Developing Mobile Apps to Protect Environment

Iulia Georgiana Rînea[1], Delilah Florea[1], Adrian Florea[2]
[1]Informatics and Computer Science Department
Samuel von Brukenthal National College of Sibiu, Huet Square, No. 5
Sibiu, Romania
(iulia_rinea, delilah_florea)@yahoo.com

[2]Computer Science and Electrical Engineering Department
Lucian Blaga University of Sibiu, Emil Cioran Street No. 4
Sibiu, Romania
adrian.florea@ulbsibiu.ro

ABSTRACT: *The widely dependence on mobility has became nowadays a priority for people. The automotive industry has lately grown impressively a lot. Most of the people give cars an important place in ranking their priorities, and this is the reason of developing an Android application for mobile phone, to help resolve certain car problems, which often occur. This application is divided into three categories that facilitate the elimination of car damages. The first part is composed of symptoms, which users visual or auditory notice (observing a particular engine noise, unusual smoke detection). This part helps the car owner to repair the vehicle with only basic information in this domain. The second part is composed of a Google Map, which shows all the car services in a specific area. This part of the application is dedicated to serious problems of the vehicle, which cannot be detected and which can damage the car. The third part consists of connecting the phone to the onboard computer through a device OBD II ELM-327 Bluetooth that allows certain functions to appear on the mobile screen display (speed, air intake, fuel level). In conclusion, we believe that this application is useful, because it helps to clarify car key issues, contributing in this way also to solving the environmental problems regarding pollution.*

Keywords: Mobile Apps, Google Maps, Vehicle Mobility, Graphical User Interface, Software Design

Received: 11 March 2017, Revised 9 April 2017, Accepted 19 April 2017

1. Introduction

This paper extends the previous work that was presented at the Ninth International Conference on Applied Informatics: Imagination, Creativity, Design, Development (ICDD 2017) [1], by presenting some types of malfunctions that may occur when cars is running and can cause increases of pollutant emissions that the proposed application might identify

in its future versions. Also some arguments for developing a mobile application running on Android operating system are added.

This paper introduces "A Diagnostic" an application developed to simplify drivers problems that may occur in traffic and also useful for environmental protection. This application is developed for the Android OS operating system, and this choice has been made following the frequent use of mobile phones to solve various problems. The types of computing devices used by consumers and enterprises have shifted significantly over the past decade. Whereas in 2010, traditional PC sales (desktops and laptops) outnumbered other computing platforms, in 2014, roughly 300 million PCs were sold compared to 200 million tablets and 1.9 billion smart phones [2]. Also, mobile computing has had an exponential growth. For example, in US, adult smartphone device ownership was at 33 % in 2011, 56 % at the end of 2013 and 64 % in early 2015 [3]. In Germany over half of consumers used in June 2014 a smartphone, compared with 41 % in 2013 [4] and 63 % of consumers ages 16 and older went online via smartphone each day. We chose Android OS and Java language because our interest in operating the mobile phone in the form of a useful device has always increased, and motivated by the introduction of Android courses in teaching process in high school.

Analyzing the information around us, we tried to establish the priorities of people nowadays and came to the conclusion that two of them are the mobile phone and the cars.

The application is therefore designed to detect vehicle malfunctions. Often these problems are encountered while the car holder is driving, which causes great discomfort. In these cases, the application will prove useful, because it helps the user to make a brief diagnosis after the symptoms they notice at the car, discovering the reason of their problem. For example, he notices that the engine sounds or knocks in an unusual way, and looking for the problem in the application, he realizes that it has occurred due to poor quality or impurity in fuel, and investigating the last fuel supply of the vehicle, he realizes that this is the problem he has and he can solve it by himself, with only basic information. He can choose the fuel model and car brand, and then a search bar appears, where he can enter the detected symptoms. He must choose the two specified criterias, because every car brand has some specified malfunctions, which differ from car to car. Premature detection of malfunctions can prevent advanced vehicle damage and can also help to keep the environment cleaner through less pollution. In chapter 4 we have presented some types of malfunctions that may occur and can cause increases in emissions that the proposed application might identify in the future.

Problems can also be very complicated, so the only option is bringing the car to a car service. If he is located in an unknown area and does not know where to call and if the machine is having serious failures, then the only option is to search for car services in that area and select those which are most convenient for him. Again, the app will make life easier by offering all the services within a radius of 10km. With the help of Google Maps, we were able to create the map, and with the help of specific features available to application developers, we were able to display the Marker service, which indicated the car services. Each of the services is accessed as an HTTP request, and returns either an JSON or XML response. All requests to a Places service must use the *https:// protocol, and include an API key*.

