AiroDiag: A Sophisticated Tool that Diagnoses and Updates Vehicles Software Over Air

Karim Mansour[1,2]
Interior Switches and Control, VALEO [1],
Nile University [2], Giza, Egypt
karim.mansour@valeo.com

Wael Farag[3,4], Mohamed ElHelw[2]
Interior Electronics, VALEO [3]
Cairo University[4], Giza, Egypt
wael.farag@valeo.com,
melhelw@nileuniversity.edu.eg

***Abstract*—This paper introduces a novel method for diagnosing embedded systems and updating embedded software installed on the electronics control units of vehicles through the Internet using client and server units. It also presents the communication protocols between the vehicle and the manufacturer for instant fault diagnosis and software update while ensuring security for both parties. AiroDiag ensures maximum vehicle efficiency for the driver and provides the manufacturer with up-to-date vehicle performance data, allowing enhanced future software deployment and minimum loss in case of vehicle recalls.**

Keywords- Vehicle; automotive; diagnostics; bootloader; Internet; communications; failures; security; mobile; wireless; performance; recall; software; update; DTC.

I. INTRODUCTION

Automotive diagnostics are usually done periodically in the garage. The technical team connects the vehicle to a diagnostic tool provided by the car manufacturer to the garage, so the technical team can read the faults in the vehicle and update its software if a new software was available. What about if all of these procedures could be done remotely over the air?

The AiroDiag will act as a background communication channel between the manufacturer and the customer (also referred to in this paper as the vehicle). It will allow the manufacturer to monitor the vehicle performance and advice the customer with the best actions in regards to any current software updates or problems discovered in the vehicle.

The AiroDiag will also be used to collect diagnostic information from the customers that will be utilized as a statistics knowledgebase for the manufacturer to create more reliable vehicles. It will be very useful in the case where there is a recall for a certain type of sold vehicles. The AiroDiag will easily allow the manufacturer to precisely detect the defected vehicles with the recall issue. Please refer to Figure 1 for further demonstration.

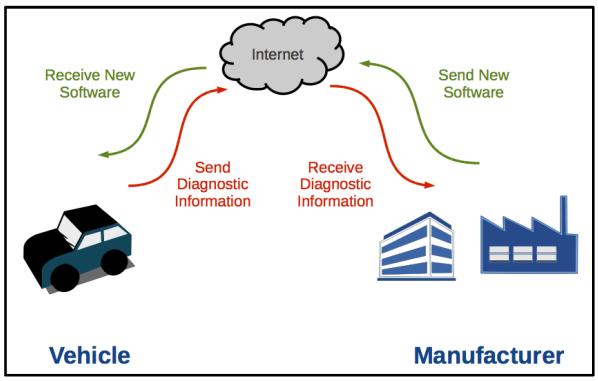

Figure 1. Main Idea: How the AiroDiag System works.

II. RELATED WORK

With the exception of Ford Sync [19], not much work has been carried out in the area of vehicle embedded software diagnosis and update over-the-air. It is a Ford project powered by Microsoft (Microsoft Auto Platform) [20] that synchronizes the whole vehicle data with the manufacturer servers. Ford Sync keeps the vehicle always connected to the world and provides features such as vehicle health report, 911 assist, Internet connectivity, navigation and command center support.

However, the main feature that makes the AiroDiag unique and different from Ford Sync is that Ford Sync lacks the most powerful feature in AiroDiag, which is ECUs software update.

III. ELEMENTS OF AUTOMOTIVE DIAGNOSTIC TOOL

Automotive diagnostic tools work by requesting data or actions from the Electronic Control Unit (ECU). The latter has a built-in diagnostic software component that is designed to communicate with the diagnostic tool by using a common communication protocol. This section describes the basic elements of automotive diagnostic tools.

A. Requested Data

The requested data consists mainly of Diagnostic Trouble Code (DTC) data. It is collective data about failures that occur while the software installed on the ECU is running. While designing the embedded system, it is possible to predict the failures that may occur while the ECU is running due to external forces; for instance, short circuits on connected motors. Defining the possible expected external failures while designing the software will help detect them and protect the

whole ECU from being damaged. Typically the software designer gives each expected failure a unique DTC identifier that is stored in a non-volatile memory and retrieved when requested by the diagnostic tool.

B. *Requested Actions*

Requested actions are those requested from the ECU to perform. For instance, one of the most important actions requested from the ECU is to update the currently installed software. In this case, the diagnostic tool sends a request, accompanied with the new data (software), to the diagnostic software component in the ECU to update its software. Two conditions must be satisfied in order for this request to be executed by the ECU.

