ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Οθόνη πληροφόρησης οδηγού απορριμματοφόρου κατά την αποκομιδή κάδων με μικροελεγκτή

Των φοιτητών:
Γεωργιάδη Κωνσταντίνου
Αρ. Μητρώου: 500030
Δραγούτα Αλκέτα
Αρ. Μητρώου: 503015

Επιβλέπων:
Γιακουμής Άγγελος

Θεσσαλονίκη 17-06-2020

Τίτλος Δ.Ε. Οθόνη πληροφόρησης οδηγού απορριμματοφόρου κατά την αποκομιδή κάδων με μικροελεγκτή

Κωδικός Δ.Ε. 19175

Ονοματεπώνυμο φοιτητών
Γεωργιάδης Κωνσταντίνος
Δραγούτας Αλκέτας
Ονοματεπώνυμο εισηγητή

Ημερομηνία ανάληψης Δ.Ε. 22-05-2019
Ημερομηνία περάτωσης Δ.Ε. 17-06-2020

Βεβαιώνουμε ότι είμαστε οι συγγραφείς αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχουμε καταγράψει τις όποιες πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνουμε ότι αυτή η εργασία προετοιμάστηκε από εμάς, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

ΕΥΧΑΡΙΣΤΙΕΣ

Με την παρούσα πτυχιακή εργασία ένα κομμάτι της ζωής μας φτάνει στο τέλος του. Με την ευκαιρία αυτή θα θέλαμε να ευχαριστήσουμε τους εξής ανθρώπους:

• Τις οικογένειές μας που μας στήριξαν οικονομικά & ψυχολογικά τόσα χρόνια και το πτυχίο μας θα αποτελέσει την ανταμοιβή των κόπων τους.
• Τον επιβλέπων καθηγητή κ. Γιακουμή ‘Αγγελο όχι μόνο για την εμπιστοσύνη & την υπομονή που έδειξε για την εκπόνηση της πτυχιακής εργασίας αλλά και για τον ζήλο & το πάθος που έδειξε και δείχνει στη διδασκαλία των μαθημάτων του

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία εστιάζει στην τεχνολογία των πλέον έξυπνων τρόπων διαχείρισης στόλου οχημάτων (FLEET MANAGEMENT).
Πραγματεύεται την ανάπτυξη περιφερειακού εξοπλισμού για ένα ολοκληρωμένο σύστημα παρακολούθησης στόλου απορριμματοφόρων με δυνατότητες ζύγισης - ταυτοποίησης κάδων απορριμμάτων. Ο εξοπλισμός θα παρέχει άμεση ενημέρωση του οδηγού του απορριμματοφόρου μέσω της οθόνης LCD για το βάρος των κάδων κατά την αποκομιδή και το συνολικό βάρος απορριμμάτων του οχήματος.
Παρακάτω θα αναλυθούν τα επιμέρους υποσυστήματα , ο απαραίτητος εξοπλισμός και οι προδιαγραφές του προαναφερθέντος συστήματος.

ABSTRACT

This dissertation focuses on the technology of the most intelligent vehicle fleet management methods (FLEET MANAGEMENT).
It deals with the development of peripheral equipment for a complete system for monitoring the fleet of garbage trucks with the possibilities of weighing - identification of waste bins. The equipment will provide immediate information to the garbage truck driver via the LCD display about the weight of the bins during collection and the total weight of the vehicle's waste.
Below we will analyze the individual subsystems, the necessary equipment and the specifications of the aforementioned system.

Περιεχόμενα

Ευρετήριο σχημάτων και εικόνων

Α’ ΜΕΡΟΣ- ΘΕΩΡΙΑ

1.ΤΗΛΕΜΑΤΙΚΗ

Η τηλεματική αποτελεί αναπόσπαστο κομμάτι της επιστήμης των τηλεπικοινωνιών και μπορεί να βελτιώσει αισθητά τον κλάδο των χερσαίων μεταφορών είτε πρόκειται για μετακινήσεις επιβατών είτε για τη μεταφορά φορτίων στα αστικά και υπεραστικά κέντρα μιας χώρας.

Τηλεματική είναι οποιαδήποτε ολοκληρωμένη χρήση των τηλεπικοινωνιών και της πληροφορικής και συνεπώς η εφαρμογή της σχετίζεται με οποιοδήποτε από τα ακόλουθα :

- Με την τεχνολογία για την αποστολή, τη λήψη και την αποθήκευση πληροφοριών μέσω τηλεπικοινωνιακών συσκευών, σε συνδυασμό με τον έλεγχο απομακρυσμένων αντικειμένων.
- Με την ενσωμάτωση των τηλεπικοινωνιών και της πληροφορικής σε οχήματα και ανάπτυξη συστημάτων κυρίως για τον έλεγχο των εν κίνηση οχημάτων.
- Με την ενσωμάτωση του παγκόσμιου συστήματος εντοπισμού θέσης (Global positioning Systems-GPS) , το οποίο περιλαμβάνεται αλλά δεν περιορίζεται η τηλεματική, με τους υπολογιστές και της τεχνολογίας των κινητών επικοινωνιών.
- Πιο λεπτομερώς , ο όρος έχει εξελιχθεί για να αναφέρεται στη χρήση τέτοιων συστημάτων στα οχήματα, όπου σε αυτή την περίπτωση ο όρος τηλεματική οχημάτων μπορεί να χρησιμοποιηθεί.

Αντιθέτως τηλεμετρία είναι η τεχνολογία η οποία επιτρέπει την μετάδοση μετρήσεων από απόσταση από την τοποθεσία προέλευσης στην τοποθεσία όπου γίνεται η επεξεργασία της πληροφορίας χωρίς να πραγματοποιείται έλεγχος στα απομακρυσμένα αντικείμενα. Η τηλεμετρία εφαρμόζεται συνήθως στις δοκιμές ιπτάμενων αντικειμένων αλλά έχει και πολλαπλές άλλες χρήσεις.
Παρά το γεγονός ότι η πλειοψηφία των συσκευών που συνδυάζουν τις τηλεπικοινωνίες και την πληροφορική δεν είναι οχήματα, αλλά συσκευές όπως κινητά τηλέφωνα και άλλες παρόμοιες συσκευές, η χρήση τους δεν περιλαμβάνετε στην τηλεματική.

Η τηλεματική σε οχήματα μπορεί να συμβάλει στη βελτίωση της απόδοσης ενός οργανισμού. Μερικές πρακτικές εφαρμογές της τηλεματικής σε οχήματα περιλαμβάνει:

- Παρακολούθηση οχήματος.
  Ο εντοπισμός του οχήματος είναι ένας τρόπος για την παρακολούθηση της θέσης, της κίνησης, την κατάσταση, και την συμπεριφορά ενός οχήματος ή του στόλου των οχημάτων. Αυτό επιτυγχάνεται μέσω ενός συνδυασμού του δέκτη Παγκόσμιου Συστήματος Εντοπισμού Θέσης και μια ηλεκτρονική συσκευή (συνήθως περιλαμβάνει ένα μόντεμ GSM -GPRS) που έχουν εγκατασταθεί σε κάθε όχημα, επικοινωνεί με τον χρήστη, με υπολογιστή, με φορητή συσκευή ή με κάποια διαδικτυακή εφαρμογή. Τα δεδομένα μετατρέπονται σε πληροφορίες από τα εργαλεία διαχείρισης αναφορών σε συνδυασμό με μια οπτική απεικόνιση σε ηλεκτρονικό λογισμικό χαρτογράφησης.
- Παρακολούθηση ρυμουλκούμενου οχήματος και container
  Έχουμε τον εντοπισμό της ρυμουλκούμενης μονάδας ενός αρθρωτού οχήματος ή ενός container μέσω μιας συσκευής εντοπισμού η οποία έχει τοποθετηθεί πάνω στην προς εντοπισμό μονάδα. Στη συνέχεια μέσω ενός δικτύου επικοινωνίας ή δορυφορικού δικτύου μεταδίδονται τα δεδομένα της θέσης σε ένα κεντρικό σημείο συλλογής.
  Η παρακολούθηση ρυμουλκούμενου μονάδων και container χρησιμοποιείται για την αύξηση της παραγωγικότητας με τη βελτιστοποίηση της χρήσης των στόλων, για την αύξηση της ασφάλειας και την δυνατότητα επαναπρογραμματισμού των κινήσεων μεταφοράς με βάση των πληροφοριών της θέσης.

Επίσης είναι σημαντικό να αναφερθεί πως η ενσωμάτωση της τηλεματικής είναι πολύ σημαντική σε container που μεταφέρουν φρέσκα ή κατεψυγμένα τρόφιμα και γενικότερα εμπορεύματα τα οποία απαιτούν συγκεκριμένες περιβαλλοντολογικές συνθήκες οι οποίες επιτυγχάνονται με τεχνητά μέσα στο εσωτερικό, για τη συλλογή δεδομένων όπως θερμοκρασία, υγρασία ή οποιαδήποτε άλλη περιβαλλοντική παράμετρο σε συνάρτηση με τον χρόνο τόσο για την ενεργοποίηση συναγερμών όσο και για την καταγραφή δεδομένων για επαγγελματικούς σκοπούς.

- •Διαχείριση στόλου οχημάτων (fleet management).
  Η διαχείριση στόλου οχημάτων είναι η διαχείριση των οχημάτων μιας εταιρίας είτε αυτά είναι αυτοκίνητα είτε φορτηγά είτε πλοία και μπορούν να περιλαμβάνουν μια σειρά από λειτουργίες διαχείρισης του στόλου.
- •Δορυφορική πλοήγηση.
  Είναι η τεχνολογία που στο πλαίσιο της τηλεματικής χρησιμοποιεί μια συσκευή GPS και ένα εργαλείο ηλεκτρονικής χαρτογράφησης ώστε ένας οδηγός οχήματος να εντοπίσει τη θέση του, να σχεδιάσει μια διαδρομή και να καθοδηγηθεί σε ένα ταξίδι.
- •Ασύρματη επικοινωνία και προειδοποίηση οχημάτων για την οδική ασφάλεια.
  Πρόκειται για ένα ηλεκτρονικό υποσύστημα σε ένα αυτοκίνητο ή οποιαδήποτε άλλο όχημα με σκοπό την ανταλλαγή πληροφοριών όπως για κινδύνους στο οδικό δίκτυο, τις θέσεις και τις ταχύτητες άλλων οχημάτων μέσω μικρής εμβέλειας ραδιοζεύξεων. Ασύρματες μονάδες εγκαθίστανται σε οχήματα και σε σταθερές θέσεις όπως κοντά σε φωτεινούς σηματοδότες και τα κυτία κλήσης έκτακτης ανάγκης κατά μήκους του δρόμου. Αισθητήρες στα αυτοκίνητα και στις σταθερές θέσεις θα παρέχουν πληροφορίες που με κάποιο τρόπο θα εμφανίζονται στους οδηγούς. Επίσης με την εφαρμογή της τηλεματικής ένα όχημα μπορεί να στέλνει ή να λαμβάνει σήματα προειδοποίησης κινδύνου αποφεύγοντας με αυτό τον τρόπο ατυχήματα συμβάλλοντας στην ενεργητική ασφάλεια των οχημάτων.

2. GLOBAL POSITIONING SYSTEM (GPS)

Το Παγκόσμιο Σύστημα Εντοπισμού (GPS) αποτελεί το σύνολο των δορυφόρων που περιστρέφονται γύρω από την γη και παρέχουν δεδομένα. Συγκεκριμένα, αποστέλλει τις λεπτομέρειες της θέσης τους από το διάστημα πίσω στη γη. Το GPS έχει πολλές εφαρμογές σε διάφορους τομείς. Διατίθεται σε κάθε χρήστη με δέκτη GPS.

Εικόνα 1Δορυφόρος GPS

Σύμφωνα με την Tom Tom (2015), το Παγκόσμιο Σύστημα Εντοπισμού (GPS) είναι ένα δορυφορικό σύστημα ραδιοπλοήγησης που αναπτύχθηκε και λειτουργεί από το Υπουργείο Άμυνας των ΗΠΑ.
Οι δορυφόροι GPS μπορούν να χρησιμοποιηθούν δωρεάν από οποιονδήποτε. Το GPS παρέχει ακριβείς πληροφορίες σχετικά με τη θέση, την ταχύτητα και τον χρόνο (PVT) σε έναν απεριόριστο αριθμό κατάλληλα εξοπλισμένων χρηστών εδάφους, θάλασσας, αέρα και χώρου.
Ο δορυφόρος GPS λειτουργεί με τη μετάδοση σημάτων στους αντίστοιχους δέκτες στο έδαφος. Οι δέκτες GPS λαμβάνουν παθητικά δορυφορικά σήματα. δεν μεταδίδουν.

Το Παγκόσμιο Σύστημα Εντοπισμού Θέσης (GPS) χρησιμοποιεί ένα δίκτυο δορυφόρων που επιτρέπει στα άτομα με δέκτες GPS να εντοπίζουν τη θέση τους οπουδήποτε στον κόσμο. Ο Navstar (1996) σημείωσε ότι μια τυπική ακολουθία παρακολούθησης μέσω δορυφόρου ξεκινάει με τον δέκτη να προσδιορίζει ποιοι δορυφόροι διαθέτουν πλήρη ορατότητα για την επιθυμητή παρακολούθηση. Εάν ο δέκτης μπορεί να προσδιορίσει αμέσως την δορυφορική ορατότητα, ο δέκτης θα στοχεύσει σε δορυφόρο για να παρακολουθήσει και να ξεκινήσει τη διαδικασία απόκτησης δεδομένων. Η ορατότητα μέσω δορυφόρου καθορίζεται με βάση το δορυφορικό ημερολόγιο GPS και την αρχική εκτίμηση του δέκτη (ή την είσοδο χρήστη) του χρόνου και της θέσης. Εάν ο δέκτης δεν έχει αποθηκεύσει τις πληροφορίες θέσης, ο δέκτης εισέρχεται σε λειτουργία "αναζήτησης", η οποία αναζητά συστηματικά τους κωδικούς PRN έως ότου επιτευχθεί κλείδωμα σε έναν από τους εξεταζόμενους δορυφόρους. Αφού παρακολουθείται επιτυχώς ένας δορυφόρος, ο δέκτης μπορεί να αποδιαμορφώσει τη ροή δεδομένων μηνυμάτων πλοήγησης και να αποκτήσει τις απαραίτητες πληροφορίες. Ανάλογα με την αρχιτεκτονική του, ένας δέκτης επιλέγει είτε το "καλύτερο" υποσύνολο των ορατών δορυφόρων για τον εντοπισμό ή τη χρήση όλων των ενεργών δορυφόρων, προκειμένου να καθοριστεί μια λύση PVT "all-in-view". Η λύση all in-view είναι συνήθως ακριβέστερη διότι χρησιμοποιεί μια δέσμη τεσσάρων δορυφορικών λύσεων, παρόλο που απαιτεί μια πιο σύνθετη επεξεργασία του δέκτη. Η λύση all-in-view είναι επίσης πιο ασφαλής, καθώς η προσωρινή απώλεια δορυφορικού σήματος (για παράδειγμα λόγω φυσικής απόφραξης κοντά στον δέκτη) δεν διακόπτει τη ροή δεδομένων PVT ενώ ο δέκτης προσπαθεί να επανασυνδέσει το χαμένο σήμα. Πολλοί δέκτες θα παρακολουθήσουν περισσότερους από τέσσερις δορυφόρους (Navstar, 1996).

Το GPS έχει αποδειχθεί ένα ευρέως αναπτυγμένο και χρήσιμο εργαλείο για το εμπόριο, τις επιστημονικές χρήσεις, την παρακολούθηση και την επιτήρηση. Διευκολύνει καθημερινές δραστηριότητες όπως τραπεζικές εργασίες, λειτουργίες κινητών τηλεφώνων και ακόμη και τον έλεγχο των ηλεκτρικών δικτύων, επιτρέποντας την καλή συγχρονισμένη μεταγωγή (Kaplan, 2011).

Το σύστημα GPS χρησιμεύει ακόμα και στους αυτοκινητόδρομους παρέχοντας:

- Υψηλότερα επίπεδα ασφάλειας και κινητικότητας για όλους τους χρήστες συστημάτων επιφανειακών μεταφορών.
- Ακριβέστερος προσδιορισμός θέσης για μεγαλύτερη ενημέρωση των επιβατών
- Αποτελεσματικότερη παρακολούθηση για την εξασφάλιση της τήρησης του χρονοδιαγράμματος, δημιουργώντας ένα σύστημα διαμετακόμισης που ανταποκρίνεται περισσότερο στις ανάγκες των χρηστών των μεταφορών.
- Καλύτερες πληροφορίες τοποθεσίας με ηλεκτρονικούς χάρτες για την παροχή συστημάτων πλοήγησης στο όχημα τόσο για εμπορικούς όσο και για ιδιωτικούς χρήστες.
- Αυξημένη αποτελεσματικότητα και μειωμένο κόστος στον έλεγχο των δρόμων.

Το σύστημα GPS παρέχει χαρτογράφηση της τοποθεσίας:

- Σημαντική αύξηση της παραγωγικότητας από πλευράς χρόνου, εξοπλισμού και εργασίας
- Λιγότεροι λειτουργικοί περιορισμοί σε σύγκριση με τις συμβατικές τεχνικές.
- Ακριβής τοποθέτηση των φυσικών χαρακτηριστικών που μπορούν να χρησιμοποιηθούν σε χάρτες και μοντέλα.
- Ταχύτερη παράδοση γεωγραφικών πληροφοριών που χρειάζονται οι υπεύθυνοι λήψης αποφάσεων.
- Η μέτρηση σε επίπεδο εκατοστόμετρων σε πραγματικό χρόνο.

Το GPS σχεδιάστηκε από το στρατό των Ηνωμένων Πολιτειών. Η ανάπτυξη ξεκίνησε τη δεκαετία του 1960 και ο πρώτος δορυφόρος κυκλοφόρησε το Φεβρουάριο του 1978. Ο πρώτος φορητός δέκτης GPS εισήχθη το 1989 από την Magellan Corp. Το 1996, ο πρόεδρος Ρόναλντ Ρέιγκαν επέτρεψε την ελεύθερη χρήση του GPS από πολιτικούς χρήστες (America.gov, 2006). Από την ανάπτυξή της, οι Η.Π.Α. έχουν υλοποιήσει αρκετές βελτιώσεις στην υπηρεσία GPS, συμπεριλαμβανομένων νέων σημάτων για αστική χρήση και αυξημένης ακρίβειας και ακεραιότητας για όλους τους χρήστες, διατηρώντας ταυτόχρονα τη συμβατότητα με τον υπάρχοντα εξοπλισμό GPS. Ο εκσυγχρονισμός του δορυφορικού συστήματος αποτελεί μια συνεχή πρωτοβουλία του Υπουργείου Άμυνας των ΗΠΑ μέσω μιας σειράς δορυφορικών εξαγορών για την κάλυψη των αυξανόμενων αναγκών των στρατιωτικών, των πολιτών και της εμπορικής αγοράς.

Από τις αρχές του 2015, οι δέκτες GPS υψηλής ποιότητας, ποιότητας FAA και Standard Service (SPS) παρέχουν οριζόντια ακρίβεια καλύτερη από 3,5 μέτρα (GPS Accuracy, 2015), παρόλο που πολλοί παράγοντες όπως η ποιότητα του δέκτη και τα ατμοσφαιρικά ζητήματα μπορούν να επηρεάσουν αυτήν την ακρίβεια.

Σύμφωνα με τους Hoffman-Wellenhof, Lichtenegger και Collins (2001). Το GPS έχει τρία τμήματα:

- Το διαστημικό τμήμα αποτελείται πλέον από 28 δορυφόρους, ο καθένας στην τροχιά του, περίπου 11.000 ναυτικά μίλια πάνω από τη Γη.
- Το τμήμα χρήστη αποτελείται από δέκτες
- Το τμήμα ελέγχου αποτελείται από επίγειους σταθμούς (πέντε από τους οποίους βρίσκονται σε όλο τον κόσμο) που βεβαιώνουν ότι οι δορυφόροι λειτουργούν σωστά.

Υπάρχουν πολλοί δέκτες που κατασκευάζονται από διαφορετικές εταιρείες. Εκτελούν διαφορετικές λειτουργίες ανάλογα με το διαθέσιμο λογισμικό. Το κατάλληλο λογισμικό μπορεί να μεταφορτωθεί στο διαδίκτυο ή να αγοραστεί.

Το Παγκόσμιο Σύστημα Εντοπισμού είναι ένας αστερισμός (ή σύνολο) τουλάχιστον 24 δορυφόρων που μεταδίδουν συνεχώς αχνά ραδιοσήματα προς τη γη. Αυτά τα ραδιοσήματα φέρουν πληροφορίες σχετικά με τη θέση του δορυφόρου και ειδικούς κωδικούς που επιτρέπουν σε κάποιον με δέκτη GPS να μετρά την απόσταση από τον δορυφόρο. Συνδυάζοντας τις αποστάσεις και τις δορυφορικές θέσεις, ο δέκτης μπορεί να βρει το γεωγραφικό πλάτος, το γεωγραφικό μήκος και το ύψος του. Τα δορυφορικά σήματα GPS είναι δωρεάν και είναι διαθέσιμα για οποιονδήποτε χρήση.

3. GENERAL PACKET RADIO SERVICES (GPRS)

Οι υπηρεσίες γενικού πακέτου ραδιοσυχνοτήτων (GPRS) είναι μια υπηρεσία ασύρματης επικοινωνίας με βάση πακέτα που υπόσχεται ρυθμούς δεδομένων από 56 έως 114 Kbps και συνεχή σύνδεση στο Internet για χρήστες κινητών τηλεφώνων και υπολογιστών.

Οι υψηλότερες ταχύτητες δεδομένων επιτρέπουν στους χρήστες να συμμετέχουν σε τηλεδιασκέψεις και να αλληλοεπιδρούν με τοποθεσίες Web πολυμέσων και παρόμοιες εφαρμογές χρησιμοποιώντας κινητές φορητές συσκευές καθώς και φορητούς υπολογιστές.

Το GPRS βασίζεται στην επικοινωνία του Global System for Mobile (GSM) και συμπληρώνει τις υπάρχουσες υπηρεσίες, όπως οι κυψελοειδείς τηλεφωνικές συνδέσεις και η υπηρεσία Short Message Service (SMS).

Θεωρητικά, οι υπηρεσίες που βασίζονται σε πακέτα GPRS κοστίζουν τους χρήστες λιγότερο από τις υπηρεσίες μεταγωγής κυκλώματος, δεδομένου ότι τα κανάλια επικοινωνίας χρησιμοποιούνται σε βάση κοινής χρήσης, ως πακέτα και δεν χρησιμοποιούνται αποκλειστικά για έναν μόνο χρήστη. Είναι επίσης πιο εύκολο να σχεδιαστούν εφαρμογές διαθέσιμες στους χρήστες κινητών τηλεφώνων, επειδή ο ταχύτερος ρυθμός δεδομένων σημαίνει ότι δεν χρειάζεται πλέον το μεσαίο λογισμικό που απαιτείται σήμερα για την προσαρμογή των εφαρμογών στην πιο αργή ταχύτητα των ασύρματων συστημάτων.

Καθώς το GPRS έχει γίνει ευρύτερα διαθέσιμο, μαζί με άλλες υπηρεσίες 3G και 4G, οι χρήστες κινητών εικονικών ιδιωτικών δικτύων (VPN) έχουν πρόσβαση στο ιδιωτικό δίκτυο συνεχώς μέσω ασύρματης σύνδεσης.

Το GPRS συμπληρώνει επίσης το Bluetooth, ένα πρότυπο για την αντικατάσταση των ενσύρματων συνδέσεων μεταξύ συσκευών με ασύρματες συνδέσεις. Εκτός από το πρωτόκολλο Internet (IP), το GPRS υποστηρίζει το X.25, ένα πρωτόκολλο που βασίζεται σε πακέτα και χρησιμοποιείται κυρίως στην Ευρώπη.

Το GPRS είναι ένα εξελικτικό βήμα προς το Enhanced Data GSM Environment (EDGE) και την Universal Mobile Telephone Service (UMTS).

Η απόδοση των εφαρμογών του Διαδικτύου σε ένα κυψελοειδές περιβάλλον χαρακτηρίζεται συνήθως από το χαμηλό διαθέσιμο εύρος ζώνης, τους χρόνους ρύθμισης των μακρών συνδέσεων και την ανεπαρκή χρήση της σπάνιας χωρητικότητας και ζεύξης. Η τυποποίηση του GPRS επικεντρώθηκε, συνεπώς, έντονα στην ανάπτυξη μιας υπηρεσίας, η οποία να υπερνικά αυτά τα μειονεκτήματα μιας κινητής πρόσβασης στο Διαδίκτυο. Οι βελτιώσεις αποκτώνται από την παροχή υπηρεσίας δεδομένων προσανατολισμένης στο πακέτο για το GSM, το οποίο:

- επιτρέπει μειωμένους χρόνους ρύθμισης σύνδεσης
- υποστηρίζει υπάρχοντα πακέτα, προσανατολισμένα πρωτόκολλα όπως X.25 και IP
- παρέχει βελτιστοποιημένη χρήση ραδιοφωνικών πόρων.

Το GPRS είναι τυποποιημένο ώστε να υποστηρίζει βέλτιστα ένα ευρύ φάσμα εφαρμογών που κυμαίνονται από πολύ συχνές μεταδόσεις μικρών όγκων δεδομένων έως σπάνιες μεταδόσεις μεσαίων έως μεγάλων όγκων δεδομένων. Δεδομένου ότι το GPRS είναι προσανατολισμένο σε πακέτα, επιτρέπει τη χρέωση βάσει όγκου σε αντίθεση με το GSM.

