ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Παρακολούθηση δεδομένων οχήματος με χρήση Arduino ADK και Android smartphone

Διπλωματική Εργασία

Σκοτίδας Αναστάσιος (159)
08/10/2013

Περιεχόμενα

Περιγραφή Διπλωματικής 2

Στόχος Project 3

Βασικές πληροφορίες οχήματος 3

Πλοήγηση και Τηλεματική 4

Εικόνα και ήχος 4

Social Media και Ενημέρωση 4

Τηλέφωνο 5

Η Διπλωματική μου 6

ΓΕΝΙΚΗ ΛΟΓΙΚΗ 6

Επί μέρους ανάλυση 7

ECU & OBD 7

OBD-II UART ADAPTER FOR ARDUINO 18

Συμβατότητα 20

Διαδικασία 20

Arduino ADK 35

Specifications Summary 35

Μνήμη 36

Προγραμματισμός 36

Arduino Software 36

Android Operating System 37

Κώδικας Android 41

Κώδικας Layout 45

Κώδικας accessory_filter.xml 48

Μελλοντικές προεκτάσεις 49

Βιβλιογραφία & Πηγές 49

Περιγραφή Διπλωματικής

Η Διπλωματική Εργασία αποτελεί μία πρόταση σχετικά με ένα διαφορετικό, σύγχρονο τρόπο πληροφόρησης του οδηγού σχετικά με διάφορα δεδομένα του οχήματος στα πλαίσια του Android ecosystem. Με τη χρήση πολύ οικονομικού hardware, η πληροφόρηση του οδηγού σχετικά με τα σημαντικά δεδομένα του οχήματος μπορεί να περάσει σε νέο επίπεδο, προσφέροντας μία σειρά από δυνατότητες συγκεντρωμένες σε μία μόνο συσκευή.

Σε μία εποχή που η πληροφόρηση διαδραματίζει σημαντικό ρόλο στην καθημερινότητά μας, η ενοποίηση των συσκευών που την παρέχουν αποτελεί άμεση προτεραιότητα. Το open source ecosystem στο οποίο λειτουργεί το Android έδωσε σημαντική ώθηση στην επίτευξη αυτού του στόχου καθώς έδωσε την ευκαιρία στις εταιρίες και τους developers να εφαρμόσουν στην πράξη τις προτάσεις τους, παρουσιάζοντας λύσεις που βρίσκουν ολοένα και μεγαλύτερη εφαρμογή. Με την ελεύθερη χρήση προγραμμάτων όπως το Eclipse, η υλοποίηση πλήθους εφαρμογών που στηρίζουν αυτό το ecosystem καθίσταται σχετικά εύκολη κάνοντας έτσι όλο και περισσότερους developers να ασχοληθούν με αυτό το αντικείμενο.

Παράλληλα με την ελεύθερη ανάπτυξη του software μειώθηκε σημαντικά το κόστος του hardware. Χαρακτηριστικό παράδειγμα αποτελεί το σύνολο των πλακετών Arduino καθώς και των εταιριών που παράγουν αντίστοιχες πλακέτες συχνά με ακόμη μικρότερο κόστος. Σε συνέχεια των πλακετών αυτών δημιουργήθηκε μία ευρεία μάζα hardware υποστήριξης όπως αισθητήρες, modules, adapters κ.α. Το χαμηλό κόστος, ο ανοικτός κώδικας προγραμματισμού τους και η εύκολη πρόσβαση σε αυτού του είδους το hardware οδήγησε σε μία κοινότητα που υλοποιεί διαρκώς καινούριες λύσεις και αυξάνεται συνεχώς.

Οι αυτοκινηστικές βιομηχανίες αποτελούν σημαντικό κομμάτι της αγοράς παράγοντας και πουλώντας σε υψηλές τιμές. Επιπλέον, διαμορφώνουν τις τιμές του κάθε μοντέλου αυτοκινήτου ανάλογα και με τις παροχές που αυτό προσφέρει και που δεν έχουν άμεση σχέση με την οδήγηση αυτη καθ’αυτή. Προκειμένου να μειώσουν αυτή την τιμή προσπαθούν να μειώσουν το κόστος παραγωγής ώστε να κάνουν πιο ελκυστικό το προϊόν τους, επιδειώκοντας παράλληλα να προσφέρουν καινοτόμες λύσεις.

Στα πλαίσια αυτής της προσπάθειας διέκρινα ότι η μετάβαση, από το τελείως close περιβάλλον στο οποίο λειτουργούν οι αυτοκινητιστικές βιομηχανίες, σε ένα open source αποτελεί μονόδρομο. Η μετάβαση αυτή δημιουργεί πολλά οφέλη σε αυτές τις βιομηχανίες, οικονομικά και μη. Από τη μία, μειώνεται σημαντικά το κόστος όσον αφορά τις διάφορες συσκευές του αυτοκινήτου. Το καντράν, ο audio player, ο video player και ο πλοηγός αντικαθίστανται από μόνο μία συσκευή το κόστος της οποίας σπάνια ξεπερνάει τα 350 ευρώ σύμφωνα με τις σημερινές τιμές. Επιπρόσθετα, κόστος εξοικονομείται και από το ανθρώπινο δυναμικό. Από τη στιγμή που ένας μεγάλος αριθμός developers θα ενδιαφερθεί να προσφέρει λύσεις στο συγκεκριμένο σύστημα, μειώνεται η ανάγκη για απασχόληση προσωπικού από την ίδια την εταιρία παραγωγής αυτοκινήτων.

Τέλος, σημαντικά είναι τα οφέλη που προκύπτουν στο κομμάτι της καινοτομίας. Για πρώτη φορά ο οδηγός θα έχει τη δυνατότητα να έχουν διαρκή πρόσβαση στην ενημέρωση και τα social media μειώνοντας κατακόρυφα τους κινδύνους σχετικά με την ασφάλεια. Επιπλέον, δίνεται η ελευθερία στον οδηγό να αποφασίσει και να διαμορφώσει ο ίδιος, ανάλογα με τις ανάγκες του, το περιβάλλον από το οποίο θα αντλεί όλες τις πληροφορίες. Χαρακτηριστικά, θα μπορεί να χρησιμοποιεί τον πλοηγό της επιλογής του, τον τρόπο με τον οποίο επιθυμεί να αναπαρίστανται τα δεδομένα του οχήματος κ.α.

Στόχος Project

Στόχος του project είναι η αφομοίωση της σύγχρονης τάσης στην τεχνολογία από τις αυτοκινητιστικές βιομηχανίες. Σύμφωνα με αυτή την τάση, η μέγιστη πληροφόρηση του χρήστη πρέπει να γίνεται όσο το δυνατό από λιγότερες συσκευές. Σήμερα, ένα αυτοκίνητο μπορεί να περιλαμβάνει το κλασσικό ταμπλό για την εμφάνιση δεδομένων όπως η ταχύτητα του οχήματος, οι στροφές του κινητήρα, η στάθμη καυσίμων κ.α., ένα πάνελ όπου γίνεται η πλοήγηση του οχήματος, είτε αυτό είναι ενσωματωμένο είτε όχι (εξωτερική συσκευή GPS), καθώς και τις συσκευές αναπαραγωγής ήχου και εικόνας (mp3 & DVD players).

Η προτεινόμενη λύση ενοποιεί όλα τα παραπάνω σε μία μόνο ταμπλέτα (tablet). Οι ταμπλέτες έχουν αποκτήσει σημαντικό μερίδιο της αγοράς και έχουν ολοένα περισσότερη διύσδειση. Πρόκειται για μία οικονομική συσκευή η οποία στα πλαίσια του λειτουργικού συστήματος Android μετατρέπεται σε μία ισχυρή λύση σε διάφορους τομείς.

Με τη χρήση της ταμπλέτας και τη λειτουργία της στα πλαίσια του Android ecosystem η εμπειρία οδήγησης αλλά και του ταξιδιού αλλάζει σημαντικά. Αναλυτικά:

Βασικές πληροφορίες οχήματος

Όλες οι βασικές πληροφορίες όπως η ταχύτητα του οχήματος, οι στροφές του κινητήρα, το επίπεδο καυσίμων, η πίεση λαδιών, τα διανυθέντα χιλιόμετρα κ.α. θα μπορούν να παρέχονται και να αναπαρίστανται με τον τρόπο που επιθυμεί ο ίδιος ο οδηγός ανάλογα με το application που έχει εγκαταστήσει στην ταμπλέτα. Θα δέχεται πληροφορίες που δεν προσφέρονται μέχρι σήμερα όπως ιστορικά στοιχεία των διαδρομών του, μέρη που επισκέπτεται συχνότερα, κατανάλωση βενζίνης μέσα σε ένα ορισμένο χρονικό διάστημα, συμβουλές οικονομικής οδήγησης, γραφική απεικόνιση του οχήματος.

Πλοήγηση και Τηλεματική

Ο χρήστης θα έχει τη δυνατότητα να χρησιμοποιεί τον πλοηγό της αρεσκείας του σύμφωνα με τις ανάγκες του και τις επιθυμίες του, χωρίς να χρειάζεται να αγοράσει ξεχωριστή συσκευή GPS (Navigator).