First, the Microcontroller Unit (MCU) must be able to write on its program flash memory. Not all MCUs have this capability. Second, the software flashed on the MCU must have a separate software block, called boot-loader that can utilize the first condition. The boot-loader is responsible of booting the system; copying code from the non-volatile memory (flash) to the volatile memory (RAM) and flashing new software to the system, by changing the data in the non-volatile memory with the data it receives via Universal Asynchronous Receiver/Transmitter (UART), Controller Area Network (CAN) bus, or Local Interconnect Network (LIN) bus.

The diagnostic software component is the one that communicates with the MCU boot-loader. Although the boot-loader can flash new software to the ECU, this is not its main function. Christopher Hallinan mentioned in his book "when power is first applied to a processor board, many elements of the hardware must be initialized before even the simplest program can run. Each architecture and processor has a set of predefined actions and configurations, which include fetching some initialization code from an on-board storage device (usually flash memory). This early initialization code is part of the boot-loader and is responsible for breathing life into the processor and related hardware components" [1].

C. *On-Board Diagnostic Protocols*

According to Helmut Frank and Uwe Schmidts; engineers in Vector [21], the developing trend of diagnostic tools is similar to the trend of electronics in the vehicle. Back in the 1990s, automotive OEMs created their own tools in-house. Specialists in different departments customized their tools precisely to meet their requirements and specific applications. This resulted in the production of individual in-house solutions within each OEM and even for different process steps [2].

In 2005 companies and manufacturers started to create the tools that began to support general standards. Such tools offered standardized software layers and interfaces that can be easily integrated into existing tool chains. Standards currently include the diagnostic data model per ISO 22901-1 ODX (Open Diagnostic Data Exchange, ASAM MCD-2D), the hardware interface per ISO 22900-2 (D-PDU API) and the interface between the runtime system and the test application per ISO 22900-3 (ASAM MCD-3D, D-Server API). Each of the programming interfaces mentioned are available to the user as software libraries [2].

Also the diagnostic units are considered in the AUTOSAR (AUTomotive Open System ARchitecture), like in the diagnostic-relevant standards per SAE J2534 and in the context of AUTOSAR standardization AUTOSAR WP4.2.2.1.4 (DCM, DEM). Furthermore, the UDS diagnostic protocol per ISO 14229-1 (Unified Diagnostic Services on CAN) will gradually replace older protocols such as the K-Line per ISO 9141-2 and "KWP2000" as well as "KWP2000 on CAN" [2].

The way the AiroDiag handles diagnostics is independent on what standards the vehicles use. It implements a higher-level layer that utilizes the already existing protocols. This will facilitate the installation of the AiroDiag as a plug-and-play tool.

IV. AIRODIAG

AiroDiag will be used to communicate between the manufacturer and its vehicles. Each vehicle that has the AiroDiag installed will be tagged with a manufacturer ID and an authentication communication key. The key is saved in the database connected to a main server installed in the manufacturer premises. The main server must always be connected to the Internet; ready for any client connection requests (clients here are the vehicles), and must always be updated with the latest ECUs software. All ECUs in the vehicle having the boot-loader and diagnostics software components implemented in their software will be connected to the AiroDiag through the CAN network or serial communication. Figure 2 illustrates this procedure.

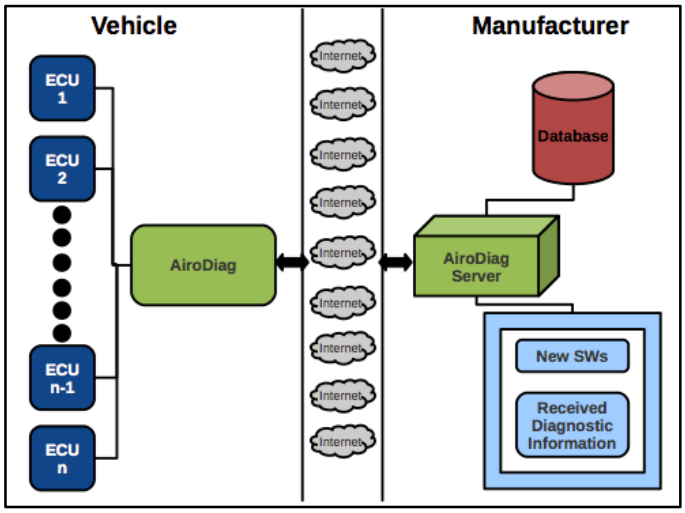

Figure 2. High Level Scope: How the AiroDiag and the AiroDiag Server communicate.