Δεδομένου ότι το υπάρχον δίκτυο GSM παρέχει μόνο υπηρεσίες μεταγωγής κυκλώματος, έχουν οριστεί δύο νέοι κόμβοι δικτύου για την υποστήριξη της μεταγωγής πακέτων: ο κόμβος υποστήριξης GPRS (SGSN) και ο κόμβος υποστήριξης GPRS Gateway (GGSN). Ο SGSN είναι υπεύθυνος για την επικοινωνία μεταξύ του κινητού σταθμού (MS) και του δικτύου GPRS. Εξυπηρετεί τον κινητό σταθμό και διατηρεί το περιβάλλον κινητικότητας. Το GGSN παρέχει τη διασύνδεση σε εξωτερικά δίκτυα πακέτων δεδομένων όπως το X.25 ή το Internet, αλλά και σε δίκτυα GPRS άλλων φορέων. Μεταφέρει τα εισερχόμενα πακέτα στον κατάλληλο SGSN για έναν συγκεκριμένο κινητό σταθμό.

Για την επικοινωνία μεταξύ των κόμβων GPRS σε ένα δημόσιο Land Mobile Network (PLMN), χρησιμοποιείται το δίκτυο Intra-PLMN Backbone με βάση IP. Το Υποσύστημα Σταθμών Βάσης GSM (BSS) χρησιμοποιείται ως κοινόχρηστος πόρος τόσο για τα στοιχεία δικτύου όσο και για τα δίκτυα μεταγωγής με πακέτα για την εξασφάλιση της συμβατότητας προς τα πίσω και τη διατήρηση των απαιτούμενων επενδύσεων για την εισαγωγή του GPRS σε βιώσιμο επίπεδο.

Εικόνα 2 GPS Module

3.1 ΠΡΩΤΟΚΟΛΛΑ GPRS

Τρία πρωτόκολλα ελέγχουν τη σύνδεση μεταξύ MS και SGSN:

- Πρωτόκολλο Εξαρτημένης Εξέλιξης Υποσελίδων (SNDCP)
- Πρωτόκολλο Λογικού Ελέγχου Σύνδεσης (LLC)
- Έλεγχος Ραδιοζεύξης / RLC / MAC)

Το πρωτόκολλο SNDCP παρέχει λειτουργικότητα σύγκλισης για να αντιστοιχίσει διαφορετικά πρωτόκολλα στον ενιαίο σύνδεσμο που υποστηρίζεται από LLC. Αυτό περιλαμβάνει πολυπλεξία πακέτων από διαφορετικά πρωτόκολλα, συμπίεση κεφαλίδας (π.χ. TCP / IP) και συμπίεση δεδομένων (π.χ. V42.bis), και κατάτμηση πακέτων μεγαλύτερων από το μέγιστο μέγεθος πακέτων δεδομένων LLC.

Το πρωτόκολλο LLC καθιερώνει μια λογική σύνδεση μεταξύ MS και SGSN. Το LLC λειτουργεί είτε με μη αναγνωρισμένο τρόπο, χωρίς να φροντίζει απώλειες πακέτων, είτε με αναγνωρισμένο τρόπο, ο οποίος εφαρμόζει αναμετάδοση και έλεγχο ροής για να διασφαλιστεί η σωστή παράδοση δεδομένων. Τα πακέτα LLC, είναι ταξινομημένα σε μικρότερα μπλοκ RLC. Το μέγεθος αυτών εξαρτάται από το εφαρμοσμένο σχήμα κωδικοποίησης.

Το RLC λειτουργεί πάντοτε με αναγνωρισμένο τρόπο με μηχανισμό ελέγχου ροής συρόμενου παραθύρου και επιλεκτικό τρόπο ARQ που παρέχει αξιόπιστη σύνδεση μεταξύ MS και BSS.
Επιπλέον, εισάγεται ένα νέο σύστημα ελέγχου πρόσβασης μεσαίου μεγέθους, προσαρμοσμένο στις απαιτήσεις της μετάδοσης δεδομένων που προσανατολίζεται στο πακέτο. Το RLC / MAC θα διασφαλίσει την ταυτόχρονη πρόσβαση σε ραδιοφωνικούς πόρους με πιο ευέλικτο τρόπο σε σύγκριση με τη μη τροποποιημένη δομή TDMA. Η ευελιξία επιτυγχάνεται με την εισαγωγή ενός λογικού καναλιού πακέτο μεταφοράς δεδομένων (PDTCH), το οποίο είναι πολυπλεγμένο σε ένα φυσικό δίαυλο δεδομένων, το κανάλι πακεταρισμένων δεδομένων (PDCH), το οποίο αντιστοιχεί σε μία χρονική περίοδο (TS) στο πλαίσιο GSM TDMA. Έως οκτώ από αυτές τις PDTCH μοιράζονται ένα PDCH.

Χρησιμοποιούνται τέσσερα διαφορετικά σχήματα κωδικοποίησης καναλιών που επιτρέπουν την απενεργοποίηση και ενεργοποίηση της δυνατότητας διόρθωσης σφαλμάτων και της απόδοσης. Η παρεμβολή γίνεται σε ένα μπλοκ RLC, το οποίο αποτελείται από 4 ριπές, με αποτέλεσμα τη σημαντική μείωση της καθυστέρησης διεμπλοκής μέχρι περίπου 18 ms. Το GPRS συμμορφώνεται με το τυπικό σχήμα πολλαπλής πρόσβασης διαίρεσης χρόνου (TDMA) του GSM, δηλαδή η δομή του GPRS είναι συμβατή με το πρότυπο GSM. Παρ 'όλα αυτά, υπάρχουν πολλές επεκτάσεις που γίνονται καλύτερα προσαρμοσμένες στις ανάγκες μιας μετάδοσης με πακέτο, π.χ. οι πόροι ανοδικής και κατερχόμενης ζεύξης χρησιμοποιούνται ανεξάρτητα. Ένας τερματικός σταθμός GPRS επιτρέπεται επίσης να λειτουργεί σε λειτουργία πολλαπλών επιπέδων για να βελτιώσει την ευελιξία και να καλύψει ένα ευρύτερο φάσμα απαιτήσεων ποιότητας υπηρεσίας. Μια ειδική δομή πολλαπλών πλαισίων έχει οριστεί για το GPRS, το οποίο χωρίζεται σε μπλοκ που περιλαμβάνουν το ίδιο TS σε τέσσερα διαδοχικά πλαίσια TDMA για τη μετάδοση ενός μοναδικού μπλοκ RLC.
Οι αναθέσεις πόρων σε ένα συγκεκριμένο τερματικό βασίζονται πάντοτε σε αυτά τα τέσσερα διαδοχικά TS. Η κατανομή αυτών των ραδιοσυχνοτήτων σε ένα τερματικό καθορίζεται από το BSS. Αυτό σημαίνει ότι οι πόροι uplink ελέγχονται επίσης από το BSS και το κράτος μέλος ενημερώνεται για τους δεσμευμένους πόρους uplink μέσω των Uplink State Flags (USF) στην κατερχόμενη ζεύξη.

4. ΜΙΚΡΟΕΛΕΓΚΤΕΣ

4.1 ΕΙΣΑΓΩΓΗ ΣΤΟΥΣ ΜΙΚΡΟΕΛΕΓΚΤΕΣ

Μικροελεγκτής είναι ένας τύπος επεξεργαστή. Ο μικροελεγκτής στην ουσία είναι μία παραλλαγή του μικροεπεξεργαστή, με την βασική διαφορά ότι ο μικροελεγκτής μπορεί να λειτουργήσει με ελάχιστα εξωτερικά συστήματα λόγω των πολλών ενσωματωμένων υποσυστημάτων που διαθέτει. Το συναντάμε σε όλα τα ενσωματωμένα συστήματα (embedded systems) ελέγχου χαμηλού και μεσαίου κόστους.

Οι μικροελεγκτές είναι αυτόνομοι και δεν χρειάζονται κανένα άλλο ολοκληρωμένο κύκλωμα για να λειτουργήσουν. Χάρις σε αυτό έχουν χαμηλή κατανάλωση ισχύος λόγω των απλούστερων διασυνδέσεων.

Η αρχιτεκτονική που χρησιμοποιούν είναι Harvard για την επικοινωνία της CPU και της μνήμης. Οι κατηγορίες που χωρίζονται οι μικροελεγκτές είναι δύο με βάση τον τρόπο που χρησιμοποιούν τις εντολές για τον υπολογισμό μιας εργασίας ανεξάρτητου μεγέθους.

Οι κατηγορίες αυτές είναι:
- Η αρχιτεκτονική RISC
- Η αρχιτεκτονική CISC

Ο προγραμματισμός των μικροελεγκτών μπορεί να γίνει είτε από γλώσσες υψηλού επιπέδου, είτε σε γλώσσα κατανοητή από τον μικροελεγκτή(π.χ. Assembly). Η πιο διαδεδομένη γλώσσα προγραμματισμού είναι η C και C++.

4.2 ΔΙΑΦΟΡΕΣ ΑΠΟ ΤΟΝ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΗ

Ο ρυθμός με τον οποίο αναπτύσσονται οι εφαρμογές σήμερα είναι μεγάλη, καθώς η λειτουργικότητα του τελικού συστήματος καθορίζεται από τα εξωτερικά περιφερειακά τα οποία διασυνδέονται με την κεντρική μονάδα (μικροεπεξεργαστή) η οποία δεν είναι εξειδικευμένη.
Σε αντίθεση με του μικροεπεξεργαστές για ενσωματωμένα συστήματα (μικροελεγκτές) οι οποίοι έχουν μικρότερες ή και μηδαμινές δυνατότητες συνεργασίας με εξωτερικά περιφερειακά η ευελιξία είναι περιορισμένη καθώς και η υπολογιστική ισχύ.

Πλεονεκτήματα μικροελεγκτών:
- Αυτονομία, μέσω της ενσωμάτωσης σύνθετων περιφερειακών υποσυστημάτων όπως μνήμες και θύρες επικοινωνίας. Έτσι πολλοί μικροελεγκτές δεν χρειάζονται κανένα άλλο ολοκληρωμένο κύκλωμα για να λειτουργήσουν.
- Η ενσωμάτωση περιφερειακών σημαίνει ευκολότερη υλοποίηση εφαρμογών λόγω των απλούστερων διασυνδέσεων. Επίσης, οδηγεί σε χαμηλότερη κατανάλωση ισχύος, μεγιστοποιώντας τη φορητότητα και ελαχιστοποιεί το κόστος της συσκευής στην οποία ενσωματώνεται ο μικροελεγκτής.
- Χαμηλό κόστος.
- Μεγαλύτερη αξιοπιστία, και πάλι λόγω των λιγότερων διασυνδέσεων.
- Μειωμένες εκπομπές ηλεκτρομαγνητικών παρεμβολών και μειωμένη ευαισθησία σε αντίστοιχες παρεμβολές από άλλες ηλεκτρικές και ηλεκτρονικές συσκευές. Το πλεονέκτημα αυτό προκύπτει από το μικρότερο αριθμό και μήκος εξωτερικών διασυνδέσεων καθώς και τις χαμηλότερες ταχύτητες λειτουργίας.

- Περισσότεροι διαθέσιμοι ακροδέκτες για ψηφιακές εισόδους-εξόδους (για δεδομένο μέγεθος ολοκληρωμένου κυκλώματος), λόγω της μη δέσμευσής τους για τη σύνδεση εξωτερικών περιφερειακών.
- Μικρό μέγεθος συνολικού υπολογιστικού συστήματος.

4.3 ΣΥΝΗΘΗ ΥΠΟΣΥΣΤΗΜΑΤΑ

Το ολοκληρωμένο σύστημα που αποτελεί τον μικροεπεξεργαστή περιέχει μόνο την Λογική και Αριθμητική Μονάδα(ALU) στοιχειώδεις καταχωρητές (registers), προσωρινή μνήμη RAM πολύ υψηλής ταχύτητας (cache memory) και, κάποιες φορές, τον ελεγκτή μνήμης (memory controller). Όμως, για τη λειτουργία ενός πλήρους ενσωματωμένου υπολογιστικού συστήματος, απαιτούνται πολλά εξωτερικά υποσυστήματα και περιφερειακά.

Κάποια από τα υποσυστήματα και τα περιφερειακά:

- Κύκλωμα συνδετικής λογικής (glue logic) για τη σύνδεση των εξωτερικών μνημών και άλλων περιφερειακών παράλληλης σύνδεσης στην αρτηρία δεδομένων (bus) του επεξεργαστή.
- Μνήμη προγράμματος (τύπου ROM, FLASH, EPROM κλπ.) η οποία περιέχει το λογισμικό του συστήματος. Σε κάποια μοντέλα, είναι δυνατό το κλείδωμα αυτής της μνήμης, μετά την εγγραφή της, ώστε να προστατευτεί το περιεχόμενό της από αντιγραφή.
- Μεγάλο μέγεθος μνήμης RAM.
- Μόνιμη μνήμη αποθήκευσης παραμέτρων λειτουργίας (τύπου EEPROM ή NVRAM) η οποία να μπορεί να γράφεται τον πυρήνα του μικροελεγκτή. Αυτή η μνήμη έχει, έναντι της FLASH, το πλεονέκτημα της δυνατότητας διαγραφής και εγγραφής οποιουδήποτε μεμονωμένου byte.
- Κύκλωμα αρχικοποίησης (reset).
- Διαχειριστή αιτήσεων διακοπής (interrupt request controller) από τα περιφερειακά.
- Κύκλωμα επιτήρησης τροφοδοσίας (brown-out detection) το οποία παρακολουθεί την τροφοδοσία και αρχικοποιεί ολόκληρο το σύστημα όταν αυτή πέσει κάτω από τα ανεκτά όρια, προλαμβάνοντας έτσι την αλλοίωση των δεδομένων.
- Κύκλωμα επιτήρησης λειτουργίας (watchdog timer) το οποίο αρχικοποιεί το σύστημα, αν αυτό εμφανίσει σημάδια δυσλειτουργίας λόγω κολλήματος (hang).
- Τοπικό ταλαντωτή για την παροχή παλμών χρονισμού (clock).
- Έναν ή περισσότερους χρονιστές-απαριθμητές υψηλής ταχύτητας (hardware timer-counter) για τη δημιουργία καθυστερήσεων, μέτρηση διάρκειας γεγονότων, απαρίθμηση γεγονότων και άλλων λειτουργιών ακριβούς χρονισμού.
- Ρολόι πραγματικού χρόνου (Real Time Clock, RTC) το οποίο τροφοδοτείται από ανεξάρτητη μπαταρία και γι'αυτό πρέπει να έχει πολύ χαμηλή κατανάλωση ρεύματος.
- Σειρά ανεξάρτητων ψηφιακών εισόδων και εξόδων (Parallel Input-Output, PIO).

4.4 ΔΙΑΔΕΔΟΜΕΝΕΣ ΚΑΤΗΓΟΡΙΕΣ ΜΙΚΡΟΕΛΕΓΚΤΩΝ

Εξαιτίας του ισχυρού ανταγωνισμού αλλά και της τάσης ενσωμάτωσης των μικροελεγκτών σε κάθε ηλεκτρική και ηλεκτρονική συσκευή , η βιομηχανία μικροελεγκτών έχει καταλήξει στην παραγωγή ανταγωνιστικών μοντέλων μαζικής παραγωγής καθώς και μικροελεγκτών για πιο εξειδικευμένες εφαρμογές.
Έχουμε λοιπόν τις εξής κατηγορίες:

1. Μικροελεγκτές (καμμιά φορά 4-bit αλλά συνήθως 8-bit) πολύ χαμηλού κόστους, γενικής χρήσης, με πολύ μικρό αριθμό ακροδεκτών (ακόμη και λιγότερους από 8). Σχεδιάζονται με έμφαση στη χαμηλή κατανάλωση ισχύος και την αυτάρκεια, ώστε να χρειάζονται ελάχιστα ή και καθόλου εξωτερικά εξαρτήματα και να μη μπορεί να αντιγραφεί εύκολα το εσωτερικό

λογισμικό τους. Απουσιάζει η δυνατότητα επέκτασης της μνήμης τους. Μερικά μοντέλα είναι ευρέως γνωστά στους ερασιτέχνες ηλεκτρονικούς.

2. Μικροελεγκτές (συνήθως 8-bit αλλά και 16 ή 32-bit) χαμηλού κόστους, γενικής χρήσης, με μέτριο έως σχετικά μεγάλο αριθμό ακροδεκτών. Διαθέτουν μεγάλο αριθμό κοινών περιφερειακών, όπως θύρες UART, I2C, SPI ή CAN, μετατροπείς αναλογικού σε ψηφιακό και ψηφιακού σε αναλογικό. Στους κατασκευαστές της Άπω Ανατολής (Ιαπωνία, Κορέα), συνηθίζεται η ενσωμάτωση ελεγκτών οθόνης υγρών κρυστάλλων και πληκτρολογίου. Μερικές φορές παρέχουν δυνατότητα εξωτερικής επέκτασης της μνήμης τους.
3. Μικροελεγκτές (κυρίως 32-bit) μέσου κόστους, γενικής χρήσης, με μεγάλο αριθμό ακροδεκτών. Χαρακτηρίζονται από έμφαση στην ταχύτητα εκτέλεσης εντολών, υψηλή αυτάρκεια περιφερειακών και μεγάλες δυνατότητες εσωτερικής ή εξωτερικής μνήμης προγράμματος (FLASH) και RAM. Στο χώρο αυτό έχουν ισχυρή παρουσία οι αρχιτεκτονικές με υψηλή μεταφερσιμότητα λογισμικού (portability) από τον ένα στον άλλο κατασκευαστή.
4. Μικροελεγκτές εξειδικευμένων εφαρμογών, οι οποίοι ενσωματώνουν συνήθως κάποιο εξειδικευμένο πρωτόκολλο επικοινωνίας το οποίο υλοποιείται πάντοτε σε hardware. Τέτοιοι μικροελεγκτές χρησιμοποιούνται σε τηλεπικοινωνιακές συσκευές όπως τα μόντεμ.

Οι περισσότερες πωλήσεις μικροελεγκτών αφορά αυτούς των 8-bit καθώς και η κατηγορία με το χαμηλότερο κόστος και το μικρότερο μέγεθος λογισμικού για το ίδιο αποτέλεσμα και αυτό συμβαίνει γιατί οι σύγχρονες οικογένειες μικροελεγκτών 8-bit έχουν πολύ βελτιωμένες επιδόσεις σε σχέση με το παρελθόν.

4.5 ΠΡΟΣΘΕΤΕΣ ΛΕΙΤΟΥΡΓΙΕΣ

Ανάλογα με την εφαρμογή για την οποία προορίζεται ένας μικροελεγκτής, μπορεί να περιέχει και:

- Μία ή περισσότερες ασύγχρονες σειριακές θύρες επικοινωνίας (Universal Asynchronous Receiver Transmitter, UART).
- Σύγχρονες σειριακές θύρες επικοινωνίας (πχ I2C, SPI, Ethernet).
- Ολόκληρα υποσυστήματα για την άμεση υποστήριξη από υλικολογισμικό (firmware) των πιο σύνθετων πρωτοκόλλων επικοινωνίας όπως CAN, HDLC, ISDN, ADSL.
- Μονάδα άμεσης εκτέλεσης πράξεων κινητής υποδιαστολής (Floating Point Processing Unit, FPU), η οποία είναι πάντοτε πιο γρήγορη από την ALU του επεξεργαστή. Τέτοιες μονάδες χαρακτηρίζουν τους μικροελεγκτές με δυνατότητες ψηφιακής επεξεργασίας σήματος (Digital Signal Processing, DSP). Τα τελευταία χρόνια, με την ευρύτατη διάδοση των φορητών συσκευών ήχου και εικόνας, παρατηρείται μια τάση σύγκλισης των μικροελεγκτών με τους DSP.[4]
- Περισσότερες από μία εισόδους για μετατροπή αναλογικού σήματος σε ψηφιακό (Analog to Digital converter, ADC).
- Μετατροπέα ψηφιακού σε αναλογικό σήμα (Digital to Analog converter, DAC).
- Ελεγκτή οθόνης υγρών κρυστάλλων (Liquid Crystal Display, LCD).
- Υποσύστημα προγραμματισμού πάνω στο κύκλωμα (τύπου ISP, βλ. παραπάνω). Χάρη σε αυτό το κύκλωμα, είναι δυνατός ο επαναπρογραμματισμός (αναβάθμιση λογισμικού) της εφαρμογής, συνδέοντας στη συσκευή μια εξωτερική συσκευή προγραμματισμού (συνήθως σε θύρα UART RS-232) ή ακόμη και από το διαδίκτυο. Αυτή η δυνατότητα απαιτεί την

προϋπαρξη λογισμικού υποδοχής (bootstrap) μέσα στη μνήμη προγράμματος και επομένως δεν μπορεί να γίνει σε τελείως άδεια μνήμη προγράμματος.

- Υποσύστημα προγραμματισμού (τύπου ISP) και διάγνωσης (συνήθως είναι το καθιερωμένο πρότυπο JTAG). Χάρη σε αυτό, είναι δυνατός ο προγραμματισμός της μνήμης προγράμματος χωρίς να προαπαιτείται κάποιο πρόγραμμα υποδοχής. Γι αυτό το λόγο, είναι ιδιαίτερα χρήσιμο στον αρχικό προγραμματισμό, πχ κατά τη συναρμολόγηση, ή σε περίπτωση σφάλματος (bug) στο λογισμικό υποδοχής το οποίο να καθιστά αδύνατη την κανονική αναβάθμιση.

4.6 ΓΕΝΙΚΗ ΔΟΜΗ ΕΝΟΣ ΜΙΚΡΟΕΛΕΓΚΤΗ

Στη δομή ενός μικροελεγκτή διακρίνουμε 2 μέρη:

Τον πυρήνα του μικροελεγκτή (Core)

- Κεντρική μονάδα επεξεργασίας (CPU)
- Μνήμη

Τα περιφερειακά (Peripherals)

- Θύρες εισόδου/εξόδου
- Θύρες σειριακής επικοινωνίας
- Οι μετρητές χρόνου
- Μετατροπέας αναλογικού σήματος σε ψηφιακό
- Θύρα παράλληλης επικοινωνίας
- Η μονάδα διαμόρφωσης πλάτους

Β’ ΜΕΡΟΣ

5. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

5.1 Λειτουργική Αρχιτεκτονική Συστήματος

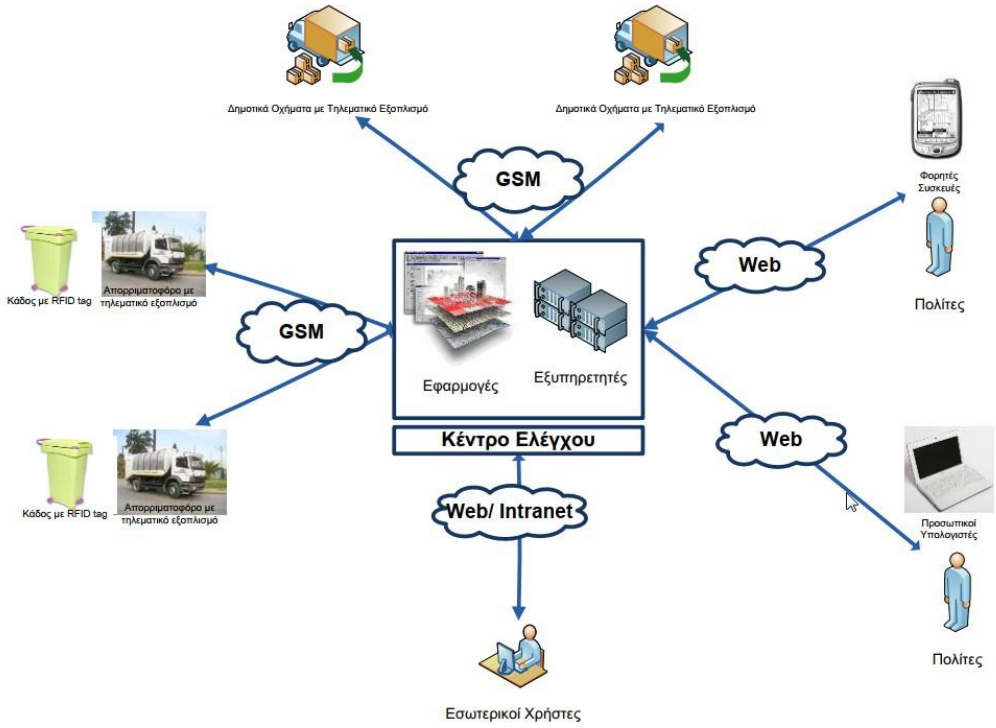
Το Ολοκληρωμένο Σύστημα Διαχείρισης Στόλου περιλαμβάνει τα ακόλουθα:

- ✓ **Την Εφαρμογή Διαχείρισης Στόλου**, που περιλαμβάνει ανταλλαγή δεδομένων με το πεδίο, τις σχετικές ΒΔ, όπως επίσης και την εποπτεία και παρακολούθηση του στόλου
- ✓ **Χαρτογραφικά Δεδομένα** (GIS ΒΔ) Τα χαρτογραφικά δεδομένα δίνουν την δυνατότητα απεικόνισης διανυσματικών με δρόμους και οικοδομικά τετράγωνα.
- ✓ **Τον Εξοπλισμό του Οχήματος** (Τηλεματική Συσκευή και λοιπός εξοπλισμός) που αποτελείται από τηλεματικό εξοπλισμό οχήματος, ο οποίος περιλαμβάνει σύστημα για αυτόματο εντοπισμό της θέσης με GPS, κατάλληλες ψηφιακές εισόδους/εξόδους για διασύνδεση με αισθητήρες και έχει τη δυνατότητα σύνδεσης με σύστημα ταυτοποίησης οδηγού, ζύγισης και ταυτοποίησης κάδων, ενώ η αποστολή των τηλεματικών δεδομένων γίνεται με ασύρματη επικοινωνία με χρήση δικτύου GSM/GPRS. Στο κέντρο ελέγχου θα εγκατασταθεί το λογισμικό διαχείρισης στόλου ***ΕΡΜΗΣ*** με ενσωματωμένο API (Application Programming Interface) για τη διασύνδεση με τρίτες εφαρμογές.
- ✓ ✓**Εξοπλισμός ζύγισης κάδου και ανάγνωσης RFID Tag** Το υποσύστημα ζύγισης και ταυτοποίησης κάδων αποτελεί μία επέκταση του συστήματος και χρησιμοποιεί σύγχρονες τεχνολογίες αφενός μεν για την αναγνώριση της ηλεκτρονικής ταυτότητας του κάθε κάδου, αφετέρου δε για την αυτοματοποιημένη καταγραφή του βάρους κατά την αποκομιδή τους από το απορριμματοφόρο.
- ✓ ✓**Εξοπλισμός λήψης των δεδομένων ζύγισης και προβολή των πληροφοριών μέσω οθόνης LCD στον οδηγό του οχήματος.** Το υποσύστημα της πληροφόρησης του οδηγού με τα δεδομένα των κάδων αποτελεί μία επέκταση του συστήματος. Χρησιμοποιεί τον μικροελεγκτή PIC18F47K40 της Microchip ώστε να προβάλει στον οδηγό του απορριμματοφόρου όλες τις απαραίτητες πληροφορίες ηλεκτρονικής ταυτότητας του κάθε κάδου και του βάρους του κατά την αποκομιδή από το απορριμματοφόρο.
- ✓ ✓**Το Δίκτυο Επικοινωνίας** Όλα τα παραπάνω συνεργάζονται για να παρέχουν ένα ικανό, ολοκληρωμένο σύστημα διαχείρισης στόλου. Τα συστήματα αυτά αλληλοεπιδρούν στα πλαίσια μίας ανοικτής αρχιτεκτονικής, η οποία υποστηρίζει την επεκτασιμότητα της πλατφόρμας.

