ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Υλοποίηση Εφαρμογής Διαγνωστικών Οχήματος σε Πλατφόρμα Android

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΗΛΙΑΣ Α. ΚΟΤΣΑΜΠΟΥΓΙΟΥΚΟΓΛΟΥ

Επιβλέπων : Ευστάθιος Συκάς
Καθηγητής

Αθήνα, Μάρτιος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Υλοποίηση Εφαρμογής Διαγνωστικών Οχήματος σε Πλατφόρμα Android

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΗΛΙΑΣ Α. ΚΟΤΣΑΜΠΟΥΓΙΟΥΚΟΓΛΟΥ

Επιβλέπων : Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την …….

.....	.....	.....
Ευστάθιος Συκάς	Μιχαήλ Θεολόγου	Γεώργιος Στασινόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2016

.....
Ηλίας Α. Κοτσαμπουγιούκογλου

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η υλοποίηση εφαρμογής για επικοινωνία με το σύστημα διάγνωσης εντός οχήματος (OBD), το οποίο όλα τα οχήματα διαθέτουν.

Στα πλαίσια αυτής της εργασίας έγινε χρήση των γνώσεων μας για τις έξυπνες συσκευές καθώς και για το σύστημα διάγνωσης εντός οχήματος (OBD). Στόχος μας ήταν η υλοποίηση μιας εφαρμογής σε περιβάλλον Android, η οποία θέλαμε να μπορεί να λαμβάνει δεδομένα σχετικά με το όχημα για τις μεταβλητές που έχουμε ορίσει. Η επικοινωνία για την ανταλλαγή δεδομένων επιτυγχάνεται μέσω Bluetooth.

Στέλνονται αιτήσεις για πέντε διαφορετικές μεταβλητές (ταχύτητα οχήματος, στροφές κινητήρα, θερμοκρασία μηχανής, κατανάλωση καυσίμου και σχετική θέση του πεντάλ του γκαζιού) και λαμβάνονται οι απαντήσεις για αυτές τις αιτήσεις, οι οποίες παρουσιάζονται στη διεπαφή χρήστη (user interface) της εφαρμογής.

Εν κατακλείδι, ως αποτέλεσμα αυτής της εφαρμογής ανταλλάσσουμε σειριακά δεδομένα μέσω Bluetooth σε πραγματικό χρόνο με το σύστημα διάγνωσης εντός οχήματος. Αυτά τα δεδομένα μπορούν να χρησιμοποιηθούν είτε από ιδιώτες, είτε από μηχανικούς για επισκευή των οχημάτων ή από εταιρίες.

Λέξεις Κλειδιά: << έξυπνες συσκευές, διαγνωστικό σύστημα εντός οχήματος, Bluetooth, Android, εφαρμογές >>

Abstract

The purpose of this thesis is the implementation of an application for on-board diagnostics (OBD).

During this work was made use of our knowledge for smartphones and for on-board diagnostics. Our goal was to implement an application in Android environment that will communicate with an elm327 device. This device is connected to an OBD socket that each vehicle has. Then the application, which is on a smartphone with Android OS communicates with the elm327 device and it exchanges serial data via Bluetooth.

Once the application makes the connection and the channel is generated it sends requests for five different variables (vehicle speed, engine speed, engine temperature, and engine fuel rate and relative accelerator pedal position) and it receives answers to these requests. Finally these values are presented in the user interface.

In conclusion, through this application we have the ability to receive real-time serial data via Bluetooth about on-board diagnostics. These data can be used either by individuals or by engineers for repair of vehicles or companies.

Keywords: << smartphones, on-board diagnostics, Bluetooth communication, Android, applications >>

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα καταρχάς να ευχαριστήσω τον υπεύθυνο καθηγητή της διπλωματικής μου εργασίας, κύριο Ευστάθιο Συκά, για την εμπιστοσύνη που μου έδειξε δίνοντας μου το εν λόγω θέμα. Στη συνέχεια ευχαριστώ τον υποψήφιο διδάκτορα κ. Ασθενόπουλο Βασίλειο για το χρόνο που μου αφιέρωσε καθώς και για τις συζητήσεις-συναντήσεις που είχαμε πάνω στο θέμα της διπλωματικής. Εν τέλει ευχαριστώ την οικογένεια μου και τους φίλους μου που με στήριξαν κατά τη διάρκεια όλης αυτής της προσπάθειας για την υλοποίηση αυτής της εφαρμογής.

Αφιερώνεται στους γονείς μου…

Πίνακας Περιεχομένων

Πίνακας Σχημάτων

1

Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιάσουμε την εξέλιξη των έξυπνων κινητών και θα αναλύσουμε κάποια γενικά στοιχεία για τις έξυπνες συσκευές. Στη συνέχεια θα αναφέρουμε τις τάσεις της τεχνολογίας και προς ποια κατεύθυνση φαίνεται να υπάρχει ανάπτυξη στον τομέα της κινητής τηλεφωνίας. Στο δεύτερο κομμάτι αυτού του κεφαλαίου θα κάνουμε μια ιστορική αναδρομή στα διαγνωστικά συστήματα εντός οχήματος και θα αναφερθούμε στις εξελίξεις γύρω από αυτά.

1.1 Η εξέλιξη των έξυπνων συσκευών (smartphones)

Έξυπνη συσκευή ή αλλιώς smartphone είναι ένα κινητό τηλέφωνο με ένα αναβαθμισμένο λειτουργικό σύστημα, το οποίο συνδυάζει τα χαρακτηριστικά ενός λειτουργικού συστήματος ενός προσωπικού υπολογιστή με άλλα χαρακτηριστικά χρήσιμα για κινητά ή για φορητή χρήση. Τυπικά, τα κινητά συνδυάζουν τις ιδιότητες των τηλεφώνων με αυτές άλλων κινητών συσκευών όπως είναι το PDA, το media player και η πλοήγηση με GPS. Πλέον οι περισσότερες έξυπνες συσκευές μπορούν να έχουν πρόσβαση στο Ίντερνετ, έχουν οθόνη αφής και κάμερα. Επίσης, από το 2012 παράγονται έξυπνες συσκευές που έχουν τη δυνατότητα για πρόσβαση σε 4G LTE internet.

Εικόνα 1: Η εξέλιξη των smartphones

1.1.1 Η αρχή των έξυπνων τηλεφώνων

Η πρώτη ιδέα ενός έξυπνου τηλεφώνου λέγεται ότι είχε συλληφθεί στα μέσα της δεκαετίας του 1970, αλλά η ιδέα δεν καρποφόρησε μέχρι σχεδόν είκοσι χρόνια αργότερα, όταν η IBM ανακοίνωσε το Simon Personal Communicator το 1992. Το Simon είχε μια μονόχρωμη οθόνη αφής, μια γραφίδα, και μία βάση φόρτισης. Παρά το γεγονός ότι ήταν σε θέση να στείλει και να λάβει μηνύματα ηλεκτρονικού ταχυδρομείου και φαξ, το Simon τεχνικά δεν ονομαζόταν έξυπνη συσκευή, αν και ουσιαστικά αυτό ακριβώς ήταν.

Το 1996 η Nokia μπήκε στον χώρο των έξυπνων συσκευών, ο οποίος δεν ήταν ακόμα και τόσο αναπτυγμένος, και κυκλοφόρησε το τηλέφωνο Nokia 9000 Communicator στον κόσμο, το οποίο είχε πληκτρολόγιο κλήσης, πλήκτρα πλοήγησης και μονοχρωματική οθόνη.

Εικόνα 2: Simon

Εικόνα 3: Nokia 9000 Communicator

Το Nokia 9000 Communicator (επίσης δεν αποκαλούταν επισήμως smartphone εκείνη την εποχή) ήταν σε θέση να χρησιμοποιήσει ηλεκτρονικό ταχυδρομείο, φαξ, περιήγηση στον Ιστό (ένα χαρακτηριστικό που έλειπε από το Simon), επεξεργασία κειμένου και λογιστικά φύλλα.

Ο όρος «smartphone» δεν επινοήθηκε παρά ένα χρόνο αργότερα, όταν η Ericsson κυκλοφόρησε το GS 88, αλλιώς γνωστό ως «Penelope». Η εξωτερική του εμφάνιση και ο σχεδιασμός του ήταν εντυπωσιακά όμοια με εκείνη του Nokia 9000 Communicator και περιλάμβανε μια οθόνη αφής στο εσωτερικό του και μια γραφίδα. Σε αυτό το διάστημα, τα έξυπνα τηλέφωνα ήταν ακόμα σε μεγάλο βαθμό ασυνήθιστα στην αγορά μαζικής κατανάλωσης.

Εικόνα 4: Ericsson GS 88

1.1.2 Από τις επιχειρήσεις προς τους καταναλωτές

Η Ιαπωνία έγινε η πρώτη αγορά κινητής τηλεφωνίας για να διαδώσει τα έξυπνα κινητά, τη στιγμή που ο υπόλοιπος κόσμος ήταν ακόμα επικεντρωμένος στο βασικό απλό κινητό τηλέφωνο επικοινωνίας.

Η πρώτη συσκευή από την Research In Motion (αργότερα ονομάστηκε BlackBerry) ήταν το BlackBerry 850, που περιλάμβανε αμφίδρομη τηλεειδοποίηση και κυκλοφόρησε το 1999 και το ακολούθησαν διάφορα μοντέλα, όπως το παγκοσμίως δημοφιλές μονόχρωμο 6200 και το έγχρωμο 7200. Επικεντρώθηκε σε μεγάλο βαθμό στο ηλεκτρονικό ταχυδρομείο και στα ευρύχωρα πληκτρολόγια, και για αυτό το λόγο το BlackBerry έγινε γνωστό εμπορικό σήμα στην αγορά των επιχειρήσεων.

Στις αρχές της δεκαετίας του 2000 τα λειτουργικά συστήματα Symbian, BlackBerry OS, Palm OS και Windows Mobile (γνωστό τότε ως PocketPC) γίνονταν όλο και πιο δημοφιλή για την κινητή τηλεφωνία. Με δυνατότητες, όπως το ηλεκτρονικό ταχυδρομείο, το φαξ και την περιήγηση στον Ιστό, έκαναν τα έξυπνα κινητά τηλέφωνα όλο και πιο

χρήσιμα και απαραίτητα. Μεταξύ του 2000 και του 2006, οι κατασκευαστές κινητών τηλεφώνων προκειμένου να επωφεληθούν από την νέα κινητή πραγματικότητα διερευνούσαν δεκάδες διαφορετικούς παράγοντες που μπορούσαν να κάνουν τη διαφορά και να αυξήσουν τη ζήτηση των έξυπνων συσκευών. Με αυτό τον τρόπο άρχισαν να κυκλοφορούν συσκευές με συρόμενα και ανάστροφα πληκτρολόγια, με περιστρεφόμενες οθόνες, με κάθετα πληκτρολόγια, ακόμη και με πολλαπλά πληκτρολόγια. Οθόνες αφής αναπτύχθηκαν επίσης εκείνο το διάστημα και βρήκαν χρήση στα έξυπνα τηλέφωνα. Βέβαια, υπήρχε και η γραφίδα τότε, επειδή αυτά τα πρώτα έξυπνα κινητά τηλέφωνα δεν ήταν τόσο εύχρηστα ακόμα για την αφή.

Όλα αυτά όμως άλλαξαν, όταν η Apple ανακοίνωσε το iPhone τον Ιανουάριο του 2007 και το εισήγαγε στην αγορά λίγους μήνες αργότερα, τον Ιούνιο. Η επαναστατική ιδέα της Apple για τις έξυπνες συσκευές σε συνδυασμό με τις ισχυρές λειτουργίες πολυμέσων, το ηλεκτρονικό ταχυδρομείο και την περιήγηση στον παγκόσμιο Ιστό, της έδωσαν μεγάλη ώθηση. Το iPhone είχε μια μεγάλη έγχρωμη οθόνη με ψηφιακοποιητή (digitizer) και διεπαφή χρήστη που ήταν επιτέλους φιλική για την αφή (finger-friendly). Και σε αντίθεση με τα άλλα τηλέφωνα, το iPhone είχε μόνο ένα κουμπί στην μπροστινή πλευρά του - το πλήκτρο Αρχική Σελίδα (Home Page) - και τρία γύρω από τα άκρα του, δύο για την ένταση του ήχου (πάνω/κάτω) και ένα για την ενεργοποίηση/αναμονή του (power/standby). Το κινητό λειτουργικό σύστημα της Apple, iOS, ήταν ακόμα σε πολύ αρχικό στάδιο εκείνη την εποχή, αλλά ήταν η πρώτη γεύση από τα νέα, σύγχρονα λειτουργικά συστήματα των έξυπνων κινητών τηλεφώνων που γνωρίζουμε σήμερα.

Εικόνα 5: iPhone

Μετά την εισαγωγή του iPhone, πολλές άλλες συσκευές με λειτουργικά συστήματα Windows Mobile και BlackBerry ξεκίνησαν και εμφανίστηκαν στην αγορά. Τα HTC Touch και Touch Pro ήταν μια προσπάθεια της HTC προς τον καταναλωτή που ήταν

εξοικειωμένος με Windows Mobile. Ομοίως, το BlackBerry Bold 9000 ήταν μια πιο σύγχρονη εκδοχή-πρόταση για το BlackBerry.

Τον Σεπτέμβριο του 2008, η Google απαντά στο iOS με το δικό της κινητό λειτουργικό σύστημα, το Android OS. Η συνέταιρός της HTC κατασκεύασε το πρώτο έξυπνο κινητό τηλέφωνο με λειτουργικό σύστημα Android για την T-Mobile, το G1, επίσης γνωστό ως «Dream».

Εικόνα 6: T-Mobile G1

Στη συνέχεια το iPhone της Apple ανανεώθηκε με ετήσια αναβάθμιση του υλικού του (hardware), όπως συνέβη συγχρόνως όμως και με τους κύριους ανταγωνιστές της. Το iPhone 3GS με λειτουργικό σύστημα iOS, τα HTC Touch Pro2, HD2 και Samsung Omnia II με λειτουργικό σύστημα Windows Mobile και τα HTC Hero, Motorola CLIQ και Samsung Galaxy με λειτουργικό σύστημα Android κατέφτασαν στην αγορά το 2009.

Εικόνα 7: Samsung Galaxy

Εκείνη την εποχή, υπήρχαν επτά σημαντικά λειτουργικά συστήματα κινητής: Symbian, BlackBerry OS, Palm OS, Windows Mobile, webOS, iOS και Android. Μέχρι το 2012, το Symbian είχε εξαφανιστεί, τα Palm OS και Windows Mobile ήταν «πεθαμένα» και αντικαταστάθηκαν από τα webOS και Windows Phone, το BlackBerry έχασε μεγάλο μερίδιο της αγοράς και τα Android και iOS άρχισαν να κυβερνούν το χώρο των έξυπνων κινητών τηλεφώνων.

1.1.3 Σήμερα και το μέλλον

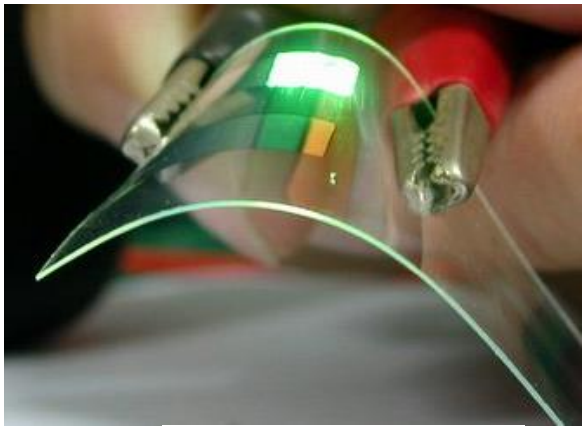
Οι περισσότερες από τις έξυπνες συσκευές στην αγορά σήμερα μοιάζουν μεταξύ τους και οι πραγματικές τους διαφορές είναι λίγες. Αυτό συμβαίνει επειδή πολύ λίγοι κατασκευαστές πειραματίζονται με τη μορφή τους και περισσότερο γίνονται μικρές αλλαγές εσωτερικά, με όλο και πιο εντυπωσιακά χαρακτηριστικά, εστιάζοντας στη βελτίωση των επιμέρους συστατικών για να επιτευχθεί ένα πιο δελεαστικό σύνολο προς το κοινό. Οι οθόνες τους τώρα ξεπερνάνε την ανάλυση 1080p και οι κάμερες έχουν πολλή καλή ανάλυση και εστίαση ενώ ο χώρος αποθήκευσης αυξάνεται σταδιακά. Η επεξεργαστική ισχύς ανεβαίνει με γοργούς ρυθμούς προς τα πάνω και πλησιάζει αυτή που έχουμε σε κονσόλες παιχνιδιών και προσωπικούς υπολογιστές. Γενικά χαρακτηριστικά όπως η ποιότητα των ηχείων, η διάρκεια ζωής της μπαταρίας, η ποιότητα κατασκευής και σχεδιασμού καθώς και ο αποθηκευτικός χώρος βρίσκονται υπό συνεχή έλεγχο.

Τι να περιμένουμε να δούμε στην ανάπτυξη των έξυπνων συσκευών στο εγγύς μέλλον, στα επόμενα πέντε με δέκα χρόνια; Αυτό είναι ίσως δύσκολο να προβλεφθεί με ακρίβεια, λόγω του γρήγορου ρυθμού που έχουν οι εξελίξεις στην τεχνολογία. Παρακάτω όμως παρουσιάζονται τέσσερις τάσεις που φαίνεται να υπάρχουν αυτή τη στιγμή.

Ο όρος «επαυξημένη πραγματικότητα» ή AR όταν χρησιμοποιείται στο πλαίσιο της τεχνολογίας των υπολογιστών αναφέρεται σε αυτό που αντιλαμβανόμαστε μέσα από τις αισθήσεις μας (συνήθως όραση) και ενισχύεται με τη χρήση των αισθητηριακών πληροφοριών από υπολογιστή, όπως είναι ο ήχος, το βίντεο, τα γραφικά και τα δεδομένα GPS. Με απλά λόγια, η AR με το συνδυασμό των δεδομένων του υπολογιστή καθιστά περισσότερες πληροφορίες διαθέσιμες για τους χρήστες σε αυτό που βλέπουν σε πραγματικό χρόνο στη ζωή τους. Αυτή η τάση αρχίζει να υπάρχει και στις έξυπνες συσκευές. Για παράδειγμα, χρησιμοποιώντας την κάμερα του τηλεφώνου, μπορούμε να πάρουμε πληροφορίες επικάλυψης σε ζωντανό χρόνο (real-time) από τις έχουμε τη δυνατότητα μπορούμε να βρούμε τους πλησιέστερους χώρους εστίασης.

Εικόνα 8: Με χρήση τις κάμερας μπορούμε να εντοπίσουμε σε πραγματικό χρόνο τους πλησιέστερους χώρους εστίασης

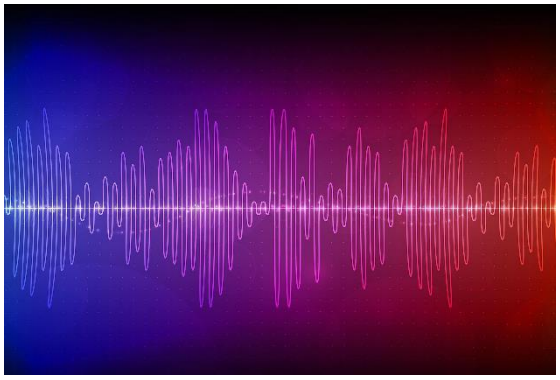
Σύντομα μπορεί οι έξυπνες συσκευές να είναι σε θέση να παρέχουν μια μεγάλη οθόνη για να παρακολουθούν οι χρήστες τις αγαπημένες τους ταινίες και να παίζουν παιχνίδια, διατηρώντας παράλληλα ένα μέγεθος τσέπης. Οθόνες που μπορούν να διπλώνονται και να ξετυλίγονται είναι πλέον πραγματικότητα χάρη στην OLED (Organic Light-Emitting Diode) τεχνολογία. Αυτές οι λεπτές οθόνες, ενδεχομένως να έχουν τη δυνατότητα στο μέλλον, να προβάλλουν και από τις δύο πλευρές τους, έτσι ώστε να μπορεί να δείξει κανείς φωτογραφίες ή βίντεο από τη μία επιφάνεια, ενώ παράλληλα να χρησιμοποιεί την άλλη για διαφορετικούς σκοπούς.



Εικόνα 9: Συσκευή με OLED τεχνολογία

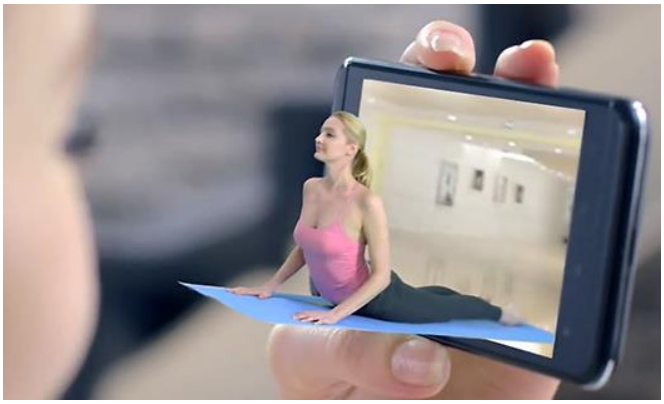
Η φωνή ελέγχου αποτελεί μία ακόμα τάση και έχει ήδη τραβήξει τη προσοχή από τη στιγμή που εμφανίστηκε η Siri (Speech Interpretation and Recognition Interface). Η Siri είναι ένα πρόγραμμα της Apple, το οποίο λειτουργεί σαν ένας έξυπνος προσωπικός βοηθός και σαν οδηγός πληροφοριών. Ο φωνητικός έλεγχος έχει υπάρξει σε πολλά από τα προηγούμενα κινητά τηλέφωνα, ακόμη και αν η λειτουργία φωνητικής αναγνώρισης ήταν

αρκετά αργή. Η έρευνα έχει γίνει για την προώθηση της ανάπτυξης του φωνητικού ελέγχου. Η Siri θα μπορούσε να σηματοδοτήσει μια σημαντική ανακάλυψη για τον φωνητικό έλεγχο και τον προγραμματισμό της αναγνώρισης που θα έπρεπε να γίνει. Αντί να αναγνωρίζει εντολές μέσω ηχητικών κυμάτων, όπως και τα περισσότερα συστήματα αναγνώρισης φωνής, η Siri αναγνωρίζει και ερμηνεύει την προφορά και τη σύνταξη της ομιλίας, όπως ακριβώς κάνουν οι άνθρωποι. Τέτοιες διεπαφές χρήστη μπορεί να αποδειχθούν αποτελεσματικές και ακριβείς.