A. *AiroDiag Implementation*

AiroDiag consists of Computer-On-Module (High-end Microcontroller or ARM-Linux based) and is connected to a Universal Mobile Telecommunications System (UMTS) or General Packet Radio Service (GPRS) module. Non-volatile memory (SD card) is connected to the computer-on-module to save all the new HEX files of all the installed ECUs in the vehicle. Please refer to Figure 3 for further demonstration. Even though, AiroDiag is similar to other diagnostic tools

available at the garage of the vehicle manufacturer, it has an extra functionality, which is being mobile. The communication protocols remain the same but with an extra module enabling it to be always connected to the Internet (IP Network).

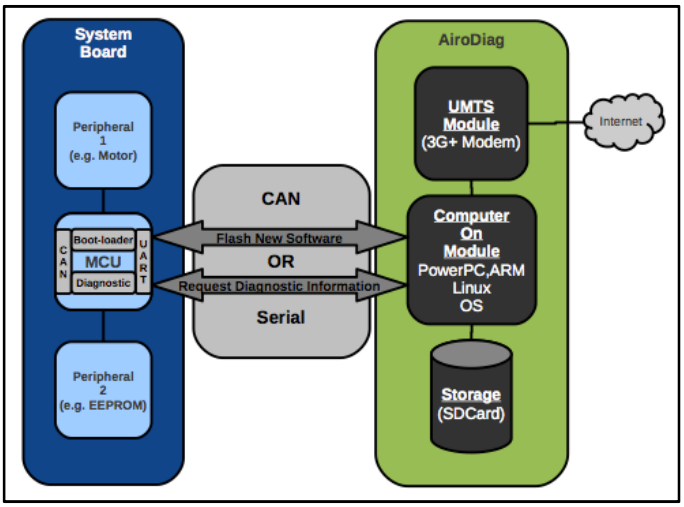

Figure 3. AiroDiag Detailed Components and How it Communicates with ECUs

B. *AiroDiag Communication Protocol*

The update and diagnostics processes could be done when either the engine starts (default) or as the driver configurations states. AiroDiag will connect to the Internet once it achieves power (configurable). As the engine is turned on, the AiroDiag will start connecting to the Internet, and then it will communicate with the static IP server in the manufacturer premises by using the protocol suggested below (please refer to Figure 4 and Figure 5).

In order to protect customer privacy, all functionalities of AiroDiag require the approval of the customer (driver). The vehicle manufacturer server cannot in any way start the connection without the customer's consent. The customer approval could be asked only once, and could be changed later on by changing the vehicle settings.

As the engine starts, the AiroDiag with a dynamic IP address requests a connection from static IP server. The server replies by asking for an authentication key, to ensure that the AiroDiag is a trusted unit. After the AiroDiag verifies itself by sending the authentication key, it should send to the server a list of all the current software versions of all the software installed on the ECUs in the vehicle. The server compares this list with the list saved in its database that concerns the same vehicle type known from the authentication key sent before, and then replies with the available updates. The user (driver) has to verify whether these updates should be installed or not. If the server receives the user's approval, the server will start to send the HEX files of the new software and the AiroDiag saves these files in its non-volatile memory (SD Card). Then it starts to flash them on the ECUs concerned after pausing them from being function. Note that the there are some critical ECUs that cannot be updated while the vehicle is moving (turned on) like the Engine ECU and the Body Controller ECU [3].

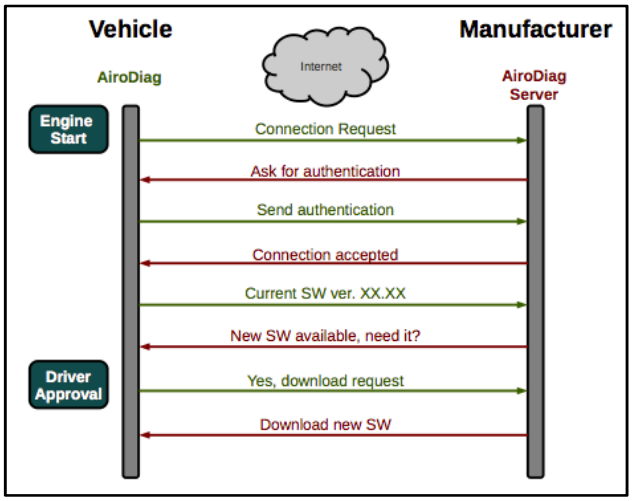

Figure 4. Communication Protocol to Send New Software

The server will ask the AiroDiag to collect DTCs data directly after sending the new HEX files. If it received an approval, it will start reading the DTCs in the vehicle on the fly. The server will make analysis based on the received DTCs and the history of the DTCs received before. The vehicle manufacturer will use the collected data to provide the vehicle owner with the best maintenance options available for the vehicle.

In case the AiroDiag found the server is down when it tries to initiate the connection, the AiroDiag shall behave normally as if it was not installed on the vehicle. It shall check the server periodically and once it is on again, it starts the protocol as explained previously.