Επιπρόσθετα, αν η ταμπλέτα διαθέτει κάρτα SIM θα μπορεί να παρακολουθεί τα δεδομένα του αυτοκινήτου αλλά και τη γεωγραφική του θέση από απομακρυσμένο smartphone ή ταμπλέτα γνωρίζοντας έτσι την κατάσταση του οχήματος ανά πάσα στιγμή.

Εικόνα και ήχος

Η αναπαραγωγή ήχου όπως το ράδιο και τα mp3 κομμάτια θα γίνονται από την ταμπλέτα αντικαθιστώντας πλήρως τον κλασσικό audio player προσφέροντας πολλές πρόσθετες δυνατότητες. Ο χρήστης αποδεσμεύεται από την ανάγκη να διαθέτει μέσο αποθήκευσης ακουστικών κομματιών (CD, USB flash drives) καθώς θα τα αναπαράγει από την ταμπλέτα. Επίσης, του δίνεται η δυνατότητα να ακούσει ραδιοφωνικούς σταθμούς που εκπέμπουν εκτός της δικής τους εμβέλειας μέσω e-radio.

Μία ακόμη συσκευή που αντικαθίσταται είναι ο DVD player με την ταμπλέτα να αναλαμβάνει και αυτό το ρόλο. Τα τελευταία χρόνια, ένα σημαντικό ποσοστό κατόχων αυτοκινήτων, στράφηκε στην αγορά συσκευών αναπαραγωγής εικόνας οι οποίες χαρακτηρίζονται από το υψηλό τους κόστος, περιορίζοντας έτσι το αριθμό των ατόμων που έχουν την ευχέρεια να τις αγοράσουν. Μέσω της προτεινόμενης λύσης, ο καθένας θα μπορεί να αναπαράγει εικόνα με μηδενικό κόστος.

Social Media και Ενημέρωση

Αναμφίβολα τα social media έχουν εισβάλει στην καθημερινότητά μας. Μέχρι τώρα, η συμμετοχή σε αυτά την ώρα της οδήγησης είναι αδύνατη και όταν γίνεται αποτελεί μεγάλο κίνδυνο αυξάνοντας σημαντικά τις πιθανότητες πρόκλησης αυτοκινητιστικού ατυχήματος.

Τη λύση σε αυτό το πρόβλημα έρχεται να δώσει η τεχνολογία αναγνώρισης φωνής. Ο οδηγός θα μπορεί να δραστηριοποιηθεί στα διάφορα social media δίνοντας φωνητικές εντολές και αντίστοιχα η ενημέρωση που θα δέχεται θα γίνεται μέσω ηχητικής αναγγελίας. Έτσι, χωρίς να χρειάζεται να ασχοληθεί με άλλη συσκευή αλλά και χωρίς να χρειάζεται να απομακρύνει τα χέρια του από το τιμόνι, ο χρήστης θα μπορεί να παρακολουθεί τις εξελίξεις στις σελίδες κοινωνικής δικτύωσης, να ενημερώνεται σχετικά με ειδήσεις που τον αφορούν κ.α.

Τηλέφωνο

Η τεχνολογία που αναφέρθηκε προηγουμένως βρίσκει εφαρμογή και σε ένα ακόμη σημαντικό θέμα όπως είναι αυτό της χρήσης κινητού τηλεφώνου για κλήσεις κατά τη διάρκεια της οδήγησης. Αν η ταμπλέτα διαθέτει κάρτα SIM θα μπορεί να πραγματοποιεί κλήσεις μειώνοντας έτσι τον κίνδυνο από τη χρήση του κινητού τηλεφώνου. Ας μην ξεχνάμε ότι η χρήση κινητού τηλεφώνου αυξάνει αρκετά τον κίνδυνο πρόκλησης τροχαίου ατυχήματος.

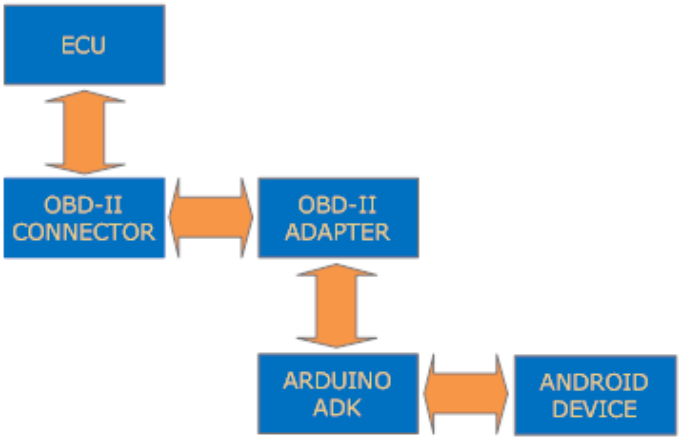
Όλα όσα αναφέρθηκαν παραπάνω έχουν ένα σημαντικό πλεονέκτημα. Είναι άμεσα υλοποιήσιμα χωρίς την ανάγκη εκτεταμένης έρευνας και ανάπτυξης. Η προτεινόμενη λύση δεν απαιτεί τη δημιουργία νέων τεχνολογιών. Αντιθέτως, παντρεύει ιδανικά τεχνολογίες που έχουν ευρεία εφαρμογή μειώνοντας έτσι δραματικά το χρόνο και το κόστος ανάπτυξης.

Η Διπλωματική μου

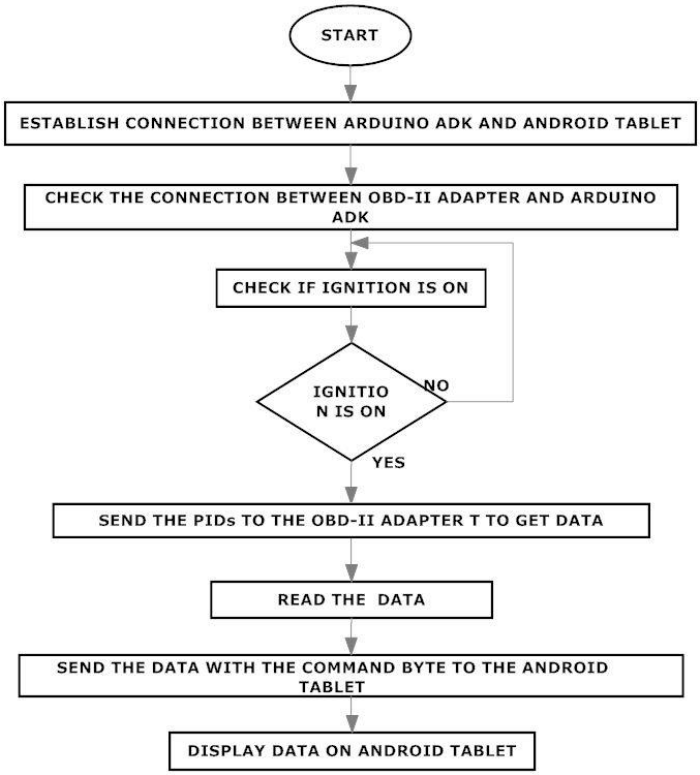
Η Διπλωματική μου επικεντρώνεται στο κομμάτι των βασικών πληροφοριών του οχήματος και την αναπαράστασή τους. Για την υλοποίηση της χρησιμοποιήθηκαν τρεις συσκευές hardware (OBD-II UART Adapter for Arduino, Arduino ADK, Android smartphone) και τρεις γλώσσες προγραμματισμού (Java, C oriented, XML).

ΓΕΝΙΚΗ ΛΟΓΙΚΗ

Ο OBD-II UART Adapter λαμβάνει τα δεδομένα από τον εγκέφαλο του αυτοκινήτου (ECU) μέσω του συστήματος OBD και τα αποστέλει στο Arduino ADK το οποίο με τη σειρά του τα αποστέλει στο Android phone/tablet για την τελική γραφική αναπαράστασή τους.



Ο Αλγόριθμος του προγράμματος απεικονίζεται παρακάτω:



Επί μέρους ανάλυση

ECU & OBD

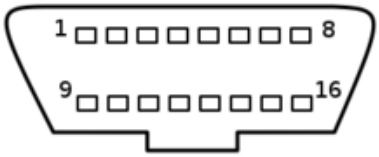
Τα αρχικά ECU προκύπτουν από το Engine Control Unit. Η ECU αποτελεί τον εγκέφαλο του αυτοκινήτου, δηλαδή τη μονάδα εκείνη που επεξεργάζεται τα διάφορα δεδομένα του οχήματος και είναι υπεύθυνη για τις ενδείξεις στο καντράν αλλά και για λειτουργίες όπως τα συστήματα ESP και ABS καθώς και των αισθητήρων παρκαρίσματος.

Η ECU επιτρέπει μόνο την ανάγνωση δεδομένων από εξωτερικό χρήστη και όχι την εγγραφή νέων κυρίως για λόγους ασφαλείας. Για την επικοινωνία χρησιμοποιείται το σύστημα OBD (On-Board Diagnostics).