Εικόνα 10: Ηχητικό κύμα

Τα έξυπνα κινητά τηλέφωνα μπορεί να έχουν ήδη φτάσει στην κορυφή όσον αφορά την ανάλυση της οθόνης τους, με το «Retina Display» της Apple να παρέχει μία ανάλυση που είναι πιο έντονη από ό,τι μπορεί να αντιληφθεί το ανθρώπινο μάτι. Ωστόσο, ακόμη και τότε, εμείς οι άνθρωποι εξακολουθούμε να θέλουμε περισσότερα. Εταιρείες κινητής τηλεφωνίας κινούνται πλέον από χαρακτηριστικά δύο διαστάσεων (2D) σε χαρακτηριστικά τριών διαστάσεων (3D) για τις οθόνες των έξυπνων συσκευών. Προς το παρόν, έχουμε λίγα τέτοια παραδείγματα 3D έξυπνων κινητών στην αγορά, όπως είναι το LG Optimus 3D και το Samsung W960 Amoled 3D.



Εικόνα 11: LG Optimus 3D

1.2 Η εξέλιξη των ενσωματωμένων συστημάτων διάγνωσης οχήματος

Ενσωματωμένο σύστημα διάγνωσης (OBD) είναι ένα σύστημα που αφορά τα αυτοκίνητα και αναφέρεται στην ικανότητα για αυτοδιάγνωση και αναφορά ενός οχήματος. Τα συστήματα OBD δίνουν πρόσβαση στον ιδιοκτήτη ή στον τεχνικό για την επισκευή του οχήματος σχετικά με την κατάσταση των διαφόρων υποσυστημάτων του οχήματος. Το ποσό των διαγνωστικών πληροφοριών που διατίθενται μέσω του OBD ποικίλλει ευρέως ανάλογα με την έκδοση του διαγνωστικού συστήματος από την εισαγωγή του, στις αρχές του 1980. Οι πρώτες εκδόσεις του OBD απλά ενημέρωναν μια ενδεικτική λυχνία δυσλειτουργίας, εάν ένα πρόβλημα εντοπιζόταν, αλλά δεν παρείχαν καμία πληροφορία ως προς τη φύση του προβλήματος. Οι σύγχρονες υλοποιήσεις OBD χρησιμοποιούν μια τυποποιημένη θύρα ψηφιακών επικοινωνιών, για να παρέχουν δεδομένα σε πραγματικό χρόνο, καθώς και μια τυποποιημένη σειρά κωδικών βλάβης διαγνωστικού ελέγχου (DTC), οι οποίοι επιτρέπουν σε κάποιον να εντοπίζει και να διορθώνει γρήγορα τυχόν δυσλειτουργίες εντός του οχήματος.

Εικόνα 12: OBD Adapter

1.2.1 Γρήγορη Ανασκόπηση

Πριν από τη δεκαετία του 1950, οι περισσότερες ανακαλύψεις των προβλημάτων στα συστήματα του αυτοκινήτου γίνονταν με λίγους μετρητές και οι υπόλοιπες μέσω των ανθρώπινων αισθήσεων. Καθώς τα οχήματα άρχισαν να γίνονται πιο περίπλοκα, τη δεκαετία του 1960, επειδή υπήρχε όλο και μεγαλύτερη εξάρτηση από τα μηχανικά όργανα των οχημάτων, ξεκίνησε η ανάγκη για πιο εξελιγμένα συστήματα διάγνωσης. Έτσι

τερματικά με οθόνη και βίντεο ξεκίνησαν να δείχνουν τα χαρακτηριστικά του ηλεκτρονικού συστήματος και, καθώς οι υπολογιστές έγιναν πιο εύκολα διαθέσιμοι εκείνη την εποχή, άλλες μετρήσεις, όπως η πίεση λαδιού και διάφορες θερμοκρασίες ενσωματώθηκαν σε ένα διαγνωστικό σύστημα μηχανής.

Όπως και σε άλλους τομείς, έτσι και στη διάγνωση βλαβών εντός των οχημάτων, οι άνθρωποι σιγά σιγά απομακρύνθηκαν όλο και περισσότερο από αυτή τη διαδικασία την οποία ανέλαβαν οι υπολογιστές και με αυτό τον τρόπο φτάσαμε στο στάδιο της εντός του οχήματος διάγνωσης (OBD). Σε αυτό το στάδιο οι αισθητήρες συνδέονται με διάφορα στοιχεία, όπως η θερμοκρασία σε ένα σύστημα ψύξης, και ταυτόχρονα ένα καλώδιο περνάει μέσα από όλα αυτά τα στοιχεία και φτάνει σε ένα τερματικό μπλοκ, το οποίο με τη σειρά του είναι συνδεδεμένο σε ένα διαγνωστικό μηχάνημα έξω από το αυτοκίνητο. Υπάρχουν επίσης, εντός του οχήματος διαγνωστικοί δείκτες, όπως προειδοποιητικά φώτα, μετρητές και αναγνώσεις από έναν υπολογιστή που γνωστοποιούν στον οδηγό ή στον τεχνικό την κατάσταση του οχήματος. Στην αρχή υπήρχαν κάποιοι μετρητές εντός του οχήματος, όπως οι δείκτες στάθμης καυσίμου, οι μετρητές του συστήματος ψύξης, οι μετρητές πίεσης λαδιού, και τα ταχύμετρα που κατέγραφαν την ταχύτητα του κινητήρα. Αργότερα, όμως, προστέθηκαν σε αυτά οι αισθητήρες οξυγόνου, οι μετρητές θερμοκρασίας του κινητήρα, καθώς και οι διατάξεις για τη μέτρηση της ροής του καυσίμου.

Εικόνα 13: Ενδεικτική λυχνία βλάβης

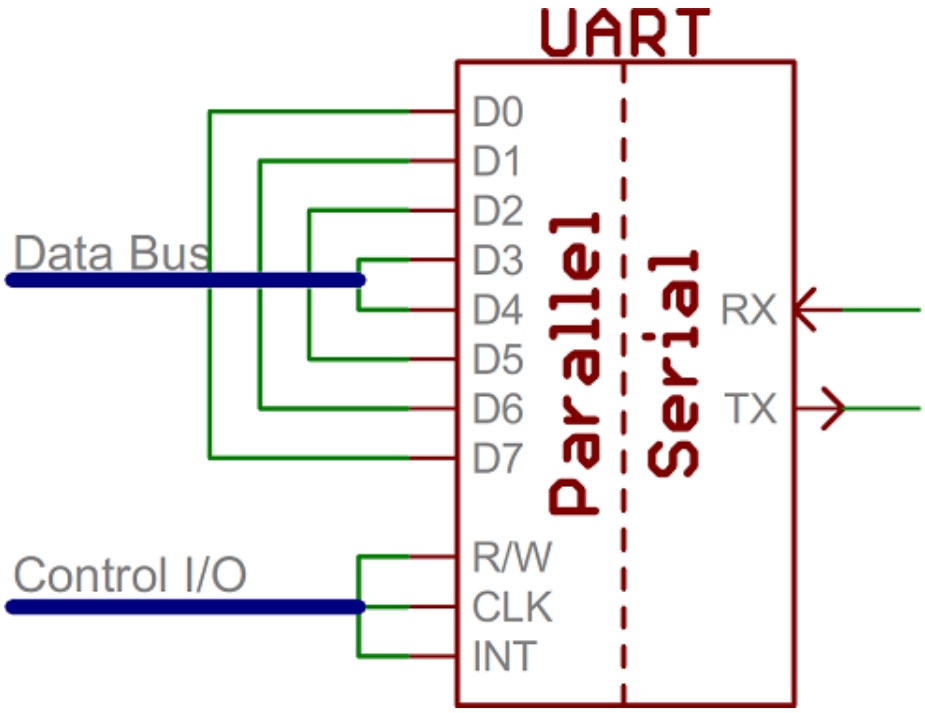
Πιο συγκεκριμένα τώρα, το 1968 η Volkswagen εισάγει το πρώτο ενσωματωμένο υπολογιστικό σύστημα σε μοντέλα της που είχαν σύστημα ψεκασμού καυσίμου, ακολουθούμενη από τη Datsun επτά χρόνια αργότερα, το 1975. Σιγά-σιγά, ενσωματωμένα υπολογιστικά συστήματα αρχίζουν να εμφανίζονται στα οχήματα των καταναλωτών, αν και σε μεγάλο βαθμό βέβαια υπαγορεύονται από την ανάγκη για συντονισμό των συστημάτων ψεκασμού καυσίμου σε πραγματικό χρόνο. Αντίστοιχα απλές OBD εφαρμογές

εμφανίζονται γενικά εκείνο το διάστημα, αν και δεν υπάρχει τυποποίηση ακόμα. Το 1980 η General Motors υλοποιεί μία δική της διεπαφή (interface) και ένα πρωτόκολλο για τον έλεγχο της ηλεκτρονικής μονάδας ελέγχου κινητήρα (ECM). Το ALDL πρωτόκολλο, όπως ονομάστηκε, επικοινωνεί στα 160 bit/s. Αρχικά υλοποιήθηκε σε οχήματα στην Καλιφόρνια για μοντέλα του 1980 και για το υπόλοιπο των Ηνωμένων Πολιτειών το 1981.

Το 1986 εμφανίζεται μια αναβαθμισμένη έκδοση του πρωτοκόλλου ALDL, το οποίο επικοινωνεί στα 8192 bit/s με ημιαμφίδρομη σηματοδότηση UART (Universal Asynchronous receiver/transmitter). Το UART είναι μία υπολογιστική συσκευή η οποία μεταφράζει δεδομένα μεταξύ παράλληλης και σειριακής μορφής. Δύο χρόνια μετά η Society of Automotive Engineers (SAE) συνιστά ένα τυποποιημένο σύνδεσμο διάγνωσης και το σύνολο των σημάτων για διαγνωστική εξέταση. Το 1991 η California Air Resources Board (CARB) απαιτεί όλα τα νέα οχήματα που πωλούνται στην Καλιφόρνια να έχουν κάποια βασική ικανότητα OBD. Οι απαιτήσεις αυτές αναφέρονται γενικά ως «OBD-I», αν και η ονομασία αυτή δεν εφαρμόζεται μέχρι την εισαγωγή του OBD-II. Η υποδοχή συνδέσμου δεδομένων (**D**ata **L**ink **C**onnector) και η θέση της δεν είναι τυποποιημένα, καθώς επίσης ούτε και το πρωτόκολλο δεδομένων. Περίπου το 1994 η CARB, παρακινούμενη από την επιθυμία για ένα εθνικό πρόγραμμα δοκιμών εκπομπών, εκδίδει το OBD-II με τις προδιαγραφές και τις εντολές που πρέπει να υιοθετηθούν από όλα τα αυτοκίνητα που πωλούνται στην Καλιφόρνια, αρχής γενομένης από το έτος 1996. Οι κωδικοί DTC και ο σύνδεσμος που πρότεινε η SAE ενσωματώνονται σε αυτές τις προδιαγραφές.

Το 1996 το OBD-II γίνεται υποχρεωτικό για όλα τα αυτοκίνητα που κατασκευάζονται και πωλούνται στις Ηνωμένες Πολιτείες. Το 2001 η Ευρωπαϊκή Ένωση κάνει το EOBD υποχρεωτικό για όλα τα βενζινοκίνητα οχήματα που πωλούνται στην Ευρωπαϊκή Ένωση. Δύο χρόνια αργότερα, το EOBD γίνεται υποχρεωτικό για όλα τα οχήματα ντίζελ που πωλούνται στην Ευρωπαϊκή Ένωση. Το 2008 όλα τα αυτοκίνητα που πωλούνται στις Ηνωμένες Πολιτείες οφείλουν να χρησιμοποιούν το πρότυπο ISO 15765-4 σηματοδότησης.

Παρακάτω θα αναφερθούμε σε διάφορες εκδόσεις του OBD, μέχρι να φτάσουμε στο OBD-II που χρησιμοποιείται σήμερα.



Εικόνα 14: UART

1.2.2 ALDL πρωτόκολλο

Το ALDL (Assembly Line Diagnostic Link) πρωτόκολλο της GM (General Motors) μερικές φορές αναφέρεται ως προκάτοχος του OBD-I διαγνωστικού συστήματος. Αυτή η διασύνδεση έγινε σε διάφορα μοντέλα και άλλαξε ανάλογα με τις μονάδες ελέγχου μηχανισμού μετάδοσης κίνησης (γνωστές και ως PCM, ECM, ECU). Διαφορετικές εκδόσεις είχαν μικρές διαφορές στις επαφές του συνδέσμου και στα ποσοστά των δεδομένων. Παλαιότερες εκδόσεις χρησιμοποιούσαν τα 160bit/s, ενώ νεότερες εκδόσεις έφτασαν μέχρι και τα 8192bit/s και χρησιμοποιούσαν αμφίδρομη επικοινωνία με το PCM.

1.2.3 M-OBD πρωτόκολλο

Multiplex OBD ή M-OBD είναι ένα OBD παραλλαγμένο πρωτόκολλο που χρησιμοποιήθηκε από την Toyota, πριν από τη διαμόρφωση του OBD-II. Το DLC3 της Toyota (Data Link Connector 3) είναι η τυπική υποδοχή δεκαέξι επαφών OBD-II, αλλά απαιτούνται τα δικά της καλώδια και το δικό της λογισμικό καθώς τα γενικά καλώδια και το λογισμικό του OBD-II δεν διασυνδέονται με αυτό.

1.2.4 OBD-I

Η ρυθμιστική πρόθεση του OBD-I ήταν να ενθαρρύνει τους κατασκευαστές αυτοκινήτων να σχεδιάζουν αξιόπιστα συστήματα ελέγχου των εκπομπών που παραμένουν αποτελεσματικά κατά τη «διάρκεια ζωής» του οχήματος. Οι κωδικοί βλάβης διαγνωστικού ελέγχου (DTC) των οχημάτων OBD-I μπορεί συνήθως να βρεθούν χωρίς να υπάρχει απαραίτητα ένα ακριβό εργαλείο σάρωσης. Κάθε κατασκευαστής χρησιμοποιούσε τη δική του υποδοχή διαγνωστικού συνδέσμου (DLC), τη δική του θέση DLC, τους δικούς του DTC ορισμούς και τη δική του διαδικασία για να διαβάσει τους κωδικούς DTC από το όχημα. Οι κωδικοί DTC από τα αυτοκίνητα OBD-I συχνά διαβάζονται μέσω των προτύπων «άναψε-σβήσε» της ενδεικτικής λυχνίας βλάβης. Με τη σύνδεση ορισμένων ακίδων του διαγνωστικού συνδέσμου, με τη βοήθεια της ενδεικτικής λυχνίας βλάβης θα αναβοσβήνει ένας διψήφιος αριθμός που αντιστοιχεί σε μια συγκεκριμένη κατάσταση σφάλματος. Ωστόσο, οι κωδικοί DTC ορισμένων αυτοκινήτων OBD-I ερμηνεύονται με διαφορετικούς τρόπους.

1.2.5 OBD-1.5

Το OBD 1.5 αναφέρεται σε μια μερική εφαρμογή του OBD-II που χρησιμοποιήθηκε από τη General Motors για ορισμένα οχήματα το 1994 και το 1995. (Η GM δεν χρησιμοποίησε τον όρο OBD 1.5 στην τεκμηρίωση για τα οχήματα αυτά. Απλά έχουν ένα OBD τμήμα και ένα OBD-II τμήμα στο εγχειρίδιο συντήρησης.)

Το pinout για τη σύνδεση ALDL σε αυτά τα αυτοκίνητα έχει ως εξής:

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16

Για τις συνδέσεις ALDL, η επαφή 9 είναι το ρεύμα δεδομένων, οι επαφές 4 και 5 είναι γειωμένες, και η επαφή 16 είναι η τάση της μπαταρίας. Ένα συμβατό εργαλείο σάρωσης απαιτείται για την ανάγνωση των κωδικών που παράγεται από το σύστημα OBD-1.5. Πρόσθετα διαγνωστικά συστήματα και κυκλώματα ελέγχου για το συγκεκριμένο αυτοκίνητο είναι επίσης διαθέσιμα σε αυτόν το σύνδεσμο.

1.2.6 OBD-II

Το OBD-II είναι μια βελτίωση του OBD-I τόσο στην ικανότητα όσο και στην τυποποίηση. Το πρότυπο OBD-II καθορίζει τον τύπο του διαγνωστικού συνδέσμου και τις επαφές του, τα διαθέσιμα ηλεκτρικά πρωτόκολλα σηματοδοσίας και τη μορφή μηνυμάτων. Επίσης, παρέχει μια υποψήφια λίστα των παραμέτρων του οχήματος για την παρακολούθησή του καθώς και τον τρόπο κωδικοποίησης των δεδομένων. Υπάρχει μία επαφή στην υποδοχή, η οποία παρέχει ενέργεια για το εργαλείο σάρωσης από τη μπαταρία του οχήματος και εξαλείφει την ανάγκη να συνδεθεί ένα εργαλείο σάρωσης σε μια πηγή τροφοδοσίας χωριστά. Ωστόσο, ορισμένοι τεχνικοί θα μπορούσαν ακόμα να συνδέσουν το εργαλείο σάρωσης σε μία βοηθητική πηγή ενέργειας για την προστασία των δεδομένων, στην ασυνήθιστη περίπτωση που ένα όχημα βιώσει μια απώλεια της ηλεκτρικής ενέργειας λόγω βλάβης. Τέλος, το πρότυπο OBD-II παρέχει μια επεκτάσιμη λίστα των κωδικών DTC. Ως αποτέλεσμα αυτής της τυποποίησης, μια συσκευή μπορεί να συλλέξει δεδομένα σχετικά με οποιοδήποτε όχημα συμβατό με το πρότυπο OBD-II. Αυτή η τυποποίηση είχε ζητηθεί από τις απαιτήσεις εκπομπών και, παρότι οι κωδικοί σχετίζονται μόνο με τις εκπομπές και τα δεδομένα που απαιτούνται για να μεταδοθεί μέσω αυτού, οι περισσότεροι κατασκευαστές έχουν κάνει την υποδοχή OBD-II ως το μοναδικό τρόπο μέσω του οποίου όλα τα συστήματα μπορούν να διαγνωστούν και να προγραμματιστούν. Οι OBD-II Diagnostic Trouble Codes (DTC) είναι τετραψήφιοι και προηγείται ένα γράμμα: P για τον κινητήρα και τη μετάδοση, B για το σώμα, C για σασί, και U για το δίκτυο.

1.2.7 EOBD

Οι EOBD (European On Board Diagnostics) κανονισμοί είναι το Ευρωπαϊκό ισοδύναμο του OBD-II. Καταγράφεται για πρώτη φορά μέσα σε κράτη μέλη της ΕΕ από την 1η Ιανουαρίου 2001 για βενζινοκίνητα αυτοκίνητα και από την 1η Ιανουαρίου 2004 για τα πετρελαιοκίνητα αυτοκίνητα. Για τα καινούρια μοντέλα, οι κανονισμοί είχαν τεθεί και εφαρμοστεί πριν από ένα χρόνο, την 1^{η} Ιανουαρίου 2000 για τα βενζινοκίνητα και την 1^{η} Ιανουαρίου 2003 για τα πετρελαιοκίνητα.

Εικόνα 15: Τυπικό απλό USB Diagnostic Interface

1.2.8 Το μέλλον για τα συστήματα διάγνωσης εντός οχήματος

Το OBD-II είναι ένα πολύ εξελιγμένο και αποτελεσματικό σύστημα για τον εντοπισμό προβλημάτων των εκπομπών αλλά, όταν πρόκειται για τη διόρθωση των προβλημάτων, δεν είναι πιο αποτελεσματικό από το OBD-I.

Επί του παρόντος, είναι υπό εξέταση τα σχέδια για το OBD-III, τα οποία θα πάνε το OBD-II ένα βήμα παραπέρα, προσθέτοντας την τηλεμετρία. Χρησιμοποιώντας μία μικρογραφία της τεχνολογίας ραδιοφωνικού αναμεταδότη, παρόμοια με εκείνη που χρησιμοποιείται ήδη για τα αυτόματα συστήματα ηλεκτρονικής είσπραξης διοδίων, ένα όχημα εξοπλισμένο με OBD-III θα είναι σε θέση να αναφέρει προβλήματα εκπομπών απευθείας σε μια ρυθμιστική αρχή. Ο αναμεταδότης θα ανακοινώνει τον αριθμό VIN του οχήματος και οποιουσδήποτε διαγνωστικούς κωδικούς που ήταν παρόντες. Το σύστημα θα μπορεί να αναφέρει αυτόματα ένα πρόβλημα εκπομπών μέσω μιας κυψελοειδούς ή δορυφορικής σύνδεσης τη στιγμή που η ενδεικτική λυχνία βλάβης (MIL) ανάβει, ή να απαντά σε ένα ερώτημα από ένα κινητό μέσω δορυφόρου.

Αυτό που κάνει αυτή την προσέγγιση τόσο ελκυστική για τις ρυθμιστικές αρχές είναι η αποτελεσματικότητά της και η εξοικονόμηση κόστους. Σύμφωνα με το ισχύον σύστημα, το σύνολο των οχημάτων σε μια περιοχή ή κράτος πρέπει να ελέγχονται μία φορά κάθε χρόνο ή κάθε δύο χρόνια για τον εντοπισμό του 30% περίπου των οχημάτων που έχουν προβλήματα εκπομπών. Με απομακρυσμένη παρακολούθηση, μέσω της τηλεμετρίας,

σε ένα όχημα εξοπλισμένο με OBD-III, η ανάγκη για περιοδικές επιθεωρήσεις θα μπορούσε να εξαλειφθεί, διότι μόνο τα οχήματα που θα ανέφεραν προβλήματα θα έπρεπε να δοκιμαστούν.

Εικόνα 16: Handheld scanner

Από τη μία πλευρά, το OBD-III, με την υποβολή των εκθέσεων τηλεμετρίας των προβλημάτων των εκπομπών, θα σώσει τους καταναλωτές από την ταλαιπωρία και το κόστος υποβολής του οχήματός τους σε ετήσια ή διετή δοκιμή εκπομπών. Εφόσον το όχημά τους δεν ανέφερε προβλήματα εκπομπής, δεν θα υπάρχει καμία ανάγκη να δοκιμαστεί. Από την άλλη πλευρά, όταν θα ανιχνεύεται ένα πρόβλημα των εκπομπών, θα είναι πολύ πιο δύσκολο κάποιος να αποφύγει να τη διόρθωσή του. Αυτός είναι ο στόχος όλων των προγραμμάτων καθαρού αέρα έτσι κι αλλιώς. Εξαλείφοντας τα οχήματα που στην πραγματικότητα προκαλούν τη μεγαλύτερη ρύπανση, ένα τέτοιο πρόγραμμα θα μπορούσε να αποφέρει σημαντικά κέρδη στη βελτίωση της ποιότητας του αέρα της κάθε χώρας. Αντίθετα, όπως έχει αυτή τη στιγμή η κατάσταση, οι ρυπαίνοντες μπορούν να ξεφεύγουν από τον εντοπισμό και κατά συνέπεια την επισκευή για έως και δύο χρόνια, σε περιοχές που έχουν διετείς επιθεωρήσεις, και, σε περιοχές που δεν έχουν προγράμματα επιθεώρησης, δεν υπάρχει κανένας τρόπος για τον εντοπισμό των οχημάτων αυτών. Έτσι το OBD-III θα τα αλλάξει όλα αυτά.

