ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΟΠΤΙΚΟΠΟΙΗΣΗ ΚΑΙ ΑΝΑΛΥΣΗ ΠΑΡΑΜΕΤΡΩΝ ΣΕ ΠΡΑΓΜΑΤΙΚΟ ΧΡΟΝΟ ΜΕΣΩ ΠΡΟΤΥΠΟΥ ΣΥΣΤΗΜΑΤΟΣ ΔΙΑΓΝΩΣΗΣ OBD (On board diagnostic)

Διπλωματική Εργασία

του

Ευθύμιου Καραμπίλη

Θεσσαλονίκη, Ιανουάριος 2024

ΟΠΤΙΚΟΠΟΙΗΣΗ ΚΑΙ ΑΝΑΛΥΣΗ ΠΑΡΑΜΕΤΡΩΝ ΣΕ ΠΡΑΓΜΑΤΙΚΟ ΧΡΟΝΟ
ΜΕΣΩ ΠΡΟΤΥΠΟΥ ΣΥΣΤΗΜΑΤΟΣ
ΔΙΑΓΝΩΣΗΣ OBD (On board diagnostic)

Ευθύμιος Καραμπίλης
Πτυχίο Τμήματος Μηχανολόγων Μηχανικών Τ.Ε, ΔΙ.ΠΑ.Ε, 2019

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ
ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων καθηγητής
Κασκάλης Θεόδωρος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ηη/μμ/εεεε

Ψάννης Κωνσταντίνος	Ξυνόγαλος Στυλιανός	Κασκάλης Θεόδωρος
.....	.....	.....

Ευθύμιος καραμπίλης

.....

Περίληψη

Η εργασία αυτή προσεγγίζει τον τομέα των διαγνωστικών οχημάτων, ηλεκτρονικών συστημάτων και ανάλυσης δεδομένων, μέσα από το πρίσμα ενός καινοτόμου συστήματος. Η μελέτη ξεκινά με μια εισαγωγή που τονίζει τη σημασία του προβλήματος, τους στόχους και τη δομή αυτού. Το θεωρητικό υπόβαθρο εξετάζει το ιστορικό πλαίσιο της διάγνωσης προβλημάτων στα οχήματα, τα πρωτοκόλλα διάγνωσης και τη μελλοντική εξέλιξη των διαγνωστικών μηχανημάτων. Παράλληλα αναλύει το ηλεκτρονικό σύστημα διαχείρισης κινητήρα και την αναγνώριση παραμέτρων που προκύπτουν από τη διεπαφή του συστήματος αυτού με τη διαγνωστική συσκευή.

Στη συνέχεια εστιάζει στην ανάλυση της πλακέτας Arduino, περιγράφοντας τα πλεονεκτήματά, τα εξαρτήματά και τον προγραμματισμό μέσω του περιβάλλοντος ανάπτυξης Arduino IDE. Ο τεχνικός εξοπλισμός του συστήματος αναλύεται λεπτομερώς, παρουσιάζοντας το Arduino Mega 2560 R3, τον προσαρμογέα OBD με ενσωματωμένο αισθητήρα MEMS, την οθόνη TFT LCD και τον αισθητήρα GPS.

Η εργασία προχωρά σε ανάλυση των διαδικασιών για την κατασκευή και τον σχεδιασμό του συστήματος αποθήκευσης και οπτικοποίησης δεδομένων, καθώς και τη χρήση εκτύπωσης 3D για τη δημιουργία περιβλήματος για την κατασκευή. Η ενότητα ανάπτυξης λογισμικού παρουσιάζει το "CarAnalysis", διευκρινίζοντας τους στόχους, τις βασικές λειτουργίες, τη γλώσσα προγραμματισμού και τις βιβλιοθήκες.

Στην εν λόγω μελέτη περιέχονται σχολιασμένοι κώδικες για την κατασκευή με Arduino και το πρόγραμμα CarAnalysis. Τέλος, ολοκληρώνεται επισημαίνοντας τα όρια της έρευνας και προτείνοντας μελλοντικές επεκτάσεις για αυτή, στο πλαίσιο των διαγνωστικών οχημάτων και της ανάλυσης δεδομένων.

Λέξεις Κλειδιά: σύστημα διάγνωσης, Arduino, OBD, Python, PIDs, data analysis, data visualisation, on board diagnostic

Abstract

This paper explores the field of vehicle diagnostics, electronic systems and data analysis through the lens of an innovative system. The study begins with an introduction that emphasizes the importance of the problem, objectives, and the structure. The theoretical background examines the historical context of vehicle problem diagnosis, diagnosis protocols and the future development of diagnostic machinery. At the same time, it analyzes the electronic engine management system and the recognition of parameters resulting from the interface of this system with the diagnostic device.

The focus then moves to Arduino, describing its definition, advantages, components and programming through Arduino IDE development environment. The technical equipment of the system is analyzed in detail, introducing the Arduino Mega 2560 R3, the OBD adapter with built-in MEMS sensor, the TFT LCD display and the GPS sensor.

This paper proceeds with an analysis of the procedures followed for the construction and design of both the data storage and visualization system, as well as the use of 3D printing to create an enclosure for the construction. The software development section introduces "CarAnalysis", clarifying its goals, core functions, programming language and libraries.

Accompanying the document are annotated codes for building with Arduino and the CarAnalysis program. The study concludes highlighting the research limitations and suggesting future extensions for this research in the context of vehicle diagnostics and data analytics.

Keywords: diagnosis system, Arduino, OBD, Python, PIDs, data analysis, data visualization

Περιεχόμενα

Κατάλογος Εικόνων

Συμβολισμοί

1. OBD: On-Board Diagnostics
2. IOT: Internet of Things
3. PID: Parameter IDs
4. EMS: Engine Management System
5. MEMS: Microelectromechanical Systems
6. PWM: Pulse Width Modulation
7. GPS: Global Positioning System
8. TFT: Thin-Film Transistor
9. LCD: Liquid Crystal Display
10. GMT: Greenwich Mean Time
11. VCC: Voltage Common Collector (or Power Supply Voltage)
12. GND: Ground
13. I2C: Inter-Integrated Circuit
14. DC: Direct Current
15. SD: Secure Digital (memory card)
16. NMEA: National Marine Electronics Association
17. CSV: Comma-Separated Values
18. PLA: Polylactic Acid
19. CAD: Computer-Aided Design
20. EXE: Executable (file)

1 Εισαγωγή

1.1 Πρόβλημα – Σημαντικότητα του θέματος

Το πρόβλημα που αντιμετωπίζουν πολλοί οδηγοί είναι η έλλειψη πρόσβασης σε έγκυρες και άμεσες πληροφορίες σχετικά με την κατάσταση και τη λειτουργία των οχημάτων τους. Συχνά, η ανίχνευση προβλημάτων στα οχήματα γίνεται κατά τη στιγμή που τα προβλήματα έχουν ήδη εξελιχθεί σε σοβαρές βλάβες. Η έλλειψη άμεσης πληροφόρησης σχετικά με την κατάσταση του κινητήρα, των συστημάτων ασφαλείας και άλλων κρίσιμων παραμέτρων εμποδίζει τους οδηγούς από το να προλαμβάνουν πιθανά προβλήματα και να λαμβάνουν έγκαιρα μέτρα συντήρησης. Αυτό το έλλειμμα ενημέρωσης επιφέρει ανασφάλεια, υψηλότερο κόστος επισκευών, καθώς και απώλεια ευκαιριών για οικονομία καυσίμου και βελτίωση της γενικής απόδοσης των οχημάτων. Η ανάπτυξη ενός συστήματος που προσφέρει ολοκληρωμένη και άμεση ανάλυση των παραμέτρων OBD σε πραγματικό χρόνο αντιμετωπίζει αποτελεσματικά αυτό το πρόβλημα, προσφέροντας στους οδηγούς ενημερωμένη επισκόπηση και δυνατότητα δράσης για τη συντήρηση και βελτίωση των οχημάτων τους.

1.2 Σκοπός – Στόχοι

Η παρούσα εργασία έχει ως κύριο στόχο την ανάλυση και οπτικοποίηση δεδομένων σε πραγματικό χρόνο για οποιοδήποτε όχημα, χρησιμοποιώντας το σύστημα διαγνωστικού ελέγχου OBD με τη χρήση μικροελεγκτή μονής πλακέτας Arduino. Το εν λόγω σύστημα είναι σε θέση να μεταφέρει και να αποθηκεύει απαραίτητες πληροφορίες και δεδομένα, τα οποία ελέγχονται από τον ηλεκτρονικό εγκέφαλο του κινητήρα ενός οχήματος, στον τελικό χρήστη. Τα δεδομένα αυτά είναι προσβάσιμα και αναλύσιμα από οποιονδήποτε χρήστη, παρέχοντας πληροφορίες σχετικά με πιθανά προβλήματα κακής λειτουργίας σε πρώιμο στάδιο. Η έγκαιρη διάγνωση αυτών των προβλημάτων μπορεί να συμβάλει στην αποφυγή σοβαρότερων βλαβών στον κινητήρα.

Επιπλέον, η ανάλυση αυτών των δεδομένων θα συνεισφέρει στη βελτίωση της οδηγικής συμπεριφοράς του χρήστη, επιτυγχάνοντας την επιθυμητή μείωση στην κατανάλωση καυσίμου και την καταπόνηση του οχήματος. Το προτεινόμενο σύστημα δίνει τη δυνατότητα στους χρήστες να είναι ενήμεροι για την επίδοση του οχήματός τους, προσφέροντας ταυτόχρονα ένα αποτελεσματικό εργαλείο για τη διαχείριση και την πρόληψη προβλημάτων.

1.3 Διάρθρωση της μελέτης

Η εργασία αρχικά, παρέχει ένα θεωρητικό υπόβαθρο που αναδεικνύει τη σημασία της διάγνωσης στον τομέα των οχημάτων, εξετάζοντας τα οχήματα, τις μονάδες ελέγχου, το ηλεκτρονικό σύστημα διαχείρισης κινητήρα, καθώς και το Internet of Things στην αυτοκινητοβιομηχανία. Στη συνέχεια, επικεντρώνεται στις πλακέτες ARDUINO, εξηγώντας τι είναι και ποια είναι τα πλεονεκτήματα της χρήσης της πλακέτας ARDUINO. Η βασική μελέτη επικεντρώνεται στη δημιουργία μιας πλακέτας ARDUINO που συνδέεται με ένα όχημα, αποθηκεύοντας δεδομένα ενώ παράλληλα αυτά τα δεδομένα οπτικοποιούνται σε πραγματικό χρόνο σε μια οθόνη. Τέλος, γίνεται ανάλυση των log data μέσω ενός προγράμματος που έχει υλοποιηθεί σε Python, προσφέροντας μια ολοκληρωμένη προσέγγιση για τη διαδικασία διάγνωσης και παρατήρησης προβλημάτων σε πρώιμο στάδιο στον τομέα της αυτοκινητοβιομηχανίας, με χρήση της τεχνολογίας ARDUINO και Python.

2 Θεωρητικό Υπόβαθρο

2.1 Οχήματα και διάγνωση

Τα οχήματα αποτελούν ένα διαδεδομένο μέσο μεταφοράς παγκοσμίως, σχεδιασμένο για να διευκολύνει τη μετακίνησή μας από ένα σημείο σε ένα άλλο. Μπορούν να κατηγοριοποιηθούν σε διάφορους τύπους, όπως αυτοκίνητα, φορτηγά, μοτοσικλέτες, λεωφορεία, τρένα, αεροπλάνα, και πολλά άλλα. Η διάγνωση αναφέρεται στη διαδικασία εντοπισμού προβλημάτων και βλαβών στα οχήματα. Η τεχνολογία διάγνωσης έχει εξελιχθεί σημαντικά και χρησιμοποιείται εκτενώς από τους μηχανικούς και τους τεχνικούς στον χώρο των οχημάτων. Με τη χρήση εργαλείων διάγνωσης, όπως συστήματα διάγνωσης βασισμένα σε υπολογιστές, συνδεδεμένα στον υπολογιστή του οχήματος, διαβάζουν τα δεδομένα από αισθητήρες και μηχανισμούς.

Η διάγνωση ανιχνεύει προβλήματα σε διάφορα υποσυστήματα του οχήματος, όπως το σύστημα κινητήρα, το σύστημα φρένων, το σύστημα ψύξης, το σύστημα εξάτμισης, κ.ά. Αυτή η πληροφορία χρησιμοποιείται για την επισκευή προβλημάτων, τη συντήρηση του οχήματος και τη βελτίωση της απόδοσης και της ασφάλειας του. Συνολικά, η διάγνωση αποτελεί σημαντικό εργαλείο για την επισκευή και τη συντήρηση των οχημάτων, βοηθώντας στην αποτελεσματική αντιμετώπιση προβλημάτων, τη βελτίωση της ασφάλειας, και την ενίσχυση της απόδοσης των οχημάτων

2.2 Ιστορική αναδρομή διάγνωσης οχημάτων

Στα πρώτα χρόνια της αυτοκινητοβιομηχανίας, οι μηχανικοί επισκευής βασίζονταν κυρίως στην παρατήρηση και τη μηχανική εμπειρία για τον εντοπισμό βλαβών. Οι πρώτες μορφές διάγνωσης περιλάμβαναν απλούς μηχανικούς μετρητές για την ανίχνευση βασικών προβλημάτων όπως η πίεση λαδιού, η θερμοκρασία κινητήρα, και η φόρτιση της μπαταρίας. Κατά τη δεκαετία του '60, με την εισαγωγή ηλεκτρονικών συστημάτων στα αυτοκίνητα, όπως τα συστήματα ψεκασμού καυσίμου, αυξήθηκε η ανάγκη για πιο προηγμένα μέσα διάγνωσης. Αρχικά, χρησιμοποιήθηκαν ειδικά εργαλεία για τη σύνδεση με τα ηλεκτρονικά συστήματα των οχημάτων, αλλά η διαδικασία ήταν περίπλοκη και χρονοβόρα. Κατά τη δεκαετία του '80, εισήχθη το OBD-I, που περιλάμβανε έναν περιορισμένο αριθμό αισθητήρων και ενσωματωμένα διαγνωστικά συστήματα. Αυτό επέτρεπε στους τεχνικούς να ελέγχουν τα βασικά υποσυστήματα του αυτοκινήτου για προβλήματα με τη χρήση ειδικών εργαλείων και αναγνώσεων κωδικών.

Το OBD-II έκανε την εμφάνισή του στις αρχές της δεκαετίας του '90 και έγινε υποχρεωτικό για όλα τα νέα αυτοκίνητα που πωλούνται στις Ηνωμένες Πολιτείες από το 1996 και στη συνέχεια σε άλλες περιοχές του κόσμου. Το OBD-II παρέχει πολύ περισσότερες πληροφορίες και δυνατότητες διάγνωσης, με τη χρήση τυποποιημένων κωδικών που αναγγέλλουν προβλήματα σε διάφορα υποσυστήματα του οχήματος. Σήμερα, το OBD-II είναι ευρέως χρησιμοποιούμενο σε αυτοκίνητα και παρέχει πληροφορίες για πολλά υποσυστήματα όπως κινητήρας, φρένα, εκπομπές και άλλα.

Εικόνα 1: Πρώτο διαγνωστικό εργαλείο

(πηγή:https://forums.aaca.org/topic/340920-using-modern-diagnostic-tools-with-vintage-cars/

2.3 Πρωτόκολλα διάγνωσης

Η διάγνωση οχημάτων χρησιμοποιεί διάφορα πρωτόκολλα για την επικοινωνία με τα ηλεκτρονικά συστήματα των οχημάτων. Κατά τη διάρκεια της εξέλιξης, διακρίνονται δύο κύριες γενιές πρωτοκόλλων διάγνωσης, το OBD-I και το OBD-II, ενώ υπάρχουν και άλλα πρωτόκολλα που χρησιμοποιούνται για ειδικούς τομείς.

1. OBD-I (On-Board Diagnostics I): Αυτό το πρωτόκολλο εισήχθη περίπου στα τέλη της δεκαετίας του '80 και χρησιμοποιήθηκε κυρίως στα πρώτα ηλεκτρονικά συστήματα διάγνωσης. Οι κατασκευαστές χρησιμοποίησαν διάφορα πρωτόκολλα και συστήματα, καθιστώντας δύσκολο τον ομαλό και ομοιογενή έλεγχο. Δεν υπάρχει τυποποιημένο πρωτόκολλο OBD-I, αλλά πολλοί κατασκευαστές χρησιμοποίησαν δικά τους πρωτόκολλα και κωδικούς.

•SAE J1850: Ένα OBD-I πρωτόκολλο που χρησιμοποιήθηκε κυρίως σε οχήματα της Ford και της General Motors στη Βόρεια Αμερική.

2. OBD-II (On-Board Diagnostics II): Εισήχθη στις αρχές της δεκαετίας του '90 και αποτελεί το πιο διαδεδομένο και τυποποιημένο πρωτόκολλο διάγνωσης στη σύγχρονη αυτοκινητοβιομηχανία. Χρησιμοποιεί στάνταρ κωδικούς για την αναφορά προβλημάτων (Diagnostic Trouble Codes - DTCs) και έχει θεσπίσει τυποποιημένους συνδέσμους (OBD-II ports) για ευκολότερη πρόσβαση.

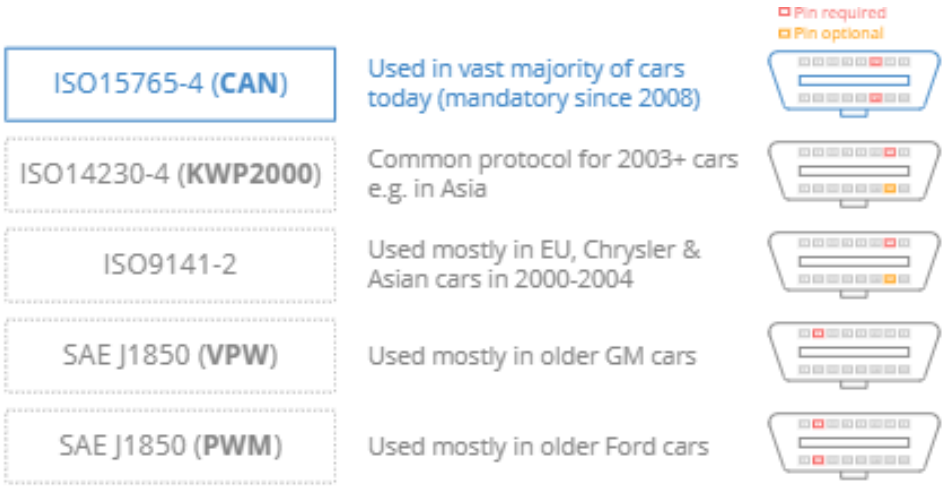
•ISO 15765-4 (CAN - Controller Area Network): Είναι το πρωτόκολλο που χρησιμοποιείται για τη διάγνωση σε πολλά σύγχρονα οχήματα. Υποστηρίζει ταχύτητες μεταφοράς δεδομένων έως και 1 Mbps και παρέχει αξιόπιστη επικοινωνία μεταξύ των διαφόρων ελεγκτών του οχήματος.

•Keyword Protocol 2000 (KWP2000): Χρησιμοποιείται κυρίως στα ευρωπαϊκά οχήματα ως ένα OBD-II πρωτόκολλο. Συχνά χρησιμοποιείται με το ISO 9141.

•UDS (Unified Diagnostic Services): Είναι ένα πρωτόκολλο που χρησιμοποιείται συχνά στα σύγχρονα οχήματα, ιδιαίτερα στα ευρωπαϊκά. Είναι βασισμένο στο ISO 14229 και παρέχει προχωρημένες δυνατότητες διάγνωσης και ελέγχου.

•ISO 22900 (MVCI, MCD-3D): Χρησιμοποιείται για την επικοινωνία μεταξύ εφαρμογών διάγνωσης και ελεγκτών οχημάτων.

Αυτά τα πρωτόκολλα καλύπτουν ένα ευρύ φάσμα οχημάτων και παρέχουν τη δυνατότητα διάγνωσης, επισκευής και συντήρησης με σύγχρονα και αποτελεσματικά μέσα.



Εικόνα 2: OBD πρωτόκολλα

(πηγή: https://www.csselectronics.com/pages/obd2-explained-simple-intro)

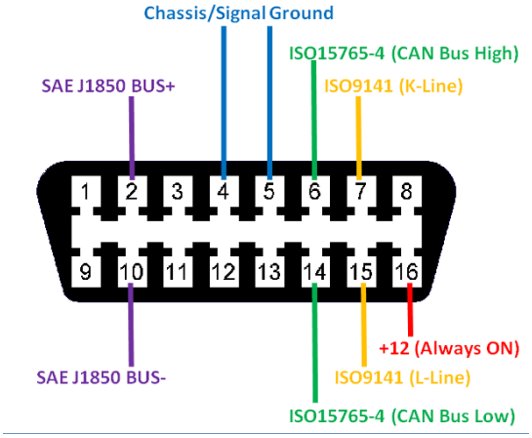
2.4 Θύρα (On-Board-Diagnostics) – διασύνδεση

Η θύρα OBD (On-Board Diagnostics) είναι μια διεπαφή που βρίσκεται σε όλα τα σύγχρονα οχήματα μετά το 1996 και χρησιμοποιείται για την επικοινωνία του οχήματος με την διαγνωστική συσκευή. Η θύρα αυτή παρέχει πρόσβαση σε διάφορες πληροφορίες και δεδομένα που σχετίζονται με την κατάσταση και τη λειτουργία του οχήματος. Η θύρα OBD συνήθως βρίσκεται κάτω από το ταμπλό του οχήματος και μπορεί να χρησιμοποιηθεί για διάφορες εφαρμογές, όπως την διάγνωση προβλημάτων, των έλεγχο των εκπομπών και τη παρακολούθηση και αναβάθμιση της απόδοσης του οχήματος.

Εικόνα 3:Θύρα διασύνδεσης OBD

(πηγη: https://www.geotab.com/blog/obd-ii/)

Ανάλογα την χρονολογία και το πολυπλεκτικό δίκτυο του κάθε αυτοκινήτου, αλλάζουν και οι διεπαφές εσωτερικά της θύρας obd. Βασικές επαφές είναι διαρκώς το σώμα (-) και το ρεύμα (+) που πάντα είναι στην ίδια θέση, όμως η επαφή επικοινωνίας με το όχημα και τα επιμέρους συστήματα αλλάζει ανάλογα με το πρωτόκολλο επικοινωνίας.



Εικόνα 4:Επαφες θύρας OBD

(πηγή: https://components101.com/connectors/obd2)

2.5 Μονάδες ελέγχου αυτοκίνητου

Οι μονάδες ελέγχου σε ένα όχημα είναι ηλεκτρονικές πλακέτες που διαχειρίζονται και ελέγχουν διάφορες λειτουργίες και συστήματα του αυτοκινήτου. Κάθε μονάδα είναι υπεύθυνη για συγκεκριμένο τομέα λειτουργίας, και η συνεργασία τους επιτρέπει τη σωστή και αποτελεσματική λειτουργία του αυτοκινήτου. Ορισμένες από τις βασικές μονάδες ελέγχου είναι :

- **Engine Control Unit (ECU):** Ελέγχει τη λειτουργία του κινητήρα, ρυθμίζοντας παραμέτρους όπως την ποσότητα του καυσίμου, τον χρονισμό της ανάφλεξης και άλλες παραμέτρους για βέλτιστη απόδοση του κινητήρα και την μείωση εκπομπής ρύπων .
- **Transmission Control Module (TCM):** Ελέγχει το κιβώτιο ταχυτήτων, ρυθμίζοντας τις αλλαγές των ταχυτήτων για ομαλή κίνηση και οικονομία καυσίμου.
- **Anti-lock Braking System (ABS) Module:** Ελέγχει το σύστημα ABS για την αποτροπή του κλειδώματος των τροχών κατά το φρενάρισμα .
- **Airbag Control Module (ACM):** Ελέγχει το συστήμα των αερόσακων, ενεργοποιώντας τους κατά την σύγκρουση.
- **Powertrain Control Module (PCM):** Ελέγχει τον κινητήρα και το κιβώτιο ταχυτήτων σε ορισμένα οχήματα.
- **Climate Control Module:** Ελέγχει το σύστημα κλιματισμού και θέρμανσης.
- **Body Control Module (BCM):** Ελέγχει διάφορες λειτουργίες του κορμού του αυτοκινήτου, όπως φώτα, κεντρικό κλείδωμα , παράθυρα.

Αυτές οι μονάδες επιτρέπουν τη συνολική ελεγχόμενη λειτουργία του αυτοκινήτου, παρέχοντας την μέγιστη απόδοση, ασφάλεια και άνεση στον οδηγό και τους επιβάτες.

2.6 Ηλεκτρονικό σύστημα διαχείρισης κινητήρα (EMS)

Το Ηλεκτρονικό Σύστημα Διαχείρισης Κινητήρα (Engine Management System - EMS) αποτελείται από διάφορους αισθητήρες που συλλέγουν δεδομένα και παρακολουθούν την λειτουργία του κινητήρα. Αυτοί οι αισθητήρες είναι κρίσιμοι για τη βέλτιστη απόδοση, αποτελεσματική καύση, καθώς και για τον έλεγχο των εκπομπών και τη γενική λειτουργία του οχήματος. Ορισμένοι από τους σημαντικότερους αισθητήρες είναι :

- **Αισθητήρας Θερμοκρασίας Κινητήρα (Engine Coolant Temperature Sensor):** Μετρά τη θερμοκρασία του ψυγείου του κινητήρα και βοηθά στην προσαρμογή του μίγματος καυσίμου-αέρα για βέλτιστη απόδοση και εκπομπές.

- **Αισθητήρας Μάζας Αέρα (Mass Airflow Sensor - MAF):** Μετρά την ποσότητα του αέρα που εισέρχεται στον κινητήρα, βοηθώντας στον υπολογισμό της ποσότητας του καυσίμου που πρέπει να ψεκαστεί.

- **Αισθητήρας Θερμοκρασίας Αέρα (Intake Air Temperature Sensor - IAT):** Μετρά τη θερμοκρασία του αέρα που εισέρχεται στον κινητήρα, παρέχοντας πληροφορίες για την πυκνότητα του αέρα ώστε να υπολογιστεί η ποσότητα καυσίμου.

- **Αισθητήρας Πεταλούδας (Throttle Position Sensor - TPS):** Μετρά τη θέση του πεντάλ γκαζιού

- **Αισθητήρας απόλυτης πίεσης (Manifold Absolute Pressure Sensor - MAP):** Μετρά την πίεση του αέρα στην ατμόσφαιρα .

- **Αισθητήρας Οξυγόνου Καυσαερίων (Oxygen Sensor - O2 Sensor):** Αυτός ο αισθητήρας τοποθετείται συνήθως στο σύστημα εξάτμισης, πριν από τον καταλύτη, και μετρά την ποσότητα οξυγόνου στα καυσαέρια. Ανιχνεύει εάν ο κινητήρας εκπέμπει φτωχό ή πλούσιο μείγμα καυσίμου-αέρα. Οι πληροφορίες που παρέχει χρησιμοποιούνται για τον έλεγχο της αναλογίας καυσίμου και αέρα που ψεκάζεται στον κινητήρα, βελτιώνοντας έτσι την απόδοση και μειώνοντας τις εκπομπές.

- **Αισθητήρας Κρουστικής Καύσης (Knock Sensor):** Είναι ένας αισθητήρας που τοποθετείται πάνω στο κορμό του κινητήρα και όταν κατά την καύση παρατηρηθεί ανωμαλία στέλνει την πληροφορία στο σύστημα διαχείρισης του κινητήρα (ECU) .

Αυτοί οι αισθητήρες συνεργάζονται για να παρέχουν τις κατάλληλες πληροφορίες στο Ηλεκτρονικό Σύστημα Διαχείρισης Κινητήρα(ECU) , επιτρέποντάς του να προσαρμόζει ακαριαία τις παραμέτρους λειτουργίας για τις διάφορες συνθήκες .

Εικόνα 5:Αισθητήρες διαχείρισης κινητήρα

(πηγή: https://www.theengineerspost.com/types-of-car-sensors/?utm_content=cmp-true)

2.7 Ταυτοποίηση παραμέτρων (PID)

Οι παράμετροι που επιθυμούμε να οπτικοποιήσουμε ορίζονται μέσω του πρωτοκόλλου OBD-II ως PIDs. Κάθε PID αντιστοιχεί σε μια παράμετρο δεδομένων του αυτοκινήτου που διατίθεται προς ανάγνωση. Τα περισσότερα πρωτόκολλα έχουν ορισμένα κοινά PIDs τα οποία χρησιμοποιούνται από όλους τους κατασκευαστές για να πληρείται το πρότυπο ISO. Όμως οι περισσότεροι μεγάλοι κατασκευαστές οχημάτων χρησιμοποιούν κρυφά PIDs για την ανάλυση κάποιον δεδομένων που δεν περιλαμβάνονται σε κάποιο κοινό πρότυπο, τα οποία δεν δημοσιεύουν. Οι παράμετροι αυτοί (PIDs) έχουν την παρακάτω μορφή :

- PID 010C - Engine RPM (Στροφές κινητήρα ανά λεπτό):

Περιγραφή: Αναφέρει τις στροφές του κινητήρα ανά λεπτό (RPM).
Παράδειγμα: "010C"

- PID 0110 - Throttle Position (Θέση πεντάλ γκαζιού):

Περιγραφή: Παρέχει τη θέση του πεντάλ γκαζιού.
Παράδειγμα: "0110"

- PID 0105 - Coolant Temperature (Θερμοκρασία ψυγείου κινητήρα):

Περιγραφή: Αναφέρει τη θερμοκρασία του ψυγείου του κινητήρα.
Παράδειγμα: "0105"

- PID 011F - Fuel Rail Pressure (Πίεση καυσίμου στον σωλήνα τροφοδοσίας):

Περιγραφή: Παρέχει την πίεση του καυσίμου στον σωλήνα τροφοδοσίας.
Παράδειγμα: "011F"

- PID 010D - Speed (Ταχύτητα οχήματος):

Περιγραφή: Αναφέρει την ταχύτητα του οχήματος.
Παράδειγμα: "010D"

2.8 Internet of things στον τομέα της αυτοκινητοβιομηχανίας και της διάγνωσης

Η ενσωμάτωση του Διαδικτύου των Πραγμάτων (IoT) στην αυτοκινητοβιομηχανία έχει επιφέρει επαναστατικές αλλαγές, επηρεάζοντας τον τρόπο που λειτουργούν, συντηρούνται και αλληλοεπιδρούν τα οχήματα. Η χρήση του IoT στον τομέα της αυτοκινητοβιομηχανίας επιφέρει αυξημένη συνδεσιμότητα, εξυπνότερες λειτουργίες και βελτιωμένη εμπειρία για τους οδηγούς. Ένα από τα κυριότερα πεδία όπου το IoT έχει επιφέρει αλλαγές είναι η συντήρηση των οχημάτων. Μέσω ενσωματωμένων αισθητήρων σε διάφορα σημεία των αυτοκινήτων, συλλέγονται συνεχώς δεδομένα για την κατάσταση των μηχανικών και ηλεκτρονικών συστημάτων. Αυτά τα δεδομένα χρησιμοποιούνται για προληπτική συντήρηση, με το σύστημα να μπορεί να προβλέπει πιθανές βλάβες και να ειδοποιεί τον οδηγό ή το εξειδικευμένο προσωπικό πριν από τον πραγματικό προβληματικό σφάλμα.