Τα επιμέρους τμήματα της εφαρμογής μπορεί να είναι τοποθετημένα οπουδήποτε είτε σε εξυπηρετητές σε χώρο του Δήμου είτε αλλού εφόσον ικανοποιούνται οι απαιτήσεις επικοινωνίας μεταξύ των τμημάτων.

Το σύστημα θα παρέχει τις διεπιφάνειες χρήσης του και γενικά τη λειτουργικότητα του στην Ελληνική γλώσσα.

Στα παρακάτω σχήματα δίνεται η υλοποίηση της πλατφόρμας:



Εικόνα 3 Υλοποίηση της πλατφόρμας

Όλα τα τμήματα θα λειτουργούν ανεξάρτητα. Σε περίπτωση σφάλματος επικοινωνίας με το πεδίο η λειτουργία τους δε θα διακόπτεται και θα παρέχεται η λειτουργικότητα τους για τα τελευταία διαθέσιμα δεδομένα παρέχοντας και σχετική ενημέρωση προς τους χρήστες τους, ενώ η επαναφορά τους σε κανονική εκτέλεση θα γίνεται αυτόματα με την αποκατάσταση της επικοινωνίας με το πεδίο. Η λειτουργική αρχιτεκτονική του έργου παρουσιάζεται στο διάγραμμα που ακολουθεί. Το κέντρο ελέγχου θα γίνεται hosted από τον Ανάδοχο σε data center υψηλών προδιαγραφών

5.2 Φυσική Αρχιτεκτονική

Η φυσική αρχιτεκτονική του συστήματος αποτελείται από δύο (2) μέρη, τον εξοπλισμό στο Κέντρο Ελέγχου και τον εξοπλισμό στα οχήματα. Η υλοποίηση αυτών των κομματιών θα γίνει παράλληλα ώστε να επιτευχθεί ο ελάχιστος δυνατός χρόνος υλοποίησης, ενώ η διασύνδεση των 2 κομματιών θα γίνει μέσω του δικτύου τηλεπικοινωνίας που θα στηθεί.

Πιο συγκεκριμένα ο τηλεματικός εξοπλισμός στα οχήματα περιλαμβάνει:

- ➢Εξοπλισμό οχήματος που αποτελείται από δέκτη GPS, ψηφιακές εισόδους/εξόδους, σύνδεση με alarm button για πραγματοποίηση επειγουσών κλήσεων και σύστημα επικοινωνίας με δυνατότητα αποστολής των δεδομένων με χρήση *GPRS/GSM*.

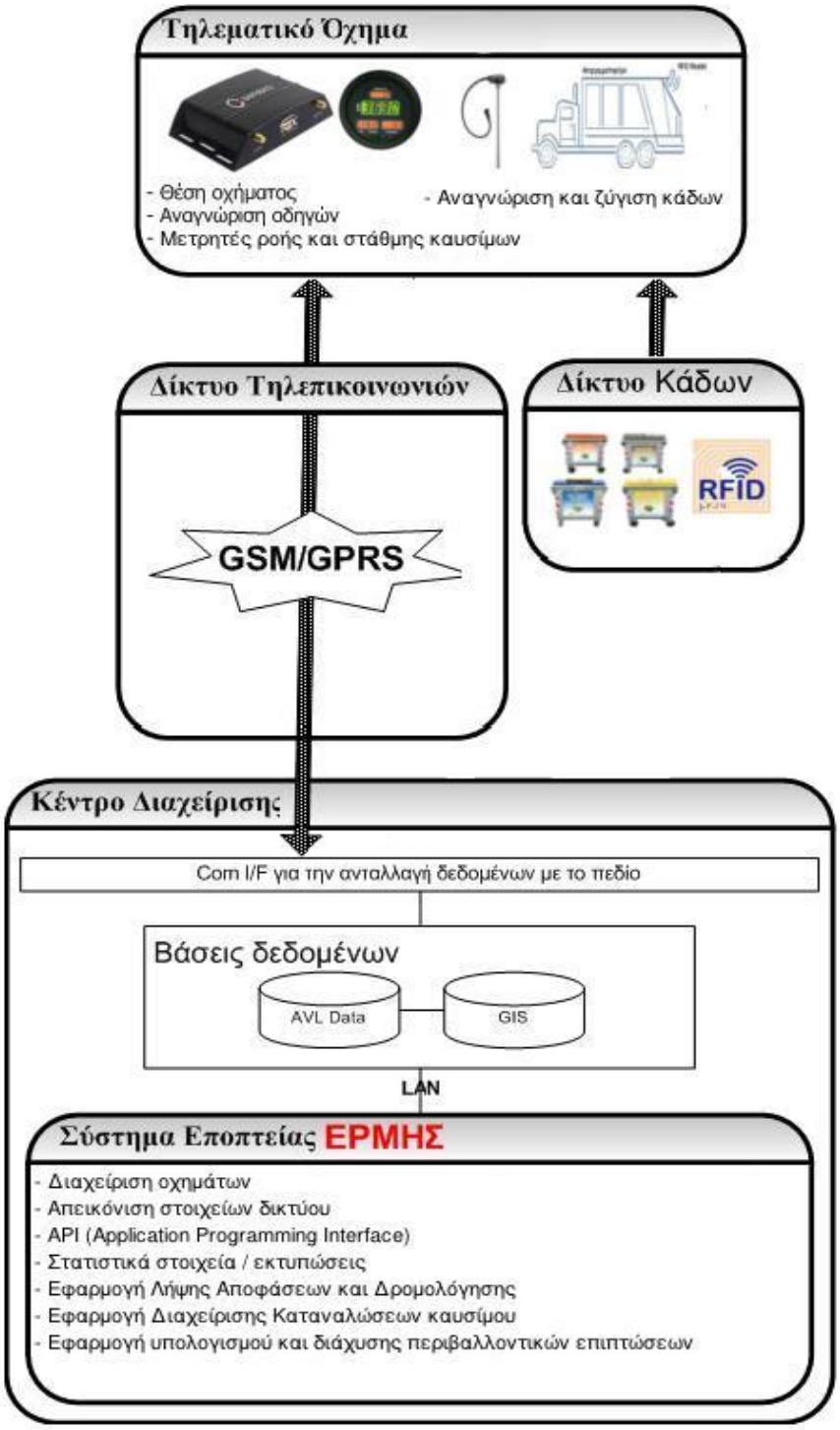
- ➢Ζυγιστικό σύστημα με χρήση RFID Tags

- ➢Σύστημα προβολής των πληροφοριών μέσω οθόνης LCD στον οδηγό του οχήματος.

- ➢Η επικοινωνία ανάμεσα στο όχημα και στο κέντρο ελέγχου θα πραγματοποιείται μέσω δεδομένων. Η επικοινωνία θα πραγματοποιείται μέσω του εξοπλισμού που θα εγκατασταθεί στο όχημα και μέσω δικτύου GSM/GPRS.

- ➢Η εκπομπή των δεδομένων θα πραγματοποιείται με τη μέθοδο των πακέτων δεδομένων. Τα οχήματα θα είναι διασυνδεδεμένα σε ένα GSM/GPRS δίκτυο. Ουσιαστικά κάθε μονάδα θα λαμβάνει μία IP διεύθυνση μέσω της οποίας θα γίνεται η αποστολή και λήψη των δεδομένων.

Στο παρακάτω σχήμα φαίνεται η προτεινόμενη φυσική αρχιτεκτονική:



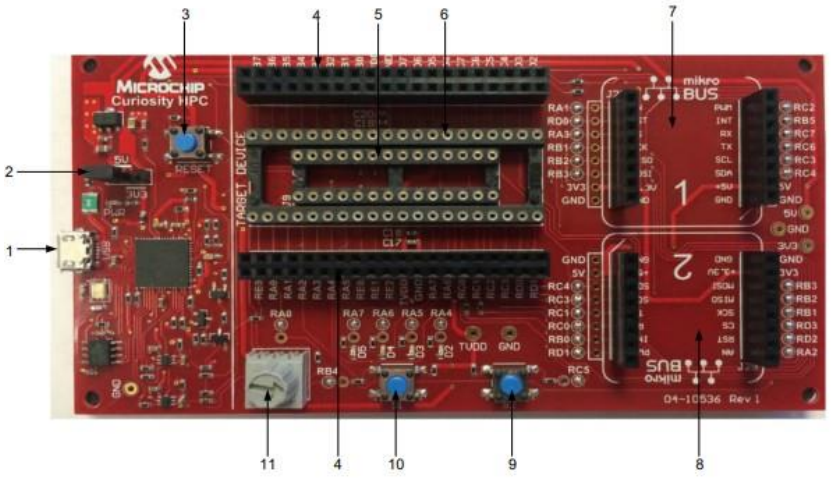
Εικόνα 4 Αρχιτεκτονική Συστήματος

6. ΣΥΣΤΗΜΑ ΥΠΟ ΑΝΑΠΤΥΞΗ

6.1 ΠΕΡΙΓΡΑΦΗ ΥΛΙΚΩΝ ΑΝΑΠΤΥΞΗΣ

6.1.1 Πλακέτα ανάπτυξης Curiosity High Pin Count (HPC)

Η πλακέτα ανάπτυξης Curiosity High Pin Count (HPC) υποστηρίζει τα Microchip 28 και 40-pin 8-bit PIC® MCUs. Περιβάλλεται από δύο μοναδικές πινοσειρές PDIP επέκτασης, οι οποίες επιτρέπουν συνδεσιμότητα σε όλες τις ακίδες των PIC MCUs.
Ο προγραμματισμός / ο εντοπισμός σφαλμάτων πραγματοποιείται μέσω του PICkit ™ On Board (PKOB),εξαλείφοντας την ανάγκη για εξωτερικό εργαλείο προγραμματισμού / εντοπισμού σφαλμάτων.
Η πλακέτα περιλαμβάνει ένα σύνολο τεσσάρων ενδεικτικών LED, ποτενσιόμετρου και μπουτόν διακόπτες. Επιπλέον, το Curiosity HPC Board ενσωματώνει δύο MikroElektronika υποδοχές microBUS ™ οι οποίες μπορούν να φιλοξενήσουν μια ποικιλία από πρόσθετες μονάδες Click ™ Board που μπορεί να χρησιμοποιηθεί για την ανάπτυξη εφαρμογών.



Εικόνα 5 Πλακέτα Ανάπτυξης

Η παραπάνω πλακέτα χρησιμοποιήθηκε κατά την διάρκεια της ανάπτυξης ώστε να είναι πιο εύκολη η διαδικασία του εντοπισμού και επιδιόρθωσης των σφαλμάτων.
Μετά το τέλος της ανάπτυξης τυπώθηκε η τελική πλακέτα.

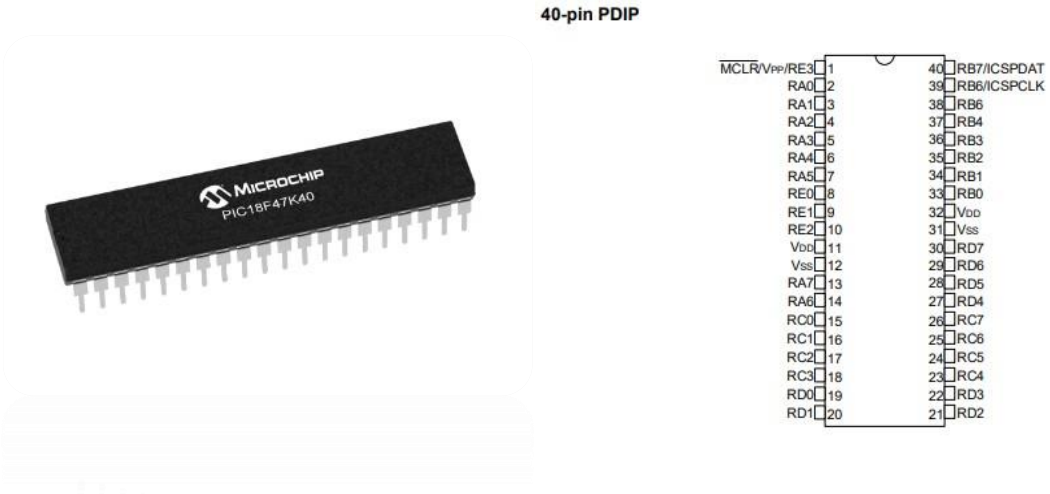
6.1.2 Μικροελεγκτής PIC18F47K40

Αυτοί οι μικροελεγκτές PIC18 (L) F27 / 47K40 διαθέτουν αναλογικό πυρήνα ανεξάρτητα από τα περιφερειακά επικοινωνίας, σε συνδυασμό με τεχνολογία eXtreme Low-Power (XLP) για ένα ευρύ φάσμα εφαρμογών γενικού σκοπού και χαμηλής ισχύος. Αυτές οι συσκευές 28/40/44-pin είναι εξοπλισμένες με 10-bit ADC με Computation (ADCC),

Προσφέρονται επίσης για ένα σύνολο ανεξάρτητων περιφερειακών πυρήνων όπως η Συμπληρωματική Γεννήτρια Κυματομορφής (CWG),
Window Watchdog Timer (WWDT), Cyclic Redundancy Check (CRC) / Memory Scan, Zero-Cross Detect (ZCD) και Peripheral Pin Select (PPS) παρέχοντας αυξημένη ευελιξία σχεδιασμού και χαμηλότερο κόστος.

Βασικά χαρακτηριστικά:
• C Compiler Optimized RISC Architecture
• Operating Speed:
– DC – 64 MHz clock input over the full VDD range
– 62.5 ns minimum instruction cycle
• Programmable 2-Level Interrupt Priority
• 31-Level Deep Hardware Stack
• Three 8-Bit Timers (TMR2/4/6) with Hardware Limit Timer (HLT)
• Four 16-Bit Timers (TMR0/1/3/5)
• Low-Current Power-on Reset (POR)
• Power-up Timer (PWRT)
• Brown-out Reset (BOR)
• Low-Power BOR (LPBOR) Option
• Windowed Watchdog Timer (WWDT):
– Watchdog Reset on too long or too short interval between watchdog clear events
– Variable prescaler selection
– Variable window size selection
– All sources configurable in hardware or software

Μνήμη
• 128K Bytes Program Flash Memory
• 3728 Bytes Data SRAM Memory
• 1024 Bytes Data EEPROM
• Programmable Code Protection
• Direct, Indirect and Relative Addressing modes



Εικόνα 6 Μικροελεγκτής

6.1.3 Θύρες εισόδου – εξόδου RS232

Το πρότυπο επικοινωνίας RS232 έχει καθιερωθεί από τη δεκαετία του '60, αλλά χάρη στην εφαρμογή του σε ένα ευρύ φάσμα συσκευών - συμπεριλαμβανομένων των μητρικών υπολογιστών PC, απέκτησε μεγάλη δημοτικότητα και εξακολουθεί να χρησιμοποιείται. Με την πάροδο του χρόνου, αυτό το πρότυπο πέρασε πολλές αναθεωρήσεις και η πιο πρόσφατη αναθεώρηση είναι TIA-232-F (R2012). Παρόλο που αναπτύχθηκε αρχικά για τη σύνδεση teletypewriters και μόντεμ, σήμερα χρησιμοποιείται για σειριακή επικοινωνία μεταξύ ενός ευρέος φάσματος διαφορετικών συσκευών.

Για την εφαρμογή του προτύπου επικοινωνίας RS232 στα MCUs, είναι απαραίτητο να μετατρέψετε τα επίπεδα τάσης σε επίπεδα αποδεκτά για τα σύγχρονα MCU. Το RS232 2 εκτελεί πλήρη μετατροπή σήματος RS232 σε UART, επιτρέποντας τη λειτουργία 3.3V και 5V. Επιπλέον, διαθέτει προστασία ± 15 kV ESD για τους ακροδέκτες RS-232 I / O. Αυτά τα χαρακτηριστικά το καθιστούν μια τέλεια λύση για μπαταρία, φορητό και φορητό εξοπλισμό, PDA, παλάμες, ψηφιακές φωτογραφικές μηχανές και άλλες συσκευές που εξακολουθούν να υποστηρίζουν το πρότυπο RS232.

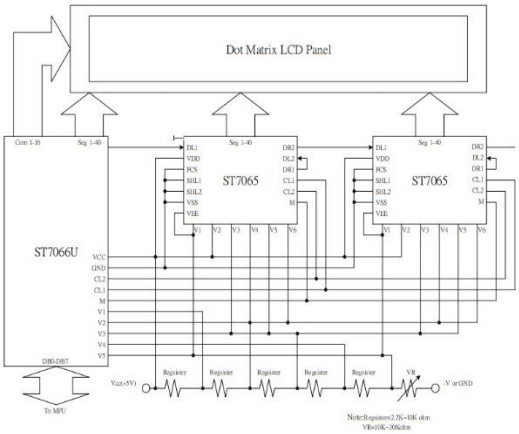
Εικόνα 7 RS232

6.1.4 Οθόνη LCD

Για την εμφάνιση των πληροφοριών ζύγισης των κάδων απορριμμάτων του απορριμματοφόρου στον οδηγό θα χρησιμοποιήσουμε μια οθόνη LCD DISPLAY, 4 γραμμών και 20 χαρακτήρων η κάθε γραμμή.

Χαρακτηριστικά:
Ανάλυση: 4x20 Χαρακτήρες
Χρώμα: Γκρι
Φωτισμός: Λευκό
Θερμοκρασία λειτουργίας: -20 to +70c
Διαστάσεις: 146.0mm x 62.5mm

Εικόνα 8 LCD Display



Εικόνα 9 LCD Display

6.2 ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΊΑΣ ΤΟΥ ΣΥΣΤΉΜΑΤΟΣ ΠΡΟΒΟΛΗΣ ΠΛΗΡΟΦΟΡΙΩΝ ΖΥΓΙΣΗΣ ΚΑΔΩΝ ΜΕΣΩ ΜΙΚΡΟΕΛΕΓΚΤΗ ΤΗΣ MICROCHIP ΣΤΟΝ ΟΔΗΓΌ ΤΟΥ ΟΧΗΜΑΤΟΣ

Το υπό ανάπτυξη σύστημα θα παρουσιάζει όλη την πληροφορία του υποσυστήματος ζύγισης κάδων στον οδηγό, παράλληλα με την αποστολή προς το κέντρο ελέγχου μέσω του τηλεματικού εξοπλισμού. Η πληροφορία που απαιτείται είναι το βάρος των απορριμμάτων του κάθε κάδου κατά την αποκομιδή του και το συνολικό βάρος των απορριμμάτων που υπάρχουν στο όχημα. Το σύστημα σχεδιάστηκε ώστε κάνει έλεγχο στην αρχή της λειτουργείας του για καταγραφή προβλημάτων του υποσυστήματος ζύγισης και του τηλεματικού εξοπλισμού.

Το σύστημα παρεμβάλετε ανάμεσα στο υποσύστημα ζύγισης κάδων και στον τηλεματικό εξοπλισμό, λαμβάνει την πληροφορία ζύγισης, την αποστέλλει στο κέντρο ελέγχου μέσω του τηλεματικού εξοπλισμού και παράλληλα την αποκωδικοποιεί ώστε να την παρουσιάσει στον οδηγό του οχήματος.

Η τελική πληροφορία που εμφανίζεται στον οδηγό από το σύστημα είναι:

- Το βάρος των απορριμμάτων του κάθε κάδου κατά την αποκομιδή του (καθαρό όπου υπάρχει RF-ID).
- Το ID του κάθε κάδου (όπου αυτό υπάρχει).
- Το συνολικό βάρος των απορριμμάτων που υπάρχουν στο όχημα από την τελευταία φορά που έγινε επίσκεψη στον ΧΥΤΑ για την απόρριψη των απορριμμάτων.

Το σύστημα διαθέτει τα παρακάτω χαρακτηριστικά:

- Μεγάλη οθόνη υγρών κρυστάλλων με οπίσθιο φωτισμό ώστε να είναι ευανάγνωστη κάτω από όλες τις συνθήκες χωρίς να δυσχεραίνει το έργο του οδηγού.
- Δύο σειριακές θήρες (RS-232) για σύνδεση με το υποσύστημα ζύγισης κάδων και τον τηλεματικό εξοπλισμό.
- Κύκλωμα τροφοδοσίας για χρήση σε οχήματα έως 35V DC
- Ψηφιακή είσοδο για τον μηδενισμό του συνολικού βάρους απορριμμάτων του οχήματος.
- Δύο LED ένδειξης λειτουργίας και βλάβης.

Για την ανάπτυξη του προγράμματος χρησιμοποιήθηκαν αναπτυξιακές πλακέτες για γρηγορότερη αποσφαλμάτωση. Η τελική κατασκευή έγινε με χρήση διάτρητης πλακέτας.

6.2.1 Ανάπτυξη προγράμματος μικροελεγκτή

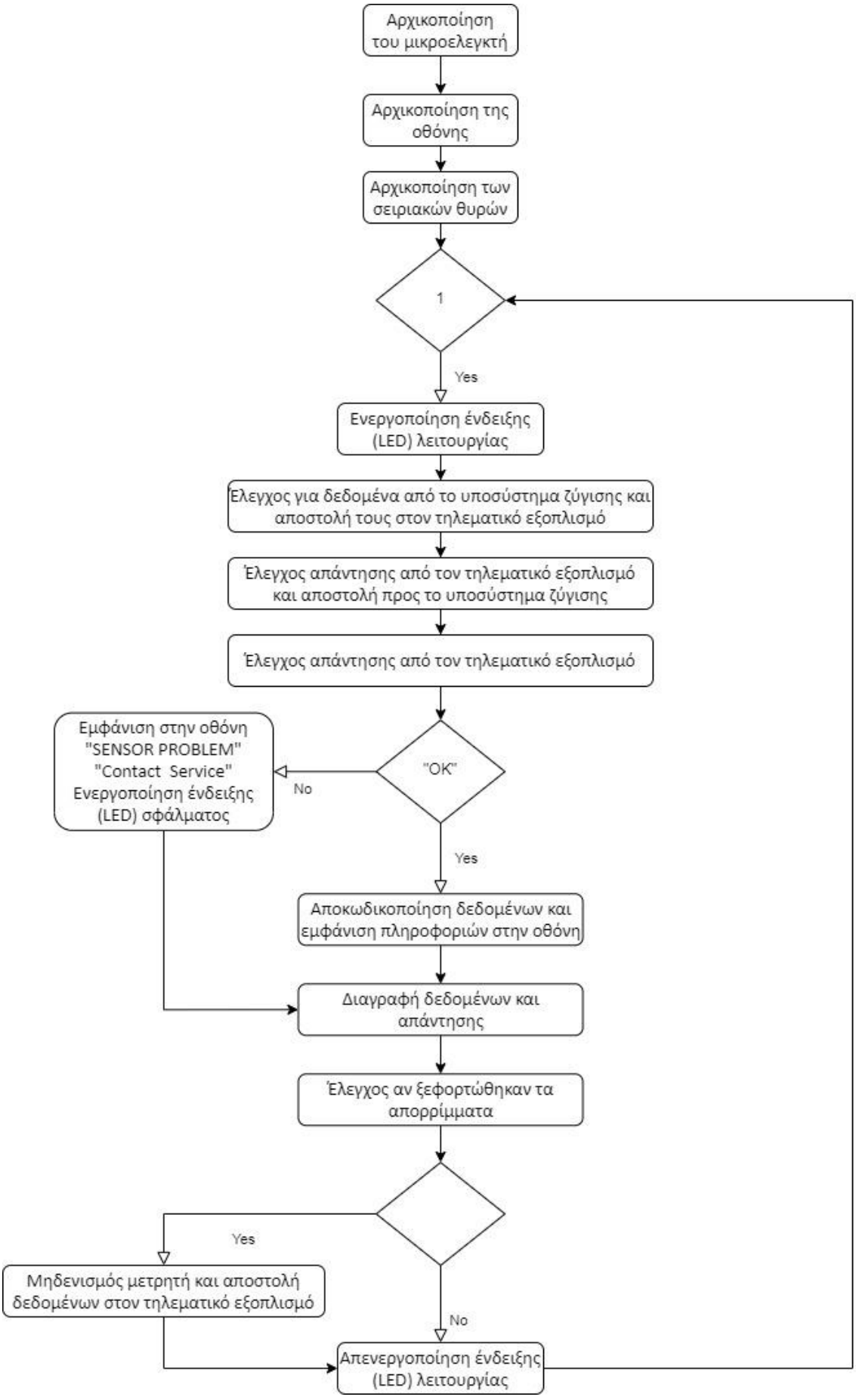
Η ανάπτυξη του προγράμματος έγινε με τη χρήση της αναπτυξιακής πλατφόρμας MPLAB IDE της εταιρείας Microchip. Η πλατφόρμα διαθέτει εργαλεία για τον γρήγορο προγραμματισμό βασικών λειτουργειών των μικροελεγκτών της εταιρείας ώστε να μειώνεται ο χρόνος ανάπτυξης των προϊόντων. Υποστηρίζει επίσης και σύνδεση διαφόρων εργαλείων ώστε να γίνεται εύκολα η αποσφαλμάτωση των υπό ανάπτυξη προγραμμάτων.