Το OBD είναι ένα σύστημα που δίνει πρόσβαση (στους κατόχους των αυτοκινήτων και τους τεχνικούς που αναλαμβάνουν την επισκευή τους) στην κατάσταση των επί μέρους υποσυστημάτων του οχήματος στους κατόχους των αυτοκινήτων και τους τεχνικούς που

αναλαμβάνουν την επισκευή τους. Το πλήθος των πληροφοριών που παρέχονται μέσω του OBD αυξήθηκε σημαντικά από το 1980 και μετά όταν και εμφανίσθηκαν οι πρώτοι on-board υπολογιστές οχήματος.

Η ανάγνωση δεδομένων γίνεται μέσω του OBD-II diagnostic connector. Ο OBD-II diagnostic connector διαθέτει 16 female pin-connectors η χρήση του καθενός φαίνεται παρακάτω:



1. Manufacturer discretion. GM: J2411 GMLAN/SWC/Single-Wire CAN. VW/Audi: Switched +12 to tell a scan tool whether the ignition is on.	**9**. - Manufacturer discretion. GM: 8192 baud ALDL where fitted.
2. Bus positive Line of SAE-J1850 PWM and SAE-1850 VPW	**10**. Bus negative Line of SAE-J1850 PWM only (not SAE-1850 VPW)
3. Ford DCL(+) Argentina, Brazil (pre OBD-II) 1997-2000, USA, Europe, etc. Chrysler CCD Bus(+)	11. Ford DCL(-) Argentina, Brazil (pre OBD-II) 1997-2000, USA, Europe, etc. Chrysler CCD Bus(-)
4. Chassis ground	**12**. -
5. Signal ground	**13**. -
6. CAN high (ISO 15765-4 and SAE-J2284)	**14**. CAN low (ISO 15765-4 and SAE-J2284)
7. K line of ISO 9141-2 and ISO 14230-4	**15**. L line of ISO 9141-2 and ISO 14230-4
8. - Manufacturer discretion. Many BMWs: A second K-Line for non OBD-II (Body/Chassis/Infotainment) systems.	**16**. Battery voltage

Το OBD υποστηρίζει πέντε signal protocols με το κάθε αυτοκίνητο να υποστηρίζει ένα από αυτά. Τα πρωτόκολλα αυτά είναι τα εξής:

- SAE J1850 PWM (pulse-width modulation — 41.6 kB/sec, standard of the Ford Motor Company)
    - pin 2: Bus+
    - pin 10: Bus–

  - High voltage is +5 V
  - Message length is restricted to 12 bytes, including CRC
  - Employs a multi-master arbitration scheme called 'Carrier Sense Multiple Access with Non-Destructive Arbitration' (CSMA/NDA)
- SAE J1850 VPW (variable pulse width — 10.4/41.6 kB/sec, standard of General Motors)
  - pin 2: Bus+
  - Bus idles low
  - High voltage is +7 V
  - Decision point is +3.5 V
  - Message length is restricted to 12 bytes, including CRC
  - Employs CSMA/NDA
- ISO 9141-2. This protocol has an asynchronous serial data rate of 10.4 kBaud. It is somewhat similar to RS-232; however, the signal levels are different, and communications happens on a single, bidirectional line without additional handshake signals. ISO 9141-2 is primarily used in Chrysler, European, and Asian vehicles.
  - pin 7: K-line
  - pin 15: L-line (optional)
  - UART signaling
  - K-line idles high, with a 510 ohm resistor to V_{batt}
  - The active/dominant state is driven low with an open-collector driver.
  - Message length is restricted to 12 bytes, including CRC
- ISO 14230 KWP2000 (Keyword Protocol 2000)
  - pin 7: K-line
  - pin 15: L-line (optional)
  - Physical layer identical to ISO 9141-2
  - Data rate 1.2 to 10.4 kBaud
  - Message may contain up to 255 bytes in the data field
- ISO 15765 CAN (250 kBit/s or 500 kBit/s). The CAN protocol was developed by Bosch for automotive and industrial control. Unlike other OBD protocols, variants are widely use outside of the automotive industry. While it did not meet the OBD-II requirements for U.S. vehicles prior to 2003, as of 2008 all vehicles sold in the US are required to implement CAN as one of their signaling protocols.
  - pin 6: CAN High
  - pin 14: CAN Low

Οι διάφορες παράμετροι που είναι διαθέσιμες χαρακτηρίζονται από τους PIDs (Parameter Identification Numbers). Οι κατασκευαστές δεν είναι υποχρεωμένοι να παρέχουν όλα τα PIDs όπως επίσης μπορούν παρέχουν και τα δικά τους PIDs. Τα βασικά PIDs εμφανίζονται παρακάτω:

Mode (hex)	PID (hex)	Data bytes returned	Description	Min value	Max value	Units	Formula †
01	00	4	PIDs supported [01 - 20]				Bit encoded [A7..D0] == [PID $01..PID $20]

01	01	4	Monitor status since DTCs cleared. (Includes malfunction indicator lamp (MIL) status and number of DTCs.)				Bit encoded.
01	02	2	Freeze DTC				
01	03	2	Fuel system status				Bit encoded.
01	04	1	Calculated engine load value	0	100	%	A*100/255
01	05	1	Engine coolant temperature	-40	215	°C	A-40
01	06	1	Short term fuel % trim—Bank 1	-100 Subtracting Fuel (Rich Condition)	99.22 Adding Fuel (Lean Condition)	%	(A-128) * 100/128
01	07	1	Long term fuel % trim—Bank 1	-100 Subtracting Fuel (Rich Condition)	99.22 Adding Fuel (Lean Condition)	%	(A-128) * 100/128
01	08	1	Short term fuel % trim—Bank 2	-100 Subtracting Fuel (Rich Condition)	99.22 Adding Fuel (Lean Condition)	%	(A-128) * 100/128
01	09	1	Long term fuel % trim—Bank 2	-100 Subtracting Fuel (Rich Condition)	99.22 Adding Fuel (Lean Condition)	%	(A-128) * 100/128
01	0A	1	Fuel pressure	0	765	kPa (gauge)	A*3
01	0B	1	Intake manifold absolute pressure	0	255	kPa (absolute)	A
01	0C	2	Engine RPM	0	16,383.75	rpm	((A*256)+B)/4
01	0D	1	Vehicle speed	0	255	km/h	A
01	0E	1	Timing advance	-64	63.5	° relative to #1 cylinder	A/2 - 64
01	0F	1	Intake air temperature	-40	215	°C	A-40
01	10	2	MAF air flow rate	0	655.35	grams/sec	((A*256)+B) / 100
01	11	1	Throttle position	0	100	%	A*100/255
01	12	1	Commanded secondary air status				Bit encoded.
01	13	1	Oxygen sensors present				[A0..A3] == Bank 1, Sensors 1-4. [A4..A7] == Bank 2...
01	14	2	Bank 1, Sensor 1: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
01	15	2	Bank 1, Sensor 2: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)

01	16	2	Bank 1, Sensor 3: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
01	17	2	Bank 1, Sensor 4: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
01	18	2	Bank 2, Sensor 1: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
01	19	2	Bank 2, Sensor 2: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
01	1A	2	Bank 2, Sensor 3: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
01	1B	2	Bank 2, Sensor 4: Oxygen sensor voltage, Short term fuel trim	0 -100(lean)	1.275 99.2(rich)	Volts %	A/200 (B-128) * 100/128 (if B==$FF, sensor is not used in trim calc)
01	1C	1	OBD standards this vehicle conforms to				Bit encoded.
01	1D	1	Oxygen sensors present				Similar to PID 13, but [A0..A7] == [B1S1, B1S2, B2S1, B2S2, B3S1, B3S2, B4S1, B4S2]
01	1E	1	Auxiliary input status				A0 == Power Take Off (PTO) status (1 == active) [A1..A7] not used
01	1F	2	Run time since engine start	0	65,535	seconds	(A*256)+B
01	20	4	PIDs supported [21 - 40]				Bit encoded [A7..D0] == [PID $21..PID $40]
01	21	2	Distance traveled with malfunction indicator lamp (MIL) on	0	65,535	km	(A*256)+B
01	22	2	Fuel Rail Pressure (relative to manifold vacuum)	0	5177.265	kPa	((A*256)+B) * 0.079
01	23	2	Fuel Rail Pressure (diesel, or gasoline direct inject)	0	655,350	kPa (gauge)	((A*256)+B) * 10
01	24	4	O2S1_WR_lambda(1): Equivalence Ratio Voltage	0 0	1.999 7.999	N/A V	((A*256)+B)*2/65535 or ((A*256)+B)/32768 ((C*256)+D)*8/65535 or ((C*256)+D)/8192
01	25	4	O2S2_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535

