Πανεπιστήμιο Πελοποννήσου

Σχολή Μηχανικών

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Πτυχιακή Εργασία

Υλοποίηση Λογισμικού για ανίχνευση βλαβών αυτοκινήτου μέσω θύρας OBD

Ιωάννης – Χρυσοβαλάντης Παθιακάκης

Θωμάς Ανδρουλάκης

Επιβλέπων καθηγητής: Δημήτριος Σωτηρόπουλος

Πάτρα, Φεβρουάριος 2023

Περίληψη

Σκοπός της εργασίας είναι η συλλογή δεδομένων από διάφορους αισθητήρες και controllers του αυτοκινήτου σε πραγματικό χρόνο και επαναλαμβανόμενα καθώς και η συλλογή των σφαλμάτων του αυτοκινήτου, η εύρεση τρόπων επίλυσης αυτών, και η διαγραφή τους από τη μνήμη του εγκεφάλου.

Ο κύριος στόχος μας είναι η δημιουργία μιας εφαρμογής η οποία θα είναι κατανοητή από όλους τους χρήστες μέσω μιας απλής διεπαφής χρήστη, έτσι ώστε η διάγνωση και η επισκευή του αυτοκινήτου να είναι πιο εύκολη και προσιτή στον καθημερινό χρήστη.

Για την επίτευξη αυτού του στόχου είναι απαραίτητη , πέρα από το λογισμικό η ύπαρξη ενός Bluetooth OBD Adapter. Τέτοιες συσκευές κυκλοφορούν σε μεγάλη κλίμακα εδώ και αρκετά χρόνια στο εμπόριο, και πλέον η απόκτηση τους είναι πολύ προσιτή λόγω τιμής και διαθεσιμότητας.

Η εφαρμογή λειτουργεί σε όλα τα αυτοκίνητα παραγωγής από το 2004 και μετά με διαγνωστική θήρα OBD ανεξαρτήτως χώρας προέλευσης και μοντέλου.

Μελλοντικός στόχος στην ανάπτυξη της εφαρμογής είναι η επίτευξη της σύνδεσης σε Αντάπτορα μέσω Wi-Fi καθώς και η προσθήκη περισσότερων αισθητήρων, λειτουργιών και μετρήσεων.

Λέξεις – Κλειδιά

OBD, EOBD, OBD2, Διάγνωση αυτοκινήτου, ELM327

Android implementation for car defect detection via OBD port

Thomas Androulakis

Ioannis - Chrisovalantis Pathiakakis

Abstract

The purpose of the project is the collection of data from various sensors and controllers of the car (repeating in real time) as well as the collection of the car's errors, finding ways to solve them, as well as deleting them from the ecu's memory.
Our main goal is to create an application that will be understood by all users through a simple user interface, so that the diagnosis and repair of the car is easier and accessible to the everyday user.
To achieve this goal, in addition to the application, the existence of a Bluetooth OBD Adapter is necessary. Such devices have been commercially available on a large scale for several years, and are now very affordable due to price and availability at many auto accessories and electronics stores.
The application works on all production cars from 2004 onwards with OBD diagnostic hunting regardless of country of origin.
A future goal in the development of the application is to achieve the connection to the Adapter via Wi-Fi as well as to add more sensors, functions and measurements.

Keywords

OBD, EOBD, OBD2, Car diagnostics, ELM327

Περιεχόμενα

Κατάλογος Εικόνων / Σχημάτων

Συντομογραφίες & Ακρωνύμια

OBD Adapter	Απευθύνεται σε μια συσκευή η οποία συνδέεται στην διαγνωστική θύρα του αυτοκινήτου και αναλαμβάνει να γεφυρώσει την επικοινωνία του εγκεφάλου με το διαγνωστικό πρόγραμμα η μηχάνημα.
Freeze Frame	Freeze Frame ή αλλιώς "παγωμένο πλαίσιο" είναι ένα στιγμιότυπο από δεδομένα την στιγμή που κάποιο εξάρτημα παρουσίασε βλάβη.
CEL	Check Engine light η αλλιώς ένδειξη σφάλματος κινητήρα είναι μια ένδειξη στον πίνακα ελέγχου ή καντράν του αυτοκινήτου η οποία μας ενημερώνει ότι έχει ανιχνευτεί κάποιο σφάλμα.
Sensors	Sensors η αλλιώς αισθητήρες είναι συσκευές οι οποίες τοποθετούνται σε διάφορα σημεία του αυτοκινήτου με σκοπό να συγκεντρώσουν πληροφορίες και να τις επιστρέψουν στην κεντρική μονάδα επεξεργασίας η αλλιώς "εγκέφαλο".
VIN	VIN η αλλιώς Vehicle Identification Number είναι ο αριθμός αναγνώρισης του αυτοκινήτου. Συνήθως αναγράφεται στο παρμπρίζ, μέσα από στην πόρτα του οδηγού η στον χώρο του κινητήρα.
Bluetooth Handler	Bluetooth Handler ονομάζουμε ένα μέρος του κώδικα ή συνάρτηση η οποία είναι υπεύθυνη για την διαχείριση της επικοινωνίας μέσω bluetooth με κάποια εξωτερική συσκευή.
Toast	Στις Android συσκευές Toast ονομάζεται ένα μικρό κείμενο σε μορφή ''Popup'' το οποίο συνήθως το χρησιμοποιούμε για να ενημερώσουμε τον χρήστη σχετικά με την λειτουργία της εφαρμογής π.χ. "App is Loading''. Μετά από μερικά δευτερόλεπτα το Toast εξαφανίζεται αυτόματα.

1. Διάγνωση Αυτοκινήτου

Η μαζική παραγωγή των αυτοκινήτων ξεκίνησε στις αρχές του 1900, και άλλαξε την βιομηχανία και τον τρόπο ζωής των ανθρώπων μέχρι και σήμερα. Ήταν και είναι γνωστό πως κάθε εφαρμογή η οποία απαιτεί μηχανικά η ηλεκτρονικά μέρη χρειάζεται συντήρηση και επισκευές.

Σε αυτό το κεφάλαιο θα προβούμε σε μια εισαγωγή για το τι είναι καθώς και το πως λειτουργεί η διάγνωση του αυτοκινήτου, ενώ παρακάτω θα επικεντρωθούμε στην διάγνωση μέσω διαγνωστικής θύρας OBD μέσω κατάλληλου λογισμικού.

1.1 Ανάγκη και σκοπός διάγνωσης

Μια κοινή γνώμη είναι πως πίσω από μια καλή επισκευή κρύβεται μια ακριβής και γρήγορη διάγνωση. Τα αυτοκίνητα πριν το 2000, δεν είχαν τον εξοπλισμό και τα ηλεκτρονικά συστήματα που διαθέτουν τα σημερινά συνεπώς η διάγνωση και επισκευή τους ήταν δυνατόν να ολοκληρωθεί με κοινά εργαλεία χειρός .

Με την εισαγωγή των σύγχρονων μέτρων ασφαλείας (αερόσακοι, ζώνες ασφαλείας), άνεσης (air condition, θερμαινόμενα καθίσματα) και υποβοήθησης (abs, esp, lane assist) τα αυτοκίνητα είναι εξοπλισμένα με πληθώρα αισθητήρων και μικροελεγκτών οι οποίοι ευθύνονται για την εύρυθμη λειτουργία των παραπάνω συστημάτων.
Στην περίπτωση που κάποιο από τα συστήματα του αυτοκινήτου αστοχήσει η βγει εκτός ορίων, το αυτοκίνητο παρουσιάζει βλάβη, μπορεί να υπολειτουργεί και πολλές φορές να μην ανταποκρίνεται.
Για την καταπολέμηση αυτού του φαινομένου συχνά χρειάζεται επισκευή η αντικατάσταση σε κάποιο εξάρτημα, ηλεκτρονικό και μη. Λόγω της εξελιγμένης φύσης των αυτοκινήτων σήμερα η διάγνωση απαιτεί ηλεκτρονικά μέσα δηλαδή διαγνωστικά μηχανήματα. Τα μηχανήματα έχουν σκοπό να σαρώσουν την μνήμη του εγκεφάλου του αυτοκινήτου για σφάλματα και έπειτα τα επιστρέφουν στον χρήστη τα αποτελέσματα της σάρωσης.

1.2 Πως γίνεται η διάγνωση αυτοκινήτου;

Τις περισσότερες φορές όταν κάποιος μικροελεγκτής η αισθητήρας ανιχνεύσει κάποιο σφάλμα, τότε ενεργοποιείται μια προειδοποιητική λυχνία στο καντράν του αυτοκινήτου με σκοπό τη γνωστοποίηση του σφάλματος στον οδηγό.

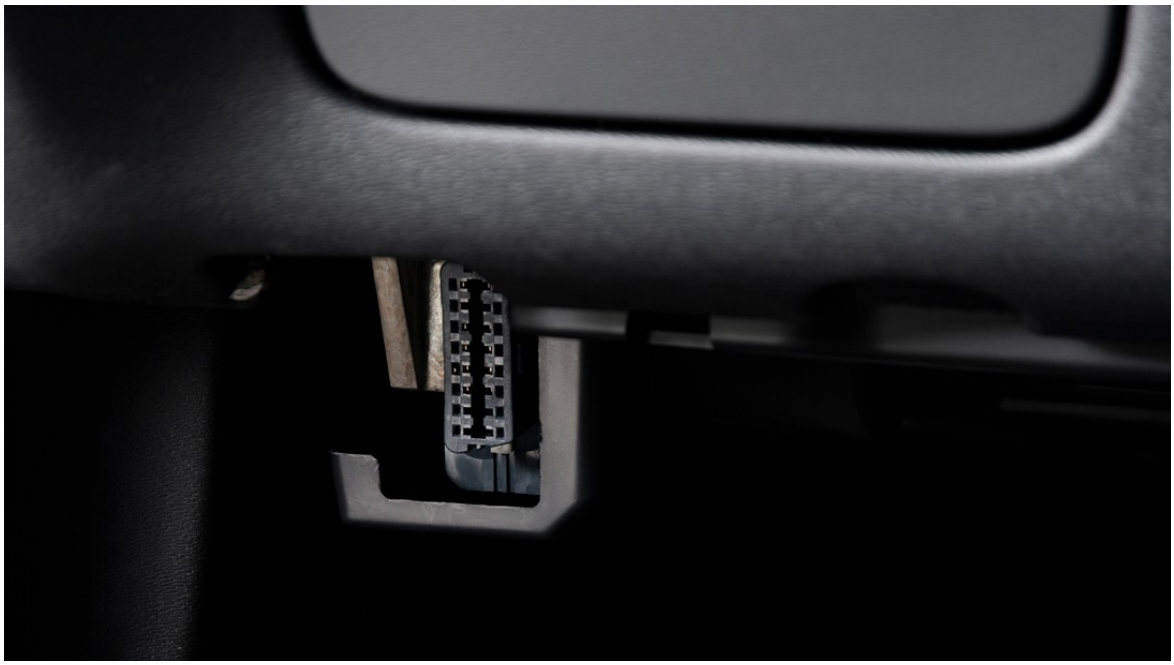
Μέσω της λυχνίας ο οδηγός γνωρίζει ότι υπάρχει κάποιο σφάλμα χωρίς να είναι ξεκάθαρο περί τίνος πρόκειται. Από αυτό το σημείο ξεκινάει ο ρόλος των διαγνωστικών μηχανημάτων όπως θα δούμε παρακάτω

Εικόνα 1: Λυχνία σφάλματος κινητήρα

Πηγή: shorturl.at/knz09

Τα αυτοκίνητα (από το 2004 και μετά) διαθέτουν διαγνωστική θύρα (OBD – On Board Diagnostics) την οποία χρησιμοποιούν όλα τα διαγνωστικά για την ανάγνωση σφαλμάτων, μνήμης και αισθητήρων. Η θύρα OBD βρίσκεται συνήθως κάτω από το ταμπλό στην μεριά του οδηγού.

Εικόνα 2: OBD - II θύρα αυτοκινήτου



Πηγή: shorturl.at/iwxX4

Η διάγνωση απαιτεί υλικό καθώς και λογισμικό κομμάτι για να μπορέσει να ολοκληρωθεί. Στο υλικό κομμάτι έχουμε μια συσκευή επικοινωνίας, της οποίας ο στόχος είναι να γεφυρώσει τον εγκέφαλο του αυτοκινήτου με το απαιτούμενο λογισμικό.

Το λογισμικό είναι ένα πρόγραμμα με απλή διεπαφή χρήστη (UI) το οποίο είναι υπεύθυνο να λαμβάνει τα μηνύματα από τον εγκέφαλο μέσω της συσκευής επικοινωνίας και να τα μετατρέπει σε κείμενο κατανοητό από τον χρήστη.

Τα συνεργεία και οι επαγγελματίες συνήθως χρησιμοποιούν συσκευές οι οποίες έχουν σχεδιασμένο δικό τους λογισμικό σύστημα από την εκάστοτε εταιρία π.χ Bosch με σκοπό την βέλτιστη ταχύτητα και αξιοπιστία των δεδομένων.

Για τις ανάγκες του καθημερινού χρήστη μια τέτοια εφαρμογή διαγνωστικού δεν αποτελεί επιλογή καθώς αυτά τα εργαλεία έχουν μεγάλο κόστος και οι προηγμένες λειτουργίες τους σπάνια θα χρησιμοποιηθούν από έναν ερασιτέχνη χρήστη.

Για την επίλυση αυτού του χάσματος έχει αναπτυχθεί μια Universal λύση η οποία δίνει τη δυνατότητα στο χρήστη με πολύ μικρό κόστος, να διαγνώσει και να λάβει παραμέτρους από το αυτοκίνητο του.

Στο κομμάτι του υλικού χρειάζεται μια συσκευή η οποία θα εισάγεται στη θύρα OBD η οποία έχει ο σκοπό την μετάδοση μηνυμάτων από το αυτοκίνητο στον χρήστη και αντίστροφα.

Για το λογισμικό μας αρκεί μια εφαρμογή, σχεδιασμένη και ανεπτυγμένη στο να χρησιμοποιεί τα κατάλληλα πρωτόκολλα και κανόνες έτσι ώστε να μπορεί να επικοινωνήσει με την πληθώρα των αυτοκινήτων.

Στην αγορά υπάρχουν διάφορες επιλογές σε OBD Adapter, οι οποίες συνήθως χρησιμοποιούνται με κάποιο τρίτο λογισμικό γραμμένο είτε για υπολογιστή είτε για κινητή συσκευή. Παρακάτω φαίνεται ένα OBD Adapter:

Εικόνα 3: ELM 327 ανταπτοράκι

Πηγή: shorturl.at/kpuC8

2. Πρωτόκολλα επικοινωνίας

Για την επίτευξη της σύνδεσης της συσκευής επικοινωνίας με το αυτοκίνητο πρέπει να υπάρχει κάποιο πρωτόκολλο επικοινωνίας. Τα αυτοκίνητα αναλόγως μοντέλου, επικοινωνούν με διάφορα πρωτόκολλα επικοινωνίας που υποστηρίζονται από την εκάστοτε εφαρμογή λογισμικού.

2.1 Τα πρωτόκολλα επικοινωνίας

Ένας αντάπτορας ELM327 4 είναι μια συσκευή της οποίας η λειτουργία είναι να διαβάζει και να μεταδίδει δεδομένα από τον εγκέφαλο του αυτοκινήτου μέσω της διαγνωστικής θύρας (OBD).

Ανάλογα με την χρονολογία και το μοντέλο του αυτοκινήτου, το λογισμικό επιλέγει συνήθως αυτόματα χωρίς την επέμβαση του χρήστη ένα από τα παρακάτω πρωτόκολλα επικοινωνίας τα οποία έχουν αναπτυχθεί με τα χρόνια.
Παρακάτω αναγράφονται ενδεικτικά κάποια πρωτόκολλα επικοινωνίας που χρησιμοποιούνται στα αυτοκίνητα τα οποία διαφέρουν ανάλογα τον κατασκευαστή και την χρονολογία παραγωγής.

- SAE J1850 PWM (41.6 kbit/s)

SAE J1850 PWM

Feature	Description
BUS +	Pin 2
BUS -	Pin 10
12V	Pin 16
GND	Pins 4, 5
Bus State:	Active when BUS + is pulled HIGH, BUS - is pulled LOW
Maximum Signal Voltage:	5V
Minimum Signal Voltage:	0V
Number of bytes:	12
Bit Timing:	'1' bit - 8uS, '0' bit - 16uS, Start of Frame - 48uS

Πηγή: shorturl.at/hGHJP

Εικόνα 4: SAE J1850 PWM (41.6 kbit/s)

Το πρωτόκολλο SAE J1850 έχει ταχύτητα 41.6Kbps και συνήθως χρησιμοποιείται σε ford οχήματα.