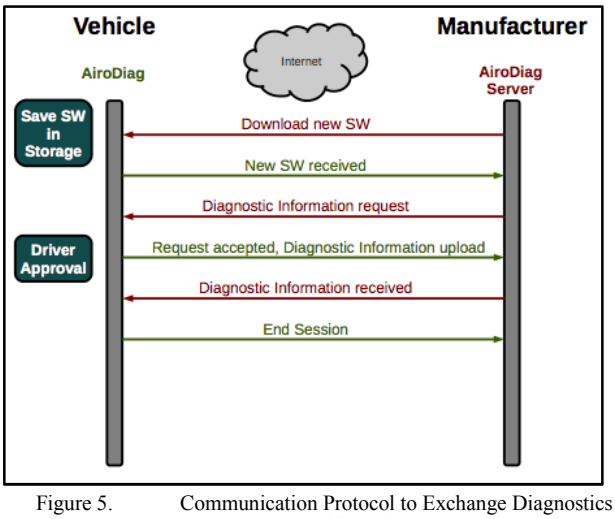

Figure 5. Communication Protocol to Exchange Diagnostics Trouble Codes

V. SECURITY

To protect users' privacy, an encrypted channel has been created between the AiroDiag and the vehicle manufacturer server. In addition to that the system is designed so only the AiroDiag can start the connection. The server only collects data after the user's approval on an encrypted channel that was initially requested by the user.

Two encryption/decryption techniques are commonly used: symmetric and asymmetric. Symmetric encryption is the more traditional form, where both sides agree on a system of encrypting and decrypting messages — the reverse of the encryption algorithm is the decryption algorithm. Modern symmetric encryption algorithms are generic and use a key to vary the algorithm. Thus, two sides can settle on a specific key to use for their communications [4]. The problem with secret keys is that anyone who knows the secret key can decrypt the message. Asymmetric encryption solves this problem by providing two related keys (a key pair). A public key that is available to anyone who wants to send a message. A second, private key is kept secret, so that only the source knows it.

VI. AIRODIAG SOFTWARE AND HARDWARE ARCHITECTURES

A full system has been implemented to illustrate the idea of AiroDiag: Vehicle ECU with a boot-loader installed, AiroDiag Client board, AiroDiag Server and Web-AiroDiag Dashboard.

A. Vehicle ECU

A demonstration (DEMO) vehicle ECU has been implemented using an Arduino Duemilanove board [6][7] with a 16X2 Liquid Crystal Display (LCD) screen and two switches attached to it. See Figure 6.

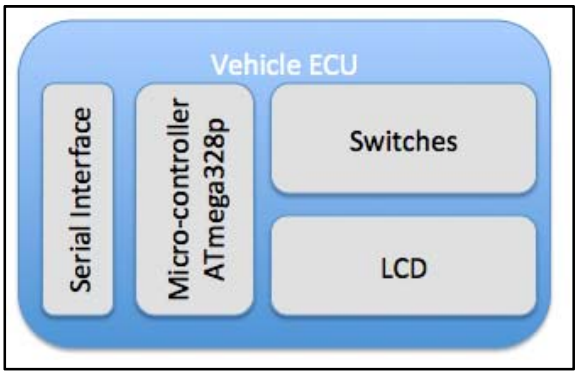

Figure 6. Vehicle ECU Hardware Architecture

Arduino boot-loader [11] has been installed on the Atmel AVR ATmega328p micro-controller [7] built-in in the Arduino Duemilanove Board. Also DEMO software has been created for the vehicle ECU to respond to the AiroDiag Client requests regarding DTCs collection or providing the current installed software version through the Diagnostics Software Component. In order to clarify what software version is installed on the vehicle ECU, the DEMO software is periodically displaying the software version of the ECU on the LCD attached every 50 milliseconds through the Application Software Component. Please refer to Figure 7.

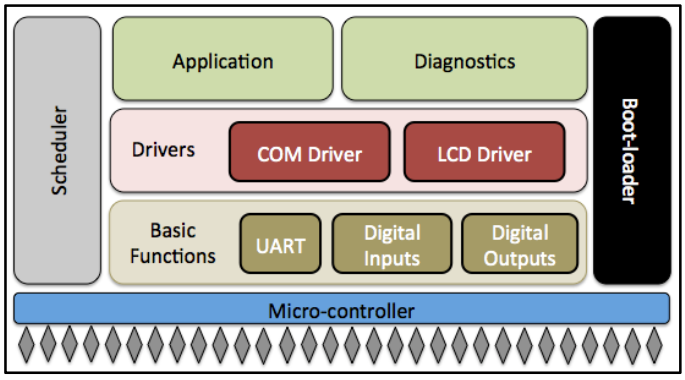

Figure 7. Vehicle ECU Software Architecture

All the communications between the vehicle ECU and the AiroDiag Client are done over UART through simple protocol designed specifically for the AiroDiag that can be easily replaced by the LIN protocol that uses the Communication (COM) software driver to access the board serial port.