Βέβαια, η δυνατότητα παρακολούθησης κάθε κομματιού της μηχανής και οποιουδήποτε οχήματος (η ύπαρξη ενός Big Brother), που θα επιτευχθεί με το OBD-III, δεν

φαίνεται να ελκύει και τόσο τους ιδιώτες. Έτσι, τα πλεονεκτήματα του OBD-III θα πρέπει να πωλούνται στο κοινό με βάση την εξοικονόμηση κόστους, την ευκολία και τη δυνατότητα για βελτίωση της ποιότητας του αέρα. Ακόμα κι έτσι, υπάρχουν αρκετά ζητήματα σχετικά με τα δικαιώματα της ιδιωτικής ζωής και της προστασίας που πρέπει να λυθούν από την κάθε κυβέρνηση. Τα ζητήματα αυτά θα πρέπει να συζητηθούν και να επιλυθούν, πριν το OBD-III γίνει αποδεκτό. Λαμβάνοντας υπόψη τις τρέχουσες τάσεις, τέτοιες δραστικές αλλαγές φαίνονται απίθανες προς το παρόν.

Μια άλλη αλλαγή που θα μπορούσε να έρθει με το OBD-III θα ήταν ο ακόμη αυστηρότερος έλεγχος των εκπομπών των οχημάτων. Οι αλγόριθμοι ανίχνευσης διακοπτόμενης λειτουργίας, που απαιτούνται σήμερα από το OBD-II, παρακολουθούν μόνο για διαλείψεις που συμβαίνουν σε συνθήκες οδήγησης. Το OBD-III θα μπορούσε να προχωρήσει ακόμη περισσότερο, απαιτώντας να ελέγχει το γκάζι για να μειωθεί η πιθανότητα των διαλείψεων, ώστε τα οχήματα νέας γενιάς να έχουν χαμηλότερες εκπομπές. Έτσι, μέχρι να βρει το δρόμο του το OBD-III μέσω της κανονιστικής διαδικασίας, το μόνο για το οποίο πρέπει να ανησυχούμε είναι η διάγνωση και η επισκευή των OBD-II εξοπλισμένων οχημάτων και όλων των μη-OBD οχημάτων.

2

Περιγραφή του Αντικειμένου

2.1 Αντικείμενο και δομή της διπλωματικής εργασίας

Αντικείμενο της παρούσας διπλωματικής είναι η υλοποίηση μίας εφαρμογής διαγνωστικών οχήματος σε πλατφόρμα Android. Σκοπός μας είναι επομένως η κατασκευή μιας τέτοιας εφαρμογής που θα πρέπει να είναι απόλυτα λειτουργική και να μπορεί να δοκιμαστεί σε κινητό που έχει για λειτουργικό του σύστημα Android. Αυτή η εφαρμογή θα συνδέεται μέσω Bluetooth σε έναν OBD Adapter, ο οποίος θα είναι συνδεδεμένος πάνω σε αυτοκίνητο και θα λαμβάνει δεδομένα ανάλογα με τους κωδικούς (PIDs) που εμείς θέλουμε να λάβουμε. Στη συγκεκριμένη περίπτωση, θα λαμβάνουμε δεδομένα σε πραγματικό χρόνο σχετικά με πέντε μεταβλητές: ταχύτητα οχήματος, στροφές κινητήρα, θερμοκρασία μηχανής, σχετική θέση του πεντάλ του γκαζιού και κατανάλωση καυσίμου.

Η δομή της διπλωματικής έχει ως εξής: στο 1ο κεφάλαιο δίνονται διάφορες εισαγωγικές πληροφορίες για τις έξυπνες συσκευές (smartphones) και για τα συστήματα διαγνωστικών οχήματος (OBD) καθώς και η τυποποίηση τους. Στο 2ο κεφάλαιο αναλύουμε το σκοπό της διπλωματικής εργασίας, καθώς και τα διάφορα σενάρια χρήσης του αντικειμένου με το οποίο θα ασχοληθούμε. Αναλύουμε επίσης τον τρόπο υλοποίησης αυτής της εφαρμογής και δίνουμε μία περιγραφή του πώς λειτουργεί και χρησιμοποιείται το OBD. Στο τέλος, θα περιγράψουμε με λίγα λόγια το λειτουργικό σύστημα Android και θα αναφερθούμε σε διάφορα στοιχεία σχετικά με αυτό το λογισμικό. Στη συνέχεια, στο 3ο κεφάλαιο, κάνουμε μια εισαγωγή σχετικά με τις τεχνολογίες που χρησιμοποιήθηκαν, περιγράφουμε τη συσκευή elm327, καθώς επίσης και τις συσκευές android που χρησιμοποιήθηκαν για τις δοκιμές κατά τη διάρκεια της υλοποίησης της εφαρμογής. Δίνεται επίσης ένα σχεδιάγραμμα και μια περιγραφή της αρχιτεκτονικής του υλικού που χρησιμοποιήθηκε (hardware). Στο τέλος του 3ου κεφαλαίου δίνεται ένα σχεδιάγραμμα και μια αναλυτική περιγραφή της αρχιτεκτονικής του λογισμικού που χρειάστηκε για την υλοποίηση της εργασίας (software). Στο 4ο κεφάλαιο, που είναι και ο επίλογος αυτής της διπλωματικής εργασίας, γίνεται μία σύνοψη και μια συνολική αξιολόγησή της, καθώς

επίσης αναφέρονται και οι δυσκολίες που αντιμετωπίστηκαν κατά την υλοποίηση αυτής της εφαρμογής. Τέλος, δίνονται κάποια χρήσιμα συμπεράσματα και αναφέρονται κάποιες ενδεχόμενες μελλοντικές επεκτάσεις που θα μπορούσαν να υπάρξουν.

Στο 5ο κεφάλαιο βρίσκονται τα παραρτήματα της παρούσας διπλωματικής εργασίας. Στο παράρτημα Α παρουσιάζεται ένας πίνακας με διάφορα μεγέθη του OBD ενώ στο παράρτημα Β αναλύεται και παρουσιάζεται ο κώδικας της εφαρμογής. Το 6ο και τελευταίο κεφάλαιο αποτελείται από τη βιβλιογραφία που χρησιμοποιήθηκε για την εκπόνηση της παρούσας εργασίας.

2.2 Σενάρια χρήσης

Η παρούσα διπλωματική εργασία θα μπορούσε να βρει αρκετές εφαρμογές στον χώρο της διάγνωσης σφαλμάτων εντός του οχήματος και όχι μόνο. Παρακάτω αναλύονται μερικά πιθανά σενάρια χρήσης της εφαρμογής που υλοποιήσαμε.

1. Ένα σενάριο είναι να θέλει κάποιος ιδιώτης να έχει πρόσβαση σε διάφορες πληροφορίες σχετικά με το αυτοκίνητό του σε πραγματικό χρόνο. Για παράδειγμα, να παρακολουθεί την κατανάλωση καυσίμου που έχει κατά την διάρκεια ενός ταξιδιού. Το μόνο που του χρειάζεται είναι κάποια έξυπνη συσκευή με λειτουργικό σύστημα Android και έναν OBD αντάπτορα, ώστε με την συγκεκριμένη εφαρμογή να έχει πρόσβαση σε τέτοια δεδομένα.

2. Ένα δεύτερο σενάριο είναι να θέλει κάποιος τεχνικός αυτοκινήτων να έχει πρόσβαση σε κωδικούς σφαλμάτων, ώστε, όταν υπάρχει κάποιο πρόβλημα, να ξέρει την αιτία και να μπορεί να το διορθώσει. Με αυτή την εφαρμογή θα μπορούσε να έχει πρόσβαση σε τέτοια δεδομένα.

3. Με την υλοποίηση αυτής της εφαρμογής είναι δυνατό να χρησιμοποιηθούν καταγραφείς δεδομένων (data loggers). Οι καταγραφείς δεδομένων έχουν σχεδιαστεί για να συλλαμβάνουν τα στοιχεία του οχήματος, ενώ το όχημα είναι σε κανονική λειτουργία, ώστε να μπορούν αργότερα να αξιοποιηθούν και να αναλυθούν. Χρήσεις καταγραφής δεδομένων περιλαμβάνουν:

- Παρακολούθηση του κινητήρα και του οχήματος, υπό κανονική λειτουργία, για το σκοπό της διάγνωσης ή της ρύθμισης.
- Μερικές ασφαλιστικές εταιρείες αυτοκινήτων προσφέρουν μειωμένα ασφάλιστρα, αν έχουν εγκατασταθεί καταγραφείς δεδομένων στο όχημα.

Εικόνα 17: TEXA OBD log. Μικρός καταγραφέας δεδομένων με τη δυνατότητα να διαβάζει τα δεδομένα αργότερα από υπολογιστή μέσω USB

Η ανάλυση των δεδομένων του μαύρου κουτιού του οχήματος μπορεί να εκτελεστεί σε περιοδική βάση αυτόματα, να μεταδοθεί ασύρματα σε τρίτους ή να ανακτηθεί για την εγκληματολογική ανάλυση μετά από ένα συμβάν, όπως είναι ένα ατύχημα ή μία παράβαση του Κ.Ο.Κ. ή μία μηχανική βλάβη. Επομένως, μία εφαρμογή, σαν αυτή που υλοποιήθηκε, θα ήταν αρκετά χρήσιμη, αν έστελνε τα δεδομένα σε έναν τέτοιο καταγραφέα.

4. Το OBD-II δεν χρησιμοποιείται για την επισκευή οχημάτων μόνο από επαγγελματίες και φαν. Οι πληροφορίες του OBD-II χρησιμοποιούνται συνήθως από συσκευές τηλεματικής οχημάτων που εκτελούν παρακολούθηση της απόδοσης των καυσίμων, την πρόληψη της μη ασφαλούς οδήγησης, καθώς και για την απομακρυσμένη διάγνωση. Με την παρακολούθηση OBD-II και τους κωδικούς DTC, μια εταιρία μπορεί να ξέρει αμέσως εάν ένα από τα οχήματα της έχει ένα πρόβλημα στον κινητήρα και ερμηνεύοντας τον κωδικό γνωρίζει αμέσως τη φύση του προβλήματος. Έτσι, όπως γίνεται αντιληπτό, μία τέτοια εφαρμογή σαν αυτή που υλοποιήθηκε στην παρούσα διπλωματική εργασία θα μπορούσε να είναι πολύ χρήσιμη για τα αυτοκίνητα μίας εταιρίας.

Η ίδια εφαρμογή θα μπορούσε να χρησιμοποιηθεί για τους παραπάνω σκοπούς με κάποιες μικρές τροποποιήσεις, ώστε, αντί η μεταφορά των δεδομένων να γίνεται μέσω Bluetooth, να γίνεται μέσω καλωδίων USB αντάπτορα ή και με προσαρμογείς WiFi συνδεδεμένους στην υποδοχή OBD-II του αυτοκινήτου. Η μόνη απαραίτητη προϋπόθεση είναι να υπάρχει κάποια έξυπνη συσκευή που να τρέχει λογισμικό Android. Βέβαια, αν κρατηθεί το ίδιο σκεπτικό με κάποιες αλλαγές ίσως σε επίπεδο κώδικα και σε γλώσσα προγραμματισμού (αντί για Java να χρησιμοποιηθεί C++), θα μπορούσε να υλοποιηθεί ενδεχομένως μια αντίστοιχη εφαρμογή για συσκευές που χρησιμοποιούν άλλο λογισμικό, όπως iOS.

Συνοψίζοντας, σε όλες τις παραπάνω περιπτώσεις εκμεταλλευόμαστε τους κωδικούς σφαλμάτων του OBD. Οπότε, η συγκεκριμένη εργασία μπορεί να χρησιμοποιηθεί σε κάποια από τα προαναφερθέντα σενάρια ή για άλλους αντίστοιχους σκοπούς είτε από ιδιώτες είτε από τεχνικούς που θέλουν να επιλύσουν κάποιο τεχνικό πρόβλημα σε ένα όχημα. Πιο ειδικά όμως, η παρούσα διπλωματική θα μπορούσε να έχει ευρεία χρήση για αναλύσεις των παραπάνω δεδομένων σε συσκευές που έχουν ως κύριο λειτουργικό τους σύστημα Android.

2.3 Τρόπος υλοποίησης

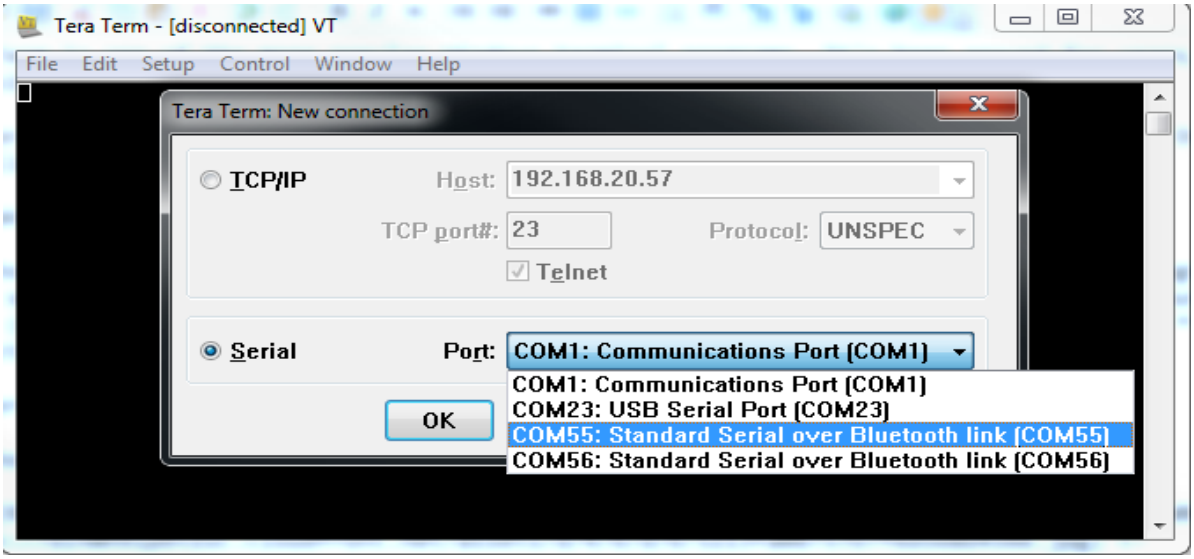
Στην παρούσα διπλωματική εργασία χρησιμοποιήθηκαν αρκετά εργαλεία προκειμένου να φτάσουμε στην τελική υλοποίηση της εφαρμογής, η οποία θέλαμε να είναι και πλήρως λειτουργική. Σε πρώτη φάση χρειαζόταν ένα περιβάλλον στο οποίο θα γινόταν η ανάπτυξη του προγράμματος και συνεπώς της εφαρμογής. Ένα τέτοιο περιβάλλον κατάλληλο για ανάπτυξη εφαρμογών Android είναι το Android Studio και επομένως αυτό επιλέχθηκε. Μετά την επιλογή του προγράμματος και πριν την ανάπτυξη του κώδικα σε Java, χρειαζόταν μία κατάλληλη διεπαφή χρήστη (User Interface) στην οποία θα απεικονίζονταν τα δεδομένα.

Στο επόμενο στάδιο, ξεκίνησε η ανάπτυξη του κώδικα της Java. Στο πρώτο κομμάτι έπρεπε να δούμε πως θα μεταφράζουμε τους κωδικούς PID από το δεκαεξαδικό σύστημα στο δεκαδικό σύστημα, ώστε να φαίνονται τα δεδομένα στα οποία θα αναφερόμασταν, για παράδειγμα ταχύτητα οχήματος, ενώ, στο δεύτερο κομμάτι, η προσπάθεια που έγινε αφορούσε την επικοινωνία μέσω Bluetooth (Bluetooth interface) και τη διαδικασία ανταλλαγής σειριακών δεδομένων μέσω του καναλιού που δημιουργείται.

Όσον αφορά την επικοινωνία μέσω Bluetooth, έγιναν αρκετοί πειραματισμοί και σε αυτούς βοήθησε το τερματικό πρόγραμμα Tera Term που χρησιμοποιήθηκε για να διαπιστωθεί η ποιότητα και η δυνατότητα επικοινωνίας μέσω Bluetooth. Δηλαδή, γινόταν αποστολή και λήψη σειριακών δεδομένων μέσω του τερματικού στην εφαρμογή και αντίστροφα. Στο τελευταίο στάδιο, έγιναν δοκιμές για να διαπιστωθεί ότι η εφαρμογή επικοινωνεί με τον OBD Adapter (συσκευή elm327) και είναι σε θέση να ανταλλάσσει δεδομένα καθώς και ότι είναι πλήρως λειτουργική.

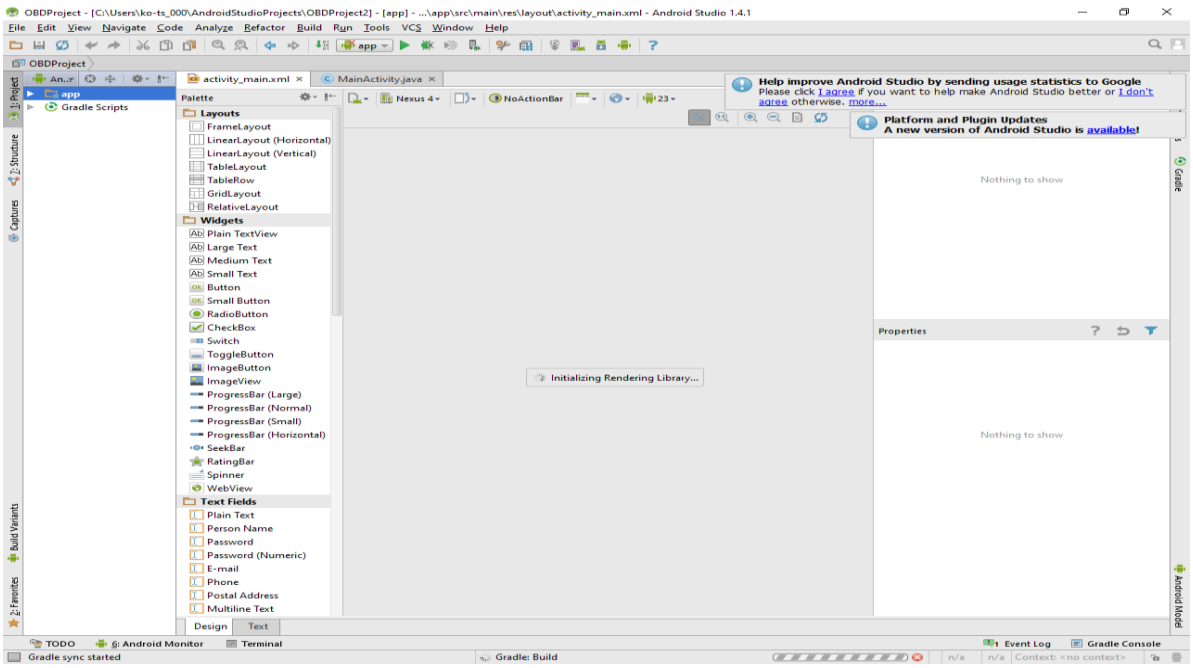
Συνολικά, τα εργαλεία που χρησιμοποιήσαμε για να υλοποιήσουμε την εφαρμογή της παρούσας διπλωματικής εργασίας ήταν:

- Υλικό
  - Δύο Android συσκευές
  - Τέσσερα οχήματα για τα τεστ
  - Δύο συσκευές elm327
  - Δύο υπολογιστές για το τρέξιμο των προγραμμάτων
  - Έναν Bluetooth αντάπτορα
- Λογισμικό
  - Tera Term
  - Android Studio
  - Java
  - Eclipse



Εικόνα 18: Tera Term

Όπως αναφέραμε παραπάνω, ένα πολύ σημαντικό κομμάτι αυτής της εργασίας ήταν η επικοινωνία της εφαρμογής μέσω Bluetooth με τη συσκευή elm327. Έτσι, σε πρώτη φάση έγιναν αρκετές δοκιμές μέσω ενός τερματικού (Tera Term) από υπολογιστή, για να επιτευχθεί και να δοκιμαστεί αυτή η επικοινωνία. Μετά από αυτές τις δοκιμές και, αφού η διεπαφή του Bluetooth ήταν λειτουργική, εφαρμόστηκε η ίδια στρατηγική με τη συσκευή elm327. Ουσιαστικά, επομένως, αυτό που κάναμε ήταν να προσομοιώσουμε την πλευρά του OBD αντάπτορα μέσω του τερματικού. Όπως αποδείχθηκε, αυτές οι δοκιμές βοήθησαν αρκετά στην τελική υλοποίηση της εφαρμογής.



Εικόνα 19: Android Studio

2.4 Περιγραφή του OBD

2.4.1 Γενικά Στοιχεία

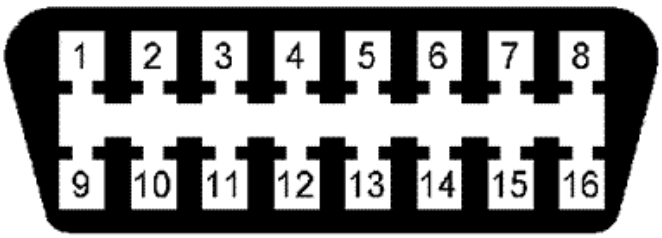
Το OBD-II παρέχει πρόσβαση σε δεδομένα από τη μονάδα ελέγχου του κινητήρα (ECU) και προσφέρει μια πολύτιμη πηγή πληροφοριών κατά την αντιμετώπιση προβλημάτων μέσα σε ένα όχημα. Το πρότυπο SAE J1979 καθορίζει μια μέθοδο για την αίτηση διαφόρων διαγνωστικών στοιχείων και έναν κατάλογο τυποποιημένων παραμέτρων που μπορεί να είναι διαθέσιμη από το ECU. Οι διάφορες παράμετροι που είναι διαθέσιμες έχουν διευθυνσιοδοτηθεί με κάποιους κωδικούς ή αλλιώς PIDs που ορίζονται στο J1979. Οι

κατασκευαστές δεν υποχρεούνται να εφαρμόσουν όλα τα PIDs που απαριθμούνται στο πρότυπο J1979 και τους επιτρέπεται να περιλαμβάνουν τα δικά τους PIDs που δεν έχουν καταγραφεί. Η αίτηση και η ανάκτηση των δεδομένων του συστήματος PID δίνει πρόσβαση σε δεδομένα απόδοσης σε πραγματικό χρόνο, καθώς και τους σημειωμένους κωδικούς DTC (κωδικοί σφαλμάτων). Μεμονωμένοι κατασκευαστές συχνά ενισχύουν τον κώδικα OBD-II με πρόσθετους κωδικούς DTC.

Τα καθήκοντα του OBD είναι τα εξής:

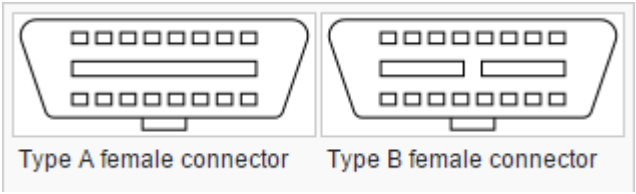
- η συνεχής παρακολούθηση όλων των εκπομπών σε κάθε όχημα
- η ανά πάσα στιγμή καταγραφή και αναφορά της σημαντικής αύξησης των εκπομπών σε όλη τη διάρκεια της λειτουργίας του οχήματος
- οι σταθερά χαμηλές εκπομπές καυσαερίων
- η προστασία των στοιχείων, όπως είναι ο καταλύτης
- η λειτουργία του ως διεπαφή για την ανάγνωση των αποθηκευμένων δεδομένων που έχει το σύστημα του οχήματος

Για τα οχήματα της ευρωπαϊκής αγοράς υπάρχει μια πρίζα δεκαέξι επαφών, όπως της παρακάτω εικόνας, που διατίθεται στα πλαίσια των κανονισμών και αναφέρεται επίσης η συμβατότητα αναφορικά με το OBD-II / EOBD και στα έγγραφα του οχήματος.