Στον τομέα της ασφάλειας, το IoT διαδραματίζει κεντρικό ρόλο. Συστήματα όπως οι αισθητήρες πρόληψης σύγκρουσης, οι κάμερες και τα συστήματα ειδοποίησης συνεργάζονται για να αυξήσουν την ασφάλεια των επιβατών και των πεζών. Επιπλέον, το IoT επιτρέπει στα οχήματα να επικοινωνούν μεταξύ τους για την αντιμετώπιση κινδύνων και την αποφυγή ατυχημάτων. Στον τομέα της ψυχαγωγίας και της εμπειρίας οδήγησης, το IoT προσφέρει συνδεσιμότητα με διάφορες ψυχαγωγικές πλατφόρμες, ενώ παράλληλα επιτρέπει την εξατομίκευση των ρυθμίσεων του οχήματος. Οι οδηγοί μπορούν να έχουν πρόσβαση σε μουσική, πλοήγηση και άλλες ψυχαγωγικές υπηρεσίες, ενώ τα εξελιγμένα συστήματα πλοήγησης βασίζονται σε δεδομένα πραγματικού χρόνου για βέλτιστη κυκλοφοριακή ροή. Συνολικά, η ενσωμάτωση του IoT στην αυτοκινητοβιομηχανία ανοίγει νέους ορίζοντες για την ασφάλεια, τη συντήρηση, την απόδοση και την οδική εμπειρία.

Εικόνα 6:Μεταξύ τους επικοινωνία αυτοκίνητων

(πηγή:https://www.thecarconnection.com/news/1107793_vehicle-to-vehicle-communication-may-finally-be-mandatory-on-new-cars)

2.9 Εξέλιξη της διάγνωσης στο μέλλον

Η εξέλιξη της διάγνωσης βλαβών στα αυτοκίνητα στο μέλλον υπόσχεται να φέρει σημαντικές τεχνολογικές και λειτουργικές βελτιώσεις. Οι εξελίξεις αυτές συνδέονται στενά με την τρομερή πρόοδο που σημειώνεται στον τομέα της τεχνητής νοημοσύνης (ΤΝ), των αισθητήρων, του Διαδικτύου των Πραγμάτων (IoT) και των μεγάλων δεδομένων.

•**Τεχνητή Νοημοσύνη και Μηχανική Μάθηση:** Οι προηγμένοι αλγόριθμοι ΤΝ και η μηχανική μάθηση θα επιτρέψουν την πρόβλεψη βλαβών προτού αυτές προκαλέσουν προβλήματα. Οι αυτό-προσαρμοζόμενοι αλγόριθμοι θα είναι σε θέση να αντιλαμβάνονται τις αλλαγές στις συνθήκες λειτουργίας του οχήματος και να προσαρμόζουν τη διαγνωστική διαδικασία ανάλογα.

•**Σύνδεση με το Cloud**: Οι διαγνωστικές πληροφορίες θα στέλνονται σε ασφαλείς υποδομές cloud για ανάλυση. Αυτό θα επιτρέψει τη σύγκριση με μεγάλα σύνολα δεδομένων, βοηθώντας στην ανίχνευση προτύπων και στη βελτιστοποίηση των διαγνωστικών αλγορίθμων.

•**Επικοινωνία Οχημάτων μεταξύ τους (V2V):** Τα οχήματα θα ανταλλάσσουν πληροφορίες σχετικά με την κατάσταση των συστημάτων τους. Αυτή η συνεργασία μεταξύ οχημάτων θα επιτρέπει προειδοποιήσεις για πιθανά προβλήματα.

•**Εξειδικευμένοι Αισθητήρες:** Η χρήση πιο εξειδικευμένων αισθητήρων, όπως τεχνολογία LIDAR για ανίχνευση αντικειμένων, θα ενισχύσει την ικανότητα διάγνωσης.

•**Απομακρυσμένη Διάγνωση:** Οι τεχνολογίες IoT θα επιτρέψουν στους κατασκευαστές και τους τεχνικούς να πραγματοποιούν απομακρυσμένη διάγνωση και παρέμβαση στα οχήματα.

Συνολικά, οι μελλοντικές εξελίξεις στη διάγνωση βλαβών θα επιφέρουν πιο αξιόπιστες, γρήγορες και προηγμένες διαδικασίες, συμβάλλοντας στην ασφάλεια, την απόδοση και τη συνολική εμπειρία των χρηστών στον τομέα των μεταφορών.

3 ARDUINO

3.1 Τι είναι το ARDUINO

Το Arduino είναι μια πλακέτα ανοικτού κώδικα (open source) με εισόδους και εξόδους εντολών, η οποία μπορεί να προγραμματιστεί με τη γλώσσα προγραμματισμού Wiring C, που είναι παρόμοια με τη C++. Επιπλέον, χρησιμοποιεί διάφορες βιβλιοθήκες γραμμένες σε C++. Το Arduino αποτελεί ιδανικό εργαλείο για την ανάπτυξη project σε συνδυασμό με τη χρήση αισθητήρων διαφόρων τύπων.

Εικόνα 7:Λογότυπο ARDUINO

(πηγή: https://www.arduino.cc/)

3.2 Πλεονεκτήματα της χρήσης πλακέτας ARDUINO

1. Χαμηλό κόστος: Το κόστος μιας πλακέτας είναι πολύ προσιτό για οποιονδήποτε θέλει να ασχοληθεί με open source programming. Υπάρχουν πακέτα που παρέχουν, εκτός από πλακέτα, μια ποικιλία ηλεκτρονικών εξαρτημάτων όπως καλώδια, αισθητήρες, ηλεκτροκινητήρες κατάλληλα για τις αρχικές δοκιμές και κατασκευές για οποιονδήποτε θέλει να πειραματιστεί .
2. Δεν βασίζεται σε καμία πλατφόρμα (cross-platform): Το λογισμικό του Arduino λειτουργεί σε όλα τα συστήματα (Windows, macOS, Linux), καλύπτοντας όλους τους χρήστες .
3. Οι ποικίλες εκδόσεις πλακετών Arduino : Κάθε μοντέλο πλακέτας Arduino που υπάρχει καλύπτει διαφορετικές ανάγκες και έχει διαφορετικές δυνατότητες ανάλογα με την χρήση

που θέλει να την κάνει ο καθένας. Επίσης υπάρχουν και εξαρτήματα επέκτασης (shield's) ενισχύουν και δίνουν νέες δυνατότητες στις πλακέτες arduino.

4. Η Απλότητα αποτελεί ίσως το πιο σημαντικό πλεονέκτημα της πλακέτας Arduino : Ένας αρχάριος χρήστης μπορεί να δημιουργήσει μια απλή κατασκευή χωρίς να έχει μεγάλη εμπειρία στον προγραμματισμό.

Εικόνα 8:Τύποι πλακετών Arduino

(πηγη:https://techatronic.com/types-of-arduino-boards-arduino-uno-mega-mini-specification/)

3.3 Πλακέτες ARDUINO:

•Arduino Uno: Μία από τις πιο κοινές και ευρέως χρησιμοποιούμενες πλακέτες, κατάλληλη για αρχάριους.

•Arduino Mega: Μεγαλύτερη από την Uno, με περισσότερες ψηφιακές και αναλογικές είσοδούς και εξόδους.

•Arduino Nano: Μία συμπαγής έκδοση της Uno, κατάλληλη για μικρότερα έργα.

•Arduino Due: Βασισμένη σε έναν ισχυρότερο επεξεργαστή ARM Cortex-M3.

•Arduino Pro Mini: Μια ελαφρώς μικρότερη έκδοση σχεδιασμένη για εφαρμογές με περιορισμένο χώρο.

•Arduino Leonardo: Το Arduino Leonardo είναι παρόμοια πλακέτα με το Arduino Micro διότι χρησιμοποιούν τον ίδιο μικροελεγκτή (ATmega32U4) με ενσωματωμένη USB επικοινωνία όπου δίνουν την δυνατότητα να τις αναγνωρίζει ο υπολογιστής σαν συσκευές πληκτρολογίου ή ποντίκι και να επικοινωνούν με αυτόν χωρίς να απαιτείται δεύτερος μικροελεγκτής για την επικοινωνία όπως το Arduino Uno που βασίζεται στον μικροελεγκτή ATmega328.

•Arduino Micro: Παρόμοια με την Leonardo, με μικρότερο μέγεθος.

•Arduino Mini: Μια παλαιότερη έκδοση, μικρότερη από την Pro Mini.

•Arduino MKR Series (MKR1000, MKRZero, κλπ.): Κατευθυνόμενες προς τις εφαρμογές IoT (Διαδίκτυο των Πραγμάτων).

•Arduino Yun: Συνδυάζει την Arduino με έναν επεξεργαστή βασισμένο σε Linux, κατάλληλη για εφαρμογές IoT.

•Arduino Esplora: Σχεδιασμένη για την ανάπτυξη παιχνιδιών και πειραματισμού.

•Arduino Pro (Pro 328 και Pro 168): Κατάλληλες για προχωρημένους χρήστες και έργα με συγκεκριμένες απαιτήσεις.

•Arduino Gemma: Μια μικρή, φορετή πλακέτα σχεδιασμένη για ύφασματα και φορετά project.

•Arduino Tian: Συνδυάζει τη δύναμη του Linux με την ευκολία της Arduino.

•Arduino Industrial: Κατευθυνόμενη προς βιομηχανικές εφαρμογές.

•Arduino Mega ADK: Παρόμοια με τη Mega, με δυνατότητες ανάπτυξης αξεσουάρ Android.

•Arduino Robot: Σχεδιασμένη για εφαρμογές ρομποτικής, με οδηγούς κινητήρα και αισθητήρες.

•Arduino DUE: Διαθέτει πυρήνα ARM 32-bit και είναι ιδανική για πιο προχωρημένους χρήστες και έργα.

3.4 Από τι αποτελείται μια πλακέτα ARDUINO

Μια πλακέτα Arduino αποτελείται από τα παρακάτω βασικά στοιχεία:

•**Μικροελεγκτής (Microcontroller):** Η καρδιά της πλακέτας, συνήθως βασίζεται σε μια οικογένεια μικροελεγκτών της εταιρείας Atmel, όπως οι ATmega328, ATmega2560, κ.ά.

•**Κρυσταλλικός Ταλαντωτής (Crystal Oscillator):** Ο χρονοκράτης του συστήματος που συνήθως λειτουργεί στα 16MHz.

•**Είσοδοι/Εξοδοι (I/O) Pins:** Διάφορες ψηφιακές και αναλογικές είσοδοι και έξοδοι που επιτρέπουν τη σύνδεση με αισθητήρες, ενεργοποιητές και άλλες συσκευές.

•**Μνήμες:**

i.Flash Memory: Για την αποθήκευση του προγράμματος.

ii.SRAM (Static Random Access Memory): Μνήμη RAM που χρησιμοποιείται κατά τη διάρκεια εκτέλεσης του προγράμματος.

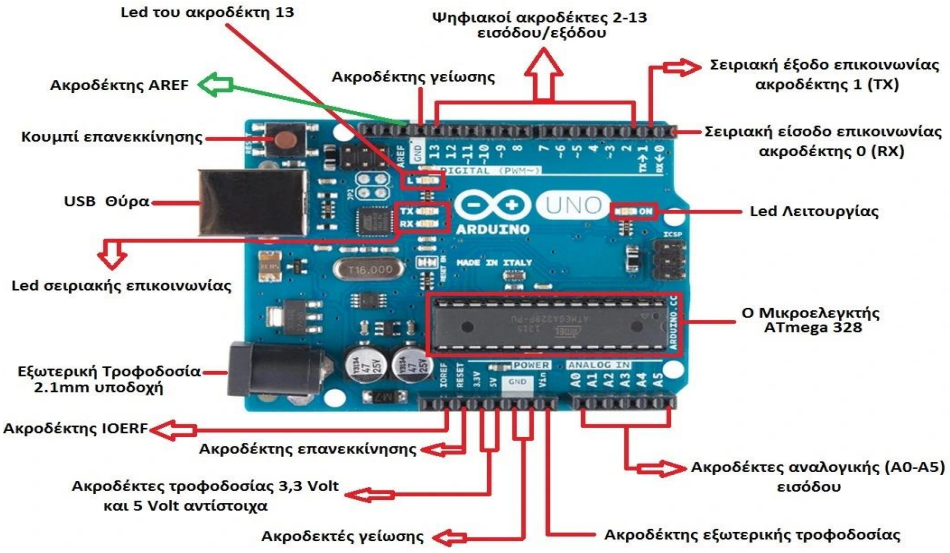
•**EEPROM (Electrically Erasable Programmable Read-Only Memory):** Για μόνιμη αποθήκευση δεδομένων που διατηρούνται ακόμη και μετά την απώλεια τροφοδοσίας.

•**Αισθητήρες/Συνδέσεις:** Περιλαμβάνει συνδέσεις USB, τροφοδοσία, και άλλες που διευκολύνουν τη χρήση και τον προγραμματισμό.

•**Ρυθμιστής Τάσης (Voltage Regulator):** Παρέχει σταθερή τάση για τη λειτουργία της πλακέτας.

•**Κουμπί επαναφοράς (Reset Button):** Για τον έλεγχο και την επαναφορά του μικροελεγκτή.

Αυτά τα βασικά στοιχεία που συνθέτουν μια πλακέτα Arduino, για δημιουργία διάφορων έργων και εφαρμογών.



Εικόνα 9:Δομή πλακέτας Arduino

(πηγή: https://projectmaniacs.files.wordpress.com/2014/11/ardu11.jpg)

3.5 Arduino IDE – Προγραμματισμός πλακέτας

Το Arduino IDE (Integrated Development Environment) είναι ένα περιβάλλον ανάπτυξης λογισμικού που χρησιμοποιείται για τον προγραμματισμό των πλακετών Arduino. Είναι σχεδιασμένο με σκοπό τη δημιουργία και τη μεταφόρτωση προγραμμάτων (γνωστών ως "sketches") στις πλακέτες Arduino, επιτρέποντας στους χρήστες να επικεντρωθούν στην ανάπτυξη του λογισμικού για τα project τους. Ο βασικός κύκλος εργασιών για τον προγραμματισμό μιας πλακέτας Arduino μέσω του Arduino IDE περιλαμβάνει τα εξής βήματα:

1.Εγκατάσταση του Arduino IDE: Εγκατάσταση του Arduino IDE από την επίσημη ιστοσελίδα της Arduino.

2.Σύνδεση της πλακέτας Arduino: Σύνδεση της πλακέτας Arduino στον υπολογιστή με χρήση ενός καλωδίου USB.

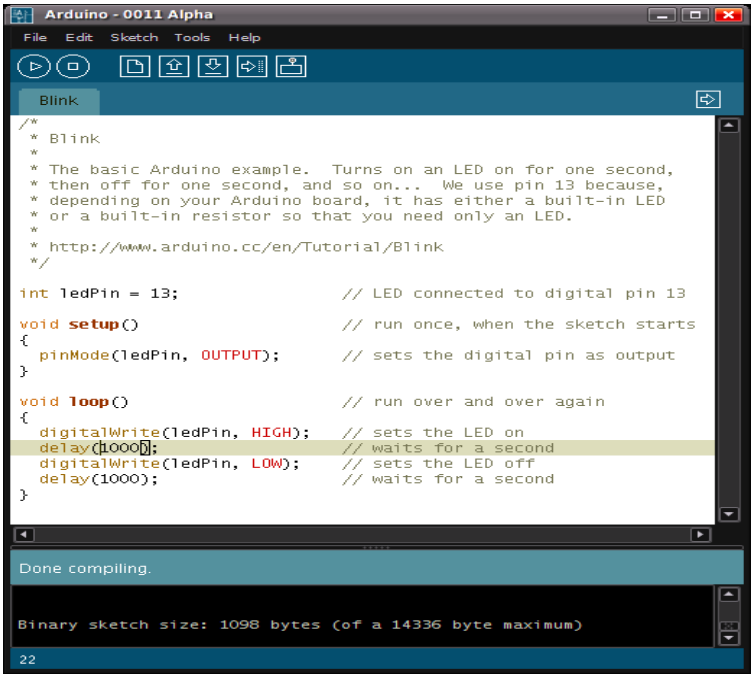
3.Επιλογή του Τύπου Πλακέτας και Θύρας: Επιλογή του σωστού τύπου πλακέτας Arduino από το μενού "Εργαλεία" (Tools) και την σωστή θύρα COM που είναι συνδεδεμένη με την πλακέτα.

4.Σύνταξη (Compile) του Sketch: εγγραφή του προγράμματος (Sketch) στο Arduino IDE και "Σύνταξη" (Compile) για τον έλεγχο σφαλμάτων στον κώδικα.

5.Μεταφόρτωση (Upload) του Sketch: Επιλογή μεταφόρτωσης "Μεταφόρτωση" (Upload) στην πλακέτα Arduino.

6.Εκτέλεση του Προγράμματος: Μόλις ολοκληρωθεί η μεταφόρτωση, του προγράμματος εκτελείται στην πλακέτα Arduino και αυτή αρχίζει να εκτελεί τις λειτουργίες τις οποίες έχει προγραμματιστεί.

Το Arduino IDE υποστηρίζει τη γλώσσα προγραμματισμού Wiring, που είναι παρόμοια με τη C++. Επίσης, παρέχει βιβλιοθήκες που επεκτείνουν τις δυνατότητες των πλακετών Arduino, καθιστώντας τον προγραμματισμό πιο εύκολο και ευέλικτο.



Εικόνα 10:Λογισμικό Arduino

(πηγή: https://el.wikipedia.org/wiki/Arduino)

4 Τεχνικός εξοπλισμός συστήματος

4.1 Στόχοι συστήματος

Στόχος είναι η κατασκευή μιας συσκευής που παρέχει σε πραγματικό χρόνο δεδομένα του κινητήρα μέσω της OBD στον χρήστη. Αυτή η συσκευή λειτουργεί ως δεύτερο όργανο πολλαπλών ενδείξεων, παρέχοντας δεδομένα με μεγαλύτερη ακρίβεια, ενώ εμφανίζει επιπλέον πληροφορίες που δεν είναι διαθέσιμες σε ένα τυπικό κοντέρ αυτοκινήτου. Τέλος , τα συνολικά δεδομένα καταγράφονται σε εξωτερική κάρτα μνήμης, προσφέροντας τη δυνατότητα μετέπειτα μελέτης και επεξεργασίας .

4.2 Τεχνικός εξοπλισμός

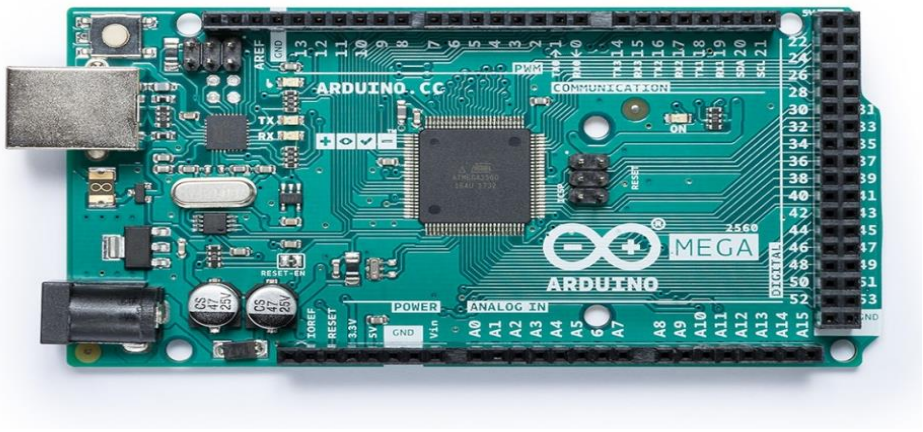
Για την κατασκευή, χρησιμοποιήθηκαν τα παρακάτω υλικά προκειμένου να δημιουργηθεί ένα ενδιαφέρον και λειτουργικό σύστημα. Βασικό στοιχείο της κατασκευής είναι το Arduino Mega 2560 R3, μια προηγμένη μονάδα ελέγχου που δίνει τη δυνατότητα προγραμματισμού και ελέγχου ποικίλων λειτουργιών.

Για την ανάγνωση μετρήσεων από το όχημα, χρησιμοποιήθηκε ένας αντάπτορας OBD όπου είναι συμβατός με τις πλακέτες Arduino. Αυτός ο αντάπτορας διευκολύνει τη σύνδεση με το OBD-II πρωτόκολλο και επιτρέπει την λήψη δεδομένων που σχετίζονται με την κατάσταση του οχήματος, όπως η ταχύτητα, οι περιστροφές του κινητήρα και άλλες χρήσιμες πληροφορίες, επίσης έχει ενσωματωμένο σύστημα MEMS.

Για την απεικόνιση αυτών των δεδομένων, ενσωματώθηκε μια οθόνη TFT 3,5 ιντσών. Αυτή η έγχρωμη οθόνη προσφέρει ευανάγνωστα πληροφορίες σε πραγματικό χρόνο, ενώ η επικοινωνία της με το Arduino Mega 2560 R3 εξασφαλίζει ομαλή λειτουργία και ανταπόκριση.

Επιπλέον, ενσωματώθηκε ένας αισθητήρας GPS για την ακριβή παρακολούθηση και αποθήκευση της τοποθεσίας του οχήματος. Αυτός ο αισθητήρας δίνει τη δυνατότητα για γεωγραφικό προσδιορισμό, παρέχοντας σημαντικές πληροφορίες για την πορεία του οχήματος. Ο συνδυασμός αυτών των στοιχείων δημιουργεί ένα σύστημα, που επιτρέπει την παρακολούθηση και την αποθήκευση αυτών των δεδομένων με έναν εντυπωσιακό και χρήσιμο τρόπο.

4.3 ARDUINO MEGA 2560 R3



Εικόνα 11: Πλακέτα Arduino mega 2560r3

(πηγή: https://store.arduino.cc/products/arduino-mega-2560-rev3)

Το Arduino Mega 2560 R3 είναι ένα από τα δημοφιλή μοντέλα Arduino. Χρησιμοποιείται σε project με υψηλές απαιτήσεις όπως η Ρομποτική. Μερικά βασικά χαρακτηριστικά του Arduino Mega 2560 R3:

- **Μικροεπεξεργαστής:**

Το Arduino Mega 2560 R3 χρησιμοποιεί τον ATmega2560, που είναι ένας 8-bit μικροεπεξεργαστής χρονισμένος στα 16 MHz.

- **Αριθμός Ψηφιακών Εισόδων/Εξόδων:**

Διαθέτει 54 ψηφιακές εισόδους/εξόδους, από τις οποίες 15 μπορούν να χρησιμοποιηθούν ως PWM έξοδοι.

- **Αριθμός Αναλογικών Εισόδων:**

Διαθέτει 16 αναλογικές εισόδους, που μπορούν να χρησιμοποιηθούν για την ανάγνωση αναλογικών σημάτων.

- **Μνήμη:**

Έχει 256 KB flash μνήμη για το πρόγραμμα, 8 KB SRAM και 4 KB EEPROM.

- **Επικοινωνία:**

Υποστηρίζει σύνδεση USB για τη μεταφορά του προγράμματος και σειριακή επικοινωνία (UART) για σύνδεση με άλλες συσκευές.

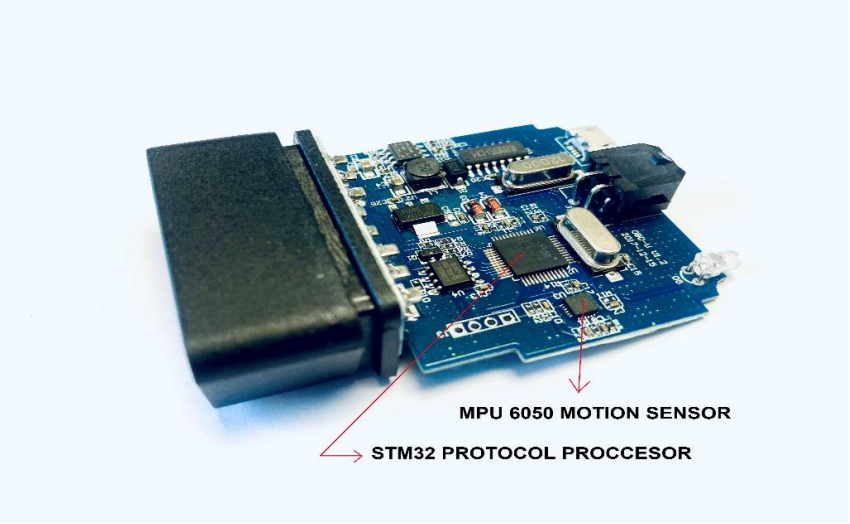
- **Επεκτασιμότητα:**

- Έχει πολλούς ακροδέκτες για τη σύνδεση επιπλέον συσκευών, όπως αισθητήρες, εκτυπωτές, οθόνες κ.ά.

- **Συμβατότητα:**

Είναι συμβατό με το περιβάλλον ανάπτυξης Arduino IDE, που καθιστά ευκολότερο τον προγραμματισμό της πλακέτας.

4.4 Αντάπτορας OBD με ενσωματωμένο αισθητήρα MEMS



Εικόνα 12:Αντάπτορας OBD συμβατός με Arduino

Το Freematics OBD-II UART λειτουργεί ως γέφυρα δεδομένων μεταξύ της θύρας OBD-II ενός αυτοκινήτου και του Arduino. Εκτός από την πρόσβαση σε δεδομένα OBD-II, ενσωματώνει επίσης έναν αισθητήρα MEMS 6 αξόνων και ένα βολτόμετρο για τη μέτρηση της ενέργειας της μπαταρίας του οχήματος.

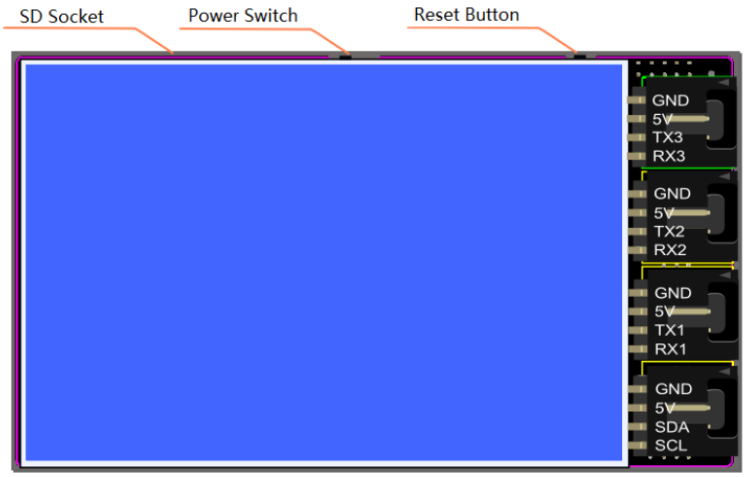
Ο αντάπτορας τροφοδοτείται από τη θύρα OBD-II και μετατρέπει την τάση από 12V σε 5V (έως 2A), τροφοδοτώντας τόσο τον εαυτό του όσο και τη συνδεδεμένη συσκευή στην παρούσα φάση την πλακέτα Arduino mega 2560 την οθόνη TFT και τον αισθητήρα GPS . Ο αντάπτορας συνδέεται στη θύρα OBD, η οποία συνήθως βρίσκεται κάτω από το τιμόνι ή ελαφρώς αριστερά από αυτό.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ:

- Υποστήριξη πρωτοκόλλων CAN bus και KWP2000.
- Ενσωματωμένος αισθητήρας MPU-6050 MEMS (επιταχυνσιόμετρο, γυροσκόπιο και αισθητήρα εξωτερικής θερμοκρασίας).
- Ενσωματωμένος αισθητήρας τάσης για την μέτρηση τάσης μπαταρίας αυτοκινήτου.
- Θύρα Micro USB για πρόσβαση και αναβάθμιση firmware OBD-II από υπολογιστή/τάμπλετ.
- Συμβατό σύνολο εντολών AT ELM327.
- Γρήγορη (έως 100Hz) ανάγνωση OBD-II PIDs διαθέσιμη από τη μονάδα ECU του οχήματος.