01	26	4	O2S3_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
01	27	4	O2S4_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
01	28	4	O2S5_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
01	29	4	O2S6_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
01	2A	4	O2S7_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
01	2B	4	O2S8_WR_lambda(1): Equivalence Ratio Voltage	0 0	2 8	N/A V	((A*256)+B)*2/65535 ((C*256)+D)*8/65535
01	2C	1	Commanded EGR	0	100	%	A*100/255
01	2D	1	EGR Error	-100	99.22	%	(A-128) * 100/128
01	2E	1	Commanded evaporative purge	0	100	%	A*100/255
01	2F	1	Fuel Level Input	0	100	%	A*100/255
01	30	1	# of warm-ups since codes cleared	0	255	N/A	A
01	31	2	Distance traveled since codes cleared	0	65,535	km	(A*256)+B
01	32	2	Evap. System Vapor Pressure	-8,192	8,192	Pa	((A*256)+B)/4 (A and B are two's complement signed)
01	33	1	Barometric pressure	0	255	kPa (Absolute)	A
01	34	4	O2S1_WR_lambda(1): Equivalence Ratio Current	0 -128	1.999 127.99	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
01	35	4	O2S2_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
01	36	4	O2S3_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32768 ((C*256)+D)/256 - 128
01	37	4	O2S4_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
01	38	4	O2S5_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
01	39	4	O2S6_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
01	3A	4	O2S7_WR_lambda(1): Equivalence Ratio	0 -128	2 128	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128

			Current				
01	3B	4	O2S8_WR_lambda(1): Equivalence Ratio Current	0 -128	2 128	N/A mA	((A*256)+B)/32,768 ((C*256)+D)/256 - 128
01	3C	2	Catalyst Temperature Bank 1, Sensor 1	-40	6,513.5	°C	((A*256)+B)/10 - 40
01	3D	2	Catalyst Temperature Bank 2, Sensor 1	-40	6,513.5	°C	((A*256)+B)/10 - 40
01	3E	2	Catalyst Temperature Bank 1, Sensor 2	-40	6,513.5	°C	((A*256)+B)/10 - 40
01	3F	2	Catalyst Temperature Bank 2, Sensor 2	-40	6,513.5	°C	((A*256)+B)/10 - 40
01	40	4	PIDs supported [41 - 60]				Bit encoded [A7..D0] == [PID $41..PID $60] See below.
01	41	4	Monitor status this drive cycle				Bit encoded.
01	42	2	Control module voltage	0	65.535	V	((A*256)+B)/1000
01	43	2	Absolute load value	0	25,700	%	((A*256)+B)*100/255
01	44	2	Command equivalence ratio	0	2	N/A	((A*256)+B)/32768
01	45	1	Relative throttle position	0	100	%	A*100/255
01	46	1	Ambient air temperature	-40	215	°C	A-40
01	47	1	Absolute throttle position B	0	100	%	A*100/255
01	48	1	Absolute throttle position C	0	100	%	A*100/255
01	49	1	Accelerator pedal position D	0	100	%	A*100/255
01	4A	1	Accelerator pedal position E	0	100	%	A*100/255
01	4B	1	Accelerator pedal position F	0	100	%	A*100/255
01	4C	1	Commanded throttle actuator	0	100	%	A*100/255
01	4D	2	Time run with MIL on	0	65,535	minutes	(A*256)+B
01	4E	2	Time since trouble codes cleared	0	65,535	minutes	(A*256)+B
01	4F	4	Maximum value for equivalence ratio, oxygen sensor voltage, oxygen sensor current, and intake manifold absolute pressure	0, 0, 0, 0	255, 255, 255, 2550	, V, mA, kPa	A, B, C, D*10
01	50	4	Maximum value for air flow rate from mass air flow sensor	0	2550	g/s	A*10, B, C, and D are reserved for future use
01	51	1	Fuel Type				From fuel type table
01	52	1	Ethanol fuel %	0	100	%	A*100/255
01	53	2	Absolute Evap system	0	327.675	kPa	((A*256)+B)/200

			Vapor Pressure				
01	54	2	Evap system vapor pressure	-32,767	32,768	Pa	((A*256)+B)-32767
01	55	2	Short term secondary oxygen sensor trim bank 1 and bank 3	-100	99.22	%	(A-128)*100/128 (B-128)*100/128
01	56	2	Long term secondary oxygen sensor trim bank 1 and bank 3	-100	99.22	%	(A-128)*100/128 (B-128)*100/128
01	57	2	Short term secondary oxygen sensor trim bank 2 and bank 4	-100	99.22	%	(A-128)*100/128 (B-128)*100/128
01	58	2	Long term secondary oxygen sensor trim bank 2 and bank 4	-100	99.22	%	(A-128)*100/128 (B-128)*100/128
01	59	2	Fuel rail pressure (absolute)	0	655,350	kPa	((A*256)+B) * 10
01	5A	1	Relative accelerator pedal position	0	100	%	A*100/255
01	5B	1	Hybrid battery pack remaining life	0	100	%	A*100/255
01	5C	1	Engine oil temperature	-40	210	°C	A - 40
01	5D	2	Fuel injection timing	-210.00	301.992	°	(((A*256)+B)-26,880)/128
01	5E	2	Engine fuel rate	0	3212.75	L/h	((A*256)+B)*0.05
01	5F	1	Emission requirements to which vehicle is designed				Bit Encoded
01	60	4	PIDs supported [61 - 80]				Bit encoded [A7..D0] == [PID $61..PID $80]
01	61	1	Driver's demand engine - percent torque	-125	125	%	A-125
01	62	1	Actual engine - percent torque	-125	125	%	A-125
01	63	2	Engine reference torque	0	65,535	Nm	A*256+B
01	64	5	Engine percent torque data	-125	125	%	A-125 Idle B-125 Engine point 1 C-125 Engine point 2 D-125 Engine point 3 E-125 Engine point 4
01	65	2	Auxiliary input / output supported				Bit Encoded
01	66	5	Mass air flow sensor				
01	67	3	Engine coolant temperature				
01	68	7	Intake air temperature sensor				
01	69	7	Commanded EGR and EGR Error				
01	6A	5	Commanded Diesel intake air flow control				

			and relative intake air flow position				
01	6B	5	Exhaust gas recirculation temperature				
01	6C	5	Commanded throttle actuator control and relative throttle position				
01	6D	6	Fuel pressure control system				
01	6E	5	Injection pressure control system				
01	6F	3	Turbocharger compressor inlet pressure				
01	70	9	Boost pressure control				
01	71	5	Variable Geometry turbo (VGT) control				
01	72	5	Wastegate control				
01	73	5	Exhaust pressure				
01	74	5	Turbocharger RPM				
01	75	7	Turbocharger temperature				
01	76	7	Turbocharger temperature				
01	77	5	Charge air cooler temperature (CACT)				
01	78	9	Exhaust Gas temperature (EGT) Bank 1				Special PID.
01	79	9	Exhaust Gas temperature (EGT) Bank 2				Special PID.
01	7A	7	Diesel particulate filter (DPF)				
01	7B	7	Diesel particulate filter (DPF)				
01	7C	9	Diesel Particulate filter (DPF) temperature				
01	7D	1	NOx NTE control area status				
01	7E	1	PM NTE control area status				
01	7F	13	Engine run time				
01	80	4	PIDs supported [81 - A0]				Bit encoded [A7..D0] == [PID $81..PID $A0]
01	81	21	Engine run time for Auxiliary Emissions Control Device(AECD)				
01	82	21	Engine run time for Auxiliary Emissions				

			Control Device(AECD)				
01	83	5	NOx sensor				
01	84		Manifold surface temperature				
01	85		NOx reagent system				
01	86		Particulate matter (PM) sensor				
01	87		Intake manifold absolute pressure				
01	A0	4	PIDs supported [A1 - C0]				Bit encoded [A7..D0] == [PID $A1..PID $C0]
01	C0	4	PIDs supported [C1 - E0]				Bit encoded [A7..D0] == [PID $C1..PID $E0]
01	C3	?	?	?	?	?	Returns numerous data, including Drive Condition ID and Engine Speed*
01	C4	?	?	?	?	?	B5 is Engine Idle Request B6 is Engine Stop Request*
02	02	2	Freeze frame trouble code				BCD encoded
03	N/A	n*6	Request trouble codes				3 codes per message frame, BCD encoded.
04	N/A	0	Clear trouble codes / Malfunction indicator lamp (MIL) / Check engine light				Clears all stored trouble codes and turns the MIL off.
05	0100		OBD Monitor IDs supported ($01 – $20)				
05	0101		O2 Sensor Monitor Bank 1 Sensor 1	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	0102		O2 Sensor Monitor Bank 1 Sensor 2	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	0103		O2 Sensor Monitor Bank 1 Sensor 3	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	0104		O2 Sensor Monitor Bank 1 Sensor 4	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	0105		O2 Sensor Monitor Bank 2 Sensor 1	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	0106		O2 Sensor Monitor Bank 2 Sensor 2	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	0107		O2 Sensor Monitor Bank 2 Sensor 3	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	0108		O2 Sensor Monitor Bank 2 Sensor 4	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	0109		O2 Sensor Monitor Bank 3 Sensor 1	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	010A		O2 Sensor Monitor Bank 3 Sensor 2	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	010B		O2 Sensor Monitor Bank	0.00	1.275	Volts	0.005 Rich to lean sensor