Επαφή (Σήμα)	Επαφή (Σώμα)	Επαφή (Σήμα)	Επαφή (Σήμα)	Επαφή (+12 V)	Πρωτόκολλο
--	4 + 5	7	15	16	ISO 9141-2
2	4 + 5	--	10	16	PWM J1850
2	4 + 5	--	--	16	VPW J1850
--	4 + 5	6	14	16	CAN Bus

Εικόνα 20: OBD Connector



Εικόνα 21: OBD Connectors

Η σύνδεση είναι τυποποιημένη κατά μία απόσταση κοντά στον οδηγό και βρίσκεται συνήθως κάτω από το ταμπλό, κοντά στο τιμόνι ή κάτω από το σταχτοδοχείο. Δεν είναι όλες οι επαφές κατειλημμένες. Ανάλογα με το πρόσθετο πρωτόκολλο, κάποιες συνδέσεις δεν μπορούν να καλυφθούν με ακίδες επαφής. Ο σύνδεσμος OBD-II είναι αναγνώσιμος για τις παραμέτρους σε όλα τα οχήματα με τον ίδιο τρόπο, αλλά τα πρωτόκολλα μεταφοράς, που χρησιμοποιούνται για το σκοπό αυτό, μπορεί να διαφέρουν. Οι κατασκευαστές δεν μπόρεσαν να συμφωνήσουν εδώ, δυστυχώς. Ως κανόνας για την General Motors, για τα αυτοκίνητα και τα ελαφρά φορτηγά το πρότυπο που εφαρμόζεται είναι το SAE J1850 VPW (Variable Pulse Width Modulation), για αυτοκίνητα Chrysler και όλων των ευρωπαϊκών και ασιατικών μοντέλων οχήματα είναι το ISO 9141 KWP (Key Word Protocol) και για τη Ford είναι το SAE J1850 PWM (Pulse Width Modulation). Για τα οχήματα από το έτος μοντέλου 1996 και μετά, το σύστημα μπορεί να προσδιοριστεί από την ανάθεση των ακίδων του πρωτοκόλλου που χρησιμοποιείται.

Παρακάτω περιγράφονται οι τρόποι λειτουργίας που περιλαμβάνει το OBD-II.

2.4.2 Τρόποι λειτουργίας

Εδώ είναι μια βασική εισαγωγή στο πρωτόκολλο επικοινωνίας OBD σύμφωνα με το πρότυπο ISO 15031:

- Λειτουργία 1η:

  Χρησιμοποιείται για να προσδιορίσει ποιες πληροφορίες των συστημάτων μετάδοσης κίνησης είναι διαθέσιμες στο εργαλείο σάρωσης.

- Λειτουργία 2η:

  Εμφανίζει στοιχεία στιγμιαίων πληροφοριών.

- Λειτουργία 3η:

  Απαριθμεί τις εκπομπές οι οποίες σχετίζονται με «επιβεβαιωμένους» διαγνωστικούς κωδικούς προβλήματος που αποθηκεύονται. Εμφανίζει τετραψήφιους κωδικούς αναγνώρισης των σφαλμάτων.

- Λειτουργία 4η:

  Χρησιμοποιείται για να καθαρίσει διαγνωστικές πληροφορίες σχετικά με τις εκπομπές. Αυτό περιλαμβάνει την εκκαθάριση των αποθηκευμένων κωδικών DTC, που εκκρεμούν ή που έχουν επιβεβαιωθεί, και των στοιχείων στιγμιαίων πληροφοριών.

- Λειτουργία 5η:

  Σχετίζεται με τον αισθητήρα οξυγόνου και τα αποτελέσματα των δοκιμών που συγκεντρώθηκαν για αυτόν. Υπάρχουν δέκα αριθμοί που διατίθενται για τη διάγνωση.

- Λειτουργία 6η:

  Σχετίζεται με αιτήσεις για τα αποτελέσματα των δοκιμών παρακολούθησης του συστήματος η οποία μπορεί να είναι είτε συνεχής είτε μη-συνεχής.

- Λειτουργία 7η:

  Σχετίζεται με αιτήσεις για διαγνωστικούς κωδικούς προβλήματος όσον αφορά τις εκπομπές που ανιχνεύονται κατά τη διάρκεια του τρέχοντος ή του τελευταίου ολοκληρωμένου κύκλου οδήγησης. Επιτρέπει στον εξωτερικό εξοπλισμό δοκιμών να αποκτήσει διαγνωστικούς κωδικούς προβλημάτων που εκκρεμούν και που εντοπίστηκαν κατά τον τρέχοντα ή τον τελευταίο ολοκληρωμένο κύκλο οδήγησης για τα εξαρτήματα/συστήματα τα οποία σχετίζονται με τις εκπομπές. Αυτή η λειτουργία χρησιμοποιείται από τεχνικούς συντήρησης μετά την επισκευή του οχήματος και μετά την εκκαθάριση διαγνωστικών πληροφοριών, για να δουν τα αποτελέσματα των δοκιμών μετά από έναν κύκλο οδήγησης και για να καθοριστεί εάν η επισκευή έχει επιλύσει το πρόβλημα.

- Λειτουργία 8η:

  Θα μπορούσε να ενεργοποιήσει την εκτός οχήματος συσκευή δοκιμής για τον έλεγχο της λειτουργίας ενός εντός οχήματος συστήματος.

- Λειτουργία 9η:

  Χρησιμοποιείται για την ανάκτηση πληροφοριών του οχήματος.

- Λειτουργία 10η:

  Απαριθμεί διαγνωστικούς κωδικούς προβλήματος σχετικά με τις εκπομπές που αποθηκεύονται.

Εμείς στην παρούσα διπλωματική εργασία χρησιμοποιήσαμε μόνο την πρώτη λειτουργία, γιατί θέλαμε να λαμβάνουμε δεδομένα σε πραγματικό χρόνο για πέντε μεταβλητές. Αλλά με αντίστοιχο τρόπο, θα μπορούσαμε να λάβουμε απαντήσεις και για τις υπόλοιπες λειτουργίες μέσω της εφαρμογής μας.

2.4.3 OBD-II PIDs

Οι OBD-II PIDs είναι κωδικοί που χρησιμοποιούνται για να ζητήσουν τα δεδομένα από ένα όχημα και χρησιμοποιούνται επομένως ως διαγνωστικό εργαλείο. Το πρότυπο SAE J1979 ορίζει πολλούς PIDs κωδικούς, αλλά οι κατασκευαστές καθορίζουν επίσης πολλούς περισσότερους για τα οχήματά τους. Συνήθως, ένας τεχνικός μιας αυτοκινητοβιομηχανίας θα χρησιμοποιήσει τους PIDs κωδικούς με ένα εργαλείο σάρωσης συνδεδεμένο με τη θύρα OBD-II του οχήματος.

Η διαδικασία έχει ως εξής:

- Ο τεχνικός εισάγει τον κωδικό (PID).
- Το εργαλείο σάρωσης στέλνει τον κωδικό στο δίαυλο δικτύου του ελεγκτή περιοχής του οχήματος (CAN-bus).
- Μια συσκευή στον δίαυλο αναγνωρίζει τον κωδικό και αναφέρει την τιμή σχετικά με αυτό το PID στο δίαυλο.
- Το εργαλείο σάρωσης διαβάζει την απάντηση και την εμφανίζει στον τεχνικό.

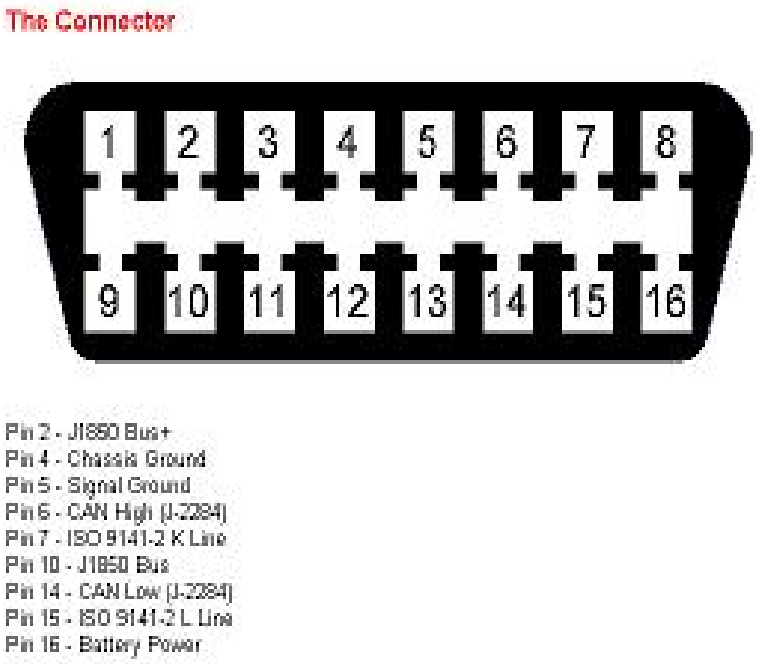
Ένας ελεγκτής δικτύου περιοχής (CAN-bus) είναι ένα πρότυπο διαύλου σχεδιασμένου για να επιτρέπει μικροελεγκτές και συσκευές να επικοινωνούν μεταξύ τους σε εφαρμογές, χωρίς έναν κεντρικό υπολογιστή. Είναι ένα μήνυμα που βασίζεται σε πρωτόκολλο, σχεδιασμένο αρχικά για πολλαπλή ηλεκτρική καλωδίωση μέσα στα αυτοκίνητα.

Κάποιοι βασικοί PID κωδικοί φαίνονται στο παράρτημα Α, όπως ορίζονται από την SAE J1979. Η αναμενόμενη απάντηση για κάθε PID δίνεται μαζί με πληροφορίες για το πώς να μεταφράσει την απάντηση σε δεδομένα.

2.4.4 Υποδοχή OBD

Η θύρα OBD, επίσης γνωστή ως ο διαγνωστικός σύνδεσμος SAE J1962 (SAE

J1962 diagnostic connector), είναι είτε άσπρη ή μαύρη, και συνήθως βρίσκεται πίσω από το ταμπλό και πάνω από το πεντάλ του φρένου. Έχει κενά που διατίθενται για δεκαέξι επαφές, αλλά ίσως να μην έχει πραγματικά δεκαέξι επαφές. Οι τοποθεσίες των επαφών ποικίλλουν ανάλογα με το πρωτόκολλο του σήματος που υποστηρίζεται από το όχημα. Η θύρα επιτρέπει τη σύνδεση με ένα εργαλείο σάρωσης, ή οποιαδήποτε άλλη συμβατή συσκευή με OBD-II με σκοπό να αποκτηθούν πληροφορίες σχετικά με το όχημα.



Εικόνα 22: Υποδοχή OBD

2.4.5 Πρωτόκολλα OBD-II

Ένα οποιοδήποτε όχημα που είναι συμβατό με το OBD-II μπορεί να χρησιμοποιήσει ένα από τα πέντε πρωτόκολλα επικοινωνίας: SAE J1850 PWM, SAE J1850 VPW, ISO9141-2, ISO14230-4 (KWP2000), και από το 2003 το ISO 15765-4/SAE J2480. Πολλοί υποστηρίζουν ότι είναι εννιά τα πρωτόκολλα. Αυτό συμβαίνει γιατί υπολογίζουν σαν ξεχωριστά τις τέσσερις υποκατηγορίες του CAN-BUS. Συνολικά έχουμε:

- ISO15765-4 (CAN-BUS)
    - ISO 15765-4 CAN (11 bit ID, 500 Kbaud)
    - ISO 15765-4 CAN (29 bit ID, 500 Kbaud)
    - ISO 15765-4 CAN (11 bit ID, 250 Kbaud)
    - ISO 15765-4 CAN (29 bit ID, 250 Kbaud)

- ISO14230-4 (KWP2000)
    - ISO14230-4 KWP (5 baud init, 10.4 Kbaud)
    - ISO14230-4 KWP (fast init, 10.4 Kbaud)
- ISO9141-2
- SAE J1850 VPW
- SAE J1850 PWM

2.5 Κινητό Λειτουργικό Σύστημα Android

2.5.1 Γενικά

Το Android είναι ένα κινητό λειτουργικό σύστημα (OS) που έχει αναπτυχθεί επί του παρόντος από την Google, βασίζεται στον πυρήνα του Linux και έχει σχεδιαστεί κυρίως για την οθόνη αφής φορητών έξυπνων συσκευών (όπως smartphones και tablets). Η διεπαφή χρήστη του Android στηρίζεται κυρίως στην άμεση χειραγώγηση, χρησιμοποιώντας χειρονομίες αφής που αντιστοιχούν σε πραγματικές δράσεις, όπως το σύρσιμο, το άγγιγμα και το τσίμπημα, με σκοπό να χειραγωγήσουν τα αντικείμενα στην οθόνη και περιλαμβάνει ακόμη ένα εικονικό πληκτρολόγιο για την εισαγωγή κειμένου. Εκτός από τις συσκευές με οθόνη αφής, η Google έχει αναπτύξει περαιτέρω το Android TV για τηλεοράσεις, Android Auto για τα αυτοκίνητα, και Android Wear για ρολόγια χειρός, το καθένα με ένα εξειδικευμένο περιβάλλον εργασίας χρήστη. Διάφορες παραλλαγές του Android χρησιμοποιούνται επίσης για φορητούς υπολογιστές, κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές και άλλα ηλεκτρονικά είδη. Αρχικά αναπτύχθηκε από την Android, Inc, την οποία αγόρασε η Google το 2005. Το Android παρουσιάστηκε το 2007, μαζί με την ίδρυση της Open Handset Alliance, η οποία είναι μια κοινοπραξία εταιρειών υλικού (hardware), λογισμικού (software) και τηλεπικοινωνιών που αφοσιώθηκαν για την προώθηση ανοιχτών προτύπων για τις κινητές συσκευές. Από τον Ιούλιο του 2013, το Google Play store είχε πάνω από ένα εκατομμύριο εφαρμογές Android «apps» που δημοσιεύθηκαν και πάνω από 50 δισεκατομμύρια εφαρμογές που έχουν ληφθεί.

Ο πηγαίος κώδικας του Android απελευθερώνεται από τη Google, σύμφωνα με άδειες ανοιχτού κώδικα, αν και οι περισσότερες συσκευές Android τελικά κυκλοφορούν με ένα συνδυασμό ανοικτού κώδικα και κλειστού λογισμικού, συμπεριλαμβάνοντας ιδιόκτητο λογισμικό που απαιτείται για την πρόσβαση σε υπηρεσίες της Google. Το Android είναι δημοφιλές με τις εταιρείες τεχνολογίας που απαιτούν ένα έτοιμο, χαμηλού κόστους και

προσαρμόσιμο λειτουργικό σύστημα για συσκευές υψηλής τεχνολογίας. Η ανοιχτή φύση του έχει ενθαρρύνει μια μεγάλη κοινότητα προγραμματιστών και ενθουσιώδεις φαν να χρησιμοποιήσουν τον ανοιχτό κώδικα ως θεμέλιο για διάφορα δικά τους έργα, τα οποία προσθέτουν νέα χαρακτηριστικά για προχωρημένους χρήστες ή για να φέρουν το Android σε συσκευές που αρχικά είχαν κυκλοφορήσει με άλλα λειτουργικά συστήματα.

2.5.2 Βασικά χαρακτηριστικά του Android

1. Επειδή οι Android συσκευές είναι συνήθως τροφοδοτούμενες από μπαταρία, το Android έχει σχεδιαστεί με τέτοιο τρόπο ώστε να κρατάει την κατανάλωση ενέργειας στο ελάχιστο. Όταν μια εφαρμογή δεν είναι σε χρήση, το σύστημα διακόπτει τη λειτουργία της, έτσι ώστε, ενώ είναι διαθέσιμη για άμεση χρήση και όχι κλειστή, δεν κάνει χρήση της μπαταρίας ή των πόρων της CPU.

2. Το Android διαχειρίζεται τις εφαρμογές που αποθηκεύονται στη μνήμη αυτόματα. Όταν η μνήμη είναι σε χαμηλά επίπεδα, το σύστημα θα αρχίσει αόρατα και αυτόματα το κλείσιμο ανενεργών διαδικασιών, αρχής γενομένης με εκείνες που έχουν μείνει ανενεργές για μεγαλύτερο διάστημα.

3. Η κύρια πλατφόρμα υλικού (hardware) για το Android είναι η ARM (ARMv7 και ARMv8-A αρχιτεκτονικές), με τις x86 και MIPS αρχιτεκτονικές να υποστηρίζονται επίσημα σε νεότερες εκδόσεις του Android. Οι απαιτήσεις για το ελάχιστο ποσό της μνήμης RAM ποικίλλουν ανάλογα με την έκδοση Android και το κινητό.

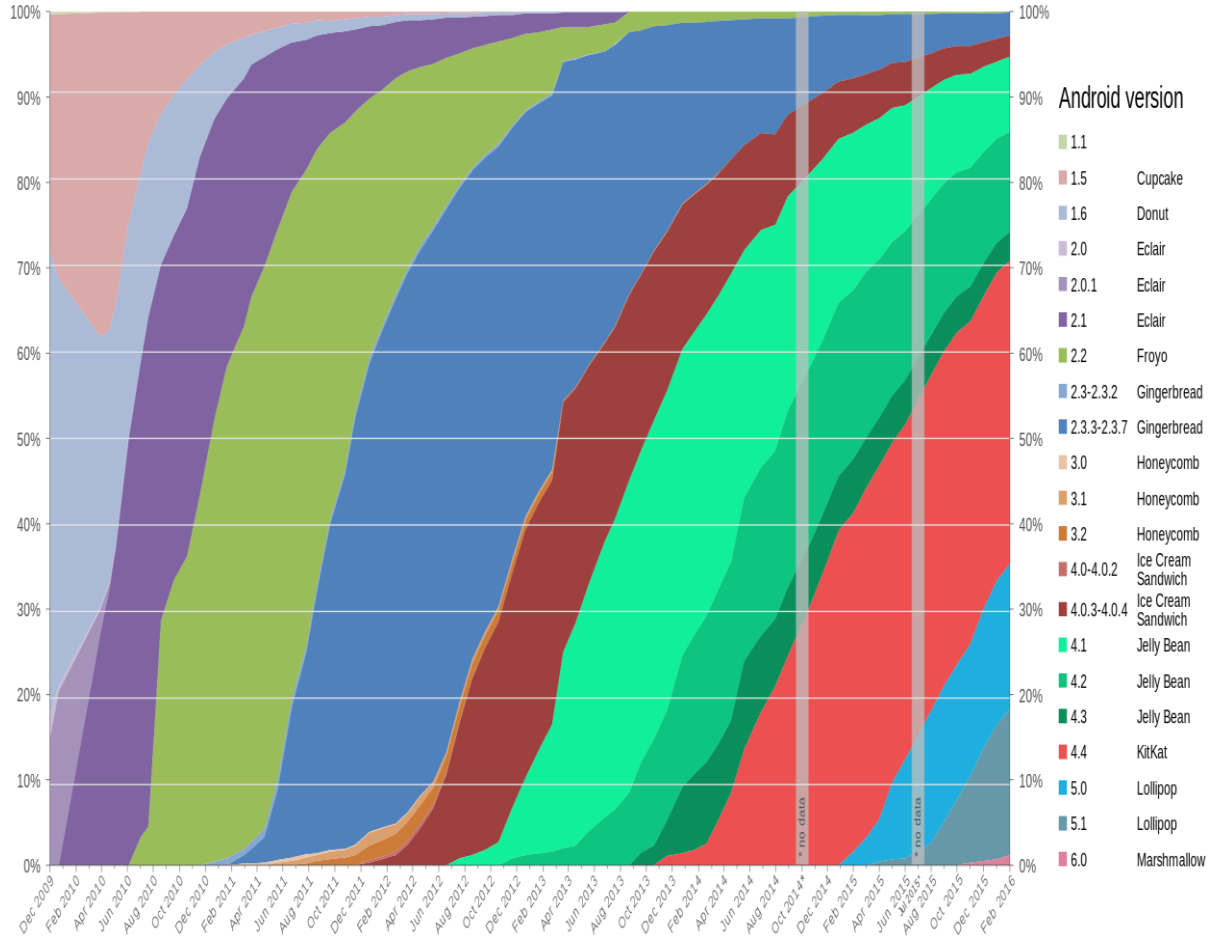
Επίσης αξίζει να αναφέρουμε ότι πλέον έχουμε φτάσει στην έκδοση Android 6.0 (γνωστή και ως Marshmallow), η οποία όμως τρέχει σε ποσοστό συσκευών μικρότερο του 1% παγκοσμίως. Σε αυτό το σημείο μας δίνεται το έναυσμα να αναφερθούμε στις εκδόσεις Android από την αρχή της ιστορίας τους.

2.5.3 Εκδόσεις Android

Η ιστορία των εκδόσεων του Android κινητού λειτουργικού συστήματος ξεκίνησε με την απελευθέρωση του Android alpha, το Νοέμβριο του 2007. Η πρώτη εμπορική έκδοση, Android 1.0, κυκλοφόρησε το Σεπτέμβριο του 2008. Το Android αναπτύσσεται συνεχώς από την Google και την κοινοπραξία Open Handset Alliance (OHA), και έχει κάνει μια σειρά από ενημερώσεις και αναβαθμίσεις για το βασικό λειτουργικό της σύστημα από την αρχική της έκδοση.

Η πιο πρόσφατη σημαντική ενημέρωση Android είναι το Android 6.0 «Marshmallow», το οποίο κυκλοφόρησε τον Οκτώβριο του 2015. Από τον Απρίλιο του 2009, οι εκδόσεις Android που αναπτύχθηκαν είχαν τη δική τους κωδική ονομασία με θέμα ζαχαροπλαστικής και κυκλοφόρησαν με αλφαβητική σειρά, αρχίζοντας με το Android 1.5 «Cupcake». Στις προηγούμενες εκδόσεις 1.0 και 1.1 δεν δόθηκαν συγκεκριμένες ονομασίες.

Παρακάτω στο γράφημα παρουσιάζονται συνολικά όλες οι εκδόσεις από την αρχή της ιστορίας τους.



Εικόνα 23: Android versions

2.5.4 Application Programming Interface (API)

Στον προγραμματισμό, μια διεπαφή προγραμματισμού εφαρμογών (API) είναι ένα σύνολο από ρουτίνες, πρωτόκολλα και εργαλεία για τη δημιουργία λογισμικού και εφαρμογών.