- SAE J1850 VPW (10.4 kbit/s)

SAE J1850 VPW	
Feature	**Description**
BUS +	Pin 2
12V	Pin 16
GND	Pins 4, 5
Bus State:	Bus idles low
Maximum Signal Voltage:	+7V
Decision Signal Voltage:	+3.5V
Minimum Signal Voltage:	0V
Number of bytes:	12
Bit Timing:	'1' bit -HIGH 64uS, '0' bit -HIGH 128uS, Start of Frame - HIGH 200uS

Πηγή: shorturl.at/hGHJP

Εικόνα 5: SAE J1850 VPW (10.4 kbit/s)

Το πρωτόκολλο SAE J1850 έχει ταχύτητα 10.4kbs και χρησιμοποιείται συνήθως σε οχήματα του ομίλου GM (Chevrolet, gmc, cadillac κ.α.)

- ISO 9141-2 (10.4 kbit/s)

ISO 9141-2	
Feature	**Description**
K Line (bidirectional)	Pin 7
L Line (unidirectional, optional)	Pin 15
12V	Pin 16
GND	Pins 4, 5
Bus State:	K Line idles HIGH. Bus is active when driven LOW.
Maximum Signal Voltage:	+12V
Minimum Signal Voltage:	0V
Number of bytes:	Message: 260, Data: 255
Bit Timing:	UART: 10400bps, 8-N-1

Πηγή: shorturl.at/hGHJP

Εικόνα 6: ISO 9141-2 (10.4 kbit/s)

Το πρωτόκολλο επικοινωνίας ISO 9141-2 χρησιμοποιείται σε Chrysler ευρωπαϊκά ή Ασιατικά αυτοκίνητα. Έχει ταχύτητα 10.4kbps.

- ISO 14230-4 KWP (10.4 kbit/s)

ISO 14230 KWP2000	
Feature	**Description**
K Line (bidirectional)	Pin 7
L Line (unidirectional, optional)	Pin 15
12V	Pin 16
GND	Pins 4, 5
Bus State:	Active when driven LOW.
Maximum Signal Voltage:	+12V
Minimum Signal Voltage:	0V
Number of bytes:	Data: 255
Bit Timing:	UART: 10400bps, 8-N-1

Πηγή: shorturl.at/hGHJP

Εικόνα 7: ISO 14230-4 KWP (10.4 kbit/s)

Το KWP2000 πρωτόκολλο ή αλλιώς Keyword Protocol 2000 παρουσιάζει άλλη μια μορφή ασύγχρονης σειριακής επικοινωνίας ή οποία υλοποιείται στα ευρωπαϊκά, ασιατικά και chrysler αυτοκίνητα με ταχύτητα 10.4kbps

- ISO 15765 CAN (500 kbit/s)

ISO 15765 CAN	
Feature	**Description**
CAN HIGH (CAN H)	Pin 6
CAN LOW (CAN L)	Pin 14
12V	Pin 16
GND	Pins 4, 5
Bus State:	Active when CANH pulled HIGH, CANL pulled LOW. Idle when signals are floating.
CANH Signal Voltage:	+3.5V
CANL Signal Voltage:	+1.5V
Maximum Signal Voltage:	CANH = +4.5V, CANL = +2.25V
Minimum Signal Voltage:	CANH = +2.75V, CANL = +0.5V
Number of bytes:	L
Bit Timing:	250kbit/sec or 500kbit/sec

Πηγή: shorturl.at/hGHJP

Εικόνα 8: ISO 15765 CAN (500 kbit/s)

Το πρωτόκολλο ISO15765 CAN έχει επιβληθεί σε όλα τα αυτοκίνητα που πωλούνται στις ΗΠΑ μετά το 2008. Στα ευρωπαϊκά αυτοκίνητα υπάρχουν μερικές υλοποιήσεις CAN από το 2003 και μετά. Η ταχύτητα μπορεί να φτάσει από 250kbps έως και 1Mbps

3. Λειτουργίες OBD

Για να επιτευχθεί η διάγνωση και οι μετρήσεις ενός αυτοκινήτου υπάρχουν μερικές λειτουργίες οι οποίες μπορούν να εκτελεσθούν για να βελτιστοποιήσουν την πιθανότητα μιας πετυχημένης επισκευής.

Οι λειτουργίες αυτές ποικίλουν ανάλογα με τη συμβατότητα και την χρονολογία κάθε αυτοκινήτου. Αναφορικά περιγράφουμε τις εννέα λειτουργίες που αναγράφονται στην παρακάτω λίστα:

1. Προβολή Δεδομένων σε πραγματικό χρόνο

2. Προβολή αποθηκευμένων σφαλμάτων
3. Προβολή δεδομένων παγωμένου πλαισίου
4. Παρακολούθηση σφαλμάτων από τον τελευταίο κύκλο οδήγησης
5. Καθαρισμός αποθηκευμένων σφαλμάτων
6. Μόνιμοι κωδικοί βλάβης διαγνωστικού ελέγχου
7. Αποτελέσματα ελέγχων και παρακολούθηση αισθητήρα O2
8. Έλεγχος λειτουργίας εξαρτήματος
9. Πληροφορίες οχήματος

3.1 Προβολή Δεδομένων σε πραγματικό χρόνο

Ίσως η πιο διαδεδομένη και χρήσιμη για την διάγνωση λειτουργία, είναι η προβολή δεδομένων σε πραγματικό χρόνο. Η λειτουργία αυτή επιτρέπει στον χρήστη να αναγνώσει δεδομένα και μετρήσεις από πληθώρα αισθητήρων του αυτοκινήτου και να ελέγξει αν οι παράμετροι λειτουργίας είναι σωστές.

Η προβολή δεδομένων σε πραγματικό χρόνο μπορεί να χρησιμοποιηθεί για να δούμε τα Volt της μπαταρίας του αυτοκινήτου, την πίεση καυσίμου, την αναλογία αέρα-καυσίμου και άλλες χρήσιμες παραμέτρους για την εύρυθμη λειτουργία του συνόλου ενός αυτοκινήτου.
Συνήθως στις εφαρμογές εμφανίζεται σε μορφή “Gauges” δηλαδή γραφικών ενδείξεων οι οποίες ενημερώνονται σε πραγματικό χρόνο και μας παρέχουν τις αλλαγές στα δεδομένα ψηφιακά και επαναλαμβανόμενα.
Ακόμη, εμφανίζεται και σαν ροή δεδομένων όπου ο χρήστης επιλέγει τις παραμέτρους που θέλει να καταγράψει και αυτές του εμφανίζονται είτε σε κείμενο είτε σε γραφική παράσταση.
Είναι προφανές λοιπόν ότι η προβολή δεδομένων σε πραγματικό χρόνο αποτελεί αναπόσπαστο εργαλείο διάγνωσης τόσο σε συνεργεία όσο και σε ερασιτέχνες χρήστες λόγω της αμεσότητας και της εγκυρότητας στα δεδομένα που προσφέρει.

3.2 Προβολή αποθηκευμένων σφαλμάτων

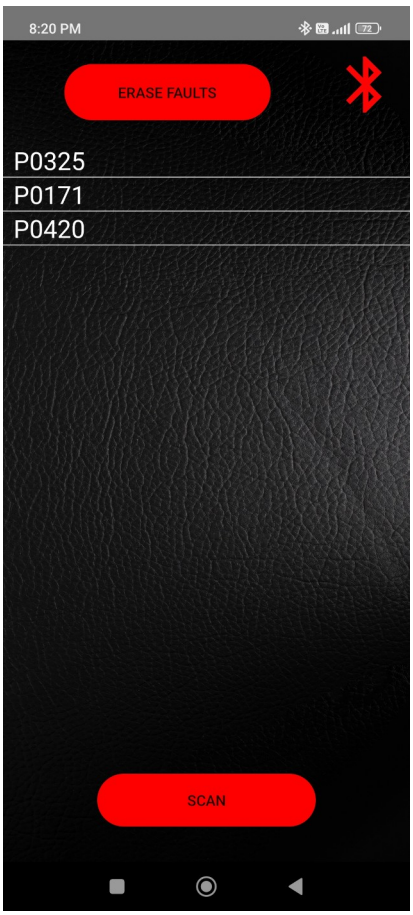
Τα σύγχρονα αυτοκίνητα είναι ρυθμισμένα να λειτουργούν υπό κάποιες συγκεκριμένες συνθήκες. Όταν ένα controller αντιληφθεί πως οι συνθήκες αυτές έχουν ξεφύγει από τα όρια, για παράδειγμα έχουμε μειωμένη πίεση καυσίμου, τότε ο εγκέφαλος ενεργοποιεί το

CEL (Check Engine Light) ή αλλιώς το λαμπάκι βλάβης και αποθηκεύει το σφάλμα στην μνήμη.

Τα σφάλματα επιστρέφονται σαν "κωδικοί σφαλμάτων". Για παράδειγμα P0087 όπου το P επισημαίνει ότι ο κωδικός 0087 βρίσκεται στο Powertrain.

Ενδεικτικά έχουμε:

- Pxxxx για Powertrain : Αναφέρεται στο Μηχανικό σύνολο
- Bxxxx για Body : Αναφέρεται συνήθως στις λειτουργίες μέσα στην καμπίνα όπως συστήματα ασφαλείας, άνεσης και βοήθειας.
- Cxxxx για Chassis : Αναφέρεται συνήθως στις λειτουργίες εκτός της καμπίνας όπως φρένα, ανάρτηση και διεύθυνση.
- Uxxxx αναφέρεται στις λειτουργίες μεταξύ των ηλεκτρονικών και των υπολογιστών του αυτοκινήτου.



Στόχος λοιπόν του λογισμικού είναι να συνδεθεί στον εγκέφαλο και να κάνει ανάγνωση του κωδικού βλάβης. Έπειτα πρέπει να περιγράψει τον κωδικό βλάβης (πχ P0087 – Fuel Rail Pressure Too Low) και σε προηγμένες εφαρμογές να δώσει στον χρήστη πληροφορίες όπως τρόπους επίλυσης ή συμβουλές σχετικά με την επιδιόρθωση.

3.3 Προβολή δεδομένων παγωμένου πλαισίου (Freeze frame)

Freeze Frame ή αλλιώς "παγωμένο πλαίσιο" είναι ένα στιγμιότυπο από δεδομένα την στιγμή που κάποιο εξάρτημα παρουσίασε βλάβη.

Ένα Freeze Frame μπορεί να μας δώσει δεδομένα όπως η ταχύτητα του οχήματος, η ταχύτητα περιστροφής του κινητήρα, θερμοκρασία και άλλα δεδομένα την στιγμή που παρουσιάστηκε και αποθηκεύτηκε κάποιο σφάλμα.

Χρησιμοποιείται στην διάγνωση όταν είναι κρίσιμο να ξέρουμε κάτω από ποιες συνθήκες το αυτοκίνητο παρουσίασε την βλάβη.

3.4 Παρακολούθηση σφαλμάτων από τον τελευταίο κύκλο οδήγησης

Σε αυτή τη λειτουργία αποθηκεύονται τα σφάλματα από τον τελευταίο ή τους τελευταίους κύκλους οδήγησης. Αναφέρεται συνήθως και ως “Pending Faults”, δηλαδή εκκρεμείς βλάβες. Οι βλάβες αυτές αποθηκεύονται στην μνήμη του αυτοκινήτου αφού ανιχνευτούν στους τελευταίους κύκλους οδήγησης.

Σε περίπτωση που ο κωδικός δεν ξανά εμφανιστεί για τους τελευταίους περίπου 40 κύκλους, τότε διαγράφεται από την μνήμη αυτόματα.
Συνήθως μαζί με την αποθήκευση ενός pending code αποθηκεύεται όπως είδαμε και πιο πάνω το Freeze Frame, με σκοπό να βοηθήσει τον τεχνικό στην διάγνωση της βλάβης.
Οι “Pending“ κωδικοί έχουν την ίδια μορφή με τους τυπικούς κωδικούς σφαλμάτων δηλαδή για παράδειγμα “Pxxxx”.

3.5 Καθαρισμός αποθηκευμένων σφαλμάτων

Όπως παραπέμπει και ο τίτλος του, ο καθαρισμός αποθηκευμένων σφαλμάτων αναλαμβάνει να εκκαθαρίσει όλα τα σφάλματα τα οποία είναι αποθηκευμένα στην μνήμη του εγκεφάλου καθώς να σβήσει και το λαμπάκι της βλάβης από το καντράν του αυτοκινήτου.

3.6 Μόνιμοι κωδικοί βλάβης διαγνωστικού ελέγχου

Σε αυτή τη λειτουργία μπορούν να αναγνωστούν οι μόνιμοι κωδικοί βλάβης.

Οι μόνιμοι κωδικοί βλάβης είναι παρόμοιοι με τους απλούς κωδικούς βλάβης (βλέπε 3.2), με την διαφορά ότι δεν μπορούν να διαγραφούν από την μνήμη με κάποιο διαγνωστικό εργαλείο η ακόμη και αποσυνδέοντας την μπαταρία από το αυτοκίνητο. Ο τρόπος διαγραφής αυτών των σφαλμάτων είναι η επιδιόρθωση του εκάστοτε προβλήματος.

3.7 Αποτελέσματα ελέγχων και παρακολούθηση αισθητήρα O2

Η παραπάνω λειτουργία μας επιτρέπει να εκτελέσουμε μερικούς διαγνωστικούς ελέγχους με σκοπό την διάγνωση σε συγκεκριμένα εξαρτήματα καθώς και την διάγνωση βλάβης

στους αισθητήρες οξυγόνου (O2) οι οποίοι είναι κρίσιμοι για την εύρυθμη και αποδοτική λειτουργία του αυτοκινήτου.

Για παράδειγμα, μπορούμε να εξετάσουμε τα Volt των αισθητήρων O2, τις αστοχίες ανάφλεξης σε κάθε κύλινδρο και άλλα.
Αξίζει να σημειωθεί ότι ανάλογα με τις λειτουργίες και την χρονολογία ενδέχεται να υπάρχει διαφοροποίηση στην λειτουργία αυτή καθώς συνήθως τα πιο σύγχρονα αυτοκίνητα διαθέτουν περισσότερους αισθητήρες και ελεγκτές οι οποίοι μας δίνουν περισσότερες πληροφορίες από τα παλαιότερα.

3.8 Έλεγχος λειτουργίας εξαρτήματος

Μια λειτουργία την οποία συναντάμε συνήθως σε πολύ προηγμένες εφαρμογές υλικού και λογισμικού είναι ο έλεγχος λειτουργίας εξαρτήματος. Ο έλεγχος λειτουργίας συστήματος μας δίνει την δυνατότητα να ελέγξουμε μέσω του λογισμικού την λειτουργία και αποδοτικότητα συγκεκριμένων εξαρτημάτων. Για παράδειγμα το βεντιλατέρ στο ψυγείο νερού, τα μοτέρ των παράθυρων και άλλα.

3.9 Πληροφορίες οχήματος

Η λειτουργία "πληροφορίες οχήματος" μας επιστρέφει πληροφορίες χρήσιμες για την ταυτοποίηση του οχήματος όπως το πλαίσιο "VIN", το μοντέλο του εγκεφάλου και άλλα.

4. PIDS – Βασικοί αισθητήρες και λειτουργία τους

Στο τέταρτο κεφάλαιο αναπτύσσεται η έννοια των PIDS και η σύνδεση τους με τους βασικούς αισθητήρες του αυτοκινήτου. Για να ζητήσουμε δεδομένα από κάποιον αισθητήρα, αρκεί να ξέρουμε το αναγνωριστικό PID ώστε να στείλουμε την εντολή η οποία θα μας επιστρέψει τα αποτελέσματα. Η διαδικασία περιγράφεται παρακάτω καθώς και η επεξήγηση κάποιον βασικών αισθητήρων που βρίσκουμε στα σύγχρονα αυτοκίνητα.

4.1 PIDS (On-Board Diagnostics Parameter IDs)

Τα PIDS είναι αναγνωριστικοί κωδικοί - εντολές οι οποίοι χρησιμοποιούνται για να ζητήσουμε δεδομένα από το αυτοκίνητο.