Το πρόγραμμα ξεκινάει με την αρχικοποίηση του μικροελεγκτή και των περιφερειακών του, προχωράει στον έλεγχο επικοινωνίας με τις συνδεδεμένες συσκευές (υποσύστημα ζύγισης κάδων και τηλεματικός εξοπλισμός) και στη συνέχεια διαβάζει συνέχεια τις σειριακές θήρες και την ψηφιακή είσοδο για να εκτελέσει τις κατάλληλες διεργασίες που χρειάζονται.

Αναλυτικότερα το πρόγραμμα εκτελεί τις παρακάτω ενέργειες:

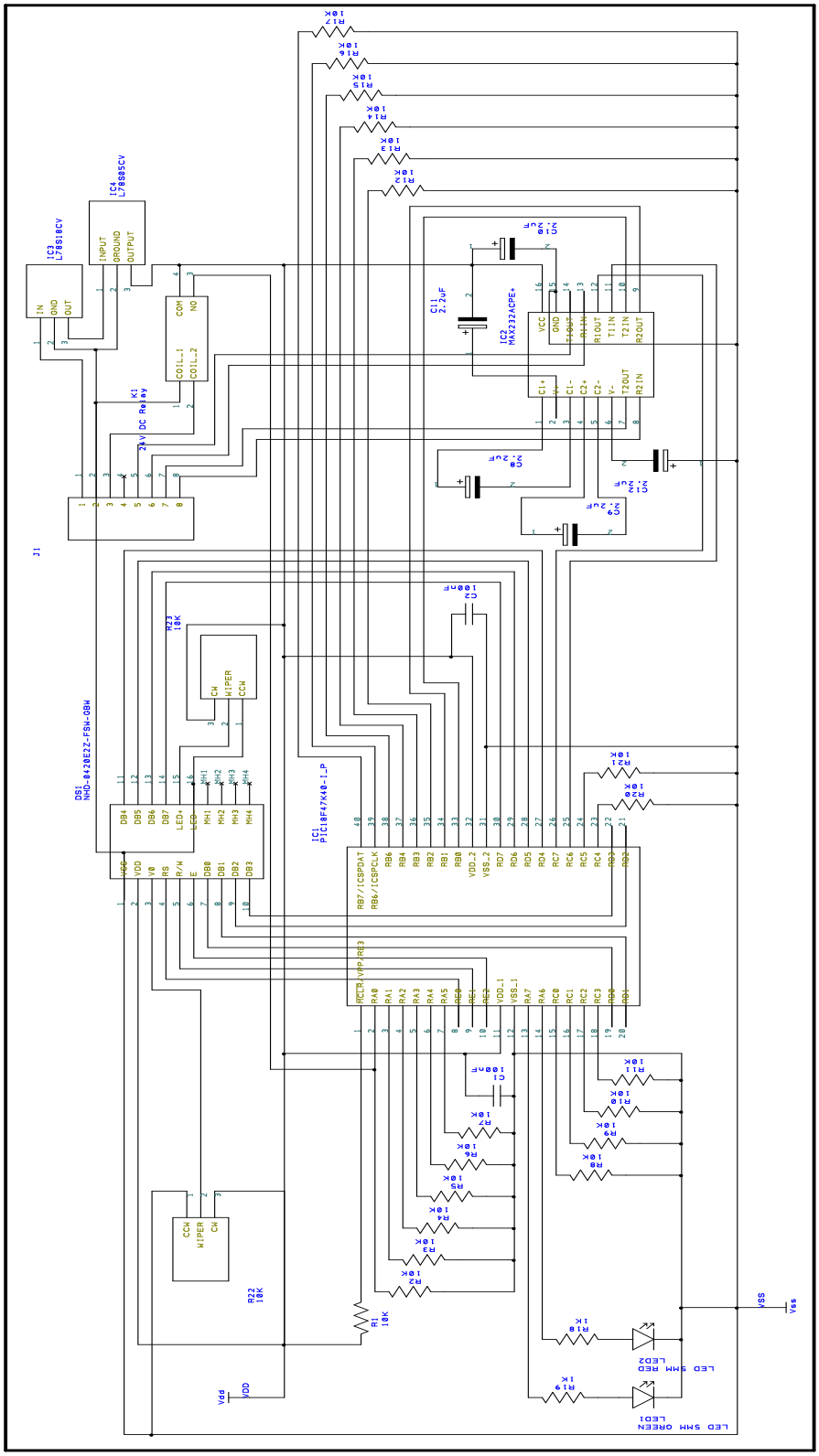
- Αρχικοποίηση του μικροελεγκτή
- Αρχικοποίηση της οθόνης
- Αρχικοποίηση των σειριακών θυρών (έλεγχος επικοινωνίας με τις συνδεδεμένες μονάδες)
- Συνεχείς ανάγνωση των θυρών για δεδομένα και της ψηφιακής εισόδου για εκτέλεση των απαραίτητων διεργασιών

Στο παρακάτω διάγραμμα ροής φαίνεται η λειτουργία του προγράμματος


Αρχικοποίηση του μικροελεγκτή
Αρχικοποίηση της οθόνης
Αρχικοποίηση των σειριακών θυρών
1
Yes
Ενεργοποίηση ένδειξης (LED) λειτουργίας
Έλεγχος για δεδομένα από το υποσύστημα ζύγισης και αποστολή τους στον τηλεματικό εξοπλισμό
Έλεγχος απάντησης από τον τηλεματικό εξοπλισμό και αποστολή προς το υποσύστημα ζύγισης
Έλεγχος απάντησης από τον τηλεματικό εξοπλισμό
"OK"
No
Εμφάνιση στην οθόνη "SENSOR PROBLEM" "Contact Service" Ενεργοποίηση ένδειξης (LED) σφάλματος
Yes
Αποκωδικοποίηση δεδομένων και εμφάνιση πληροφοριών στην οθόνη
Διαγραφή δεδομένων και απάντησης
Έλεγχος αν ξεφορτώθηκαν τα απορρίμματα
Yes
Μηδενισμός μετρητή και αποστολή δεδομένων στον τηλεματικό εξοπλισμό
No
Απενεργοποίηση ένδειξης (LED) λειτουργίας

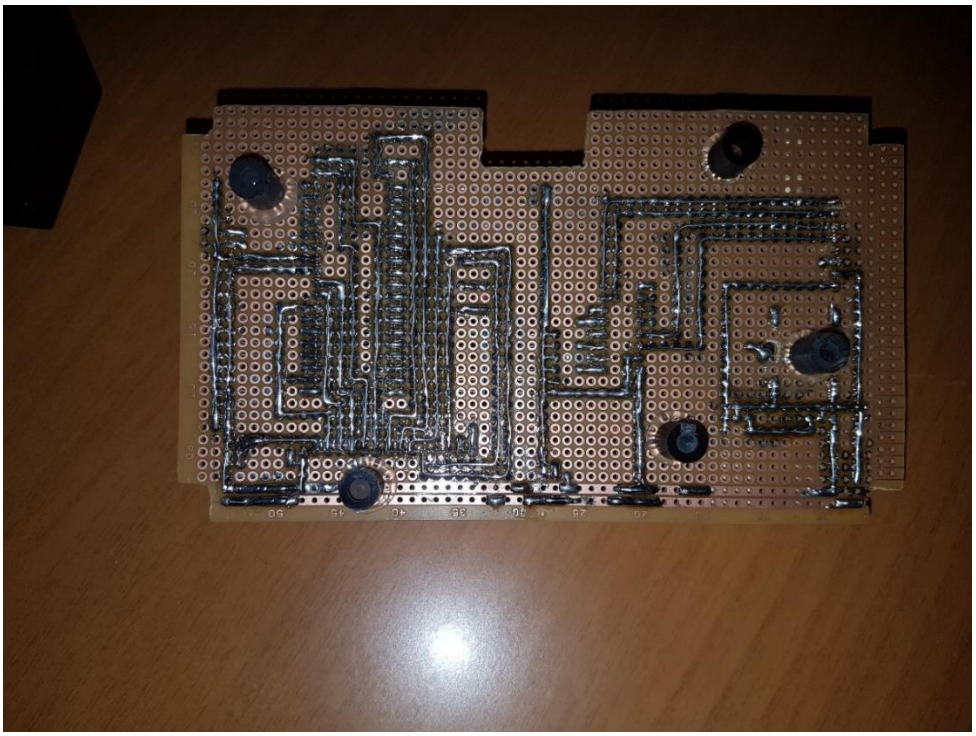
6.2.2 Σχηματικό διάγραμμα

Το σχηματικό διάγραμμα έγινε με τη χρήση του προγράμματος DesignSpark PCB 9.0 το οποίο διαθέτει δωρεάν άδεια χρήσης. Το διάγραμμα υπάρχει και στο παράρτημα Β σε μεγεθυμένη μορφή.

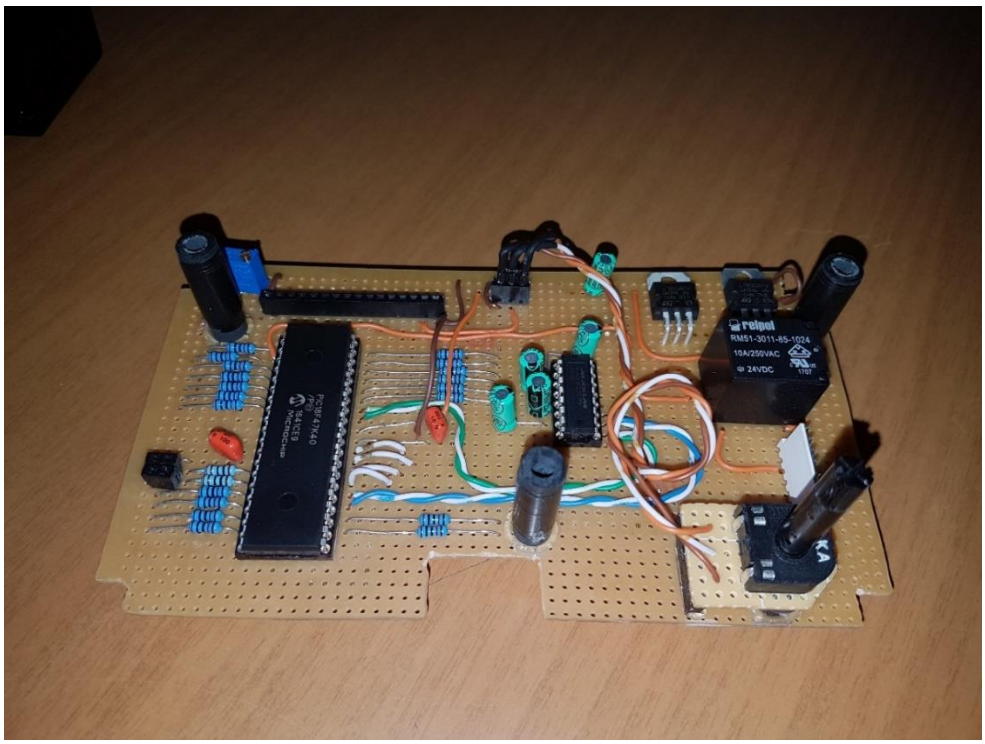


6.2.3 Ανάπτυξη πλακέτας

Για την ανάπτυξη της πλακέτας χρησιμοποιήθηκε διάτρητη πλακέτα στην οποία τοποθετήθηκαν όλα τα υλικά και χαράχτηκαν οι διάδρομοι των συνδέσεων στην συνέχεια.
Προτιμήθηκε αυτή η μέθοδος για να μειωθεί το κόστος και ο χρόνος παραγωγής της τελικής κατασκευής. Όταν υπάρξει ζήτηση για μεγαλύτερες ποσότητες θα γίνει παραγωγή τυπωμένης πλακέτας.



Εικόνα 10 Διαδρομές Πλακέτας Μονάδας



Εικόνα 11 Τοποθέτηση Εξαρτημάτων Μονάδος

Εικόνα 12 Πλακέτα και LCD Οθόνη

Εικόνα 13 Μονάδα Πληροφόρησης Οδηγού

Εικόνα 14 Μονάδα Πληροφόρησης Οδηγού τοποθετημένη στο όχημα

7. ΠΕΡΙΓΡΑΦΗ ΥΠΟΣΥΣΤΗΜΑΤΩΝ ΟΧΗΜΑΤΟΣ

7.1 ΤΗΛΕΜΑΤΙΚΟΣ ΕΞΟΠΛΙΣΜΟΣ ΟΧΗΜΑΤΩΝ

Η συσκευή Fox AVL επιτρέπει τον απομακρυσμένο εντοπισμό του οχήματος, την παρακολούθηση και την κεντρική διαχείριση του στόλου χρησιμοποιώντας το Παγκόσμιο Σύστημα Προσδιορισμού Θέσης(GPS), την υπηρεσία ασύρματης μεταφοράς δεδομένων (GPRS) και το Internet.

Η θέση του οχήματος προσδιορίζεται από το δέκτη GPS που έχει τοποθετηθεί στο όχημα, το οποίο συλλέγει δεδομένα GPS (γεωγραφικό μήκος και πλάτος, ταχύτητα, κατεύθυνση) σε πραγματικό χρόνο. Με τη χρήση του ενσωματωμένου GPRS μόντεμ είναι δυνατή η αποστολή των στοιχείων για τη θέση του οχήματος και την κατάσταση του στο κέντρο παρακολούθησης. Η συσκευή εντοπισμού οχήματος έχει περίβλημα ABS, φέρει αριθμό κατασκευής εργοστασίου και κεραίες GPS/GSM.

Είναι σχεδιασμένη και κατασκευασμένη σύμφωνα με τις απαραίτητες προδιαγραφές των κατασκευαστών των διαφόρων τύπων αυτοκινήτων. Η συσκευή έχει έγκριση τύπου CE, ETS καθώς και έγκριση 95/54 σύμφωνα με τον κανονισμό της 72/245/EEC, πού επιβάλει την πιστοποίηση όλων των ηλεκτρονικών συσκευών που τοποθετούνται σε οχήματα μετά την κατασκευή τους στο εργοστάσιο. Η συσκευή έχει δύο λυχνίες ένδειξης κατάστασης (status leds) του συστήματος, υποδοχή κάρτας SIM και υποδοχή για τις κεραίες GPS/GSM καθώς υποδοχές για τις συνδέσεις της τροφοδοσίας και των άλλων εισόδων / εξόδων. Χρησιμοποιεί το σύστημα GPS για το συνεχή υπολογισμό της θέσης του οχήματος και την υπηρεσία GPRS για την άμεση και οικονομική αποστολή και λήψη δεδομένων.

Στα βασικά της χαρακτηριστικά συμπεριλαμβάνονται:

■Μικρό μέγεθος - ευελιξία στην εγκατάσταση
■Τηλεπικοινωνίες με οποιονδήποτε συνδυασμό GPRS, SMS
■Ενσωματωμένος δέκτης GPS με εξωτερική κεραία
■Λειτουργία με τάση 12V ή 24V
■Πλήρως προγραμματιζόμενη για την κάλυψη κάθε εφαρμογής
■11 ψηφιακές και έως 4 αναλογικές είσοδοι και έως 8 ψηφιακές έξοδοι (I/O)
■Θύρα RS232 για επικοινωνία με περιφερειακές συσκευές (μία θύρα)
■Μετάδοση μηνυμάτων σε πραγματικό χρόνο
■Διαθέτει κεραία GPS υψηλής απόδοσης για προσδιορισμό της γεωγραφικής θέσης 20 καναλιών
■Υποστηρίζει το πρωτόκολλο NMEA-0183
■Λειτουργίας back up battery
■Ο πυρήνας GSM/GPRS λειτουργεί σε 4 περιοχές συχνοτήτων 850/900/1800/1900 MHz
■Δυνατότητα καταγραφής και μεταγενέστερης αποστολής μηνυμάτων
■Ασύρματη αλλαγή του προγραμματισμού ή over the air μέσω GPRS
■Διαθέτει ενσωματωμένους πυρήνες GPS/GSM/GPRS
■Η λειτουργία της μονάδος υποστηρίζει πολλαπλά προφίλ λειτουργίας και να έχει τη δυνατότητα να μεταβάλλει τη συνολική παραμετροποίηση, για παράδειγμα: συχνότητα μετάδοσης θέσης, συνθήκες συναγερμού, αλλαγή βάρδιας, επιτρεπόμενη γεωγραφική ζώνη κίνησης (Geofence), Way point. Όλη η διαχείριση μπορεί να γίνει και από τον κεντρικό εξυπηρετητή του Συστήματος.
■Δυνατότητα καταγραφής μέχρι και 30.000 μηνυμάτων και μεταγενέστερης αποστολής τους όταν αποκατασταθεί η σύνδεση με τον κεντρικό διακομιστή. Κατά το διάστημα αυτό τα στοιχεία παραμένουν ακέραια και δεν απαιτείται επικοινωνία για την αποφόρτωσή τους. Η συσκευή μπορεί να αποθηκεύσει τοπικά τα δεδομένα για πάνω από 6 μήνες ακόμα και όταν η περίοδος λήψης της θέσης είναι ρυθμισμένη στο ελάχιστο, για να το επιτύχει αυτό μέσω αλγορίθμου αλλάζει την διάρκεια λήψης σε τέτοιο βαθμό ώστε να μπορεί να εκμεταλλευτεί με το βέλτιστο τρόπο την υπάρχουσα διαθέσιμη μνήμη.

■Δυνατότητα προγραμματισμού του συστήματος (από την μονάδα ή και τον server) για αυτόματη αποστολή SMS σε περιπτώσεις που:

- το όχημα βρίσκεται σε παρατεταμένη στάση
- το όχημα υπερβαίνει το επιτρεπτό όριο ταχύτητας
- το όχημα υπερβαίνει το επιτρεπτό όριο στροφών του κινητήρα
- το όριο αποκλίνει των γεωγραφικών ορίων ευθύνης του
- έχει πατηθεί το κουμπί πανικού
- το όχημα δεν βρίσκετε στο σωστό δρομολόγιο.
- το όχημα ανυψώσει κάδο με βάρος μεγαλύτερο του επιτρεπτού

■Ανάγνωση SIM καρτών για σύνδεση με GPRS δίκτυο

■Ενσωματωμένη εφεδρική μπαταρία για αδιάλειπτη παροχή ενέργειας προς τη συσκευή

■Ασύρματη αλλαγή του προγραμματισμού over the air μέσω GPRS ή SMS, χωρίς επιστροφή του οχήματος στη βάση του (αμαξοστάσιο)

■Η μονάδα διαθέτει led καλής λειτουργίας και δυνατότητα οροδιάγνωσης βλαβών μέσω σύνδεσης σε υπολογιστή αλλά και απομακρυσμένα.

■Δυνατότητα εντοπισμού κίνησης χωρίς χρήση του κινητήρα (π.χ. ρυμούλκηση).

■Δυνατότητα προγραμματισμού του συστήματος (από την μονάδα ή και τον server) για αυτόματη αποστολή SMS σε περιπτώσεις που:

- το όχημα βρίσκεται σε παρατεταμένη στάση
- το όχημα υπερβαίνει το επιτρεπτό όριο ταχύτητας
- το όριο αποκλίνει των γεωγραφικών ορίων ευθύνης του
- το όχημα δεν βρίσκετε στο σωστό δρομολόγιο.
- το όχημα ανυψώσει κάδο με βάρος μεγαλύτερο του επιτρεπτού

■Επικοινωνία και μεταφορά δεδομένων και φωνής μέσω GSM από την ίδια μονάδα επικοινωνίας (SMS, DATA, GPRS)

■Δυνατότητα οροδιάγνωσης βλαβών και ενδείξεων καλής λειτουργίας της μονάδος.

■Δυνατότητα σύνδεσης και ελέγχου συμβάντων όπως ανύψωση κάδου, λειτουργία πρέσας/ανατροπής κλπ. κατανάλωσης καυσίμων

■Η λειτουργία της μονάδος υποστηρίζει πολλαπλά προφίλ λειτουργίας και διαθέτει τη δυνατότητα μεταβολής της συνολικής παραμετροποίησης, στις εξής παραμέτρους :

- συχνότητα μετάδοσης θέσης
- συνθήκες συναγερμού
- αλλαγή βάρδιας
- επιτρεπόμενη γεωγραφική ζώνη κίνησης (Geofence)
- Way points

Η συσκευή λειτουργεί με τροφοδοσία 9 και 36 volts. Έτσι δεν απαιτεί εξωτερικούς μετασχηματιστές τάσεως για την εφαρμογή της στα οχήματα (επιβατηγά ή και επαγγελματικά). Η κατά προσέγγιση κατανάλωση κατά την λειτουργία της είναι μέγιστο 200mA (εξαρτάται από την λειτουργία). Σε κατάσταση αναμονής η κατανάλωση υπολογίζεται ανάμεσα στο εύρος 3 έως 30 mA.

Εικόνα 15 Μονάδα GPS

7.2 ΠΕΡΙΓΡΑΦΗ ΕΞΟΠΛΙΣΜΟΥ ΖΥΓΙΣΗΣ ΚΑΙ ΤΑΥΤΟΠΟΙΗΣΗΣ ΚΑΔΩΝ

Το υποσύστημα ζύγισης και ταυτοποίησης κάδων χρησιμοποιεί σύγχρονες τεχνολογίες με σκοπό την αναγνώριση της ηλεκτρονικής ταυτότητας του κάθε κάδου, καθώς και την αυτοματοποιημένη καταγραφή του βάρους των απορριμμάτων κατά την αποκομιδή τους από το απορριμματοφόρο.

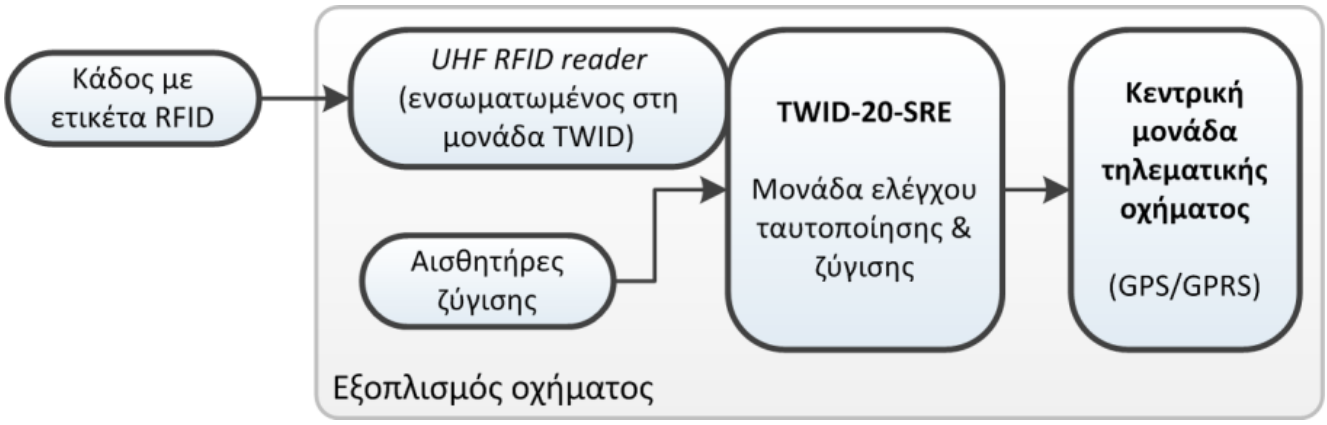
Η ηλεκτρονική ταυτοποίηση γίνεται μέσω τεχνολογίας RFID (UHF Gen2) χρησιμοποιώντας ειδική συσκευή ανάγνωσης (reader) που είναι συμβατή με τα διεθνώς αναγνωρισμένα πρότυπα ISO 18000-6C, ETSI EN 302 208 & EPCGlobal Class1 Gen2. Το σύστημα έχει τη δυνατότητα ζύγισης των απορριμμάτων και αναγνώρισης της ταυτότητας κάθε κάδου κατά την αποκομιδή του από το όχημα

Σε κάθε όχημα εγκαθίσταται μια κεντρική μονάδα ελέγχου και επεξεργασίας δεδομένων ζύγισης/ταυτοποίησης, η οποία περιλαμβάνει συσκευή ανάγνωσης ετικετών τεχνολογίας RFID (reader) και συνδέεται με αισθητήρες ζύγισης. Σε κάθε κάδο τοποθετείται μια έξυπνη ετικέτα (RFID smart tag), η οποία φέρει μοναδικό ηλεκτρονικό κωδικό ταυτότητας και είναι ειδικά σχεδιασμένη για λειτουργία σε εξωτερικές συνθήκες περιβάλλοντος.

Κατά την αποκομιδή των απορριμμάτων, ο κάδος ζυγίζεται και παράλληλα γίνεται ανάγνωση της ηλεκτρονικής ταυτότητάς του από τον αναγνώστη RFID. Η διαδικασία αυτή είναι δυναμική και εκτελείται πλήρως αυτόματα κατά την ανύψωση του κάδου χωρίς να απαιτείται καμία επιπλέον ενέργεια του προσωπικού του οχήματος. Το γενικό σχηματικό διάγραμμα των υποσυστημάτων ζύγισης και ταυτοποίησης φαίνεται στο επόμενο σχήμα.