- Παροχή τάσης σε συνδεδεμένες συσκευές με DC 5V (έως 2A)

4.5 TFT LCD οθόνη



Εικόνα 13: Οθόνη LCD TFT

(πηγή: https://www.dfrobot.com/product-1500.html)

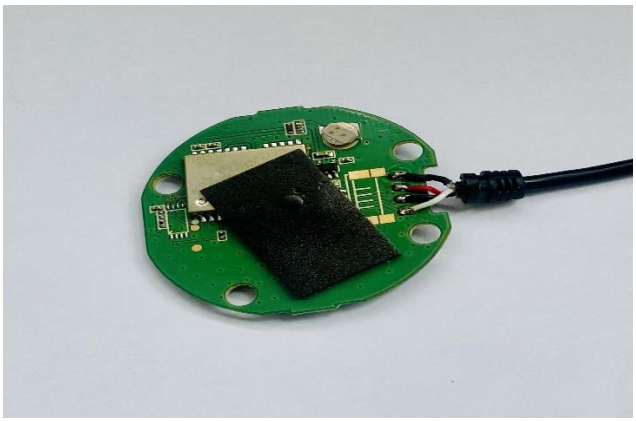
Η TELEMATICS 3.5" TFT LCD Shield είναι μια οθόνη συμβατή με το Arduino, σχεδιασμένη από την DFRobot, με ανάλυση 320x480, έχει τρεις σειριακές θύρες και μια διεπαφή IIC. Είναι πλήρως συμβατή με τα Arduino Mega1280/2560 και Bluno Mega1280/2560. Διαθέτει έναν διακόπτη τάσης στην πλακέτα που υποστηρίζει την αλλαγή της τάσης εξόδου μεταξύ 3.3V και 5V. Επιπλέον, η οθόνη διαθέτει υποδοχή MicroSD στο πίσω μέρος που μπορεί να χρησιμοποιηθεί για αποθήκευση δεδομένων έως 32GB.

Προδιαγραφές:

- Μέγεθος Οθόνης: 3.5"
- Ανάλυση: 320x480
- Τροφοδοσία: 5V
- Τάση Εξόδου: 3.3/5V
- Λειτουργία Έλεγχου Φωτισμού: Σήμα PWM D8
- Υποστηρίζει κάρτα MicroSD (Έως 32GB)
- Σειριακές Θύρες: 3
- Διεπαφή IIC: 1
- Συμβατό με Arduino DUE/Mega 1280/2560
- Συμβατό με το Bluno Mega 1280/2560
- Διαστάσεις: 100x57mm/3.93x2.24 ίντσες

- Βάρος: 70g

4.6 Αισθητήρας GPS



Εικόνα 14: Δέκτης GPS συμβατός με Arduino

Ο Δέκτης GPS για το Arduino είναι μια συσκευή που ενσωματώνει ένα GPS module και μια κεραία. Χρησιμοποιώντας τη βιβλιοθήκη TinyGPS, το Arduino μπορεί να λάβει γεωγραφικές συντεταγμένες (γεωγραφικό πλάτος και μήκος, υψόμετρο), ταχύτητα, κατεύθυνση και ώρα GMT.

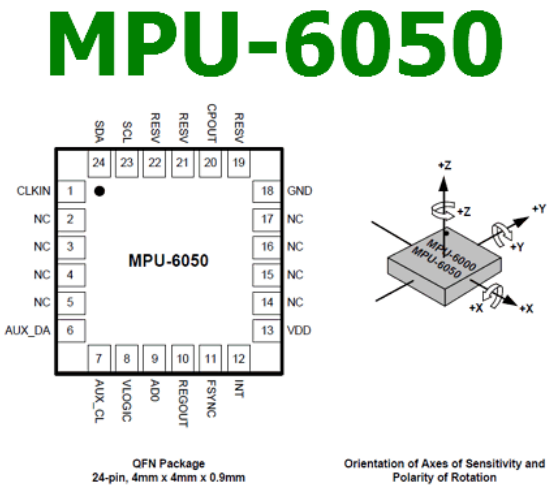
Ο ρυθμός ανανέωσης είναι ένας σημαντικός δείκτης απόδοσης ενός δέκτη GPS. Οι περισσότεροι δέκτες στα κινητά τηλέφωνα παρέχουν έναν ρυθμό ανανέωσης με ταχύτητα 1Hz, πράγμα που σημαίνει ότι μπορεί να ανακτηθεί μόνο ένα σύνολο δεδομένων κατά τη διάρκεια ενός δευτερολέπτου. Για τους δέκτες GPS με 5Hz, το διάστημα ανανέωσης δεδομένων είναι πολύ μικρότερο και, επομένως, μπορεί να χρησιμοποιηθεί για πιο απαιτητικές εφαρμογές (π.χ. σε γρήγορα κινούμενα οχήματα).
Ο δέκτης GPS έχει 4 επαφές , η πρώτη επαφή είναι το ρεύμα(+) VCC , η δεύτερη επαφή είναι το σώμα (-) GND και οι επόμενες δυο επαφές είναι για την μεταφορά δεδομένων RX , TX .

Προδιαγραφές:

- Έξοδος 5Hz
- Σειριακή διεπαφή TTL 115200bps
- Τάση εισόδου 3.3V-5V
- Δέκτης 50 καναλιών
- Ακρίβεια 2.5m (Αυτόνομο) / <2m [SBAS]
- Θερμοκρασία λειτουργίας: -40°C έως 85°C
- Κατανάλωση ισχύος: 3.3V @ 41mA
- Hot Start: 1s
- Warm Start: 32s

- Cold Start: 32s
- Κεραμική κεραία 25*25*2mm
- Μέγεθος Μονάδας 25*28*7.5mm

4.7 Ενσωματωμένος αισθητήρας MPU6050



Εικόνα 15: Αισθητήρας MPU6050

(πηγή: https://www.datasheetcafe.com/mpu-6050-datasheet-pdf/)

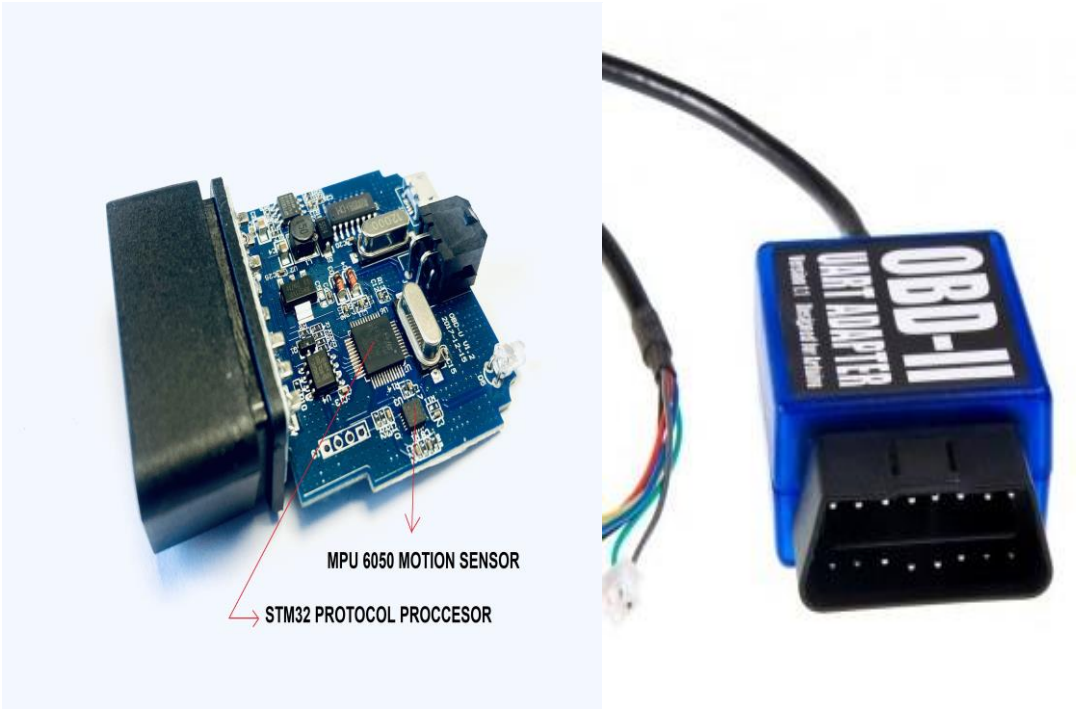
Το MPU6050 είναι ένα δημοφιλές ενσωματωμένο κύκλωμα (IC) που συνδυάζει 3-άξονες γυροσκοπίου και 3-άξονες επιταχυνσιόμετρου σε ένα μόνο πακέτο. Αναπτύχθηκε από την InvenSense (τώρα μέρος της TDK Corporation) και χρησιμοποιείται ευρέως σε εφαρμογές που απαιτούν ανίχνευση κίνησης και παρακολούθηση.

Μερικά βασικά χαρακτηριστικά και λεπτομέρειες σχετικά με το MPU6050:

- Μετρά τόσο τον ρυθμό περιστροφής (γωνιακή ταχύτητα) όσο και τη γραμμική επιτάχυνση κατά μήκος τριών κάθετων αξόνων.
- Το γυροσκοπικό μετρά τον ρυθμό περιστροφής (γωνιακή ταχύτητα) γύρω από τους άξονες X, Y και Z. Παρέχει δεδομένα σε μοίρες ανά δευτερόλεπτο (dps).
- Το επιταχυνσιόμετρο μετρά τη γραμμική επιτάχυνση κατά μήκος των αξόνων X, Y και Z. Παρέχει δεδομένα σε μονάδες επιτάχυνσης μέτρου ανά δευτερόλεπτο (m/s^2) ή δύναμη βαρύτητας (g).
- Το MPU6050 επικοινωνεί με έναν μικροελεγκτή ή άλλες συσκευές μέσω του πρωτοκόλλου επικοινωνίας I2C (Inter-Integrated Circuit).
- χρησιμοποιείται συχνά σε εφαρμογές όπως μονάδες μέτρησης αδράνειας (IMUs), drones, ρομποτικά, χειριστήρια παιχνιδιών, συστήματα εικονικής πραγματικότητας και άλλα project που απαιτούν ανίχνευση κίνησης και παρακολούθηση προσανατολισμού.

5 Κατασκευή συστήματος αποθήκευσης δεδομένων και οπτικοποίησης

Αρχική πρόκληση της διαδικασίας κατασκευής υπήρξε η ανάγκη αποτελεσματικής επικοινωνίας με τη θύρα OBD (On-Board Diagnostics). Για την επίτευξη αυτού του στόχου, επιλέχτηκε ένας αντάπτορας που ονομάζεται OBDII Uart Adapter συμβατός με τη χρήση του Arduino Mega 2560 και παράλληλα πληρώντας τις απαιτήσεις επικοινωνίας και μεταφοράς δεδομένων μέσω των OBD πρωτόκολλων.



Εικόνα 16: OBDII UART ADAPTER

Το OBDII UART Adapter για Arduino λειτουργεί ως γέφυρα δεδομένων μεταξύ της OBD-II θύρας του οχήματος και της σειριακής θύρας UART του Arduino, έτσι επιτυγχάνεται η ανάγνωση δεδομένων όπως η ταχύτητα ή οι στροφές του κινητήρα από τον υπολογιστή της ECU του αυτοκινήτου. Ο αντάπτορας ενσωματώνει έναν αισθητήρα MPU-6050 για δεδομένα επιτάχυνσης αλλά και γυροσκοπικά δεδομένα με διασύνδεση I2C. Ο αισθητήρας επιτάχυνσης μπορεί να χρησιμοποιηθεί για τη μέτρηση της επιτάχυνσης και της δύναμης G. Ο γυροσκοπικός αισθητήρας μπορεί να χρησιμοποιηθεί για τη μέτρηση του προσανατολισμού του αυτοκινήτου και τις αλλαγές στην κατεύθυνση.

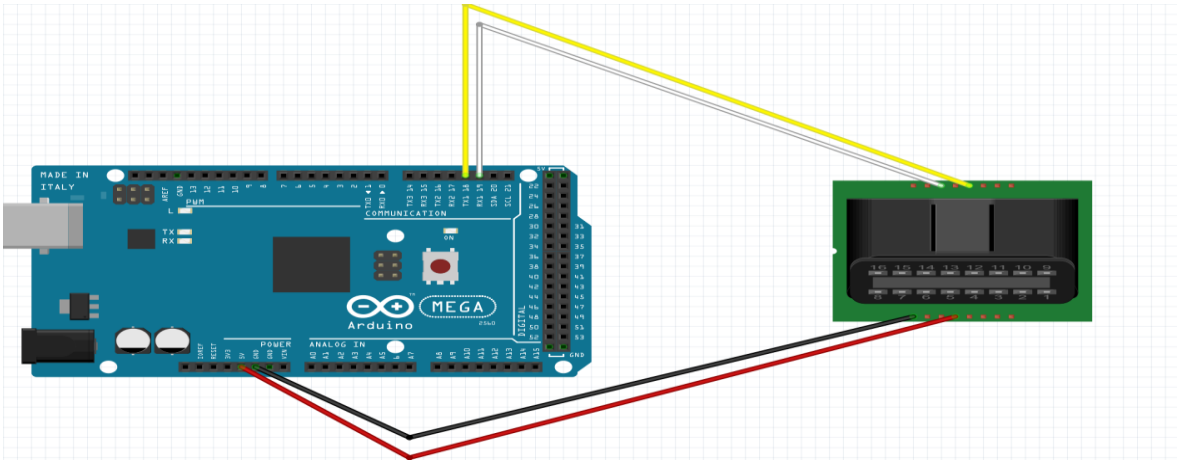
Εκτός από την παροχή πρόσβασης σε δεδομένα OBD-II, παρέχει και μετατρέπει τα 12V DC της θύρας OBD σε 5V για την τροφοδοσία του Arduino και τις συνδεδεμένες συσκευές πάνω σε αυτό.

Ο αντάπτορας διαθέτει ένα κανονικό βύσμα που μπορεί να συνδεθεί στη θύρα OBD-II, συνήθως κάτω από το ταμπλό του αυτοκινήτου. Ένα καλώδιο βγαίνει από τον προσαρμογέα και χωρίζεται σε 4 επαφές

- VCC : ρεύμα τροφοδοσίας 5v

- GND : σώμα
- Rx : επαφή επικοινωνίας δεδομένων
- Tx : επαφή επικοινωνίας δεδομένων

Η διασύνδεση γίνεται όπως φαίνεται στο παρακάτω διάγραμμα :



Εικόνα 17: Διασύνδεση OBDII Uart στην πλακέτα Arduino mega2560

Έπειτα χρειαζόταν μια οθόνη η οποία θα μετέδιδε τα δεδομένα σε πραγματικό χρόνο με ικανοποιητική ανάλυση των δεδομένων, για αυτό το λόγο επιλέχτηκε μια οθόνη TFT LCD 3.5” με ανάλυση 320x480. Η οθόνη LCD διαθέτει μια υποδοχή κάρτας MicroSD στο πίσω μέρος που μπορεί να χρησιμοποιηθεί για αποθήκευση δεδομένων έως και 32GB με αποτέλεσμα να μην χρειάζεται ένα ξεχωριστό SD module απλοποιώντας την κατασκευή.

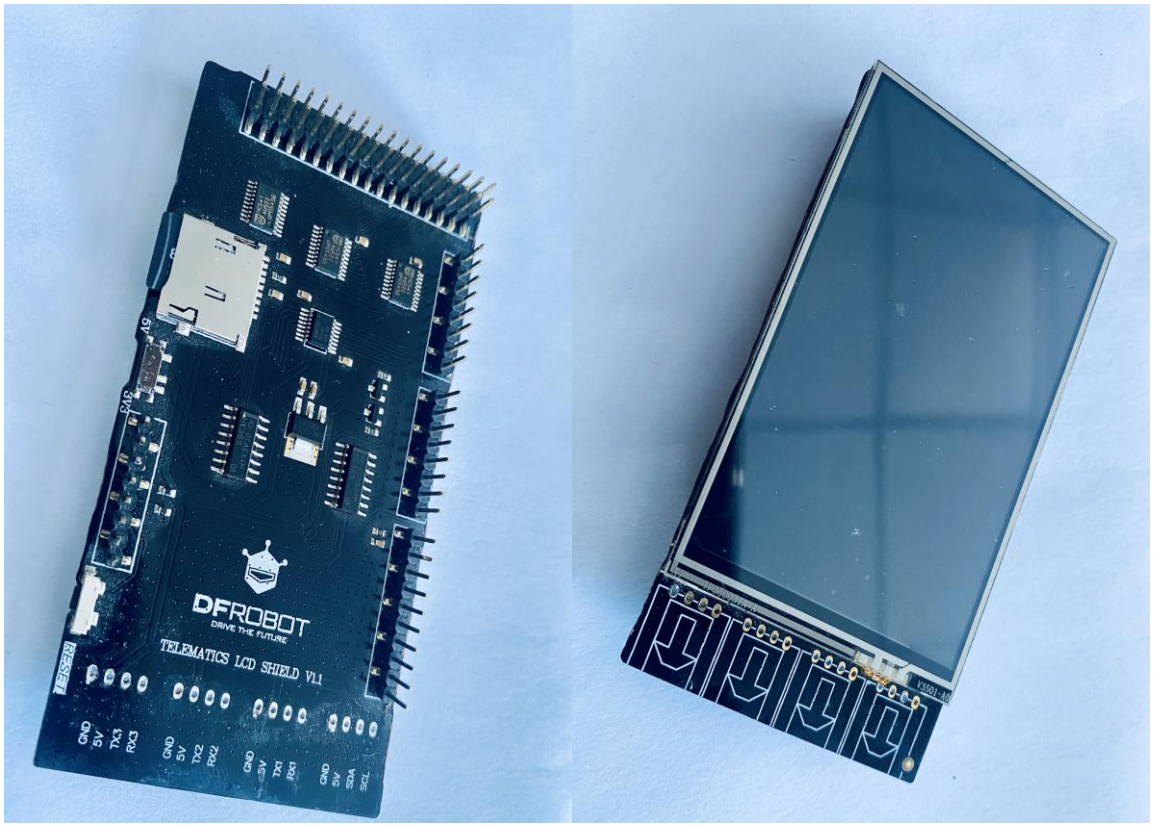
Για την σύνδεση της οθόνης στο Arduino Mega 2560 οι επαφές ενσωματώνονται απευθείας πάνω στην πλακέτα χωρίς να χρειάζεται η χρήση καλωδίων. Οι επαφές συνδέονται όπως φαίνεται παρακάτω:

- Σύνδεση των Δεδομένων για την χρήση SD κάρτας :

- D52 συνδέεται στο DATA0.
- D50 συνδέεται στο CMD.
- D52 συνδέεται στο CLK.
- D53 συνδέεται στο CD/DATA3.

- Σύνδεση επαφών μεταφοράς εικόνας :

- οι επαφές που χρησιμοποιούνται είναι: D2, D3, D4, D5, D6, D8, D22 έως D41, D50 έως D53.

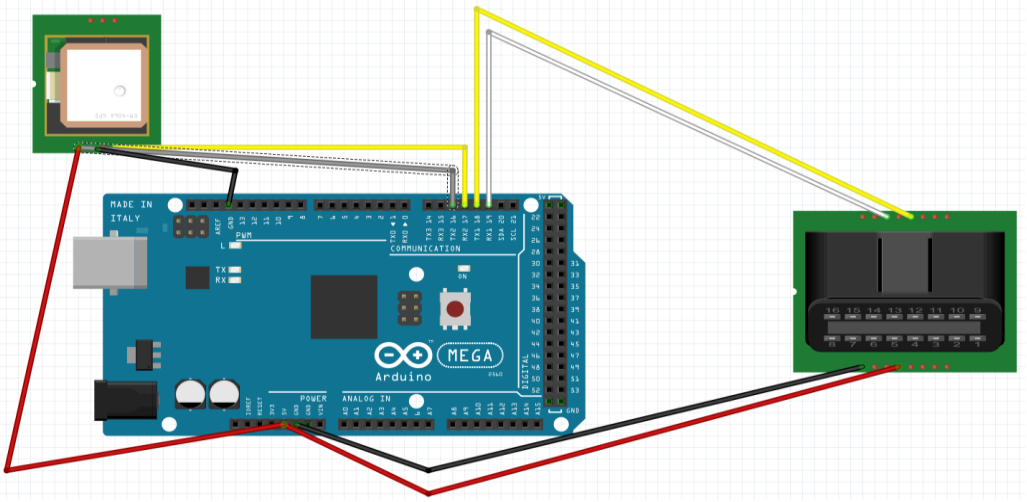


Εικόνα 18: TFT LCD 3.5”

(πηγή: https://wiki.dfrobot.com/TELEMATICS_3.5__TFT_Touch_LCD_Shield_SKU__DFR0387)

Τελικό κομμάτι της κατασκευής όσον αφορά το hardware είναι το GPS module. Αυτό το module επιτρέπει τη λήψη γεωγραφικών συντεταγμένων ακριβείας, χρησιμοποιώντας τα σήματα από δορυφόρους. Με διαθέσιμες βιβλιοθήκες, η ενσωμάτωση του GPS module μας δίνει την δυνατότητα παρακολούθησης της θέσης του οχήματος.

Ο τρόπος διασύνδεσης του GPS module στην πλακέτα φαίνεται στο παρακάτω διάγραμμα:



Εικόνα 19: Διασύνδεση GPS module στην πλακέτα Arduino mega256

Για να λειτουργήσει η κατασκευή και να μπορέσουν όλα τα module να αναγνωριστούν από την κεντρική πλακέτα πρέπει να προγραμματιστεί. Για τον προγραμματισμό της εγκαταστάθηκε στον υπολογιστή το Arduino Software (IDE). Η έκδοση που χρησιμοποιείται είναι Arduino 2.2.1, πρέπει να αναφερθεί ότι σε επόμενες εκδόσεις ο κώδικας μπορεί να χρειάζεται τροποποιήσεις για να είναι λειτουργικός.

Αρχικά δηλώθηκαν οι βιβλιοθήκες που χρειάζονται για να λειτουργήσουν τα modules που είναι συνδεμένα στο Arduino:

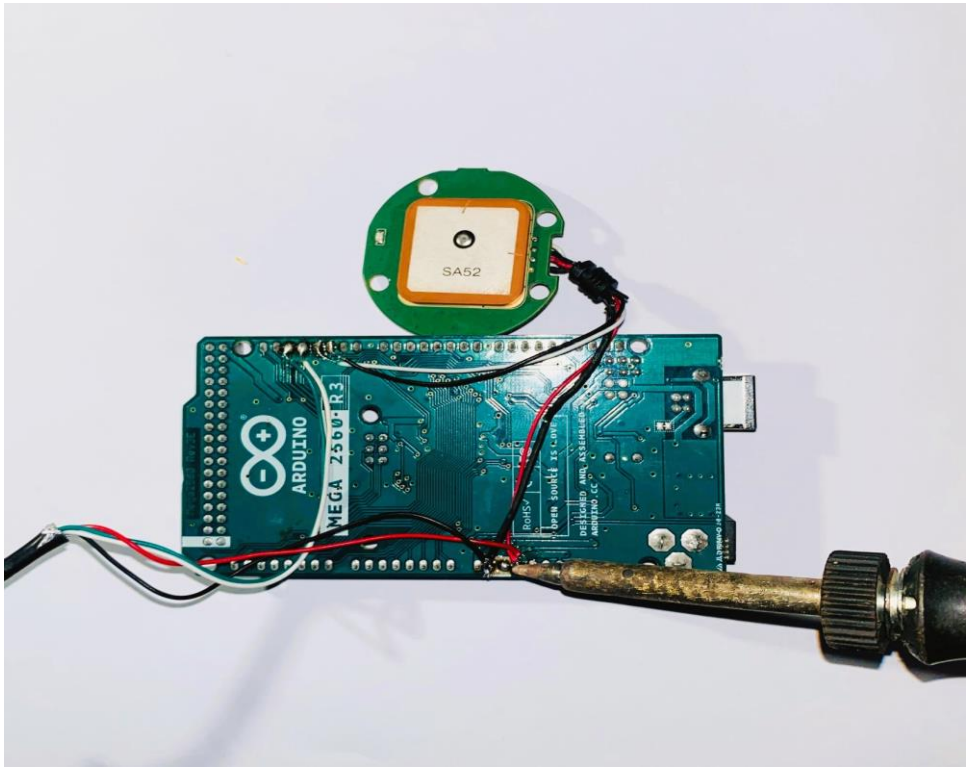
- <Arduino.h>
- <SD.h>
- <Wire.h>
- <SPI.h>
- <OBD.h>
- <MultiLCD.h>
- <TinyGPS.h>

Οι 4 πρώτες βιβλιοθήκες βρίσκονται προεγκατεστημένες στο Arduino IDE και αφορούν την σειριακή επικοινωνία, την αποθήκευση και αναζήτηση δεδομένων. Οι υπόλοιπες βρίσκονται στο https://github.com/ . Η OBD.h χρησιμοποιείται για να επικοινωνήσει ο αντάπτορας OBDII Uart με το όχημα και να διαβάσει δεδομένα από αυτό βάσει των αναγκών του χρήστη. Η MultiLCD.h καθιστά λειτουργική την TFT LCD οθόνη. Η TinyGPS έχει στόχο την συλλογή δεδομένων του Gps. Και τέλος η SD.h επιτρέπει την εγγραφή των δεδομένων σε μια SD κάρτα.

Ο κώδικας επισυνάπτεται στην ενότητα κώδικας κατασκευής Arduino στο τέλος της εργασίας με σχόλια.

5.1 Διαδικασία κατασκευής

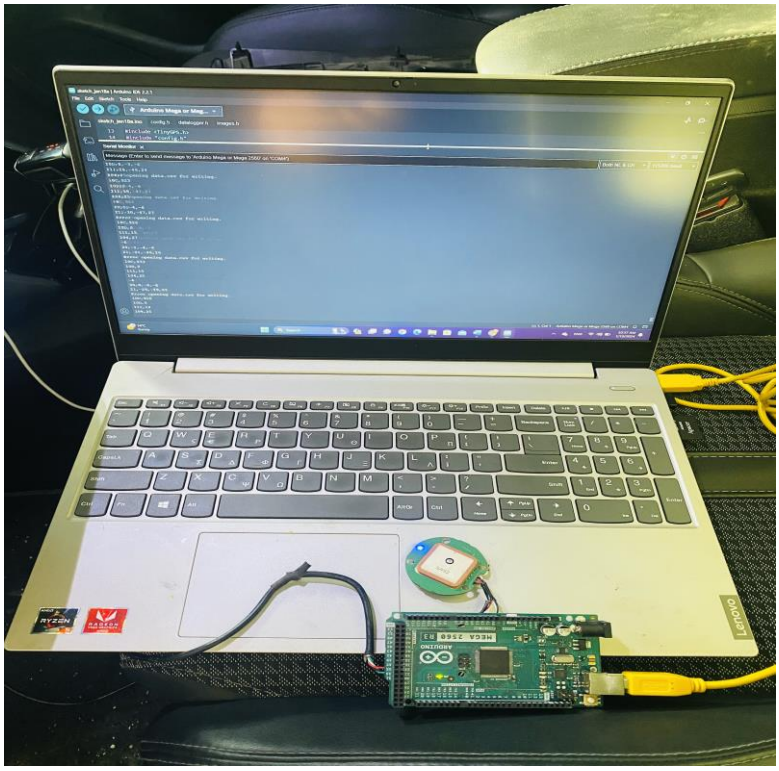
Αρχικά έγινε η συγκόλληση των module GPS και OBDII Uart adapter βάσει των επαφών που συνδέεται το κάθε module.



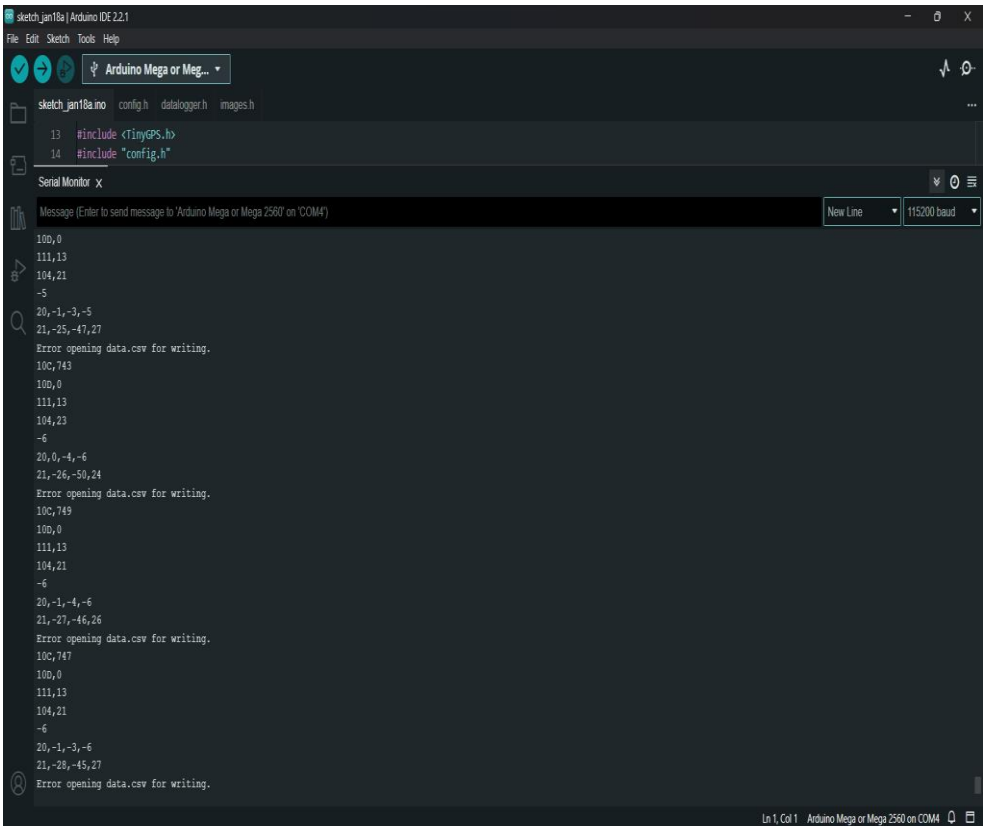
Εικόνα 20: Συγκόλληση module GPS και OBDII Uart adapter

Στη συνέχεια έπρεπε να γίνει σύνδεση με το όχημα προκειμένου να φανεί αν υπάρχει επικοινωνία με τον εγκέφαλο του κινητήρα. Εφόσον η επικοινωνία ήταν επιτυχής, πρέπει να ελεγχθούν τα δεδομένα που μπορούν να εξαχθούν βάσει των εντολών που δίνονται μέσω του κώδικα, καθώς και αν οι πληροφορίες που εμφανίζονται είναι σωστές.

