Test may consider specifically the downstream Oxygen Sensor output. The downstream sensor is mounted after the catalytic converter on the exhaust pipe and measures the toxic substances actually emitted by the vehicle. Monitoring both the upstream and the downstream sensors output would allow estimating the actual catalytic converter efficiency. Saving the test result would provide a diagram of the catalytic converter degradation as a function of the car age. Moreover, several efficiency tests may be automatically performed by the application, which would inform the user about the test results through multimodal messages.

We are now considering improvements to our application for specific purposes. The functions that may be enhanced or created would depend on specific needs. Car rental or other companies with vehicle fleets may be interested in logging-oriented functionalities to monitor their customers' driving habits. Engine status data, together with GPS coordinates, provide information about the driver behaviour in a certain place. Logged data may be sent wirelessly to a gathering point where they would be analysed by a desktop-based specific software. In addition, several values related to the engine (such as temperature and engine load) can be considered to study the vehicles' behaviour, that is, how they react to the environment.

## References

1. Society of Automotive Engineers – SAE International. http://www.sae.org
2. Manfred Boy. Challenges in Automotive Software Engineering. Proc. ICSE 2006, ACM Press (2006), 33-42.
3. Tamer Nadeem, Sasan Dashtinezhad, Chunyuan Liao and Liviu Iftode. TrafficView: Traffic Data Dissemination using Car-to-Car Communication. Mobile Computing And Communications Review, Volume 8, Number 3, 6-19.
4. C. L. Robinson, L. Caminiti, D. Caveney, K. Laberteaux. Efficient coordination and transmission of data for cooperative vehicular safety applications. Proc. 3rd International workshop on Vehicular ad hoc networks, ACM Press (2006), pp.10-19.
5. V. Bychkovsky, K. Chen, M. Goraczko, H. Hu, B. Hull, A. Miu, E. Shih, Y. Zhang, H. Balakrishnan, S. Madden. Data management in the CarTel mobile sensor computing system. Proc. SIGMOD 2006, ACM Press (2006), pp.730-732.
6. G. Houben, cJ. Van den Bergh, K. Luyten, K. Coninx. Interactive Systems on the Road: Development of Vehicle User Interfaces for Failure Assistance. Proc. First Workshop on Wireless Vehicular Communications and Services for Breakdown Support and Car Maintenance (W-CarsCare), 2005, pp.84-89. http://citeseer.ist.psu.edu/houben05interactive.html .
7. L. Chittaro and L. De Marco. Driver Distraction Caused by Mobile Devices: Studuying and Reducing Safety Risks. Proc. International Workshop on Mobile Technologies and Health: Benefits and Risks, Udine, June 2004.
8. http://www.scantool.net .
9. Dana Peters. OBD Gauge for Pocket PC and PalmOS, http://www.qcontinuum.org/obdgauge .
10. Vital Engineering Ltd website, http://www.vitalengineering.co.uk .
11. Promi Esd 02 bluetooth module datasheet. Initum website support section, http://www.initium.co.kr .
12. ELM323 OBD (ISO) interpreter datasheet. Datasheet section of the ELM Electronics website. http://www.elmelectronics.com .
13. Loquendo Embedded Technolgy. http://www.loquendo.com
14. ISO Document "Road vehicles -- Diagnostic systems -- Requirements for interchange of digital information"