Εικόνα 16 Ζύγιση και Ταυτοποίηση κάδων

Η αρχιτεκτονική του υποσυστήματος και οι διασυνδέσεις με τον υπόλοιπο τηλεματικό εξοπλισμό του οχήματος φαίνονται στο επόμενο μπλοκ διάγραμμα:



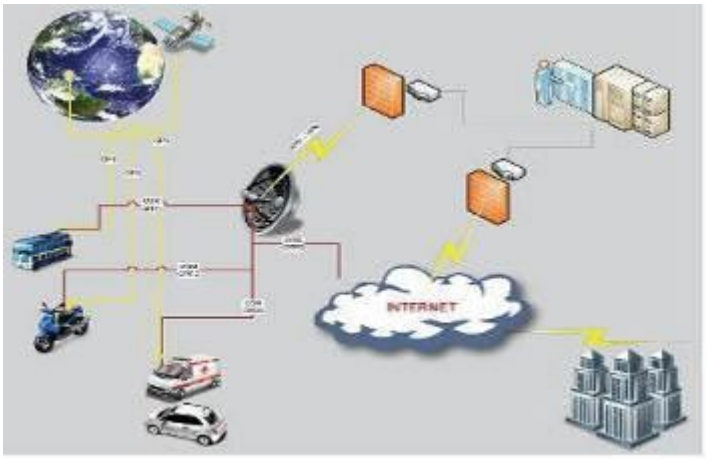
Εικόνα 17 Ζύγιση και Ταυτοποίηση κάδων

Εικόνα 18 RFID smart tag

8. ΛΟΓΙΣΜΙΚΟ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗΣ ΣΤΟΛΟΥ ΑΠΟΡΡΙΜΑΤΟΦΟΡΩΝ ΟΧΗΜΑΤΩΝ

8.1 ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ

Ο ΕΡΜΗΣ είναι ένα ολοκληρωμένο σύστημα Οργάνωσης και Διοίκησης Στόλου Οχημάτων. Παρέχει δυνατότητα πρόσβασης στο σύστημα μέσω Internet. Χρησιμοποιεί τις αποδεδειγμένα αξιόπιστες τεχνολογίες GPS (Παγκόσμιο Δορυφορικό Σύστημα Εντοπισμού Θέσης), GSM (Σύστημα Κινητής Τηλεφωνίας) και GIS (γεωγραφικό πληροφοριακό σύστημα) για να παρέχει στους χρήστες τα απαραίτητα μέσα για την αποτελεσματικότερη διαχείριση του στόλου οχημάτων σε πραγματικό χρόνο, με έμφαση στην μείωση του κόστους λειτουργίας του, αύξηση της ποιότητας εξυπηρέτησης των πελατών αλλά και της συνολικής αξιοπιστίας της επιχείρησης/ οργανισμού. Υποστηρίζει όλες τις πλατφόρμες λειτουργικών συστημάτων από Windows XP και μετά.



Εικόνα 19 Εφαρμογή ΕΡΜΗΣ

8.2 ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΑΣ ΕΡΜΗ

Ο ΕΡΜΗΣ είναι μια εφαρμογή ταυτόχρονης διαχείρισης οχημάτων, η οποία αποτελεί ένα εξαιρετικά ευέλικτο και ισχυρό εργαλείο αξιολόγησης των δραστηριοτήτων εντοπισμού πιθανών προβλημάτων και λήψης αποφάσεων για τη βελτίωση της λειτουργίας του οργανισμού, παρέχοντας:

- Δυνατότητες επιχειρησιακής αξιοποίησης των δεδομένων που αποστέλλουν οι συσκευές τηλεματικής κατά τη διάρκεια εκτέλεσης των εργασιών.
- Λειτουργίες MIS μέσω αναφορών που αξιοποιούν τα συσσωρευμένα δεδομένα του συστήματος τηλεματικής.

Ο ΕΡΜΗΣ είναι λογισμικό με εύχρηστα μενού στην ελληνική γλώσσα. Το user interface να είναι φιλικό και δίνεται μέσα σε γραφικό περιβάλλον. Μέσω του λογισμικού διαχείρισης στόλου είναι δυνατή η συνεχής παρακολούθηση της τρέχουσας θέσης των οχημάτων σε πραγματικό χρόνο, αποτυπωμένη σε ψηφιακό χάρτη, καθώς και η κατάσταση εκτέλεσης του εκάστοτε δρομολογίου των οχημάτων.
Τα γεγονότα αποστέλλονται και εμφανίζονται/καταγράφονται στο λογισμικό διαχείρισης στόλου ΕΡΜΗΣ, σε πραγματικό χρόνο, δίνοντας παράλληλα την δυνατότητα για περαιτέρω αναπαραγωγή και στατιστική επεξεργασία. Τα γεγονότα αποστέλλονται μέσω της τηλεματικής συσκευής εντοπισμού, η οποία και εγκαθίσταται στο έκαστο όχημα.

Η εφαρμογή καταγράφει στη βάση δεδομένων και στα log αρχεία την ώρα και ημ/νία αποστολής των δεδομένων από τα οχήματα σε περίπτωση αδυναμίας αποστολής λόγω μη ύπαρξης δικτύου κινητής τηλεφωνίας. Η εφαρμογή παρακολουθεί συνεχώς τα οχήματα που βρίσκονται συνδεδεμένα στο σύστημα και πληροφορεί το υπόλοιπο σύστημα για την κατάστασή τους με οπτικές ενδείξεις στον server. Επιπλέον, παρέχει δυνατότητα παραλαβής των log αρχείων ασφαλείας που κρατούνται στο σύστημα του οχήματος, σε περίπτωση διακοπής του δικτύου GPRS. Η λειτουργία αυτή επιτρέπει στον χειριστή να επιλέξει και να λάβει αρχείο log που περιέχει όλες τις πληροφορίες της διαδρομής του οχήματος, καθώς και να κάνει καταχώρηση των δεδομένων αυτών στην βάση δεδομένων του συστήματος.

Με αυτό τον τρόπο υπάρχει μία πλήρη εικόνα του δρομολογίου ενός οχήματος χωρίς την ανάγκη, το όχημα, να είναι «online». Επίσης, προσφέρει ενημέρωση της βάσης δεδομένων του εξυπηρετητή με τα αρχεία log του συστήματος και αποστέλλει τα δεδομένα θέσης οχημάτων, σημάτων συναγερμού, σημάτων κατάστασης των οχημάτων στα τερματικά του υπόλοιπου συστήματος. Η εφαρμογή αποστέλλει τις πληροφορίες που λαμβάνει από τα οχήματα, σε όλα τα τερματικά / σταθμούς εργασίας. Παράλληλα, είναι σε θέση να προσφέρει έλεγχο της διαθεσιμότητας της βάσης δεδομένων του συστήματος.

Πιο αναλυτικά, ο χρήστης μπορεί να λαμβάνει πληροφορίες για τα ακόλουθα:
✓Ημερομηνία αποστολής μηνύματος από όχημα.
✓Ώρα αποστολής μηνύματος από όχημα.
✓Τύπος μηνύματος (θέσης, κατάστασης, κ.λ.π.).
✓Ταυτότητα οχήματος που στέλνει το μήνυμα.
✓Διεύθυνση IP που έχει το όχημα στο δίκτυο.

Οι συσκευές εντοπισμού με την εκκίνηση του οχήματος ενεργοποιούνται και συνδέονται στον κεντρικό εξυπηρετητή για να στείλουν τα καταγεγραμμένα γεγονότα.

Η σύνδεση γίνεται είτε μέσω UDP πακέτων, είτε μέσω TCP ανάλογα με το μοντέλο που χρησιμοποιείται. Τα πακέτα μεταφέρονται μέσω internet ή μέσω ασφαλούς VPN ανάλογα με την εγκατάσταση και τις εκάστοτε ρυθμίσεις. Επίσης, προσφέρεται δυνατότητα να φαίνεται ο αριθμός μηνυμάτων που έχουν σταλεί στον κεντρικό εξυπηρετητή, ο αριθμός μηνυμάτων που έχουν αποσταλεί από τον Κεντρικό Διακομιστή προς τα τερματικά παρακολούθησης, καθώς και ο αριθμός μηνυμάτων που έχουν επεξεργαστεί και καταχωρηθεί στην κεντρική βάση δεδομένων.

Επίσης, η εφαρμογή προσφέρει δυνατότητα εμφάνισης των παρακάτω στοιχείων:
✓Κωδικό του μηνύματος
✓Το μήκος του μηνύματος σε bytes
✓Την ταυτότητα του οχήματος που το έστειλε
✓Ώρα μηνύματος
✓Πλάτος και μήκος θέσης
✓Κατεύθυνση οχήματος
✓Κατάσταση οχήματος
✓Καθαρό βάρος σκουπιδιών
✓Προστιθέμενο βάρος
✓Ταυτότητα κάδου (σε σύνδεση με αισθητήρα)
✓Ροή καυσίμων – κατανάλωση (σε σύνδεση με αισθητήρα σε μελλοντική επέκταση)
✓Κωδικός συναγερμού (σε σύνδεση με αισθητήρα σε μελλοντική επέκταση)

Τα γεγονότα που διαχειρίζεται το λογισμικό είναι διάφορα, όπως ignition, στίγμα, υπέρβαση ταχύτητας, είσοδος/έξοδος σε geofence, θερμοκρασία, driver id, μετρήσεις αισθητήρων, μέτρησης καυσίμων**,** βάρους κάδων, κ.α. Για τον προσδιορισμό της γεωγραφικής θέσης χρησιμοποιούνται

δορυφόροι GPS που έχουν παγκόσμια κάλυψη. Τα πακέτα πληροφοριών όταν φτάνουν στον κεντρικό εξυπηρετητή αποθηκεύονται στη βάση δεδομένων αφού πρώτα επεξεργαστούν, ώστε να παρέχουν επιπλέον ενημέρωση για τυχόν άλλα γεγονότα, όπως είναι η υπέρβαση ταχύτητας, είσοδος/έξοδος σε geofence (alert βάση αποθηκευμένων σεναρίων).

Ταυτόχρονα τα πακέτα πληροφοριών επεξεργάζονται και από τη μηχανή reporting, ώστε να γίνεται σύνοψη των στιγμάτων, τα οποία στη συνέχεια χρησιμοποιούνται στις αναφορές προς τους χρήστες. Οι χρήστες συνδέονται από απόσταση με τον κεντρικό εξυπηρετητή, εισάγοντας τα στοιχεία ασφαλείας τους. Η σύνδεση μπορεί να είναι είτε μέσω internet, είτε μέσω Lan, είτε και μέσω VPN για μεγαλύτερη ασφάλεια. Το σύστημα φιλτράρει τα οχήματα και τις πληροφορίες βάση του χρήστη που συνδέθηκε. Η αρχιτεκτονική του λογισμικού είναι Client – Server και υποστηρίζει τα λειτουργικά συστήματα LINUX, WINDOWS.

Η εμφάνιση συγκεκριμένων οχημάτων στο χάρτη θα είναι με κριτήρια όπως ταυτότητα και όνομα κατηγορία, αρ. κυκλοφορίας, τρέχων οδηγός, στίγμα, ταχύτητα και απόσταση του οχήματος.

- Η αναπαραγωγή διαδρομής οχήματος με ή χωρίς ίχνος προσφέρεται με δυνατότητα ανεξάρτητης μεγέθυνσης και σμίκρυνσης στο παράθυρο.
- Η επιλογή οχήματος και η παρακολούθηση της τροχιάς του στο χάρτη είναι σε πραγματικό χρόνο προς όποια κατεύθυνση κινείται.
- Η δημιουργία χειριστών του κόμβου για παρακολούθηση των οχημάτων έχει την δυνατότητα χειρισμού από πολλούς χρήστες (Multi user) όπου ο καθένας θα μπορεί
να παρακολουθεί τα οχήματα της δικαιοδοσίας του.
- Στην εφαρμογή θα περιλαμβάνονται οδηγίες χρήσης και διαχείρισης στα Ελληνικά
- Παρέχεται η δυνατότητα παρακολούθηση on-line του οχήματος με/χωρίς ίχνος (τροχιά) για λιγότερο από 10 δευτερόλεπτα (στην εν λόγω περίπτωση 5) και ο χρόνος να μπορεί να ρυθμιστεί. Ταυτόχρονα να εμφανίζονται οι θέσεις των οχημάτων στο πίνακα «κατάστασης οχημάτων», στιγμιαία ταχύτητα (GPS), η συνολική διανυθείσα απόσταση και ο οδηγός του οχήματος εφόσον υπάρχει στο όχημα .
- Υπάρχει η δυνατότητα από τον χρήστη να επιλέξει συγκεκριμένο χρονικό (ημερολογιακό) διάστημα που τον ενδιαφέρει, να μπορεί να αλλάζει τα χαρακτηριστικά του απεικονιζόμενου στο χάρτη δρομολογίου, όπως πάχος και χρωματισμός γραμμής, το στυλ και το μέγεθος των σημείων (στιγμάτων), καθώς και να αποτυπώσει το ίχνος του οχήματος ως μία διαδρομή.
- Εμφανίζεται ο συνολικός χρόνος στάσης/στάθμευσης .
- Παρέχεται η μέτρηση της διανυθείσας απόστασης σε χιλιόμετρα .
- Παρέχεται η δυνατότητα στον χρήστη να τον ειδοποιεί αυτόματα το σύστημα ότι το όχημα δεν στέλνει συντεταγμένες, δεν έχει δίκτυο κινητής τηλεφωνίας GPRS .
- Στο χάρτη υπάρχει η δυνατότητα εισαγωγής ή εύρεσης σημείων ενδιαφέροντος με βάση γεωγραφικές συντεταγμένες, με οδό και αριθμό.
- Ο χρήστης έχει την δυνατότητα να δημιουργήσει μία θέση ενδιαφέροντος όπως Νοσοκομεία, Κλινικές, Δημόσια πάρκα, Σχολεία, Γήπεδα κλπ. Κάθε χρήστης θα έχει δικό του κωδικό πρόσβασης στην εφαρμογή με τα ανάλογα δικαιώματα. Ο υπεύθυνος του συστήματος (administrator) θα μπορεί να μεταβάλει τις ρυθμίσεις πρόσβασης στην εφαρμογή και συγκεκριμένα :
- Τον ορισμό κωδικών πρόσβασης ανά χρήστη
- Την πρόσβασης χρηστών μέσω κωδικού χρήστη και κωδικοποιημένου συνθήματος (Password) και θα έχει ελεγχόμενη διάρκεια.
- Τα επίπεδα πρόσβασης χρηστών με τον ορισμό των λειτουργιών που θα δικαιούται να εκτελέσει, όπως και τις κατηγορίες οχημάτων που θα του επιτρέπεται να διαχειρίζεται. Με τον τρόπο αυτό ο διαχειριστής του συστήματος (administrator) θα έχει τη δυνατότητα να δημιουργεί πολλαπλά επίπεδα δικαιωμάτων με απλό και εύχρηστο τρόπο και κατηγορίες χρηστών (User, super-users and Administrator accounts). Όπως για παράδειγμα πρόσθεση ή αφαίρεση πληροφοριών – εικονιδίων πάνω στους χάρτες ανάλογά με το επίπεδο ασφαλείας που έχουμε ορίσει.

➢Τον πλήρη έλεγχο και τη διαχείριση όλου του συστήματος και των παραμέτρων αυτού από τον υπεύθυνο (administrator) του συστήματος ή από κατάλληλα εξουσιοδοτημένα άτομα. Το σύστημα θα έχει τη δυνατότητα αυτόματης δημιουργίας αναφορών για την δραστηριότητα ενός οχήματος ή και μιας ομάδας οχημάτων που ανήκουν στον ίδιο στόλο ή κατηγορία για το χρονικό διάστημα που του ζητηθεί. Διατηρείται η δυνατότητα δημιουργίας νέων αναφορών σύμφωνα με τις ανάγκες του Δήμου. Θα υπάρχει δυνατότητα άμεση εξαγωγή στοιχείων από τη Βάση Δεδομένων και η επιλογή του επιθυμητού χρονικού διαστήματος.

Οι αναφορές συστήματος θα παρέχονται είναι:
✓Συγκριτικές αναφορές ανά ομάδα οχημάτων
✓Αναλυτικές αναφορές ανά όχημα
✓Μεγάλος αριθμός προκαθορισμένων αναφορών
✓Δυνατότητα δημιουργίας εξειδικευμένων αναφορών .

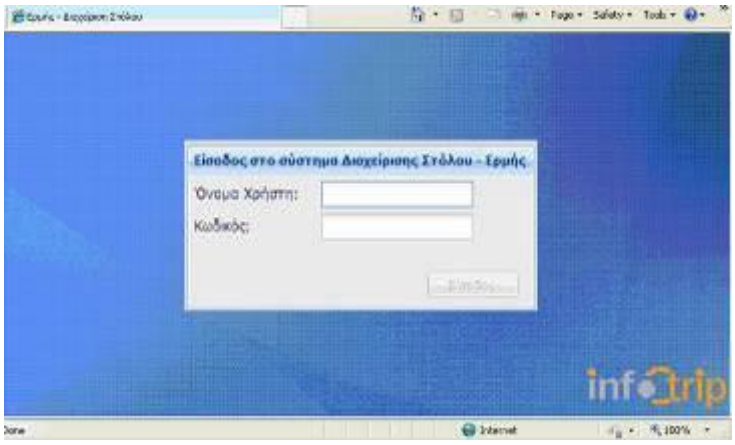
Από το server θα υπάρχει η δυνατότητα για αυτόματη αποστολή SMS σε περιπτώσεις που:
✓το όχημα βρίσκεται σε παρατεταμένη στάση
✓το όχημα υπερβαίνει το επιτρεπτό όριο ταχύτητας
✓το όριο αποκλίνει των γεωγραφικών ορίων ευθύνης του
✓Το όχημα δεν βρίσκεται στο σωστό δρομολόγιο
✓Ανύψωση κάδου μεγαλύτερου από το επιτρεπτό βάρος
✓Κάθε προβολή χάρτη θα μπορεί να εκτυπωθεί, αποθηκευτεί ή να αντιγραφεί.

8.3 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΕΦΑΡΜΟΓΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΣΤΟΛΟΥ

8.3.1 ΕΡΜΗΣ ΔΙΑΧΕΙΡΙΣΗ ΧΡΗΣΤΩΝ

Η πρόσβαση στην εφαρμογή γίνεται μέσω internet, χωρίς την απαίτηση εγκατάστασης οποιασδήποτε εφαρμογής, στον υπολογιστή κάθε εξουσιοδοτημένου χρήστη.

Κάθε χρήστης έχει δικό του κωδικό πρόσβασης στην εφαρμογή, ανάλογα με τα δικαιώματα που του έχουν ανατεθεί. Υπάρχει δυνατότητα χειρισμού της εφαρμογής από πολλούς χρήστες (multi user). Ελάχιστος αριθμός χρηστών κόμβου ένας (1) με δυνατότητα επέκτασης της παρακολούθησης των οχημάτων από περισσότερους από ένα χρήστες του λογισμικού (multiuser).



Εικόνα 20 Είσοδος στο σύστημα

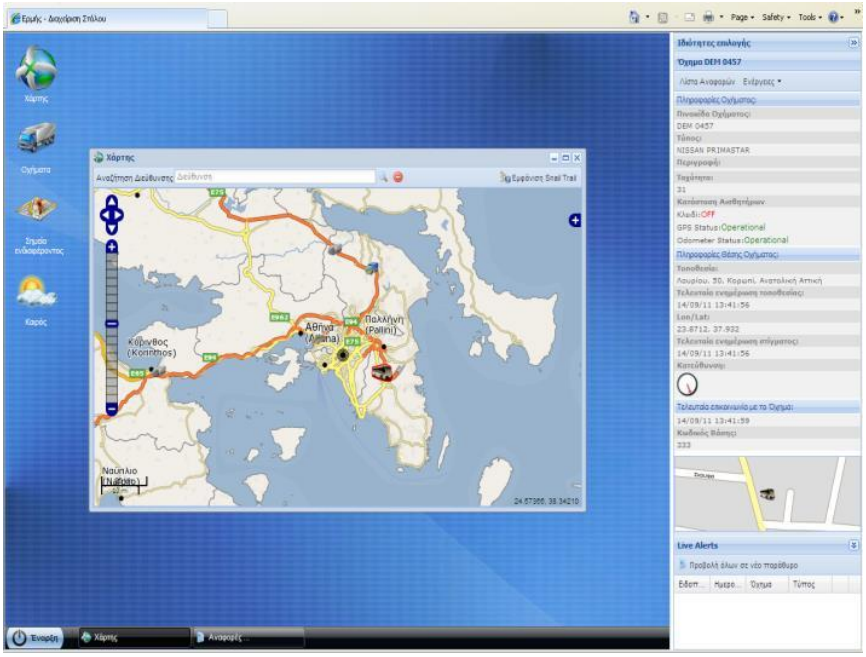
Ο υπεύθυνος του συστήματος (administrator) μπορεί να μεταβάλει τις ρυθμίσεις πρόσβασης στην εφαρμογή και πιο συγκεκριμένα:
➢ Τον ορισμό κωδικών πρόσβασης ανά χρήστη

- Την πρόσβαση χρηστών μέσω κωδικού χρήστη και κωδικοποιημένου συνθήματος (Password) με ελεγχόμενη διάρκεια.
- Τα επίπεδα πρόσβασης χρηστών με τον ορισμό των λειτουργιών που δικαιούται να εκτελεί, όπως και τις κατηγορίες οχημάτων που του επιτρέπετε να διαχειρίζεται. Με τον τρόπο αυτό ο διαχειριστής του συστήματος (administrator) έχει τη δυνατότητα να δημιουργεί πολλαπλά επίπεδα δικαιωμάτων με απλό και εύχρηστο τρόπο (User, super-users and Administrator accounts) όπως για παράδειγμα πρόσθεση ή αφαίρεση πληροφοριών – εικονιδίων πάνω στους χάρτες ανάλογά με το επίπεδο ασφαλείας που έχει ορίσει.
- Να ορίσει τις κατηγορίες οχημάτων που θα επιτρέπεται να παρακολουθεί.
- Τον πλήρη έλεγχο και τη διαχείριση όλου του συστήματος και των παραμέτρων αυτού από τον υπεύθυνο (administrator) του συστήματος ή από κατάλληλα εξουσιοδοτημένα άτομα.

8.3.2 ΑΠΕΙΚΟΝΙΣΗ ΚΑΙ ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΟΧΗΜΑΤΩΝ

Ο χρήστης του συστήματος μπορεί να παρακολουθεί πως κινούνται σε πραγματικό χρόνο (on – line) ταυτόχρονα όλα τα οχήματα του στόλου των οχημάτων του στόλου με απεικόνιση της θέσης του οχήματος σε ψηφιακό χάρτη, καθώς επίσης και να επιλέγει το κατάλληλο εικονίδιο, το οποίο απεικονίζει τη θέση του εκάστοτε οχήματος στο κέντρο του ψηφιακού χάρτη. Το εικονίδιο είναι δυναμικό και μπορεί να αλλάζει χρώμα ανάλογα με την κατάσταση του οχήματος.

Επιπλέον, παρέχεται δυνατότητα άμεσης αναφοράς θέσης επιλεγμένων οχημάτων ή όλων των οχημάτων του στόλου στον χάρτη ή σε πίνακα με κριτήρια όπως κατηγορία οχήματος, αριθμός κυκλοφορίας, τρέχων οδηγός, στίγμα, ταχύτητα, τρέχων οδηγός, απόσταση οχήματος, κτλ.



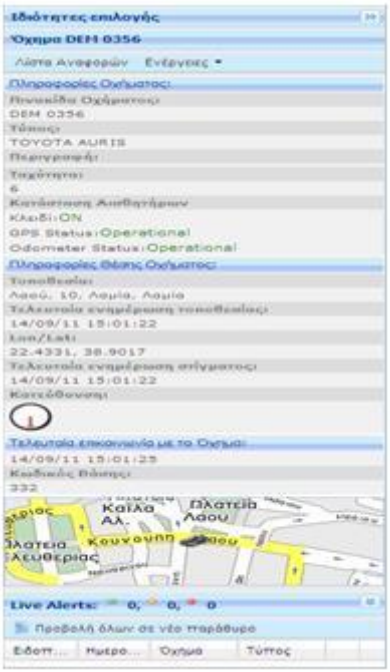
Εικόνα 21 Απεικόνιση οχημάτων

Το σύστημα μεταδίδει τη θέση και την κατάσταση του οχήματος on-line ανά 5 δευτερόλεπτα, ανά συγκεκριμένη διανυθείσα απόσταση, κάθε φορά που ανοίγει και κλείνει ο κινητήρας. Ο χρήστης της εφαρμογής, μπορεί να αναζητήσει πληροφορία για την τρέχουσα θέση του οχήματος ανά πάσα στιγμή. Η τηλεματική μονάδα εντοπισμού λαμβάνει ασύρματα το αίτημα και μεταδίδει άμεσα την θέση του οχήματος. Έτσι, μέσω εύκολης και γρήγορης επιλογής εμφανίζονται οι θέσεις των οχημάτων

στο πίνακα «κατάστασης οχημάτων», η στιγμιαία ταχύτητα (GPS), η συνολική διανυθείσα απόσταση και ποιος είναι ο οδηγός του οχήματος (ταυτοποίηση οδηγού με όχημα).