Στην πρώτη φάση δεν χρειάζεται η χρήση της οθόνης γιατί το Arduino IDE δίνει την δυνατότητα χρήσης serial monitor, ενός παραθύρου που επιτρέπει την αποστολή αλλά και την εμφάνιση δεδομένων τα οποία επεξεργάζεται η πλακέτα Arduino. Το Serial Monitor είναι χρήσιμο εργαλείο για τον προγραμματιστή κατά τη διάρκεια της ανάπτυξης κώδικα και βοηθά στο debugging, κατά τον προγραμματισμό για την επίλυση σφαλμάτων.



Εικόνα 21: Προσπάθεια διασύνδεσης με το όχημα



Εικόνα 22: Serial monitor communication

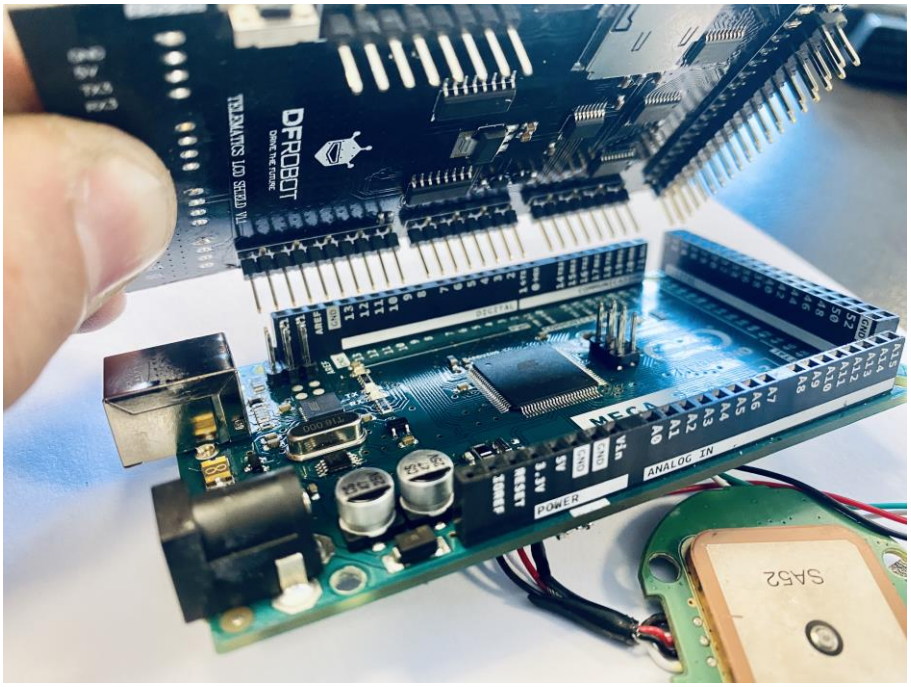
Οι εντολές που μπορούν να δοθούν για την ανάκτηση δεδομένων έχουν την παρακάτω μορφή:

- setBaudRate – set adapter serial baudrate
- readPID – read specified OBD-II PID and return parsed value
- getVoltage – measure car battery voltage
- readAccel – read accelerometer X/Y/Z values
- memsInit – initialize motion sensor
- memsRead – read motion sensor data

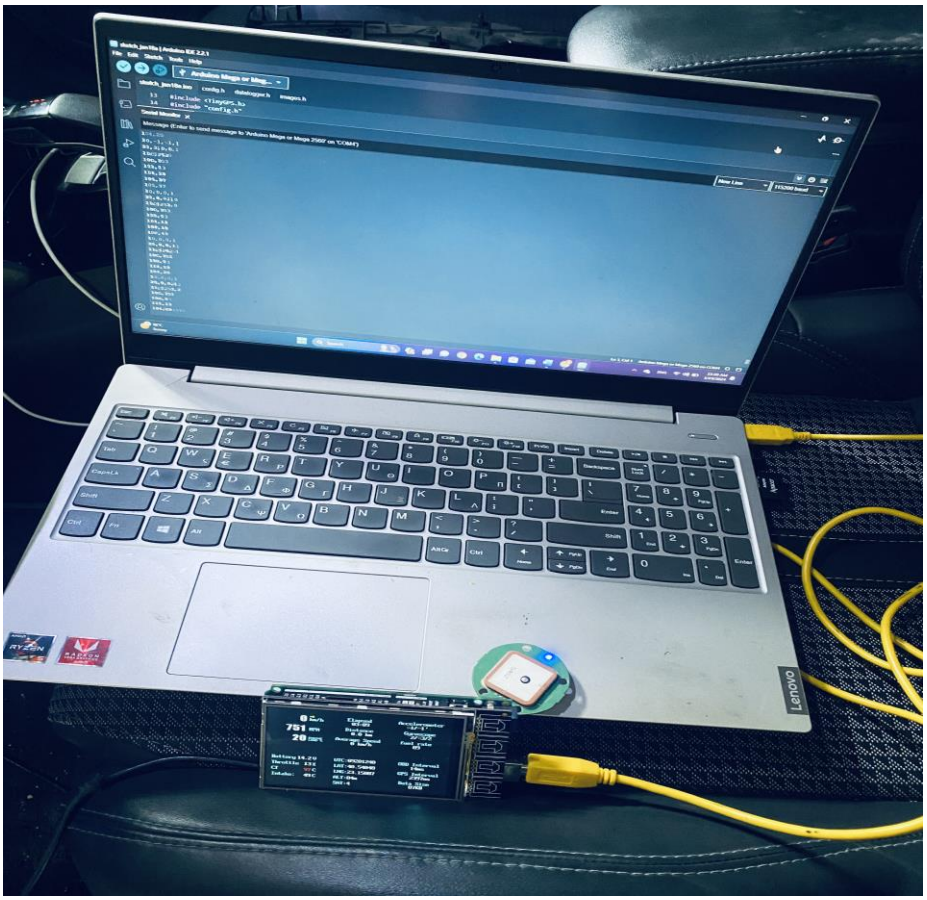
Όπως φαίνεται και στην (*Εικόνα 21)* ο τρόπος εξαγωγής δεδομένων είναι κωδικοποιημένος -όπως αναφέρθηκε και στην ενότητα « 2.7 Ταυτοποίηση παραμέτρων (PID)»- έχει τη μορφή string και σε κάθε επανάληψη στην πρώτη σειρά υπάρχει το δεδομένο των στροφών του κινητήρα που εμφανίζεται με την ακόλουθη μορφή PID 010C - Engine RPM (Στροφές κινητήρα ανά λεπτό). Το ίδιο ισχύει για όλα τα δεδομένα που εξάγονται από την θύρα OBD και στη συνέχεια ο κώδικας τα μεταφράζει σε πληροφορίες και τα αποθηκεύει σε μια κατανοητή μορφή ως προς το χρήστη.

Έπειτα εφόσον η διασύνδεση ήταν επιτυχής και η πλακέτα Arduino διαβάζει τα δεδομένα, έγινε σύνδεση της οθόνης LCD. Για να μπορέσει να λειτουργήσει η οθόνη, εκτός από την χρήση της βιβλιοθήκης ώστε να ξέρει η πλακέτα Arduino σε ποιες εξόδους να δώσει εντολές, τα βασικά στοιχεία που να δηλωθήκαν στον κώδικα είναι:

- Καθορισμός του Περιεχομένου: Τι πληροφορίες θέλουμε από το σύστημα παρουσιάζει π.χ. (στροφές κινητήρα, φορτίο κινητήρα, θερμοκρασία νερού κινητήρα).
- Διαμόρφωση των Γραφικών Στοιχείων: Τι γραφικά στοιχεία θα χρησιμοποιηθούν όπως κείμενο, γραφικές εικόνες, χρώματα, γραμμές κ.λπ.
- Οργάνωση της Σελίδας: οργάνωση των πληροφορίων στην οθόνη. Πού θα βρίσκεται το κείμενο, πού θα εμφανίζονται τα γραφικά στοιχεία και πώς θα διαχειρίζονται τον χώρο.
- Προσαρμογή του Κώδικα: Προσαρμογές που δίνουν κάποιες ιδιότητες στις μεταβαλλόμενες τιμές.



Εικόνα 23: Τοποθέτηση οθόνης στην πλακέτα Arduino



Εικόνα 24: Δεδομένα στην οθόνη στην πλακέτα Arduino

Για να έχει το σύστημα σωστά δεδομένα από το GPS module πρέπει να ρυθμίσουμε κάποιες παραμέτρους στον κώδικα οι οποίες είναι:

- Baud Rate (Ρυθμός Μεταβίβασης Δεδομένων): Στην κατασκευή έχει οριστεί στα 115200 bps.
- Διαμόρφωση του GPS Module: Με την διαμόρφωση νοείται η παραμετροποίηση της ακρίβειας του GPS , του χρόνου ανανέωσης και αποστολής δεδομένων και η αναγνώριση δορυφόρων .

Όσον αφορά την αποθήκευση δεδομένων, αναπτύχθηκε κώδικας με τον οποίο αρχικά γίνεται έλεγχος του συστήματος για χρήση SD κάρτας. Εφόσον η απάντηση στο σύστημα είναι θετική, τότε δημιουργείται ένα αρχείο σε CSV μορφή όπου οι τιμές αποθηκεύονται σε στήλες και κάθε στήλη έχει το δικό της όνομα, το οποίο αντιστοιχεί στη μέτρηση που αποθηκεύει. Η τιμή που αποθηκεύεται είναι αποκωδικοποιημένη και δεν έχει πλέον την μορφή παραμέτρου PID, ως αποτέλεσμα είναι πιο κατανοητή σε οποιονδήποτε χρήστη.

Παρακάτω παρουσιάζονται τρία διαφορετικά datalog από τρία διαφορετικά οχήματα:

	A	B	C	D	E	F	G	H	I	J	K	L
2	Time	Latitude (deg)	Longitude (de	Vehicle sp	Engine coolant ter	Intake manifold a	Engine RPM (RPM)	Intake air temperature (°C)	Relative throttle position (%)	Accel X (m/s²)	Accel Y (m/s²)	Accel Z (m/s²)
3	01/17/2024 01:3	40.53402	23.04496	0	0	0	0	0	0	1.966058	5.576111	3.116539
4	01/17/2024 01:3	40.53402	23.04496	0	0	0	0	0	0	-0.1842092	0.2400488	0.5682814
5	01/17/2024 01:3	40.53402	23.04496	0	61	101	771.75	39	49.80392	-0.2108219	-0.06551377	0.5201316
6	01/17/2024 01:3	40.53402	23.04496	0	61	101	770	39	49.80392	-0.518745	2.270092	2.03004
7	01/17/2024 01:3	40.53402	23.04496	0	61	101	771	39	49.80392	-0.085086	-0.5623164	0.9003299
8	01/17/2024 01:3	40.534	23.04499	0	61	101	769	39	49.80392	-1.268349	-1.73096	-1.348336
9	01/17/2024 01:3	40.534	23.04499	0	61	101	767.25	39	49.80392	0.1277458	0.6554297	-1.148927
10	01/17/2024 01:3	40.53398	23.045	0	61	101	766.75	39	49.80392	0.05253525	0.001588301	-0.3910956
11	01/17/2024 01:3	40.53398	23.04499	0	61	101	771.5	39	49.80392	0.1748735	0.1332916	-0.3631249
12	01/17/2024 01:3	40.53398	23.04499	0	61	101	770.5	39	49.80392	0.6635866	-0.1922753	0.307981
13	01/17/2024 01:3	40.53399	23.04499	0	61	101	763.75	39	49.80392	0.4039765	-0.6005709	0.4237258
14	01/17/2024 01:3	40.53399	23.04497	1	61	105	906.75	39	49.80392	-0.3418587	-0.6687555	0.5113088
15	01/17/2024 01:3	40.53399	23.04495	3	61	105	923.5	39	99.60784	-0.1127103	0.2271414	0.8086876
16	01/17/2024 01:3	40.534	23.04493	4	61	104	875	39	99.60784	-0.2718412	0.5612798	-0.301996
17	01/17/2024 01:3	40.53402	23.04491	4	61	104	855.25	39	99.60784	0.0987947	0.2334285	-0.6563938
18	01/17/2024 01:3	40.53402	23.04491	3	61	103	840.75	40	100	0.5210926	0.07792876	-0.1558455
19	01/17/2024 01:3	40.53402	23.04491	0	61	101	823	40	100	0.3135414	0.1858709	0.04141208
20	01/17/2024 01:3	40.53402	23.04491	0	61	100	768.5	40	100	-0.09163505	-0.1986495	0.1413261
21	01/17/2024 01:3	40.53402	23.04491	0	61	100	765.5	40	100	0.06061552	0.5553061	0.07310412
22	01/17/2024 01:3	40.53402	23.04493	0	61	100	759.25	40	100	-0.9079888	9.364595E-05	-0.7874948
23	01/17/2024 01:3	40.53402	23.04492	0	61	100	984.25	40	100	-0.2228317	-0.2421255	0.759236
24	01/17/2024 01:3	40.534	23.04492	3	61	102	922	40	100	0.2252029	1.174839	-0.8764469
25	01/17/2024 01:3	40.53398	23.04489	7	61	103	1371.25	40	100	-0.5900253	0.9025254	-0.2937401
26	01/17/2024 01:3	40.53397	23.04485	10	61	105	1723.5	39	100	-0.7492921	-0.1567671	-0.4326211
27	01/17/2024 01:3	40.53398	23.0448	12	61	102	1744.5	39	100	0.1294956	-0.3089534	0.3287428
28	01/17/2024 01:3	40.53398	23.0448	12	61	101	1163	39	100	0.4997096	-1.348955	-0.4283241

Εικόνα 25: Datalog οχήματος Mercedes b-class w246

	A	B	C	D	E	F	G	H	I	J	K	L
1	Time	Latitude (deg)	Longitude (deg)	Vehicle speed (km/h)	Intake manifold absc	Engine RPM (RP	Intake air temperature (°C)	Relative throttle position (%)		Accel X (m/s²)	Accel Y (m/s²)	Accel Z (m/s²)
2	01/12/2024 06:46	40.51	22.98015	0	Engine coolant temperature (°C)		0	0		0.3796884	1.695745	2.787754
3	01/12/2024 06:46	40.50973	22.97979	75	0	0	0	0		0.15234	0.06788442	-0.1678499
4	01/12/2024 06:46	40.50947	22.97945	73	0	0	0	39		0	-0.137633	-1.319342
5	01/12/2024 06:46	40.50922	22.9791	72	85	43	2646	39		0	0.7375544	0.08591041
6	01/12/2024 06:46	40.50908	22.97892	72	85	80	2655	39		0	-0.3659647	0.5146592
7	01/12/2024 06:46	40.50893	22.97871	75	85	114	2745	39		0	-0.1470674	-0.09676102
8	01/12/2024 06:46	40.50864	22.97832	78	85	113	2854	39		0	-0.332996	-0.0214281
9	01/12/2024 06:46	40.50849	22.97812	80	85	122	2937	39		0	-0.5019575	0.9757788
10	01/12/2024 06:46	40.50834	22.97792	81	85	103	2996	38		0	-1.31197	-2.157439
11	01/12/2024 06:46	40.50805	22.97754	79	85	35	2944	38		0	-0.3414781	-0.8498416
12	01/12/2024 06:46	40.50791	22.97735	79	85	61	2907	38		0	0.06020181	0.02608945
13	01/12/2024 06:46	40.50763	22.97697	79	85	94	2940	38		0	-0.4005418	0.4414253
14	01/12/2024 06:46	40.5075	22.9768	76	85	22	2863	38		0	0.7609835	0.9162788
15	01/12/2024 06:46	40.50736	22.97663	74	85	22	2757	38		0	0.2193545	0.2215926
16	01/12/2024 06:46	40.50706	22.97632	74	85	66	2731	38		0	0.1949244	0.1184411
17	01/12/2024 06:46	40.50691	22.97613	76	85	121	2804	38		0	0.8494585	-0.1368242
18	01/12/2024 06:46	40.50659	22.97573	80	85	158	2943	39		0	0.7005984	0.3996307
19	01/12/2024 06:47	40.50645	22.97553	82	85	150	3073	39		0	-0.2083215	0.3109969
20	01/12/2024 06:47	40.5063	22.97533	81	85	46	3029	39		0	0.5852169	0.4178012
21	01/12/2024 06:47	40.50603	22.97492	81	85	50	2981	39		0	-0.1747746	-0.3330204
22	01/12/2024 06:47	40.5059	22.97471	81	85	80	3000	38		0	-0.7532729	-0.3432076
23	01/12/2024 06:47	40.50575	22.97453	81	85	65	3016	38		0	-2.290354	-0.7313457
24	01/12/2024 06:47	40.50562	22.97432	80	85	36	2942	38		0	-0.2111774	0.298582
25	01/12/2024 06:47	40.50537	22.97391	78	85	27	2904	38		0	-0.3353133	-0.3147191
26	01/12/2024 06:47	40.50513	22.9735	78	85	56	2846	38		0	0.338221	0.3964324
27	01/12/2024 06:47	40.505	22.9733	78	85	76	2866	38		0	0.4839686	0.4140635

Εικόνα 26: Datalog οχήματος Smart fortwo (450)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Time	Latitude (deg)	Longitude (de	Vehicle speed (km/h)	Engine coolant tem	Intake manifold absolu	Engine RPM (RPM)	Intake air temperature	Relative throttle position (%)	Accel X (m	Accel Y (m	Accel Z (m	Accel (Gra
2	12/27/2023 05:2	40.53407	23.04489	0	0	0	0	0	0	-0.97439	-0.80375	6.342048	1.488505
3	12/27/2023 05:2	40.53407	23.04489	0	0	0	0	0	0	-0.46008	-0.05671	1.898209	1.719025
4	12/27/2023 05:2	40.53407	23.04489	0	87	104	801	31	4.705883	0.052659	0.055671	0.761822	1.293909
5	12/27/2023 05:2	40.53408	23.04489	0	87	104	799	31	4.705883	-0.33874	0.064539	0.291264	1.646875
6	12/27/2023 05:2	40.53408	23.04489	0	87	104	800	31	4.705883	-0.14763	0.187047	0.040938	1.643432
7	12/27/2023 05:2	40.53409	23.0449	0	87	104	800	31	4.705883	-0.06945	0.190062	0.209103	1.655557
8	12/27/2023 05:2	40.5341	23.04489	0	87	104	800	31	4.705883	-0.06227	0.042705	-0.12481	1.705703
9	12/27/2023 05:2	40.5341	23.04489	0	87	104	799.5	31	4.705883	-0.03434	-0.02339	-0.03048	1.704356
10	12/27/2023 05:2	40.5341	23.04489	0	87	104	800	31	4.705883	-0.03313	-0.03084	0.129149	1.723666
11	12/27/2023 05:2	40.5341	23.04489	0	87	104	800.5	31	4.705883	-0.01554	-0.02178	0.057046	1.710343
12	12/27/2023 05:2	40.5341	23.04489	0	87	104	799.5	31	4.705883	0.021001	-0.00563	-0.10232	1.669628
13	12/27/2023 05:2	40.5341	23.04489	0	87	104	799.5	31	4.705883	-0.12161	-0.0697	0.195483	1.737587
14	12/27/2023 05:2	40.5341	23.04489	0	87	104	800	31	4.705883	0.072537	0.03379	0.096995	1.603615
15	12/27/2023 05:2	40.5341	23.04489	0	87	104	800	31	4.705883	0.057131	0.134276	0.099658	1.574725
16	12/27/2023 05:2	40.53416	23.0449	0	87	104	799.5	31	4.705883	-0.01262	0.124051	-0.00073	1.621728
17	12/27/2023 05:2	40.53416	23.0449	0	87	104	800.5	31	4.705883	0.55075	-0.08739	-1.78152	0.998124
18	12/27/2023 05:2	40.53417	23.0449	0	87	104	799	31	4.705883	0.00888	0.293419	-0.01186	1.456621
19	12/27/2023 05:2	40.53416	23.04491	0	87	104	799	31	4.705883	-0.04318	-0.01712	-0.01802	1.443149
20	12/27/2023 05:2	40.53416	23.04491	0	87	104	800	31	4.705883	-0.05821	0.04659	-0.1395	1.387764
21	12/27/2023 05:2	40.53416	23.04491	0	87	104	800	31	4.705883	0.041781	0.181017	0.209434	1.274599
22	12/27/2023 05:2	40.53416	23.04491	0	87	104	799.5	31	4.705883	0.192954	0.287202	0.508119	1.111738
23	12/27/2023 05:2	40.53416	23.04491	0	87	104	800.5	31	4.705883	0.05773	0.070604	-0.00938	1.248105
24	12/27/2023 05:2	40.53416	23.04491	0	87	104	800	31	4.705883	0.03483	0.031877	0.085101	1.27894
25	12/27/2023 05:2	40.53416	23.04492	0	87	104	800.5	31	4.705883	-0.05245	-0.02821	-0.00187	1.380878
26	12/27/2023 05:2	40.53416	23.04492	0	87	104	800	31	4.705883	-0.01242	-0.04528	-0.10486	1.37055
27	12/27/2023 05:2	40.53416	23.04492	0	87	104	800	31	4.705883	-0.02246	0.013512	-0.09178	1.414259

Εικόνα 27: Datalog οχήματος Fiat ducato 2017

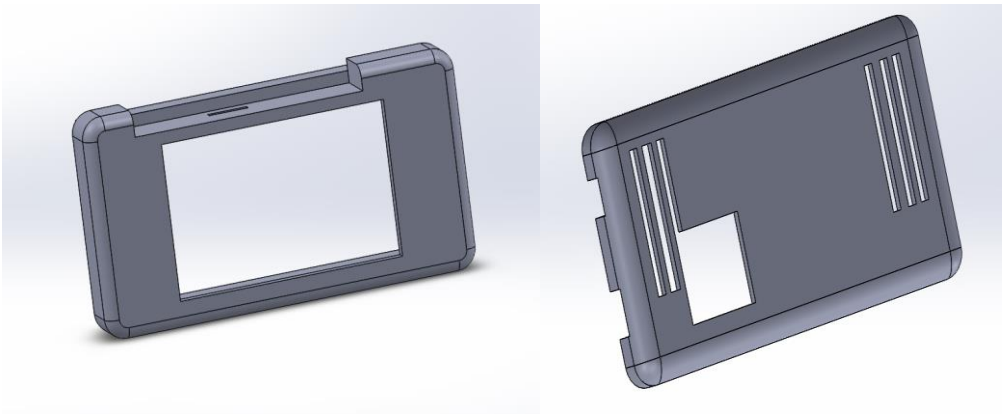
Στη συνέχεια της εργασίας παρουσιάζεται η δημιουργία του προγράμματος για την ανάλυση και οπτικοποίηση δεδομένων σε μορφή διαγραμμάτων. Το πρόγραμμα αυτό μπορεί να ελέγξει τα δεδομένα, να εντοπίσει προβλήματα στο όχημα και να δώσει συμβουλές, καθώς επίσης να δείξει τη διαδρομή του οχήματος σε χάρτη με timestamps.

5.2 Μορφοποίηση κατασκευής με την χρήση 3d printer

Εφόσον η κατασκευή ήταν πλέον λειτουργική, για την προστασία της πλακέτας σχεδιάστηκε ένα περίβλημα με πλαστικό (PLA) με την τεχνολογία εκτύπωσης 3D. Με τον σχεδιασμό, εξασφαλίστηκε όχι μόνο η ασφαλής τοποθέτηση του Arduino Mega 2560 μέσα στο περίβλημα, αλλά και η εύκολη πρόσβαση σε όλες τις θύρες, δίνοντας ένα κομψό χαρακτήρα στην κατασκευή. Τα υλικά που χρησιμοποιήθηκαν διασφαλίζουν την ανθεκτικότητα και την προστασία από εξωτερικούς παράγοντες, ενώ ο σχεδιασμός μελετήθηκε προσεκτικά, για να ανταποκρίνεται στις ανάγκες του χρήστη.

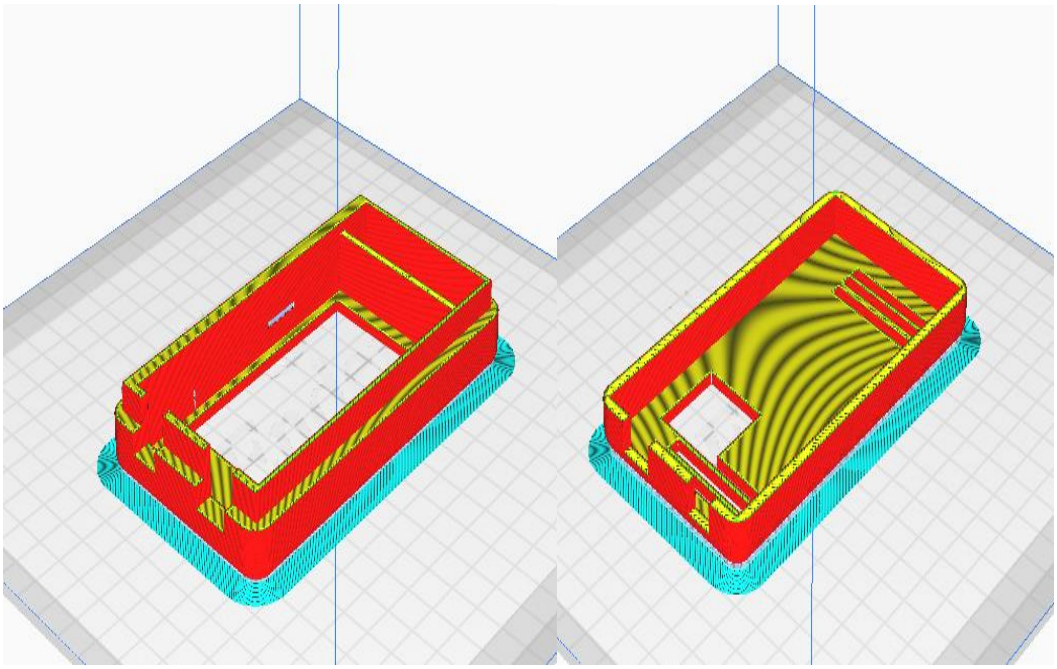
Αρχικά έπρεπε να μετρηθεί κάθε διάσταση της πλακέτας ώστε να γίνει σχεδιασμός του κατάλληλου περιβλήματος, που θα εξυπηρετούσε όλες τις ανάγκες της πλακέτας π.χ. (επαφές σύνδεσης σε εμφανή σημεία, εξαγωγή και επανατοποθέτηση της micro SD card).

Με την χρήση προγράμματος σχεδιασμού CAD δημιουργήθηκε το τελικό μοντέλο όπως φαίνεται στις παρακάτω φωτογραφίες:



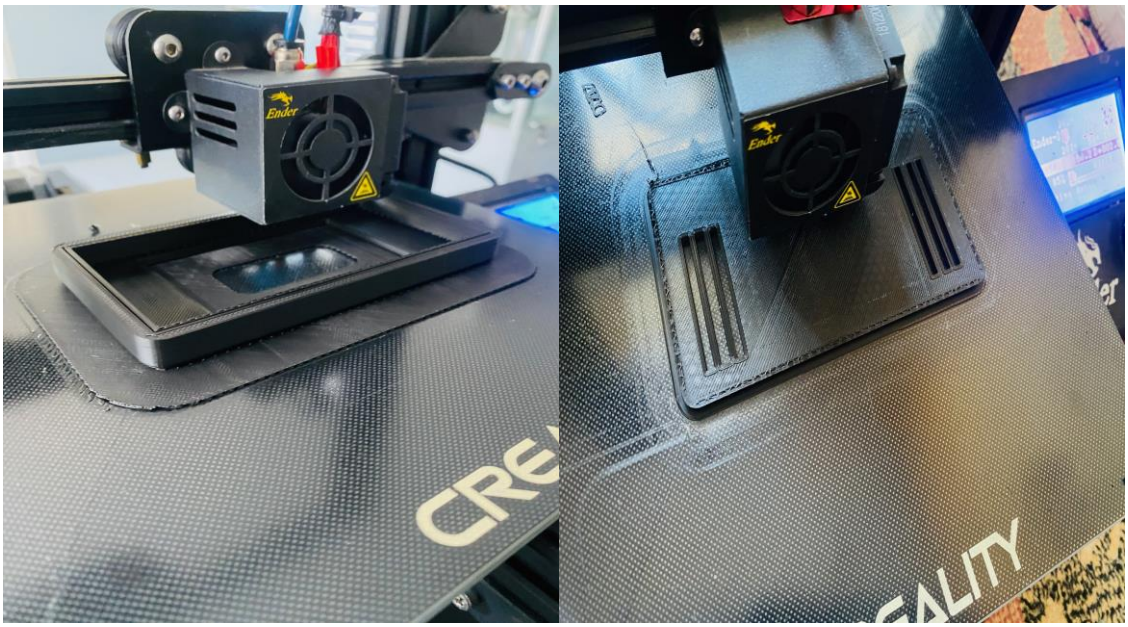
Εικόνα 28: Σχεδιασμός περιβλήματος CAD

Τα σχέδια πέρασαν από επεξεργασία ώστε να μπορούν να εκτυπωθούν. Κατά την επεξεργασία χρησιμοποιήθηκε ένα πρόγραμμα για slicing, δηλαδή διαίρεση του αντικειμένου εκτύπωσης σε μικρές λεπτές οριζόντιες στρώσεις. Σημαντικό στοιχείο για την ολοκλήρωση της εκτύπωσης αποτελεί η ρύθμιση των παραμέτρων εκτύπωσης όπως είναι α) η ταχύτητα εκτύπωσης, β) η πυκνότητα του αντικειμένου αλλά και γ) η θερμοκρασία εκτύπωσης. Παρακάτω βλέπουμε το μοντέλο σε διαδικασία slicing στο πρόγραμμα επεξεργασίας:



Εικόνα 29: Επεξεργασία μοντέλου 3d εκτύπωσης(slicing)

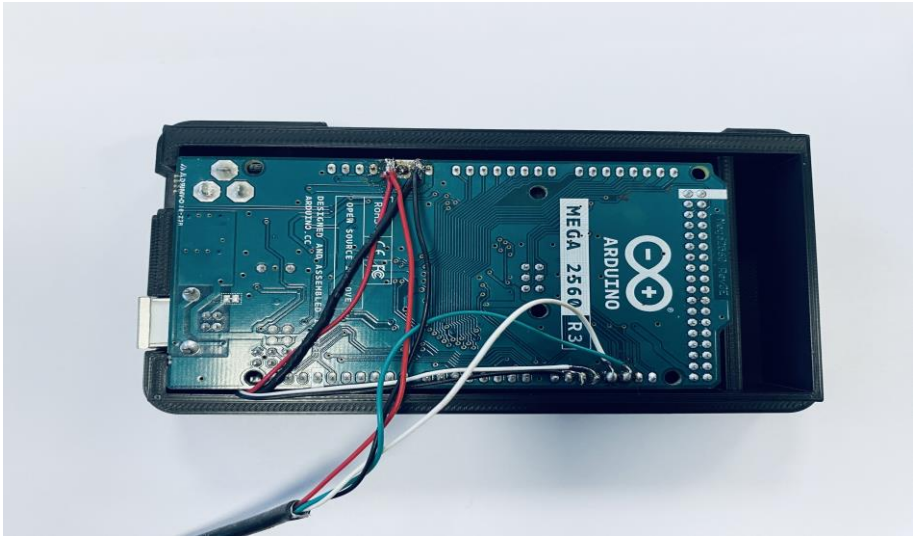
Το τελικό αποτέλεσμα όπως φαίνεται παρακάτω:



Εικόνα 30: 3d εκτύπωση

5.3 Φωτογραφίες τελικού αποτελέσματος κατασκευής

Εικόνα 31: Τελικό αποτέλεσμα



Εικόνα 32: Εφαρμογή πλακέτας στο περίβλημα

Εικόνα 33: Εφαρμογή κατασκευής πάνω στο όχημα με την χρήση μαγνητικής βάσης

5.4 Προσωπική συνεισφορά στην κατασκευή

Λαμβάνοντας υπόψη τις απαιτήσεις της κατασκευής και έπειτα από έρευνα αγοράς έγινε προσεκτική επιλογή υλικών και με τη χρήση εργαλείων όπως π.χ (ηλεκτρικό κολλητήρι). Έγιναν οι απαραίτητες συνδέσεις ώστε να υπάρχει διασύνδεση αναμεσά στην κεντρική πλακέτα και τα εξαρτήματα, ακολουθώντας τις απαιτήσεις του κάθε εξαρτήματος. Έπειτα ακολούθησε η διαδικασία ελέγχου για να επιβεβαιωθεί η σωστή λειτουργία των Module, όπως τροφοδοσία τάσης και επικοινωνία μετά τη συγκόλληση.