Κάθε PID αναφέρεται σε έναν συγκεκριμένο τύπο δεδομένου. Για παράδειγμα για να ζητήσουμε την "θερμοκρασία εισαγόμενου αέρα ή Intake Air Temperature" αποστέλλουμε στον εγκέφαλο του κινητήρα τον κωδικό "0F". Μετά την λήψη του κωδικού ο εγκέφαλος μας επιστρέφει την ζητούμενη τιμή, η οποία και μετατρέπεται σε κείμενο για να είναι κατανοητή από τον χρήστη. Απώτερος σκοπός της διαδικασίας αυτής είναι να υπάρξει επανάληψη, δηλαδή να έχουμε αποστολή και λήψη μηνύματος επαναλαμβανόμενα αφού ο χρήστης το ζητήσει. Η συμβατότητα από αυτοκίνητο σε αυτοκίνητο διαφέρει αφού συχνά δεν έχουν όλα τα αυτοκίνητα τη δυνατότητα να απαντήσουν σε όλα τα PID λόγω της έλλειψης ή διαφοροποίησης συγκεκριμένων αισθητήρων. Στον παρακάτω πίνακα φαίνονται τα διαθέσιμα PIDS που υποστηρίζονται από το λογισμικό μας.

Αναγνωριστικά Παραμέτρων	PIDS	Μονάδες Μέτρησης
Coolant Temperature (Θερμοκρασία ψυκτικού υγρού)	05	°C
Engine Load (Φορτίο κινητήρα)	04	%
Fuel Pressure (Πίεση Καυσίμου)	0A	Kpa
Intake Air Temperature (Θερμοκρασία εισαγόμενου αέρα)	0F	°C
Intake Pressure (Πίεση εισαγωγής)	0B	Kpa
Long Term Fuel Trim Bank 1 (Μακροπρόθεσμη περικοπή καυσίμου)	07	%
Long Term Fuel Trim Bank 2 (Μακροπρόθεσμη περικοπή καυσίμου)	09	%
RPM (ΣΑΛ - Στροφές ανά λεπτό)	0C	Rpm
Short Term Fuel Trim Bank 1 (Βραχυπρόθεσμη περικοπή καυσίμου)	06	%
Short Term Fuel Trim Bank 2 (Βραχυπρόθεσμη περικοπή καυσίμου)	08	%
Throttle Position (Θέση πεντάλ γκαζιού)	11	%
Timing Advance (προ πορεία ανάφλεξης)	0E	° before TDC
Vehicle Speed (Ταχύτητα οχήματος)	0D	km/h

Εικόνα 9: Αναγνωριστικά παραμέτρων της εφαρμογής μας

4.2 Βασικοί αισθητήρες και λειτουργία τους

Κατά τη χρήση της εφαρμογής εμφανίζονται τα ονόματα διάφορων αισθητήρων και παραμέτρων οι οποίοι χρησιμοποιούνται ευρέως στις αυτοκινητοβιομηχανίες. Οι αισθητήρες και οι παράμετροι είναι άμεσα προσπελάσιμοι από την εκάστοτε εφαρμογή και τα δεδομένα τους εμφανίζονται στον χρήστη σε πραγματικό χρόνο. Παρακάτω φαίνονται μερικοί από τους αισθητήρες, τις παραμέτρους και τις μετρήσεις που χρησιμοποιούνται συχνότερα κατά την διάγνωση και τον έλεγχο του αυτοκινήτου.

4.2.1 Knock Sensor (Αισθητήρας Κρούσης)

Ο Knock Sensor ή αισθητήρας κρούσης είναι ζωτικής σημασίας για τον κινητήρα του αυτοκινήτου. Παρομοιάζεται συνήθως με ένα μικρό ακουστικό αφού στόχος του είναι να ‘’ακούει” για κτύπους στους κυλίνδρους του κινητήρα οι οποίοι μπορούν να προκληθούν από κακή ποιότητα καυσίμου, λάθος αναλογία αέρα-καυσίμου στους θαλάμους καύσης ή ακόμα και υψηλές θερμοκρασίες.

Ο εγκέφαλος αφού λάβει ως παράμετρο τον knock sensor ρυθμίζει τις διάφορες παραμέτρους του κινητήρα όπως την προπορεία ανάφλεξης και τον ψεκασμό καυσίμου ανάλογα, ώστε να ελαχιστοποιήσει τον κτύπο ο οποίος μπορεί να δημιουργήσει βλάβες η και ολική καταστροφή στον κινητήρα.

Πολλά αυτοκίνητα αν ανιχνεύσουν ‘’κρούση’’ στον κινητήρα είναι προγραμματισμένα να ενεργοποιούν το ‘’Limp’’ mode το οποίο είναι μια λειτουργία κατά την οποία ο κινητήρας περιορίζει αισθητά την δύναμη και το εύρος στροφών που λειτουργεί με σκοπό να προστατευτεί μέχρι να γίνει η διάγνωση και η επισκευή.

Σε αυτή την περίπτωση, συνήθως ανάβει και η λυχνία ένδειξης βλάβης του κινητήρα ή αλλιώς Check Engine Light (CEL).

Το σήμα που εκπέμπει ο Knock Sensor μεταφράζεται σε Volt, όπου όσο περισσότερα volt, τόσο μεγαλύτερος και ο “κτύπος”.

Εικόνα 10: Αισθητήρας Κρούσης

Πηγή: shorturl.at/fnrCE

4.2.2 Air Intake Temperature Sensor (Αισθητήρας Θερμοκρασίας Εισαγόμενου Αέρα)

Για να επιτευχθεί η εύρυθμη και αποδοτική λειτουργία του κινητήρα, πρέπει ανά πάσα στιγμή να επιτυγχάνεται σωστή αναλογία αέρα και καυσίμου στους θαλάμους καύσης.

Είναι γνωστό πως ανάλογα με την θερμοκρασία, μεταβάλλεται και η πυκνότητα του αέρα. Για να πετύχει λοιπόν την σωστή αναλογία πρέπει ο εγκέφαλος, ο οποίος ρυθμίζει το μείγμα να γνωρίζει την θερμοκρασία του εισαγόμενου αέρα, έτσι ώστε να βελτιστοποιήσει την λειτουργία του κινητήρα και να ελαχιστοποιήσει τυχόν κτύπους οι οποίοι μπορεί να προκύψουν από λάθος αναλογία αέρα - καυσίμου.

Αν παρουσιάσει βλάβη ο ‘’Αισθητήρας θερμοκρασίας εισαγόμενου αέρα’’ τότε το αυτοκίνητο πιθανών να παρουσιάσει μειωμένη δύναμη, δονήσεις στο ρελαντί και τις περισσότερες φορές θα ενεργοποιηθεί και η λυχνία βλάβης στο καντράν του οδηγού, η οποία θα χρειαστεί λογισμικό διάγνωσης για να αναγνωστεί.

Εικόνα 11: Αισθητήρας Θερμοκρασίας Εισαγόμενου Αέρα

Πηγή: shorturl.at/lGLW1

4.2.3 Intake Manifold Pressure Sensor (Αισθητήρας Πίεσης Εισαγωγής)

Ο αισθητήρας πίεσης εισερχόμενου αέρα η αλλιώς Intake Manifold Pressure Sensor μετράει ανά πάσα στιγμή την πίεση του εισερχόμενου αέρα. Ο εγκέφαλος χρειάζεται να γνωρίζει την πίεση του εισερχόμενου αέρα έτσι ώστε να ρυθμίσει σωστά την αναλογία καυσίμου με στόχο την αποδοτικότερη λειτουργία.

Αν παρουσιάσει βλάβη, ανάβει συνήθως η λυχνία βλάβης στο καντράν του αυτοκινήτου και πιθανών το αυτοκίνητο να μην ξεκινάει, η να παρουσιάζει κακή λειτουργία και μεγάλη κατανάλωση καυσίμου.

Εικόνα 12: Αισθητήρας Πίεσης Εισαγωγής

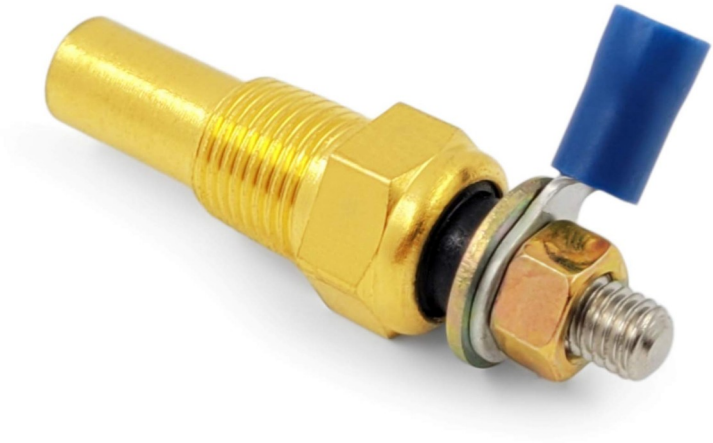
Πηγή: shorturl.at/dmQRU

4.2.4 Coolant – Oil Temperature Sensors (Αισθητήρες Θερμοκρασίας Ψυκτικού – Λαδιού)

Όμοιοι στην λειτουργία τους οι αισθητήρες θερμοκρασίας λαδιού και ψυκτικού υγρού έχουν σαν στόχο την αποτύπωση της θερμοκρασίας του ψυκτικού και του λαδιού στο καντράν του οδηγού.

Ενώ η θερμοκρασία του ψυκτικού υγρού υπάρχει σαν ένδειξη σχεδόν σε όλα τα σύγχρονα αυτοκίνητα, η ένδειξη της θερμοκρασίας λαδιού συναντάται συνήθως στα αυτοκίνητα υψηλών επιδόσεων.

Εικόνα 13: Αισθητήρες Θερμοκρασίας Ψυκτικού – Λαδιού



Πηγή: shorturl.at/aDXZ0

4.2.5 Oil Pressure Sensor (Αισθητήρες Πίεσης Λαδιού)

Ο αισθητήρας πίεσης του λαδιού συναντάται συνήθως σε όλα τα αυτοκίνητα αφού η σωστή λίπανση είναι πολύ σημαντική για την μακροζωία του κινητήρα.

Ο αισθητήρας μετράει συνεχώς την πίεση του λαδιού και σε περίπτωση που ελαττωθεί κάτω από τα εργοστασιακά δεδομένα, ανάβει μια προειδοποιητική λυχνία στο καντράν του αυτοκινήτου.

Μέσω των διαγνωστικών μηχανημάτων είναι δυνατή η διάγνωση της ορθής λειτουργίας του αισθητήρα καθώς και η ανάγνωση της πίεσης του λαδιού ανά πάσα στιγμή.

Εικόνα 14: Αισθητήρες Πίεσης Λαδιού

Πηγή: shorturl.at/hrwyQ

4.2.6 Throttle Position Sensor (Αισθητήρας Θέσης Γκαζιού)

Ο αισθητήρας θέσης πεντάλ γκαζιού είναι συνήθως τοποθετημένος κατευθείαν πάνω στην πεταλούδα του γκαζιού με σκοπό να διαβάζει και να μεταδίδει σε ποσοστό "%" το άνοιγμα της πεταλούδας. Συνεπώς αν έχουμε πατήσει το γκάζι στο μέγιστο βλέπουμε συνήθως σαν επιστρεφόμενη τιμή 100% (μερικές φορές και 99%).

Όπως και οι περισσότεροι αισθητήρες στόχος του είναι να βοηθήσει στην βελτιστοποίηση του μείγματος αέρα-καυσίμου, την αποδοτικότητα καθώς και το ρελαντί του αυτοκινήτου. Μέσω των διαγνωστικών μηχανημάτων και εφαρμογών μπορούμε να διαβάσουμε το ποσοστό του ανοίγματος της πεταλούδας γκαζιού, δεδομένο το οποίο μπορεί να μας χρησιμεύσει στην διάγνωση της ίδιας της πεταλούδας αλλά και άλλων εξαρτημάτων.

Εικόνα 15: Αισθητήρας Θέσης Γκαζιού

Πηγή: shorturl.at/glKS9

4.2.7 Fuel Pressure Sensor (Αισθητήρας Πίεσης Καυσίμου)

Όπως παραπέμπει και η ονομασία του ο αισθητήρας πίεσης καυσίμου, μετράει την πίεση του καυσίμου από το ντεπόζιτο του αυτοκινήτου στον κινητήρα. Η σταθερή και υψηλή πίεση καυσίμου είναι αναγκαία για την εύρυθμη και οικολογική λειτουργία του αυτοκινήτου. Ο αισθητήρας πίεσης καυσίμου η αλλιώς Fuel Pressure Sensor στέλνει συνεχώς δεδομένα σχετικά με την πίεση του καυσίμου στον εγκέφαλο και αν ανιχνευτεί κάποια τιμή εκτός των εργοστασιακών ορίων παρουσιάζει σφάλμα, το οποίο είναι προσπελάσιμο από τα διαγνωστικά μηχανήματα και εφαρμογές.

Μέσω των διαγνωστικών μπορούμε να αναγνώσουμε την πίεση καυσίμου ανά πάσα στιγμή έτσι ώστε να προβούμε σε διάγνωση.

Η πίεση συνήθως μας επιστρέφεται σε μονάδες μέτρησης “Bar” ή “Psi” ανάλογα την κάθε υλοποίηση.

Εικόνα 16: Αισθητήρας Πίεσης Καυσίμου

Πηγή: shorturl.at/joAST

4.2.8 Oxygen Sensor (Αισθητήρας Οξυγόνου)

Τα σύγχρονα αυτοκίνητα συχνά διαθέτουν δύο αισθητήρες οξυγόνου.

Ο πρώτος Αισθητήρας Οξυγόνου είναι τοποθετημένος πριν τον καταλυτικό μετατροπέα και χρησιμοποιείται για να μετρήσει πόσο άκαυστο οξυγόνο υπάρχει στα καυσαέρια. Έτσι ενώ βρίσκεται σε συνεχή επικοινωνία με τον εγκέφαλο του αυτοκινήτου συμβάλει στην βελτιστοποίηση της αναλογίας αέρα - καυσίμου και συνεπώς στην αποδοτικότητα και οικονομία.

Ο δεύτερος αισθητήρας οξυγόνου είναι τοποθετημένος μετά τον καταλυτικό μετατροπέα και είναι υπεύθυνος να μετράει τα ρυπογόνα αέρια που εκπέμπει η εξαγωγή του αυτοκινήτου με σκοπό την ενημέρωση του οδηγού σε περίπτωση που ο καταλυτικός μετατροπέας χρειαστεί αλλαγή λόγω χαμηλής απόδοσης η βλάβης.

Αν αναγνώσουμε τον αισθητήρα οξυγόνου μέσω διαγνωστικού μπορούμε να διαγνώσουμε βλάβες καθώς και να καταλάβουμε αν το αυτοκίνητο κάνει σωστή καύση και αποδίδει σωστά.

Οι αισθητήρες οξυγόνου μας επιστρέφουν συνήθως Volt στο περιθώριο 0.1V έως 0.9V, όπου 0.1 V θεωρούμε ότι το μείγμα είναι “στεγνό”, άρα έχουμε πολύ άκαυστο αέρα, ενώ στα 0.9V έχουμε “πλούσιο” μείγμα άρα λίγο άκαυστο αέρα.

Εικόνα 17: Αισθητήρας Οξυγόνου

Πηγή: shorturl.at/lnFKV

5. Σχεδιασμός εφαρμογής

Το λογισμικό που αναπτύξαμε είναι βασισμένο σε Android 13 και συνδέεται ασύρματα μέσω bluetooth σε οποιοδήποτε OBD-II Universal Adapter.

Βασίζεται σε απλό UI (User Interface) το οποίο το καθιστά εύκολο και άμεσο στο χρήση τόσο για την ανάγνωση σφαλμάτων όσο και για την προβολή διαφόρων παραμέτρων και μετρήσεων.

Η λειτουργία του βασίζεται στην επαναλαμβανόμενη ανάγνωση των αισθητήρων η οποία επιστρέφεται στον χρήστη σε απλό κείμενο, χωρίς να χρειάζεται κάποια μετατροπή από την μεριά του.

Στόχος μας είναι να γίνει πιο προσιτή και κατανοητή η διάγνωση του αυτοκινήτου από τον μέσο χρήστη χωρίς την ύπαρξη περίπλοκων και δυσνόητων παρεμβάσεων.