The Google Places API Web Service uses a place ID to uniquely identify a place. After the marker has been set, the user can click on it and some option will appear on the display.

The third part of the app tries to offer something unique to car owners, namely the possibility of communicating with the car. The only way of communicating with the machine they knew was through the mechanic, who through some devices managed to connect the PC to the car's computer. Now, with this application, they have the chance to do this on their own with the help of their mobile phone. With an OBD II (on-board diagnostics) device, the ELM-327, it is possible to connect this device to the mobile phone via Bluetooth. There are many sensors throughout the car: oxygen sensors, engine knock sensors, manifold pressure sensors and so on. Each one of these sensors sends a signal to the car's computer the Engine Control Unit (ECU). The ECU uses that information to adjust different elements of the engine operation, the fuel injection or the spark timing for example. If the information that the ECU gets from one of its sensors is too far out of whack, it saves a code called a Diagnostic Trouble Code (DTC), which are shown by the application. Through this OBD interface, it is permissible to charge certain mobile computer parameters to the mobile phone, such as speed, oil level, air intake, fuel type, etc. The general workflow of the application functionality should go like this:

1. Connect to the OBD II adapter through Bluetooth;

2. Initialize OBD II adapter with AT commands;

3. Continuously get data from the vehicle through issuing the corresponding PID (Parameter ID) codes.

2. The Graphical User Interface

The first access to the functions of the application is realized by a menu, containing 3 buttons, representing the three parts mentioned in the introduction, which can be seen in Figure 1.

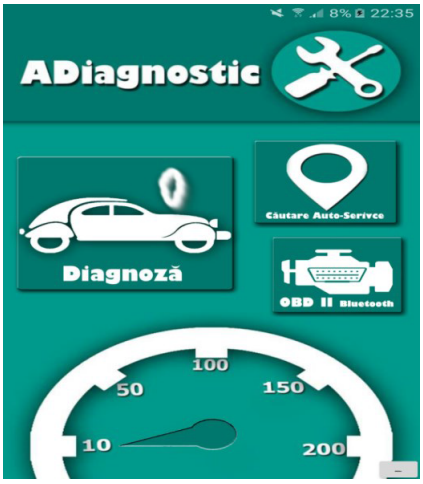

Figure 1. Main Interface

Figure 2. Diagnosis Button

This button allows identification of a malfunction after the symptoms reported by the user.

Figure 3. Loading map button

This button allows the display of car services according to the user's location.

Figure 4. OBD-II connection button

This button allows the phone to connect to the OBD II interface.

3. Application Software Design

3.1 The Programming Environment

This app was created with the Java-based, Android object-oriented language, created in the Android Studio 2.2.2 programming environment. For compatibility, we used the Android SDK (Software Development Kit), which is needed to develop and test the application. It has been tested both on the phone and on the virtual device, Android Emulator,

which can be created in this programming environment [5], [6]. For testing the third part of the app, the one with the OBD II, it was necessary to use a real car, to have access to the on-board computer. The application requires two API Keys, one for map implementation and one for Google Place API implementation. We got the two APIs through Google Console Developer through creating the project and adding the key obtained by registering it.

3.2 Code Sections

3.2.1 Diagnoza Activity

In this section, you can choose the fuel type of your machine and its brand, which are passed as parameters for later diagnosis.

```
spinner1.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
if (position == 1 || position==2){
spinner2.setAdapter(adapter2);
iv.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
new Handler().postDelayed(new Runnable() {
@Override
public void run() {
String mStr=spinner2.getSelectedItem().toString();
Intent mIntent = new Intent(DiagnozaActivity.this, Search.class);
mIntent.putExtra("Value", mStr);
startActivity(mIntent);
} 1000)} } ); }
```

The figure 5 shows a diagnosis for an Alfa Romeo type Diesel engine:

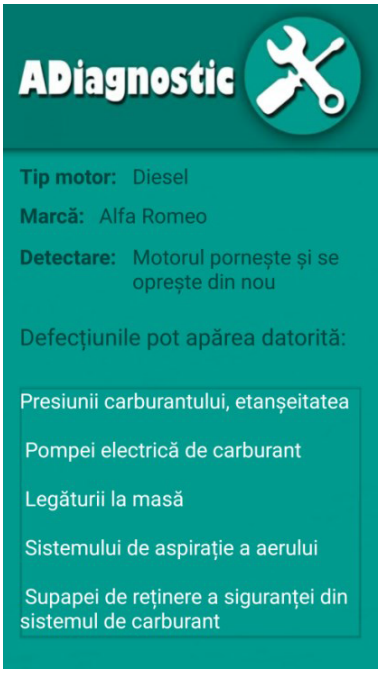

Figure 5. ECU malfunctions

To set the search function, it was created a method to call up existing features in the Android Search libraries.

```
private void setupSearchView() {
mSearchView.setIconifiedByDefault(false);
mSearchView.setOnQueryTextListener(this);
mSearchView.setSubmitButtonEnabled(false);
mSearchView.setQueryHint(getString(R.string. cheese_hunt_hint));}
```

3.2.2 Maps Activity

To set up the map, the functions below are called, creating a variable in where the map is saved (*GoogleMap*), and then will be set up the style (*json_style*) into resources. All functions will be called using the GoogleMap variable (**mMap**) [7], [8].

```
mMap = googleMap;
MapStyleOptions style=MapStyleOptions.loadRawResourceStyle (this,R.raw. json_style);
mMap.setMapStyle(style);
```

In the next section of the code, I will present the usage of GPA over a certain radius (**prox_rad** = 10000).

```
googlePlacesUrl.append("& radius=" + prox_rad);
googlePlacesUrl.append("& key=" + "AIzaSyATuUiZUkEc_UgHuqsBJa1oqaODI-3mLs0");
return (googlePlacesUrl.toString());
```

The picture illustrated in Figure 6 shows a service location map relative to the driver's position that has a malfunction.

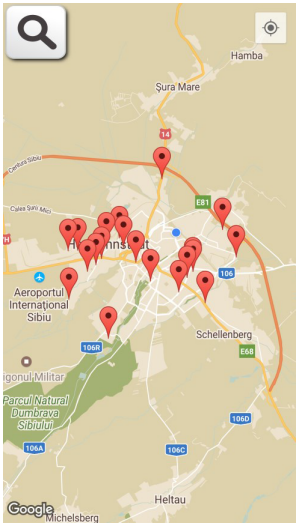

Figure 6. Services on map

3.2.3 OBD Activity

The Bluetooth initialization for connecting the onboard computer to the OBD II device [9] is done as follows:

```
//initializare Bluetooth
final BluetoothAdapter btAdapter = BluetoothAdapter. getDefaultAdapter();
```

```
if (btAdapter != null)
bluetoothDefaultIsEnable= btAdapter.isEnabled();
```

Function to start GPS and Bluetooth:

```
private void startLiveData() {
tl.removeAllViews();
doBindService();
// pornirea comenzii
new Handler().post(queueCommands);
if (prefs.getBoolean(ConfigActivity. ENABLE_GPS_KEY, false))
gpsStart();
else
gpsStatusTextView.setText(getString(R.string.status_gps_not_used));
wakeLock.acquire();
if (prefs.getBoolean(ConfigActivity.ENABLE_FULL_LOGGING_KEY, false)) {
prefs.getString(ConfigActivity.dfulllog_key,
getString(R.string.defaultdirname)));} }

private void stopLiveData() {
gpsStop();
doUnbindService();
releaseWakeLockIfHeld();
if (writec != null) { writec.closeLogCSVWriter();} }
```

Figure 7 illustrates the parameters provided by the OBD II, which is connected to the machine:

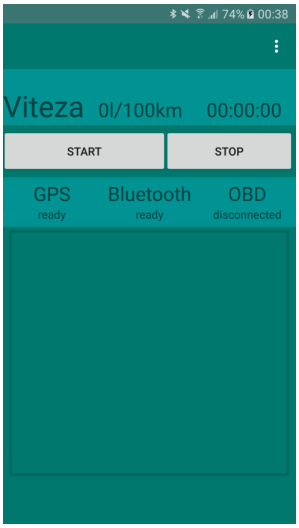

Figure 7. OBD II result

3.2.4 Permissions

To access the different bluetooth, internet, location, and other options, a series of permissions were required that are set in the manifest project file, namely:

```
<uses-permission android:name= "android.permission.BLUETOOTH"/>
<uses-permission android:name= "android.permission.INTERNET" />
```

```
<uses-permission android:name= "android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name= "myapplication.myapplication.MAPS_RECEIVE" />
<uses-permission android:name= "android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name= "android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name= "com.android.vending.BILLING" />
<uses-permission android:name= "android.permission.WRITE_SETTINGS" />
<uses-permission android:name= "android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name= "android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.BLUETOOTH_PRIVILEGED"/>
<uses-permission android:name="android.permission.READ_LOGS" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

4. Types of Pollutant Emissions-Causing Failures

The pollution values change (exceeding set points) depending on several factors such as [10]:

(a) The quality of the fuels and lubricants;
(b) The state of wear of the mechanical components of the engine and its auxiliary installations;
(c) The state of wear of electrical and electronic components;
(d) The running distance on which the vehicle is used daily;
(e) The quality of car maintenance;
(f) The driving quality (skill) of the vehicles.

Among all of these causes, the (d) ÷ (f) problems can be controlled by the vehicle driver, and the detection of an unusual smoke due to lack of his skills will not be a surprise. However, the A Diagnostic application can not solve the problem in this case. For the (a) ÷ (c) situations, when reporting unpredictable gas emissions, the developed application may suggest to the driver the type of fault and its severity. All these proposals, correlated with the remembrance of previous actions on fueling, car maintenance, etc. can accurately identify the malfunction.

(a) Fuel quality directly influences the amount of exhaust gases, namely by increasing the amount of carbon monoxide (CO), using an inappropriate type of oil can increase the value of hydrocarbons (HC).

(b) Increased amount of hydrocarbons (HC) shows that there are unburned components in the exhaust gases. Engine oil may enter the combustion chamber and increases the value of HC. This is possible due to advanced wear of the following components:

- Valve stem oil seal - their purpose is to seal the valves tail; if advanced wear shows, the oil "trickles" on the valves in the combustion chamber where it is disposed on the exhaust system.
- Wear segments of pistons - as in the case of the oil seal rings their wear can lead to penetration of oil into the combustion chamber resulting in massive increase in hydrocarbon and carbon monoxide.

(c) For Homogeneous Charge Spark Ignition (HCSI) vehicles, equipped with spark-ignition engine turbocharged, a wear of the turbine can lead to a significant increase in hydrocarbons. Also a high percentage of HC can influence the catalyst that acts to change the content of chemicals in the exhaust gases, by converting polluting elements (HC, CO and NOx), harmful to the environment, in safe, neutral substances. Chemical transformations in the catalyst are achieved using noble metals: Platinum (Pt), Palladium (Pd) or Rhodium (Rh). For most catalysts the flash point temperature is around 250 ... 300°C. Therefore it is important that the catalyst temperature reaches the optimal functioning value (400 ... 800°C) as soon as possible after starting the engine. Positioning the catalyst closer to the engine, on the exhaust manifold will facilitate its fastest warming. The efficiency of the catalyst is monitored by two Lambda probes, one probe before the catalyst and one probe post catalyst. Depending on the level of oxygen measured by the two lambda probes, computer injection can determine whether the catalyst

is in rated parameters or damaged. The wear of electrical and electronic components (lambda) directly influences the amount of harmful substances and pollutants (CO) in exhaust gases especially at accelerated speed. If one of the lambda probes (oxygen probe) is not working correctly it sends wrong information to the monitoring computer causing a change in the value of the mixture, rich or poor according to the information sent. Figure 8 comparatively illustrates emissions from Opel Vectra car, HCSI type, Euro 2, having faulty lambda probe (left), with replaced lambda probe (right) respectively, both in idle mode and in accelerated speed idling.

(d) Using the car for short distances repeatedly leads to its malfunction and at a high percentage of pollutants in the exhaust gases because it works in a permanent regime of "*shock*" with rich mixture. Figure 9 illustrates a comparison of emissions of an Opel Astra car, HCSI type, Euro 2 at low temperatures (left) or at normal temperatures (right) both in idle mode and at accelerated speed idling. Basically it can be observed that on short distances, the catalyst does not reach its optimal operating temperature and the hydrocarbon emissions at both speeds and that of the CO at accelerated speed are higher.