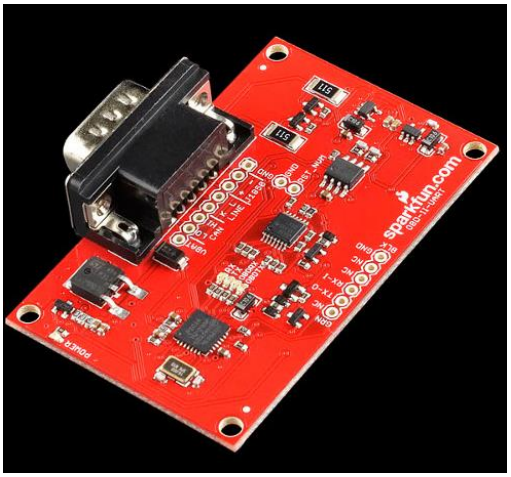
			3 Sensor 3				threshold voltage
05	010C		O2 Sensor Monitor Bank 3 Sensor 4	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	010D		O2 Sensor Monitor Bank 4 Sensor 1	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	010E		O2 Sensor Monitor Bank 4 Sensor 2	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	010F		O2 Sensor Monitor Bank 4 Sensor 3	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	0110		O2 Sensor Monitor Bank 4 Sensor 4	0.00	1.275	Volts	0.005 Rich to lean sensor threshold voltage
05	0201		O2 Sensor Monitor Bank 1 Sensor 1	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	0202		O2 Sensor Monitor Bank 1 Sensor 2	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	0203		O2 Sensor Monitor Bank 1 Sensor 3	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	0204		O2 Sensor Monitor Bank 1 Sensor 4	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	0205		O2 Sensor Monitor Bank 2 Sensor 1	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	0206		O2 Sensor Monitor Bank 2 Sensor 2	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	0207		O2 Sensor Monitor Bank 2 Sensor 3	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	0208		O2 Sensor Monitor Bank 2 Sensor 4	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	0209		O2 Sensor Monitor Bank 3 Sensor 1	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	020A		O2 Sensor Monitor Bank 3 Sensor 2	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	020B		O2 Sensor Monitor Bank 3 Sensor 3	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	020C		O2 Sensor Monitor Bank 3 Sensor 4	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	020D		O2 Sensor Monitor Bank 4 Sensor 1	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	020E		O2 Sensor Monitor Bank 4 Sensor 2	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	020F		O2 Sensor Monitor Bank 4 Sensor 3	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
05	0210		O2 Sensor Monitor Bank 4 Sensor 4	0.00	1.275	Volts	0.005 Lean to Rich sensor threshold voltage
09	00	4	mode 9 supported PIDs 01 to 20				Bit encoded
09	01	1x5	VIN Message Count in command 09 02				Returns 1 line/packet (49 01 05 00 00 00 00), where 05 means 05 packets will be returned in VIN digits.

09	02	5x5	Vehicle identification number (VIN)				Returns the VIN as a multi-frame response using the ISO 15765-2 protocol. This is typically five frames, with the first frame encoding the size and count.
09	03	varies	calibration ID message count from mode $09 pid 04				
09	04	varies	calibration ID				Returns multiple lines, ASCII coded
09	05	varies	calibration verification numbers message count from mode $09 pid 06				Returns multiple lines, ASCII coded
09	06	4	calibration verification numbers				
09	07		in-use performance tracking message count from mode $09 pid 08				
09	08		in-use performance tracking				
09	09		ECU name message count from mode $09 pid 0a				
09	0a		ECU name				
09	0b		in-use performance tracking				

OBD-II UART ADAPTER FOR ARDUINO

Η συσκευή αυτή συνδέεται στον OBD-II diagnostic connector και λειτουργεί ως μια γέφυρα μεταξύ του συστήματος OBD-II που χρησιμοποιεί ο εγκέφαλος του αυτοκινήτου και της σειριακής UART του Arduino μέσω της κατάλληλης βιβλιοθήκης. Επιπλέον, τροφοδοτεί το Arduino με ρεύμα μέσω της υποδοχής του βύσματος.

Αρχικά για τον ίδιο σκοπό χρησιμοποιήθηκε ο OBD-II-UART adapter της εταιρίας Sparkfun. Μετά από ένα μήνα προσπάθειας να επικοινωνήσω με αυτόν τον adapter απευθύνθηκα στην εταιρία η οποία με πληροφόρησε πως το συγκεκριμένο προϊόν αντιμετωπίζει μία σειρά προβλημάτων με αποτέλεσμα να μην μπορεί να χρησιμοποιηθεί τελικά για την υλοποίηση της Διπλωματικής Εργασίας.



OBD-II-UART της Sparkfun

Στη συνέχεια ανακάλυψα τον adapter που τελικά χρησιμοποίησα και οποίος διατίθεται από το website http://arduinodev.com και απεικονίζεται παρακάτω.

Συμβατότητα

Η συσκευή είναι γενικά συμβατή με τις παρακάτω κατηγορίες αυτοκινήτων χωρίς όμως να σημαίνει πως όσα πληρούν τις προϋποθέσεις είναι απαραιτήτως συμβατά.

- United States (Gas) 1996+
- United States (Diesel) 2004+
- Canada (Gas) 1998+
- Europe + UK (Gas) 2001+
- Europe + UK (Diesel) 2004+
- Australia + NZ (Gas/Diesel) 2006+

Επιπλέον είναι συμβατή 100% με τα περισσότερα μοντέλα Arduino όπως Arduino UNO, Arduino Duemilanove, Arduino Leonardo, Arduino Micro, Arduino Nano, Arduino Mini, Arduino Pro Mini, Arduino MEGA 1280/2560/ADK.

Διαδικασία

Η υποδοχή για τη συσκευή βρίσκεται συνήθως κάτω από το τιμόνι ή κοντά σε αυτό. Ένα καλώδιο εξέρχεται από το βύσμα και καταλήγει σε δύο 2-pin connectors. Στον ένα connector βρίσκεται η τροφοδοσία (VCC) και η γείωση (GND) ενώ στον άλλο connector βρίσκονται τα καλώδια που μεταφέρουν τα δεδομένα από και προς τον εγκέφαλο του αυτοκινήτου (Rx/Tx).

Power Connector:

- Red: VCC (connecting to Arduino's VCC)
- Black: GND (connecting to Arduino's GND)

OBD-II Data Connector:

- Yellow: Tx (connecting to Arduino's serial Rx / D0)
- Blue: Rx (connecting to Arduino's serial Tx / D1)

Για τη λειτουργία παρέχονται δύο βιβλιοθήκες ο κώδικας των οποίων παρατίθεται παρακάτω:

OBD.h

```
#define OBD_TIMEOUT_SHORT 2000 /* ms */

#define OBD_TIMEOUT_LONG 7000 /* ms */

#define OBD_TIMEOUT_INIT 3000 /* ms */

#define OBD_SERIAL_BAUDRATE 38400
```

```
#define OBD_RECV_BUF_SIZE 64

#ifndef OBDUART

#if defined(__AVR_ATmega32U4__) || defined(__AVR_ATmega2560__) ||
defined(__AVR_ATmega1280__)

#define OBDUART Serial1

#else

#define OBDUART Serial

#endif

#endif


// mode 0 pids

#define PID_RPM 0x0C

#define PID_SPEED 0x0D

#define PID_THROTTLE 0x11

#define PID_ENGINE_LOAD 0x04

#define PID_COOLANT_TEMP 0x05

#define PID_INTAKE_TEMP 0x0F

#define PID_MAF_FLOW 0x10

#define PID_ABS_ENGINE_LOAD 0x43

#define PID_AMBIENT_TEMP 0x46

#define PID_FUEL_PRESSURE 0x0A

#define PID_INTAKE_MAP 0x0B

#define PID_BAROMETRIC 0x33

#define PID_TIMING_ADVANCE 0x0E

#define PID_FUEL_LEVEL 0x2F

#define PID_RUNTIME 0x1F
```

```
#define PID_DISTANCE 0x31


unsigned int hex2uint16(const char *p);

unsigned char hex2uint8(const char *p);


class COBD

{

public:

COBD():dataMode(1),errors(0) {}

void begin();

bool init(bool passive = false);

bool readSensor(byte pid, int& result, bool passive = false);

bool isValidPID(byte pid);

void sleep(int seconds);

// Query and GetResponse for advanced usage only

void sendQuery(byte pid);

char* getResponse(byte& pid, char* buffer);

bool getResponseParsed(byte& pid, int& result);

byte dataMode;

byte errors;

//char recvBuf[OBD_RECV_BUF_SIZE];

protected:

static int normalizeData(byte pid, char* data);

static int getPercentageValue(char* data)

{
```

```
return (int)hex2uint8(data) * 100 / 255;

}

static int getLargeValue(char* data)

{

return hex2uint16(data);

}

static int getSmallValue(char* data)

{

return hex2uint8(data);

}

static int getTemperatureValue(char* data)

{

return (int)hex2uint8(data) - 40;

}

virtual bool available();

virtual char read();

virtual void write(const char* s);

virtual void write(const char c);

virtual void initIdleLoop() {}

virtual void dataIdleLoop() {}

byte pidmap[4 * 4];

};
```