Η εφαρμογή περιλαμβάνει πέντε βασικές λειτουργίες:

- Gauges
- Data Stream
- Diagnostics
- Contact
- Exit

των οποίων οι λειτουργίες επεξηγούνται παρακάτω καθώς και κάποιες τεχνικές πληροφορίες.

Στο παρακάτω flowchart παρουσιάζεται η λειτουργία της εφαρμογής. Εκτελείται το αρχικό μενού όπου εμφανίζονται οι πέντε βασικές λειτουργίες : Gauges , Diagnostics , Live Data Stream , Contact και τέλος το κουμπί εξόδου.

Για να μπορέσει να εκτελέσει οποιαδήποτε λειτουργία πρέπει να επιτευχθεί η σύζευξη της Android συσκευής με το OBD Adapter.

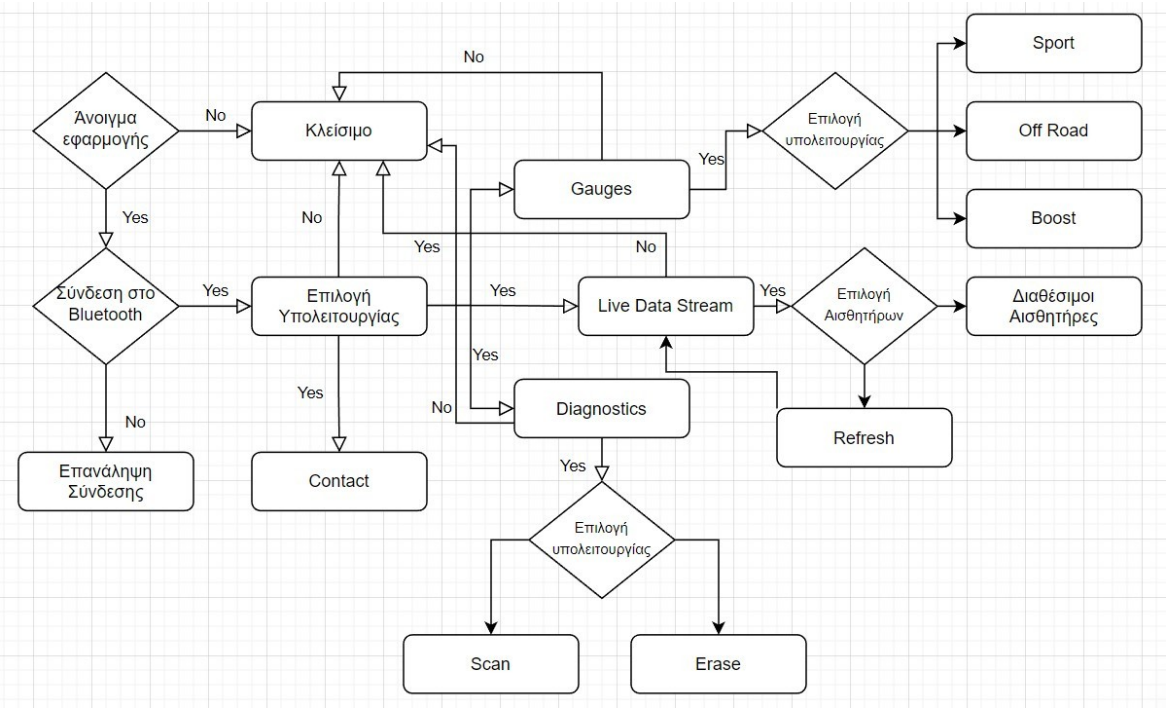
Αν ο χρήστης επιλέξει την λειτουργία Gauges του δίνεται η δυνατότητα επιλογής τριών προεπιλεγμένων οθονών : Sport , Off Road και Boost. Κάθε οθόνη προβάλλει διάφορους αισθητήρες και ο χρήστης επιλέγει ανάμεσα τους ανάλογα με τις ανάγκες του και τις συνθήκες οδήγησης.

Έπειτα αν επιλέξει τη λειτουργία Diagnostics του δίνεται η επιλογή να σαρώσει την μνήμη του εγκεφάλου για σφάλματα καθώς και η επιλογή διαγραφής όλων των αποθηκευμένων σφαλμάτων.

Αν επιλέξει τη λειτουργία Live Data Stream , εκτελείται ένας έλεγχος για τους διαθέσιμους προς ανάγνωση αισθητήρες και έπειτα δίνεται η επιλογή στον χρήστη να προσπελάσει τους διαθέσιμους αισθητήρες του οχήματός του.

Μέσω της αρχικής οθόνης υπάρχει και η δυνατότητα επιλογής της λειτουργίας Contact , στην οποία ο χρήστης μας στέλνει κάποιο μήνυμα είτε για την παροχή βοήθειας σχετικά με το λογισμικό ή για να προτείνει αλλαγές και βελτιώσεις.

Τέλος επιλέγοντας το κουμπί Exit , η εφαρμογή τερματίζει τη λειτουργία της και αποδεσμεύεται η σύνδεση του Bluetooth με το OBD Adapter.



Εικόνα 18: Το flow chart της εφαρμογής

5.1 Σύνδεση μέσω bluetooth

Η πλατφόρμα του Android υποστηρίζει δίκτυα Bluetooth όπου επιτρέπεται ανταλλαγή ασύρματων δεδομένων με άλλες συσκευές. Η πρόσβαση στη λειτουργία Bluetooth γίνεται με την χρήση BluetoothAPI, όπου επιτρέπει στην εφαρμογή να συνδέονται συσκευές Bluetooth, επιτρέποντας ασύρματες λειτουργίες point–to–point και multipoint. Εμείς χρησιμοποιήσαμε την point–to–point καθώς θέλαμε να χρησιμοποιήσουμε αρχιτεκτονική Send–Receive όπου θα εξηγηθεί παρακάτω.

Όλα τα Bluetooth APIs είναι διαθέσιμα με την χρήση του package android.bluetooth και είναι απαραίτητο ώστε να γίνει ύπαρξη σύνδεσης Bluetooth.

Για να επιτευχθεί η επικοινωνία μεταξύ δύο συσκευών με Bluetooth πρέπει πρώτα να σχηματίσουν ένα κανάλι επικοινωνίας χρησιμοποιώντας μια διαδικασία σύζευξης. Η μία συσκευή καθίσταται διαθέσιμη για εισερχόμενα αιτήματα σύνδεσης ενώ η άλλη χρησιμοποιεί μια διαδικασία εντοπισμού υπηρεσίας για τον εντοπισμό της πρώτης. Αφού γίνει η ανίχνευση και η αποδοχή του αιτήματος σύζευξης, οι δύο συσκευές ανταλλάσσουν

κλειδιά ασφαλείας όπου και αποθηκεύονται για μελλοντική χρήση. Όταν ολοκληρωθεί η διαδικασία της σύζευξης οι δύο συσκευές μπορούν να ανταλλάξουν πληροφορίες με τις δύο τεχνικές: point to point ή multipoint. Μόλις τελειώσει η περίοδος σύνδεσης, η συσκευή που ξεκίνησε την διαδικασία της σύζευξης αποδεσμεύει το κανάλι επικοινωνίας. Οι δύο συσκευές παραμένουν συνδεμένες, έτσι ώστε σε μελλοντική χρήση να μπορούν να επανασυνδεθούν αυτόματα, εφόσον βρίσκονται σε εμβέλεια και δεν έχει αφαιρεθεί η σύζευξη μεταξύ τους.

Η διαδικασία εύρεσης απομακρυσμένων συσκευών Bluetooth είτε μέσω σάρωσης είτε με αναζήτηση στη λίστα των συζευγμένων συσκευών γίνεται με την χρήση της λειτουργίας BluetoothAdapter του πακέτου android.bluetooth. Αυτή η λειτουργία μόλις εντοπίσει μια συσκευή που είναι ανιχνεύσιμη, δηλαδή ανταποκρίνεται στο αίτημα εύρεσης και έχει κοινοποιήσει το όνομα της, την κατηγορία της και το μοναδική διεύθυνση MAC (τιμή 48-bit που προσδιορίζει μοναδικά μια συσκευή Bluetooth), εκτελεί την διαδικασία εντοπισμού και μπορεί στην συνέχεια να ξεκινήσει σύνδεση με αυτή την συσκευή που ανιχνεύτηκε.

Η παραπάνω λειτουργία για να επιτευχθεί σε εκδόσεις Android 8.0 (Επίπεδο API 26) ή παλαιότερες χρειάζεται η άδεια πρόσβασης τοποθεσίας. Αυτό συμβαίνει καθώς οι συσκευές με δυνατότητα εντοπισμού ενδέχεται να αποκαλύπτουν δεδομένα σχετικά με την τοποθεσία του χρήστη. Σε νεότερες εκδόσεις γίνεται αυτόματη διαφύλαξη από το ίδιο το Android και δεν χρειάζεται η εφαρμογή να ζητά άδειες τοποθεσίας.

Για να δημιουργηθεί μια σύνδεση μεταξύ δύο συσκευών πρέπει να εφαρμοστούν τεχνικές και από την πλευρά του διακομιστή (server-side) και από την πλευρά του πελάτη (client-side). Η μία συσκευή πρέπει να ανοίξει ένα Server Socket και η άλλη πρέπει να συνδεθεί σε αυτό με την χρήση της διεύθυνσης MAC της συσκευής διακομιστή. Η εφαρμογή που θα λειτουργήσει ως διακομιστής κρατάει ανοιχτό ένα BluetoothServerSocket που έχει σκοπό να ακούει τα εισερχόμενα αιτήματα σύνδεσης. Για να υπάρξει μοναδική σύνδεση μεταξύ διακομιστή–πελάτη χρησιμοποιείται κοινό Universally Unique Identifier (UUID) όπου είναι μια τυποποιημένη μορφή 128-bit. Το OBD–II Adapter που θα λειτουργήσει ως πελάτης πρέπει να φέρει παρόμοιο UUID ώστε το αίτημα σύνδεσης του να γίνει αποδεκτό από το BluetoothServerSocket του διακομιστή. Ο διακομιστής και ο πελάτης θεωρούνται

συνδεδεμένοι όταν έχουν συνδεδεμένο BluetoothSocket στο ίδιο κανάλι RFCOMM (Radio Frequency communication: σύνολο πρωτοκόλλων μεταφοράς).

Όταν και διακομιστής και πελάτης είναι συνδεδεμένοι σε ένα BluetoothSocket μπορεί να ξεκινήσει η διαδικασία μεταφοράς δεδομένων μεταξύ των δύο συσκευών. Η μεταφορά γίνεται με InputStream και OutputStream με τις μεθόδους getInputStream() και getOutputStream() αντίστοιχα. Η ανάγνωση και εγγραφή των δεδομένων γίνεται με τις read(byte[]) και write(byte[]). Μια ιδιαιτερότητα στην ανάγνωση και εγγραφή των δεδομένων είναι ότι πρέπει να χρησιμοποιηθεί συγκεκριμένο νήμα ροής. Αυτό συμβαίνει καθώς η μέθοδος read(byte[]) μπλοκάρει έως ότου υπάρχει κάτι προς ανάγνωση από την ροή. Η μέθοδος write(byte[]) συνήθως δεν μπλοκάρει, αλλά μπορεί να μπλοκάρει αν δεν γίνει κλήση της read(byte[]) αρκετά γρήγορα. Για να αποφύγουμε κάτι τέτοιο χρησιμοποιήσαμε την μέθοδο send–receive ή αλλιώς point–to–point ώστε να έχουμε ελεγχόμενη ροή δεδομένων.

```
public void write(byte[] buffer) {
    try {

        mmOutStream.write(buffer);

    } catch (IOException e) {
        Log.e(TAG,  msg: "Exception during write", e);
    }
}
```

Εικόνα 20: Η μέθοδος write(byte[]) του διακομιστή

```
public void write(byte[] out) {
    // Create temporary object
    ConnectedThread r;
    // Synchronize a copy of the ConnectedThread
    synchronized (this) {
        if (mState != STATE_CONNECTED) return;
        r = mConnectedThread;
    }
    // Perform the write unsynchronized
    r.write(out);
}
```

Εικόνα 19: Η μέθοδος write(byte[]) της συσκευής

5.2 Η λειτουργία Gauges

Στην πρώτη θέση του αρχικού μενού της εφαρμογής υπάρχει η λειτουργία Gauges, η αλλιώς στην ελληνική μετάφραση “μετρητές” .

Μπαίνοντας στην λειτουργία ο χρήστης έρχεται σε επαφή με τρεις διαφορετικές υπολειτουργίες και έντεκα μετρητές αισθητήρων.
Αφού συνδεθεί η κινητή συσκευή του χρήστη με το OBD adapter, ξεκινάει ο κάθε μετρητής να μεταδίδει μια πληροφορία από τον εγκέφαλο του αυτοκινήτου, ασύρματα και επαναλαμβανόμενα στο περιβάλλον διεπαφής.

Οι τρεις κατηγορίες είναι χωρισμένες σε “Sport”, “Off Road” και “Boost” έτσι ώστε να παρέχουν στον χρήστη πληροφορίες σχετικά με τις οδικές συνθήκες αλλά και τις ανάγκες του χρήστη.

Η κατηγορία Sport, είναι βελτιστοποιημένη για να παρέχει πληροφορίες χρήσιμες για χρήση επιδόσεων, όπως για παράδειγμα η οδήγηση σε μια πίστα αυτοκινήτου. Με την χρήση κατάλληλου κώδικα επιστρέφει πληροφορίες όπως, Στροφές Κινητήρα, μοίρες προ πορείας ανάφλεξης, φορτίο κινητήρα, θερμοκρασία ψυκτικού υγρού, πίεση εισαγωγής αέρα και θερμοκρασία εισαγόμενου αέρα.

Η δεύτερη κατηγορία ονομάζεται “Off Road” αφού στόχος πλέον είναι η βελτιστοποίηση της ασφάλειας και της οδήγησης εκτός δρόμου. Στην κατηγορία “Off Road”. Έχουμε πληροφορίες όπως Στροφές κινητήρα, θερμοκρασία νερού, γεωγραφικό μήκος και πλάτος και πυξίδα.

Τέλος, η τελευταία κατηγορία ονομάζεται “Boost” έχει έναν μόνο μετρητή, την πίεση εισαγωγής ή αλλιώς πίεση υπερπλήρωσης και έχει έναν μοναδικό σκοπό, την μέτρηση της υπερπλήρωσης του κινητήρα από την τουρμπίνα που πιθανών να εξοπλίζεται το αυτοκίνητο. Ο δείκτης προβάλλει σε πραγματικό χρόνο και επαναλαμβανόμενα την πίεση της εισαγωγής, έτσι ώστε ο χρήστης να μπορέσει να διαγνώσει πιθανά προβλήματα ή να πραγματοποιήσει μια απλή μέτρηση της πίεσης.

Ο ευρύς σκοπός αυτής της λειτουργίας είναι να εισάγει ομαλά τον χρήστη στην εφαρμογή και να του προβάλει προεπιλεγμένες χρήσιμες πληροφορίες χωρίς να χρειαστεί να επιλέξει ο ίδιος χειροκίνητα συγκεκριμένους αισθητήρες και μετρήσεις.

5.2.1 Υλοποίηση συνάρτησης Gauges

Για να φτάσει η πληροφορία του κάθε μετρητή στην διεπαφή του χρήστη χρειάστηκαν κάποιες βασικές διαδικασίες. Πρώτη και βασικότερη είναι η σύνδεση του χρήστη με το OBD – II Adapter με την χρήση Bluetooth. Χωρίς την σύνδεση δεν μπορεί να προχωρήσει καμία διαδικασία από τις παρακάτω. Η σύνδεση μπορεί να πραγματοποιηθεί με το άγγιγμα της εικόνας με το εικονίδιο του Bluetooth στο πάνω δεξί μέρος της οθόνης και στην συνέχεια την επιλογή του OBD–II Adapter. Όταν γίνει επιτυχής η σύνδεση σημαίνει ότι μπορούμε πλέον να στείλουμε μηνύματα στον εγκέφαλο του αυτοκινήτου. Τα δύο πρώτα μηνύματα που θα σταλθούν στον εγκέφαλο του αυτοκινήτου είναι τα ATSP0 και

ATDP. Για να σταλθεί οποιοδήποτε μήνυμα και να θεωρηθεί αποδεκτό από τον εγκέφαλο του αυτοκινήτου πρέπει να δεχθεί κωδικοποίηση σε bytes με την χρήση της μεθόδου getBytes.