Ένα API εκφράζει ένα στοιχείο του λογισμικού από την άποψη των λειτουργιών του, των εισόδων του, των εξόδων του και των βασικών μορφών του. Ένα καλό API

καθιστά ευκολότερη τη δυνατότητα να αναπτυχθεί ένα πρόγραμμα, παρέχοντας όλα τα δομικά στοιχεία, τα οποία στη συνέχεια τοποθετούνται μαζί από τον προγραμματιστή.

Έτσι και το Android έχει το δικό του API χωρισμένο σε διάφορα επίπεδα με το API level 1 να αντιστοιχίζεται στην έκδοση Android 1.0. Σήμερα έχουμε φτάσει στο επίπεδο API 23 που αντιστοιχίζεται στην τελευταία έκδοση Android την 6.0-6.0.1 γνωστή και ως Marshmallow.

2.5.5 Android Studio

Στην παρούσα διπλωματική εργασία κάναμε χρήση του προγράμματος Android Studio, το οποίο μας προσέφερε το κατάλληλο προγραμματιστικό περιβάλλον, για να μπορέσουμε να υλοποιήσουμε την εφαρμογή μας, η οποία θέλαμε να τρέχει σε λογισμικό Android. Χρειάστηκαν γνώσεις και σε επίπεδο Java και σε επίπεδο Android. Έγινε χρήση των APIs που παρέχει η Google καθώς και του SDK kit που παρέχεται μέσω του Android Studio. Η εφαρμογή που υλοποιήσαμε είναι λειτουργική για συσκευές με Android API level 14 και πάνω. Αυτό σημαίνει ότι έχει δοκιμαστεί για συσκευές που έχουν από την έκδοση 4.0 και μετά. Οπότε δεν είναι σίγουρο ότι η εφαρμογή διαγνωστικών οχήματος που υλοποιήσαμε θα είναι πλήρως λειτουργική σε μοντέλα κινητών με παλιότερες εκδόσεις Android, καθώς δεν έχει δοκιμαστεί και δεν έχει φτιαχτεί για τέτοιες εκδόσεις.

2.5.6 Software development kit (SDK)

Ένα κιτ ανάπτυξης λογισμικού (SDK) είναι συνήθως ένα σύνολο εργαλείων ανάπτυξης λογισμικού που επιτρέπει τη δημιουργία εφαρμογών για ένα συγκεκριμένο πακέτο λογισμικού. Για τη δημιουργία εφαρμογών, θα πρέπει να υπάρχει κατεβασμένο ένα ειδικό κιτ ανάπτυξης λογισμικού. Για παράδειγμα, η ανάπτυξη μιας εφαρμογής Android απαιτεί SDK με Java, για iOS apps ένα SDK iOS με Swift, και για το MS Windows το .NET SDK με πλαίσιο .NET.

Μπορεί να είναι κάτι τόσο απλό, όσο η εφαρμογή ενός ή περισσοτέρων διεπαφών (API) με τη μορφή κάποιων βιβλιοθηκών για τη διασύνδεση σε μια συγκεκριμένη γλώσσα προγραμματισμού, ή να περιλαμβάνει εξελιγμένο υλικό που να μπορεί να επικοινωνεί με ένα συγκεκριμένο ενσωματωμένο σύστημα. Κοινά εργαλεία περιλαμβάνουν εγκαταστάσεις εντοπισμού σφαλμάτων και άλλα βοηθητικά προγράμματα παρουσιάζονται συχνά σε ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE). Το SDK περιλαμβάνει επίσης συχνά δείγμα

κώδικα και την υποστήριξη με τεχνικές σημειώσεις ή άλλα δικαιολογητικά, για να συμβάλει στην αποσαφήνιση σημείων από το πρωτογενές υλικό αναφοράς.

Στη δική μας περίπτωση, το Android Studio παρέχει το δικό του SDK Manager, μέσω του οποίου δίνεται η δυνατότητα για λήψη διάφορων βιβλιοθηκών σχετικά με τα API levels. Αν χρησιμοποιούσαμε άλλη πλατφόρμα, σαν το Eclipse για παράδειγμα, θα έπρεπε να εγκαταστήσουμε μόνοι μας αυτά τα εργαλεία. Χρησιμοποιήθηκε λοιπόν υλικό με δείγματα κώδικα για τις εκδόσεις KitKat, Lollipop και Marshmallow, γιατί θέλαμε η υλοποίηση της εφαρμογής να είναι λειτουργική σε όλα τα κινητά με τις τελευταίες εκδόσεις λογισμικού Android.

3

Αρχιτεκτονική του συστήματος

3.1 Εισαγωγή

Η αρχιτεκτονική του συστήματος χωρίζεται σε δύο κομμάτια, το ένα αφορά το υλικό και το άλλο το λογισμικό. Όσον αφορά το υλικό κομμάτι, έχουμε την OBD θύρα που βρίσκεται στο όχημα, στην οποία διασυνδέεται μία συσκευή elm327 που αναλαμβάνει να ανιχνεύσει το OBD πρωτόκολλο και να το ερμηνεύσει. Από την άλλη πλευρά, έχουμε μία έξυπνη συσκευή με Android (συγκεκριμένα ένα κινητό με Android) που επικοινωνεί με την elm327 μέσω Bluetooth και ανταλλάσσουν μεταξύ τους σειριακά δεδομένα. Όσον αφορά το δεύτερο κομμάτι, δηλαδή το λογισμικό, η όλη δουλειά γίνεται από την εφαρμογή που έχουμε αποθηκεύσει στην έξυπνη συσκευή μας, την οποία έχουμε προγραμματίσει να κάνει κάποιες διαδικασίες. Οι διαδικασίες αυτές σχετίζονται με την επικοινωνία, την ανταλλαγή δεδομένων και την παρουσίασή τους σε κατάλληλη φόρμα. Παρακάτω θα περιγράψουμε τη συσκευή elm327, θα παρουσιάσουμε τις έξυπνες συσκευές που χρησιμοποιήσαμε και τέλος θα αναλύσουμε τόσο το υλικό όσο και το λογισμικό κομμάτι.

3.2 Περιγραφή συσκευής ELM327

3.2.1 Ιστορικά

Στα τέλη του 1970 άρχισε η εισαγωγή των υπολογιστών-εγκεφάλων στα οχήματα. Έτσι, στις αρχές της δεκαετίας του 1980, είχε γίνει ολοένα και πιο δύσκολο για τους μηχανικούς αλλά και τους ερασιτέχνες χρήστες να εργαστούν πάνω στα δικά τους οχήματα. Καθ' όλη τη δεκαετία του 1980 και μέχρι τα μέσα της δεκαετίας του 1990, κάθε κατασκευαστής αυτοκινήτων είχε τα δικά του πρότυπα και πρωτόκολλα (Standards), και ήταν ένας πραγματικός πονοκέφαλος, ακόμη και για εξειδικευμένους τεχνικούς, να καταφέρουν να συμβαδίσουν με όλα αυτά. Αυτό άρχισε να αλλάζει με την εισαγωγή του OBD-II, το οποίο είναι ένα πρότυπο που εφαρμόστηκε από τις αυτοκινητοβιομηχανίες σε

όλο τον κόσμο, στα ερασιτεχνικά και επαγγελματικά εργαλεία σάρωσης-διάγνωσης που μπορεί να είχαν τιμή ακόμα μεγαλύτερη από δεκάδες χιλιάδες ευρώ. Μέχρι πριν από λίγα χρόνια, ακόμη και τα βασικά διαγνωστικά εργαλεία συχνά κόστιζαν εκατοντάδες ευρώ. Αυτές οι πολύ απλές συσκευές μπορούσαν να διαβάσουν και να καθαρίσουν τους κωδικούς, αλλά συνήθως δεν έδιναν καμία πρόσβαση στα PIDs που είναι τόσο χρήσιμα στη διάγνωση των προβλημάτων. Η συσκευή ELM327 (Programmed Microcontroller) είναι ένα μικρό σε μέγεθος και πολύ χαμηλού κόστους εργαλείο που βοήθησε να γεφυρωθεί αυτό το χάσμα. Οι συσκευές που χρησιμοποιούν την ELM327 ακόμα δεν κατέχουν τις δυνατότητες των επαγγελματικών εργαλείων σάρωσης, αλλά βάζουν πολλές πληροφορίες στα χέρια των χρηστών.

Ο μικροελεγκτής ELM327 λειτουργεί ως γέφυρα μεταξύ του εγκεφάλου (ECU) του αυτοκινήτου και ενός υπολογιστή (Laptop, Pc) ή μίας φορητής συσκευής (Android - iOS). Ο ELM327 είναι ικανός να επικοινωνεί με το OBD-II σύστημα του οχήματος και στη συνέχεια να εκτελεί μετεγκατάσταση των δεδομένων μέσω USB καλωδίου, WiFi ή Bluetooth (στη δική μας περίπτωση μας ένοιαζε η επικοινωνία μέσω Bluetooth), ανάλογα με τη συγκεκριμένη εφαρμογή. Υποστηρίζει έναν αριθμό διαφορετικών SAE και ISO πρωτοκόλλων, καθώς και τη δυνατότητα να επικοινωνεί με οποιοδήποτε όχημα με OBD-II υποδοχή. Το σύνολο εντολών που χρησιμοποιείται από τον ELM327 δεν είναι ταυτόσημο με το σύνολο εντολών Hayes, αλλά είναι αρκετά παρόμοιο.

3.2.2 Εντολές Hayes

Το σύνολο εντολών Hayes είναι μια συγκεκριμένη γλώσσα εντολών που αρχικά αναπτύχθηκε από τον Dennis Hayes για το Hayes Smartmodem 300 baud modem το 1981.

Το σύνολο εντολών αποτελείται από μια σειρά σύντομων συμβολοσειρών κειμένου που μπορούν να συνδυαστούν για να παράγουν εντολές για λειτουργίες, όπως κλήση, ανάρτηση, και έχουν την ικανότητα να αλλάξουν επίσης τις παραμέτρους της σύνδεσης. Η συντριπτική πλειοψηφία των dial-up μόντεμ χρησιμοποιούν εντολές Hayes σε πολλές παραλλαγές.

Το σύνολο εντολών καλύπτει μόνο τις λειτουργίες που στηρίζονται από τα πρώτα μόντεμ των 300 bit/s. Όταν νέες εντολές απαιτήθηκαν για τον έλεγχο της επιπλέον λειτουργικότητας σε υψηλότερης ταχύτητας μόντεμ, μια ποικιλία από πρότυπα προέκυψαν για να καλύψουν αυτές τις ανάγκες.

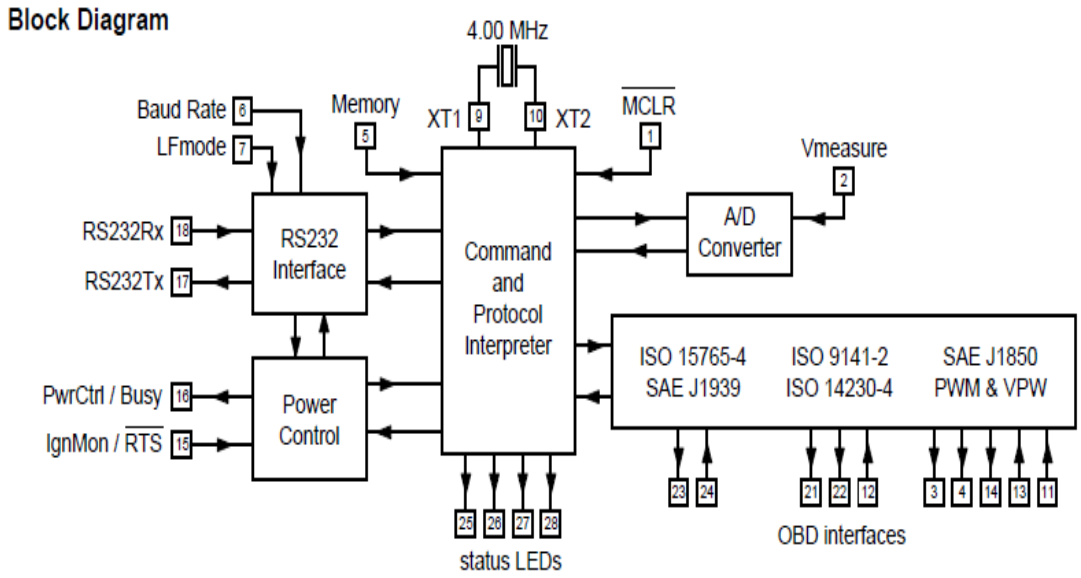
3.2.3 Γενικά

Σχεδόν όλα τα αυτοκίνητα που παράγονται σήμερα απαιτείται από το νόμο να παρέχουν μια διεπαφή για την σύνδεση του διαγνωστικού εξοπλισμού δοκιμών. Τα δεδομένα που μεταφέρονται σε αυτές τις διεπαφές ακολουθούν διάφορα πρότυπα, αλλά κανένα από αυτά δεν είναι άμεσα χρησιμοποιήσιμο από υπολογιστές ή έξυπνες συσκευές. Η συσκευή ELM327 έχει σχεδιαστεί για να λειτουργεί ως γέφυρα μεταξύ αυτών των On-Board Diagnostics θυρών (OBD) και του προτύπου RS232 σειριακής διασύνδεσης.

Εκτός του ότι είναι σε θέση να ανιχνεύει αυτόματα και να ερμηνεύει εννέα OBD πρωτόκολλα, η ELM327 επίσης παρέχει υποστήριξη για επικοινωνίες υψηλής ταχύτητας, λειτουργία χαμηλής ύπνο-εξουσίας (sleep mode) και το πρότυπο J1939 για φορτηγά και λεωφορεία. Επίσης, είναι πλήρως προσαρμόσιμη, αν θέλει κάποιος να αλλάξει κάτι, για να ταιριάζει περισσότερο στις ανάγκες του.

Αυτές οι επαφές της συσκευής υπάρχουν για διάφορους σκοπούς και χρησιμοποιούνται σε πολλές εφαρμογές αλλά δεν θα αναλυθούν στην παρούσα διπλωματική εργασία. Για αυτό το λόγο δε θα μπούμε σε πολλές τεχνικές λεπτομέρειες.

(Όποιος ενδιαφέρεται μπορεί να ανατρέξει στο έγγραφο της Elm Electronics Inc. ή στην αντίστοιχη ιστοσελίδα της www.elmelctronics.com σχετικά με τη συσκευή elm327)



Εικόνα 24: Block διάγραμμα της συσκευής elm327

3.2.4 AT εντολές

Παρακάτω παρουσιάζονται κάποιες από τις εντολές που παρέχει ο μικροελεγκτής elm327 και οι οποίες φάνηκαν χρήσιμες στη παρούσα διπλωματική εργασία.

AT Z
Κάνει επαναφορά όλων των ρυθμίσεων που είχε από τον κατασκευαστή.

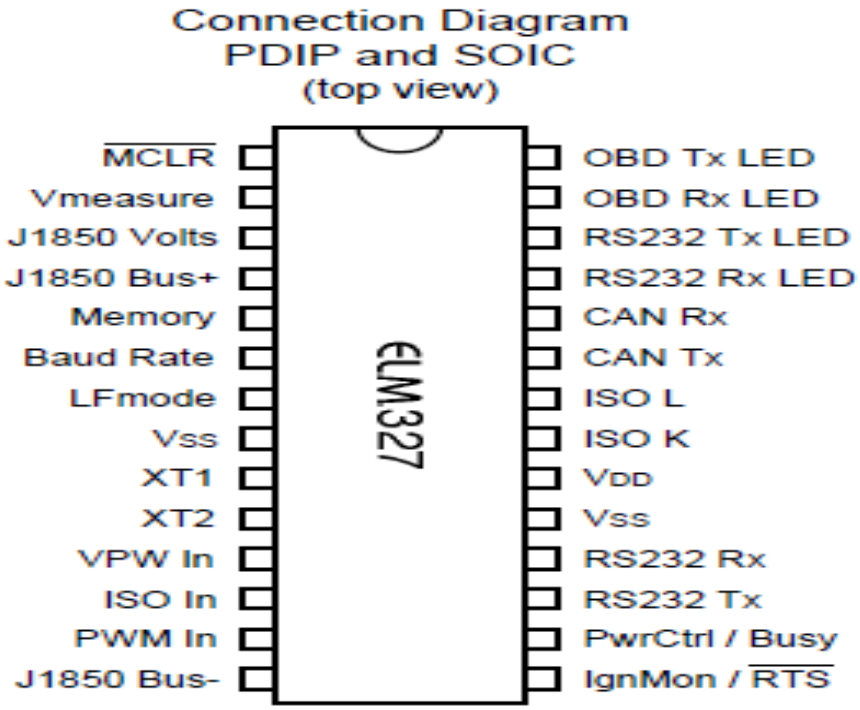
AT D
Επαναφέρει όλες τις προκαθορισμένες τιμές.

AT E0
Με αυτή την εντολή απενεργοποιείται η ηχώ.

AT DP
Περιγράφει το πρωτόκολλο που χρησιμοποιείται στο τρέχον όχημα.

AT SP 0
Αφού δοκιμάσει τα πρωτόκολλα με τη σειρά, βρίσκει ποιο χρησιμοποιεί το τρέχον όχημα, το σώζει και το χρησιμοποιεί αυτόματα.

Τις παραπάνω εντολές τις χρησιμοποιήσαμε, είτε μέσω τερματικού για να δοκιμάσουμε την επικοινωνία με την elm327, είτε στον κώδικα της εφαρμογής για να κάνουμε κάποιες αρχικοποιήσεις στη σύνδεση.



Εικόνα 25: Διάγραμμα των pins της συσκευής elm327

3.2.5 Η επικοινωνία με τη συσκευή ELM327

Η συσκευή ELM327 αναμένει να επικοινωνήσει με έναν υπολογιστή μέσω σειριακής σύνδεσης RS232. Παρόλο που οι σύγχρονοι υπολογιστές δεν παρέχουν συνήθως μια τέτοια σειριακή σύνδεση, υπάρχουν διάφοροι τρόποι με τους οποίους μπορούμε να δημιουργήσουμε μια τέτοια «εικονική σειριακή θύρα». Οι πιο συνηθισμένες συσκευές είναι USB με προσαρμογείς RS232, αλλά υπάρχουν αρκετές άλλες, όπως κάρτες PC, συσκευές Ethernet, ή Bluetooth για σειριακή προσαρμογή.

Από τη στιγμή που θα λυθεί ο τρόπος σύνδεσης, όσον αφορά την πλευρά του υλικού κομματιού, χρειαζόμαστε και ένα τερματικό, για να στέλνουμε και να λαμβάνουμε δεδομένα. Η απλούστερη μέθοδος είναι η χρήση ενός προγράμματος από τα πολλά «τερματικά» που είναι διαθέσιμα (όπως το HyperTerminal, ZTerm, Tera Term). Με αυτό τον τρόπο, θα έχουμε τη δυνατότητα να πληκτρολογήσουμε άμεσα τους χαρακτήρες από το πληκτρολόγιό μας. (Στην παρούσα διπλωματική εργασία χρησιμοποιήσαμε το TeraTerm για τις δοκιμές μας και για να μπορούμε να ανταλλάξουμε σειριακά δεδομένα μέσω Bluetooth.)

Από τη στιγμή που είχαμε εγκαταστήσει μία σύνδεση μέσω Bluetooth, μπορούσαμε να στείλουμε σειριακά δεδομένα μέσω του τερματικού και να λάβουμε απαντήσεις από τη συσκευή elm327. Στις δοκιμές, που έγιναν, στείλαμε, για παράδειγμα, «AT Z» μέσω του τερματικού Tera Term και λάβαμε ως απάντηση «OK», που σημαίνει ότι έκανε επαναφορά όλες τις ρυθμίσεις. Ένα δεύτερο παράδειγμα είναι, όταν στείλαμε «010C» τη στιγμή που το όχημα ήταν ακινητοποιημένο και είχαμε μόνο την μπαταρία της μηχανής ανοικτή, λάβαμε ως απάντηση «41 0C 00 00», που σημαίνει ότι οι στροφές κινητήρα είναι μηδέν (λογικό, αφού το όχημα ήταν σβηστό).

3.2.6 Περιγραφή του RS-232

Στον τομέα των τηλεπικοινωνιών, το RS-232 είναι ένα πρότυπο για τη σειριακή μετάδοση δεδομένων. Ορίζει επίσημα τα σήματα που συνδέονται μεταξύ ενός DTE (τερματικός εξοπλισμός δεδομένων), όπως είναι ένα τερματικό υπολογιστή, και ενός DCE (εξοπλισμός για επικοινωνία και μεταφορά δεδομένων), όπως είναι ένα μόντεμ. Το πρότυπο RS-232 χρησιμοποιείται συνήθως στον υπολογιστή για τις σειριακές του θύρες. Το πρότυπο καθορίζει τα ηλεκτρικά χαρακτηριστικά και το χρονοδιάγραμμα των σημάτων, τη σημασία των σημάτων, καθώς και το φυσικό μέγεθος και τις επαφές των συνδεόμενων συσκευών.

3.3 Περιγραφή των Android συσκευών που χρησιμοποιήθηκαν

Στην παρούσα διπλωματική εργασία για την υλοποίηση της εφαρμογής διαγνωστικών εντός οχήματος χρησιμοποιήθηκαν δυο συγκεκριμένες έξυπνες συσκευές (smartphones) με λογισμικό Android για την εκτέλεση και τις δοκιμές, έως ότου η εφαρμογή (application) να γίνει λειτουργική για τον σκοπό μας. Στην πρώτη φάση χρησιμοποιήθηκε το Samsung Galaxy Core I8262 και στη συνέχεια το LG G3.

Προφανώς, το δεύτερο μας έδωσε περισσότερες δυνατότητες τόσο λόγω του πιο καινούριου λογισμικού Android του όσο και λόγω της αρχιτεκτονικής του (καλύτερο επεξεργαστή, μεγαλύτερη RAM).

3.3.1 Samsung Galaxy Core I8262

Το πρώτο είχε τα εξής βασικά χαρακτηριστικά:

- ✓ Τεχνολογία: GSM/HSPA
- ✓ Λειτουργικό Σύστημα: Android OS, v4.1.2 (Jelly Bean)
- ✓ Chipset: Qualcomm MSM8225 Snapdragon S4 Play
- ✓ CPU: Dual-core 1.2 GHz Cortex-A5
- ✓ GPU: Adreno 203
- ✓ Μνήμη: 8 GB (4.5 GB user available), 1 GB RAM
- ✓ Bluetooth: v3.0, A2DP
- ✓ Πίσω (Βασική) Κάμερα: 5 MP, f/2.6, autofocus, LED flash, video 480p@30fps

Εικόνα 26: Samsung Galaxy Core I8262

3.3.2 LG G3

Αντίθετα το δεύτερο είχε τα εξής βασικά χαρακτηριστικά:

- ✓ Τεχνολογία: GSM/HSPA/LTE
- ✓ Λειτουργικό Σύστημα: Android OS, v5.0.2 (Lollipop)
- ✓ Chipset: Qualcomm MSM8974AC Snapdragon 801
- ✓ CPU: Quad-core 2.5 GHz Krait 400
- ✓ GPU: Adreno 330
- ✓ Μνήμη: 32 GB, 3 GB RAM
- ✓ Bluetooth: v4.0, A2DP, LE, apt-X
- ✓ Πίσω (Βασική) Κάμερα: 13 MP, f/2.4, 29mm, phase detection/laser autofocus, OIS, dual-LED (dual tone) flash, video 2160p@30fps, 1080p@30fps, HDR, stereo sound rec.