OBD.cpp

```
#include <Arduino.h>

#include <avr/pgmspace.h>

#include "OBD.h"

#define MAX_CMD_LEN 6


const char PROGMEM s_initcmd[][MAX_CMD_LEN] = {"ATZ\r","ATE0\r","ATL1\r","ATI\r"};

const char PROGMEM s_searching[] = "SEARCHING";

const char PROGMEM s_cmd_fmt[] = "%02X%02X 1\r";

const char PROGMEM s_cmd_sleep[] = "atlp\r";

const char PROGMEM s_cmd_vin[] = "0902\r";

const char PROGMEM s_response_begin[] = "41 ";


unsigned int hex2uint16(const char *p)

{

char c = *p;

unsigned int i = 0;

for (char n = 0; c && n < 4; c = *(++p)) {

if (c >= 'A' && c <= 'F') {

c -= 7;

} else if (c>='a' && c<='f') {

c -= 39;

      } else if (c == ' ') {

         continue;

      } else if (c < '0' || c > '9') {
```

```
break;
      }
i = (i << 4) | (c & 0xF);
n++;
}
return i;
}
unsigned char hex2uint8(const char *p)
{
unsigned char c1 = *p;
unsigned char c2 = *(p + 1);
if (c1 >= 'A' && c1 <= 'F')
c1 -= 7;
else if (c1 >='a' && c1 <= 'f')
c1 -= 39;
else if (c1 < '0' || c1 > '9')
return 0;
if (c2 >= 'A' && c2 <= 'F')
c2 -= 7;
else if (c2 >= 'a' && c2 <= 'f')
c2 -= 39;
else if (c2 < '0' || c2 > '9')
return 0;

return c1 << 4 | (c2 & 0xf);
```

```
}

void COBD::sendQuery(unsigned char pid)

{

char cmd[8];

sprintf_P(cmd, s_cmd_fmt, dataMode, pid);

write(cmd);

}

bool COBD::readSensor(byte pid, int& result, bool passive)

{

// send a query command

sendQuery(pid);

// wait for reponse

bool hasData;

unsigned long tick = millis();

do {

dataIdleLoop();

} while (!(hasData = available()) && millis() - tick < OBD_TIMEOUT_SHORT);

if (!hasData) {

errors++;

return false;

}

// receive and parse the response

return getResponseParsed(pid, result);

}
```

```
bool COBD::available()
{
return OBDUART.available();
}
char COBD::read()
{
return OBDUART.read();
}

void COBD::write(const char* s)
{
OBDUART.write(s);
}

void COBD::write(const char c)
{
OBDUART.write(c);
}

int COBD::normalizeData(byte pid, char* data)
{
int result;
switch (pid) {
case PID_RPM:
result = getLargeValue(data) >> 2;
```

```
break;

case PID_FUEL_PRESSURE:

result = getSmallValue(data) * 3;

break;

case PID_COOLANT_TEMP:

case PID_INTAKE_TEMP:

case PID_AMBIENT_TEMP:

result = getTemperatureValue(data);

break;

case PID_ABS_ENGINE_LOAD:

result = getLargeValue(data) * 100 / 255;

break;

case PID_MAF_FLOW:

result = getLargeValue(data) / 100;

break;

case PID_THROTTLE:

case PID_ENGINE_LOAD:

case PID_FUEL_LEVEL:

result = getPercentageValue(data);

break;

case PID_TIMING_ADVANCE:

result = (getSmallValue(data) - 128) >> 1;

break;

case PID_DISTANCE:

case PID_RUNTIME:
```

```
result = getLargeValue(data);
break;
default:
result = getSmallValue(data);
}
return result;
}

char* COBD::getResponse(byte& pid, char* buffer)
{
unsigned long startTime = millis();
byte i = 0;

for (;;) {
if (available()) {
char c = read();
buffer[i] = c;
if (++i == OBD_RECV_BUF_SIZE - 1) {
// buffer overflow
break;
}
if (c == '>' && i > 6) {
// prompt char reached
break;
}
```

```
} else {
buffer[i] = 0;
unsigned int timeout;
if (dataMode != 1 || strstr_P(buffer, s_searching)) {
timeout = OBD_TIMEOUT_LONG;
} else {
timeout = OBD_TIMEOUT_SHORT;
}
if (millis() - startTime > timeout) {
// timeout
errors++;
break;
}
dataIdleLoop();
}
}
buffer[i] = 0;

char *p = buffer;
while ((p = strstr_P(p, s_response_begin))) {
p += 3;
byte curpid = hex2uint8(p);
if (pid == 0) pid = curpid;
if (curpid == pid) {
errors = 0;
```

```
p += 2;
if (*p == ' ')
return p + 1;
}
}
return 0;
}

bool COBD::getResponseParsed(byte& pid, int& result)
{
char buffer[OBD_RECV_BUF_SIZE];
char* data = getResponse(pid, buffer);
if (!data) {
// try recover next time
write('\r');
return false;
}
result = normalizeData(pid, data);
return true;
}

void COBD::sleep(int seconds)
{
char cmd[MAX_CMD_LEN];
strcpy_P(cmd, s_cmd_sleep);
```

```
write(cmd);
if (seconds) {
delay((unsigned long)seconds << 10);
write('\r');
}
}
bool COBD::isValidPID(byte pid)
{
if (pid >= 0x7f)
return false;
pid--;
byte i = pid >> 3;
byte b = 0x80 >> (pid & 0x7);
return pidmap[i] & b;
}
void COBD::begin()
{
OBDUART.begin(OBD_SERIAL_BAUDRATE);
}
bool COBD::init(bool passive)
{
unsigned long currentMillis;
unsigned char n;
char prompted;
char buffer[OBD_RECV_BUF_SIZE];
```

```
for (unsigned char i = 0; i < sizeof(s_initcmd) / sizeof(s_initcmd[0]); i++) {
if (!passive) {
char cmd[MAX_CMD_LEN];
strcpy_P(cmd, s_initcmd[i]);
write(cmd);
}
n = 0;
prompted = 0;
currentMillis = millis();
for (;;) {
if (available()) {
char c = read();
if (c == '>') {
buffer[n] = 0;
prompted++;
} else if (n < OBD_RECV_BUF_SIZE - 1) {
buffer[n++] = c;
}
} else if (prompted) {
break;
} else {
unsigned long elapsed = millis() - currentMillis;
if (elapsed > OBD_TIMEOUT_INIT) {
// init timeout
//WriteData("\r");
```

```
return false;
}
initIdleLoop();
}
}
}
// load pid map
memset(pidmap, 0, sizeof(pidmap));
for (byte i = 0; i < 4; i++) {
byte pid = i * 0x20;
sendQuery(pid);
char* data = getResponse(pid, buffer);
if (!data) break;
data--;
for (byte n = 0; n < 4; n++) {
if (data[n * 3] != ' ')
break;
pidmap[i * 4 + n] = hex2uint8(data + n * 3 + 1);
}
}
errors = 0;
return true;
}
```

Arduino ADK

Το Arduino ADK είναι μία πλακέτα μικροεπεξεργαστή που βασίζεται στο Arduino ATmega2560. Βασική του διαφορά είναι ότι διαθέτει ενσωματωμένο USB host ώστε να συνδέεται με Android συσκευές. Διαθέτει 54 digital input/output pins, 16 analog inputs, 4 UARTs (hardware serial ports), έναν 16 MHz crystal oscillator, μία υποδοχή USB, ένα βύσμα τροφοδοσίας και ένα κουμπί reset.

Specifications Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Μνήμη

Το Arduino ADK διαθέτει μία 256 KB flash memory για να αποθηκεύει τον κώδικα (από τα οποία τα 8 KB χρησιμοποιούνται για τον bootloader), 8 KB SRAM και 4 KB EEPROM

Προγραμματισμός

Το Arduino ADK μπορεί να προγραμματιστεί με τη χρήση του Arduino software που μπορεί να το κατεβάσει οποιοσδήποτε από το site του Arduino και είναι ανοικτού κώδικα. Η συσκευή έρχεται με έτοιμο bootloader έτσι μπορεί κάποιος να ανεβάσει καινούριο κώδικα χωρίς τη χρήση εξωτερικού hardware programmer.

Τα δεδομένα που λαμβάνει από το OBD-II adapter τα αποστέλει μέσω USB στην Android συσκευή και αποστέλονται μέσω byte-streams. Η κλάση AndroidAccessory παρέχει μεθόδους για ανάγνωση και εγγραφή δεδομένων. Συγκεκριμένα, χρησιμοποιείται ένας πίνακας που αποτελείται από bytes. Το πρώτο byte είναι command type, δηλαδή ορίζει τι είδους δεδομένα αποστέλονται. Τα υπόλοιπα bytes μπορούν να αναφέρονται σε άλλα χαρακτηριστικά των δεδομένων ή να είναι bytes δεδομένων.

COMMAND 0xF	VALUE 0xF	VALUE 0xF	VALUE 0xF

Στον δικό μου κώδικα χρησιμοποιώ έναν πίνακα (sntmsg[]) αποτελούμενο από έξι bytes. Με το πρώτο byte ενημερώνω το Android ότι του αποστέλω δεδομένα από τον OBD-II adapter. Τα υπόλοιπα δύο bytes αποτελούν τις τιμές των χιλιομέτρων ανά ώρα και των στροφών του κινητήρα.