Η προσωπική συνεισφορά στο κομμάτι του λογισμικού επικεντρώθηκε στη βελτιστοποίηση του τρόπου αποθήκευσης των δεδομένων, καθώς προσδιορίσθηκε ένα αδύναμο σημείο στο πρόγραμμα. Αρχικά, παρατηρήθηκε ότι η αποθήκευση των δεδομένων σε μορφή PID δημιουργούσε προβλήματα κατανόησης και δυσκολίες στον χειρισμό τους από οδηγούς και το λογισμικό. Έτσι, έγινε μεταποίηση στον τρόπο αποθήκευσης δεδομένων, χρησιμοποιώντας πιο κατανοητές στήλες με τίτλους. Αυτή η αλλαγή επέτρεψε την οργάνωση των δεδομένων σε κατηγορίες, καθιστώντας τις τιμές πιο ευανάγνωστες και προσβάσιμες. Καθώς οι τίτλοι αντικατέστησαν τους PID αριθμούς, η διαχείριση και η αντίληψη των δεδομένων από τους οδηγούς και το λογισμικό έγινε σαφέστερη. Η διαδικασία αυτή απαιτούσε τροποποιήσεις στον κώδικα, συμπεριλαμβανομένης της αναδιάρθρωσης του τρόπου αποθήκευσης των δεδομένων και της διαδικασίας ανάγνωσης από το λογισμικό. Με αυτή την παρέμβαση, επιτεύχθηκε μια ολοκληρωμένη και κατανοητή διαχείριση των πληροφοριών, εξασφαλίζοντας ομαλότερη επεξεργασία και ανταλλαγή δεδομένων μεταξύ οδηγών και λογισμικού.

Αντιμετωπίζοντας την ανάγκη να προστατευθεί η κατασκευή και να ενσωματωθούν όλα τα στοιχεία, όπως η κεραία GPS, η πλακέτα, η οθόνη και η χρήση της micro SD, έγινε ο σχεδιασμός ενός περιβλήματος. Ο σχεδιασμός αυτός πραγματοποιήθηκε με τη χρήση ενός προγράμματος CAD και το τελικό περίβλημα εκτυπώθηκε με τη χρήση της τεχνολογίας 3D printing. Κατά τη διάρκεια της διαδικασίας σχεδιασμού, λαμβάνοντας υπόψη τις διαστάσεις και τις ανάγκες της κατασκευής, δημιουργήθηκε ένα περίβλημα που θα προστατεύει αποτελεσματικά όλα τα εσωτερικά στοιχεία. Κάθε νέα εκτύπωση συνοδευόταν από μετρήσεις και διορθώσεις, εξασφαλίζοντας τη συνεχή βελτίωση του περιβλήματος. Με την τεχνολογία 3D printing, επιτεύχθηκε το τελικό περίβλημα που όχι μόνο προστατεύει την κατασκευή, αλλά επιτρέπει επίσης την ορθή τοποθέτηση και λειτουργία όλων των εξαρτημάτων.

Κατά τη διενέργεια δοκιμών με διάφορα οχήματα και διαφορετικές διαδρομές, παρατηρήθηκαν λάθη στον τρόπο καταγραφής δεδομένων, αλλά και οπτικοποίησης αυτών στην οθόνη, που προήλθαν από εσφαλμένες παραμέτρους στο πρόγραμμα. Για να αντιμετωπιστούν αυτά τα προβλήματα, πραγματοποιήθηκαν πολλές επαναλήψεις, ύστερα από διορθώσεις στον κώδικα και δοκιμές για επαλήθευση. Στόχος ήταν να εξαλειφθούν τα προβλήματα που προκαλούνταν από εσφαλμένες

παραμέτρους, ενισχύοντας έτσι την αξιοπιστία και την ακρίβεια των δεδομένων που καταγράφονταν. Μέσω συστηματικής διαδικασίας ελέγχου και διόρθωσης, επετεύχθη η ομαλή λειτουργία του προγράμματος και η αξιόπιστη καταγραφή των δεδομένων κατά τις δοκιμές.

6 Κατασκευή λογισμικού ανάλυσης δεδομένων (CarAnalysis)

6.1 Στόχος προγράμματος

Ο στόχος του προγράμματος CarAnalysis είναι να παρέχει μια ολοκληρωμένη λύση για τον έλεγχο, την οπτικοποίηση και την ανάλυση δεδομένων οχημάτων από ένα αρχείο CSV. Με βάση τα δεδομένα αυτά, το πρόγραμμα επιτρέπει στο χρήστη να επιλέγει και να οπτικοποιεί συγκεκριμένα δεδομένα, προσφέροντας έναν κατανοητό και απλό τρόπο ανάλυσης. Παράλληλα, παρέχει στον χρήστη ένα εργαλείο που όχι μόνον οπτικοποιεί τα τεχνικά δεδομένα του οχήματός του, αλλά επίσης τον ενημερώνει για πιθανά προβλήματα και του παρέχει συγκεκριμένες συμβουλές βελτίωσης.

Το πρόγραμμα επιτυγχάνει έγκαιρη διαχείριση των προβλημάτων του οχήματος, συνεισφέροντας ταυτόχρονα στην ενημέρωση και εκπαίδευση του οδηγού για την βελτίωση της οδηγικής του συμπεριφοράς.

6.2 Γλώσσα προγραμματισμού

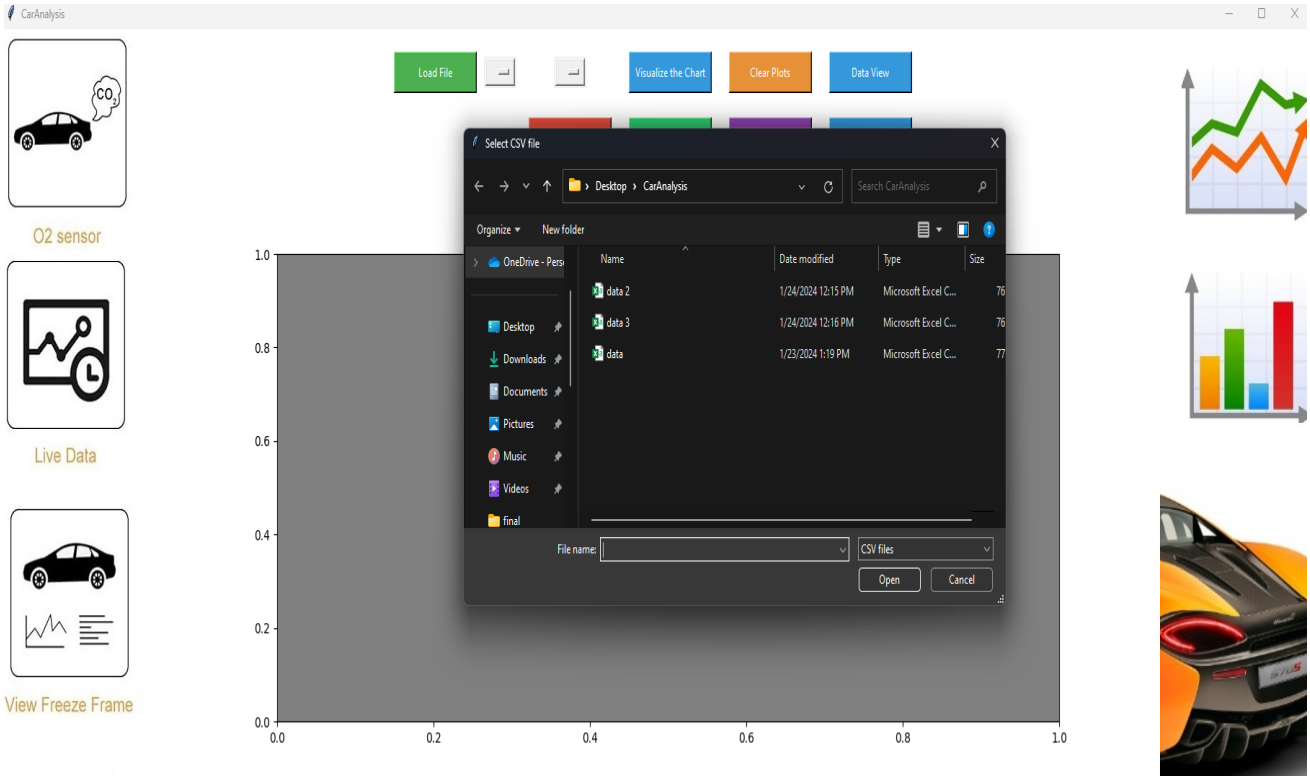
Για την ανάπτυξη του λογισμικού CarAnalysis, επιλέχθηκε η γλώσσα προγραμματισμού Python λόγω της ευανάγνωστης σύνταξής της, η οποία διευκολύνει την ανάπτυξη και συντήρηση του κώδικα, καθιστώντας τον προγραμματισμό προσιτό και κατανοητό. Παράλληλα επιτρέπει την αποτελεσματική επεξεργασία και οπτικοποίηση δεδομένων, μέσω των βιβλιοθηκών που διαθέτει όπως τα Pandas και το Matplotlib, βοηθώντας έτσι στην επίτευξη των στόχων του έργου.

Ειδικότερα, δημιουργήθηκε ένα Γραφικό Περιβάλλον Χρήστη (GUI), το οποίο λειτουργεί ως κεντρική διασύνδεση μεταξύ του χρήστη και του προγράμματος. Το GUI προσφέρει εύκολη πλοήγηση και επιλογή δεδομένων από αρχεία CSV.

Εικόνα 34:Python symbol

6.3 Βασικές λειτουργίες προγράμματος

1. Πρώτο βήμα είναι η επιλογή **LOAD FILE** για την εισαγωγή αρχείου CSV.

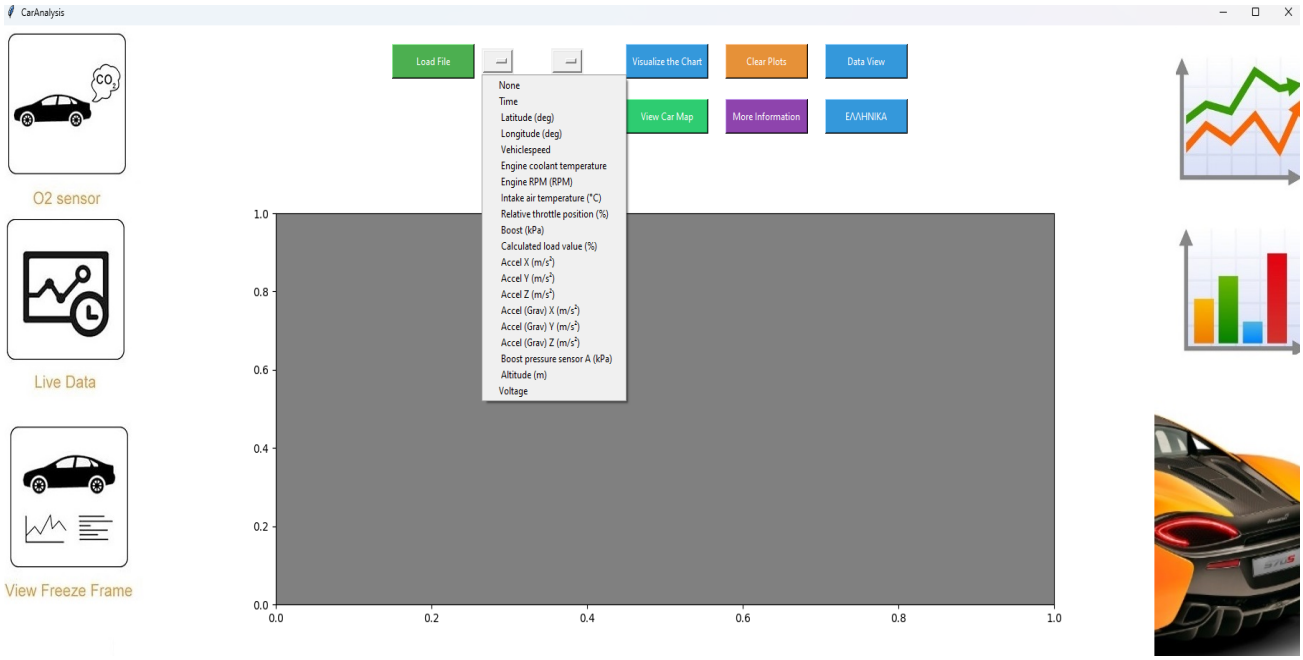


Εικόνα 35:Load file

Λειτουργία επιλογής CSV αρχείου:

```
# Load CSV Button
    self.load_button = tk.Button(self.frame, text="Load File", command=self.load_csv, bg="#4CAF50", fg="white",width=15, height=2)
    self.load_button.grid(row=0, column=3, padx=5, pady=5)



# Csv file loader
     def load_csv(self):
    file_path = filedialog.askopenfilename(title="Select CSV file", filetypes=[("CSV files", "*.csv")])
    if file_path:
        self.df = pd.read_csv(file_path)
        columns = self.df.columns
```

2. Δεύτερο βήμα είναι η επιλογή των δεδομένων που θέλουμε να χρησιμοποιήσουμε με την χρήση του **drop down menu**. Μπορούμε να επιλέξουμε ένα δεδομένο και είτε να αναλύσουμε μόνο αυτό, είτε να το συγκρίνουμε με κάποιο άλλο δεδομένο ταυτόχρονα.

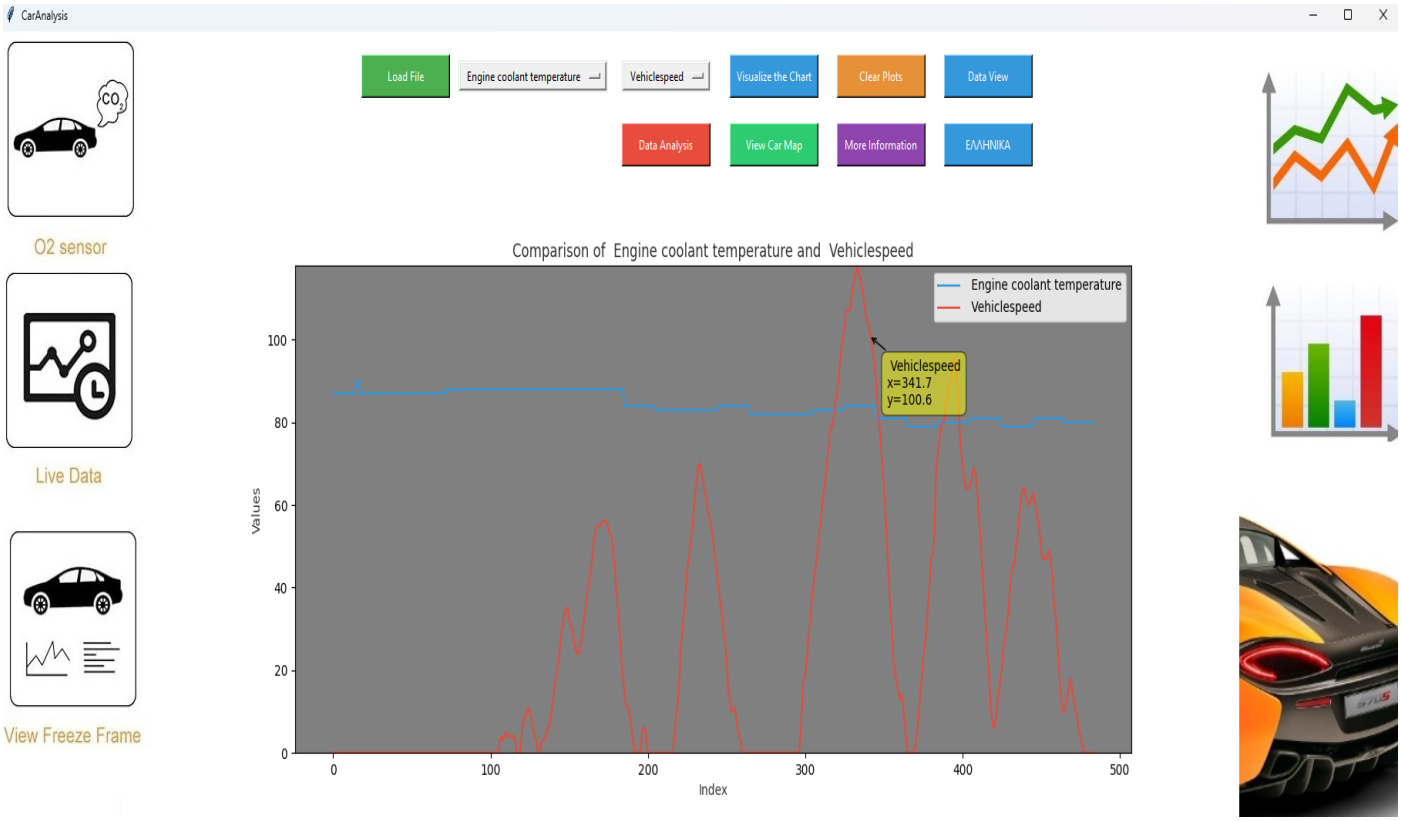


Εικόνα 36:Drop down menu

Λειτουργία χρήσης drop down menu:

```
# Dropdown for selecting the first column
        self.column_var1 = tk.StringVar()
        self.column_dropdown1 = tk.OptionMenu(self.frame, self.column_var1, "")
        self.column_dropdown1.grid(row=0, column=4, padx=5, pady=5)

        # Dropdown for selecting the second column
        self.column_var2 = tk.StringVar()
        self.column_dropdown2 = tk.OptionMenu(self.frame, self.column_var2, "")
        self.column_dropdown2.grid(row=0, column=5, padx=12, pady=12)

# Clear previous options
            self.column_dropdown1['menu'].delete(0, 'end')
            self.column_dropdown2['menu'].delete(0, 'end')

            # Add "None" option to both dropdowns
            self.column_dropdown1['menu'].add_command(label="None",          command=tk._setit(self.column_var1, ""))
            self.column_dropdown2['menu'].add_command(label="None", command=tk._setit(self.column_var2, ""))

            # Add columns to the dropdowns
            for column in columns:
                self.column_dropdown1['menu'].add_command(label=column, command=tk._setit(self.column_var1, column))
                self.column_dropdown2['menu'].add_command(label=column, command=tk._setit(self.column_var2, column))

            # Set the default selection to "None"
                self.column_var1.set("")
                self.column_var2.set("")
```

3. Τρίτο βήμα είναι η επιλογή οπτικοποίησης των δεδομένων με το κουμπί **Visualize the Chart** και η εκκαθάριση του επιλεγμένου διαγράμματος με το κουμπί **Clear Plots**.



Εικόνα 37:Data visualization

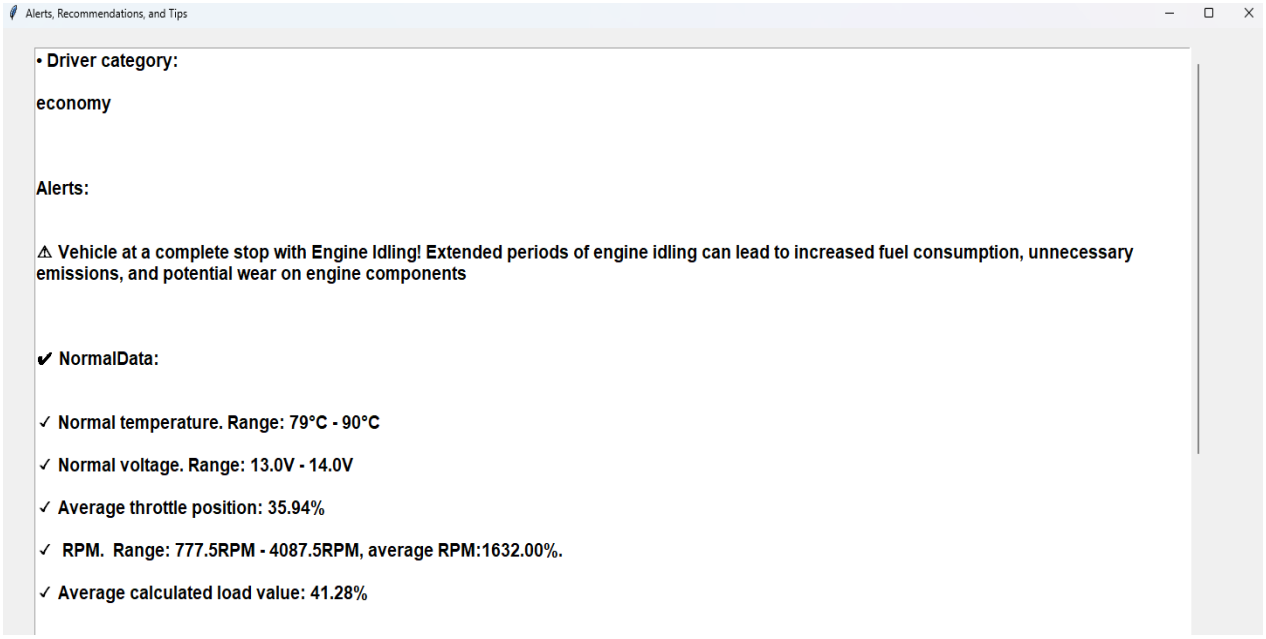
Λειτουργία οπτικοποίησης δεδομένων:

```
        # Visualize Button
        self.visualize_button = tk.Button(self.frame, text="Visualize the Chart", command=self.visualize_data, bg="#3498db",
fg="white",width=15, height=2)
        self.visualize_button.grid(row=0, column=6, padx=12, pady=12)

        # Clear Button
        self.clear_button = tk.Button(self.frame, text="Clear Plots", command=self.clear_plots, bg="#E69138", fg="white",width=15, height=2)
        self.clear_button.grid(row=0, column=7, padx=12, pady=12)

        # Data visualisation button
    def visualize_data(self):
        if self.df is not None:
            self.selected_column1 = self.column_var1.get()
            self.selected_column2 = self.column_var2.get()

            self.ax1.clear()

            if self.selected_column1:
                self.ax1.plot(self.df.index, self.df[self.selected_column1], label=self.selected_column1, color="#3498db")

            if self.selected_column2:
                self.ax1.plot(self.df.index, self.df[self.selected_column2], label=self.selected_column2, color="#e74c3c")

            if self.selected_column1 or self.selected_column2:
                self.ax1.set_title(f"Comparison of {self.selected_column1} and {self.selected_column2}", color="#333333")
                self.ax1.set_xlabel("Index", color="#333333")
                self.ax1.set_ylabel("Values", color="#333333")
                self.ax1.legend()

                # Adjust y-axis limits based on the selected columns
                if self.selected_column1 and self.selected_column2:
                    min_value = min(self.df[self.selected_column1].min(), self.df[self.selected_column2].min())
                    max_value = max(self.df[self.selected_column1].max(), self.df[self.selected_column2].max())
                elif self.selected_column1:
                    min_value, max_value = self.df[self.selected_column1].min(), self.df[self.selected_column1].max()
                elif self.selected_column2:
                    min_value, max_value = self.df[self.selected_column2].min(), self.df[self.selected_column2].max()
                else:
                    min_value, max_value = 0, 1  # Default values if no column is selected

                self.ax1.set_ylim(min_value, max_value)

                self.canvas1.draw()
```

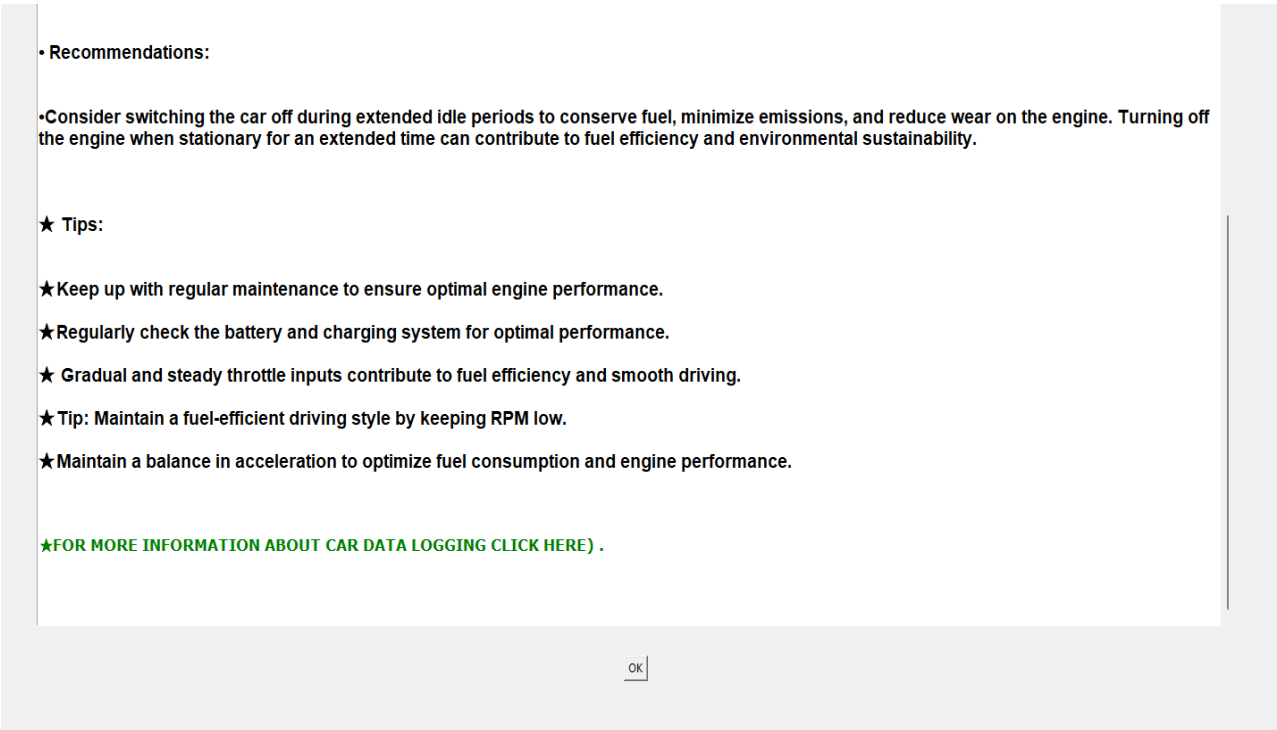
4. Τέταρτο βήμα είναι η ανάλυση δεδομένων με το κουμπί **Data analysis**. Με την χρήση αυτής της λειτουργίας το πρόγραμμα κατηγοριοποιεί τον χρήστη σε μια κατηγορία οδηγού «Aggressive ή «Economy-friendly». Έπειτα βάσει κάποιων δομών ελέγχου εμφανίζει τα προβλήματα που βρίσκει στο αρχείο, πληροφορίες για τον τρόπο διαχείρισης αυτών και τέλος, δίνει συμβουλές για την βελτίωση και την ενημέρωση του οδηγού.