Εικόνα 27: LG G3

Κατά τη διάρκεια της υλοποίησης της παρούσας διπλωματικής εργασίας, παρατηρήθηκαν αρκετές διαφοροποιήσεις για την εφαρμογή από το ένα μοντέλο κινητού στο άλλο. Τα στοιχεία που έπαιξαν σημαντικό ρόλο για τις αλλαγές αυτές ήταν:

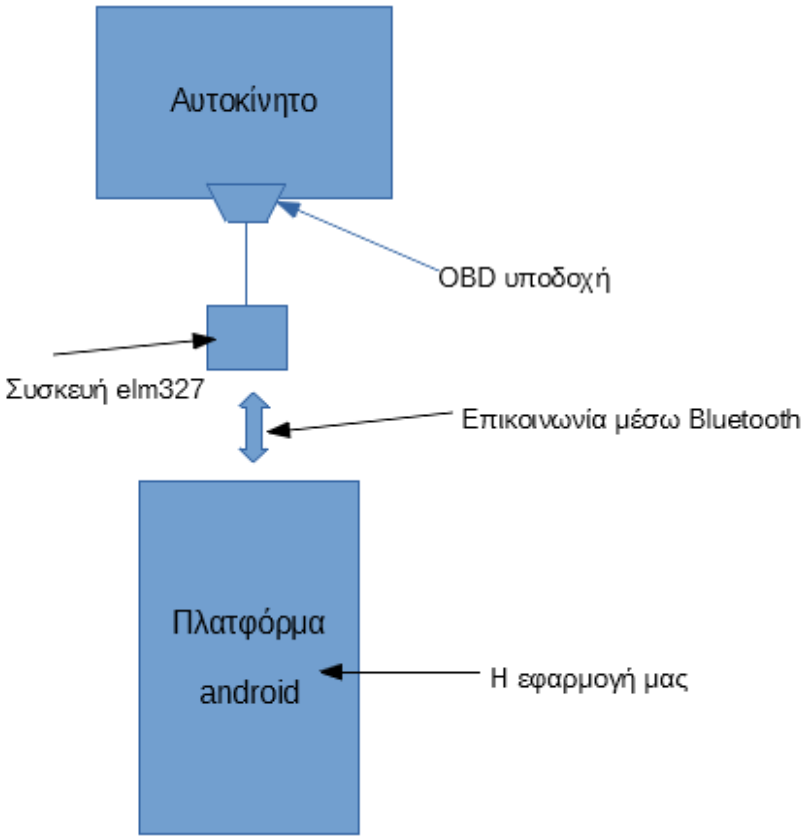
- ✓ Μνήμη RAM
- ✓ Επεξεργαστής
- ✓ Έκδοση λειτουργικού συστήματος Android
- ✓ Bluetooth

Τα παραπάνω στοιχεία προσέφεραν:

- ✓ Ταχύτητα
- ✓ Λιγότερα crash της εφαρμογής
- ✓ Λιγότερα bugs στην εφαρμογή
- ✓ Αξιοπιστία στη λειτουργία του Bluetooth

Για αυτό, σαν συμπέρασμα θα μπορούσαμε να πούμε ότι είναι καλύτερο να μπορούν να γίνουν δοκιμές με όσο πιο τελευταίας τεχνολογίας έξυπνες συσκευές γίνεται και λόγω των δυνατοτήτων που δίνουν αλλά και λόγω της ποιοτικής ανάλυσης που μπορούν να έχουν.

3.4 Σχεδιάγραμμα και περιγραφή αρχιτεκτονικής hardware

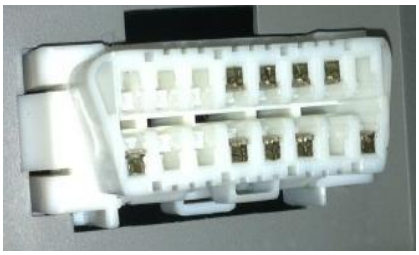


Εικόνα 28: Σχεδιάγραμμα της αρχιτεκτονικής του υλικού (hardware)

Στην παραπάνω εικόνα φαίνεται η αρχιτεκτονική του υλικού που αξιοποιήθηκε στην παρούσα διπλωματική εργασία. Όπως είναι φανερό, το κάθε όχημα έχει μία OBD υποδοχή, η οποία συνήθως βρίσκεται στη θέση του οδηγού κάτω από το τιμόνι. Η ακριβής θέση διαφέρει από όχημα σε όχημα. Η εφαρμογή δοκιμάστηκε σε τέσσερα διαφορετικά οχήματα και η θέση όντως διέφερε. Σε αυτή την υποδοχή συνδέουμε τη συσκευή elm327, η οποία έχει την ικανότητα να διαβάσει και να ερμηνεύσει σχεδόν όλα τα πρωτόκολλα OBD-II. Από την άλλη πλευρά έχουμε την έξυπνη συσκευή (στη συγκεκριμένη περίπτωση ένα κινητό με Android λειτουργικό σύστημα) και με την εφαρμογή που έχουμε υλοποιήσει, είμαστε σε θέση να πάρουμε δεδομένα σε πραγματικό χρόνο. Η επικοινωνία και η μεταφορά δεδομένων γίνεται μέσω Bluetooth.

Στην παρούσα διπλωματική εργασία, στις δοκιμές που έγιναν προκειμένου να υλοποιηθεί η συγκεκριμένη εφαρμογή χρησιμοποιήθηκαν τα εξής:

- ✓ Δύο υπολογιστές (για συγγραφή του κώδικα και προσομοίωση της elm327 μέσω τερματικού)
- ✓ Δυο Android συσκευές (Samsung galaxy Core I8262 και LG G3)
- ✓ Δυο διαφορετικές συσκευές elm327 (obd2ecu και Vgate)
- ✓ Τέσσερα διαφορετικά οχήματα
- ✓ Ένας Bluetooth αντάπτορας



Εικόνα 29: OBD υποδοχή

Εικόνα 30: Συσκευή elm327

3.5 Σχεδιάγραμμα και περιγραφή αρχιτεκτονικής software

Από την πλευρά του λογισμικού, η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για τη υλοποίηση της εφαρμογής Android ήταν η Java, γιατί κυρίως σε αυτή υλοποιούνται οι περισσότερες εφαρμογές. Η κύρια διεργασία χωρίστηκε σε τέσσερα βασικά μέρη.

3.5.1 Μέθοδος διεπαφής χρήστη και μέθοδος Bluetooth

Σε αυτή τη μέθοδο γίνεται η διασύνδεση με την διεπαφή χρήστη (user interface), η οποία έχει σχεδιαστεί σε ένα αρχείο μορφής xml (το οποίο παραθέτουμε στο Παράρτημα

Β). Καταρχήν έχουμε προγραμματίσει την εφαρμογή μας έτσι ώστε να ενεργοποιεί το Bluetooth της (αν δεν είναι ήδη ανοιχτό) και να ψάχνει να εντοπίσει άλλες συσκευές με Bluetooth στο γύρω χώρο. Αφού τις βρει, κάνει το ζευγάρωμα των συσκευών (pairing λέγεται η διαδικασία αυτή) και τις αποθηκεύουμε σε μία λίστα. Στην ίδια μέθοδο έχουν συμπεριληφθεί κάποιες επιπλέον ενέργειες και συναρτήσεις. Για παράδειγμα, μία ενέργεια έχει να κάνει με το γεγονός ότι θέλουμε η λίστα, στην οποία αποθηκεύουμε τις συσκευές με Bluetooth, να δείχνει τα στοιχεία με άσπρο χρώμα, γιατί έχουμε μαύρη εικόνα στη διεπαφή χρήστη και επομένως, για να βλέπουμε ποιες συσκευές έχουν στην κοντινή μας περιοχή Bluetooth, πρέπει να μπορούμε να τις αναγνώσουμε. Στην ίδια μέθοδο κάνουμε και διάφορες αρχικοποιήσεις μεταβλητών που θα χρησιμοποιηθούν στην πορεία.

3.5.2 Μέθοδος δημιουργίας αρχείου και μέθοδος αποθήκευσης δεδομένων

Σε αυτές τις δύο μεθόδους δημιουργούμε το αρχείο στην SD card του κινητού στο κατάλληλο path, αν δεν υπάρχει ήδη, και στη συνέχεια, με τη δεύτερη μέθοδο αποθηκεύουμε τα δεδομένα σε κατάλληλη μορφή στο αρχείο αυτό. Την πρώτη μέθοδο την καλούμε, όταν πατάμε το κουμπί Start, ενώ τη δεύτερη την καλούμε στον Handler που κάνουμε την επεξεργασία των δεδομένων. Και τα δύο αυτά σημεία παρακάτω θα αναλυθούν περαιτέρω.

3.5.3 Μέθοδος δημιουργίας καναλιού

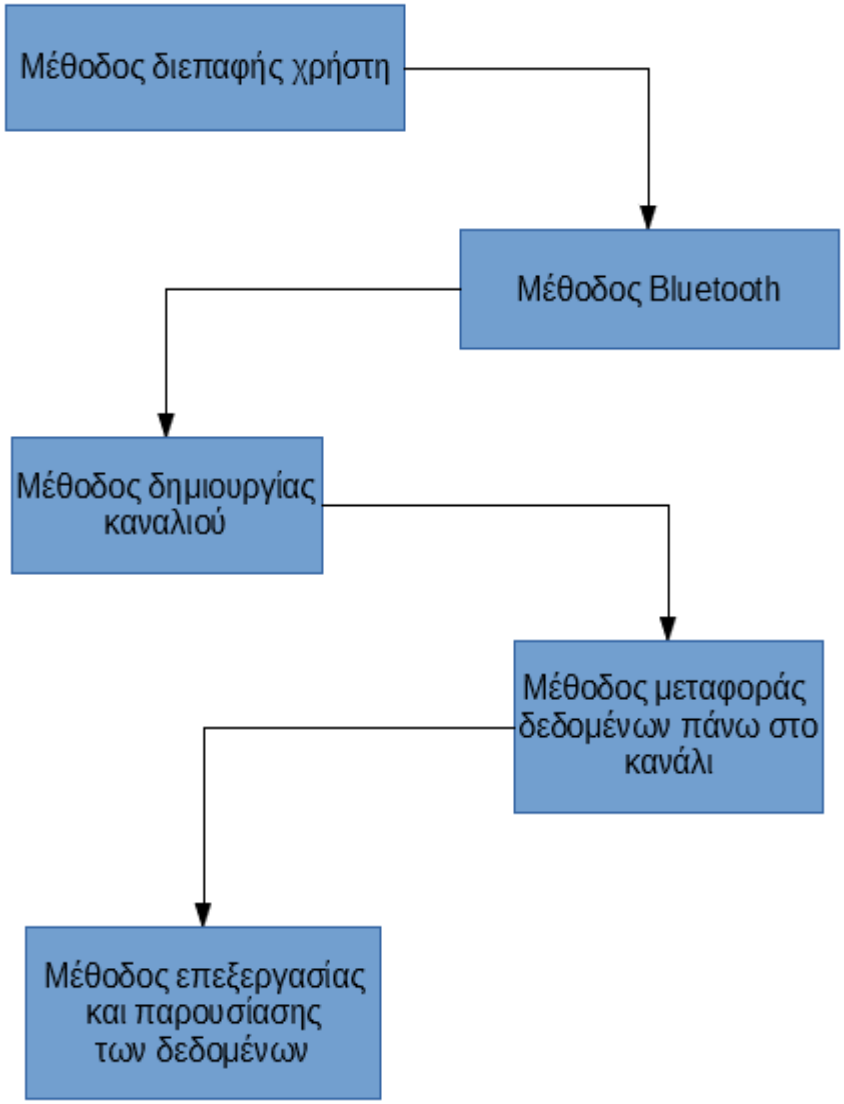
Στη συνέχεια χρησιμοποιούμε μία μέθοδο για να διαχειριστούμε το κουμπί (button Start) που έχουμε βάλει στη διεπαφή χρήστη. Έχουμε προγραμματίσει με τέτοιο τρόπο το κουμπί ώστε, όταν το πατάμε να ξεκινάει η διαδικασία για τη διασύνδεση της συσκευής elm327 και της εφαρμογής που έχουμε στο Android κινητό μας. Με το που πατηθεί το κουμπί λοιπόν η εφαρμογή αρχίζει και ψάχνει στη λίστα τη συσκευή elm327 που έχουμε χρησιμοποιήσει (δύο συσκευές χρησιμοποιήθηκαν κατά τη διάρκεια των δοκιμών με ονόματα obd2ecu και Vgate αντίστοιχα). Επίσης για τον έλεγχο των τιμών που λαμβάνει από τον OBD αντάπτορα κατά τη διάρκεια των δοκιμών έχουμε ρυθμίσει την εφαρμογή να δημιουργεί και να σώζει αυτές τις τιμές σε ένα αρχείο μορφής txt μέσα στη μνήμη του κινητού. Αυτό το κάνουμε για να μπορούμε να ελέγχουμε τι μας στέλνει η συσκευή elm327 και να μπορούμε να εντοπίσουμε τυχόν σφάλματα. [γραμμές 364-387]

3.5.4 Μέθοδος επεξεργασίας και παρουσίασης δεδομένων

Όταν το κανάλι και η σύνδεση είναι έτοιμα, ένα άλλο κομμάτι κώδικα (thread) αναλαμβάνει να κάνει δύο εργασίες. Η μία αφορά την αποστολή των αιτήσεων (requests)

STOPPED >105 7F 10 12 STOPPED
>15A STOPPED >15E 7F 15 11
21 - 11 - 53 : STOPPED >105 7F 10 12
> 7F 10 12 >015A >10D 7F 10 12
STOPPED >105 7F 10 12 STOPPED
>15A STOPPED >15E 7F 15 11
21 - 11 - 53 : 015A ED >105 7F 10 12 :
7F 10 12 >015A >10D 7F 10 12
STOPPED >105 7F 10 12 STOPPED
>15A STOPPED >15E 7F 15 11
21 - 11 - 53 : STOPPED >15E 7F 15 11

Εικόνα 31: Δείγμα του αρχείου που κρατάει τα δεδομένα



Εικόνα 32: Απλοποιημένο διάγραμμα ροής του λογισμικού (software)

των κωδικών διάγνωσης εντός οχήματος (PIDs) που έχουμε ορίσει και η δεύτερη αναλαμβάνει τις απαντήσεις (answers) σε αυτές τις αιτήσεις. Σε αυτό το τελευταίο κομμάτι γίνεται και η επεξεργασία των απαντήσεων μέσω ενός διαχειριστή (handler), ώστε από το δεκαεξαδικό σύστημα (hex) να μεταφράζονται κατάλληλα σε δεκαδικό σύστημα (decimal) και να παρουσιάζονται μέσω της εφαρμογής στη διεπαφή χρήστη που έχουμε φτιάξει. Η ρύθμιση έχει γίνει έτσι ώστε να στέλνονται δύο αιτήσεις μέσα σε ένα δευτερόλεπτο για την κάθε τιμή. Για παράδειγμα, στέλνει 010C και παίρνει απάντηση 41 0C 00 00, που αυτό μεταφράζεται 0 rpm για τις στροφές κινητήρα. Ελέγχουμε αν το πρώτο byte είναι 41, γιατί τότε γνωρίζουμε ότι είναι απάντηση στον 1^{ο} τρόπο λειτουργίας του OBD. Μετά ελέγχουμε το 2^{ο} byte (εδώ είναι 0C) και καταλαβαίνουμε ότι αναφέρεται σε στροφές κινητήρα. Οπότε, τα δύο επόμενα bytes μας δίνουν την ακριβή τιμή (εδώ 0). Αντίστοιχα, το ίδιο γίνεται για όλες τις τιμές. Σημαντική παρατήρηση είναι ότι τα πράγματα δεν είναι τόσο απλά όσο τα περιγράφουμε παραπάνω, γιατί η επικοινωνία είναι ασύγχρονη και τα γεγονότα δεν είναι συγχρονισμένα. Δηλαδή, για να γίνει αντιληπτό, αυτό σημαίνει ότι μπορεί να στέλνουμε αίτηση για μια άλλη τιμή και να λαμβάνουμε απάντηση για την προηγούμενη τιμή ακόμα. Αλλά βέβαια όλα αυτά έχουν ληφθεί υπόψη στον κώδικα και έχουν παραμετροποιηθεί. [ConnectedThread: γραμμές 430-544 και Handler: γραμμές 90-207]

3.5.5 Μεταβλητές παρουσίασης

Τέλος, να αναφέρουμε ότι στην εφαρμογή που υλοποιήσαμε παίρνουμε τιμές για πέντε μεταβλητές. Αυτές είναι ταχύτητα οχήματος, στροφές κινητήρα, θερμοκρασία μηχανής, κατανάλωση καυσίμου και σχετική θέση του πεντάλ του γκαζιού. Η εφαρμογή δοκιμάστηκε σε τέσσερα οχήματα και έτυχε και στα τέσσερα να μη λαμβάνουμε τιμές για τις δυο μεταβλητές (κατανάλωση καυσίμου και σχετική θέση του πεντάλ του γκαζιού) και αυτό σχετίζεται με τον κατασκευαστή, γιατί δεν έχουν όλα τα οχήματα όλα τα PIDs (μόνο αυτά που έχει ορίσει ο κατασκευαστής). Όμως έχουμε ρυθμίσει την εφαρμογή να λαμβάνει υπόψη της τέτοιες περιπτώσεις και να εμφανίζει «NO DATA» για τις αντίστοιχες μεταβλητές.

Αξίζει να αναφέρουμε τα PIDs για τη κάθε μεταβλητή που χρησιμοποιήσαμε και μερικές από τις απαντήσεις που πήραμε και πώς αυτές μεταφράζονται. Το πρώτο byte αφορά τον τρόπο λειτουργίας του OBD, το δεύτερο αφορά τη μεταβλητή και τα υπόλοιπα είναι η τιμή της μεταβλητής. Έτσι έχουμε:

- Ταχύτητα οχήματος > 010D

    - 41 0D 00 > 0 km/h
    - 41 0D 15 > 21 km/h
    - 41 0D 50 > 80 km/h
    - 41 0D 70 > 112 km/h
    - 41 0D AF > 175 km/h

- Στροφές κινητήρα > 010C

    - 41 0C 00 00 > 0 rpm
    - 41 0C 05 00 > 1280rpm
    - 41 0C 03 AF > 943 rpm
    - 41 0C 09 13 > 2323 rpm
    - 41 0C 11 11 > 4369 rpm

- Θερμοκρασία μηχανής > 0105

    - 41 05 50 > 40 °C
    - 41 05 62 > 58 °C
    - 41 05 6E > 70 °C
    - 41 05 71 > 73 °C
    - 41 05 7A > 82 °C

- Σχετική θέση του πεντάλ του γκαζιού > 015A

    - 41 5A 34 > 20%
    - 41 5A 5F > 37%
    - 41 5A 6A > 41%
    - 41 5A 72 > 44 %
    - 41 5A 83 > 51%

- Ρυθμός καυσίμου μηχανής > 015E

    - 41 5E 00 00 > 0 L/h
    - 41 5E 00 15 > 1.05 L/h
    - 41 5E 00 50 > 4 L/h
    - 41 5E 15 50 > 272.8 L/h
    - 41 5E 50 70 > 1029.6 L/h

Όσον αφορά τις δύο τελευταίες μεταβλητές βέβαια, τις έχουμε ελέγξει μόνο μέσω του τερματικού από υπολογιστή, γιατί στην πράξη κανένα από τα τέσσερα οχήματα στα οποία έχουμε δοκιμάσει την εφαρμογή δεν έστελνε δεδομένα σχετικά με αυτές και αυτό συνέβη λόγω του κατασκευαστή τους. Αντίθετα, όπως προαναφέραμε, σε αυτή την περίπτωση έχουμε προγραμματίσει την εφαρμογή να δείχνει τη φράση «NO DATA» και έτσι καταλαβαίνουμε ότι για αυτές τις μεταβλητές δε λαμβάνουμε δεδομένα.

4

Επίλογος

Στο παρόν κεφάλαιο, αρχικά συνοψίζεται η έως τώρα πορεία μας και στη συνέχεια παρουσιάζονται τα εμπόδια που ξεπεράστηκαν κατά τη διάρκεια της υλοποίησης της εφαρμογής καθώς και οι τρόποι με τους οποίους αυτά αντιμετωπίστηκαν. Έπειτα, γίνεται μια συνολική αξιολόγηση και αναλύονται τα πιο σημαντικά συμπεράσματα στα οποία καταλήξαμε στην παρούσα διπλωματική εργασία. Τέλος, αναφέρονται οι κατευθύνσεις οι οποίες θα μπορούσαν να αποτελέσουν αξιόλογο υλικό για περαιτέρω έρευνα.

4.1 Σύνοψη

Στην αρχή της εργασίας παρουσιάσαμε μία ιστορική αναδρομή για τις έξυπνες συσκευές από την εμφάνισή τους μέχρι σήμερα καθώς επίσης και μία ανασκόπηση-αναδρομή στα διαγνωστικά συστήματα εντός οχήματος (OBD-II).

Στη συνέχεια, δείξαμε κάποιες πτυχές στις οποίες η παρούσα διπλωματική εργασία ενδεχομένως να έβρισκε εφαρμογή στην πράξη και εξηγήσαμε τον τρόπο με τον οποίο καταφέραμε τελικά να την υλοποιήσουμε. Περιγράψαμε επίσης το σύστημα OBD και αναφερθήκαμε στο κινητό λειτουργικό σύστημα Android.

Στο κύριο μέρος της εργασίας, περιγράψαμε τη συσκευή elm327 καθώς και τον τρόπο με τον οποίο αυτή λειτουργεί και ερμηνεύει τα διαφορετικά OBD πρωτόκολλα. Η εφαρμογή δοκιμάστηκε σε δύο κινητά με διαφορετική έκδοση Android. Τέλος, αναφερθήκαμε στο υλικό κομμάτι που χρησιμοποιήθηκε καθώς και στο λογισμικό με το οποίο επιτεύχθηκε η εφαρμογή.

Έτσι, παρακάτω θα αναφέρουμε τις δυσκολίες που συναντήσαμε κατά την εκπόνηση αυτής της εργασίας και θα καταλήξουμε σε κάποια χρήσιμα συμπεράσματα.

4.2 Αντιμετώπιση Προβλημάτων

Κατά τη διάρκεια υλοποίησης της παρούσας διπλωματικής εργασίας παρουσιάστηκαν κάποιες δυσκολίες, οι οποίες έπρεπε να αντιμετωπιστούν. Σε αυτή την παράγραφο θα τις παρουσιάσουμε και θα αναλύσουμε με ποιους τρόπους ξεπεράστηκαν.

1. Σημαντική απόφαση έπρεπε να ληφθεί για την παρουσίαση της διεπαφής χρήστη της εφαρμογής. Το δίλημμα ήταν αν η παρουσίαση των τιμών θα έπρεπε να είναι σε απλό κείμενο (texts) ή σε κάτι πιο περίπλοκο. Τελικά, μία εικόνα ήταν αρκετή για το πίσω επίπεδο (background) και πάνω της σε πέντε διαφορετικά σημεία συμμετρικά παρουσιάζονται οι τιμές.