ATSP0: Αυτόματος εντοπισμός πρωτοκόλλου επικοινωνίας.
ATDP: Επιστρέφει το διαγνωστικό πρωτόκολλο που χρησιμοποιείται μετά την παραπάνω εντολή.

```
public void initCommands() {

    String protocol = "ATSP0";
    message_PID(protocol);


    String verify = "ATDP";
    message_PID(verify);

    Toast.makeText( context: this,  text: "Init Commands", Toast.LENGTH_LONG).show();

}
```

Εικόνα 21: Μηνύματα αρχικοποίησης για την επικοινωνία με τον εγκέφαλο του αυτοκινήτου

Αφού σταλθούν οι δύο παραπάνω εντολές ο εγκέφαλος του αυτοκινήτου μπορεί να απαντήσει σε οποιαδήποτε εντολή του σταλεί. Στην συγκεκριμένη λειτουργία θέλουμε απαντήσεις από αναγνώσεις αισθητήρων σε πραγματικό χρόνο. Για κάθε αισθητήρα πρέπει να σταλεί διαφορετική εντολή για να επιστρέψει ο εγκέφαλος την τιμή του αισθητήρα. Στο παρακάτω πίνακα φαίνονται οι εντολές που πρέπει να σταλούν για κάθε αισθητήρα που χρησιμοποιήθηκε στην λειτουργία.

Στροφές Κινητήρα	010C
Μοίρες προ πορείας ανάφλεξης	010E
Φορτίο Κινητήρα	0104
Θερμοκρασία Ψυκτικού Υγρού	0105
Πίεση Εισαγωγής Αέρα	010B
Θερμοκρασία Εισαγόμενου Αέρα	010F

Εικόνα 22: Εντολές για την λήψη δεδομένων από τους αισθητήρες

Αφού σταλούν οι παραπάνω κωδικοί ο εγκέφαλος στέλνει απαντήσεις για τον καθένα ξεχωριστά. Μια απάντηση για τις στροφές κινητήρα για παράδειγμα είναι της μορφής 41 0C 1A F8. Η κάθε απάντηση αποτελείται από δύο σκέλη. Συγκεκριμένα στο παράδειγμα το πρώτο σκέλος είναι το 41 0C. Το 41 σημαίνει ότι το συγκεκριμένο μήνυμα είναι απάντηση από τον εγκέφαλο του αυτοκινήτου και το 0C δείχνει από ποιον αισθητήρα έρχεται αυτή η απάντηση. Το υπόλοιπο μέρος της απάντησης κρύβει την τιμή που θέλουμε από τον κάθε αισθητήρα. Η τιμή αυτή για να φτάσει από το 1A F8 χρειάζεται κάποια κωδικοποίηση. Συγκεκριμένα για τις στροφές κινητήρα πρέπει να γίνει κωδικοποίηση σε δεκαδικό αριθμό και μετά η διαίρεση του με το τέσσερα.

```
if (ecu_message.toUpperCase().startsWith("41 0C")) {
    rpm_value = ecu_message.substring(6, 11);
    rpm_value = rpm_value.replaceAll( regex: "\\s+",  replacement: "");
    calculated = (int) Math.round(Integer.decode("0x" + rpm_value.toUpperCase()) / 4);
    rpmText.setText(Double.toString(calculated), TextView.BufferType.EDITABLE);

}
```

Εικόνα 23: Κώδικας για την μετατροπή της απάντησης από τον εγκέφαλο του αυτοκινήτου

Στον παρακάτω πίνακα φαίνονται οι κωδικοποιήσεις που χρειάζονται να γίνουν σε κάθε απάντηση τιμής αισθητήρα από τον εγκέφαλο του αυτοκινήτου.

Στροφές Κινητήρα	A / 4
Μοίρες προ πορείας ανάφλεξης	A / 2 - 64
Φορτίο Κινητήρα	A / 2.55
Θερμοκρασία Ψυκτικού Υγρού	A - 40
Πίεση Εισαγωγής Αέρα	A
Θερμοκρασία Εισαγόμενου Αέρα	A - 40

Εικόνα 24: Μετατροπές των δεδομένων από την απάντηση του εγκεφάλου

Όπου A είναι η τιμή που αναφέραμε παραπάνω (1A F8) αφού δεχθεί κωδικοποίηση σε Bytes με την μέθοδο getBytes.

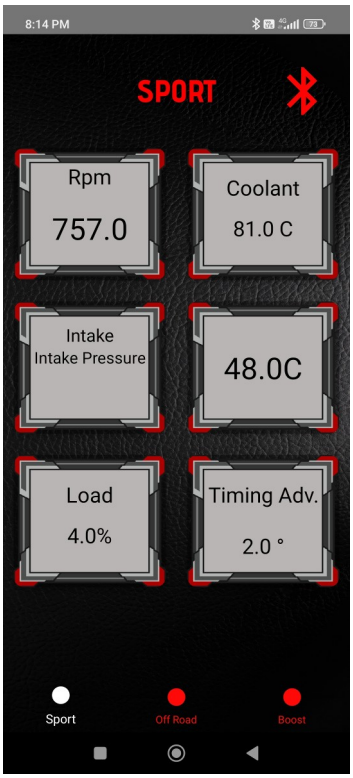
Εφόσον έχουμε τις κωδικοποιήσεις των απαντήσεων από τους αισθητήρες τώρα βασικό βήμα είναι η δημιουργία της ροής των δεδομένων. Αρχικά δημιουργείται η ροή των αιτημάτων που στέλνονται στον εγκέφαλο του αυτοκινήτου για τις τιμές των αισθητήρων. Αυτή η ροή χρειάζεται, καθώς κάθε αίτημα που στέλνεται στον εγκέφαλο επιστρέφει μία

και μοναδική απάντηση. Για να υπάρχει λοιπόν συνεχόμενη ροή σε πραγματικό χρόνο θα πρέπει να στέλνονται συνεχόμενα συγχρονισμένα αιτήματα προς τον εγκέφαλο και να λαμβάνονται αντίστοιχα οι απαντήσεις χωρίς πρόβλημα. Για την συνεχόμενη και συγχρονισμένη ροή των αιτημάτων χρησιμοποιήθηκαν τα αντικείμενα Timer και TimerTask. Η Timer επιτρέπει σε νήματα (threads) να προγραμματίσει διεργασίες για μεταγενέστερη εκτέλεση με βάση την περιοδικότητα της εργασίας καθώς και την καθυστέρηση μεταξύ τους. Η TimerTask ουσιαστικά είναι η εργασία που θα μοιραστεί στα νήματα με τους περιορισμούς της περιοδικότητας και της καθυστέρησης που θα έχει τεθεί. Για τις απαντήσεις του εγκεφάλου και ταυτόχρονα την αρχιτεκτονική point-to–point του Bluetooth που εξηγήθηκε παραπάνω χρησιμοποιήθηκε το αντικείμενο Handler. Ο κύριος λόγος επιλογής του είναι ότι μπορεί να λαμβάνει μηνύματα καταγραφής από το Bluetooth και ταυτόχρονα να δέχεται συγχρονισμένες απαντήσεις από τον εγκέφαλο του αυτοκινήτου χωρίς να τις μπλοκάρει. Αυτό γίνεται καθώς χρησιμοποιεί διαφορετικό νήμα από αυτό της Timer και περιμένει με κατάλληλο κώδικα να γίνει πρώτα το αίτημα ώστε να στείλει μετά την απάντηση.

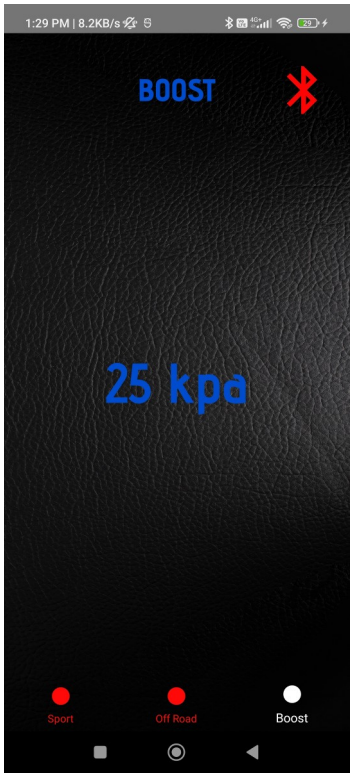
5.2.2 Στιγμιότυπα Οθόνης

Εικόνα 28: Αρχικό μενού

Εικόνα 27: Υπολειτουργία off road



Εικόνα 26: Υπολειτουργία sport



Εικόνα 25: Υπολειτουργία boost

5.3 Η λειτουργία Live Data Stream

Το δεύτερο κατά σειρά στοιχείο στην διεπαφή χρήστη ονομάζεται Live Data Stream ή αλλιώς Ζωντανή Ροή Δεδομένων.

Η λειτουργία Live Data Stream δίνει την δυνατότητα στον χρήστη να επιλέξει αυτόνομα τα δεδομένα που επιθυμεί να προβληθούν στην οθόνη του μέσα από μια πλήρη λίστα αισθητήρων και πληροφοριών.

Όπως και στην λειτουργία Gauges έτσι και στην Live Data Stream ο χρήστης συνδέεται μέσω Bluetooth στο OBD Adapter και έχει έπειτα την δυνατότητα να επιλέξει τους αισθητήρες και τις πληροφορίες που θέλει να αναγνώσει από τον εγκέφαλο του αυτοκινήτου.

Οι επιλογές του χρήστη θα φανούν σε μια λίστα (ListView) στην οθόνη του και θα ενημερώνονται σε πραγματικό χρόνο, μέχρι να διακόψει ο χρήστης την σύνδεση με το OBD Adapter ή να σβήσει το αυτοκίνητο.

Οι επιλογές του χρήστη μέχρι σήμερα είναι

- Coolant Temperature (Θερμοκρασία ψυκτικού υγρού)
- Engine Load (Φορτίο κινητήρα)
- Fuel Pressure (Πίεση Καυσίμου)
- Intake Air Temperature (Θερμοκρασία εισαγόμενου αέρα)
- Intake Pressure (Πίεση εισαγωγής)
- Long Term Fuel Trim Bank 1 (Μακροπρόθεσμη περικοπή καυσίμου)
- Long Term Fuel Trim Bank 2 (Μακροπρόθεσμη περικοπή καυσίμου)
- RPM (Στροφές ανά λεπτό)
- Short Term Fuel Trim Bank 1 (Βραχυπρόθεσμη περικοπή καυσίμου)
- Short Term Fuel Trim Bank 2 (Βραχυπρόθεσμη περικοπή καυσίμου)
- Throttle Position (Θέση πεντάλ γκαζιού)
- Timing Advance (προ πορεία ανάφλεξης)
- Vehicle Speed (Ταχύτητα οχήματος)

Η διαδικασία ζωντανής ροής δεδομένων είναι αναπόσπαστο κομμάτι στην διάγνωση του αυτοκινήτου αφού παρέχει γρήγορα και με ακρίβεια πληροφορίες οι οποίες χρησιμεύουν στην εύρεση προβλημάτων, τα οποία πιθανών να μην έχουν ενεργοποιήσει την λυχνία

βλάβης έτσι ώστε να διαγνωστούν μέσω κωδικού βλάβης (DTC–Diagnostic Trouble Code) .

5.3.1 Υλοποίηση λειτουργίας Live Data Stream

Ομοίως όπως και στην λειτουργία Gauges ο χρήστης πρέπει να συνδεθεί στο Bluetooth με το άγγιγμα της εικόνας στο πάνω δεξί μέρος της οθόνης και την επιλογή της OBD συσκευής έχοντας την τοποθετημένη σωστά στο αυτοκίνητο του. Αφού συνδεθεί στέλνονται αυτόματα οι δύο εντολές για τον εντοπισμό πρωτοκόλλου ATSP0 και

ATDP σε μορφή bytes όπως έχει αναλυθεί και στην παραπάνω λειτουργία.

Ο χρήστης στη συνέχεια με το άγγιγμα του κουμπιού στο πάνω μέρος της οθόνης μπορεί να επιλέξει ποιους αισθητήρες θέλει να διαβάσει. Αφού γίνει η επιλογή γίνεται έλεγχος αν όλοι οι αισθητήρες που έχει επιλέξει ο χρήστης υποστηρίζονται από το αυτοκίνητο του.

Η διαδικασία του ελέγχου περιέχει τα εξής στάδια:

Αρχικά στέλνεται η εντολή “0100” στον εγκέφαλο του αυτοκινήτου ώστε να επιστρέψει τους διαθέσιμους αισθητήρες που διαθέτει. Η απάντηση είναι της μορφής “BE1FA813” και χρειάζεται κωδικοποίηση ώστε να καταλάβουμε ποιους αισθητήρες διαθέτει το αυτοκίνητο. Η πρώτη κωδικοποίηση που πραγματοποιείται είναι η μετατροπή της απάντησης σε δυαδικό σύστημα. Άρα καταλήγουμε να έχουμε έναν αριθμό δυαδικής μορφής όπου κάθε του ψηφίο σημαίνει αν υποστηρίζεται ή όχι ένας αισθητήρας. Συγκεκριμένα όπου “1” σημαίνει ότι ο αισθητήρας υποστηρίζεται και όπου ¨0¨ ότι δεν υποστηρίζεται. Αυτός ο δυαδικός αριθμός ξεκινάει σε σειρά από τον πρώτο αισθητήρα με κωδικό ¨01¨ έως και τον αισθητήρα με κωδικό “20”. Στον παρακάτω πίνακα φαίνεται μία απάντηση από έναν εγκέφαλο αυτοκινήτου και αναλυτικά ποιοι αισθητήρες υποστηρίζονται.
Παρακάτω φαίνονται οι δύο συναρτήσεις :

- supportedPids

  Η supportedPids αναλαμβάνει την αποστολή της εντολής “0100” στον εγκέφαλο του αυτοκινήτου έτσι ώστε να επιστραφεί σε δεκαεξαδική μορφή η απάντηση από τον εγκέφαλο.

```java
public void supportedPids(View v){

    String supportedPids = "0100";
    message_PID(supportedPids);
    Toast.makeText( context: this,  text: "Checking available PIDS", Toast.LENGTH_SHORT).show();

    Button sellectPids = (Button) findViewById(R.id.sellectPids);
    Button scanPIDS = (Button) findViewById(R.id.scanPIDS);

    sellectPids.setVisibility(View.VISIBLE);
    //scanPIDS.setVisibility(View.INVISIBLE);

}
```

Εικόνα 29: Συνάρτηση supportedPids

- HexToBinary

  Η HexToBinary αναλαμβάνει την μετατροπή της δεκαεξαδικής απάντησης του εγκεφάλου σε δυαδική μορφή. Από τη δυαδική μορφή με τη βοήθεια του παρακάτω πίνακα (βλέπε Εικόνα 29) συγκεντρώνουμε την λίστα των διαθέσιμων προς ανάγνωση αισθητήρων.

```java
public String hexToBinary(String hex) {

    String binary = "";
    hex = hex.toUpperCase();

    // initializing the HashMap class
    HashMap<Character, String> hashMap
            = new HashMap<~>();

    // storing the key value pairs
    hashMap.put('0', "0000");
    hashMap.put('1', "0001");
    hashMap.put('2', "0010");
    hashMap.put('3', "0011");
    hashMap.put('4', "0100");
    hashMap.put('5', "0101");
    hashMap.put('6', "0110");
    hashMap.put('7', "0111");
    hashMap.put('8', "1000");
    hashMap.put('9', "1001");
    hashMap.put('A', "1010");
    hashMap.put('B', "1011");
    hashMap.put('C', "1100");
    hashMap.put('D', "1101");
    hashMap.put('E', "1110");
    hashMap.put('F', "1111");
```

Εικόνα 31: Συνάρτηση HexToBinary

```java
    int i;
    char ch;

    for (i = 0; i < hex.length(); i++) {
        // extracting each character
        ch = hex.charAt(i);

        if (hashMap.containsKey(ch))

            binary += hashMap.get(ch);

        else {
            binary = "Invalid Hexadecimal String";
            return binary;
        }
    }
    char[] availablePIDS = binary.toCharArray();

    for (int j = 0; j < availablePIDS.length; j++) {

        if (binary.charAt(j) == '1') {

            supportedPIDS.add(PIDS[j]);
        }
    }
    // returning the converted Binary
    return binary;
}
```