Μπορούν να προστεθούν και άλλα δεδομένα προσθέτοντας τον κατάλληλο κώδικα ζητώντας τα PIDs που επιθυμεί κάποιος και αυξάνοντας το μέγεθος του πίνακα sntmsg[].

Arduino Software

Ο κώδικας που “ανεβαίνει” στο Arduino είναι ο εξής:

```
#include <Max3421e.h>

#include <Usb.h>

#include <Arduino.h>

#include <OBD.h>

#include <AndroidAccessory.h>


#define COMMAND_OBD 0xF        // Ορίζεται το TAG το οποίο όταν θα διαβάσει το Android θα
                                  καταλάβει ότι έχει δεδομένα προς παραλαβή


byte sntmsg[6];          // Ο πίνακας με τον οποίο στέλνονται τα δεδομένα στο Android


 // Ορίζονται τα χαρακτηριστικά της σύνδεσης που πρέπει να είναι τα ίδια με τον κώδικα του
Android

AndroidAccessory acc("Anastasios_Skotidas", //manufacturer

            "Auto_Data", //model

            "Diplwmatiki_Ergasia", //description

            "1.0", //verion

            "http://arduinodev.com",

            "0000000012345678");


COBD obd;

void setup()        // Εδώ γίνεται η αρχικοποίηση κυρίως με την έναρξη των σειριακών θυρών και
                    τον ορισμό του Baudrate στο οποίο θα δουλεύουν

{

 Serial.begin(115200);     // Εκκίνηση σειριακής οθόνης για εμφάνιση δεδομένων στον
                              υπολογιστή

 obd.begin();        // Εκκίνηση της σειριακής θύρας με την οποία συνδέεται το OBD-II adapter με
                        το Arduino στο ορισμένο από τη βιβλιοθήκη Baudrate

 acc.powerOn();     // Έναρξη της επικοινωνίας με το Android
```

```
 while (!obd.init());      // Αρχικοποίηση της OBD-II connection μέχρι αυτή να επιτύχει
}

void loop()               // Εδώ εκτελείται ο κυρίως κώδικας
{
 int kmh;
 if (obd.readSensor(PID_SPEED, kmh)) {        // Η ταχύτητα του οχήματος διαβάζεται και
                                               αποθηκεύεται στη μεταβλητή 'kmh'
  Serial.println("KMH: ");
  Serial.println(kmh);
 }

 int rpm;
 if (obd.readSensor(PID_RPM, rpm)) {          // Οι στροφές του κινητήρα διαβάζονται και
                                               αποθηκεύονται στη μεταβλητή 'rpm'
  Serial.println("RPM: ");
  Serial.println(rpm);
 }

 int throttle;
 if (obd.readSensor(PID_THROTTLE, throttle)) {   // Η κλίση του πεντάλ διαβάζεται και
                                                  αποθηκεύεται στη μεταβλητή 'throttle'
  Serial.println("THROTTLE: ");
  Serial.println(throttle);
 }
```

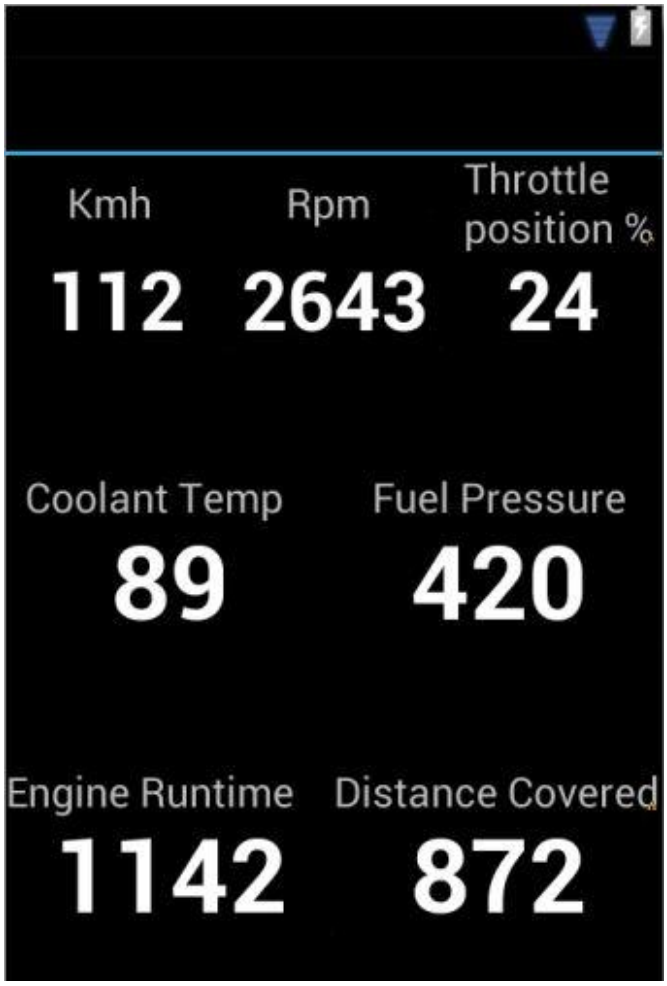
```
 int coolant;
 if (obd.readSensor(PID_COOLANT_TEMP, coolant)) {    // Η θερμοκρασία του κινητήρα
                                                     διαβάζεται και αποθηκεύεται στη
                                                     μεταβλητή 'coolant'
  Serial.println("COOLANT_TEMP: ");
  Serial.println(coolant);
 }

 if (acc.isConnected()) {              // Το Arduino ελέγχει αν το Android phone είναι συνδεδεμένο
                                       και στέλνει τον πίνακα με το TAG και τα δεδομένα

    sntmsg[0] = COMMAND_OBD;
    sntmsg[1] = (byte)kmh;             // Οι μεταβλητές μετατρέπονται σε μορφή byte ώστε να
                                       αποσταλούν
    sntmsg[2] = (byte)(rpm >> 8);
    sntmsg[3] = (byte)rpm;
    sntmsg[4] = (byte)coolant;
    sntmsg[5] = (byte)throttle;

    acc.write(sntmsg, 6);
  }
}
```

Android Operating System

Το Android αποτελεί ένα λειτουργικό σύστημα βασισμένο στο Linux και το οποίο σχεδιάστηκε κυρίως για οθόνες αφής σε συσκευές όπως smartphone και tablet computers. Αρχικά αναπτύχθηκε από την Android Inc., την οποία η Google στήριζε οικονομικά μέχρι το 2005 όταν και εξαγοράσθηκε πλήρως από αυτή.

Το Android παρουσιάστηκε το 2007 με την ίδρυση του Open Handset Alliance: ένα consortium αποτελούμενο από hardware, software, and εταιρίες τηλεπικοινωνιών που είχε ως στόχο να αναπτύξει ανοικτά πρότυπα για κινητές συσκευές. Το Android είναι ανοικτού κώδικα και η Google διακινεί τον κώιδκα κάτω από την Apache License. Αυτός ο ανοικτός κώδικας επιτρέπει στο software να είναι ελεύθερα παραμετροποιήσιμο και να διανέμεται από κατασκευαστές συσκευών, εταιρίες κινητής τηλεφωνίας και developers.

Για τη δημιουργία της εφαρμογής Android που θα εμφανίζει τα δεδομένα του οχήματος χρησιμοποιήθηκε το εργαλείο ανάπτυξης εφαρμογών Eclipse.



Η λήψη των δεδομένων που αποστέλει το Arduino γίνεται μέσω του πίνακα buffer[]. Η εφαρμογή ελέγχει τον πίνακα buffer[] και όταν διαβάσει στην πρώτη θέση του το command byte COMMAND_OBD, που ορίσαμε στο Arduino, μετατρέπει τα δεδομένα σε ακέραιους αριθμούς και τους προβάλει στο Layout.