Πιο αναλυτικά, στη λίστα κατάστασης οχημάτων απεικονίζονται οι εξής πληροφορίες:

- Α/Α οχήματος
- Όνομα οχήματος
- Αριθμός κυκλοφορίας
- Κατηγορία οχήματος
- Χρονική διάρκεια από το τελευταίο στίγμα
- Τρέχουσα ταχύτητα οχήματος
- Διανυθείσα απόσταση
- Κατεύθυνση οχήματος
- Ακριβής διεύθυνση



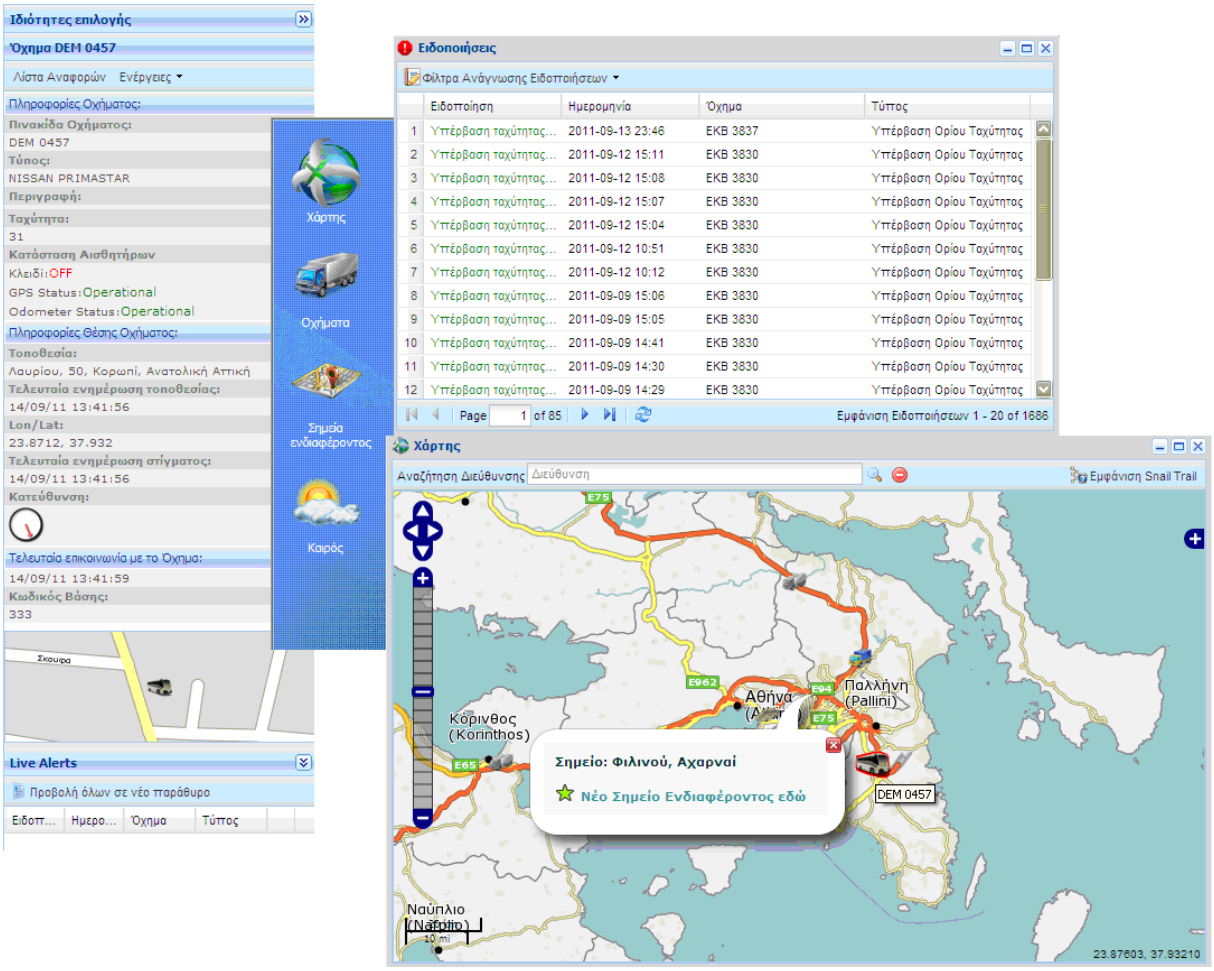
Εικόνα 22 Λίστα Κατάστασης Οχημάτων

Μέσω της συσκευής οχήματος το κέντρο μπορεί να ενημερώνεται σε πραγματικό χρόνο για καταστάσεις, όπως υπέρβαση ταχύτητας, θέση σε λειτουργία της μηχανής του οχήματος, καθώς και άλλα γεγονότα που μπορεί να παρακολουθούνται σ' ένα όχημα μέσω των ψηφιακών εισόδων του εξοπλισμού οχήματος. Επιπλέον, υπάρχει η δυνατότητα για κάθε διαφορετική κατάσταση στην οποία βρίσκεται το όχημα, να μεταβάλλεται το χρώμα με το οποίο απεικονίζεται το εν λόγω όχημα στον ψηφιακό χάρτη, καθώς και να εμφανίζεται ειδικό παράθυρο δηλωτικό της κατάστασης συναγερμού.

Όλα τα μηνύματα που λαμβάνονται μπορούν να καταγράφονται στη βάση δεδομένων, ώστε να είναι διαθέσιμα για στατιστικές αναφορές.

Επίσης, υπάρχει η δυνατότητα ο χρήστης να μπορεί να επιλέξει για κάποιο συγκεκριμένο χρονικό (ημερολογιακό) διάστημα, να λάβει αναφορά για την κίνηση του οχήματος, δυνατότητα αλλαγής των χαρακτηριστικών του δρομολογίου που εμφανίζεται στον χάρτη, όπως πάχος και χρωματισμός γραμμής, το στυλ και το μέγεθος των σημείων (στιγμάτων).

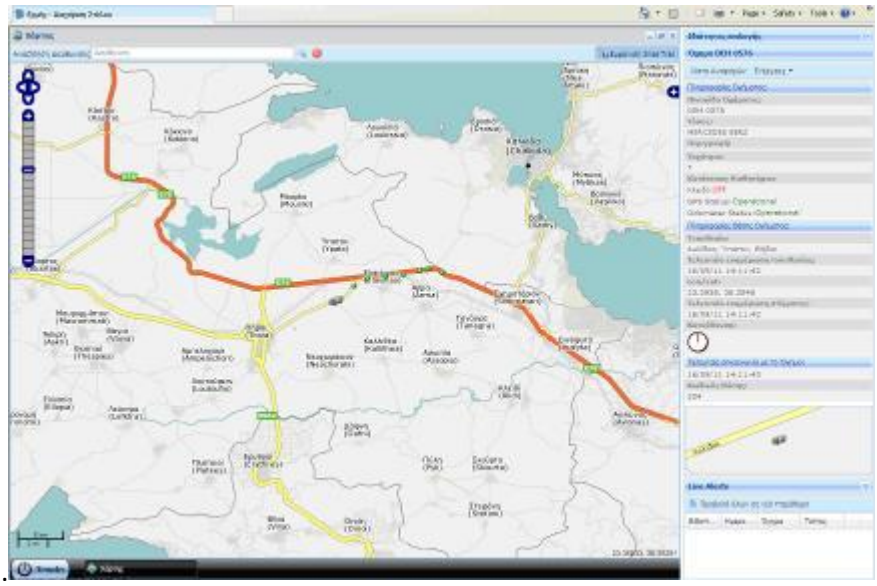
Ο χάρτης δεν ανανεώνεται παρά μόνο η θέση των οχημάτων σε αυτόν. Κάθε νέο στίγμα ταυτόχρονα θα εισάγεται στην βάση δεδομένων του server για αποθήκευση και θα αποστέλλεται σε πραγματικό χρόνο σε όλα τα τοπικά και απομακρυσμένα τερματικά clients του κεντρικού συστήματος (διαφορετικά σημεία πρόσβασης). Η εμφάνιση της νέας θέσης των οχημάτων θα είναι άμεση (real time) στην οθόνη του κάθε client. Δεν θα γίνεται συνολική ανανέωση της θέσης των οχημάτων, παρά μόνο των οχημάτων που κινήθηκαν.



Εικόνα 23 Εφαρμογή ΕΡΜΗ

8.3.3 ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΟΧΗΜΑΤΟΣ ΜΕ ΙΧΝΟΣ (LIVE SNAILTRAIL)

Αυτή η λειτουργία ενεργοποιείται κατά την διάρκεια παρακολούθησης του οχήματος που είναι εν κινήσει και εμφανίζει επιπλέον τη διαδρομή του οχήματος με μια ακολουθία γραμμών πάνω στον χάρτη



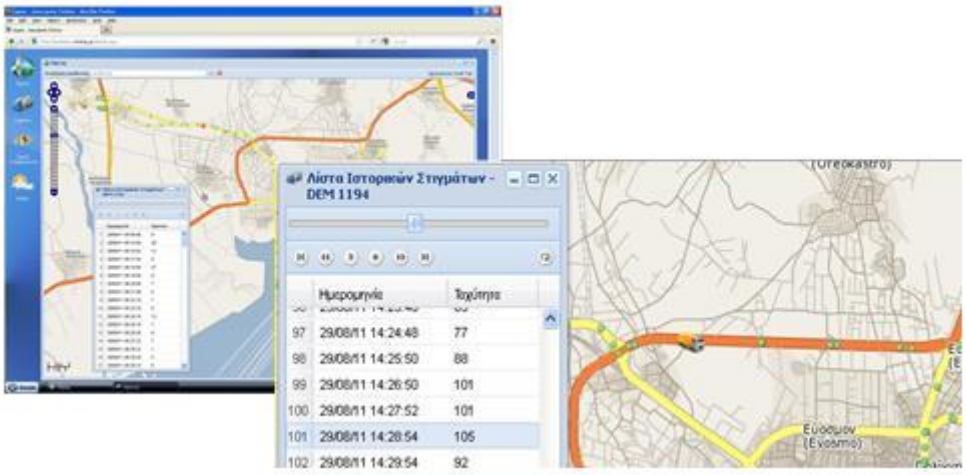
Εικόνα 24 Απεικόνιση Χάρτη

8.3.4 ΑΝΑΠΑΡΑΓΩΓΗ ΔΙΑΔΡΟΜΗΣ ΟΧΗΜΑΤΟΣ – ΙΣΤΟΡΙΚΟΣΤΙΓΜΑΤΩΝ

Με την επιλογή αυτή ο χρήστης μπορεί να αποτυπώσει την διαδρομή του οχήματος, με μια συνεχόμενη γραμμή πάνω στον χάρτη όπου με drop down επιλέγει απευθείας τη διαδρομή ενός ή περισσοτέρων οχημάτων καθώς και να ορίσει να λάβει πληροφόρηση για ορισμένο χρονικό διάστημα.

Ανάλογα με την βάση και το πλήθος των ιστορικών στιγμάτων που είναι διαθέσιμα ο χρήστης μπορεί να εμφανίσει πολλές εβδομάδες / μήνες πριν ιστορικό στιγμάτων και γεγονότων. Επιπλέον, παρέχεται δυνατότητα ανεξάρτητης μεγένθυσης και σμίκρυνσης στο παράθυρο για κάθε όχημα.

Η εφαρμογή παρέχει δυνατότητα τήρησης ιστορικών δεδομένων που συλλέγονται από τα οχήματα (δεν υπάρχει περιορισμός οχημάτων), τα οποία δουλεύουν 24 ώρες το 24ώρο, για επτά ήμερες την εβδομάδα για τουλάχιστον 1 χρόνο. Ο κύκλος τήρησης των δεδομένων είναι πλήρως παραμετροποιησίμος.

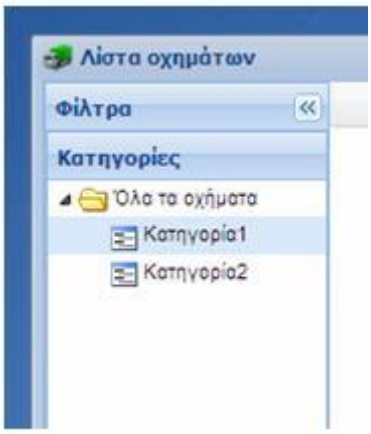


Εικόνα 25 Αναπαραγωγή διαδρομής οχήματος – Ιστορικό στιγμάτων

8.3.5 ΚΑΤΗΓΟΡΙΕΣ ΟΧΗΜΑΤΩΝ - ΕΜΦΑΝΙΣΗ / ΑΠΟΚΡΥΨΗ ΕΚΑΣΤΟΥ ΟΧΗΜΑΤΟΣ

Το σύστημα παρέχει τη δυνατότητα δημιουργίας κατηγοριών οχημάτων ανάλογα με τον τύπο τους ή τον χρήστη και από την λίστα κατάστασης οχημάτων υπάρχει η δυνατότητα εμφάνισης/απόκρυψης στον/από τον χάρτη οποιουδήποτε οχήματος.

Επίσης, υπάρχει η ευκολία εμφάνισης/απόκρυψης όλων των οχημάτων, εφόσον αυτά ανήκουν σε συγκεκριμένες κατηγορίες.

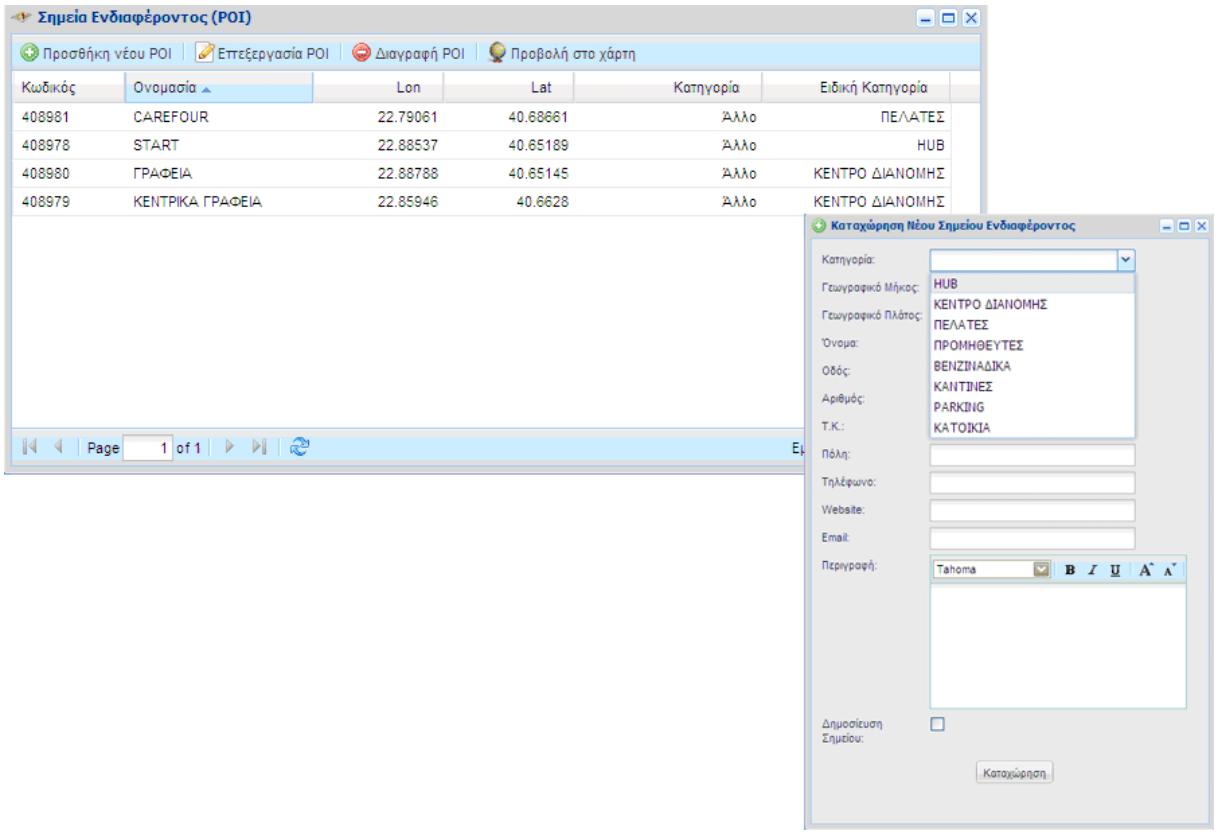


Εικόνα 26 Κατηγορίες οχημάτων

8.3.6 ΣΗΜΕΙΑ ΕΝΔΙΑΦΕΡΟΝΤΟΣ

Η εφαρμογή διαχειρίζεται τα σημεία ενδιαφέροντος, τα οποία μπορούν να απεικονίζονται στο χάρτη με διαφορετικό εικονίδιο το έκαστο, ανάλογα με την κατηγορία στην οποία ανήκει.

Η αναζήτηση σημείου ενδιαφέροντος γίνεται είτε με την κατηγορία drop down menu, είτε ονομαστικά εισάγοντας τα αρχικά του σημείου ενδιαφέροντος. Επιπλέον, παρέχει δυνατότητα συσχετισμού θέσης στάσης οχημάτων με επιλεγμένα σημεία ενδιαφέροντος.



Εικόνα 27 Προσθήκη Σημείου Ενδιαφέροντος

Επίσης, ο χρηστής έχει την δυνατότητα να δημιουργήσει μία θέση ενδιαφέροντος, όπως Νοσοκομεία, Κλινικές, Δημόσια πάρκα, Σχολεία, Γήπεδα, κλπ.

Γνωρίζοντας την διεύθυνση του σημείου ενδιαφέροντος που επιθυμεί να προσθέσει, το αναζητά και του δίνει το όνομα που επιθυμεί μαζί με πληροφορίες όπως κατηγορία, τηλέφωνο, e-mail, κτλ. Επιπρόσθετα, ο χρήστης μπορεί να γράψει σχόλια επιλέγοντας το σημείο του χάρτη με τον κέρσορα, όπου θα υπάρχει εικονίδιο και μπορεί να εμφανίζει παράθυρο με όλες τις πληροφορίες. Έτσι, ο χρήστης μπορεί να ζητήσει να λάβει αναφορά κίνησης οχήματος ανά σημείο ενδιαφέροντος, π.χ. πόσες φορές πέρασε ένα όχημα /ή το σύνολο των οχημάτων από το επιλεγμένο σημείο ενδιαφέροντος.

Εικόνα 28 Απεικόνιση Σημείου Ενδιαφέροντος στον χάρτη

8.3.7 ΓΕΩΚΩΔΙΚΟΠΟΙΗΣΗ (GEOCODING) ΚΑΙ ΑΝΤΙΣΤΡΟΦΗ ΓΕΩΚΩΔΙΚΟΠΟΙΗΣΗ (REVERSE GEOCODING)

Η γεωκωδικοποίηση επιτρέπει στο χρήστη την εύρεση μιας διεύθυνσης στο χάρτη. Η διεύθυνση ορίζεται από το όνομα του δρόμου ή διασταύρωση, με αριθμό ή Τ.Κ., ή περιοχή/δήμο. Στην περίπτωση που το αποτέλεσμα της αναζήτησης επιστρέψει περισσότερα του ενός αποτελέσματα, ο χρήστης καλείται να επιλέξει το επιθυμητό. Με βάση τη λειτουργία αυτή είναι δυνατή η εισαγωγή στοιχείων που αφορούν τη θέση κάδων καθαριότητας και των λοιπών σημείων ενδιαφέροντος.

Ο τρόπος εμφάνισης της διεύθυνσης στο χάρτη είναι πλήρως παραμετροποιημένος από το χρήστη. Αντίστροφα επίσης έχει την δυνατότητα, δίνοντας γεωγραφικές συντεταγμένες γνωστές από άλλες πηγές όπως π.χ. GPS να εντοπιστούν στον χάρτη και να μετατραπούν σε ταχυδρομική διεύθυνση (Reverse Geocoding).

8.3.8 ΠΛΗΡΟΦΟΡΙΕΣ ΓΙΑ ΤΟΝ ΚΑΙΡΟ

Το σύστημα παρέχει πληροφορίες για τις καιρικές συνθήκες, εμφανίζοντας πάνω στον χάρτη αντίστοιχο εικονίδιο ανάλογα των καιρικών συνθηκών (ηλιοφάνεια, βροχόπτωση, συννεφιά, κτλ.) που επικρατούν από περιοχή. Ειδικότερα παρέχονται πληροφορίες όπως θερμοκρασία, κατεύθυνση και ταχύτητα ανέμου, κτλ.

Εικόνα 29 Απεικόνιση οχημάτων

Καιρός

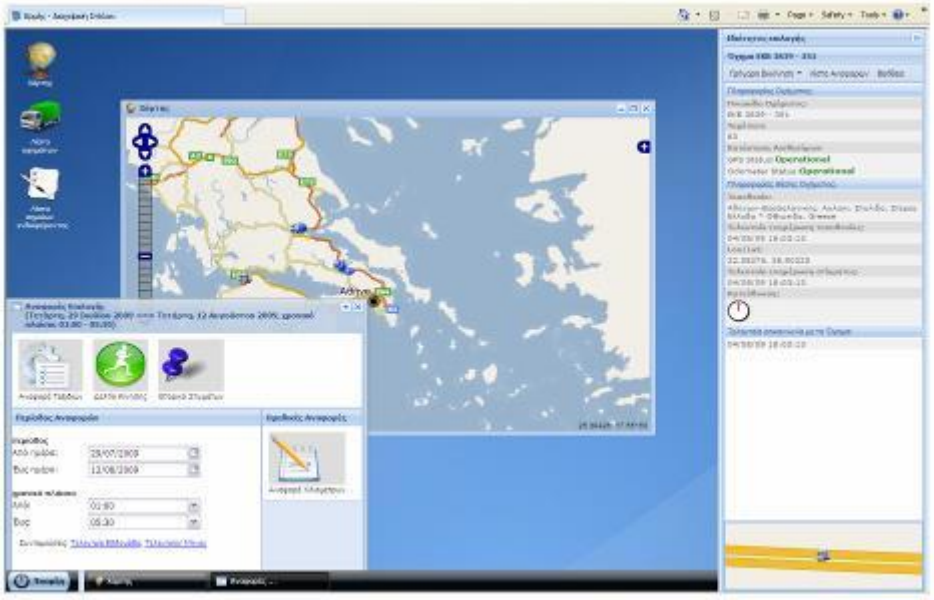
Προβολή στο χάρτη

	Κωδικός Σταθμού	Όνομα Σταθμού	Κατάσταση	Θερμοκρασία	Κατεύθυνση Ανέμου	Ταχύτητα Ανέ..
1	LGAL	Alexandroupolis (Military)	50% Chance of Storms	23	ΒΑ	27
2	LGAD	Andravida (Air Force Base)	Rain	24	ΒΒΔ	16
3	LGRX	Araxos (Air Force Base)	Rain	23	ΔΒΔ	14
4	LGAV	Athens Eleftherios Venizel...	Rain	23	ΒΑ	23
5	LGEL	Elefsis (Air Force Base)	30% Chance of Storms	24	Β	11
6	LGKL	Kalamata (Air Force Base)	30% Chance of Storms	24	ΒΒΔ	11

Εικόνα 30 Πληροφορίες Καιρού

8.3.9 ΙΣΤΟΡΙΚΑ ΔΕΔΟΜΕΝΑ

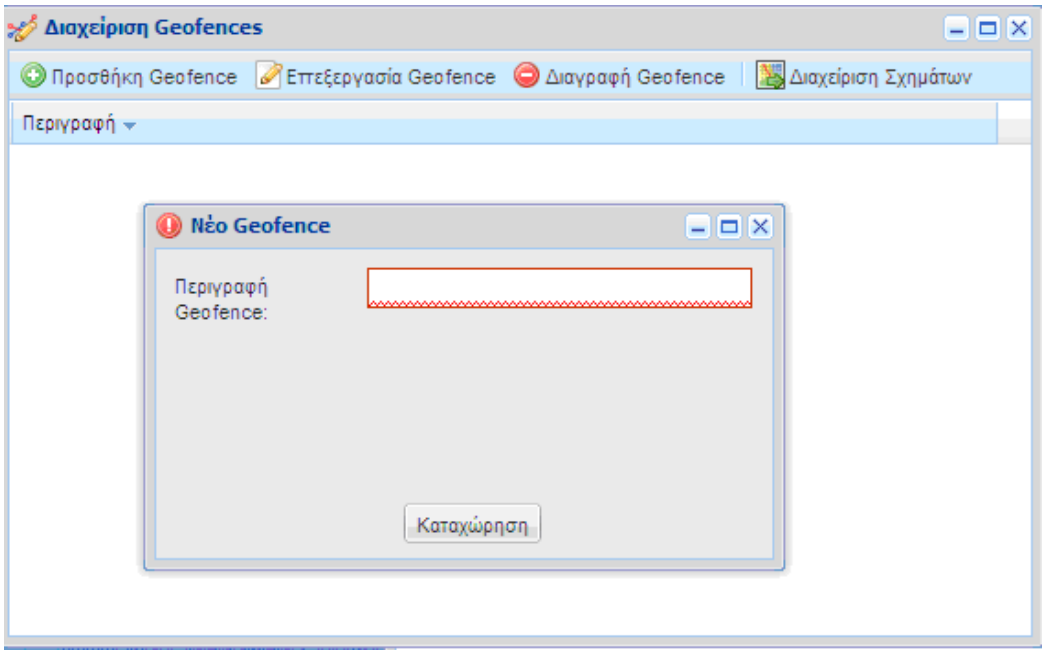
Τα στοιχεία που συλλέγονται από όλα τα οχήματα (δεν υπάρχει περιορισμός οχημάτων), τα οποία δουλεύουν 24 ώρες το 24ώρο, για επτά ήμερες την εβδομάδα, την τρέχουσα στιγμή μπορούν να αποθηκεύονται και να διατηρούνται στη βάση για παραπάνω από 1 χρόνο.