2. Ένα άλλο εμπόδιο που παρουσιάστηκε κατά τη διάρκεια υλοποίησης της παρούσας εφαρμογής ήταν το γεγονός ότι το πρώτο κινητό (Samsung Galaxy Core I8262) είχε πιο παλιά έκδοση λογισμικού (Android 4.1.2) και παρουσίαζε διάφορα σφάλματα. Όλα αυτά επιλύθηκαν, όταν αρχίσαμε να κάνουμε τις δοκιμές μας στο πιο καινούριο κινητό (LG G3) με την πιο ανανεωμένη έκδοση λογισμικού που παρείχε.

3. Το αμέσως επόμενο πρόβλημα που αντιμετωπίστηκε ήταν η διαδικασία επεξεργασίας των δεδομένων. Έπρεπε να πάρουμε τα bytes που στέλνονταν, να τα μετατρέψουμε σε string και να το σπάσουμε σε κομμάτια. Έπειτα, αυτά τα κομμάτια θέλαμε να τα κάνουμε ακεραίους, να τα επεξεργαστούμε και να τα μετατρέψουμε σε δεκαδικό σύστημα. Τέλος, έπρεπε να παρουσιάσουμε τις τιμές σε μορφή string και πάλι. Οπότε, έπρεπε να βρούμε τον κατάλληλο τρόπο υλοποίησης αυτής της διαδικασίας μέσω της γλώσσας προγραμματισμού Java.

4. Αφού ξεπεράσαμε το προηγούμενο εμπόδιο, έπρεπε να δούμε πώς θα κάνουμε την επικοινωνία των συσκευών μέσω Bluetooth. Αυτό ήταν ένα αρκετά δύσκολο κομμάτι, γιατί, ενώ υπάρχουν δείγματα κώδικα πάνω σε αυτό, δεν είναι πάντα λειτουργικά για όλες τις περιπτώσεις. Σε αυτό το κομμάτι έγιναν αρκετές δοκιμές μέσω του τερματικού Tera Term από υπολογιστή.

5. Το πιο δύσκολο μέρος όμως ήταν η ανταλλαγή δεδομένων. Στην αρχή, είχαμε προσομοιώσει την πλευρά της συσκευής elm327 μέσω του τερματικού. Όμως, δεν

είναι το ίδιο εύκολο να στέλνουμε τις απαντήσεις χειροκίνητα με το να στέλνονται αυτόματα, όταν εντοπίζεται κάποια αίτηση, όπως κάνει η συσκευή elm327.

6. Τέλος, είχαμε να αντιμετωπίσουμε διάφορα σφάλματα που είχαν να κάνουν, είτε με τον buffer, είτε με τις απαντήσεις στις αιτήσεις. Αυτά τα προβλήματα λύθηκαν κυρίως με παραμετροποίηση των τιμών.

4.3 Αξιολόγηση

Ήρθε η ώρα να γίνει μια συνολική αξιολόγηση της εφαρμογής που υλοποιήθηκε. Η εφαρμογή αυτή είναι ό,τι πρέπει για μια πρώτη γεύση με το Android και με τη Java, καθώς είναι αρκετά λειτουργική και απλή. Από την άλλη πλευρά όμως, η απλότητα αυτή δεν είναι και τόσο θετική για να έχουμε ευρεία χρήση της εφαρμογής. Θα έπρεπε να γίνουν βελτιώσεις κυρίως στο λογισμικό κομμάτι, όπως για παράδειγμα να περιλαμβάνονται περισσότερες τιμές ή να δείχνει τα αποτελέσματα με διαφορετικό τρόπο και σίγουρα να έχει περισσότερες επιλογές, ώστε οποιοσδήποτε χρήστης, ή ιδιώτης ή μηχανικός αυτοκινήτων ή εταιρία, να είναι ικανοποιημένος.

Άρα, συνολικά η εφαρμογή είναι αρκετά λειτουργική και εκτελεί την εργασία που της έχουμε αναθέσει, δηλαδή να ανταλλάσσει σειριακά δεδομένα μέσω Bluetooth με τη συσκευή elm327. Για να έχουμε όμως μια πιο ευρεία χρήση της, υπάρχουν πολλά σημεία στα οποία θα μπορούσε να επεκταθεί μελλοντικά και μερικά από αυτά αναφέρονται παρακάτω.

4.4 Συμπεράσματα

Από την παρούσα διπλωματική εργασία προέκυψαν κάποια ενδιαφέροντα συμπεράσματα:

1. Ένα από αυτά είναι το γεγονός ότι το Android διατίθεται ελεύθερα και, μέσω του Παγκόσμιου ιστού, ο καθένας έχει πρόσβαση σε αυτό και αποκτά ταυτόχρονα την ικανότητα να φτιάξει τις δικές του εφαρμογές για συσκευές που χρησιμοποιούν Android για το λειτουργικό τους σύστημα. Στη συγκεκριμένη περίπτωση, είχαμε παραδείγματα για το πώς λειτουργεί η διεπαφή Bluetooth, κάτι που ήταν ζωτικής

σημασίας για την παρούσα διπλωματική καθώς η όλη επικοινωνία βασιζόταν σε αυτό.

2. Εκτός αυτού, μέσα από όλη αυτή τη διαδικασία διαπιστώνει κανείς ότι η Java είναι μία σημαντική γλώσσα προγραμματισμού, γιατί είναι αντικειμενοστραφής και χρησιμοποιείται σε πάρα πολλές συσκευές. Μπορεί να διαχειριστεί αντικείμενα και να ορίσει τι θα κάνει το καθένα από αυτά. Για παράδειγμα, το κουμπί Start, που τοποθετήθηκε στην εφαρμογή, έχει προγραμματιστεί έτσι ώστε, με το που θα πατηθεί, να ξεκινάει τη διαδικασία σύνδεσης με τη συσκευή elm327 και να σχηματίζει το κανάλι πάνω στο οποίο θα μεταφερθούν τα δεδομένα.

3. Ένα άλλο χρήσιμο συμπέρασμα είναι ότι τα διαγνωστικά εντός οχήματος θα παίξουν ακόμα μεγαλύτερο ρόλο στο εγγύς μέλλον (ήδη είναι απαραίτητα) και ως εκ τούτου εφαρμογές, σαν αυτή που υλοποιήθηκε στην παρούσα διπλωματική εργασία, θα είναι αρκετά χρήσιμες.

4. Επίσης, πρέπει να αναφερθεί ότι, αν και η εφαρμογή είναι πλήρως λειτουργική, για να έχει ευρεία χρήση θα χρειαζόταν αρκετές αλλαγές στο κομμάτι του κώδικα, ώστε να δίνει περισσότερες επιλογές και να μειωθούν τυχόν σφάλματα.

4.5 Μελλοντικές επεκτάσεις

Με βάση όσα έχουν προαναφερθεί, γίνεται αντιληπτό ότι υπάρχουν κάποια σημεία τα οποία θα μπορούσαν να επεκταθούν ή να ακολουθηθεί μια διαφορετική αντιμετώπιση-προσέγγιση και επομένως είναι άξια σχολιασμού.

- Πρώτα από όλα, θα μπορούσε να υλοποιηθεί μία πιο ωραία σχεδιαστικά διεπαφή χρήστη, η οποία να δίνει και περισσότερες επιλογές ως προς τις μεταβλητές που θα μπορούσε να λαμβάνει από τη συσκευή elm327. Επίσης, αντί για κουμπί θα μπορούσε να εμφανίζει μία λίστα με όλες τις συσκευές που έχουν Bluetooth στη γύρω περιοχή και να πατάμε πάνω στο όνομα της συσκευής elm327, ώστε να ξεκινάει η διαδικασία σύνδεσης και η λήψη των απαντήσεων κατά συνέπεια.

- Δεύτερον, μια άλλη επέκταση της παρούσας εφαρμογής αφορά τον τρόπο λειτουργίας του OBD για τον οποίο λαμβάνει τιμές. Εμείς χρησιμοποιήσαμε μόνο τον πρώτο τρόπο λειτουργίας. Ο πρώτος τρόπος λειτουργίας αφορά τις τρέχουσες τιμές του οχήματος (λαμβάνει δεδομένα για την ταχύτητα, για παράδειγμα, κατά την κίνηση του οχήματος). Οπότε, μπορεί μελλοντικά να ενισχυθεί και επομένως να στέλνονται και PIDs, με σκοπό την εξακρίβωση τυχόν σφαλμάτων εντός του οχήματος. Ή ακόμα και να επεκταθεί και για τους υπόλοιπους τρόπους λειτουργίας που έχει το OBD.

- Επίσης, μία κατεύθυνση πάνω στην οποία θα μπορούσε να γίνει μελέτη είναι να βρεθεί τρόπος να στέλνονται οι αιτήσεις και να λαμβάνονται οι απαντήσεις με συγχρονισμένο τρόπο. Αυτό σημαίνει ότι θα στέλνουμε μια αίτηση και δεν θα στέλνουμε την επόμενη, αν δεν έρθει πρώτα η απάντηση στην προηγουμένη αίτηση. Αντίθετα, στην υλοποίηση της παρούσας διπλωματικής εργασίας η εφαρμογή μας στέλνει αιτήσεις και λαμβάνει απαντήσεις με ασύγχρονο τρόπο, δηλαδή δεν περιμένουμε. Στέλνουμε αιτήσεις συνεχόμενα και λαμβάνουμε απαντήσεις. Οπότε, δεν περιμένουμε να έρθει η απάντηση, για να στείλουμε την επόμενη αίτηση. Μια σκέψη για την υλοποίηση του συγχρονισμού είναι να κρατάμε κάποιες «σημαίες» (flags). Μια άλλη ιδέα, για να υλοποιηθεί το παραπάνω, είναι να στέλνουμε (OutputStream) πρώτα και να περιμένουμε μέχρι να λάβουμε (InputStream).

- Ακόμη, τι θα γινόταν αν κάποιος, για παράδειγμα μια εταιρία, ήθελε να καταγράφει δεδομένα ξεχωριστά για το κάθε όχημα; Τη λύση σε ένα τέτοιο ενδεχόμενο θα έδινε η δυνατότητα να υπάρχει επιλογή στην εφαρμογή, ώστε να δημιουργείται ξεχωριστό αρχείο καταγραφής δεδομένων για το κάθε όχημα. Θα μπορούσε επομένως η εφαρμογή να επεκταθεί και να λειτουργεί σαν καταγραφέας δεδομένων (data logger) και να υπάρχει επιλογή να μπορείς να προσθέτεις κάθε όχημα ξεχωριστά με σκοπό να σώζεις τα δεδομένα του οχήματος σε ξεχωριστά αρχεία, τα οποία θα μπορούσαν να διαβαστούν και να αναλυθούν αργότερα για διάφορους σκοπούς.

- Ένα ακόμα σενάριο είναι να μπορούσε αυτή η εφαρμογή να γίνει λειτουργική για όλες τις εκδόσεις Android που χρησιμοποιούνται, κυρίως στη σημερινή εποχή. Και

αυτό γιατί, όπως αναφέραμε στην παρουσίαση των προβλημάτων, σε παλιότερες εκδόσεις είχαμε περισσότερα σφάλματα να εντοπίσουμε, κυρίως σχετικά με την επικοινωνία μέσω Bluetooth.

5

Παραρτήματα

5.1 Παράρτημα Α: Πίνακας διαθέσιμων μέσω OBD μεγεθών

PID (hex)	Data bytes returned	Description	Min value	Max value	Units	Formula
00	4	PIDs supported [01 - 20]				Bit encoded [A7...D0] == [PID $01...PID $20]
01	4	Monitor status since DTCs cleared. (Includes malfunction indicator lamp (MIL) status and number of DTCs.)				Bit encoded.
02	2	Freeze DTC				
03	2	Fuel system status				Bit encoded.
04	1	Calculated engine load value	0	100	%	A*100/255
05	1	Engine coolant temperature	-40	215	°C	A-40
06	1	Short term fuel %	-100 Subtracting	99.22 Adding	%	(A-128) * 100/128

		trim—Bank 1	Fuel (Rich Condition)	Fuel (Lean Condition)		
07	1	Long term fuel % trim—Bank 1	-100 Subtracting Fuel (Rich Condition)	99.22 Adding Fuel (Lean Condition)	%	(A-128) * 100/128
08	1	Short term fuel % trim—Bank 2	-100 Subtracting Fuel (Rich Condition)	99.22 Adding Fuel (Lean Condition)	%	(A-128) * 100/128
09	1	Long term fuel % trim—Bank 2	-100 Subtracting Fuel (Rich Condition)	99.22 Adding Fuel (Lean Condition)	%	(A-128) * 100/128
0A	1	Fuel pressure	0	765	kPa (gauge)	A*3
0B	1	Intake manifold absolute pressure	0	255	kPa (absolute)	A
0C	2	Engine RPM	0	16,383.75	rpm	((A*256)+B)/4
0D	1	Vehicle speed	0	255	km/h	A

0E	1	Timing advance	-64	63.5	° relative to #1 cylinder	(A-128)/2
0F	1	Intake air temperature	-40	215	°C	A-40
10	2	MAF air flow rate	0	655.35	grams /sec	((A*256)+B) / 100
11	1	Throttle position	0	100	%	A*100/255
12	1	Commanded secondary air status				Bit encoded.
13	1	Oxygen sensors present				[A0...A3] == Bank 1, Sensors 1-4. [A4...A7] == Bank 2
14	2	Bank 1, Sensor 1: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
15	2	Bank 1, Sensor 2: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
16	2	Bank 1, Sensor 3: Oxygen sensor	0	1.275	Volts	A/200 (B-128) * 100/128 (if

		voltage, Short term fuel trim	-100(lean)	99.2(rich)	%	B==$FF, sensor is not used in trim calc)
17	2	Bank 1, Sensor 4: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
18	2	Bank 2, Sensor 1: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
19	2	Bank 2, Sensor 2: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
1A	2	Bank 2, Sensor 3: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
1B	2	Bank 2, Sensor 4: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
1C	1	OBD standards this vehicle conforms to				Bit encoded.

1D	1	Oxygen sensors present				Similar to PID 13, but [A0...A7] == [B1S1, B1S2, B2S1, B2S2, B3S1, B3S2, B4S1, B4S2]
1E	1	Auxiliary input status				A0 == Power Take Off (PTO) status (1 == active) [A1...A7] not used
1F	2	Run time since engine start	0	65,535	seconds	(A*256)+B
20	4	PIDs supported [21 - 40]				Bit encoded [A7..D0] == [PID $21..PID $40]
21	2	Distance travelled with malfunction indicator lamp (MIL) on	0	65,535	km	(A*256)+B
22	2	Fuel rail Pressure (relative to manifold vacuum)	0	5177.265	kPa	((A*256)+B) * 0.079
23	2	Fuel rail Pressure (diesel, or gasoline direct inject)	0	655,350	kPa (gauge)	((A*256)+B) * 10
24	4	O2S1_WR_lambda (1): Equivalence Ratio Voltage	0 0	1.999 7.999	N/A V	((A*256)+B)*2/65535 or ((A*256)+B)/32768 ((C*256)+D)*8/65535 or ((C*256)+D)/8192

25	4	O2S2_WR_lambda (1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
26	4	O2S3_WR_lambda (1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
27	4	O2S4_WR_lambda (1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
28	4	O2S5_WR_lambda (1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
29	4	O2S6_WR_lambda (1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
2A	4	O2S7_WR_lambda (1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
2B	4	O2S8_WR_lambda (1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
2C	1	Commanded EGR	0	100	%	A*100/255

2D	1	EGR Error	-100	99.22	%	(A-128) * 100/128
2E	1	Commanded evaporative purge	0	100	%	A*100/255
2F	1	Fuel Level Input	0	100	%	A*100/255
30	1	# of warm-ups since codes cleared	0	255	N/A	A
31	2	Distance travelled since codes cleared	0	65,535	km	(A*256)+B
32	2	Evap. System Vapour Pressure	-8,192	8191.75	Pa	((A*256)+B)/4 (A and B are two's complement signed)
33	1	Barometric pressure	0	255	kPa (Absolute)	A
34	4	O2S1_WR_lambda (1): Equivalence Ratio Current	0 -128	1.999 127.99	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
35	4	O2S2_WR_lambda (1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
36	4	O2S3_WR_lambda (1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32768 ((C*256)+D)/256 - 128

37	4	O2S4_WR_lambda (1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
38	4	O2S5_WR_lambda (1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
39	4	O2S6_WR_lambda (1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
3A	4	O2S7_WR_lambda (1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
3B	4	O2S8_WR_lambda (1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
3C	2	Catalyst Temperature Bank 1, Sensor 1	-40	6,513.5	°C	((A*256)+B)/10 - 40
3D	2	Catalyst Temperature Bank 2, Sensor 1	-40	6,513.5	°C	((A*256)+B)/10 - 40
3E	2	Catalyst Temperature Bank 1, Sensor 2	-40	6,513.5	°C	((A*256)+B)/10 - 40

3F	2	Catalyst Temperature Bank 2, Sensor 2	-40	6,513.5	°C	((A*256)+B)/10 - 40
40	4	PIDs supported [41 - 60]				Bit encoded [A7…D0] == [PID $41...PID $60]
41	4	Monitor status this drive cycle				Bit encoded.
42	2	Control module voltage	0	65.535	V	((A*256)+B)/1000
43	2	Absolute load value	0	25,700	%	((A*256)+B)*100/255
44	2	Fuel/Air commanded equivalence ratio	0	2	N/A	((A*256)+B)/32768
45	1	Relative throttle position	0	100	%	A*100/255
46	1	Ambient air temperature	-40	215	°C	A-40
47	1	Absolute throttle position B	0	100	%	A*100/255
48	1	Absolute throttle position C	0	100	%	A*100/255
49	1	Accelerator pedal position D	0	100	%	A*100/255

4A	1	Accelerator pedal position E	0	100	%	A*100/255
4B	1	Accelerator pedal position F	0	100	%	A*100/255
4C	1	Commanded throttle actuator	0	100	%	A*100/255
4D	2	Time run with MIL on	0	65,535	minutes	(A*256)+B
4E	2	Time since trouble codes cleared	0	65,535	minutes	(A*256)+B
4F	4	Maximum value for equivalence ratio, oxygen sensor voltage, oxygen sensor current, and intake manifold absolute pressure	0, 0, 0, 0	255, 255, 255, 2550	, V, mA, kPa	A, B, C, D*10
50	4	Maximum value for air flow rate from mass air flow sensor	0	2550	g/s	A*10, B, C, and D are reserved for future use
51	1	Fuel Type				From fuel type table
52	1	Ethanol fuel %	0	100	%	A*100/255
53	2	Absolute Evap system Vapour Pressure	0	327.675	kPa	((A*256)+B)/200

54	2	Evap system vapour pressure	-32,767	32,768	Pa	((A*256)+B)-32767
55	2	Short term secondary oxygen sensor trim bank 1 and bank 3	-100	99.22	%	(A-128)*100/128 (B-128)*100/128
56	2	Long term secondary oxygen sensor trim bank 1 and bank 3	-100	99.22	%	(A-128)*100/128 (B-128)*100/128
57	2	Short term secondary oxygen sensor trim bank 2 and bank 4	-100	99.22	%	(A-128)*100/128 (B-128)*100/128
58	2	Long term secondary oxygen sensor trim bank 2 and bank 4	-100	99.22	%	(A-128)*100/128 (B-128)*100/128
59	2	Fuel rail pressure (absolute)	0	655,350	kPa	((A*256)+B) * 10
5A	1	Relative accelerator pedal position	0	100	%	A*100/255
5B	1	Hybrid battery pack remaining life	0	100	%	A*100/255
5C	1	Engine oil temperature	-40	210	°C	A - 40

5D	2	Fuel injection timing	-210.00	301.992	°	(((A*256)+B)-26,880)/128
5E	2	Engine fuel rate	0	3212.75	L/h	((A*256)+B)*0.05
5F	1	Emission requirements to which vehicle is designed				Bit Encoded
60	4	PIDs supported [61 - 80]				Bit encoded [A7...D0] == [PID $61...PID $80]
61	1	Driver's demand engine - percent torque	-125	125	%	A-125
62	1	Actual engine - percent torque	-125	125	%	A-125
63	2	Engine reference torque	0	65,535	Nm	A*256+B
64	5	Engine percent torque data	-125	125	%	A-125 Idle B-125 Engine point 1 C-125 Engine point 2 D-125 Engine point 3 E-125 Engine point 4
65	2	Auxiliary input / output supported				Bit Encoded

5.2 Παράρτημα Β: Κώδικας Android

5.2.1 Διεπαφή χρήστη (σε xml)

```
<!--
Στο παρακάτω xml αρχείο έχουμε σε κείμενο την απεικόνιση του παραπάνω
σχήματος.
-->

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@mipmap/dashboard">


<!-- Εδώ είναι η λίστα στην οποία προσθέτουμε τις συσκευές που εντοπίζουμε
με Bluetooth -->
    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:visibility="invisible">
    </ListView>

<!-- Εδώ απεικονίζουμε τη τιμή των στροφών του κινητήρα -->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/rpm"
        android:id="@+id/rpmView"
        android:width="100dp"
        android:textColor="#ffffff"
        android:gravity="center"
        android:layout_above="@+id/speedView"
        android:layout_toRightOf="@+id/tempView"
        android:layout_toEndOf="@+id/tempView"
        android:layout_marginLeft="88dp"
        android:layout_marginStart="88dp" />


<!-- Εδώ απεικονίζουμε τη τιμή της ταχύτητας του οχήματος -->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/speed"
        android:id="@+id/speedView"
        android:width="250dp"
        android:textColor="#ff0000"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:textAlignment="center"
        android:gravity="center"
        android:textSize="40sp" />
```

```
<!-- Εδώ απεικονίζουμε τη τιμή της θερμοκρασίας της μηχανής -->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/temp"
        android:id="@+id/tempView"
        android:width="100dp"
        android:textColor="#ffffff"
        android:layout_marginTop="25dp"
        android:gravity="center"
        android:layout_below="@+id/speedView"
        android:layout_alignLeft="@+id/speedView"
        android:layout_alignStart="@+id/speedView" />

<!-- Εδώ απεικονίζουμε τη τιμή της σχετικής θέσης του πεντάλ του γκαζιού
-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/accel"
        android:id="@+id/rel_ac_pp"
        android:width="100dp"
        android:textColor="#ffffff"
        android:gravity="center"
        android:layout_alignTop="@+id/tempView"
        android:layout_alignLeft="@+id/rpmView"
        android:layout_alignStart="@+id/rpmView" />

<!-- Εδώ έχουμε το κουμπί Start -->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button"
        android:id="@+id/button"
        android:onClick="Start"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="37dp"
        android:textColor="#fffefe" />

<!-- Εδώ απεικονίζουμε τη τιμή της κατανάλωσης καυσίμων -->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/frate"
        android:id="@+id/en_frate"
        android:textColor="#ffffff"
        android:layout_alignTop="@+id/rpmView"
        android:layout_alignRight="@+id/tempView"
        android:layout_alignEnd="@+id/tempView" />
</RelativeLayout>
```