Κώδικας Android

```
package com.example.try_1;

import java.io.FileDescriptor;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.ParcelFileDescriptor;
import android.util.Log;
import android.view.Menu;
import android.widget.TextView;
import com.android.future.usb.UsbAccessory;
import com.android.future.usb.UsbManager;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.ToggleButton;

public class MainActivity extends Activity {

	private static final String TAG =
MainActivity.class.getSimpleName();

	private PendingIntent mPermissionIntent;
	private static final String ACTION_USB_PERMISSION =
"com.android.example.USB_PERMISSION";
	private boolean mPermissionRequestPending;

	private UsbManager mUsbManager;
	private UsbAccessory mAccessory;
	private ParcelFileDescriptor mFileDescriptor;
	private FileInputStream mInputStream;
	private FileOutputStream mOutputStream;

	private static final byte COMMAND_OBD = 0xF;

	private TextView textView_kmh;
	private TextView textView_rpm;
	private TextView textView_coolant_temp;
	private TextView textView_throttle;

	/** Καλείται όταν το activity δημιουργείται για πρώτη φορά */
	@Override
	public void onCreate(Bundle savedInstanceState) {
```

```
            super.onCreate(savedInstanceState);
            mUsbManager = UsbManager.getInstance(this);
            mPermissionIntent = PendingIntent.getBroadcast(this, 0, new
Intent(
                    ACTION_USB_PERMISSION), 0);
            IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
            filter.addAction(UsbManager.ACTION_USB_ACCESSORY_DETACHED);
            registerReceiver(mUsbReceiver, filter);
            setContentView(R.layout.activity_main);
            textView_kmh = (TextView) findViewById(R.id.textView_kmh);
            textView_rpm = (TextView) findViewById(R.id.textView_rpm);
            textView_coolant_temp = (TextView)
            findViewById(R.id.textView_coolant_temp);
            textView_throttle = (TextView)
            findViewById(R.id.textView_throttle);
        }

        /**
         * Καλείται όταν η activity επανέρχεται από διακοπή και αμέσως μετά
την onCreate() */
        @Override
        public void onResume() {
            super.onResume();
            if (mInputStream != null && mOutputStream != null) {
                return;
            }
            UsbAccessory[] accessories = mUsbManager.getAccessoryList();
            UsbAccessory accessory = (accessories == null ? null :
accessories[0]);
            if (accessory != null) {
                if (mUsbManager.hasPermission(accessory)) {
                    openAccessory(accessory);
                } else {
                    synchronized (mUsbReceiver) {
                        if (!mPermissionRequestPending) {
                            mUsbManager.requestPermission(accessory,
                                    mPermissionIntent);
                            mPermissionRequestPending = true;
                        }
                    }
                }
            } else {
                Log.d(TAG, "mAccessory is null");
            }
        }

        /** Καλείται όταν η activity παύεται από το σύστημα */
        @Override
        public void onPause() {
            super.onPause();
            closeAccessory();
        }

        /**
         * Καλείται όταν η activity δε χρειάζεται πια και διαγράφεται από το
activity stack */
```

```
        @Override
        public void onDestroy() {
            super.onDestroy();
            unregisterReceiver(mUsbReceiver);
        }

        /** Δημιουργείτε η σύνδεση μέσω USB */
        private final BroadcastReceiver mUsbReceiver = new
BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                String action = intent.getAction();
                if (ACTION_USB_PERMISSION.equals(action)) {
                    synchronized (this) {
                        UsbAccessory accessory =
UsbManager.getAccessory(intent);
                        if (intent.getBooleanExtra(
                                UsbManager.EXTRA_PERMISSION_GRANTED, false))
{
                            openAccessory(accessory);
                        } else {
                            Log.d(TAG, "permission denied for accessory "
                                    + accessory);
                        }
                        mPermissionRequestPending = false;
                    }
                } else if
(UsbManager.ACTION_USB_ACCESSORY_DETACHED.equals(action)) {
                    UsbAccessory accessory =
UsbManager.getAccessory(intent);
                    if (accessory != null && accessory.equals(mAccessory)) {
                        closeAccessory();
                    }
                }
            }
        };

        private void openAccessory(UsbAccessory accessory) {
            mFileDescriptor = mUsbManager.openAccessory(accessory);
            if (mFileDescriptor != null) {
                mAccessory = accessory;
                FileDescriptor fd = mFileDescriptor.getFileDescriptor();
                mInputStream = new FileInputStream(fd);
                mOutputStream = new FileOutputStream(fd);
                Thread thread = new Thread(null, commRunnable, TAG);
                thread.start();
                Log.d(TAG, "accessory opened");
            } else {
                Log.d(TAG, "accessory open fail");
            }
        }

        private void closeAccessory() {
            try {
                if (mFileDescriptor != null) {
                    mFileDescriptor.close();
```

```
            }
        } catch (IOException e) {
        } finally {
            mFileDescriptor = null;
            mAccessory = null;
        }
    }



    /*==========================================================*/
    //Εδώ λαμβάνει μέρος η παραλαβή των δεδομένων από το Arduino ADK/
    /*==========================================================*/
    Runnable commRunnable = new Runnable() {
        @Override
        public void run() {
            int ret = 0;
            byte[] buffer = new byte[255];     // Στον πίνακα buffer
αποθηκεύονται τα δεδομένα που αποστέλει το Arduino
            while (ret >= 0) {
                try {
                    ret = mInputStream.read(buffer);
                } catch (IOException e) {
                    break;
                }
                switch (buffer[0]) {
                case COMMAND_OBD:
                 final int KMH = (buffer[1] & 0xFF);
                 final int RPM = ((buffer[2] & 0xFF) << 8)
                                    + (buffer[3] & 0xFF);
                 final int COOLANT_TEMP = (buffer[4] & 0xFF);
                 final int THROTTLE = (buffer[5] & 0xFF);

                 runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            textView_kmh.setText(String.valueOf(KMH));
                            textView_rpm.setText(String.valueOf(RPM));
                             textView_coolant_temp.setText(String.valueOf
                             (COOLANT_TEMP));
                             textView_throttle.setText(String.valueOf(THR
                             OTTLE));
                        }
                    });
                    break;
                default:
                    Log.d(TAG, "unknown msg: " + buffer[0]);
                    break;
                }
            }
        }
    };
}
```

Κώδικας Layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/TextView17"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_weight="1"
            android:paddingLeft="30dp"
            android:text="Kmh"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:textSize="20sp" />

        <TextView
            android:id="@+id/TextView18"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_weight="1"
            android:paddingLeft="30dp"
            android:text="Rpm"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:textSize="20sp" />

        <TextView
            android:id="@+id/TextView16"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_weight="1"
            android:paddingLeft="10dp"
            android:text="Throttle position %"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:textSize="20sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/textView_kmh"
```

```
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="0"
            android:textColor="#ffffffff"
            android:textSize="50sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/textView_rpm"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="0"
            android:textColor="#ffffffff"
            android:textSize="50sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/textView_throttle"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="0"
            android:textColor="#ffffffff"
            android:textSize="50sp"
            android:textStyle="bold" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="60dp" >

        <TextView
            android:id="@+id/TextView11"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Fuel Level"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:textSize="20sp" />

        <TextView
            android:id="@+id/TextView12"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Fuel Pressure"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:textSize="20sp" />
    </LinearLayout>
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:id="@+id/textView_fuel_level"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center"
        android:text="0"
        android:textColor="#ffffffff"
        android:textSize="50sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textView_fuel_pressure"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center"
        android:text="0"
        android:textColor="#ffffffff"
        android:textSize="50sp"
        android:textStyle="bold" />


</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="33dp"
    android:layout_marginTop="50dp" >

    <TextView
        android:id="@+id/TextView03"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:layout_weight="1"
        android:text="Engine Runtime"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/TextView02"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:layout_weight="1"
        android:text="Distance Covered"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:textSize="20sp" />
</LinearLayout>

<LinearLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/textView_runtime"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="0"
            android:textColor="#ffffffff"
            android:textSize="50sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/textView_distance"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="0"
            android:textColor="#ffffffff"
            android:textSize="50sp"
            android:textStyle="bold" />
    </LinearLayout>

</LinearLayout>
```

Κώδικας accessory_filter.xml

Στο αρχείο accessory_filter.xml βρίσκεται ο κώδικας που χρησιμοποιείται ώστε να καταλάβει η εφαρμογή ότι η Android συσκευή έχει συνδεθεί με το Arduino ADK. Τα στοιχεία του πεδίου usb-accessory πρέπει να είναι ίδια με αυτά που βρίσκονται στον κώδικα του Arduino ADK.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <usb-accessory manufacturer="Anastasios_Skotidas" model="Auto_Data"
version="1.0" />
</resources>
```

Μελλοντικές προεκτάσεις

Η παρούσα Διπλωματική αναπτύσει μόνο το κομμάτι που αφορά στην εφαρμογή που προβάλει τα δεδομένα του οχήματος. Μελλοντική εργασία μπορεί να αποτελέσει η ανάπτυξη επι μέρους εφαρμογών που θα σχετίζονται με συγκεκριμένες λειτουργίες όπως: audio player, video player, navigation app και εφαρμογές παροχής ειδήσεων. Οι εφαρμογές αυτές μπορούν υλοποιηθούν από εταιρίες, developers ή να αποτελέσουν θέμα μελλοντικών Διπλωματικών Εργασιών.

Τέλος, πολύ σημαντικό κομμάτι είναι η δημιουργία ενός interface μέσα στο οποίο θα λειτουργούν όλες οι παραπάνω εφαρμογές. Ουσιαστικά θα πρόκειται για μία ειδικά διαμορφωμένη ROM και έναν Launcher που θα βελτιστοποιεί όλες τις λειτουργίες που έχουν αναφερθεί.

Βιβλιογραφία & Πηγές

http://en.wikipedia.org/wiki/Engine_control_unit

http://en.wikipedia.org/wiki/On-board_diagnostics

http://en.wikipedia.org/wiki/OBD-II_PIDs

http://arduinodev.com/

https://www.sparkfun.com/tutorials/294

http://arduino.cc/en/Main/ArduinoBoardADK

http://en.wikipedia.org/wiki/Android_%28operating_system%29

Digital Book: Beginning Android ADK with Arduino, Mario Bohmer (March 2012)