Εικόνα 31 Ιστορικά δεδομένα

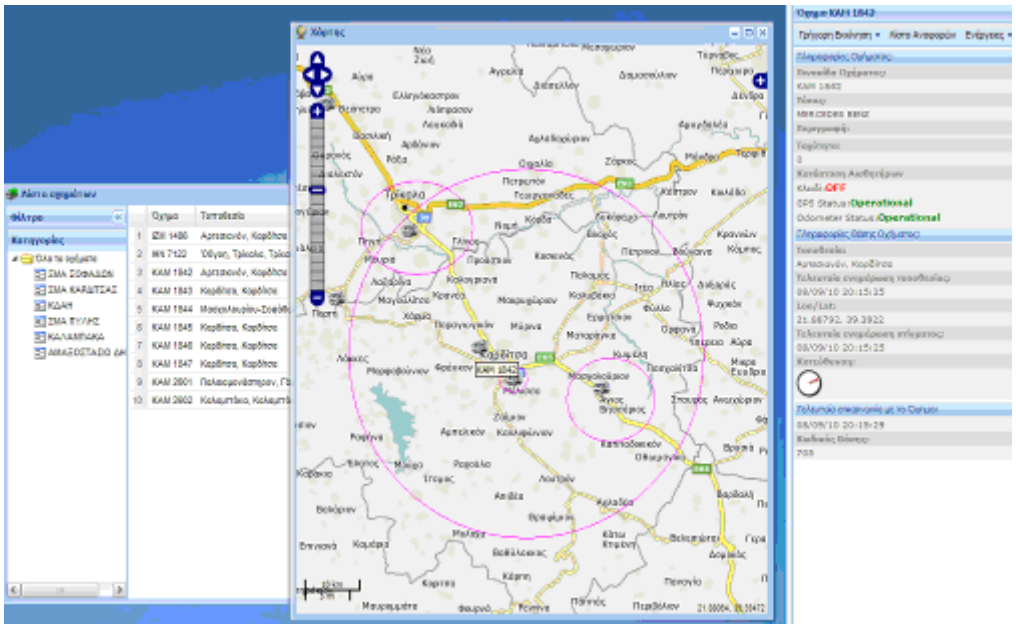
8.3.10 GEOFENCING

Κατά την εγκατάσταση της συσκευής εντοπισμού υπάρχει η δυνατότητα να προγραμματιστεί, ούτως ώστε να έχει περιορισμό στο χώρο κίνησης του εκάστοτε οχήματος και εάν ξεφύγει από τα προκαθορισμένα όρια να ενεργοποιείται η δορυφορική μονάδα και να στέλνει alarm στον κεντρικό υπολογιστή και στον αντίστοιχο χρήστη.

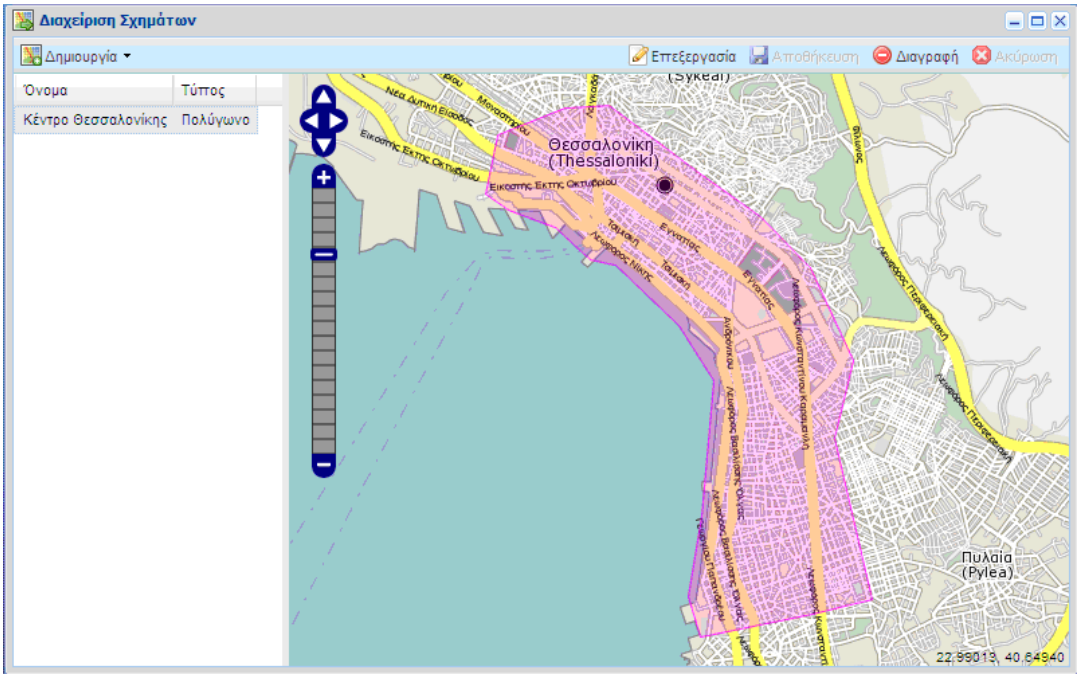


Εικόνα 32 Διαχείρηση Geofences

Ταυτόχρονα ανοίγει νέο παράθυρο στην εφαρμογή όπου ο χρήστης αντιλαμβάνεται από ποια όρια κίνησης έχει βγει το όχημα και θέτει τον μηχανισμό ασφαλείας σε λειτουργία. Οι περιορισμοί αυτοί μπορεί να είναι τα όρια μιας πόλης ή τα όρια δήμων ή οικοδομικών τετραγώνων. Αυτό μπορεί να συνδεθεί και με σειρήνα ασφαλείας που θα βρίσκεται στις εγκαταστάσεις του χρήστη.



Εικόνα 33 Geofences



Εικόνα 34 Geofences

8.3.11 ΑΝΑΦΟΡΕΣ – ΕΚΤΥΠΩΣΕΙΣ – ΣΤΑΤΙΣΤΙΚΑ

Το σύστημα έχει τη δυνατότητα αυτόματης δημιουργίας αναφορών για την δραστηριότητα ενός οχήματος ή μιας ομάδας οχημάτων, στον ίδιο στόλο ή στην ίδια κατηγορία για το χρονικό διάστημα που θα επιλέξει (επιλογή ημέρας και ώρας), εμφανίζοντας πληροφορίες για την ακριβή οδό, χρόνο εκκίνησης και στάσεων, ταχύτητα κίνησης, απόσταση που διένυσε και καταστάσεις συναγερμών.

Οι αναφορές παρουσιάζονται είτε στον χάρτη ή σε μορφή πίνακα. Επίσης, υπάρχει η δυνατότητα άμεσης εξαγωγής στοιχείων από τη Βάση Δεδομένων και η επιλογή του επιθυμητού χρονικού διαστήματος (Υπάρχει δυνατότητα δημιουργίας αναφορών επιλέγοντας χρονική περίοδο και όχημα αναφοράς).

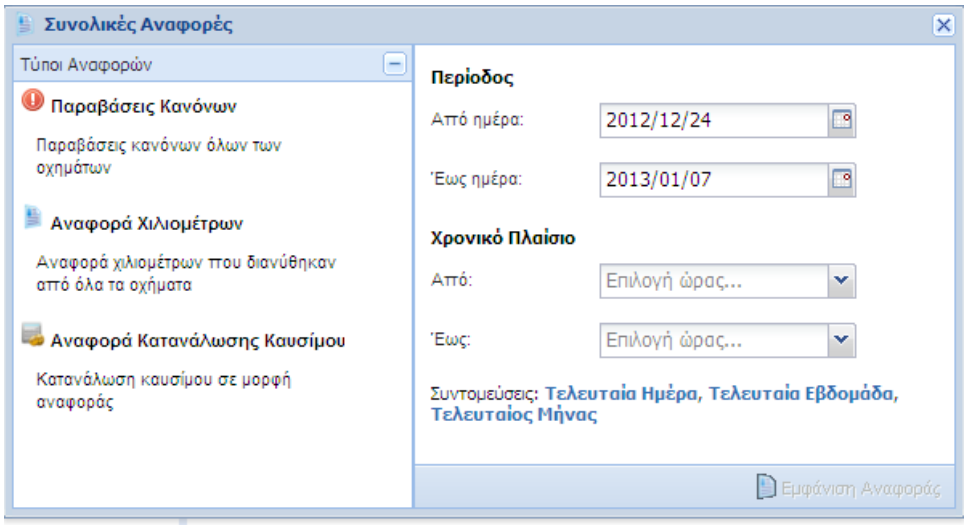
Οι παρεχόμενες αναφορές συστήματος αφορούν:
■ Συγκριτικές αναφορές ανά ομάδα οχημάτων
■ Αναλυτικές αναφορές ανά όχημα
■ Μεγάλος αριθμός προκαθορισμένων αναφορών
■ Δυνατότητα δημιουργίας εξειδικευμένων αναφορών (ανά διαδρομή, ανά όχημα, ανά δρομολόγιο, ανά ημέρα, ανά οποιοδήποτε χρονικό διάστημα).■ Αποστολές / κινήσεις ανά όχημα, ανά χρονική περίοδο. Επίσης θα εμφανίζονται πληροφορίες όπως η ώρα έναρξης, ο τόπος και η διάρκεια.
■ Χιλιόμετρα που έχουν διανυθεί ανά όχημα, ανά ομάδα οχημάτων, ανά χρονική περίοδο.
■ Αναζήτηση και αναπαράσταση δρομολογίου οχημάτων επί του ψηφιακού χάρτη που θα περιλαμβάνει στοιχεία όπως:

- απεικόνιση των σημείων συλλογής κάδων (στάσεις)
- ακριβή χρόνο που έγινε η στάση
- Βάρος κάδου
- Μεταβολή βάρους οχήματος
- επιπλέον στοιχεία που αφορούν στη συγκεκριμένη στάση που θα είναι εύκολα προσβάσιμα στον χρήστη του συστήματος (ταχύτητα οχήματος, κατάσταση οχήματος, ακριβή ώρα, κλπ).

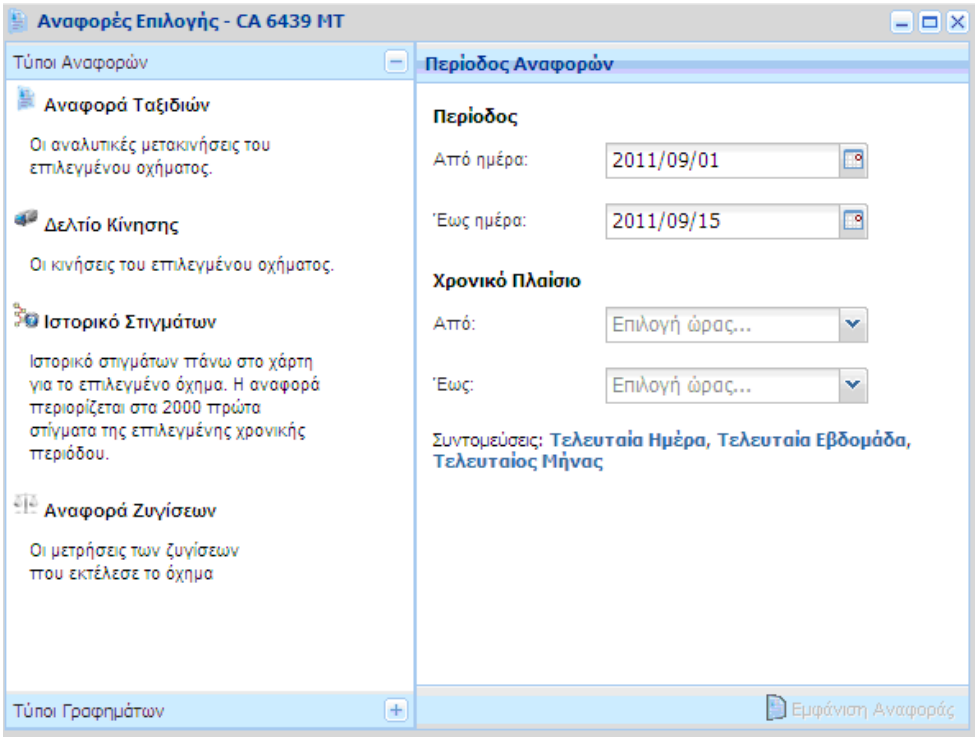
Από το server θα πρέπει να υπάρχει η δυνατότητα για αυτόματη αποστολή SMS και επιπλέον των προδιαγραφών με e-mail σε περιπτώσεις που:

- ✓ το όχημα βρίσκεται σε παρατεταμένη στάση
- ✓ το όχημα υπερβαίνει το επιτρεπτό όριο ταχύτητας
- ✓ το όριο αποκλίνει των γεωγραφικών ορίων ευθύνης του
- ✓ Το όχημα δεν βρίσκετε στο σωστό δρομολόγιο
- ✓ Ανύψωση κάδου μεγαλύτερου από το επιτρεπτό βάρος. Κάθε αναφορά και οι πληροφορίες που εμφανίζονται στον χάρτη μπορεί να εκτυπωθεί, αποθηκευτεί ή να αντιγραφεί.

Πιο αναλυτικά, η εφαρμογή παρέχει δυνατότητα εξαγωγής και εισαγωγής δεδομένων σε μορφή αρχείων τύπου ASCII, Excel, κλπ.. Δείγματα αναφορών που υπάρχουν στο σύστημα Διαχείρισης Στόλου, είναι τα ακόλουθα:



Εικόνα 35 Αναφορές



Εικόνα 36 Αναφορές

8.3.11.1 ΣΥΝΟΛΙΚΗ ΑΝΑΦΟΡΑ ΚΙΝΗΣΗΣ

Η αναφορά αυτή χρησιμοποιείται για την ανάλυση των χρόνων κίνησης όλων των οχημάτων ενός στόλου για την χρονική περίοδο της επιλογής του χρήστη. Η πληροφορία που υπάρχει σε αυτήν την αναφορά απεικονίζει το χρονικό διάστημα που έχουν κινηθεί τα οχήματα, τον αριθμό των στάσεων που έχουν γίνει, το σύνολο των ωρών κίνησης καθώς και το σύνολο ωρών στάσεων. Στη συνέχεια από τα στοιχεία αυτά υπολογίζεται ο ημερήσιος μέσος όρος κίνησης.
Ταυτόχρονα τα στοιχεία αυτά μπορούν να απεικονιστούν και γραφικά σε pie & bar chart επιτρέποντας την εύκολη συγκριτική μελέτη.

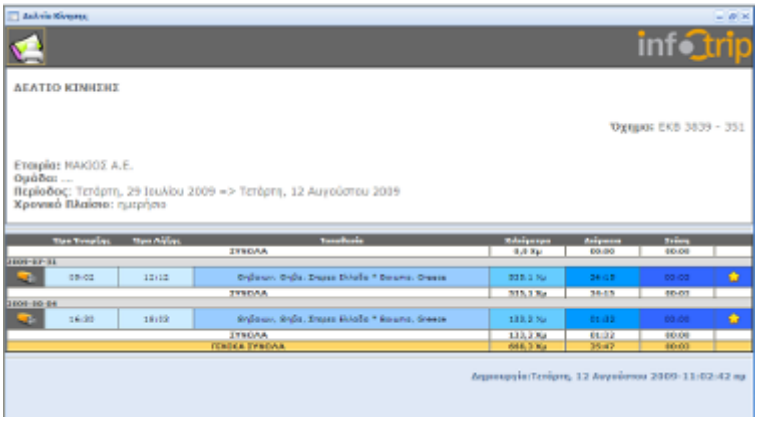
Εικόνα 37 Συνολικές Αναφορές

8.3.11.2 ΣΥΓΚΡΙΤΙΚΟΣ ΠΙΝΑΚΑΣ ΚΙΝΗΣΗΣ ΟΧΗΜΑΤΩΝ

Η αναφορά αυτή δίνει γραφική αναπαράσταση της χρήσης των οχημάτων ενός στόλου κατά τη διάρκεια μιας ημέρας ή και περισσοτέρων (δυνατότητα επιλογής χρονικού διαστήματος) και ένα ποσοστό παραγωγής των οχημάτων αυτών.

8.3.11.3 ΔΕΛΤΙΟ ΚΙΝΗΣΗΣ

Η αναφορά αυτή χρησιμοποιείται για την ανάλυση των δρομολογίων ενός οχήματος και απεικονίζει πληροφορίες από την αφετηρία μέχρι την κάθε στάση: ακριβή ώρα έναρξης του δρομολογίου, ακριβή ώρα στάσης, γεωγραφικό σημείο της στάσης, τη διανυθείσα απόσταση έως την στάση (σε χιλιόμετρα ή μίλια), τον χρόνο που ταξίδεψε το όχημα έως την στάση, καθώς και τον χρόνο παραμονής στην στάση. Στο τέλος της αναφοράς υπάρχουν τα σύνολα για όλα τα στοιχεία για το χρονικό διάστημα που έχει επιλέξει ο χρήστης.



Εικόνα 38 Δελτίο Κίνησης

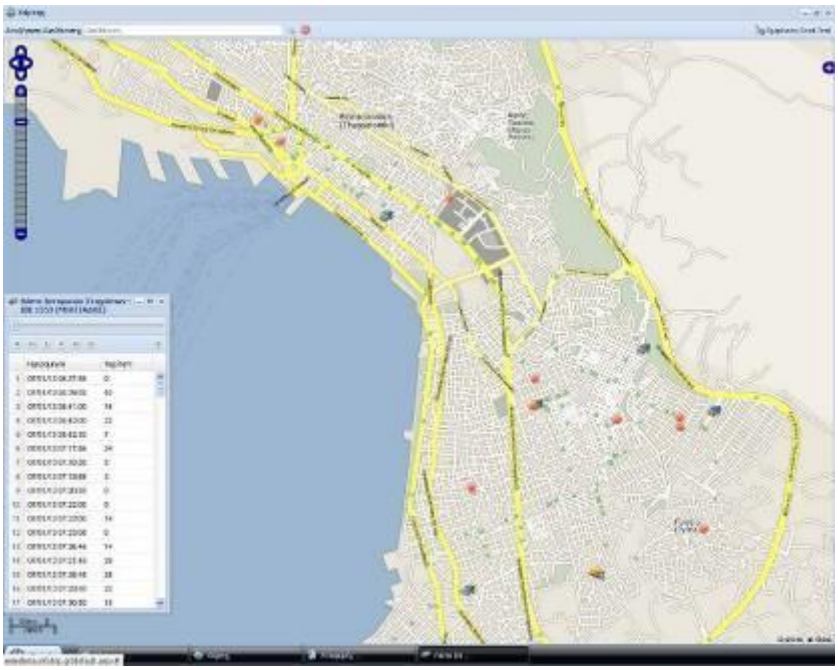
8.3.11.4 ΔΕΛΤΙΟ ΑΝΑΦΟΡΑΣ ΤΑΞΙΔΙΟΥ

Η αναφορά αυτή απεικονίζει πληροφορίες για το χρονικό διάστημα που έχει παραμείνει το όχημα σε μια στάση ή σ' ένα σημείο συλλογής κάδων, το χρόνο που χρειάστηκε για να φτάσει εκεί, την απόσταση που κάλυψε έως εκεί, τον ακριβή χρόνο που σταμάτησε στο συγκεκριμένο σημείο και τον ακριβή χρόνο που έφυγε από το σημείο.

Κατάσταση	Ώρα Έναρξης	Ώρα Λήξης	Διάρκεια	Χιλιόμετρα	Τοποθεσία
2011-09-14					
	00:00	08:37	08:37	0 Χμ	Μενεμένη, Θεσσαλονίκη
		08:57	00:20	18.7 Χμ	
		09:43	00:45	0 Χμ	Νέα Μεσημβρία, Άγιος Αθανάσιος
	09:43	09:55	00:13	7.7 Χμ	
	09:56	10:57	01:01	0 Χμ	Νέα Μαγνησία, Εχέδωρος
	10:57	11:10	00:13	12.6 Χμ	
	11:11	12:30	01:19	0 Χμ	Μενεμένη, Θεσσαλονίκη
	12:30	12:38	00:09	1.2 Χμ	

Εικόνα 39 Δελτίο Στάσεων

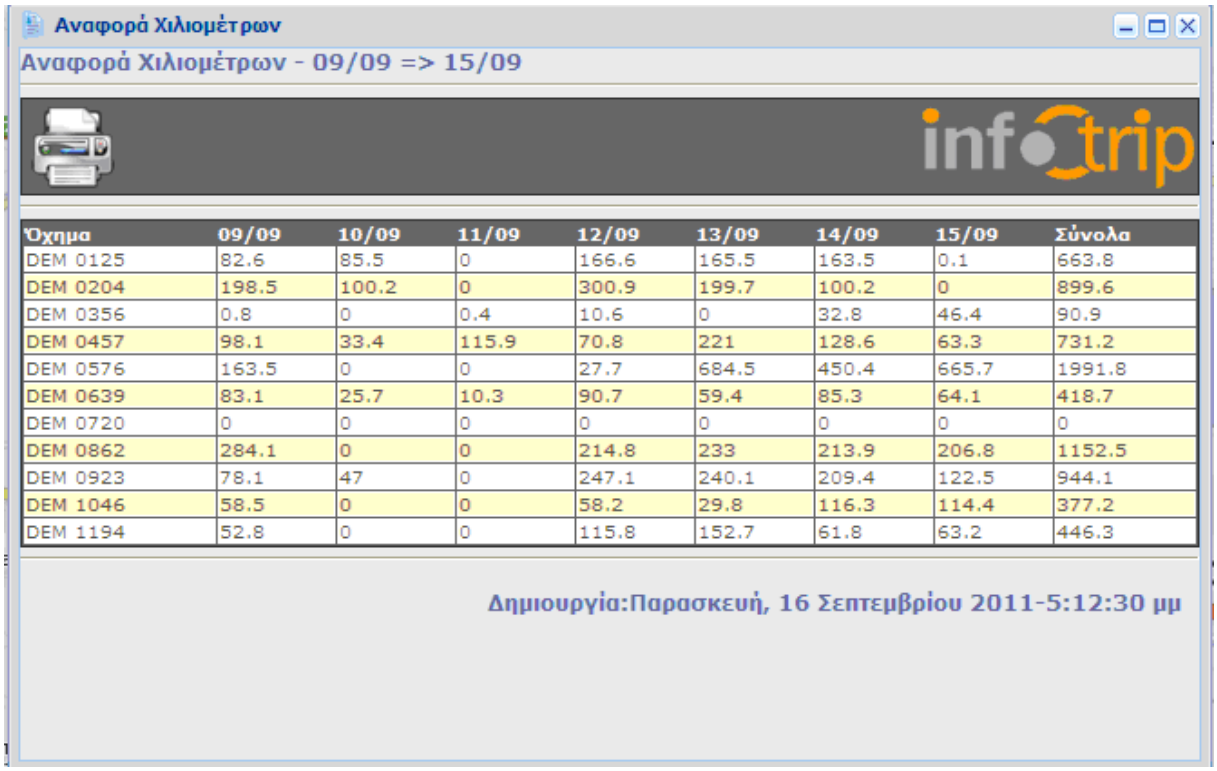
Επιπλέον, παρέχεται η δυνατότητα απεικόνισης των στάσεων πάνω στο χάρτης. Η διάρκεια που κάποιο όχημα παραμένει ακίνητο είναι παραμετροποιήσιμο.



Εικόνα 40 Απεικόνιση Στάσεων

8.3.11.5 ΔΕΛΤΙΟ ΧΙΛΙΟΜΕΤΡΩΝ

Η αναφορά χιλιομέτρων εμφανίζει όλα τα οχήματα που είναι καταχωρημένα στην εφαρμογή με τα αντίστοιχα χιλιόμετρα που έχει διανύσει το καθένα για κάθε μέρα της χρονικής περιόδου που έχει επιλέξει ο χρήστης.



Όχημα	09/09	10/09	11/09	12/09	13/09	14/09	15/09	Σύνολα
DEM 0125	82.6	85.5	0	166.6	165.5	163.5	0.1	663.8
DEM 0204	198.5	100.2	0	300.9	199.7	100.2	0	899.6
DEM 0356	0.8	0	0.4	10.6	0	32.8	46.4	90.9
DEM 0457	98.1	33.4	115.9	70.8	221	128.6	63.3	731.2
DEM 0576	163.5	0	0	27.7	684.5	450.4	665.7	1991.8
DEM 0639	83.1	25.7	10.3	90.7	59.4	85.3	64.1	418.7
DEM 0720	0	0	0	0	0	0	0	0
DEM 0862	284.1	0	0	214.8	233	213.9	206.8	1152.5
DEM 0923	78.1	47	0	247.1	240.1	209.4	122.5	944.1
DEM 1046	58.5	0	0	58.2	29.8	116.3	114.4	377.2
DEM 1194	52.8	0	0	115.8	152.7	61.8	63.2	446.3

Εικόνα 41 Αναφορά Χιλιομέτρων

8.3.11.6 ΔΕΛΤΙΟ ΣΤΑΣΕΩΝ

Η αναφορά αυτή θα απεικονίζει πληροφορίες για το χρονικό διάστημα που έχει μείνει το όχημα σε μια στάση, το χρόνο που χρειάστηκε για να φτάσει εκεί, την απόσταση που κάλυψε έως εκεί, τον ακριβή χρόνο που σταμάτησε εκεί και τον ακριβή χρόνο που έφυγε από εκεί.