5.2.2 Κώδικας Java

```
1. package com.example.ilias.obdproject;
2.
3. import android.annotation.SuppressLint;
4. import android.bluetooth.BluetoothAdapter;
5. import android.bluetooth.BluetoothDevice;
6. import android.bluetooth.BluetoothSocket;
7. import android.content.BroadcastReceiver;
8. import android.content.Context;
9. import android.content.Intent;
10. import android.content.IntentFilter;
11. import android.graphics.Color;
12. import android.os.Environment;
13. import android.os.Handler;
14. import android.os.Message;
15. import android.support.v7.app.AppCompatActivity;
16. import android.os.Bundle;
17. import android.view.View;
18. import android.view.ViewGroup;
19. import android.view.WindowManager;
20. import android.widget.ArrayAdapter;
21. import android.widget.Button;
22. import android.widget.ListView;
23. import android.widget.TextView;
24. import android.widget.Toast;
25.
26. import java.io.BufferedWriter;
27. import java.io.File;
28. import java.io.FileNotFoundException;
29. import java.io.FileWriter;
30. import java.io.IOException;
31. import java.io.InputStream;
32. import java.io.OutputStream;
33. import java.util.Calendar;
34. import java.util.Set;
35. import java.util.Timer;
36. import java.util.TimerTask;
37. import java.util.UUID;
38.
39. public class MainActivity extends AppCompatActivity {
40.
41.     ArrayAdapter<String> listAdapter;
42.     ListView listView;
43.     BluetoothAdapter btAdapter;
44.     Set<BluetoothDevice> devicesArray;
45.     BroadcastReceiver receiver;
46.
47.     public static final UUID MY_UUID =UUID.fromString("00001101-0000"+
48.                                             "-1000-8000-00805F9B34FB");
49.
50.     protected static final int SUCCESS_CONNECT = 0;
51.     protected static final int MESSAGE_READ = 1;
52.     IntentFilter filter;
53.     TextView rpmView, speedView, tempView, en_frateView, rel_acppView;
54.     Button st;
55.
56.     FileWriter pw;
57.     BufferedWriter bf;
58.
```

```
59.     int flag1=0;
60.     int flag2=0;
61.     int flag3=0;
62.     int flag4=0;
63.     int flag5=0;
64.
65.     int check1 = 0;
66.     int check2 = 0;
67.     int check3 = 0;
68.     int times1 = 0;
69.     int times2 = 0;
70.     int times3 = 0;
71.
72.     Handler mHandler = new Handler(){
73.          @SuppressLint("SetTextI18n")
74.          @Override
75.          public void handleMessage(Message msg) {
76.             super.handleMessage(msg);
77.
78.             switch(msg.what){
79.
80.                case SUCCESS_CONNECT:
81.                     ConnectedThread connectedThread = new
82.                             ConnectedThread((BluetoothSocket)msg.obj);
83.                     Toast.makeText(getApplicationContext(), "CONNECT",
84.                                                 Toast.LENGTH_LONG).show();
85.                     String s="";
86.                     connectedThread.write(s.getBytes());
87.                     connectedThread.start();
88.                     break;
89.
90.                case MESSAGE_READ:
91.                     byte[] readBuf = (byte[])msg.obj;
92.                     String string = new String(readBuf);
93.
94.                     saveData(string);
95.
96.                     String[] a;
97.                     int temp1,temp2,rpm, speed, temperature, accelpp;
98.                     double eng_frate;
99.
100.
101.                    // Διαχείριση και επεξεργασία του διαβάσματος
102.                    // των απαντήσεων
103.                    a = string.split("\\s+");
104.
105.                    try {
106.                      for (int i = 0; i < a.length; i++) {
107.
108.                         if (a[i].equals("41") && a[i].length()==2) {
109.                            flag1 += 1;
110.                            flag2 += 1;
111.                            flag3 += 1;
112.                            flag4 += 1;
113.                            flag5 += 1;
114.                            if (a[i + 1].length()==2) {
115.                               switch (a[i + 1]) {
116.                                  case "0C":
117.                                    flag1 = 0;
118.                                    times1 += 1;
119.                                     temp1 = Integer.parseInt(a[2], 16);
```

```
                temp2 = Integer.parseInt(a[3], 16);
                rpm = ((temp1 * 256) + temp2) / 4;
               if (times1 == 1){
                  check1 = rpm;
               }
               if ((a[i + 2].length() == 2) &&
                       (a[i + 3].length() == 2)) {
                 if (Math.abs(check1-rpm)<=1500) {
                  rpmView.setText(Integer.toString
                              (rpm) + " rpm");
                   check1 =rpm;
                 }
               }
               break;
             case "0D":
               flag2 = 0;
               times2 += 1;
               temp1 = Integer.parseInt(a[2],16);
               speed = temp1;
               if (times2 == 1){
                  check2 = speed;
               }
               if ((a[i + 2].length() == 2) {
                 if (Math.abs(check2-speed)<=25) {
                speedView.setText(Integer.toString
                              (speed) + " km/h");
                   check2 = speed;
                 }
               }
               break;
            case "05":
               flag3 = 0;
               times3 += 1;
               temp1 = Integer.parseInt(a[2],16);
               temperature = temp1-40;
               if (times3 == 1){
                  check3 = temperature;
               }
             if ((a[i + 2].length() == 2) {
              if(Math.abs(check3-temperature)<= 8){
               tempView.setText(Integer.toString
                            (temperature) + " °C");
               check3 = temperature;
              }
             }
              break;
           case "5A":
               flag4 = 0;
               temp1 = Integer.parseInt(a[2],16);
               accelpp = (temp1*100)/255;
            if ((a[i + 2].length() == 2) {
              rel_acppView.setText(Integer.toString
                              (accelpp) + " %");
            }
            break;
           case "5E":
               flag5 = 0;
               temp1 = Integer.parseInt(a[2], 16);
               temp2 = Integer.parseInt(a[3], 16);
             eng_frate = ((temp1*256)+temp2)*0.05;
             if ((a[i + 2].length() == 2) &&
```

```
                                        (a[i + 3].length() == 2)) {
                                  en_frateView.setText(Double.toString
                                        (eng_frate) + " L/h");
                                }
                                break;
                            }
                        }
                    }
                    if (flag1 > 50) {
                            rpmView.setText("NO DATA rpm");
                    }
                    if (flag2 > 50) {
                            speedView.setText("NO DATA km/h");
                    }
                    if (flag3 > 50) {
                            tempView.setText("NO DATA °C");
                    }
                    if (flag4 > 50) {
                            rel_acppView.setText("NO DATA %ac");
                    }
                    if (flag5 > 50) {
                            en_frateView.setText("NO DATA L/h");
                    }


                }

              } catch (Exception e) {}
               break;
            }
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //Remove notification bar
        this.getWindow().setFlags(WindowManager.LayoutParams.
        FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);

        //keep screen on
        getWindow().addFlags(WindowManager.LayoutParams.
                                        FLAG_KEEP_SCREEN_ON);

        init();

        if (btAdapter == null) {
           Toast.makeText(getApplicationContext(),"No Bluetooth " +
                        "detected", Toast.LENGTH_LONG).show();
        }else {
           if (!btAdapter.isEnabled()) {
                    turnOnBT();
                    finish();
           }
           getPairedDevices();
           startDiscovery();
        }
    }

```

```
242.    // Δημιουργία αρχείου txt στην SD card του κινητού
243.    // με σκοπό να ελέγχουμε τι στέλνει η συσκευή elm327
244.    private void createFile(){
245.     String root = Environment.getExternalStorageDirectory().
246.                           toString();
247.     File myDir = new File(root + "/my_apps");
248.     myDir.mkdirs();
249.     File file = new File (myDir, "my_data.txt");
250.     try {
251.         pw = new FileWriter(file, true);
252.         bf = new BufferedWriter(pw);
253.     } catch (FileNotFoundException e) {
254.         e.printStackTrace();
255.     } catch (IOException e) {
256.         e.printStackTrace();
257.     }
258.
259.    }
260.
261.    // Σώζουμε τα δεδομένα στο αρχείο που έχουμε δημιουργήσει
262.    private void saveData(String data){
263.       try {
264.         Calendar c = Calendar.getInstance();
265.         int hour = c.get(Calendar.HOUR);
266.         int minutes = c.get(Calendar.MINUTE);
267.         int date = c.get(Calendar.DATE);
268.
269.         String yolo = String.format("%d - %d - %d : %s\n", date,
270.                   hour, minutes, data);
271.         bf.write(yolo);
272.       } catch (FileNotFoundException e) {
273.         e.printStackTrace();
274.       } catch (IOException e) {
275.         e.printStackTrace();
276.       }
277.    }
278.
279.    // Αυτή η μέθοδος ψάχνει για Bluetooth συσκευές
280.    private void startDiscovery() {
281.         btAdapter.cancelDiscovery();
282.         btAdapter.startDiscovery();
283.    }
284.
285.    // Αυτή η μέθοδος ανοίγει το Bluetooth της συσκευής μας
286.    // αν δεν είναι ήδη ανοιχτό
287.    private void turnOnBT() {
288.         Intent intent =new
289.                   Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
290.         startActivityForResult(intent, 1);
291.    }
292.
293.    // Αυτή η μέθοδος βρίσκει ποιες συσκευές είναι ήδη paired
294.    // και τις σώζει σε μία λίστα
295.    private void getPairedDevices() {
296.         devicesArray = btAdapter.getBondedDevices();
297.         if(devicesArray.size()>0){
298.             for(BluetoothDevice device:devicesArray){
299.                   listAdapter.add(device.getName() + "\n" +
300.                                             device.getAddress());
301.
302.             }
```

```
303.          }
304.      }
305.
306.      // Αυτή η μέθοδος αρχικοποιεί τις μεταβλητές μας και κάνει
307.      // κάποιες επιπλέον διαδικασίες που αναλύονται παρακάτω
308.      private void init() {
309.          listView=(ListView)findViewById(R.id.listView);
310.          rpmView=(TextView)findViewById(R.id.rpmView);
311.          speedView=(TextView)findViewById(R.id.speedView);
312.          tempView=(TextView)findViewById(R.id.tempView);
313.          rel_acppView=(TextView)findViewById(R.id.rel_ac_pp);
314.          en_frateView=(TextView)findViewById(R.id.en_frate);
315.          st=(Button)findViewById(R.id.button);
316.
317.
318.          // Εδώ στη λίστα που προσθέτουμε τις συσκευές που
319.          // εντοπίζονται με Bluetooth κάνουμε τα στοιχεία της να
320.          // είναι με χρώμα άσπρο
321.          // Όλη αυτή η διαδικασία μπορεί να παραλειφθεί
322.          // καθώς δεν παίζει ρόλο κάπου στην εφαρμογή μας
323.          // παρά μόνο επειδή θέλαμε να ελέγχουμε ποιες άλλες συσκευές
324.          // με Bluetooth βρίσκονται στον ίδιο χώρο
325.          listAdapter= new ArrayAdapter<String>(this,
326.           android.R.layout.simple_list_item_1,0){
327.            @Override
328.            public View getView(int position, View convertView,
329.                                ViewGroup parent) {
330.              View view = super.getView(position, convertView, parent);
331.              TextView text = (TextView)view.findViewById(android.R.id.
332.                                                                  text1);
333.             // εδώ φαίνεται αυτό
334.              text.setTextColor(Color.WHITE);
335.              return view;
336.            }
337.           };
338.
339.          listView.setAdapter(listAdapter);
340.          btAdapter = BluetoothAdapter.getDefaultAdapter();
341.          filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
342.
343.
344.
345.          // Δημιουργούμε έναν BroadcastReceiver για ACTION FOUND
346.          //για να ανιχνεύσουμε άλλες συσκευές με Bluetooth
347.          receiver = new BroadcastReceiver() {
348.             public void onReceive(Context context, Intent intent) {
349.                 String action = intent.getAction();
350.                 // When discovery finds a device
351.                 if (BluetoothDevice.ACTION_FOUND.equals(action)) {
352.                     // Get the BluetoothDevice object from the Intent
353.                     BluetoothDevice device = intent.getParcelableExtra
354.                                         (BluetoothDevice.EXTRA_DEVICE);
355.                     // Add the name and address to an array adapter to
356.                     // show in a ListView
357.                     listAdapter.add(device.getName() + "\n" +
358.                                                 device.getAddress());
359.                 }
360.             }
361.          };
362.
363.
```

```
364.          // Register the BroadcastReceiver
365.          IntentFilter filter = new IntentFilter(BluetoothDevice.
366.                                                          ACTION_FOUND);
367.          registerReceiver(receiver, filter);
368.          // Don't forget unregister during onDestroy
369.     }
370.
371.     @Override
372.     protected void onDestroy() {
373.          super.onDestroy();
374.          unregisterReceiver(receiver);
375.     }
376.     // Αυτή η μέθοδος δεν μας αφήνει να προχωρήσουμε τη διαδικασία
377.     // αν δεν ενεργοποιήσουμε πρώτα το Bluetooth της συσκευής μας
378.     @Override
379.     protected void onActivityResult(int requestCode, int resultCode,
380.                                     Intent data) {
381.          super.onActivityResult(requestCode, resultCode, data);
382.          if(resultCode == RESULT_CANCELED){
383.             Toast.makeText(getApplicationContext(),"Bluetooth must " +
384.                     "be enabled to continue", Toast.LENGTH_SHORT).
385.                                                              show();
386.             finish();
387.          }
388.     }
389.
390.     // Αυτή η μέθοδος ξεκινάει τη σύνδεση με τον OBD αντάπτορα
391.     // και ταυτόχρονα δημιουργεί το αρχείο που αναφέραμε παραπάνω
392.     // αν δεν υπάρχει ήδη στη μνήμη του κινητού μας
393.     public void Start(View v){
394.          if (btAdapter.isDiscovering()) {
395.             btAdapter.cancelDiscovery();
396.          }
397.          if(devicesArray.size()>0){
398.             for(BluetoothDevice device:devicesArray){
399.                     if(device.getName().equals("JON-SNOW") ||
400.                             device.getName().equals("TASOS-PC") ||
401.                             device.getName().equals("ILIAS-PC") ||
402.                             device.getName().equals("OBD2ECU")) ||
403.                             device.getName().equals("obd2ecu")) ||
404.                             device.getName().equals("Vgate")) {
405.
406.                         createFile();
407.                         ConnectThread connect = new
408.                             ConnectThread(device);
409.                         connect.start();
410.                     }
411.             }
412.          }else{
413.             Toast.makeText(getApplicationContext(), "There is no
414.                     Bluetooth device", Toast.LENGTH_SHORT).show();
415.          }
416.     }
417.
418.     // Αυτό είναι το thread που εγκαθιστά το κανάλι Bluetooth
419.     // Την καλούμε όταν πατήσουμε το κουμπί Start
420.     private class ConnectThread extends Thread {
421.
422.          private final BluetoothSocket mmSocket;
423.          private final BluetoothDevice mmDevice;
424.
```

```
425.         public ConnectThread(BluetoothDevice device) {
426.           BluetoothSocket tmp = null;
427.           mmDevice = device;
428.           try {
429.              tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
430.           } catch (IOException e) {}
431.           mmSocket = tmp;
432.         }
433.
434.         public void run() {
435.           btAdapter.cancelDiscovery();
436.            try {
437.              mmSocket.connect();
438.            } catch (IOException connectException) {
439.              try {
440.                    mmSocket.close();
441.              } catch (IOException closeException) {}
442.              return;
443.            }
444.
445.         mHandler.obtainMessage(SUCCESS_CONNECT,
446.                                              mmSocket).sendToTarget();
447.         }
448.     }
449.
450.     // Αυτό είναι το thread που διαχειρίζεται την ανταλλαγή δεδομένων
451.     // Στέλνουμε εδώ τις αιτήσει για τις τιμές που θέλουμε
452.     // Και παίρνουμε τις απαντήσεις σε αυτές τις αιτήσεις τις
453.     // οποίες διαχειριζόμαστε στον Handler
454.     private class ConnectedThread extends Thread {
455.         private final BluetoothSocket mmSocket;
456.         private final InputStream mmInStream;
457.         private final OutputStream mmOutStream;
458.
459.         public ConnectedThread(BluetoothSocket socket) {
460.            mmSocket = socket;
461.            InputStream tmpIn = null;
462.            OutputStream tmpOut = null;
463.
464.            try {
465.                    tmpIn = socket.getInputStream();
466.                    tmpOut = socket.getOutputStream();
467.            } catch (IOException e) {}
468.
469.            mmInStream = tmpIn;
470.            mmOutStream = tmpOut;
471.         }
472.
473.         public void run() {
474.            byte[] buffer = new byte[1024];
475.            int bytes;
476.
477.            while (true) {
478.                    try {
479.
480.                       bytes = mmInStream.read(buffer);
481.
482.                       // Εδώ ότι διαβάζουμε το στέλνουμε στον Handler
483.                       mHandler.obtainMessage(MESSAGE_READ, bytes, -1,
484.                                                  buffer).sendToTarget();
485.
```

```
                    try {
                        Thread.sleep (50);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                } catch (IOException e) {}
            }
        }

        public void write(byte[] bytes) {
            try {
                mmOutStream.write(bytes);

                // κάποιες AT εντολές
                String s2="AT Z\r";
                mmOutStream.write(s2.getBytes());
                Thread.sleep(200);

                String s3="AT E0\r";
                mmOutStream.write(s3.getBytes());
                Thread.sleep(200);

                String s="AT SP 0\r";
                mmOutStream.write(s.getBytes());
                Thread.sleep(200);


                Timer timer5 = new Timer();
                timer5.schedule(new TimerTask() {
                    @Override
                    public void run() {
                    //τα request
                        String rp = "010C\r";
                        String spd = "010D\r";
                        String tp = "0105\r";
                        String acp = "015A\r";
                        String eng = "015E\r";

                        try {

                            mmOutStream.write(rp.getBytes());
                            try {
                                    Thread.sleep (50);
                            } catch (InterruptedException e) {
                                    e.printStackTrace();
                            }
                            mmOutStream.write(spd.getBytes());
                            try {
                                    Thread.sleep (50);
                            } catch (InterruptedException e) {
                                    e.printStackTrace();
                            }
                            mmOutStream.write(tp.getBytes());
                            try {
                                    Thread.sleep (50);
                            } catch (InterruptedException e) {
                                    e.printStackTrace();
                            }
                            mmOutStream.write(acp.getBytes());
                            try {
                                    Thread.sleep (50);
```

```
547.                            } catch (InterruptedException e) {
548.                                    e.printStackTrace();
549.                            }
550.                            mmOutStream.write(eng.getBytes());
551.                            try {
552.                                    Thread.sleep (50);
553.                            } catch (InterruptedException e) {
554.                                    e.printStackTrace();
555.                            }
556.
557.                        } catch (IOException e) {
558.                            e.printStackTrace();
559.                        }
560.                    }
561.                },0,500);
562.            }catch (IOException e) {}
563.        }
564.    }
565. }
```

5.2.3 Manifest xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.ilias.obdproject" >

<!-- Εδώ δίνουμε τα permissions για χρήση του Bluetooth και για να
αποθηκεύσουμε στην SD card του κινητού για να μπορούμε να σώσουμε το
αρχείο που θέλουμε να κρατάει τα δεδομένα που μας στέλνει η συσκευή elm327
-->
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name=
               "android.permission.WRITE_EXTERNAL_STORAGE"/>


    <application
        android:allowBackup="true"


        <!-- Εδώ ορίζουμε το εικονίδιο που θέλουμε να έχει η εφαρμογή μας
         -->
        android:icon="@mipmap/car_new"
        android:label="@string/app_name"
        android:supportsRtl="true"

        <!-- Εδώ ορίζουμε να μην υπάρχει ActionBar στο Theme μας -->
        android:theme="@style/AppTheme.NoActionBar" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar"
        <!-- Εδώ ορίζουμε να κάνει περιστροφή η απεικόνιση και να φαίνεται
           πλάγια η διεπαφή χρήστη  -->
            android:screenOrientation="landscape" >
```

```
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

5.2.4 Styles xml

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
    <style name="AppTheme.NoActionBar">
        <item name="windowActionBar">false</item>
        <item name="windowNoTitle">true</item>
    </style>
    <style name="AppTheme.AppBarOverlay"
parent="ThemeOverlay.AppCompat.Dark.ActionBar" />
    <style name="AppTheme.PopupOverlay"
parent="ThemeOverlay.AppCompat.Light" />

</resources>
```

5.2.5 Strings xml

```
<resources>

  <string name="app_name">OBDProject</string>
  <string name="rpm">RPM</string>
  <string name="speed">Speed</string>
  <string name="temp">TEMP</string>
  <string name="accel">RL_ACPP</string>
  <string name="button">Start</string>
  <string name="frate">EN_FRATE</string>

</resources>
```

6

Βιβλιογραφία / Παραπομπές

[1] Martin Campbell-Kelly, Daniels D. Garcia-Swartz, "From Mainframes to Smartphones: A History of the International Computer Industry", June 2015

[2] Bob Henderson, John Haynes, "OBD-II & Electronic Engine Management Systems", 1st Edition, November 2006

[3] Steven Hoober, "The Social History of the Smartphone", July 2014

[4] AutoTap – OBDII Automotive Diagnostic Tool, "OBDII: Past, Present and Future", *B&B Electronics Manufacturing Company, Inc.*, 2012

[5] Elm Electronics Inc., "ELM327", www.elmelectronics.com 2005-2014

[6] Peter David, "OBD II Diagnostics Secrets Revealed", August 2012

[7] Jan Axelson, "Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems", December 2007

[8] Al Santini, "Functions, Monitors and Diagnostic Techniques", 1st Edition, June 2010

[9] Mandy Conception, "OBD-2 Automotive Repair Strategies", Kindle Edition, June 2011

[10] Brian Hardy, "Android Programming: The Big Nerd Ranch Guide", April 2013

[11] Marco Gargenta, "Learning Android", March 2011

[12] Rogers Cadenhead, Laura Lemay, "Java 6", June 2007

[13] A. X. A. Sim, B. Sitohang, "OBD-II standard car engine diagnostic software development", *IEEE*, November 2014

[14] Ed Burnette, "Hello Android: Introducing Google's Mobile Development Platform", February 2014

[15] Jeff Friesen, "Learn Java for Android Development", August 2010

[16] Juhani Lehtimaki, "Smashing Android Ui", October 2012

[17] Muhamad Usama Bin Aftab, "Learning Android Intents", January 2014

[18] S. H. Baek, D. W. Jeong, Y. S. Park, H. S. Kim, J. W. Jang, “Implementation Vehicle Driving State System with OBD-II”, *IEEE,* October 2011

[19] Mark Nelson, “Serial Communication Developer’s Guide”, 2nd Edition, December 1999

[20] Neil Smyth, “Android Studio Development Essentials: Android 5 Edition”, July 2014

[21] Bruce Taplin, “Smartphone History: Evolution & Revolution”, Kindle Edition, January 2013

[22] Evan Koblentz, “Abacus to smartphone: The evolution of mobile and portable computers”, July 2015

[23] Sarah Mitroff, “The Android era: From G1 to Lollipop”, May 2014

[24] Lawrence Harte, “Introduction to Bluetooth”, 2nd Edition, November 2009