Εικόνα 30: Συνάρτηση HexToBinary

Δεκαεξαδικό	Δυαδικό	Υποστηρίζονται	Κωδικός PID
B	1	Ναι	1
	0	Όχι	2
	1	Ναι	3
	1	Ναι	4
E	1	Ναι	5
	1	Ναι	6
	1	Ναι	7
	0	Όχι	8
1	0	Όχι	9
	0	Όχι	0A
	0	Όχι	0B
	1	Ναι	0C
F	1	Ναι	0D
	1	Ναι	0E
	1	Ναι	0F
	1	Ναι	10
A	1	Ναι	11
	0	Όχι	12
	1	Ναι	13
	0	Όχι	14
8	1	Ναι	15
	0	Όχι	16
	0	Όχι	17
	0	Όχι	18
1	0	Όχι	19
	0	Όχι	1A
	0	Όχι	1B
	1	Ναι	1C
3	0	Όχι	1D
	0	Όχι	1E
	1	Ναι	1F
	1	Ναι	20

Πηγή: https://en.wikipedia.org/wiki/OBD-II_PIDs

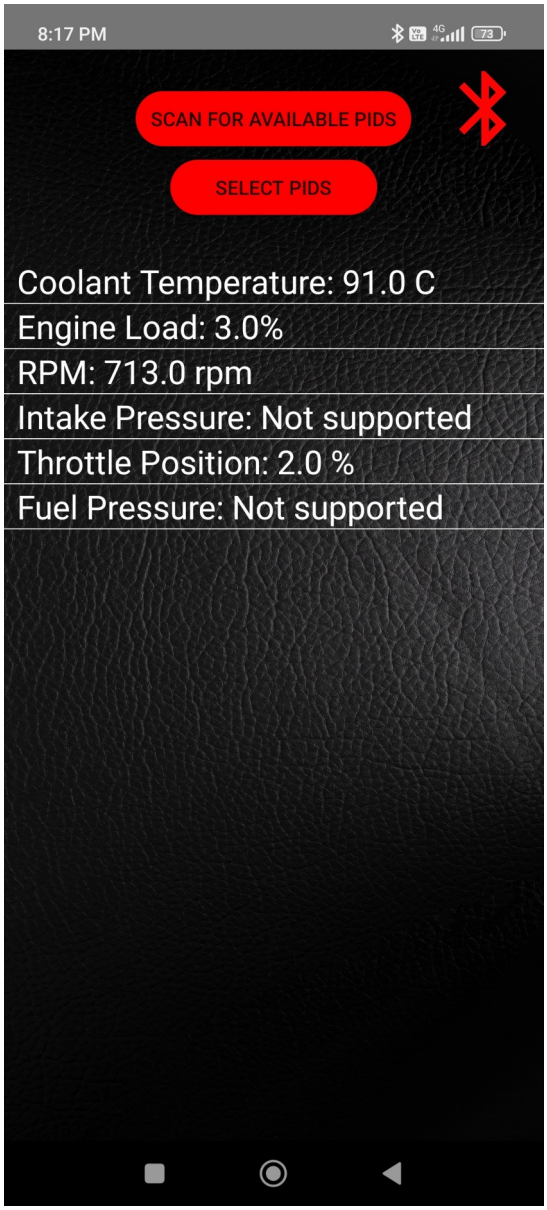
Εικόνα 32: Απάντηση εγκεφάλου αυτοκινήτου για υποστηριζόμενους αισθητήρες

Όταν γίνει ο παραπάνω έλεγχος για τους υποστηριζόμενους αισθητήρες επιστέφονται σε μορφή λίστας οι αισθητήρες που επίλεξε ο χρήστης με την τιμή τους σε πραγματικό χρόνο. Οι αισθητήρες που επίλεξε και δεν υποστηρίζονται εμφανίζονται στην λίστα με τον κωδικό του αισθητήρα και το μήνυμα “Not supported”.

Δίνεται και η επιλογή στον χρήστη να κάνει επανεκκίνηση την λειτουργία και να επιλέξει άλλους αισθητήρες για ανάγνωση πατώντας το κουμπί “Refresh” που έχει αντικαταστήσει το προηγούμενο κουμπί επιλογής αισθητήρων.

5.3.2 Στιγμιότυπα Οθόνης

Αρχικό μενού



Εικόνα 33: Λειτουργία Live Data Stream

5.4 Η λειτουργία Diagnostics

Χαρακτηριστικό εργαλείο στη διάγνωση των βλαβών ενός αυτοκινήτου είναι η ανάγνωση των διαγνωστικών κωδικών σφαλμάτων (Diagnostic Trouble Codes DTCs).

Η λειτουργία Diagnostics σαρώνει τον εγκέφαλο του αυτοκινήτου για τυχόν κωδικούς σφαλμάτων και τους προβάλει στην διεπαφή του χρήστη.
Χρήσιμο χαρακτηριστικό του λογισμικού μας είναι η αναζήτηση του κωδικού βλάβης για παραπάνω πληροφορίες μέσω browser (πχ Google Chrome). Ο χρήστης έχει την δυνατότητα να πατήσει πάνω στον διαγνωστικό κωδικό που έχει επιστρέψει ο εγκέφαλος του αυτοκινήτου του και θα αναδυθεί ένα παράθυρο στον προεπιλεγμένο browser του κινητού τηλεφώνου το οποίο θα κάνει αναζήτηση για πληροφορίες και διορθώσεις του εκάστοτε κωδικού βλάβης.
Επιπρόσθετα, υπάρχει η δυνατότητα της διαγραφής των κωδικών βλάβης από την μνήμη του εγκεφάλου, πατώντας το κουμπί Erase Codes μέσω της λειτουργίας Diagnostics.
Οι κωδικοί βλάβης ανάλογα το είδος τους μπορεί να έχουν την μορφή:

- Pxxxx για Powertrain: Αναφέρονται στο μηχανικό σύνολο
- Bxxxx για Body: Αναφέρονται συνήθως στις λειτουργίες μέσα στην καμπίνα όπως συστήματα ασφαλείας, άνεσης και βοήθειας.
- Cxxxx για Chassis: Αναφέρονται συνήθως στις λειτουργίες εκτός της καμπίνας όπως φρένα, ανάρτηση και διεύθυνση.
- Uxxxx αναφέρονται στις λειτουργίες μεταξύ των ηλεκτρονικών και των υπολογιστών του αυτοκινήτου

5.4.1 Υλοποίηση Λειτουργίας Diagnostics

Ακολουθώντας το λειτουργικό μοτίβο των προηγούμενων λειτουργιών, η εφαρμογή απαιτεί από τον χρήστη να συνδεθεί χειροκίνητα μέσω Bluetooth στην OBD συσκευή, η οποία πρέπει να είναι τοποθετημένη στην κατάλληλη θύρα του αυτοκινήτου.

Η αποστολή μηνυμάτων αυτής της λειτουργίας ξεκινάει με την αποστολή των πρώτων μηνυμάτων ATSP0 και ATDP όπως και στις παραπάνω κλάσεις με σκοπό την αρχικοποίηση της σύνδεσης μέσω της συνάρτησης initCommands().
Όπως αναφέρουμε και παραπάνω ισχύει ότι:

- ATSP0: Αυτόματος εντοπισμός πρωτοκόλλου επικοινωνίας.
- ATDP: Επιστρέφει το διαγνωστικό πρωτόκολλο που χρησιμοποιείται μετά την παραπάνω εντολή.

Έπειτα, αφού ολοκληρωθεί η σύνδεση και η αποστολή των παραπάνω εντολών η λειτουργία της λειτουργίας συνεχίζεται αφού ο χρήστης επιλέξει μια από τις δύο διαθέσιμες λειτουργίες, Scan ή Erase Faults.

Στην περίπτωση που ο χρήστης επιλέξει την επιλογή Scan, τότε τρέχει η συνάρτηση ScanFaultCommand η οποία στέλνει στον εγκέφαλο του αυτοκινήτου την εντολή "03" μέσω της συνάρτησης messagePID. Η εντολή "03" ξεκινάει τη σάρωση στη μνήμη του εγκεφάλου του αυτοκινήτου με σκοπό την εύρεση σφαλμάτων.

Την διαχείριση της αποστολής και λήψης μηνυμάτων αναλαμβάνει η συνάρτηση Handler, η οποία και είναι υπεύθυνη για την συνεχή επικοινωνία της συσκευής OBD με το κινητό τηλέφωνο του χρήστη.

Αφού ολοκληρωθεί η σάρωση του εγκεφάλου επιστρέφεται η απάντηση η οποία περιλαμβάνει τα σφάλματα. Αξίζει να σημειωθεί ότι αν τα σφάλματα δεν χωράνε σε μια απάντηση τότε επιστρέφεται κατευθείαν άλλη μια που περιέχει τα επόμενα σφάλματα και ούτω καθεξής.

Μια επιστροφή απάντησης από τον εγκέφαλο μπορεί να έχει τη μορφή 43 03 25 01 71 04 20 και κάθε τέτοια απάντηση περιλαμβάνει μέχρι τρία σφάλματα. Το "43" στην αρχή της απάντησης μας ενημερώνει ότι δεχόμαστε απάντηση από τον εγκέφαλο του αυτοκινήτου σε αίτημα μηνύματος τύπου "03". Το υπόλοιπο μήνυμα χωρισμένο σε τρία κομμάτια είναι τα σφάλματα του αυτοκινήτου. Συγκεκριμένα "0325", "0171" και "0420" που σημαίνουν αντίστοιχα "Πυράκια στον κινητήρα", "Στεγνό μείγμα καυσίμου" και "Βλάβη στον καταλύτη".

Αν ο εγκέφαλος δεν ανιχνεύσει σφάλματα συνήθως η απάντηση που επιστρέφει είναι "0000 0000".

Η εφαρμογή, αφού αποκωδικοποιήσει τα μηνύματα μέσω της συνάρτησης scanfault, τα επιστρέφει σε μια λίστα στο περιβάλλον διεπαφής του χρήστη.

Στην περίπτωση που ο χρήστης επιλέξει να διαγράψει τα σφάλματα από τη μνήμη του εγκεφάλου και πατήσει το κουμπί Erase Faults, τότε μέσω της συνάρτησης eraseFaultCommand και messagePID στέλνεται το μήνυμα "04".

Το μήνυμα αυτό λειτουργεί και πάλι ως εντολή στον εγκέφαλο και τον προτρέπει στην διαγραφή όλων των σφαλμάτων από την μνήμη, χωρίς να απαιτείται κάποια άλλη διεργασία πέρα από την αποστολή του παραπάνω μηνύματος.

Αφού διαγραφούν τα σφάλματα η εφαρμογή μέσω "Toast" μας επιστρέφει την επιβεβαίωση ότι η εντολή απεστάλη επιτυχώς και τα σφάλματα έχουν πλέον διαγραφεί από την μνήμη του εγκεφάλου.

Στο παρακάτω στιγμιότυπο φαίνονται οι τρεις συναρτήσεις:

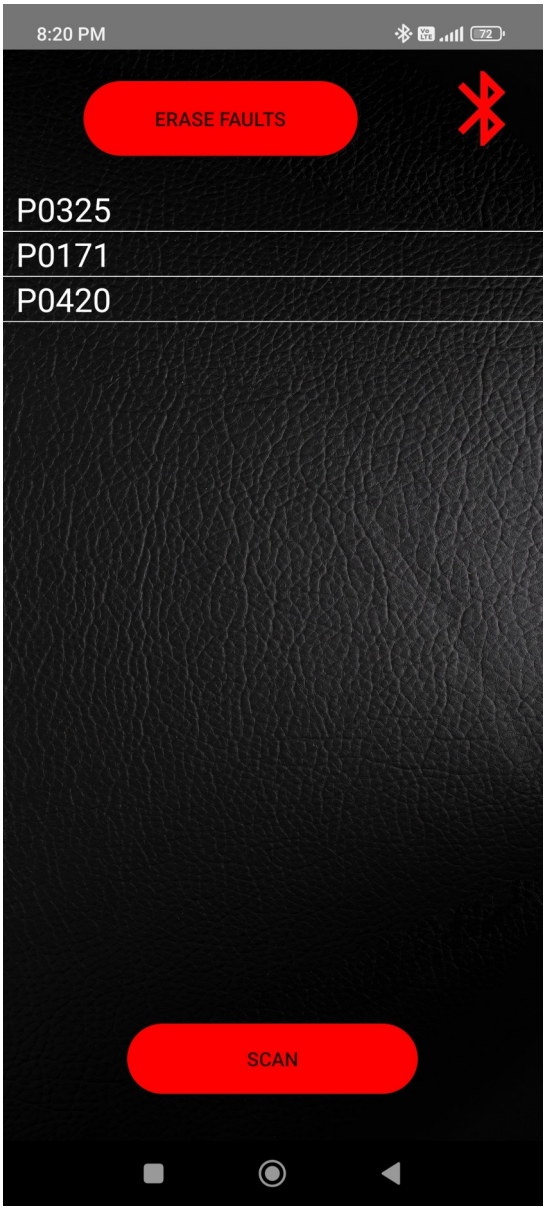
- ScanFaultCommand
- eraseFaultCommand
- messagePID

```
public void eraseFaultCommand(View v) {

    String message = "04";
    message_PID(message);
    Toast.makeText( context: this,  text: "Faults deleted", Toast.LENGTH_LONG).show();

}

public void scanFaultCommand(View v) {

    String message = "03";
    message_PID(message);
    Toast.makeText( context: this,  text: "Scaning", Toast.LENGTH_LONG).show();

}

public void message_PID(String pid) {

    if (BTService.getState() == BtHelper.STATE_CONNECTED) {

        if (pid.length() > 0) {
            pid = pid + "\r";
            byte[] send_pid = pid.getBytes(StandardCharsets.UTF_8);
            BTService.write(send_pid);
        }
    }
}
```

Εικόνα 34: Απόσπασμα κώδικα των τριών συναρτήσεων scanFaultCommand, eraseFaultCommand, message_PID

5.4.2 Στιγμιότυπα Οθόνης

Αρχικό μενού



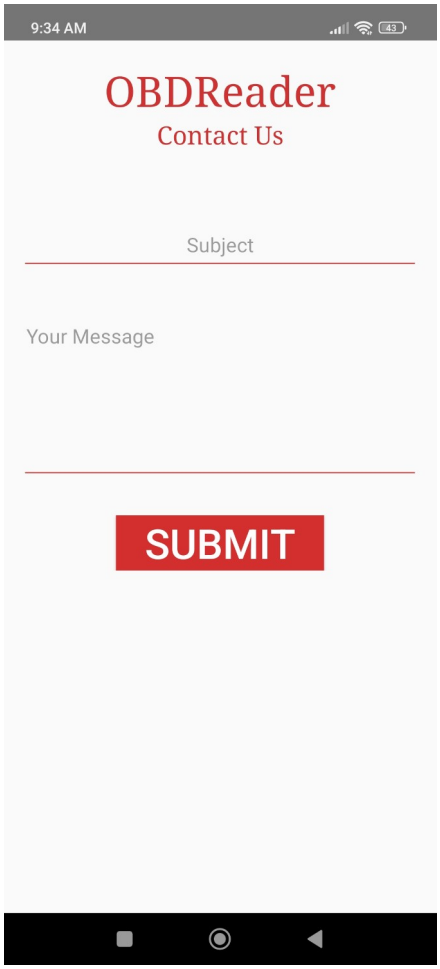
Εικόνα 35: Λειτουργία Diagnostics

5.5 Η λειτουργία Contact

Η τελευταία λειτουργία ονομάζεται Contact και έχει ως στόχο την επικοινωνία με τον χρήστη για τυχόν προβλήματα αλλά και προτάσεις για μελλοντικές αναβαθμίσεις της εφαρμογής.

Ο χρήστης συμπληρώνει σε μια φόρμα το email του και το κείμενο που επιθυμεί να αποστείλει και εμείς λαμβάνουμε το μήνυμα του στο email μας έτσι ώστε να βοηθήσουμε με την επίλυση του εκάστοτε προβλήματος ή να λάβουμε την ανάλογη κριτική για την εφαρμογή.

5.5.2 Στιγμιότυπα Οθόνης



Εικόνα 36: Λειτουργία Contact

6. Κύρια Προβλήματα κατά την υλοποίηση

Κατά την υλοποίηση της εφαρμογής αντιμετωπίστηκαν αρκετές δυσκολίες και προβλήματα τόσο στην λειτουργική όσο και στην αρχιτεκτονική μορφή του λογισμικού.