5. Παράδειγμα δομής ελέγχου: με βάση την καταγραφή των στροφών του κινητήρα και τις τιμές που έχουν καταγραφεί για την ταχύτητα του οχήματος, το σύστημα ελέγχει τη συχνότητα επανάληψης

των περιπτώσεων στις οποίες η ταχύτητα είναι μηδενική και οι στροφές του κινητήρα κάτω από χίλιες. Εάν η συχνότητα επανάληψης υπερβαίνει κάποιο όριο, σημαίνει ότι ο οδηγός είναι για αρκετή ώρα σταματημένος με ανοιχτή τη μηχανή, πράγμα που καταπονεί τον κινητήρα αλλά και επιβαρύνει το περιβάλλον. Στην περίπτωση αυτή, το πρόγραμμα εμφανίζει ειδοποίηση (alert) εξηγώντας το πρόβλημα που βρέθηκε και δίνοντας στον οδηγό την αντίστοιχη συμβουλή (πχ. να σβήνει τον κινητήρα, όπως φαίνεται στις παρακάτω εικόνες)



Εικόνα 38:Alerts,Recommendations and Tips (1)



Εικόνα 39: Alerts,Recommendations and Tips (2)

Λειτουργία ανάλυσης δεδομένων:

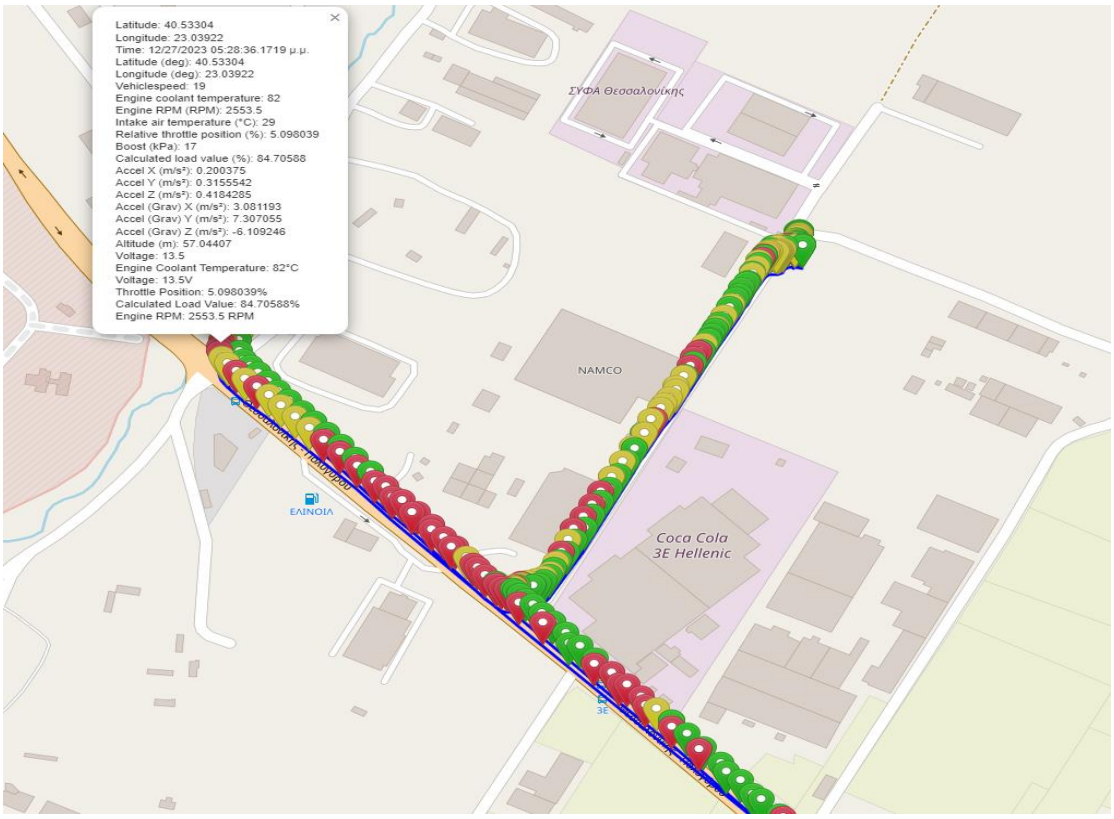
```
#Check list building

    def check_list(self):
      if self.df is not None:
        required_columns = [' Engine coolant temperature', 'Voltage', ' Relative throttle position (%)', ' Vehiclespeed', ' Calculated load value (%)', '
Engine RPM (RPM)']

        alert_symbol = "\u26A0"  # Unicode character for the alert symbol

        # Initialize variables outside the loop
        alerts = ""
        recommendations = ""
        normaldata=""
        Tips = ""
        driver_category = ""

        for column_name in required_columns:
          try:
            column_data = self.df[column_name]
          except KeyError:
```

```
            # If the column is not found, print a message and continue to the next column
            print(f"Column '{column_name}' not found. Skipping...")
            continue

        # RPM Analysis

        if column_name == ' Engine RPM (RPM)':
            max_rpm = column_data.max()
            min_rpm = column_data.min()
            average_rpm = column_data.mean()

            if average_rpm > 3800:
                alerts += f"{alert_symbol} High RPM detected! Maximum RPM recorded: {average_rpm:.2f}%\n\n"
                recommendations += "\u2022Recommendation: Avoid over-revving the engine to prevent excessive wear and damage.\n\n"
                Tips += "\u2605Tip: Shift gears smoothly and avoid aggressive acceleration to keep RPM within a safe range.\n\n"
                driver_category = f"\u26A0 Aggressive"
            elif 1500< average_rpm < 3800:
                normaldata += f"\u2713  RPM.  Range: {min_rpm}RPM - {max_rpm}RPM, average RPM:{average_rpm:.2f}%.\n\n"
                Tips += "\u2605Tip: Maintain a fuel-efficient driving style by keeping RPM low.\n\n"
                driver_category = "Economy-friendly"
            else:
                normaldata += f"\u2713  RPM.average: {average_rpm:.2f}%\n\n"
                Tips += "\u2605Driving with the RPM in too low of a gear can lead to engine strain and decreased fuel efficiency.\n\n"
                driver_category = "Inexperienced"




        if column_name == ' Engine coolant temperature':
            max_temp = column_data.max()
            min_temp = column_data.min()

            # Temperature Analysis
            if max_temp > 100:
                alerts += f"{alert_symbol} Overheating problem! Maximum temperature recorded: {max_temp}°C\n\n"
                recommendations+= "\u2022Please visti a workshop, Problem in the cooling system.
\n\n"
            elif min_temp < 60:
                alerts += f"{alert_symbol} Low temperature. Minimum temperature recorded: {min_temp}°C\n\n"
                recommendations += "\u2022Consider driving at a higher speed to increase engine temperature, or check for potential issues with the
cooling system.\n\n\n\n"
            else:
                normaldata += f"\u2713 Normal temperature. Range: {min_temp}°C - {max_temp}°C\n\n"
                Tips += "\u2605Keep up with regular maintenance to ensure optimal engine performance.\n\n"

        elif column_name == 'Voltage':
            max_voltage = column_data.max()
            min_voltage = column_data.min()

            # Voltage Analysis
            if max_voltage > 15:
                alerts += f"{alert_symbol} Overcharging problem! Maximum voltage recorded: {max_voltage}V\n\n"
                recommendations += "\u2022Visit a workshop to inspect and address alternator or battery faults immediately.\n\n"
            elif min_voltage < 11:
                alerts += f"{alert_symbol} Undercharging problem! Minimum voltage recorded: {min_voltage}V\n\n"
                recommendations += "\u2022Check the alternator, battery, and charging system. Charge or replace the battery if necessary.\n\n"
```

```
            else:
                normaldata += f"\u2713 Normal voltage. Range: {min_voltage}V - {max_voltage}V\n\n"
                Tips += "\u2605Regularly check the battery and charging system for optimal performance.\n\n"

        elif column_name == ' Vehiclespeed':
            max_speed = column_data.max()
            min_speed = column_data.min()
            min_rpm = column_data.min()

            # Check for frequent idling instances
            count_idle_instances = ((self.df[' Vehiclespeed'] == 0) & (self.df[' Engine RPM (RPM)'] < 1000)).sum()
            threshold_idle_instances = 5  # Adjust as needed


            # Speed Analysis
            if max_speed > 150:
                alerts += f"{alert_symbol} High speed detected! Maximum speed recorded: {max_speed} km/h\n\n"
                recommendations += "\u2022Please drive within the recommended speed limits for your safety.\n\n"
            elif min_speed == 0 and min_rpm < 1000:
                alerts += f"{alert_symbol} Vehicle at a complete stop with Engine Idling! Extended periods of engine idling can lead to increased fuel consumption, unnecessary emissions, and potential wear on engine components\n\n"
                recommendations += "\u2022Consider switching the car off during extended idle periods to conserve fuel, minimize emissions, and reduce wear on the engine. Turning off the engine when stationary for an extended time can contribute to fuel efficiency and environmental sustainability.\n\n"
            else:
                normaldata += f"\u2713 Normal speed. Range: {min_speed} km/h - {max_speed} km/h\n\n"
                Tips += "\u2605Adhering to the normal speed limit promotes safety and reduces potential risks.\n\n"




        elif column_name == ' Relative throttle position (%)':

            average_throttle = column_data.mean()


            # Throttle Position Analysis
            if average_throttle > 80:
                alerts += f"{alert_symbol} High throttle position detected! Maximum throttle position recorded: {average_throttle:.2f}%\n\n"
                recommendations += "\u2022 Practice smooth and gradual acceleration to improve fuel efficiency and reduce engine wear.\n\n"
            elif average_throttle < 20:
                alerts += f"{alert_symbol} Low throttle position detected! Minimum throttle position recorded: {average_throttle:.2f}%\n\n"
                recommendations += "\u2022 Avoid aggressive driving and maintain a steady throttle position for better fuel economy.\n\n"
            else:
                normaldata += f"\u2713 Average throttle position: {average_throttle:.2f}%\n\n"
                Tips += "\u2605 Gradual and steady throttle inputs contribute to fuel efficiency and smooth driving.\n\n"

            # Load Value Analysis
        elif column_name == ' Calculated load value (%)':

            average_load_value = column_data.mean()

            if average_load_value > 80:
                alerts += f"{alert_symbol} High average calculated load value! Average load value: {average_load_value:.2f}%\n\n"
                recommendations += "\u2022Avoid heavy acceleration and consider checking the air filter, fuel injectors, and other components for optimal engine performance.\n\n"
        elif average_load_value < 20:
```

```
            alerts += f"{alert_symbol} Low average calculated load value! Average load value: {average_load_value:.2f}%\n\n"
            recommendations += "\u2022Ensure proper acceleration to improve engine efficiency and power.\n\n"
        else:
            normaldata += f"\u2713 Average calculated load value: {average_load_value:.2f}%\n\n"
            Tips += "\u2605Maintain a balance in acceleration to optimize fuel consumption and engine performance.\n\n"

    # Consolidate all messages into one messagebox
    final_title = "Alerts, Recommendations, and Tips"
    tip_url = "https://www.mobil.com/en/sap/personal-vehicles/car/vehicle-maintenance/reduce-fuel-consumption."
    self.show_custom_message_box(final_title, alerts, recommendations,normaldata,driver_category,Tips, tip_url)

else:
    messagebox.showinfo("Information", "No data.")
```

Τέλος ακόμα ένα βασικό στοιχείο του προγράμματος είναι η δημιουργία γραφημάτων σε χάρτη, με την χρήση των δεδομένων Latitude και Longitude από το αρχείο. Στα γραφήματα αυτά ο χρήστης μπορεί να δει όλη την διαδρομή που έκανε στον χάρτη, αλλά και τα δεδομένα την εκάστοτε χρονική στιγμή στο συγκριμένο σημείο. Ο χρωματισμός βασίζεται στην τιμή φορτιού κινητήρα την συγκεκριμένη στιγμή.



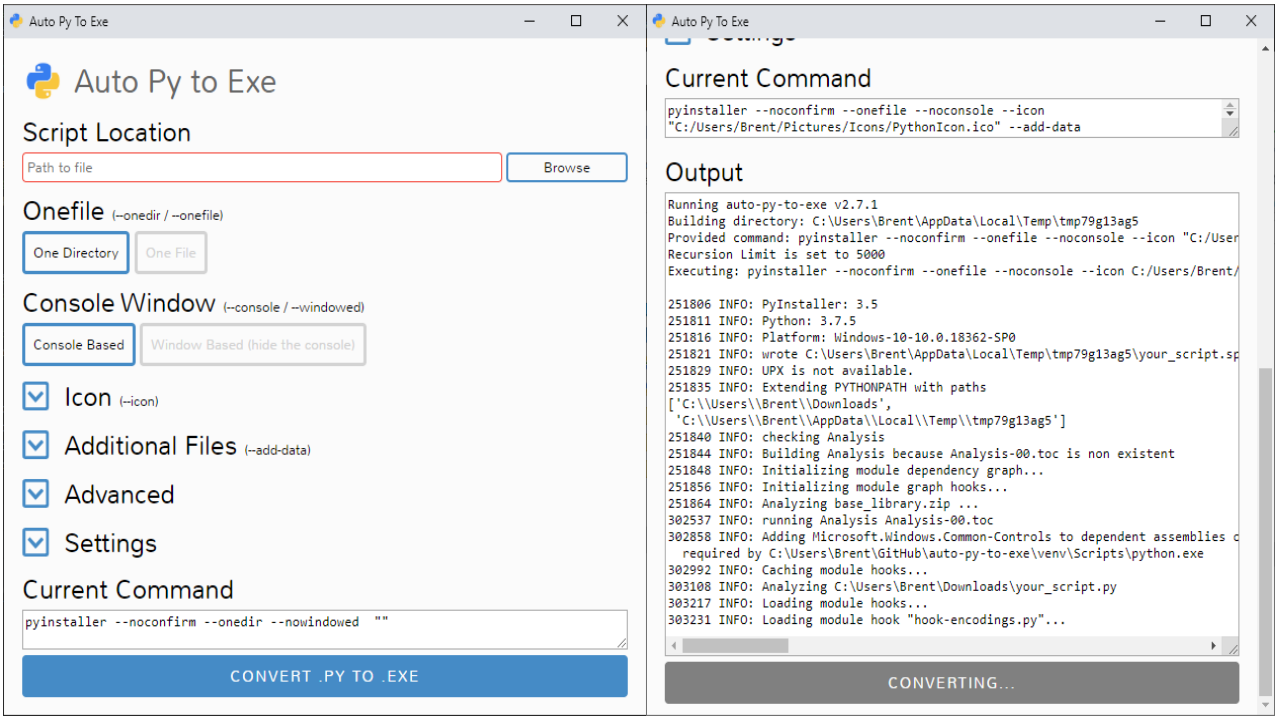
Εικόνα 40:Data point markers

Λειτουργια Map ploting:

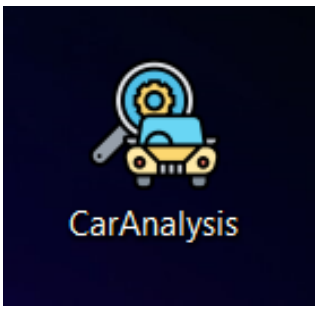
```
def view_map(self):
    if self.df is not None and ' Latitude (deg)' in self.df.columns and ' Longitude (deg)' in self.df.columns:
        latitudes = self.df[' Latitude (deg)'].tolist()
```

```
        longitudes = self.df[' Longitude (deg)'].tolist()

        # Create a folium map centered at the first location
        car_map = folium.Map(location=[latitudes[0], longitudes[0]], zoom_start=15)

        # Add a line to the map representing the car's movement
        folium.PolyLine(list(zip(latitudes, longitudes)), color='blue').add_to(car_map)

        # Add a marker for each data point
        for i, (lat, lon) in enumerate(zip(latitudes, longitudes)):
            engine_load = self.df[' Calculated load value (%)'].iloc[i]

            # Dynamically include all non-empty columns in the popup content
            popup_content = f"Latitude: {lat}<br>Longitude: {lon}<br>"
            for column in self.df.columns:
                if column in ['Vehicle Speed', 'Engine RPM'] or (self.df[column].iloc[i] != 0 and pd.notna(self.df[column].iloc[i])):
                    popup_content += f"{column}: {self.df[column].iloc[i]}<br>"


            # Set marker color based on engine load
            if engine_load > 80:
                marker_color = 'red'
            elif 40 <= engine_load <= 80:
                marker_color = 'yellow'
            else:
                marker_color = 'green'

            icon = folium.CustomIcon(icon_image="https://raw.githubusercontent.com/pointhi/leaflet-color-markers/master/img/marker-icon-2x-
{}.png".format(marker_color),
                                     icon_size=(25, 41), icon_anchor=(12, 41), popup_anchor=(1, -34))
            folium.Marker(location=[lat, lon], popup=folium.Popup(popup_content, max_width=300),
                          icon=icon).add_to(car_map)

        # Save the map to an HTML file
        map_file_path = "car_movement_map.html"
        car_map.save(map_file_path)

        # Open the HTML file in a web browser
        webbrowser.open(map_file_path)
    else:
        messagebox.showinfo("Information", "No data or ' Latitude (deg)' and ' Longitude (deg)' columns in the dataset.")
```

Προκειμένου να γίνει το πρόγραμμα πιο φιλικό προς τον χρήστη, μετατράπηκε από python scrypt σε εκτελέσιμο πρόγραμμα .exe ώστε να εκτελείται με ένα κλικ από την επιφάνεια εργασίας. Για να γίνει αυτή η μετατροπή χρησιμοποιήθηκε ένα converter εργαλείο της python, που ονομάζεται auto-py-to-exe. Μέσα στο πρόγραμμα δίνεται και η δυνατότητα επιλογής της εικόνας αρκεί αυτή να είναι σε .ico μορφή.



Εικόνα 41: Py to EXE converter



Εικόνα 42: Desktop executable file

6.4 Βιβλιοθήκες Προγράμματος

Ο κώδικας περιλαμβάνει τη χρήση πολλών βιβλιοθηκών που εξυπηρετούν διάφορες λειτουργίες, από γραφική διεπαφή μέχρι οπτικοποίηση δεδομένων. Παρακάτω γίνεται αναφορά στις βιβλιοθήκες που χρησιμοποιήθηκαν:

tkinter (tk): Αυτή η βιβλιοθήκη χρησιμοποιείται για τη δημιουργία γραφικών διεπαφών χρήστη (GUI) και είναι ενσωματωμένη στις περισσότερες εκδόσεις της Python.

filedialog: Η βιβλιοθήκη filedialog του tkinter χρησιμοποιείται για τη δημιουργία παραθύρων διαλόγου ώστε ο χρήστης να μπορεί να επιλέξει αρχεία.

Toplevel: Χρησιμοποιείται για τη δημιουργία επιπλέον παραθύρων στην κεντρική εφαρμογή.

messagebox: Αυτή η βιβλιοθήκη παρέχει παραθυράκια μηνυμάτων και προειδοποιήσεων.

scrolledtext: Επιτρέπει τη δημιουργία πεδίου κειμένου με δυνατότητα κύλισης.

FigureCanvasTkAgg: Χρησιμοποιείται για την ενσωμάτωση γραφικών αντικειμένων Matplotlib σε ένα παράθυρο tkinter.

matplotlib.pyplot: Αποτελεί μια βιβλιοθήκη για τον σχεδιασμό γραφημάτων και plotting.

pandas: Χρησιμοποιείται για την εργασία με δεδομένα σε μορφή πίνακα.

folium: Χρησιμοποιείται για τη δημιουργία διαδραστικών χαρτών.

webbrowser: Χρησιμοποιείται για τον έλεγχο προγράμματος περιήγησης του χρήστη.

googletrans: Χρησιμοποιείται για τη μετάφραση κειμένου.

mplcursors: Προσθέτει δυνατότητες με την χρήση του κέρσορα σε διαγράμματα.

Οι βιβλιοθήκες αυτές ενισχύουν την εφαρμογή, προσφέροντας δυνατότητες όπως διάλογοι αρχείων, γραφικές παραστάσεις, επικοινωνία με τον χρήστη και πολλά άλλα.

7 Συμπεράσματα και Μελλοντικές Επεκτάσεις

Με την ολοκλήρωση της εργασίας, παρατηρείται ότι η χρήση της τεχνολογίας ενισχύει τις δυνατότητες άντλησης και επεξεργασίας δεδομένων σχετικών με τη λειτουργία ενός οχήματος. Παραδείγματα δεδομένων που μπορούν να εξαχθούν αποτελούν τα σχετικά με το σύστημα διαχείρισης κινητήρα και δεδομένα που προκύπτουν από την κίνηση του οχήματος, ενώ παράλληλα απλοποιείται η διαδικασία εξαγωγής συμπερασμάτων για την οδηγική συμπεριφορά. Η εργασία φαίνεται να έχει πετύχει τους αρχικούς της στόχους, όπως προκύπτει από τις δοκιμές της κατασκευής πάνω σε διάφορα οχήματα και την επιτυχή χρήση των αντίστοιχα αποθηκευμένων δεδομένων από το λογισμικό (CarAnalysis). Η δυνατότητα του λογισμικού να αντλεί πληροφορίες από διάφορα οχήματα, ανεξαρτήτως μάρκας ή μοντέλου, επιτρέπει την ευρεία εφαρμογή του συστήματος στην καθημερινότητα των οδηγών.

Σημαντικό δείκτη επιτυχίας του λογισμικού αποτελεί η ορθή χρήση των αποθηκευμένων δεδομένων, η οποία προσφέρει σταθερότητα και συνέπεια στην παροχή πληροφοριών. Η δυνατότητα

αξιοποίησης ιστορικών δεδομένων βοηθά στην ανίχνευση των τάσεων και τη διαμόρφωση προσαρμοσμένων συμβουλών για κάθε οδηγό, με βάση τις προηγούμενες επιδόσεις του.

Τέλος, όσον αφορά τον οδηγό και γενικότερα την εμπειρία του χρήστη το λογισμικό ανάλυσης δεδομένων προσφέρει πολλαπλά οφέλη, από τη βελτιστοποίηση της οδηγικής συμπεριφοράς μέχρι την πρόληψη πιθανών προβλημάτων στα οχήματα. Μέσω της οπτικοποίησης των δεδομένων ο οδηγός αποκτά πλέον πρόσβαση σε δεδομένα σχετικά με τον κινητήρα και την κίνηση του οχήματός του, τα οποία μεταφράζονται επιτυχώς στις αντίστοιχες συμβουλές. Στόχο των ενδείξεων και συμβουλών που εμφανίζονται από το λογισμικό αποτελεί μακροπρόθεσμα η αλλαγή της συμπεριφοράς του οδηγού, με νέες δυνατότητες για βελτίωση της οδηγικής του ικανότητας όπως και την δυνατότητα να συνεισφέρει στην προστασία του περιβάλλοντος.

Η υπάρχουσα διπλωματική, αποτελεί μια σταθερή βάση για μελλοντικές επεκτάσεις και βελτιώσεις. Ένα βασικό στοιχείο μελλοντικής επέκτασης είναι η ενσωμάτωση της χρήσης υπολογιστικού νέφους για να διευκολυνθεί η απομακρυσμένη αποθήκευση και πρόσβαση στα δεδομένα. Η εμπειρία του χρήστη θα μπορούσε να βελτιωθεί σημαντικά μέσω της χρήσης του συστήματος από μια εφαρμογή κινητού, η οποία θα προσέφερε ειδοποιήσεις σε πραγματικό χρόνο κατά την κίνηση. Σχετικά με τις δυνατότητες διάγνωσης, η ακρίβεια των προβλέψεων καθώς και η εμφάνιση προληπτικών προτάσεων, βασιζόμενων σε ιστορικά δεδομένα, θα ενισχύονταν σημαντικά με την ενσωμάτωση της μηχανικής μάθησης. Καταληκτικά, οι δυνατότητες του συστήματος θα μπορούσαν να αξιοποιηθούν περεταίρω μέσω της διασύνδεσης αυτού με άλλες έξυπνες συσκευές στο αυτοκίνητο ή στο σπίτι.

Βιβλιογραφία

[1] Admin (2022) 'Types of Arduino Boards | Arduino Uno | Mega | Mini | Specification,' Techatronic, 10 February. https://techatronic.com/types-of-arduino-boards-arduino-uno-mega-mini-specification/.

[2] Anusha, P. et al. (2021) 'WITHDRAWN: Smart internet of vehicle maintenance system,' Materials Today: Proceedings [Preprint]. https://doi.org/10.1016/j.matpr.2020.11.303.

[3] Arduino - home (no date). https://www.arduino.cc/.

[4] Baek, S.-H. and Jang, J.-W. (2015) 'Implementation of integrated OBD-II connector with external network,' Information Systems, 50, pp. 69–75. https://doi.org/10.1016/j.is.2014.06.011.

[5] dfrobot.com (no date) TELEMATICS 3.5" TFT Touch LCD Shield (Discontinued). https://www.dfrobot.com/product-1500.html.

[6] Kumar, R. and Jain, A. (2023) 'Driving behavior analysis and classification by vehicle OBD data using machine learning,' The Journal of Supercomputing, 79(16), pp. 18800–18819. https://doi.org/10.1007/s11227-023-05364-3.

[7] ODB-II Connector (no date). https://components101.com/connectors/obd2.

[8] Pinout (2022) MPU-6050 Datasheet PDF - Motion Tracking Device. https://www.datasheetcafe.com/mpu-6050-datasheet-pdf.

[9] Project Maniacs (no date). https://projectmaniacs.wordpress.com/.

[10] Rodríguez, A.R., Álvarez, J.R.V. and Rodríguez, R.I. (2018) 'Implementation of an OBD-II diagnostics tool over CAN-BUS with Arduino,' Sistemas & Telemática, 2018, Vol 16, Issue 45, P31, 16(45). https://doi.org/10.18046/syt.v16i45.2747.

[11] Stanleyhuangyc (no date) GitHub - stanleyhuangyc/Freematics: Official source code repository for Freematics. https://github.com/stanleyhuangyc/Freematics.

[12] Tang, L., Huang, W. and You, J. (2019) 'The design of the intelligent car based on the Arduino UNO and Lab VIEW,' Journal of Physics: Conference Series, 1288(1), p. 012071. https://doi.org/10.1088/1742-6596/1288/1/012071.

[13] Vehicle-to-Vehicle communication technology (2018).

[14] https://ieeexplore.ieee.org/document/8500189.

[15] Yousef (2023) Different types of car Sensors and their Functions [PDF]. https://www.theengineerspost.com/types-of-car-sensors/?utm_content=cmp-true.

[16] Συνεισφέροντες στα εγχειρήματα Wikimedia (2023) Arduino. https://el.wikipedia.org/wiki/Arduino.