B. AiroDiag Client

The AiroDiag Client has been implemented on a Beagle Board (Version xM, Revision B) with ARM-Cortex-A8 1GHz, 512MB RAM and 2GB Memory Card. It has four Universal Serial Bus (USB) ports; in which two of them have been used to connect the AiroDiag to the HUAWEI E153 3G Modem and the vehicle ECU that has a USB interface as shown on Figure 8.

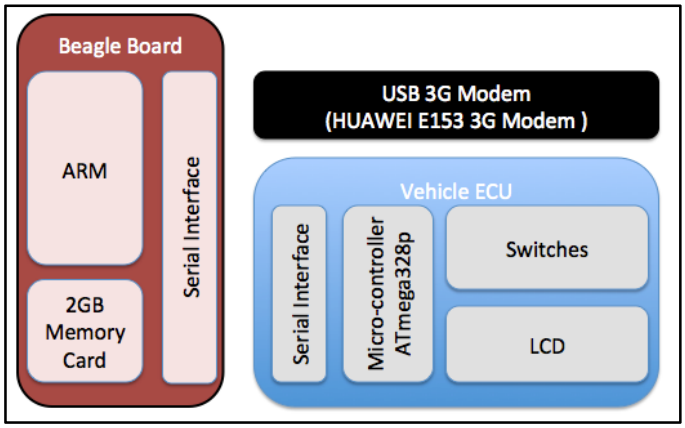

Figure 8. AiroDiag Client Hardware Architecture

To be able to handle all the peripherals of the Beagle Board, Ubuntu LINUX kernel is installed for the ARM provided by it. One of the major advantages of Linux is the array of open source applications (web servers, ftp, telnet, ntp, ssl, sql, email, and the list goes on)[12].

Open source software named AVRDUDE has been installed over the LINUX kernel to be able to flash new embedded software on the Vehicle ECU through the boot-loader installed on it. AVRDUDE is a program for downloading and uploading the on-chip memories of Atmel's AVR microcontrollers. It can program the Flash and EEPROM, and where supported by the serial programming protocol, it can program fuse and lock bits. AVRDUDE also supplies a direct instruction mode allowing

one to issue any programming instruction to the AVR chip regardless of whether AVRDUDE implements that specific feature of a particular chip [13].

Python [14] has been downloaded over the LINUX kernel to abstract the kernel drivers with easy and stable APIs. Python abstraction layer has been used to create a flashing driver that communicates with the AVRDUDE, serial driver to communicate with the vehicle ECU and a secured network driver to communicate with AiroDiag Server.

All the data that is being communicated through the network driver is being encrypted and decrypted using the Advanced Encrypted Standard (AES). The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called ciphertext; decrypting the ciphertext converts the data back into its original form, called plaintext [15]. This has been done through the python library named M2Crypto [16].

Finally a software layer called AiroDiag Client Manager has been designed to handle and manage all the software installed on the Beagle Board. The AiroDiag Client Manager is a python script that is initiated through the AiroDiag Booting Driver, which is automatically run when the Beagle Board is turned on. For more illustration on how all the AiroDiag Client Software Components are communicating with each other please refer to Figure 9.

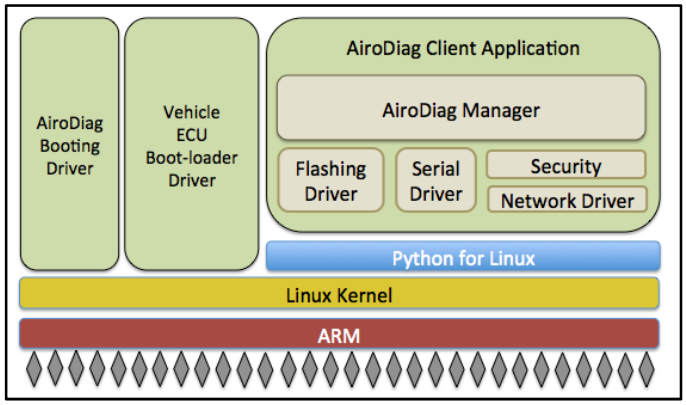

Figure 9. AiroDiag Client Software Architecture

C. AiroDiag Server

The AiroDiag Server is simply implemented on remote server that has Xeon L5420 CPUs (quad-core, 2.5 GHz) [8] and two 1TB hard drives which is connected to the Internet with a static IP address. It has Ubuntu 10.10 LINUX installed on it. MySQL [18] has been installed on the LINUX kernel to act as a database to save the data requested from AiroDiag Client (DTCs) and to save the new software versions (Executable files; HEX files) that shall be transmitted and downloaded to the vehicle. And to manage the whole server, python script (AiroDiag Server Manager) has been designed an implemented. Please refer to Figure 10.