Σαφώς οι λειτουργικές δυσκολίες είναι πολύ περισσότερες από τις αρχιτεκτονικές μέχρι να καταφέρουμε να πετύχουμε την σύνδεση και λήψη απάντησης από τον εγκέφαλο του αυτοκινήτου. Η πρώτη δυσκολία που αντιμετωπίσαμε ήταν η ομαλή σύνδεση της εφαρμογής μας με την OBD-II συσκευή.

Για αρχή ήταν αναγκαία η μελέτη στο πως λειτουργεί η σύνδεση σε εξωτερική συσκευή μέσω Bluetooth.
Αντιμετωπίσαμε προβλήματα τα οποία δεν μας επέτρεπαν να συνδεθούμε στο Bluetooth μέσω της εφαρμογής και ευθύνονταν σε δυο παράγοντες. Ο πρώτος παράγοντας ήταν ότι πρέπει η πρώτη σύνδεση - σύζευξη του κινητού τηλεφώνου και του αντάπτορα πρέπει να γίνει μέσω των εργοστασιακών ρυθμίσεων του κινητού. Αφού πραγματοποιηθεί για πρώτη φορά η σύζευξη, η συσκευή εμφανίζεται κανονικά στην λίστα των διαθέσιμων συσκευών Bluetooth.
Ο δεύτερος παράγοντας είχε να κάνει με τα “app permissions”, όπου λύσαμε το πρόβλημα προσθέτοντας τα παρακάτω τέσσερα νέα Permissions στο αρχείο manifest του Project.
1) <uses-permission android:name="android.permission.BLUETOOTH" />
2) <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
3) <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
4) <uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH" />
```

Εικόνα 37: Τα δικαιώματα που χρειάστηκαν να προστεθούν στο manifest της εφαρμογής

Αφού λύσαμε τα παραπάνω προβλήματα με την σύνδεση στο Bluetooth έπρεπε να επιλέξουμε ποια πρωτόκολλα επικοινωνίας θα χρησιμοποιήσουμε και θα στείλουμε στον εγκέφαλο. Μετά από κάποια μελέτη καταλήξαμε ότι η επιλογή αυτόματου εντοπισμού πρωτοκόλλου είναι η πιο αξιόπιστη λύση καθώς ανάλογα το αυτοκίνητο επιλέγεται το καταλληλότερο πρωτόκολλο επικοινωνίας. Η αυτόματη επιλογή πρωτοκόλλου επικοινωνίας δίνεται μέσω της εντολής “ATSP0” η οποία αποστέλλεται από το πρόγραμμα μας στην OBD συσκευή.

Εφόσον έχουμε επιτύχει την σύνδεση με την OBD-II συσκευή και την αποστολή των πρώτων μηνυμάτων εντοπισμού πρωτοκόλλου επικοινωνίας έπρεπε να καταλάβουμε πως δεχόμαστε πληροφορίες από τον εγκέφαλο του αυτοκινήτου. Σαν πρώτη δοκιμή στείλαμε τον κωδικό του αισθητήρα στροφών του κινητήρα για να δούμε τι απάντηση θα έχουμε. Η επιστροφή από τον εγκέφαλο ήταν σε δεκαεξαδική μορφή όποτε ήθελε κάποια κωδικοποίηση για να ελέγξουμε την ορθότητα της τιμής. Με την βοήθεια εγχειριδίων και κάποιων βιβλιογραφιών (βλέπε Βιβλιογραφία) βρήκαμε την κωδικοποίηση που χρειάζεται η συγκεκριμένη επιστροφή ώστε να φανεί η πραγματική τιμή των στροφών του κινητήρα. Αφού εφαρμόστηκαν οι κωδικοποιήσεις και ελέγχθηκε η εγκυρότητα της απάντησης έπρεπε να βρούμε έναν τρόπο να γίνεται γρήγορη ανανέωση αυτής της τιμής ώστε να φαίνεται ότι αλλάζει σε πραγματικό χρόνο. Όπως αναφέραμε και παραπάνω χρησιμοποιήσαμε το αντικείμενο Timer αφού αποκλείσαμε οποιαδήποτε άλλη μέθοδο επανάληψης που βρήκαμε.

```
Timer vehicle_speed_timer = new Timer();
TimerTask vehicle_speed_task = new TimerTask() {
    @Override
    public void run() {
        pid_vehicle_speed();
    }
};
vehicle_speed_timer.scheduleAtFixedRate(vehicle_speed_task, delay, period_vehicle_speed);
```

Εικόνα 38: Η χρήση της Timer για συνεχόμενη επικοινωνία

Το πρόβλημα με την Timer ήταν να βρούμε την σωστή αλληλουχία καθυστέρησης και επανάληψης για να μπορούμε να στέλνουμε και να δεχόμαστε πολλαπλά αιτήματα και απαντήσεις αισθητήρων. Για την αντιμετώπιση αυτού δεν υπήρχε μια συγκεκριμένη λύση, έπρεπε να κάνουμε πολλές δοκιμές στέλνοντας πολλούς αισθητήρες ταυτόχρονα και έχοντας καταλήξει ποιοι αισθητήρες χρειάζονται μεγαλύτερη ταχύτητα ανανέωσης και ποιοι όχι, βρήκαμε τις κατάλληλες τιμές καθυστέρησης και επανάληψης.

Η λειτουργία DataStream αφού υλοποιήθηκε παρουσίασε μια αστάθεια στη λειτουργία της αφού μετά από ένα χρονικό περιθώριο παρουσίαζε απώλεια σύνδεσης. Το πρόβλημα αυτό διορθώθηκε κατά τη διαδικασία του Debugging αφού βελτιστοποιήθηκε ο Handler και ο κώδικας λήψης των μηνυμάτων. Το πρόβλημα παρουσιαζόταν όταν ζητούσαμε πληροφορία που δεν υποστηριζόταν από το αυτοκίνητο. Έτσι υλοποιήθηκε ένας έλεγχος ο οποίος μας επιστρέφει πρώτα τους διαθέσιμους αισθητήρες έτσι ώστε να μην μπορεί να δυσλειτουργήσει το πρόγραμμα λόγω ασυμβατότητας δεδομένων.

```
private final Handler mhandler = handleMessage(msg) → {
        switch (msg.what) {
            case MESSAGE_STATE_CHANGE:
                if (D) Log.i(TAG, msg: "MESSAGE_STATE_CHANGE: " + msg.arg1);
                switch (msg.arg1) {
                    case BtHelper.STATE_CONNECTED:
                        Log.d(TAG, msg: "STATE Connected");
                        connected = true;
                        initCommands();
                        break;
                    case BtHelper.STATE_CONNECTING:
                        Log.d(TAG, msg: "STATE Connecting");
                        break;
                    case BtHelper.STATE_NONE:
                        connected = false;
                        Log.d(TAG, msg: "STATE None");
                        break;
                }
                break;
```

Εικόνα 39: Απόσπασμα της συνάρτηση της Handler (a)

```
case MESSAGE_WRITE:
    Log.d(TAG, msg: "STATE writing");
    byte[] writeBuf = (byte[]) msg.obj;
    // construct a string from the buffer
    String writeMessage = new String(writeBuf);
    if (D) Log.d(TAG, msg: "written = '" + writeMessage + "'");
    break;
case MESSAGE_READ:
    Log.d(TAG, msg: "STATE reading");
    if (paused) {
        Log.d(TAG, msg: "STATE Paused in reading");
        break;
    }
    byte[] readBuf = (byte[]) msg.obj;
    // construct a string from the valid bytes in the buffer
    String readMessage = new String(readBuf, offset: 0, msg.arg1);
    if (D) Log.d(TAG, readMessage);