8 Επισυναπτόμενοι κώδικες με σχολιασμούς

8.1 Κώδικας προγράμματος CarAnalysis

- **CarAnalysis.py**

```
#Created by efthimios karabilis (mai20023)


import tkinter as tk
from tkinter import filedialog, Toplevel, messagebox,scrolledtext
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.pyplot as plt
import pandas as pd
import folium
import webbrowser
from folium import plugins
from functools import partial
from PIL import Image, ImageTk
from googletrans import Translator
import mplcursors
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
# from matplotlib.widgets import Zoom


ENGLISH_TRANSLATIONS = {
    "Load File": "Load File",
    "Visualize the Chart": "Visualize the Chart",
    "Clear Plots": "Clear Plots",
    "ΕΛΕΓΧΟΣ": "ΕΛΕΓΧΟΣ",
    "View Car Map": "View Car Map",
    "DataView": "DataView",
}

GREEK_TRANSLATIONS = {
```

```
    "Load File": "Φόρτωση Αρχείου",
    "Visualize the Chart": "Οπτικοποίηση Διαγράμματος",
    "Clear Plots": "Καθαρισμός Διαγραμμάτων",
    "ΕΛΕΓΧΟΣ": "Έλεγχος",
    "View Car Map": "Προβολή Χάρτη Αυτοκινήτου",
    "DataView": "Προβολή Δεδομένων",
}


class CSVVisualizerApp:
    def __init__(self, root):
        self.root = root
        self.root.title("CarAnalysis")



        # Load background image
        background_image = Image.open("bg.jpg")  # Replace with the path to your image
        self.background_photo = ImageTk.PhotoImage(background_image)

        # Create a label to hold the background image
        background_label = tk.Label(root, image=self.background_photo)
        background_label.place(relwidth=1, relheight=1)  # Cover the entire window

        self.df = None
        self.selected_column1 = None
        self.selected_column2 = None
        self.language = "english"  # Default language
        self.translations = ENGLISH_TRANSLATIONS  # Default translations


        # Frame for widgets
        self.frame = tk.Frame(root, bg="white")
        self.frame.pack(padx=10, pady=10)









        # Load CSV Button
        self.load_button = tk.Button(self.frame, text="Load File", command=self.load_csv, bg="#4CAF50",
fg="white",width=15, height=2)
        self.load_button.grid(row=0, column=3, padx=5, pady=5)

        # Dropdown for selecting the first column
        self.column_var1 = tk.StringVar()
        self.column_dropdown1 = tk.OptionMenu(self.frame, self.column_var1, "")
        self.column_dropdown1.grid(row=0, column=4, padx=5, pady=5)

        # Dropdown for selecting the second column
```

```
        self.column_var2 = tk.StringVar()
        self.column_dropdown2 = tk.OptionMenu(self.frame, self.column_var2, "")
        self.column_dropdown2.grid(row=0, column=5, padx=12, pady=12)
 # Visualize Button
        self.visualize_button = tk.Button(self.frame, text="Visualize the Chart", command=self.visualize_data, bg="#3498db",
fg="white",width=15, height=2)
        self.visualize_button.grid(row=0, column=6, padx=12, pady=12)

        # Clear Button
        self.clear_button = tk.Button(self.frame, text="Clear Plots", command=self.clear_plots, bg="#E69138",
fg="white",width=15, height=2)
        self.clear_button.grid(row=0, column=7, padx=12, pady=12)

        # Check list Button
        self.check_temp_button = tk.Button(self.frame, text="Data Analysis", command=self.check_list,bg="#E74C3C",
fg="white",width=15, height=2)
        self.check_temp_button.grid(row=1, column=5, padx=12, pady=12)

        # View Map Button
        self.view_map_button = tk.Button(self.frame, text="View Car Map", command=self.view_map, bg="#2ecc71",
fg="white",width=15, height=2)
        self.view_map_button.grid(row=1, column=6, padx=12, pady=12)

        # More Information Button
        self.more_info_button = tk.Button(self.frame, text="More Information", command=self.open_more_info,
bg="#8e44ad", fg="white",width=15, height=2)
        self.more_info_button.grid(row=1, column=7, padx=12, pady=12)

        # Language Switch Button
        self.language_switch_button = tk.Button(self.frame, text="ΕΛΛΗΝΙΚΑ", command=self.toggle_language,
bg="#3498db", fg="white",width=15, height=2)
        self.language_switch_button.grid(row=1, column=8, padx=12, pady=12)



        # Data View Button
        self.data_view_button = tk.Button(self.frame, text="Data View", command=self.show_data_view, bg="#3498db",
fg="white",width=15, height=2)
        self.data_view_button.grid(row=0, column=8, padx=12, pady=12)


        # Matplotlib figures
        self.figure1, self.ax1 = plt.subplots(figsize=(14,9))
        self.ax1.set_facecolor('gray')


        # Canvas for the first plot
        self.canvas1 = FigureCanvasTkAgg(self.figure1, master=self.root)
        self.canvas1.get_tk_widget().pack(side=tk.LEFT, expand=1)
        # self.canvas1.get_tk_widget().pack_propagate(False)

        # Enable pan and zoom using mpl_connect
```

```
    self.canvas1.mpl_connect("scroll_event", self.on_scroll)
    self.canvas1.mpl_connect("button_press_event", self.on_button_press)
    self.canvas1.mpl_connect("button_release_event", self.on_button_release)
    self.canvas1.mpl_connect("motion_notify_event", self.on_mouse_move)
    self.canvas1.get_tk_widget().bind("<Enter>", self.enter_canvas)
    self.canvas1.get_tk_widget().bind("<Leave>", self.leave_canvas)

    # Flag to track whether the mouse button is pressed
    self.mouse_button_pressed = False



def on_scroll(self, event):
    if event.button == 'up':
        self.ax1.set_xlim(self.ax1.get_xlim()[0] * 1.2, self.ax1.get_xlim()[1] * 1.2)
        self.ax1.set_ylim(self.ax1.get_ylim()[0] * 1.2, self.ax1.get_ylim()[1] * 1.2)
    elif event.button == 'down':
        self.ax1.set_xlim(self.ax1.get_xlim()[0] / 1.2, self.ax1.get_xlim()[1] / 1.2)
        self.ax1.set_ylim(self.ax1.get_ylim()[0] / 1.2, self.ax1.get_ylim()[1] / 1.2)

    self.canvas1.draw()

def on_button_press(self, event):
    if event.button == 1:  # Check if left mouse button is clicked
        self.mouse_button_pressed = True

def on_button_release(self, event):
    if event.button == 1:
        self.mouse_button_pressed = False
        # Reset the last position when releasing the mouse button
        self.last_x = None
        self.last_y = None

def on_mouse_move(self, event):
    if self.mouse_button_pressed and self.last_x is not None and self.last_y is not None:
        # Calculate the change in mouse position
        dx = event.x - self.last_x
        dy = event.y - self.last_y

        # Convert pixel coordinates to data coordinates
        xlim = self.ax1.get_xlim()
        ylim = self.ax1.get_ylim()
        x_data = xlim[0] - dx * (xlim[1] - xlim[0]) / self.canvas1.get_width_height()[0]
        y_data = ylim[0] - dy * (ylim[1] - ylim[0]) / self.canvas1.get_width_height()[1]

        # Update the xlim and ylim
        self.ax1.set_xlim(x_data, x_data + (xlim[1] - xlim[0]))
        self.ax1.set_ylim(y_data, y_data + (ylim[1] - ylim[0]))

        self.canvas1.draw()
```

```
    # Save the current mouse position
    self.last_x = event.x
    self.last_y = event.y




def on_button_release(self, event):
  if event.button == 1:
    self.mouse_button_pressed = False
    # Reset the last position when releasing the mouse button
    self.last_x = None
    self.last_y = None

    # Remove all text annotations on the plot
    for annotation in self.ax1.texts:
        annotation.remove()

    self.canvas1.draw()



def enter_canvas(self, event):
    # Show the cursor when entering the canvas
    self.root.config(cursor='arrow')

def leave_canvas(self, event):
    # Show the cursor when leaving the canvas
    self.root.config(cursor='arrow')



    # Csv file loader
def load_csv(self):
    file_path = filedialog.askopenfilename(title="Select CSV file", filetypes=[("CSV files", "*.csv")])
    if file_path:
        self.df = pd.read_csv(file_path)
        columns = self.df.columns

        # Clear previous options
        self.column_dropdown1['menu'].delete(0, 'end')
        self.column_dropdown2['menu'].delete(0, 'end')

        # Add "None" option to both dropdowns
        self.column_dropdown1['menu'].add_command(label="None", command=tk._setit(self.column_var1, ""))
        self.column_dropdown2['menu'].add_command(label="None", command=tk._setit(self.column_var2, ""))

        # Add columns to the dropdowns
        for column in columns:
            self.column_dropdown1['menu'].add_command(label=column, command=tk._setit(self.column_var1, column))
            self.column_dropdown2['menu'].add_command(label=column, command=tk._setit(self.column_var2, column))
```

```
        # Set the default selection to "None"
        self.column_var1.set("")
        self.column_var2.set("")




    # Data visualisation button
def visualize_data(self):
    if self.df is not None:
        self.selected_column1 = self.column_var1.get()
        self.selected_column2 = self.column_var2.get()

        self.ax1.clear()

        if self.selected_column1:
            self.ax1.plot(self.df.index, self.df[self.selected_column1], label=self.selected_column1, color="#3498db")

        if self.selected_column2:
            self.ax1.plot(self.df.index, self.df[self.selected_column2], label=self.selected_column2, color="#e74c3c")

        if self.selected_column1 or self.selected_column2:
            self.ax1.set_title(f"Comparison of {self.selected_column1} and {self.selected_column2}", color="#333333")
            self.ax1.set_xlabel("Index", color="#333333")
            self.ax1.set_ylabel("Values", color="#333333")
            self.ax1.legend()

            # Adjust y-axis limits based on the selected columns
            if self.selected_column1 and self.selected_column2:
                min_value = min(self.df[self.selected_column1].min(), self.df[self.selected_column2].min())
                max_value = max(self.df[self.selected_column1].max(), self.df[self.selected_column2].max())
            elif self.selected_column1:
                min_value, max_value = self.df[self.selected_column1].min(), self.df[self.selected_column1].max()
            elif self.selected_column2:
                min_value, max_value = self.df[self.selected_column2].min(), self.df[self.selected_column2].max()
            else:
                min_value, max_value = 0, 1  # Default values if no column is selected

            self.ax1.set_ylim(min_value, max_value)

            self.canvas1.draw()

         # Enable zooming with the scroll wheel using mplcursors
        mplcursors.cursor(hover=True)




# Translate button from Greek to English vice versa
```

```
    def translate(self, text):
        return self.translations.get(text, text)

    def set_language(self, language):
        if language == "english":
            self.translations = ENGLISH_TRANSLATIONS
        elif language == "greek":
            self.translations = GREEK_TRANSLATIONS
        else:
            return  # Unknown language

        # Update button text and other UI elements
        self.load_button["text"] = self.translate("Load File")
        self.visualize_button["text"] = self.translate("Visualize the Chart")
        self.clear_button["text"] = self.translate("Clear Plots")
        self.check_temp_button["text"] = self.translate("ΕΛΕΓΧΟΣ")
        self.view_map_button["text"] = self.translate("View Car Map")

        # Update language switch button text
        self.language_switch_button["text"] = "ΕΛΛΗΝΙΚΑ" if self.language == "english" else "ENGLISH"

    def toggle_language(self):
        if self.language == "english":
            self.language = "greek"
        else:
            self.language = "english"
        self.set_language(self.language)




     # Link for More information Button
    def open_more_info(self):

        link_to_open = 'https://obdsoftware.my.site.com/s/article/How-to-read-OBDII-live-data-A-mechanic-guide'
        webbrowser.open(link_to_open)


    def clear_plots(self):
        self.ax1.clear()
        self.canvas1.draw()

    def show_custom_message_box(self, title, alerts, recommendations,normaldata,driver_category, tips, tip_url=None):
     custom_box = tk.Toplevel(self.frame)
     custom_box.title(title)

    # Create a scrolled text widget to display the messages
     text_widget = scrolledtext.ScrolledText(custom_box, wrap=tk.WORD, width=120, height=40, font=("Verdana",
12,"bold"))
     text_widget.grid(row=0, column=0, padx=40, pady=20, columnspan=2, sticky=tk.W+tk.E+tk.N+tk.S)
```

```
    # Insert messages with formatting
    text_widget.insert(tk.INSERT, "\u2022 Driver category:\n\n" +driver_category+ "\n\n\n\n", "bold")
    text_widget.tag_configure("bold", font=("Arial", 14, "bold"))
    text_widget.insert(tk.INSERT, "Alerts:\n\n\n" + alerts + "\n\n", "bold")
    text_widget.insert(tk.INSERT, "\u2714 NormalData:\n\n\n" + normaldata + "\n\n", "bold")
    text_widget.insert(tk.INSERT, "\u2022 Recommendations:\n\n\n" + recommendations + "\n\n", "bold")
    text_widget.insert(tk.INSERT, "\u2605 Tips:\n\n\n" + tips + "\n\n", "bold")



    # Add hyperlink to the Tips section if tip_url is provided
    if tip_url:
        link_tag = "link"
        text_widget.tag_configure(link_tag, foreground="green", underline=False)
        text_widget.insert(tk.INSERT, f"\u2605FOR MORE INFORMATION ABOUT CAR DATA LOGGING CLICK
HERE) .\n\n\n\n", link_tag)
        text_widget.tag_bind(link_tag, "<Button-1>", lambda event, url=tip_url: self.open_link(event, url))
        text_widget.tag_bind(link_tag, "<Enter>", lambda event, widget=text_widget: widget.config(cursor="hand2"))
        text_widget.tag_bind(link_tag, "<Leave>", lambda event, widget=text_widget: widget.config(cursor=""))

    text_widget.config(state=tk.DISABLED)  # Make the text widget read-only

    ok_button = tk.Button(custom_box, text="OK", command=custom_box.destroy)
    ok_button.grid(row=1, column=0, columnspan=2, pady=10)




    #Data view button as raw file with centering
    def show_data_view(self):
        if self.df is not None:
            # Create a new window for data view
            data_view_window = tk.Toplevel(self.root)
            data_view_window.title("Data View")

            # Create a text widget to display data
            data_text = tk.Text(data_view_window, wrap="none", font=("Courier", 10))  # Use a monospaced font for better
alignment
            data_text.pack(expand=True, fill="both")

            # Format the data for better alignment
            formatted_data = self.format_data_for_view()

            # Insert the formatted data into the text widget
            data_text.insert("1.0", formatted_data)

            # Add a vertical scrollbar to the text widget
            scrollbar_y = tk.Scrollbar(data_view_window, command=data_text.yview)
            scrollbar_y.pack(side="right", fill="y")
            data_text.config(yscrollcommand=scrollbar_y.set)
```

```
            # Add a horizontal scrollbar to the text widget
            scrollbar_x = tk.Scrollbar(data_view_window, command=data_text.xview, orient="horizontal")
            scrollbar_x.pack(side="bottom", fill="x")
            data_text.config(xscrollcommand=scrollbar_x.set)

        else:
            messagebox.showinfo("Information", "No data.")


    def format_data_for_view(self):
        # Adjust column widths for better alignment
        formatted_data = self.df.to_string(index=False)
        lines = formatted_data.split("\n")
        header_line = lines[0]
        separator_line = "+".join(["-" * (len(col) + 2) for col in header_line.split()])
        lines.insert(1, separator_line)
        return "\n".join(lines)


    def open_link(self, event, url):
     import webbrowser
     webbrowser.open(url)




      #Check list building

    def check_list(self):
      if self.df is not None:
        required_columns = [' Engine coolant temperature', 'Voltage', ' Relative throttle position (%)', ' Vehiclespeed', '
Calculated load value (%)', ' Engine RPM (RPM)']

        alert_symbol = "\u26A0"  # Unicode character for the alert symbol

        # Initialize variables outside the loop
        alerts = ""
        recommendations = ""
        normaldata=""
        Tips = ""
        driver_category = ""

        for column_name in required_columns:
            try:
                column_data = self.df[column_name]
            except KeyError:
                # If the column is not found, print a message and continue to the next column
                print(f"Column '{column_name}' not found. Skipping...")
                continue

            # RPM Analysis
```

```
        if column_name == ' Engine RPM (RPM)':
            max_rpm = column_data.max()
            min_rpm = column_data.min()
            average_rpm = column_data.mean()

            if average_rpm > 3800:
                alerts += f"{alert_symbol} High RPM detected! Maximum RPM recorded: {average_rpm:.2f}%\n\n"
                recommendations += "\u2022Recommendation: Avoid over-revving the engine to prevent excessive wear and damage.\n\n"
                Tips += "\u2605Tip: Shift gears smoothly and avoid aggressive acceleration to keep RPM within a safe range.\n\n"
                driver_category = f"\u26A0 Aggressive"
            elif 1500< average_rpm < 3800:
                normaldata += f"\u2713  RPM.  Range: {min_rpm}RPM - {max_rpm}RPM, average RPM:{average_rpm:.2f}%.\n\n"
                Tips += "\u2605Tip: Maintain a fuel-efficient driving style by keeping RPM low.\n\n"
                driver_category = "Economy-friendly"
            else:
                normaldata += f"\u2713  RPM.average: {average_rpm:.2f}%\n\n"
                Tips += "\u2605Driving with the RPM in too low of a gear can lead to engine strain and decreased fuel efficiency.\n\n"
                driver_category = "Inexperienced"




        if column_name == ' Engine coolant temperature':
            max_temp = column_data.max()
            min_temp = column_data.min()

            # Temperature Analysis
            if max_temp > 100:
                alerts += f"{alert_symbol} Overheating problem! Maximum temperature recorded: {max_temp}°C\n\n"
                recommendations+= "\u2022Please visti a workshop, Problem in the cooling system.\n\n"
            elif min_temp < 60:
                alerts += f"{alert_symbol} Low temperature. Minimum temperature recorded: {min_temp}°C\n\n"
                recommendations += "\u2022Consider driving at a higher speed to increase engine temperature, or check for potential issues with the cooling system.\n\n\n\n"
            else:
                normaldata += f"\u2713 Normal temperature. Range: {min_temp}°C - {max_temp}°C\n\n"
                Tips += "\u2605Keep up with regular maintenance to ensure optimal engine performance.\n\n"

        elif column_name == 'Voltage':
            max_voltage = column_data.max()
            min_voltage = column_data.min()

            # Voltage Analysis
            if max_voltage > 15:
```

```
                alerts += f"{alert_symbol} Overcharging problem! Maximum voltage recorded: {max_voltage}V\n\n"
                recommendations += "\u2022Visit a workshop to inspect and address alternator or battery faults
immediately.\n\n"
            elif min_voltage < 11:
                alerts += f"{alert_symbol} Undercharging problem! Minimum voltage recorded: {min_voltage}V\n\n"
                recommendations += "\u2022Check the alternator, battery, and charging system. Charge or replace the battery if
necessary.\n\n"
            else:
                normaldata += f"\u2713 Normal voltage. Range: {min_voltage}V - {max_voltage}V\n\n"
                Tips += "\u2605Regularly check the battery and charging system for optimal performance.\n\n"

        elif column_name == ' Vehiclespeed':
            max_speed = column_data.max()
            min_speed = column_data.min()
            min_rpm = column_data.min()

            # Check for frequent idling instances
            count_idle_instances = ((self.df[' Vehiclespeed'] == 0) & (self.df[' Engine RPM (RPM)'] < 1000)).sum()
            threshold_idle_instances = 5  # Adjust as needed


            # Speed Analysis
            if max_speed > 150:
                alerts += f"{alert_symbol} High speed detected! Maximum speed recorded: {max_speed} km/h\n\n"
                recommendations += "\u2022Please drive within the recommended speed limits for your safety.\n\n"
            elif min_speed == 0 and min_rpm < 1000:
                alerts += f"{alert_symbol} Vehicle at a complete stop with Engine Idling! Extended periods of engine idling can
lead to increased fuel consumption, unnecessary emissions, and potential wear on engine components\n\n"
                recommendations += "\u2022Consider switching the car off during extended idle periods to conserve fuel,
minimize emissions, and reduce wear on the engine. Turning off the engine when stationary for an extended time can
contribute to fuel efficiency and environmental sustainability.\n\n"
            else:
                normaldata += f"\u2713 Normal speed. Range: {min_speed} km/h - {max_speed} km/h\n\n"
                Tips += "\u2605Adhering to the normal speed limit promotes safety and reduces potential risks.\n\n"




        elif column_name == ' Relative throttle position (%)':

            average_throttle = column_data.mean()


            # Throttle Position Analysis
            if average_throttle > 80:
                alerts += f"{alert_symbol} High throttle position detected! Maximum throttle position recorded:
{average_throttle:.2f}%\n\n"
                recommendations += "\u2022 Practice smooth and gradual acceleration to improve fuel efficiency and reduce
engine wear.\n\n"
            elif average_throttle < 20:
                alerts += f"{alert_symbol} Low throttle position detected! Minimum throttle position recorded:
```

```
{average_throttle:.2f}%\n\n"
                recommendations += "\u2022 Avoid aggressive driving and maintain a steady throttle position for better fuel economy.\n\n"
              else:
                normaldata += f"\u2713 Average throttle position: {average_throttle:.2f}%\n\n"
                Tips += "\u2605 Gradual and steady throttle inputs contribute to fuel efficiency and smooth driving.\n\n"

              # Load Value Analysis
            elif column_name == ' Calculated load value (%)':

              average_load_value = column_data.mean()

              if average_load_value > 80:
                alerts += f"{alert_symbol} High average calculated load value! Average load value: {average_load_value:.2f}%\n\n"
                recommendations += "\u2022Avoid heavy acceleration and consider checking the air filter, fuel injectors, and other components for optimal engine performance.\n\n"
            elif average_load_value < 20:
              alerts += f"{alert_symbol} Low average calculated load value! Average load value: {average_load_value:.2f}%\n\n"
                recommendations += "\u2022Ensure proper acceleration to improve engine efficiency and power.\n\n"
            else:
              normaldata += f"\u2713 Average calculated load value: {average_load_value:.2f}%\n\n"
              Tips += "\u2605Maintain a balance in acceleration to optimize fuel consumption and engine performance.\n\n"

          # Consolidate all messages into one messagebox
        final_title = "Alerts, Recommendations, and Tips"
        tip_url = "https://www.mobil.com/en/sap/personal-vehicles/car/vehicle-maintenance/reduce-fuel-consumption."
        self.show_custom_message_box(final_title, alerts, recommendations,normaldata,driver_category,Tips, tip_url)

      else:
        messagebox.showinfo("Information", "No data.")




    def view_map(self):
     if self.df is not None and ' Latitude (deg)' in self.df.columns and ' Longitude (deg)' in self.df.columns:
        latitudes = self.df[' Latitude (deg)'].tolist()
        longitudes = self.df[' Longitude (deg)'].tolist()

        # Create a folium map centered at the first location
        car_map = folium.Map(location=[latitudes[0], longitudes[0]], zoom_start=15)

        # Add a line to the map representing the car's movement
        folium.PolyLine(list(zip(latitudes, longitudes)), color='blue').add_to(car_map)

        # Add a marker for each data point
        for i, (lat, lon) in enumerate(zip(latitudes, longitudes)):
            engine_load = self.df[' Calculated load value (%)'].iloc[i]
```

```
            # Dynamically include all non-empty columns in the popup content
            popup_content = f"Latitude: {lat}<br>Longitude: {lon}<br>"
            for column in self.df.columns:
                if column in ['Vehicle Speed', 'Engine RPM'] or (self.df[column].iloc[i] != 0 and pd.notna(self.df[column].iloc[i])):
                    popup_content += f"{column}: {self.df[column].iloc[i]}<br>"


            # Set marker color based on engine load
            if engine_load > 80:
                marker_color = 'red'
            elif 40 <= engine_load <= 80:
                marker_color = 'yellow'
            else:
                marker_color = 'green'

            icon = folium.CustomIcon(icon_image="https://raw.githubusercontent.com/pointhi/leaflet-color-
markers/master/img/marker-icon-2x-{}.png".format(marker_color),
                                     icon_size=(25, 41), icon_anchor=(12, 41), popup_anchor=(1, -34))
            folium.Marker(location=[lat, lon], popup=folium.Popup(popup_content, max_width=300),
                          icon=icon).add_to(car_map)

        # Save the map to an HTML file
        map_file_path = "car_movement_map.html"
        car_map.save(map_file_path)

        # Open the HTML file in a web browser
        webbrowser.open(map_file_path)
    else:
        messagebox.showinfo("Information", "No data or ' Latitude (deg)' and ' Longitude (deg)' columns in the dataset.")

def center_window(window, width, height):
    screen_width = window.winfo_screenwidth()
    screen_height = window.winfo_screenheight()

    x = (screen_width - width) // 2
    y = (screen_height - height) // 2

    window.geometry(f"{width}x{height}+{x}+{y}")

def main():
    root = tk.Tk()
    root.configure(bg="#d6d2d2")  # Set the window background color

    # Create splash screen with background image
    splash = Toplevel(root)
    splash.title("CarAnalysis")
    splash.overrideredirect(True)  # Remove title bar

    # Load background image for the splash screen
    splash_bg_image = Image.open("bgs.png")  # Replace with the path to your image
    splash_bg_photo = ImageTk.PhotoImage(splash_bg_image)
```

```
    # Create a label to hold the background image
    splash_bg_label = tk.Label(splash, image=splash_bg_photo)
    splash_bg_label.image = splash_bg_photo
    splash_bg_label.pack()

    # Add text directly on the image
    tk.Label(splash, text="CarAnalysis", font=("Verdana", 60,"bold"), fg="white", bg="orange").place(relx=0.49, rely=0.5,
anchor="center")
    tk.Label(splash, text="mai20023", font=("Verdana", 20,"bold"), fg="white", bg="orange").place(relx=0.52, rely=0.4,
anchor="center")

    # Center the splash window
    center_window(splash, 1000, 600)

    root.withdraw()

    app = CSVVisualizerApp(root)



    splash_closed = tk.StringVar()
    splash_closed.set("no")

    def close_splash():
        splash_closed.set("yes")

    splash.after(4000, close_splash)

    def check_splash():
        if splash_closed.get() == "yes":
            splash.destroy()  # Destroy the splash window
            root.deiconify()  # Show the main application window
            root.update_idletasks()  # Ensure all idle tasks are complete
            center_window(root, 1820, 750)  # Center the main window
        else:
            root.after(100, check_splash)

    root.after(100, check_splash)
    root.mainloop()

if __name__ == "__main__":
    main()
```