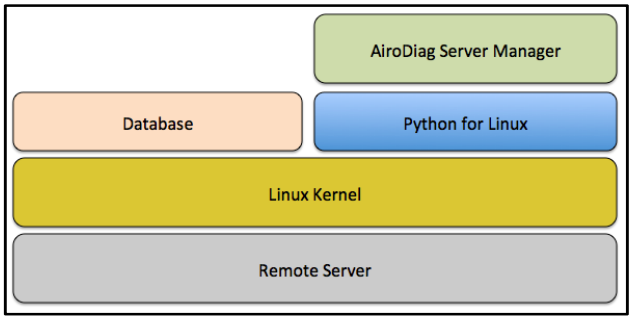

Figure 10. AiroDiag Server Software Architecture

D. Web-AiroDiag Dashboard

The Web-AiroDiag Dashboard is an online dashboard that monitors all the AiroDiag systems implemented. It demonstrates the idea of how can the manufacturer keep eye on what software version is installed on a current vehicle, what is the last software version available in the manufacture and what is the performance of all the vehicles that has the AiroDiag system installed. A DEMO version of the Web-AiroDiag Dashboard is available on [17], for demonstration purposes only. The Web-AiroDiag Dashboard gets all of its data from the MySQL database installed on the AiroDiag Server. See Figure 11.

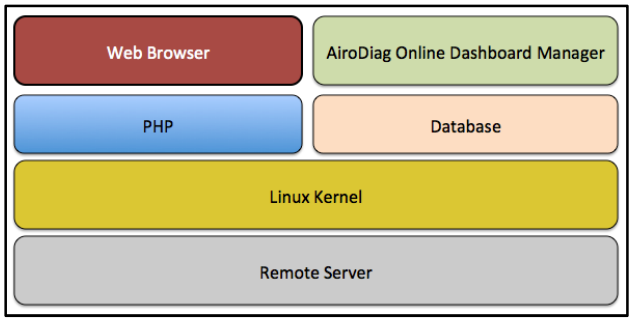

Figure 11. Web-AiroDiag Dashboard Software Architecture

VII. EXPERIEMNTAL RESULTS

A. Response Time

The following chart in Figure 12 illustrates the response time of the AiroDiag system. It shows how much time it takes for the server to respond to a client request and answers it with a feedback. This test has been done over a two bytes frame as a transporter for the command and the response.

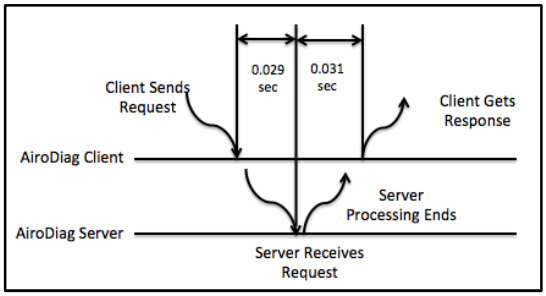

Figure 12. AiroDiag Response Time

B. Encryption and Decryption Time

The timings achieved below in Figure 13 are obtained from the AiroDiag Client, since its execution time is much slower than the server. As it shown timings could be considered acceptable since most of the executable files are in the range shown in the figure.

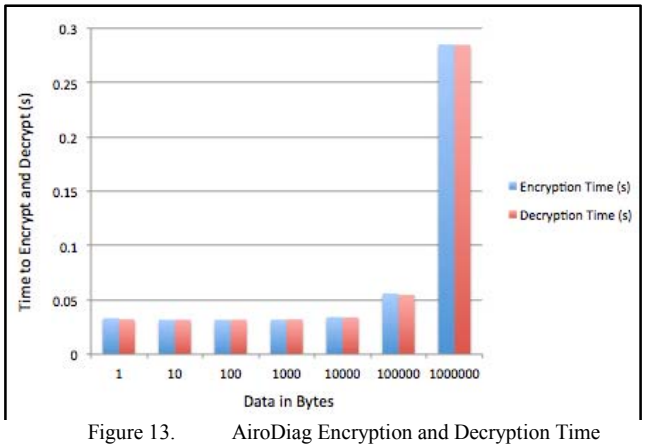

Figure 13. AiroDiag Encryption and Decryption Time

C. Sending Data Time

Sending Data Time considers both, frames (commands and responses) and executable (HEX) files sent from the AiroDiag Server to the AiroDiag Client. Please refer to Figure 14.