    calculations(readMessage);
    break;
```

Εικόνα 40: Απόσπασμα της συνάρτηση της Handler (b)

7. Μελλοντικοί Στόχοι

Στους μελλοντικούς στόχους του προγράμματος έχουμε την συμπλήρωση της λίστας των διαθέσιμων προς ανάγνωση αισθητήρων καθώς και την βελτιστοποίηση του ρυθμού αποστολής και λήψης πληροφορίας, έτσι ώστε να είναι δυνατή η παρακολούθηση περισσότερων αισθητήρων ταυτόχρονα χωρίς να υπάρχει μείωση στον ρυθμό ανανέωσης(refresh rate). Θα ήταν θεμιτό επίσης να προστεθούν και αισθητήρες οι οποίοι βρίσκονται περισσότερο στα ηλεκτροκίνητα αυτοκίνητα όπως Volt στα κελιά της μπαταρίας, θερμοκρασία μπαταρίας κ.α.

Στόχος μας ακόμα είναι η δημιουργία λειτουργιών οι οποίες θα μετράνε στοιχεία επιδόσεων του αυτοκινήτου όπως χρόνοι 0-100χλμ, 100-0, τελική ταχύτητα κ.α.
Τέλος στους στόχους μας είναι η υλοποίηση μιας νέας λειτουργίας μέσω της οποίας ο χρήστης μπορεί να αποθηκεύει και να μοιράζεται το ιστορικό σέρβις και επισκευών του αυτοκινήτου του καθώς και πιθανές μετατροπές τις οποίες έχει εγκαταστήσει.

Βιβλιογραφία

Ακολουθούν οι βιβλιογραφικές αναφορές (πηγές) της Εργασίας.

Word help - Getting started with Word 2016. (29 Αυγούστου 2016). Ανακτήθηκε από https://support.office.com/en-us/word

Wikipedia contributors. (2023, February 1). On-board diagnostics. In *Wikipedia, The Free Encyclopedia*. Retrieved 06:27, February 8, 2023, from https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&oldid=1136795977

Wikipedia contributors. (2023, February 6). OBD-II PIDs. In *Wikipedia, The Free Encyclopedia*. Retrieved 06:29, February 8, 2023, from https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&oldid=1137763001

Deitel, P., Deitel, H., & Deitel, A. (2014). *Android how to Program*. Prentice Hall Press.

Obd Auto Doctor, (2020). *Diagnostic Trouble Codes explained*. Obd Auto Doctor, Retrieved 09:20, February 8, 2023, https://www.obdautodoctor.com/tutorials/diagnostic-trouble-codes-explained/

Wikipedia contributors. (2022, November 25). ELM327. In *Wikipedia, The Free Encyclopedia*. Retrieved 06:40, February 8, 2023, from https://en.wikipedia.org/w/index.php?title=ELM327&oldid=1123676302

Nyvip, (2015). *Diagnostic trouble codes (DTCs)*. Nyvip, Retrieved 09:20, February 8, 2023, https://www.nyvip.org/PublicSite/OBDII/diagnostic-trouble-codes.html#:~:text=These

Obd Auto Doctor, (2020). *OBD2 freeze frame explained.* Obd Auto Doctor, Retrieved 09:20, February 8, 2023, https://www.obdautodoctor.com/tutorials/obd-freeze-frame-explained/

Tony Markovich, (2020). *What Is a Knock Sensor?.* The Drive, Retrieved 09:20, February 8, 2023, https://www.thedrive.com/cars-101/35148/what-is-a-knock-sensor

Autlog, (2020). *INTAKE MANIFOLD PRESSURE SENSOR.* Autlog, Retrieved 09:20, February 8, 2023, http://www.autlog-germany.de/en/products/sensors/intake-manifold-pressure-sensor/

ELM327, (2020). *OBD to RS232 Interpreter.* ELM327, Retrieved 09:20, February 8, 2023, https://www.elmelectronics.com/wp-content/uploads/2016/07/ELM327DS.pdf

SparkFun Electronics, (-). *Getting Started with OBD-II*, SparkFun Electronics, Retrieved 09:20, February 8, 2023, https://learn.sparkfun.com/tutorials/getting-started-with-obd-ii/obd-ii-protocols

CSS Electronics, (2022). *OBD2 Explained - A Simple Intro [2022]*, CSS Electronics, Retrieved 09:20, February 8, 2023, https://www.csselectronics.com/pages/obd2-explained-simple-intro

Android Developers, (2021). *Bluetooth overview, Android Developers,* Retrieved 09:20, February 8, 2023, *https://developer.android.com/guide/topics/connectivity/bluetooth*

Android Developers, (2023). *Meet Android Studio, Android Developers,* Retrieved 09:20, February 8, 2023, *https://developer.android.com/studio/intro*

TechTarget Contributor, (2023). *Android Studio*, Android Studio, Retrieved 09:20, February 8, 2023, https://www.techtarget.com/searchmobilecomputing/definition/Android-Studio#

Παράρτημα Α: Το Android Studio

Το Android Studio είναι το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για την ανάπτυξη εφαρμογών Android με βάση το Intellij IDEA. Έχει ισχυρά εργαλεία επεξεργασίας κώδικα και προγραμματιστών της Intellij. Εκτός από αυτά προσφέρει ακόμη περισσότερες δυνατότητες που ενισχύουν την παραγωγικότητα δημιουργίας μιας εφαρμογής Android.

Μία από αυτές είναι το ευέλικτο σύστημα κατασκευής που βασίζεται σε Gradle σύστημα. Το Gradle είναι ένα εργαλείο αυτοματισμού κατασκευής γνωστό για την ευελιξία του στην δημιουργία λογισμικού. Χρησιμοποιείται για την αυτοματοποίηση της δημιουργίας της εφαρμογής που περιλαμβάνει την μεταγλώττιση, τη σύνδεση και την συσκευασία του κώδικα. Ουσιαστικά κάνει την σύνθεση της εφαρμογής αυτόματα χωρίς να ασχοληθεί ο προγραμματιστής.

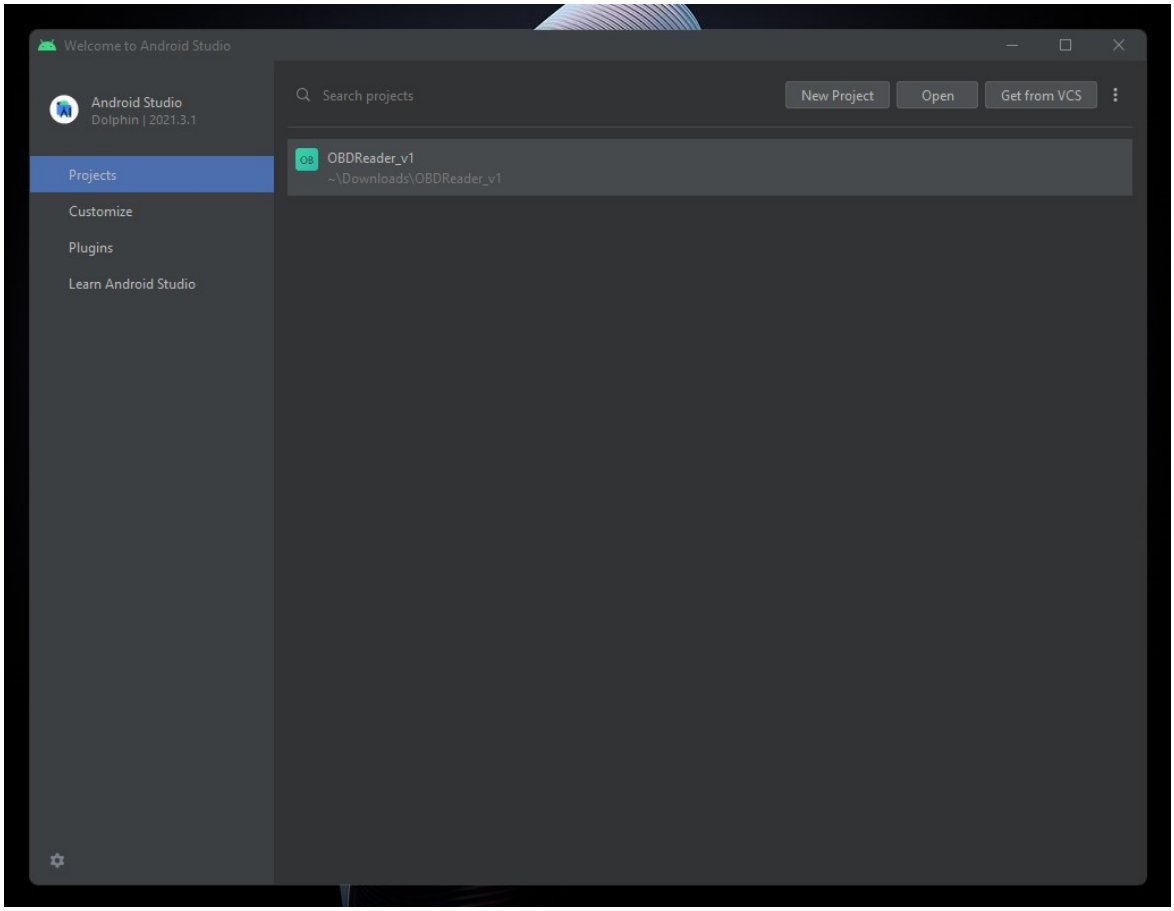
Έχει ένα ενοποιημένο περιβάλλον όπου μπορεί να γίνει ανάπτυξη εφαρμογής για όλες τις συσκευές Android. Παρέχει επίσης ένα γρήγορο εξομοιωτή για να εμφανίζεται σε πραγματικό χρόνο η εφαρμογή.

Δημιουργία Έργου

Για την δημιουργία έργου ακολουθούν τα εξής παρακάτω βήματα:

1. Άνοιγμα της εφαρμογής Android studio

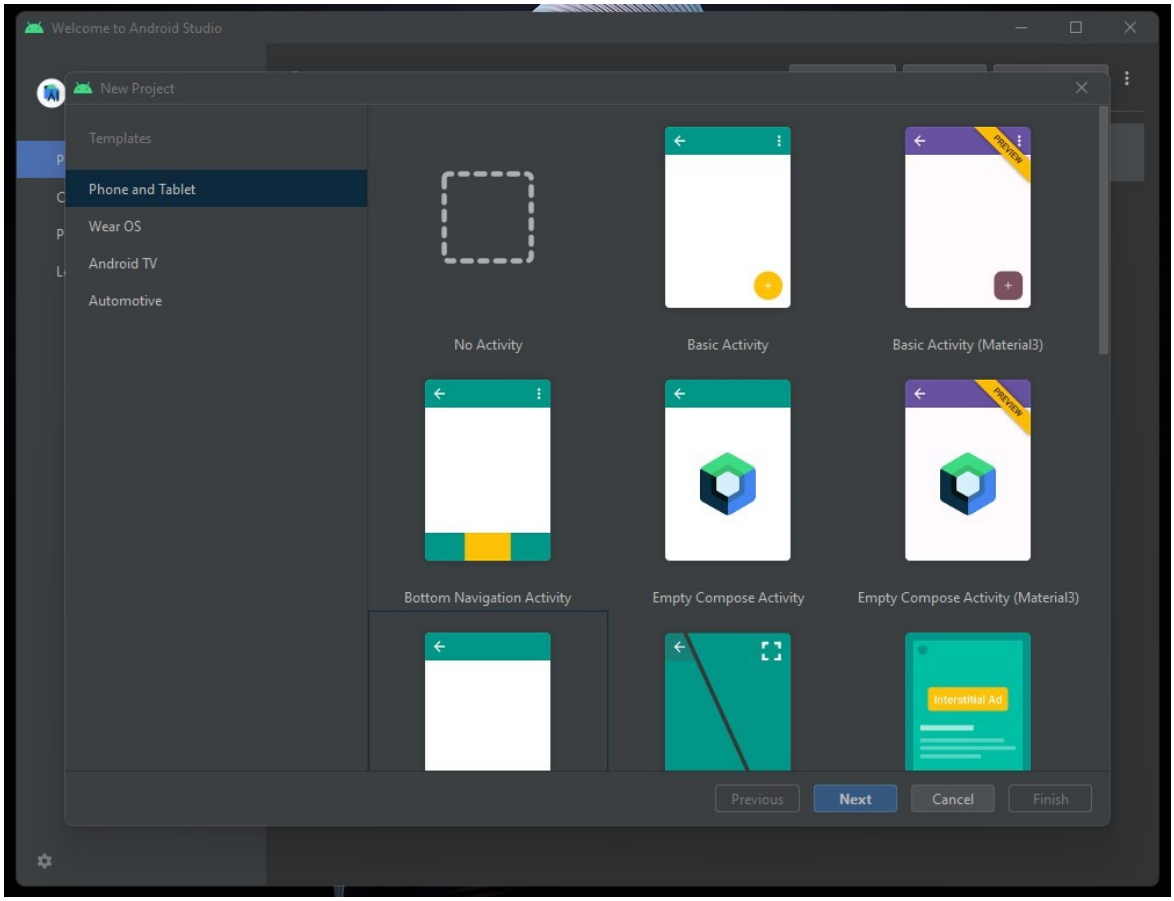
2. Όταν ανοίξει θα εμφανιστεί ένα παράθυρο όπως φαίνεται και στην παρακάτω εικόνα.



3. Σε αυτό το παράθυρο στο δεξί μέρος φαίνονται παλιότερα έργα που έχουν δημιουργηθεί.

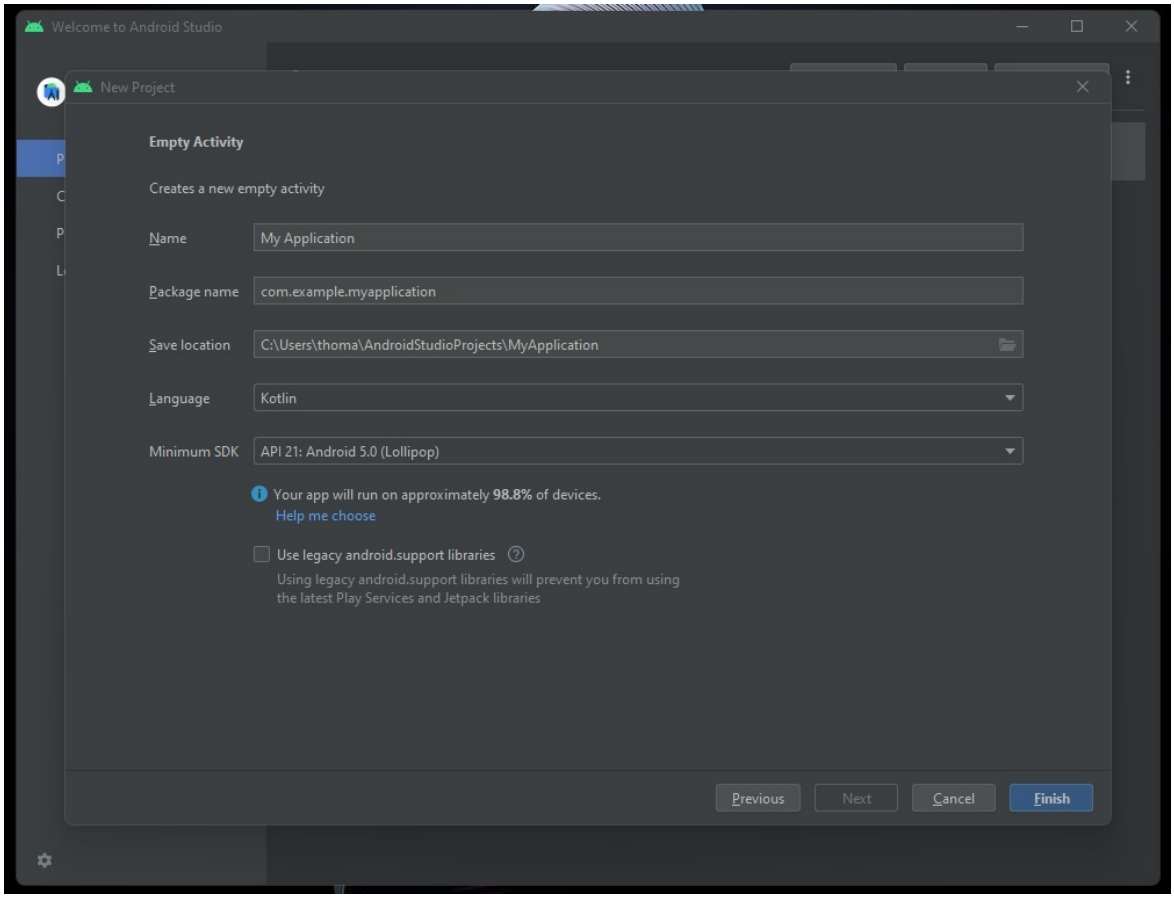
4. Για την δημιουργία νέου πατάμε το κουμπί που αναγράφει “New Project”.

5. Θα εμφανιστεί νέο παράθυρο για την επιλογή διάφορων προεπιλογών για την δημιουργία της εφαρμογής όπως την μορφή της αρχικής κλάσης.



6. Αφού γίνει η επιλεχθούν οι προεπιλογές πατάμε το κουμπί που αναγράφει “Next”.

7. Θα εμφανιστεί νέο παράθυρο που θα ζητάει να ονομαστεί η εφαρμογή. Το “Package Name” και το “Save Location” είναι προκαθορισμένα με το που γραφτεί η ονομασία της εφαρμογής αλλά μπορεί να αλλάξει σε περίπτωση που θέλει ο χρήστης. Στην συνέχεια δίνονται δύο επιλογές για την κύρια γλώσσα γραφής της εφαρμογής αυτές της Kotlin και της Java. Τέλος επιλέγουμε το “Minimum SDK” που σαν προεπιλογή έχει αυτό που καλύπτει το μεγαλύτερο ποσοστό κινητών Android. Αφού γίνουν τα παραπάνω βήματα πατάμε το κουμπί που αναγράφει “Finish” που βρίσκεται στο κάτω δεξί μέρος του παραθύρου ώστε να γίνει η πλήρης ολοκλήρωση της δημιουργίας του έργου.

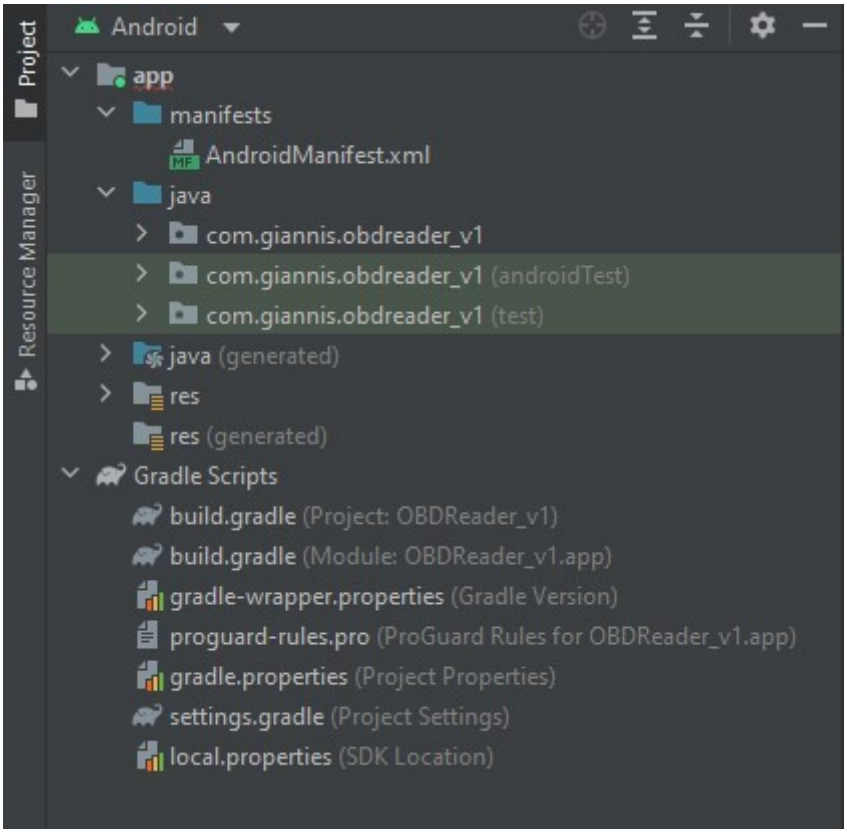


Δομή Έργου

Κάθε έργο στο Android Studio περιέχει λειτουργικές μονάδες με αρχεία πηγαίου κώδικα και αρχεία πόρων.

Οι τύποι ενοτήτων περιλαμβάνουν ενότητες εφαρμογών Android, Ενότητες βιβλιοθήκης και ενότητες Google App Engine.

Από προεπιλογή, το Android Studio εμφανίζει τα αρχεία του έργου στην προβολή έργου Android, όπως φαίνεται και στην παρακάτω εικόνα.



Αυτή η προεπιλογή δομείται ανά λειτουργικές μονάδες για να παρέχει γρήγορη πρόσβαση στα βασικά δομικά αρχεία του έργου.

Όλα τα αρχεία έκδοσης είναι ορατά στο ανώτατο επίπεδο κάτω από το Gradle Scripts και κάθε λειτουργική μονάδα εφαρμογής περιέχει τους ακόλουθους φακέλους:

manifests: Περιέχει το αρχείο AndroidManifest.xml που έχει βασικές πληροφορίες για τις άδειες όπως για παράδειγμα της τοποθεσίας και την δομή των κλάσεων.

Java: Περιέχει τα αρχεία πηγαίου κώδικα Java, συμπεριλαμβανομένου του κώδικα δοκιμής JUnit.

Res: Περιέχει όλους τους πόρους χωρίς κώδικα, όπως διατάξεις XML, συμβολοσειρές διεπαφής χρήστη και εικόνες bitmap.

Διεπαφή Χρήστη

Το κύριο παράθυρο του Android Studio αποτελείται από πολλές λογικές περιοχές όπου φαίνονται και επεξηγούνται παρακάτω.

1. Η γραμμή εργαλείων επιτρέπει την πραγματοποίηση ενός ευρέως φάσματος ενεργειών, όπως της εκτέλεσης της εφαρμογής καθώς και την εκκίνηση πολλών εργαλείων Android.

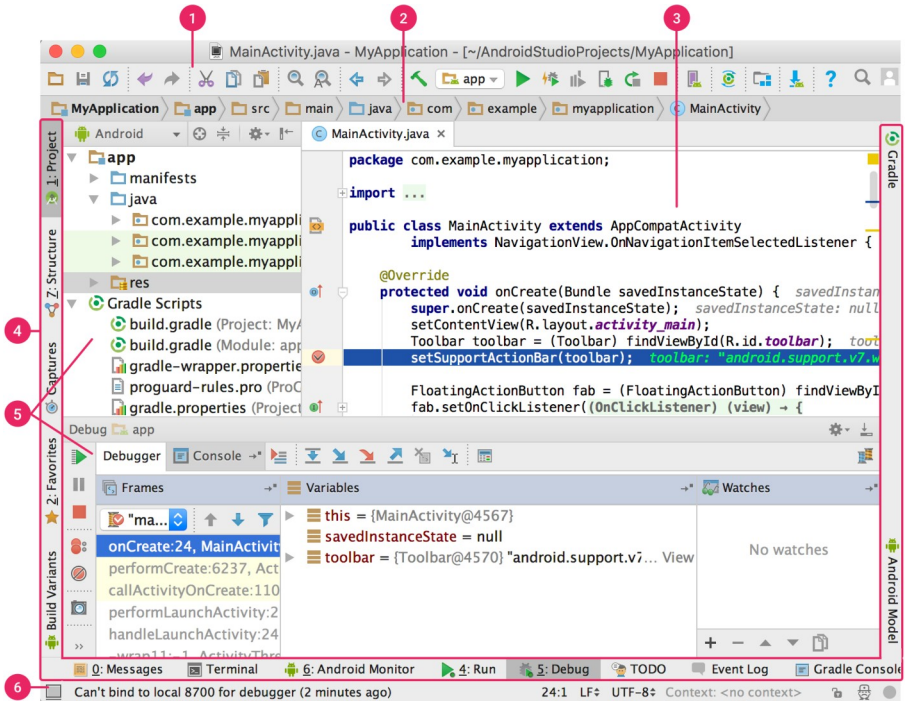
2. Η γραμμή πλοήγησης που βοηθά στην πλοήγηση του έργου καθώς και στο άνοιγμα των αρχείων για επεξεργασία.

3. Το παράθυρο του επεξεργαστή που γίνεται η γραφή του πηγαίου κώδικα. Ανάλογα με τον τρέχοντα τύπο αρχείου, ο επεξεργαστής μπορεί να αλλάξει.

4. Η γραμμή παραθύρου εργαλείων που είναι γύρω από το εξωτερικό του παραθύρου IDE και περιέχει κουμπιά που επιτρέπουν την ανάπτυξη και σύμπτυξη μεμονωμένα παράθυρα εργαλείων.

5. Τα παράθυρα εργαλείων που δίνουν πρόσβαση σε συγκεκριμένες εργασίες όπως διαχείριση έργου, αναζήτηση, έλεγχος έκδοσης κ.α.

6. Η γραμμή κατάστασης που εμφανίζει την κατάσταση του έργου και του ίδιου του IDE, καθώς και τυχόν προειδοποιήσεις ή μηνύματα.



shorturl.at/uACE9