8.2 Κώδικας κατασκευής Arduino

- **Arduinolog.ino**

```
/* Reference code for Freematics OBD-II Advanced Telematics Kit
 * Original code distributed under the BSD license
 * Written by Stanley Huang <support@freematics.com.au>
 *
 * Modified by Efthimios Karabilis ,(2024),for my research paper
 * "ΟΠΤΙΚΟΠΟΙΗΣΗ ΚΑΙ ΑΝΑΛΥΣΗ ΠΑΡΑΜΕΤΡΩΝ ΣΕ ΠΡΑΓΜΑΤΙΚΟ ΧΡΟΝΟ ΜΕΣΩ ΠΡΟΤΥΠΟΥ ΣΥΣΤΗΜΑΤΟΣ
ΔΙΑΓΝΩΣΗΣ OBD (On board diagnostic)"
 *
 * Enhancements:
 * - Clear data categorization into specific columns with clear titles
 *   for improved readability and ease of data analysis.
 * - Modified displayed data for enhanced visualization.
 */



#include <Arduino.h>
#include <SPI.h>
#include <Wire.h>
#include <OBD.h>
#include <MultiLCD.h>
#include <TinyGPS.h>
#include "setup.h"
#if ENABLE_DATA_LOG
#include <SD.h>
#endif
#include "Narcoleptic.h"
#include "images.h"
#include "datalogger.h"

// logger states
#define STATE_SD_READY 0x1
#define STATE_OBD_READY 0x2
#define STATE_GPS_CONNECTED 0x4
#define STATE_GPS_READY 0x8
#define STATE_MEMS_READY 0x10
#define STATE_GUI_ON 0x20
    int acc[3];
    int gyro[3];
```

```
    int mag[3];
    int temp;
    int32_t lat, lng;
#if USE_GPS
// GPS logging can only be enabled when there is additional hardware serial UART
#define GPSUART Serial2
TinyGPS gps;
#endif


static uint8_t lastFileSize = 0;
static uint32_t lastRefreshTime = 0;
static uint32_t distance = 0;
static uint32_t startTime = 0;
static uint16_t lastSpeed = 0;
static uint32_t lastSpeedTime = 0;
static uint32_t gpsDate = 0;
#if USE_GPS
static uint32_t lastGPSDataTime = 0;
static int gpsSpeed = -1;
#endif

byte state = 0;

void processMEMS();
void processGPS();

CDataLogger logger;

#ifdef OBD_ADAPTER_I2C
class CMyOBD : public COBDI2C
#else
class CMyOBD : public COBD
#endif
{
public:
    void dataIdleLoop()
    {
        if (!(state & STATE_GUI_ON)) {
          delay(10);
          return;
        }

#if USE_GPS
        uint32_t t = millis();
        while (GPSUART.available() && millis() - t < MAX_GPS_PROCESS_TIME) {
            processGPS();
        }
#endif
    }
};
```

```
CMyOBD obd;

void setColorByValue(int value, int threshold1, int threshold2, int threshold3)
{
   if (value < 0) value = -value;
   if (value < threshold1) {
     lcd.setColor(RGB16_WHITE);
   } else if (value < threshold2) {
     byte n = (uint32_t)(threshold2 - value) * 255 / (threshold2 - threshold1);
     lcd.setColor(255, 255, n);
   } else if (value < threshold3) {
     byte n = (uint32_t)(threshold3 - value) * 255 / (threshold3 - threshold2);
     lcd.setColor(255, n, 0);
   } else {
     lcd.setColor(255, 0, 0);
   }
}

void showPIDData(byte pid, int value)
{
   char buf[8];
   switch (pid) {
   case PID_RPM:
      lcd.setFontSize(FONT_SIZE_XLARGE);
      lcd.setCursor(14, 8);
      if (value >= 10000) break;
      setColorByValue(value, 2500, 3500, 5000);
      lcd.printInt(value, 6);
      break;
   case PID_SPEED:
      if (value < 1000) {
         lcd.setFontSize(FONT_SIZE_XLARGE);
         lcd.setCursor(50,3);
         setColorByValue(value, 60, 100, 160);
         lcd.printInt(value, 3);

#if USE_GPS
         if (gpsSpeed != -1) {
            lcd.setFontSize(FONT_SIZE_SMALL);
            lcd.setCursor(110, 3);
            lcd.setColor(RGB16_YELLOW);
            int diff = gpsSpeed - value;
            if (diff >= 0) {
               lcd.write('+');
               lcd.printInt(diff);
            } else {
               lcd.write('-');
               lcd.printInt(-diff);
            }
            lcd.write(' ');
```

```
            }
#endif
        }
        break;
    case PID_ENGINE_LOAD:
        lcd.setFontSize(FONT_SIZE_XLARGE);
        lcd.setCursor(50, 13);
        if (value >= 100) value = 99;
        setColorByValue(value, 75, 80, 100);
        lcd.printInt(value, 3);
        break;
    case PID_THROTTLE:
        lcd.setFontSize(FONT_SIZE_MEDIUM);
        lcd.setCursor(102, 27);
        if (value >= 100) value = 99;
        setColorByValue(value, 50, 75, 100);
        lcd.printInt(value, 2);
        break;
    case PID_COOLANT_TEMP:
        if (value < 100) {
            lcd.setFontSize(FONT_SIZE_MEDIUM);
            lcd.setCursor(102, 30);
            setColorByValue(value, 50, 70, 75);
            lcd.printInt(value, 2);
        }
        break;
    case PID_INTAKE_TEMP:
        if (value >= 0 && value < 100) {
            lcd.setFontSize(FONT_SIZE_MEDIUM);
            lcd.setCursor(102, 33);
            lcd.printInt(value, 2);
        }
        break;
    }
    lcd.setColor(RGB16_WHITE);
}

void fadeOutScreen()
{
    // fade out backlight
    for (int n = 254; n >= 0; n--) {
        lcd.setBackLight(n);
        delay(3);
    }
}

void fadeInScreen()
{
    for (int n = 1; n <= 255; n++) {
        lcd.setBackLight(n);
        delay(6);
```

```
    }
}

void initScreen()
{
    lcd.clear();



    lcd.setColor(RGB16_CYAN);
    lcd.setFontSize(FONT_SIZE_MEDIUM);
    lcd.setCursor(110,4);
    lcd.print("km/h");
    lcd.setCursor(110, 9);
    lcd.print("RPM");
    lcd.setFontSize(FONT_SIZE_SMALL);
    lcd.setCursor(110, 14);
    lcd.print("ENGINE");
    lcd.setCursor(110, 15);
    lcd.print("LOAD %");


    lcd.setFontSize(FONT_SIZE_MEDIUM);
    lcd.setCursor(208, 3);
    lcd.print("Elapsed");
    lcd.setCursor(204, 8);
    lcd.print("Distance");
    lcd.setCursor(180, 13);
    lcd.print("Average Speed");

    lcd.setCursor(16, 24);
    lcd.print("Battery     V");
    lcd.setCursor(16, 27);
    lcd.print("Throttle    %");
    lcd.setCursor(16, 30);
    lcd.print("CT        °C");
    lcd.setCursor(16, 33);
    lcd.print("Intake:     C");

    lcd.setCursor(180, 24);
    lcd.print("UTC:");
    lcd.setCursor(180, 27);
    lcd.print("LAT:");
    lcd.setCursor(180, 30);
    lcd.print("LNG:");
    lcd.setCursor(180, 33);
    lcd.print("ALT:");
    lcd.setCursor(180, 36);
    lcd.print("SAT:");

    lcd.setCursor(340, 3);
```

```
    lcd.print("Accelerometer");
    lcd.setCursor(356, 8);
    lcd.print("Gyroscope");


    lcd.setCursor(348, 24);
    lcd.print("OBD Interval");

    lcd.setCursor(348, 29);
    lcd.print("GPS Interval");

    lcd.setCursor(352, 34);
    lcd.print("Data Size");

    //lcd.setColor(0xFFFF);
    /*
    lcd.setCursor(32, 4);
    lcd.print("%");
    lcd.setCursor(68, 5);
    lcd.print("Intake Air");
    lcd.setCursor(112, 4);
    lcd.print("C");
    */

    state |= STATE_GUI_ON;

    fadeInScreen();
}

#if ENABLE_DATA_LOG
bool checkSD()
{
    Sd2Card card;
    SdVolume volume;
    state &= ~STATE_SD_READY;
    pinMode(SS, OUTPUT);

    lcd.setFontSize(FONT_SIZE_MEDIUM);
    if (card.init(SPI_HALF_SPEED, SD_CS_PIN)) {
        const char* type;
        switch(card.type()) {
        case SD_CARD_TYPE_SD1:
            type = "SD1";
            break;
        case SD_CARD_TYPE_SD2:
            type = "SD2";
            break;
        case SD_CARD_TYPE_SDHC:
            type = "SDHC";
            break;
        default:
```

```
            type = "SDx";
        }

        lcd.print(type);
        lcd.write(' ');
        if (!volume.init(card)) {
            lcd.println("No FAT!");
            return false;
        }

        uint32_t volumesize = volume.blocksPerCluster();
        volumesize >>= 1; // 512 bytes per block
        volumesize *= volume.clusterCount();
        volumesize >>= 10;

        lcd.print((int)volumesize);
        lcd.print("MB");
    } else {
        lcd.print("SD Card ");
        lcd.setColor(RGB16_RED);
        lcd.draw(cross, 16, 16);
        lcd.setColor(RGB16_WHITE);
        lcd.println();
        return false;
    }

    if (!SD.begin(SD_CS_PIN)) {
        lcd.println("Bad SD");
        return false;
    }

    state |= STATE_SD_READY;
    return true;
}
#endif

#if USE_GPS
void processGPS()
{
    // process GPS data
    char c = GPSUART.read();
    if (!gps.encode(c))
        return;

    // parsed GPS data is ready
    uint32_t time;
    uint32_t date;

    logger.dataTime = millis();

    gps.get_datetime(&date, &time, 0);
```

```
  if (date != gpsDate) {
    // log date only if it's changed and valid
    int year = date % 100;
    if (date < 1000000 && date >= 10000 && year >= 15 && (gpsDate == 0 || year - (gpsDate % 100) <= 1)) {
      logger.logData(PID_GPS_DATE, (int32_t)date);
      gpsDate = date;
    }
  }
  logger.logData(PID_GPS_TIME, (int32_t)time);


  gps.get_position(&lat, &lng, 0);

  byte sat = gps.satellites();

  // show GPS data interval
  lcd.setFontSize(FONT_SIZE_MEDIUM);
  if (lastGPSDataTime) {
    lcd.setCursor(380, 31);
    lcd.printInt((uint16_t)logger.dataTime - lastGPSDataTime);
    lcd.print("ms");
    lcd.printSpace(2);
  }

  // keep current data time as last GPS time
  lastGPSDataTime = logger.dataTime;

  // display UTC date/time
  lcd.setFlags(FLAG_PAD_ZERO);
  lcd.setCursor(216, 24);
  lcd.printLong(time, 8);

  // display latitude
  lcd.setCursor(216, 27);
  lcd.print((float)lat / 100000, 5);
  // display longitude
  lcd.setCursor(216, 30);
  lcd.print((float)lng / 100000, 5);
  // log latitude/longitude
  logger.logData(PID_GPS_LATITUDE, lat);
  logger.logData(PID_GPS_LONGITUDE, lng);

  // display altitude
  int32_t alt = gps.altitude();
  lcd.setFlags(0);
  if (alt > -1000000 && alt < 1000000) {
    lcd.setCursor(216, 33);
    lcd.print(alt / 100);
    lcd.print("m ");
  }
  // log altitude
```

```
    logger.logData(PID_GPS_ALTITUDE, (int)(alt / 100));

    // display number of satellites
    if (sat < 100) {
        lcd.setCursor(216, 36);
        lcd.printInt(sat);
        lcd.write(' ');
    }

    // only log these data when satellite status is good
    if (sat >= 3) {
        gpsSpeed = gps.speed() * 1852 / 100000;
        logger.logData(PID_GPS_SPEED, gpsSpeed);
    }
}
#endif

void processMEMS()
{


    if (!obd.memsRead(acc, gyro, mag, &temp)) return;

    logger.dataTime = millis();

    acc[0] /= ACC_DATA_RATIO;
    acc[1] /= ACC_DATA_RATIO;
    acc[2] /= ACC_DATA_RATIO;
    gyro[0] /= GYRO_DATA_RATIO;
    gyro[1] /= GYRO_DATA_RATIO;
    gyro[2] /= GYRO_DATA_RATIO;

    // display MEMS data
    lcd.setFontSize(FONT_SIZE_MEDIUM);
    lcd.setCursor(362, 5);
    setColorByValue(acc[0], 50, 100, 200);
    lcd.print(acc[0]);
    setColorByValue(acc[1], 50, 100, 200);
    lcd.write('/');
    lcd.print(acc[1]);
    setColorByValue(acc[2], 50, 100, 200);
    lcd.write('/');
    lcd.print(acc[2]);
    Serial.println(acc[2]);
    lcd.printSpace(8);

    // display gyro data
    lcd.setCursor(374, 10);
    lcd.setColor(RGB16_WHITE);
    lcd.print(gyro[0]);
    lcd.write('/');
```

```
    lcd.print(gyro[1]);
    lcd.write('/');
    lcd.print(gyro[2]);
    lcd.printSpace(8);


    // log x/y/z of accelerometer
    logger.logData(PID_ACC, acc);
    // log x/y/z of gyro meter
    logger.logData(PID_GYRO, gyro);
}

void logOBDData(byte pid, int value)
{
    char buffer[64];
    // send query for OBD-II PID
    logger.dataTime = millis();
    // display data
    showPIDData(pid, value);

    // log data to SD card
    logger.logData(0x100 | pid, value);

    if (pid == PID_SPEED) {
        // estimate distance travelled since last speed update
        distance += (uint32_t)(value + lastSpeed) * (logger.dataTime - lastSpeedTime) / 6000;
        // display speed
        lcd.setFontSize(FONT_SIZE_MEDIUM);
        lcd.setCursor(220, 10);
        lcd.printInt(distance / 1000);
        lcd.write('.');
        lcd.printInt(((uint16_t)distance % 1000) / 100);
        lcd.print(" km");
        // calculate and display average speed
        int avgSpeed = (unsigned long)distance * 3600 / (millis() - startTime);
        lcd.setCursor(220, 15);
        lcd.printInt(avgSpeed);
        lcd.print(" km/h");

        lastSpeed = value;
        lastSpeedTime = logger.dataTime;
    }
#if ENABLE_DATA_LOG
    // flush SD data every 1KB
    byte dataSizeKB = logger.dataSize >> 10;
    if (dataSizeKB != lastFileSize) {
        logger.flushFile();
        lastFileSize = dataSizeKB;
        // display logged data size
        lcd.setFontSize(FONT_SIZE_MEDIUM);
        lcd.setCursor(380, 36);
```

```
        lcd.print((unsigned int)(logger.dataSize >> 10));
        lcd.print("KB");
    }
#endif
}

void processTouch()
{
    int x, y;
    if (lcd.getTouchData(x, y)) {
      Serial.print("X:");
      Serial.print(x);
      Serial.print(" Y:");
      Serial.println(y);
    }
}

void showECUCap()
{
    static const byte PROGMEM pidlist[] = {PID_ENGINE_LOAD, PID_COOLANT_TEMP, PID_FUEL_PRESSURE,
PID_INTAKE_MAP, PID_RPM, PID_SPEED, PID_TIMING_ADVANCE, PID_INTAKE_TEMP, PID_MAF_FLOW,
PID_THROTTLE, PID_AUX_INPUT,
        PID_EGR_ERROR, PID_COMMANDED_EVAPORATIVE_PURGE, PID_FUEL_LEVEL,
PID_CONTROL_MODULE_VOLTAGE, PID_ABSOLUTE_ENGINE_LOAD, PID_AMBIENT_TEMP,
PID_COMMANDED_THROTTLE_ACTUATOR, PID_ETHANOL_FUEL,
        PID_FUEL_RAIL_PRESSURE};

    lcd.setFontSize(FONT_SIZE_MEDIUM);
    lcd.setColor(RGB16_WHITE);
    for (byte i = 0, n = 0; i < sizeof(pidlist) / sizeof(pidlist[0]); i++, n += 2) {
      byte pid = pgm_read_byte(pidlist + i);
      bool valid = obd.isValidPID(pid);
      lcd.setCursor(320, n);
      lcd.setColor(valid ? RGB16_GREEN : RGB16_RED);
      lcd.draw(valid ? tick : cross, 16, 16);
      lcd.setColor(RGB16_WHITE);
      lcd.print(" 0");
      lcd.print((int)pid | 0x100, HEX);
    }
    int values[sizeof(pidlist)];
    bool scanned = false;
    bool touched = false;
    for (uint32_t t = millis(); millis() - t < 5000; ) {
      for (byte i = 0, n = 0; i < sizeof(pidlist) / sizeof(pidlist[0]); i++, n += 2) {
        byte pid = pgm_read_byte(pidlist + i);
        if (obd.isValidPID(pid)) {
          int value;
          lcd.setCursor(392, n);
          if (obd.readPID(pid, value)) {
            if (!scanned || value == values[i])
              lcd.setColor(RGB16_CYAN);
```

```
            else if (value > values[i])
              lcd.setColor(RGB16_BLUE);
            else
              lcd.setColor(RGB16_RED);
            byte n = lcd.print(value);
            for (; n < 4; n++) lcd.print(' ');
            values[i] = value;
          } else {
            lcd.setColor(RGB16_YELLOW);
            lcd.print("N/A");
          }
        }
      }
      scanned = true;
    }
}

void reconnect()
{
    fadeOutScreen();
#if ENABLE_DATA_LOG
    logger.closeFile();
#endif
    lcd.clear();
    state &= ~(STATE_OBD_READY | STATE_GUI_ON);
    //digitalWrite(SD_CS_PIN, LOW);
    for (;;) {
        if (obd.init())
            break;

        obd.enterLowPowerMode();
        Narcoleptic.delay(10000);
        obd.leaveLowPowerMode();
    }
    // re-initialize
    state |= STATE_OBD_READY;
    startTime = millis();
    lastSpeedTime = startTime;
    lastSpeed = 0;
    distance = 0;
#if ENABLE_DATA_LOG
    logger.openFile();
#endif
    initScreen();
}

// screen layout related stuff
void showStates()
{
    lcd.setFontSize(FONT_SIZE_MEDIUM);
    lcd.setColor(RGB16_WHITE);
```

```
    lcd.setCursor(0, 10);
    lcd.print("MEMS ");
    lcd.setColor((state & STATE_MEMS_READY) ? RGB16_GREEN : RGB16_RED);
    lcd.draw((state & STATE_MEMS_READY) ? tick : cross, 16, 16);

#if USE_GPS
    lcd.setColor(RGB16_WHITE);
    lcd.setCursor(60, 10);
    lcd.print(" GPS ");
    if (state & STATE_GPS_CONNECTED) {
        lcd.setColor(RGB16_GREEN);
        lcd.draw(tick, 16, 16);
    } else {
        lcd.setColor(RGB16_RED);
        lcd.draw(cross, 16, 16);
    }
#endif
    lcd.setColor(RGB16_WHITE);
}

void testOut()
{
    const char cmds[][6] = {"ATZ\r", "ATH1\r", "ATRV\r", "0100\r", "0902\r"};
    char buf[128];
    lcd.setFontSize(FONT_SIZE_SMALL);
    lcd.setCursor(0, 13);

    for (byte i = 0; i < sizeof(cmds) / sizeof(cmds[0]); i++) {
      const char* cmd = cmds[i];
      lcd.setColor(RGB16_WHITE);
      lcd.print("Sending ");
      lcd.println(cmd);
      Serial.println(cmd);
      lcd.setColor(RGB16_CYAN);
      if (obd.sendCommand(cmd, buf, sizeof(buf))) {
        char *p = strstr(buf, cmd);
        if (p)
            p += strlen(cmd);
        else
            p = buf;
        Serial.println(p);
        while (*p == '\r') p++;
        while (*p) {
            lcd.write(*p);
            if (*p == '\r' && *(p + 1) != '\r') {
                lcd.write('\n');
            }
            p++;
        }
        lcd.println();
      } else {
```

```
        lcd.println("Timeout");
        Serial.println("Timeout");
      }
      delay(500);
    }
    lcd.println();
}

void setup()
{
  Serial.begin(115200);
#if USE_GPS
    GPSUART.begin(GPS_BAUDRATE);
    lastGPSDataTime = 0;
#endif
    logger.initSender();

    lcd.begin();
    lcd.setFontSize(FONT_SIZE_MEDIUM);
    lcd.setColor(0xFFE0);
    lcd.println("MAI20023 CAR DATA LOGGER");
    lcd.println();
    lcd.setColor(RGB16_WHITE);

#if ENABLE_DATA_LOG
    if (checkSD()) {
        uint16_t index = logger.openFile();
        lcd.println();
        if (index > 0) {
            lcd.print("File ID:");
            lcd.println(index);
        } else {
            lcd.println("No File");
        }
    }
#endif

    byte version = obd.begin();
#ifdef OBD_ADAPTER_I2C
    lcd.print("OBD-II I2C Adapter ");
#else
    lcd.print("OBD-II UART Adapter ");
#endif
      if (version) {
      lcd.print("Ver. ");
      lcd.print(version / 10);
      lcd.print('.');
      lcd.println(version % 10);
    } else {
      lcd.setColor(RGB16_RED);
      lcd.draw(cross, 16, 16);
```

```
    lcd.setColor(RGB16_WHITE);
  }

#ifdef OBD_ADAPTER_I2C
  Wire.begin();
#endif
  if (version && obd.memsInit())
    state |= STATE_MEMS_READY;

  showStates();

#if USE_GPS
  unsigned long t = millis();
  while (GPSUART.available()) GPSUART.read();
  do {
    if (GPSUART.available() && GPSUART.read() == '\r') {
      state |= STATE_GPS_CONNECTED;
      break;
    }
  } while (millis() - t <= 2000);
  showStates();
#endif

  // this will send a bunch of commands and display response
  testOut();

  // initialize the OBD until success
  while (!obd.init(OBD_PROTOCOL));
  state |= STATE_OBD_READY;

  char buf[64];
  if (obd.getVIN(buf, sizeof(buf))) {
    lcd.setFontSize(FONT_SIZE_MEDIUM);
    lcd.setColor(RGB16_WHITE);
    lcd.print("VIN:");
    lcd.setColor(RGB16_YELLOW);
    lcd.println(buf);
  }

  uint16_t dtc[6];
  int num = obd.readDTC(dtc, sizeof(dtc) / sizeof(dtc[0]));
  lcd.setColor(RGB16_WHITE);
  lcd.print(num);
  lcd.println(" DTC found");
  if (num > 0) {
    lcd.setColor(RGB16_YELLOW);
    for (byte i = 0; i < num; i++) {
      lcd.print(dtc[i], HEX);
      lcd.print(' ');
    }
  }
```

```
    lcd.println();

    showECUCap();
    lcd.setCursor(0, 28);
    lcd.setColor(RGB16_YELLOW);
    lcd.setFontSize(FONT_SIZE_MEDIUM);

    fadeOutScreen();
    initScreen();

    startTime = millis();
    lastSpeedTime = startTime;
    lastRefreshTime = millis();
      File dataFile = SD.open("data.csv", FILE_WRITE);

    if (dataFile) {
      // If the file opened successfully, write the CSV header
     dataFile.println("Time,Latitude (deg),Longitude (deg), Engine coolant temperature, Relative throttle position (%), Boost (kPa), Calculated load value (%), Accel X (m/s²), Accel Y (m/s²), Accel Z (m/s²), Accel (Grav) X (m/s²), Accel (Grav) Y (m/s²), Accel (Grav) Z (m/s²), Altitude (m),Voltage
");

      // Close the file for now
      dataFile.close();
    }
}



void loop()
{
    static byte index2 = 0;
    const byte pids[]= {PID_RPM, PID_SPEED, PID_THROTTLE, PID_ENGINE_LOAD};
    const byte pids2[] = {PID_COOLANT_TEMP, PID_INTAKE_TEMP, PID_ENGINE_FUEL_RATE};
    int values[sizeof(pids)] = {0};
    uint32_t pidTime = millis();
    // read multiple OBD-II PIDs
    byte results = obd.readPID(pids, sizeof(pids), values);
    pidTime = millis() - pidTime;
    if (results == sizeof(pids)) {
      for (byte n = 0; n < sizeof(pids); n++) {
        logOBDData(pids[n], values[n]);
      }
    }
    byte pid = pids2[index2 = (index2 + 1) % sizeof(pids2)];
    // check validation and read a single OBD-II PID
    if (obd.isValidPID(pid)) {
      int value;
      if (obd.readPID(pid, value)) {
        logOBDData(pid, value);
      }
```

```
    }

    if (state & STATE_MEMS_READY) {
      processMEMS();
    }

    if (logger.dataTime -  lastRefreshTime >= 1000) {
      float v = obd.getVoltage();
      if (v > 0) {
        lcd.setCursor(84, 24);
        lcd.setFontSize(FONT_SIZE_MEDIUM);
        lcd.print(v, 1);
      }

      char buf[12];
        // display elapsed time
        unsigned int sec = (logger.dataTime - startTime) / 1000;
        sprintf(buf, "%02u:%02u", sec / 60, sec % 60);
        lcd.setFontSize(FONT_SIZE_MEDIUM);
        lcd.setCursor(220, 5);
        lcd.print(buf);
        // display OBD time
        if (results) {
          lcd.setCursor(380, 26);
          lcd.print((uint16_t)(pidTime / results));
          lcd.print("ms ");
        }
        lastRefreshTime = logger.dataTime;
    }

    if (obd.errors >= 3) {
      reconnect();
    }

#if USE_GPS
    if (millis() - lastGPSDataTime > GPS_DATA_TIMEOUT || gps.satellites() < 3) {
      // GPS not ready
      state &= ~STATE_GPS_READY;
    } else {
      // GPS ready
      state |= STATE_GPS_READY;
    }
#endif

// Open the data file in append mode
 File dataFile = SD.open("data.csv", FILE_WRITE);

  if (dataFile) {
    // Get the current timestamp
    unsigned long timestamp = millis();
```

```
    // Write the data as a CSV line
    dataFile.print(logger.dataTime);
     dataFile.print(",");
    dataFile.print((float)lat / 100000, 5);
    dataFile.print(",");
    dataFile.print((float)lng / 100000, 5);
    dataFile.print(",");
    dataFile.print(obd.getVoltage(), 1);
    dataFile.print(",");
    dataFile.print(PID_COOLANT_TEMP);
    dataFile.print(",");
    dataFile.print(values[0]);
    dataFile.print(",");
    dataFile.print(values[1]);
    dataFile.print(",");
    dataFile.print(values[2]);
    dataFile.print(",");
    dataFile.print(values[3]);
    dataFile.print(",");
    dataFile.print(acc[0]);
    dataFile.print(",");
    dataFile.print(acc[1]);
    dataFile.print(",");
    dataFile.print(acc[2]);
    dataFile.print(",");
    dataFile.print(gyro[0]);
    dataFile.print(",");
    dataFile.print(gyro[1]);
    dataFile.print(",");
    dataFile.print(gyro[2]);
    dataFile.print(",");
    dataFile.print(mag[0]);
    dataFile.print(",");
    dataFile.print(mag[1]);
    dataFile.print(",");
    dataFile.print(mag[2]);
    dataFile.print(",");


    // Close the file
    dataFile.close();
  } else {
    Serial.println("Error opening data.csv for writing.");
  }
}
```

- **Datalog.h**

```
* Freematics Data Logger Class
* Distributed under GPL v2.0
* Written by Stanley Huang <stanleyhuangyc@gmail.com>

#define FORMAT_BIN 0
#define FORMAT_TEXT 1

typedef struct {
    uint32_t time;
    uint16_t pid;
    uint8_t flags;
    uint8_t checksum;
    float value[3];
} LOG_DATA_COMM;

#define PID_GPS_LATITUDE 0xA
#define PID_GPS_LONGITUDE 0xB
#define PID_GPS_ALTITUDE 0xC
#define PID_GPS_SPEED 0xD
#define PID_GPS_HEADING 0xE
#define PID_GPS_SAT_COUNT 0xF
#define PID_GPS_TIME 0x10
#define PID_GPS_DATE 0x11

#define PID_ACC 0x20
#define PID_GYRO 0x21
#define PID_COMPASS 0x22
#define PID_MEMS_TEMP 0x23
#define PID_BATTERY_VOLTAGE 0x24

#define PID_DATA_SIZE 0x80

#define FILE_NAME_FORMAT "DAT%05d.CSV"
#define FILE_PATH "/DATA/"

#if ENABLE_DATA_OUT

#if USE_SOFTSERIAL
SoftwareSerial SerialRF(A2, A3);
#elif defined(RF_SERIAL)
#define SerialRF RF_SERIAL
#else
#define SerialRF Serial
#endif
```

```
#endif

#if ENABLE_DATA_LOG
static File sdfile;
#endif

typedef struct {
   uint8_t pid;
   char name[3];
} PID_NAME;

const PID_NAME pidNames[] PROGMEM = {
{PID_ACC, {'A','C','C'}},
{PID_GYRO, {'G','Y','R'}},
{PID_COMPASS, {'M','A','G'}},
{PID_GPS_LATITUDE, {'L','A','T'}},
{PID_GPS_LONGITUDE, {'L','N','G'}},
{PID_GPS_ALTITUDE, {'A','L','T'}},
{PID_GPS_SPEED, {'S','P','D'}},
{PID_GPS_HEADING, {'C','R','S'}},
{PID_GPS_SAT_COUNT, {'S','A','T'}},
{PID_GPS_TIME, {'U','T','C'}},
{PID_GPS_DATE, {'D','T','E'}},
{PID_BATTERY_VOLTAGE, {'B','A','T'}},
{PID_DATA_SIZE, {'D','A','T'}},
};

class CDataLogger {
public:
   CDataLogger()
   {
      m_lastDataTime = 0;
#if ENABLE_DATA_CACHE
      cacheBytes = 0;
#endif
   }
   void initSender()
   {
#if ENABLE_DATA_OUT
      SerialRF.begin(STREAM_BAUDRATE);
#endif
   }
   byte genTimestamp(char* buf, bool absolute)
   {
     byte n;
     if (absolute || dataTime >= m_lastDataTime + 60000) {
       // absolute timestamp
       n = sprintf(buf, "#%lu,", dataTime);
     } else {
       // incremental timestamp
```

```
      n = sprintf(buf, "%u,", (unsigned int)(dataTime - m_lastDataTime));
    }
    return n;
  }
  void record(const char* buf, byte len)
  {
#if ENABLE_DATA_LOG
#if STREAM_FORMAT == FORMAT_BIN
    dataSize += sdfile.write(buf, len);
#else
    char tmp[12];
    byte n = genTimestamp(tmp, dataSize == 0);
    dataSize += sdfile.write(tmp, n);
    dataSize += sdfile.write(buf, len);
    sdfile.println();
    dataSize += 3;
#endif
#endif
    m_lastDataTime = dataTime;
  }
  void dispatch(const char* buf, byte len)
  {
#if ENABLE_DATA_CACHE
    if (cacheBytes + len < MAX_CACHE_SIZE - 10) {
      cacheBytes += genTimestamp(cache + cacheBytes, cacheBytes == 0);
      memcpy(cache + cacheBytes, buf, len);
      cacheBytes += len;
      cache[cacheBytes++] = '\n';
      cache[cacheBytes] = 0;
    }
#endif
#if ENABLE_DATA_OUT
    SerialRF.write(buf, len);
    SerialRF.println();
#endif
  }
  void logData(const char* buf, byte len)
  {
#if ENABLE_DATA_OUT
#if STREAM_FORMAT != FORMAT_BIN
    dispatch(buf, len);
#endif
#endif
    record(buf, len);
  }
  void logData(uint16_t pid)
  {
    char buf[8];
    byte len = translatePIDName(pid, buf);
#if ENABLE_DATA_OUT
#if STREAM_FORMAT != FORMAT_BIN
```

```
        dispatch(buf, len);
#endif
#endif
        record(buf, len);
    }
    void logData(uint16_t pid, int value)
    {
        char buf[16];
        byte n = translatePIDName(pid, buf);
        byte len = sprintf(buf + n, "%d", value) + n;
#if STREAM_FORMAT == FORMAT_BIN
        LOG_DATA_COMM ld = {dataTime, pid, 1, 0, value};
        ld.checksum = getChecksum((char*)&ld, 12);
        dispatch((const char*)&ld, 12);
#else
        dispatch(buf, len);
#endif
        record(buf, len);
    }
    void logData(uint16_t pid, int32_t value)
    {
        char buf[20];
        byte n = translatePIDName(pid, buf);
        byte len = sprintf(buf + n, "%ld", value) + n;
#if ENABLE_DATA_OUT
#if STREAM_FORMAT == FORMAT_BIN
        LOG_DATA_COMM ld = {dataTime, pid, 1, 0, value};
        ld.checksum = getChecksum((char*)&ld, 12);
        dispatch((const char*)&ld, 12);
#else
        dispatch(buf, len);
#endif
#endif
        record(buf, len);
    }
    void logData(uint16_t pid, uint32_t value)
    {
        char buf[20];
        byte n = translatePIDName(pid, buf);
        byte len = sprintf(buf + n, "%lu", value) + n;
#if STREAM_FORMAT == FORMAT_BIN
        LOG_DATA_COMM ld = {dataTime, pid, 1, 0, value};
        ld.checksum = getChecksum((char*)&ld, 12);
        dispatch((const char*)&ld, 12);
#else
        dispatch(buf, len);
#endif
        record(buf, len);
    }
    void logData(uint16_t pid, int values[])
    {
```

```
        char buf[24];
        byte n = translatePIDName(pid, buf);
        byte len = sprintf(buf + n, "%d,%d,%d", values[0], values[1], values[2]) + n;
#if STREAM_FORMAT == FORMAT_BIN
        LOG_DATA_COMM ld = {dataTime, pid, 3, 0, {values[0], values[1], values[2]}};
        ld.checksum = getChecksum((char*)&ld, 20);
        dispatch((const char*)&ld, 20);
#else
        dispatch(buf, len);
#endif
        record(buf, len);
    }
#if ENABLE_DATA_LOG
    uint16_t openFile(uint16_t logFlags = 0, uint32_t dateTime = 0)
    {
        uint16_t fileIndex;
        char filename[24] = FILE_PATH;

        dataSize = 0;
        if (SD.exists(filename)) {
            for (fileIndex = 1; fileIndex; fileIndex++) {
                sprintf(filename + sizeof(FILE_PATH) - 1, FILE_NAME_FORMAT, fileIndex);
                if (!SD.exists(filename)) {
                    break;
                }
            }
            if (fileIndex == 0)
                return 0;
        } else {
            SD.mkdir(filename);
            fileIndex = 1;
            sprintf(filename + sizeof(FILE_PATH) - 1, FILE_NAME_FORMAT, 1);
        }

        sdfile = SD.open(filename, FILE_WRITE);
        if (!sdfile) {
            return 0;
        }
        m_lastDataTime = dateTime;
        return fileIndex;
    }
    void closeFile()
    {
        sdfile.close();
    }
    void flushFile()
    {
        sdfile.flush();
    }
#endif
    uint32_t dataTime;
```

```
    uint32_t dataSize;
#if ENABLE_DATA_CACHE
    char cache[MAX_CACHE_SIZE];
    int cacheBytes;
#endif
private:
    byte getChecksum(char* buffer, byte len)
    {
        uint8_t checksum = 0;
        for (byte i = 0; i < len; i++) {
          checksum ^= buffer[i];
        }
        return checksum;
    }
    byte translatePIDName(uint16_t pid, char* text)
    {
#if STREAM_FORMAT == FORMAT_TEXT && USE_FRIENDLY_PID_NAME
        for (uint16_t n = 0; n < sizeof(pidNames) / sizeof(pidNames[0]); n++) {
            uint16_t id = pgm_read_byte(&pidNames[n].pid);
            if (pid == id) {
                memcpy_P(text, pidNames[n].name, 3);
                text[3] = ',';
                return 4;
            }
        }
#endif
        return sprintf(text, "%X,", pid);
    }
    uint32_t m_lastDataTime;
};
```

- **setup.h**

```
#ifndef CONFIG_H_INCLUDED
#define CONFIG_H_INCLUDED

/*****************************************
* OBD-II Adapter options
*****************************************/
#define OBD_ADAPTER_MODEL OBD_MODEL_I2C
//#define OBD_ADAPTER_MODEL OBD_MODEL_UART
#define OBD_PROTOCOL PROTO_AUTO

/*****************************************
* Data logging options
```

```
**************************************/
// enable(1)/disable(0) data logging (if SD card is present)
#define ENABLE_DATA_LOG 1
#define SD_CS_PIN SS

/**************************************
* Data streaming options
**************************************/
// enable(1)/disable(0) data streaming
#define ENABLE_DATA_OUT 1

// uses software(1)/hardware(0) serial for data streaming
#define USE_SOFTSERIAL 0
#define RF_SERIAL Serial
#define STREAM_BAUDRATE 115200 /* bps */

// this defines the format of data streaming
// FORMAT_BIN is required by Freematics OBD iOS App
// FORMAT_TEXT for text-based, text names for PID
#define STREAM_FORMAT FORMAT_TEXT

/**************************************
* GPS configuration
**************************************/
#define USE_GPS 1
#define GPSUART Serial2
#define MAX_GPS_PROCESS_TIME 50 /* ms */
#define GPS_DATA_TIMEOUT 2000 /* ms */

// GPS baudrate could be 38400bps or 115200bps
#define GPS_BAUDRATE 115200 /* bps */

/**************************************
* Accelerometer & Gyro
**************************************/
#define USE_MPU6050 1
//#define USE_MPU9150 1
#define ACC_DATA_RATIO 160
#define GYRO_DATA_RATIO 256
#define COMPASS_DATA_RATIO 8

/**************************************
* Timeout/interval options
**************************************/
#define ACC_DATA_INTERVAL 200 /* ms */

/**************************************
* LCD module (uncomment only one)
**************************************/
LCD_R61581 lcd; /* 3.5" CTE35IPS/R61581 based LCD */
//LCD_SSD1289 lcd; /* 3.2" SSD12389 based LCD */
```

```
//LCD_ILI9325D lcd; /* 2.8" ILI9325 based LCD */
//LCD_ILI9341 lcd; /* 2.4" ILI9341 based SPI LCD */
//LCD_Null lcd;

#endif
```