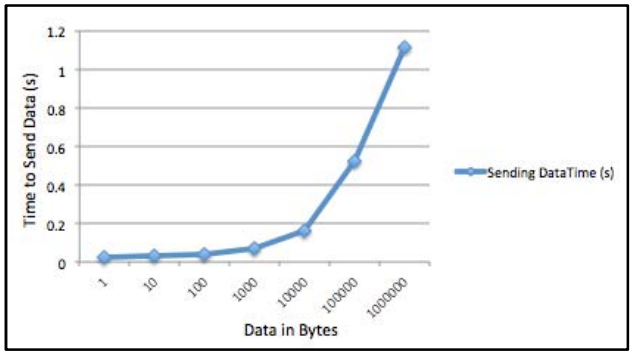

Figure 14. AiroDiag Sending Data Time

D. Flashing Time

Table I shows how much the AiroDiag Client takes to flash new software. These values have been achieved while the AiroDiag was communicating with the vehicle ECU through UART. These results will differ in case the communication is done over CAN.

TABLE I. AIRODDIAG FLASHING TIME

Flashed Data Size	*Flashing Time*
4399 bytes	5 seconds
5554 bytes	5.5 seconds
6054 bytes	6.5 seconds
7161 bytes	7 seconds

E. Booting Time

Booting Time is the time needed for the AiroDiag Client to start the first communication frame with the AiroDiag Server once the vehicle engine has been started. Figure 15 shows the average time for the AiroDiag Client to boot.

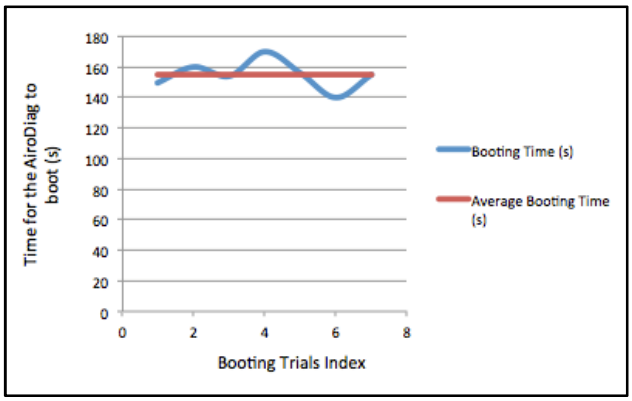

Figure 15. AiroDiag Booting Time

VIII. ISSUES AND RISKS

A. Issues

As illustrated in the results, the AiroDiag Client takes a lot of time to boot (Average 155 seconds). This is due to Ubuntu 10.10 installed on the Beagle Board; it has a lot of functionalities that is not needed for the AiroDiag. Also the LINUX kernel is initializing many unused peripherals, which leads to a long boot time.

B. Risks

Due to the fact the AiroDiag is on the Internet Network, this creates a risk that the communication between the AiroDiag Client and Server maybe hacked, even after the encryption process, this part is not guaranteed.

IX. FUTURE WORK: AIRODIAG 2.0

AiroDiag 2.0 could be considered as the future work for the AiroDiag to overcome the issues and risks mentioned above and to enhance the usability of it through real vehicles.

First, LINUX will be replaced with a real-time Operating System that is customized specifically for the AiroDiag not a generic one as the Ubuntu. It will be installed on ARM or PowerPC, the matter still under research. This will solve the problem of the booting time. Connecting to the Internet through the UMTS module will be the longest time in the booting.

Secondly, The communication between the Vehicle ECU and the AiroDiag Client will be on CAN rather than UART or LIN. This will ease the integration of the AiroDiag system to any vehicle.

Finally, more research will be done on how to increase the security of the channel between the AiroDiag Client and Server.

X. Return on Investements

What currently happens is that the vehicle owner goes to the garage periodically each 10,000 km approximately to check the vehicle performance and fix any failures in it. The technician performs a full check-up on the vehicle with the fixed diagnostic tool provided by the manufacturer. If there is a problem found, the technician starts to fix it or replace the defected part. After this takes place, the technician changes the ECUs software if there were new non-critical software updates.

Experimentally it takes around 10 minutes average to read all the DTCs in the vehicle, and around another 30 minutes to analyze and fix the found failures. If it is assumed that the garage will only fix one vehicle at a time. Then it will take 34.7 working years to fix 100,000 vehicles, assuming one technician working.

If the AiroDiag is installed, the user will not pay a visit to the garage except if there was a problem that requires the garage technician to interfere. Some problems could be solved by adjusting some calibrations from the server or by updating the software on the fly. Also the garage technician will not spend time reading DTCs nor analyzing them since this work has been already done by the vehicle manufacturer. Refer to Table II below to see results and comparison between current diagnostics and diagnostics after the AiroDiag System has been installed.