(e) Not replacing maintenance elements such as spark plugs, can lead to the destruction of the catalyst, by producing a low intensity spark or nor producing it at all, and by the accumulation of unburned gas in the catalyst (gasoline vapors) and its exposure to high temperatures lead to detonation. Burning the air-fuel mixture in the catalyst, may raise its temperature up to 1400 °C resulting its destruction. Air-fuel mixture too rich (shortage of oxygen) caused by not replacing the air filter or too lean (fuel shortage) caused by not replacing fuel filter, influences directly by accumulating unburnt gas in the catalyst, and as in the previous case, leads to its destruction.

(f) Car driving quality and misuse by obstructing the engine operation resulting in the deposition of soot exhaust, turbo does not work in required parameters, generating significant increases in emissions.

Some of these technical features are included in the developed application, others can be extended into next version of developed application.

Figure 8. Emission measurements with defective Lambda probe (left) and with replaced Lambda probe (right) respectively

Figure 9. Emissions measuring at low temperature(left) and regular temperature(right)

5. Conclusions and Further Development

This paper describes our self-developed Android application designed to detect vehicle malfunctions. The application is useful not only because it helps the user to make a brief diagnosis after the symptoms they notice at the car, discovering the reason of their problem, but also, in some more serious problem, it may provides insights related to nearby car services. Connecting the smartphone by Bluetooth to an on board diagnostic devices, allows certain functions to appear on the mobile and collecting some useful information on smartphone memory.

By getting the application up to this level, we are still thinking about designing more and more future improvements. We will try to develop this application in favor of environmental protection. Busy traffic and cars are negative factors, which affect it, and such an application can not only develop the user's knowledge in this domain, but can also reduce the emission of many inert exhaust gases that can cause environmental disaster. By doing this, we will try to create a server through which the user can connect, always providing feedback when it solves a malfunction with one's own application, making the user aware of his contribution to the quality of the environment. With this information we could make some statistics to see how much it can protect the environment. On the other hand, the big change to the application, which we want to do is to make a diagnosis: the phone analyzes the on-board computer, giving accurately the defects of the car, the user being exempt from paying the diagnosis of the car services. We think that the application can help both, the environment and the economy.

Acknowledgements

We would like to thank Mr. Rinea David from SC Lativicom SRL and Dr. Lucian Roman from SC Autohaus Huber SRL Sibiu which has given some relevant insights and technical advices regarding the influence of fuels and vehicles' operating parameters on air pollution.

References

[1] Rînea I.G. (2017). A Diagnostic, *In*: Proceedings of the Ninth International Conference on Applied Informatics Imagination, Creativity, Design, Development (ICDD), May 25-27, Sibiu, Romania.

[2] Gartner, Inc. (2015). *Gartner Says Worldwide Device Shipments to Grow 1.5 Percent, to Reach 2.5 Billion Units in* 2015, Egham, UK, July 6, 2015, *http://www.gartner.com/newsroom/id/3088221*, last accessed 10th April 2017.

[3] Müller, H., Gove, J. L., Webb, J. S., Cheang, A. (2015). *Understanding and Comparing Smartphone and Tablet Use: Insights from a Large-Scale Diary Study*. *In*: Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction (pp. 427-436). ACM, December, 2015.

[4] eMarketer. (2017). *Smartphones Are All the Rage in Germany,* 24 *June* 2014*, https://www.emarketer.com/Article/Smartphones-All-Rage-Germany/1010942* last accessed 10th April 2017.

[5] Riley, D. (2012). *Using Mobile Phone Programming to Teach Java and Advanced Programming to Computer Scientists*, *In*: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, pp. 541-546, North Carolina, USA, 2012.

[6] Android Developer Tools, 2014, *https://developer.android.com/studio/intro/index.html,* accessed 02 March 2017..

[7] Google Developers (GPA services), *https://developers.google.com/places/*, accessed 02 March 2017.

[8] Stack Overflow, *http://stackoverflow.com/questions/26562848/how-to-integrate-google-maps-in-android-app*, accessed 02 March 2017.

[9] Informații OBD. (2017). *Despre interfață și funcționare* (in Romanian), *http://blog.lemberg.co.uk/how-guide-obdii-reader-app-development*, accessed 02 March 2017.

[10] Florea, A., Berntzen, L., Johannessen, M. R., Stoica, D., Naicu, I. S., Cazan, V. (2017). *Low Cost Mobile Embedded System for Air Quality Monitoring: Air quality real-time monitoring in order to preserve citizens' health, In:* The Sixth International Conference on Smart Cities, Systems, Devices and Technologies, SMART 2017, June 25 - 29, 2017- Venice, Italy.