TABLE II. Comparisons

#	**Comparing diagnostics approaches based on 100,000 vehicles**		
	Situation	***Garage***	***AiroDiag***
1	Read DTCs	1000000 minutes	10 minutes
2	Analyze Failures	1000000 minutes	05 minutes
3	Flash new software to the entire vehicle ECUs	6000000 minutes	60 minutes
5	Cost of RECALL	High	Low

The AiroDiag opens plenty of business doors for the vehicles' manufacturers. It builds a channel of trust between the customer and the manufacturer, as the customer will always be instantly advised with the best solution for any failures in the vehicle. In addition to that, the customer may not experience the problem if it can be directly fixed from the server. In particular if the problem was from a software bug that has been fixed by updating the ECU software.

Furthermore connecting the vehicle with the Internet will give the manufacturer the opportunity to offer services through a lot of useful applications such as locating or controlling the vehicle remotely from a mobile device. These services could be exchanged for annual fees or free of charge to distinguish the manufacturer from its competitors.

XI. Conclusion

Installing the AiroDiag in the vehicles will not only update the vehicle's ECUs software in case of bugs have been discovered or periodically send the vehicle's stored failures to ensure better vehicle maintenance and improved future software designs, but it will keep the vehicle constantly connected with the whole world through the IP network (Internet). The AiroDiag can act as the basic platform that will lead the automotive industry to enter the ubiquitous networking era, where everything is connected; the cars, traffic signs, buildings, phones, etc. –all connected to each other.

As it has been illustrated, the design of the AiroDiag system (unit + server) is simple and its implementation is not costly, especially after the IP version 6 has been out, allowing a lot of devices to enter the Internet world. So, why not the automotive industry be one of the first industries that can benefit from this technology [9][10].

References

[1] Christopher Hallinan, Embedded Linux Primer: A Practical Real-World Approach, 2nd ed., Prentice Hall, 2010, pp.157–188.

[2] Helmut Frank, Uwe Schmidts, "Vehicle diagnostics – the whole story," Vector, Vector Press.

[3] Automotive Centeral Body Controller Guide, Texas Instruments, Visited October 2011, <http://www.ti.com/lit/sg/slyb143/slyb143.pdf>

[4] Jawwad Ahmad, Ben Garrison, Jim Gruen, Chris Kelly, and Hunter Pankey, "4G Wireless Systems," 2003.

[5] BeagleBoard-xM System Reference Manual, Revision C.1.0, April 4, 2010.

[6] The Arduino Duemilanove Schematic, Revision 1.0, Published 2009, Visited 2011 <www.arduino.cc>

[7] ATmega328p Datasheet, AVR Family, Atmel, Visited October 2011 <www.atmel.com>

[8] Xeon L5420 CPU, Intel Processors, Intel, Visited October 2011 <www.intel.com>

[9] Gyu Myoung Lee, Jun Kyun Choi, and Taesoo Chung, Doug Montgomery, "Standardization for ubiuitous networking in IPv6-vases NGN", ITU-Y Kaleidoscope Event – Innovations in NGN, May 2008.

[10] Gyu Myoung Lee, Jun Kyun Choi, Noel Crespi, "Object Identification for Ubiquitous Networking", ICACT 2009, February 2009.

[11] Arduino Bootloader, Arduino Development Environment, Visited February 2012 <www.arduino.cc>

[12] Bob Japenga, Why Use Linux for Real-Time Embedded Systems Revision A, pp.5

[13] Avrdude Documentation, AVRDUDE, Version 5.11, Published August 2011, Visisted February 2012 <http://www.nongnu.org/avrdude>

[14] Python, Programming Language, Visited February 2012 <www.python.org>

[15] Advanced Encryption Standard, Federal Information Processing Standards Publication 197, Published Novemver 26, 2011.

[16] M2Crypto, M2Crypto Documentation, Visited February 2012 , http://chandlerproject.org/bin/view/Projects/MeTooCrypto>

[17] Web-AiroDiag Dashboard, AiroDiag Project, Visited February 2012 <www.kmftech.com/airodiag>

[18] MySQL Documentation, MySQL, Oracle, Visited February 2012 <http://dev.mysql.com/doc>

[19] Ford Sync, Ford Project, Ford, Visited February 2012 <www.ford.com/technology/sync>

[20] Microsoft Auto Plateform Overview, Version 3.1, Published November 2008, Visited February 2012 <http://www.microsoft.com/auto>

[21] Vector, Tooling Compnay, Visited February 2012 <www.vector.com>