8.3.11.7 ΔΕΛΤΙΟ ΚΑΔΩΝ ΑΝΑ ΟΧΗΜΑ

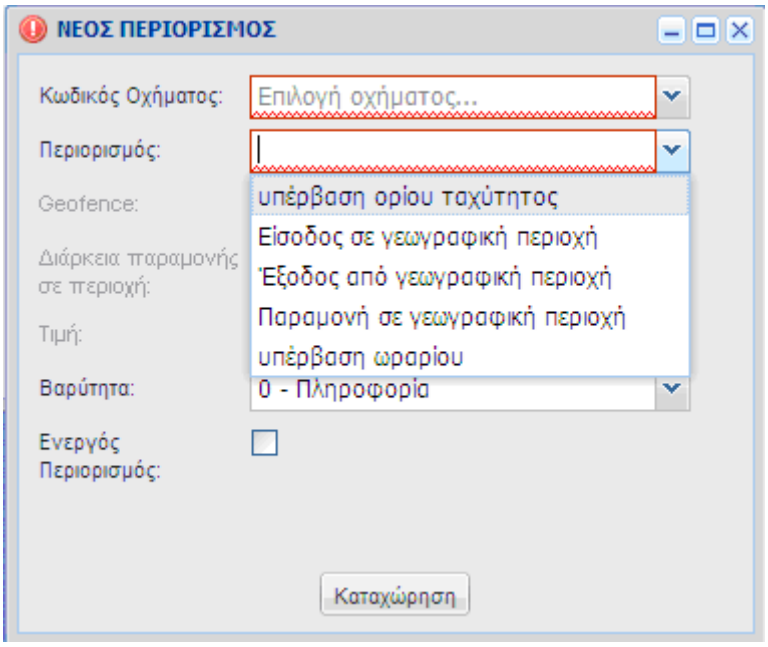
Η αναφορά αυτή θα απεικονίζει πληροφορίες για τους κάδους που συλλέχθηκαν από το όχημα, το χρόνο συλλογής, τις θέσεις των κάδων και το βάρος των σκουπιδιών.

8.3.11.8 ΔΕΛΤΙΟ ΚΑΔΩΝ ΑΝΑ ΟΙΚΙΣΜΟ

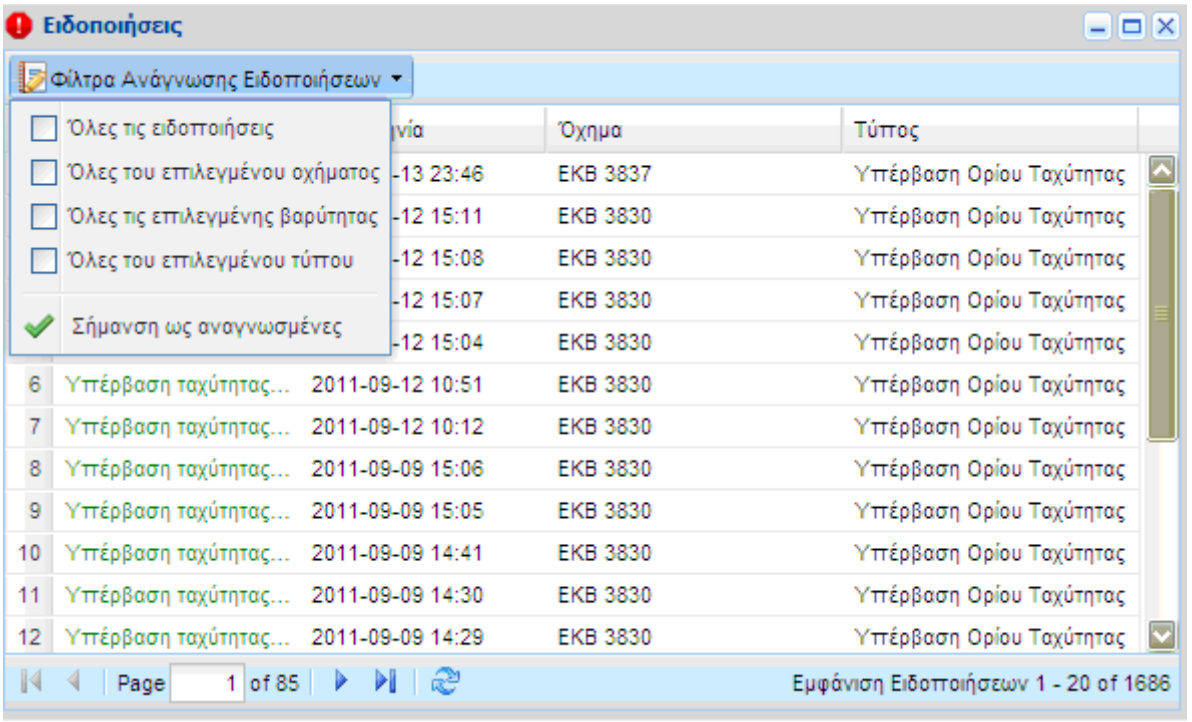
Η αναφορά αυτή θα απεικονίζει πληροφορίες για τους κάδους που συλλέχθηκαν από το συγκεκριμένο οικισμό, το χρόνο συλλογής, τις θέσεις των κάδων και το βάρος των σκουπιδιών, άσχετα αν η εργασία έγινε με ένα ή περισσότερα οχήματα.

8.3.11.9 ΑΝΑΦΟΡΑ ΓΙΑ ΠΑΡΑΒΙΑΣΕΙΣ ΚΑΝΟΝΩΝ ΟΧΗΜΑΤΩΝ

Ο διαχειριστής του συστήματος έχει τη δυνατότητα να εισάγει στο σύστημα διάφορους περιορισμούς για τη λειτουργία του στόλου των οχημάτων, όπως π.χ. υπέρβαση ορίου ταχύτητας και να λαμβάνει ενημέρωση για τυχόν παραβιάσεις.



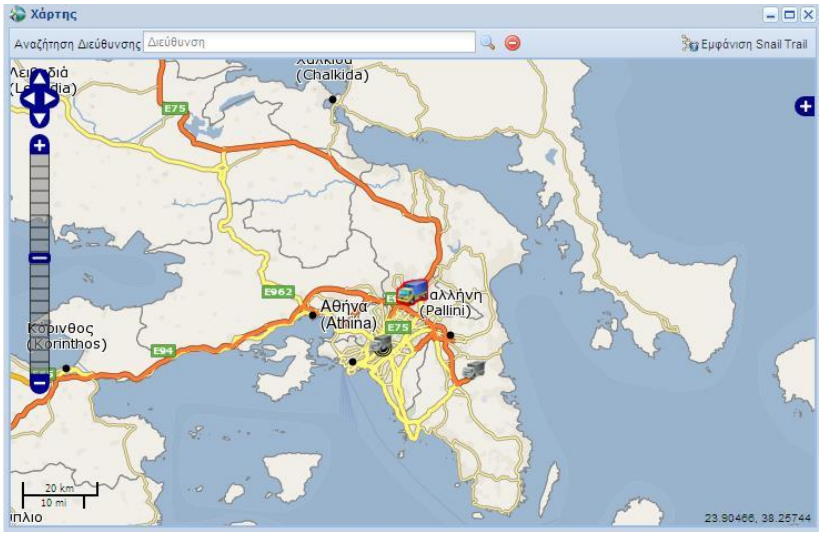
Εικόνα 42 Εισαγωγή περιορισμού Οχήματος



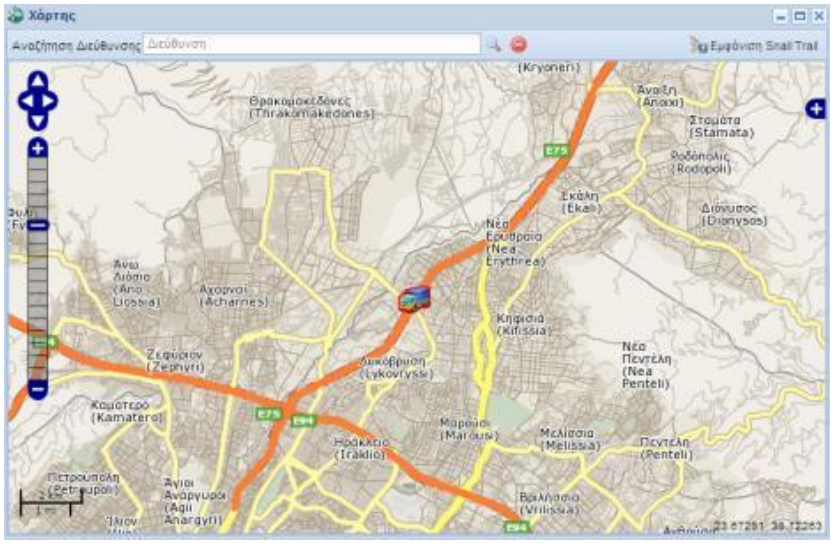
Εικόνα 43 Αναφορά παραβίασης κανόνων

8.3.12 ΧΑΡΤΗΣ ΕΦΑΡΜΟΓΗΣ

Στους ψηφιακούς χάρτες απεικονίζονται με ακρίβεια (μικρότερη των 5 μέτρων) τα οχήματα του στόλου σε πραγματικό χρόνο καθώς και τα σημεία ενδιαφέροντος. Η κλίμακα του χάρτη μπορεί να αλλάζει βάσει τις εκάστοτε επιλογές του χρήστη.
Ο χρήστης του συστήματος μπορεί να βλέπει στον χάρτη ταυτόχρονα όλα τα οχήματα, καθώς επίσης και να επιλέγει το κατάλληλο εικονίδιο, το οποίο απεικονίζει τη θέση του εκάστοτε οχήματος στο ψηφιακό χάρτη. Το εικονίδιο είναι δυναμικό και μπορεί να αλλάζει χρώμα ανάλογα με την κατάσταση του οχήματος. Επιπλέον, υπάρχει δυνατότητα για άνοιγμα πολλαπλών παραθύρων παρακολούθησης πολλών οχημάτων στο χάρτη ταυτόχρονα και ανεξάρτητο zoom in/zoom out σε κάθε παράθυρο.



Εικόνα 44 Χάρτης Εφαρμογής



Εικόνα 45 Χάρτης Εφαρμογής

Οι ψηφιακοί χάρτες της εφαρμογής παρέχουν:
✓ Πληροφορίες για βασικούς δρόμους με γραφική απεικόνιση (αστικό επίπεδο, προάστιο, πόλη ή επαρχία)
✓ Εντοπισμός και εστίαση των οχημάτων

✓ Δυνατότητα παρουσίασης σημείων ενδιαφέροντος (κτίρια, αξιοθέατα, υπαίθριοι σταθμοί αυτοκινήτων κτλ.)
✓ Δυνατότητα εισαγωγής ψηφιακού χάρτη (raster ή vector format) της περιοχής κάλυψης.
✓ Δυνατότητα εκτύπωσης του χάρτη και των πληροφοριών που εμφανίζονται πάνω σε αυτόν.
✓ Κάθε προβολή χάρτη μπορεί να αποθηκευτεί ή να αντιγραφεί

9. ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΙ ΣΤΟΧΟΙ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

Τα πλεονεκτήματα του συστήματος είναι πολλαπλά, αναφέρουμε ενδεικτικά τα παρακάτω:

➢**Άμεση ενημέρωση οδηγού της κατάστασης του φορτίου**
- Ως αποτέλεσμα να μην υπερφορτώνεται και καταπονείται το όχημα
- Γνώση και καταγραφή του συνολικού απόβαρου κατά την απόθεση των απορριμμάτων στο κέντρο συλλογής

➢**Μείωση του κόστους**
- Βελτίωση ελέγχου κατανάλωσης καυσίμου
- Μείωση του κόστους επικοινωνιών
- Καλύτερος έλεγχος του κόστους λειτουργίας του στόλου

➢**Αύξηση παραγωγικότητας**
- Βελτίωση της παραγωγικότητας με βάση το ωράριο εργασίας
- Έλεγχος της ταχύτητας των οχημάτων, με αποτέλεσμα την μείωση των προστίμων αλλά και της φθοράς αυτών
- Αποτελεσματικότερος αριθμός διεκπεραιωμένων εργασιών με υψηλότερη αποτελεσματικότητα

➢**Βελτίωση του επιχειρηματικού σχεδιασμού και πρόβλεψης**
- Βελτίωση σχεδιασμού δρομολογίων
- Ιστορικά δεδομένα και πρόβλεψη στον ανασχεδιασμό δρομολογίων

➢**Αυξημένη ασφάλεια**
- Βελτίωση οδικής συμπεριφοράς: ταχύτητα, άσκοπη οδήγηση
- Άμεσος εντοπισμός σε περίπτωση κινδύνου
- Ευελιξία σε περίπτωση κλοπών

➢**Φιλική και εύχρηστη προς τον χρήστη εφαρμογή**
Το περιβάλλον εργασίας του χρήστη είναι πλήρως γραφικό (GUI) χρησιμοποιώντας όλα τα γνωστά χαρακτηριστικά (ποντίκι, παράθυρα, μενού λειτουργιών, κουμπιά λειτουργιών, λίστες επιλογής κλπ). Η διεπαφή χρήστη έχει ενιαία σχεδιαστική φιλοσοφία ώστε να μην μπερδεύεται ο χρήστης.

Αυτό αφορά τόσο τη χρήση κοινής χρωματικής παλέτας όσο και τη χρήση κοινών συμβολισμών για ομοειδείς και παρόμοιες λειτουργίες.

Οι διάφορες λειτουργίες της εφαρμογής είναι απλά σχεδιασμένες ώστε να είναι λογική η αλληλουχία των βημάτων, να ελαχιστοποιηθούν τα βήματα που απαιτούνται για την ολοκλήρωση μια ενέργειας, υπάρχει σαφής ένδειξη σε πιο βήμα μιας λειτουργίας βρίσκεται ο χρήστης και πως μπορεί να προχωρήσει στο επόμενο ή προηγούμενο βήμα, σαφής ένδειξη σε ποια σελίδα βρίσκεται ο χρήστης και ποια ήταν η διαδρομή που ακολούθησε για να φτάσει καθώς και σε ποιες σελίδες ανώτερου ή κατώτερου επιπέδου μπορεί να μετακινηθεί.

10. ΕΥΧΡΗΣΤΙΑ – ΧΡΗΣΤΙΚΟΤΗΤΑ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

Με τον όρο χρηστικότητα χαρακτηρίζουμε τον βαθμό ευχρηστίας μιας εφαρμογής, δηλαδή το πόσο εύκολο είναι η χρήση της για τον χρήστη. Επηρεάζεται από διάφορους παράγοντες όπως το πόσο εύκολη και διαισθητική είναι η πλοήγηση, το πόσο κατανοητά είναι τα κείμενα, την ευκολία με την οποία θα βρει ο χρήστης τις πληροφορίες που αναζητεί, την ταχύτητα με την οποίαν «φορτώνει» η εφαρμογή, κτλ.

Πώς θα αξιολογήσουμε τη χρηστικότητα ενός συστήματος; Μέσα από επιμέρους στόχους όπως:
✓Ευκολία εκμάθησης
✓Ταχύτητα εκτέλεσης των επί μέρους εργασιών
✓Μικρή συχνότητα λαθών των χρηστών
✓Διατήρηση της ικανότητας χρήσης με το χρόνο

Γενικότερα ένα σύστημα όσο μεγάλο και πολύπλοκο να είναι, θεωρείται χρηστικό, όταν και ο πιο άπειρος χρήστης μπορεί να το χρησιμοποιήσει χωρίς ιδιαίτερη δυσκολία ή εκπαίδευση.
Σημαντικό ρόλο παίζουν το σωστά δομημένο και ξεκάθαρο navigation, η οργάνωση του περιεχομένου (information architecture), η συνέχεια (consistency) του layout (και το user interface γενικότερα), τα μηνύματα λάθους και feedback, οι σωστά δομημένες και καθαρές φόρμες, στοιχεία που αποτελούν τον βασικό κορμό της χρηστικότητας ενός σύνθετου συστήματος.

10.1 ΦΙΛΙΚΟΤΗΤΑ ΔΙΕΠΑΦΗΣ

Ένα ελκυστικό look & feel αποτελεί βασική προϋπόθεση για την επίτευξη συχνών και επαναλαμβανόμενων επισκέψεων (σε συνδυασμό πάντα με το έγκυρο, ενημερωμένο και επίκαιρο περιεχόμενο), ειδικά σε περιπτώσεις όπου η εφαρμογή στοχεύει σε διάφορες ομάδες χρηστών. Στις εφαρμογές όπου προσφέρονται μεγάλοι όγκοι και διάφοροι τύποι δεδομένων, η διεπαφή θα πρέπει να εμπλουτίζεται με χαρακτηριστικά που αφενός μεν βελτιστοποιούν τον χρόνο απόκρισης και αφετέρου αυξάνουν τον ενθουσιασμό, ενθαρρύνουν τις συχνές επισκέψεις και παρέχουν ταχύτατη και φιλική πρόσβαση στο περιεχόμενο.
Ένα ελκυστικό look & feel θα συμβάλλει επίσης τόσο στη γρήγορη αποδοχή του συστήματος από τους χρήστες του, όσο και στη ταχύτατη εκμάθηση / αφομοίωσή του. Στα συστήματα τηλεματικής που λειτουργούν και εκτός των φυσικών ορίων ενός οργανισμού (απομακρυσμένη πρόσβαση μέσω Internet) και εξυπηρετούν μεγάλους όγκους δεδομένων να μην περιπλέκει τους χρήστες και να παρέχει ταχύτατη, φιλική & δομημένη πρόσβαση στο περιεχόμενο και στις λειτουργίες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. V.S. Frost, B. Melamed: Traffic Models For Telecommunications Networks; IEEE Communications Magazine, March 1994, pp. 70 - 81.
2. J. Cain, D.J. Goodman: General Packet Radio Service in GSM; IEEE Communication Magazine, October 1997, pp. 122 - 131.
3. Matthew A. Turk and Alex P. Pentland. Face recognition using Eigenfaces. Vision and Modeling Group , the Media Laboratory, Massachusetts Institute of Technology
4. Y. Zhao and Z. Ye, "A Low Cost GSM/GPRS Based Wireless Home Security System", IEEE Transactions on Consumer Electronics, vol. 54, no. 2, (2008).
5. http://ww1.microchip.com/downloads/en/devicedoc/40001856a.pdf
6. http://ww1.microchip.com/downloads/en/DeviceDoc/PIC18LF27_47K40-Data-Sheet-40001844E.pdf
7. https://www.mikroe.com/rs232-2-click
8.https://el.wikipedia.org/wiki/%CE%9C%CE%B9%CE%BA%CF%81%CE%BF%CE%B5%CE%BB%CE%B5%CE%B3%CE%BA%CF%84%CE%AE%CF%82
9.www.infotrip.gr

ΠΑΡΑΡΤΗΜΑ Α – Κώδικας

main.c

```
#include "mcc.h"
#include "serial.h"

void Lcd_initialize (void);

void main(void)
{
   // Initialize the device
   SYSTEM_Initialize();

   // Enable the Global Interrupts
   INTERRUPT_GlobalInterruptEnable();

   // Enable the Peripheral Interrupts
   INTERRUPT_PeripheralInterruptEnable();

   Lcd_initialize();
   Serial_Initialize();

   while (1)
   {
      ledstatus_SetHigh();
      __delay_ms(100);
      Serial1_input();
      __delay_ms(500);
      Serial2_input();
      MsgG_check();
      Msg_clear();
      Check_unload();
      ledstatus_SetLow();
      __delay_ms(5);
   }
}
```

```
/**
 End of File
*/
```

lcd.c

```
#include "mcc.h"
#include "lcd.h"

#define LcdData          PORTD
#define LcdDataDir       TRISD
#define LcdRS            PORTEbits.RE0
#define LcdRW            PORTEbits.RE1
#define LcdEN            PORTEbits.RE2
#define LcdRSDir         TRISEbits.TRISE0
#define LcdRWDir         TRISEbits.TRISE1
#define LcdENDir         TRISEbits.TRISE2
#define IN               1
#define OUT              0

//start Msg LCD  "12345678901234567890"
char Msg1[]    ="  K A O U S S I S   ";
char Msg2[]    ="    Powered by     ";
char Msg3[]    =" Swarco Hellas S.A. ";

/* Function to initialization display */
void Lcd_initialize (void)
{
   //set direction to out on all related pins
   LcdDataDir = 0x00;
   LcdRSDir = OUT;
   LcdRWDir = OUT;
   LcdENDir = OUT;

   //lcd initialization sample script from lcd datasheet
   LcdEN = 0;
   __delay_ms(50);   //Wait >40 msec after power is applied
   Lcd_com(0x30);    //command 0x30 = Wake up
   __delay_ms(10);   //must wait 5ms, busy flag not available
   Lcd_com(0x30);    //command 0x30 = Wake up #2
```

```
    __delay_us(200);  //must wait 160us, busy flag not available
    Lcd_com(0x30);    //command 0x30 = Wake up #3
    __delay_us(200);  //must wait 160us, busy flag not available
    Lcd_com(0x38);    //Function set: 8-bit/2-line
    __delay_us(50);   //function execution time 37us
    Lcd_com(0x10);    //Set cursor
    __delay_us(50);   //function execution time 37us
    Lcd_com(0x0C);    //Display ON; Cursor OFF; Blinking Cursor OFF;
    __delay_us(50);   //function execution time 37us
    Lcd_com(0x06);    //Entry mode set
    __delay_us(50);   //function execution time 37us
    Lcd_clear();      //clear display
    Lcd_line(1);
    Lcd_str(Msg1);
    Lcd_line(2);
    Lcd_str(Msg2);
    Lcd_line(3);
    Lcd_str(Msg3);
    Lcd_custom();       //Custom characters
}

/* Function to clear display */
void Lcd_clear (void)
{
    Lcd_com(0x01); //clear display command
    __delay_ms(3); //function execution time 1.52ms
}

/* Function to send the command to LCD */
void Lcd_com (unsigned char i)
{
    //sample script from lcd datasheet
    LcdData = i;
    //set control pins to write command
    LcdRS = 0;
    LcdRW = 0;
    //trigger lcd
```

```
   LcdEN = 1;
   __delay_us(400);
   LcdEN = 0;
}


/* Function to send the data to LCD */
void Lcd_dat (unsigned char i)
{
   //sample script from lcd datasheet
   LcdData = i;
   //set control pins to write command
   LcdRS = 1;
   LcdRW = 0;
   //trigger lcd
   LcdEN = 1;
   __delay_us(400);
   LcdEN = 0;
}

/* Function to set cursor at start of line x of LCD */
void Lcd_line (unsigned char i)
{
   switch (i)
   {
        case 1:   Lcd_com(0x80);     break; //command to set cursor on DDRAM address 0x00 (line 1)
        case 2:   Lcd_com(0xC0);     break; //command to set cursor on DDRAM address 0x40 (line 2)
        case 3:   Lcd_com(0x94);     break; //command to set cursor on DDRAM address 0x14 (line 3)
        case 4:   Lcd_com(0xD4);     break; //command to set cursor on DDRAM address 0x54 (line 4)
   }
}

/* Function to send string to LCD */
void Lcd_str (const unsigned char *x)
{
   while(*x)
   {
```

```
      Lcd_dat(*x++);
   }
}

/* Function to save custom characters to LCD */
void Lcd_custom (void)
{
   int i;
   // Save custom characters
   Lcd_com(0x40);            //Set cursor to the beginning of CGRAM address

   //#0:"gr D"
   Lcd_dat(0b00100);
   Lcd_dat(0b01010);
   Lcd_dat(0b01010);
   Lcd_dat(0b01010);
   Lcd_dat(0b10001);
   Lcd_dat(0b10001);
   Lcd_dat(0b11111);
   Lcd_dat(0b00000);

   //#1:"gr L"
   Lcd_dat(0b00100);
   Lcd_dat(0b01010);
   Lcd_dat(0b01010);
   Lcd_dat(0b01010);
   Lcd_dat(0b10001);
   Lcd_dat(0b10001);
   Lcd_dat(0b10001);
   Lcd_dat(0b00000);

   //#2:"gr F"
   Lcd_dat(0b00100);
   Lcd_dat(0b11111);
   Lcd_dat(0b10101);
   Lcd_dat(0b10101);
   Lcd_dat(0b11111);
```

```
   Lcd_dat(0b00100);
   Lcd_dat(0b00100);
   Lcd_dat(0b00000);
}

void Lcd_barosk (void)
{
   //gr BAROS
   Lcd_dat(0b01000010);
   Lcd_dat(0b01000001);
   Lcd_dat(0b01010000);
   Lcd_dat(0b01001111);
   Lcd_dat(0b11110110);
   //"_"
   Lcd_dat(0b00100000);
   //gr KADOY
   Lcd_dat(0b01001011);
   Lcd_dat(0b01000001);
   Lcd_dat(0);
   Lcd_dat(0b01001111);
   Lcd_dat(0b01011001);
   //":_"
   Lcd_dat(0b00111010);
   Lcd_dat(0b00100000);
}

void Lcd_barossum1 (void)
{
   //gr SYNOLIKO
   Lcd_dat(0b11110110);
   Lcd_dat(0b01011001);
   Lcd_dat(0b01001110);
   Lcd_dat(0b01001111);
   Lcd_dat(1);
   Lcd_dat(0b01001001);
   Lcd_dat(0b01001011);
   Lcd_dat(0b01001111);
```

```
    //"_"
    Lcd_dat(0b00100000);
    //gr BAROS
    Lcd_dat(0b01000010);
    Lcd_dat(0b01000001);
    Lcd_dat(0b01010000);
    Lcd_dat(0b01001111);
    Lcd_dat(0b11110110);
}

void Lcd_barossum2 (void)
{
    //FORTIOY
    Lcd_dat(2);
    Lcd_dat(0b01001111);
    Lcd_dat(0b01010000);
    Lcd_dat(0b01010100);
    Lcd_dat(0b01001001);
    Lcd_dat(0b01001111);
    Lcd_dat(0b01011001);
    //":_"
    Lcd_dat(0b00111010);
    Lcd_dat(0b00100000);
}

void Lcd_clearsum1 (void)
{
    //"_"
    Lcd_dat(0b00100000);
    Lcd_dat(0b00100000);
    Lcd_dat(0b00100000);
    Lcd_dat(0b00100000);
    Lcd_dat(0b00100000);
    //gr MHDENISMOS
    Lcd_dat(0b01001101);
    Lcd_dat(0b01001000);
    Lcd_dat(0);
```

```
   Lcd_dat(0b01000101);
   Lcd_dat(0b01001110);
   Lcd_dat(0b01001001);
   Lcd_dat(0b11110110);
   Lcd_dat(0b01001101);
   Lcd_dat(0b01001111);
   Lcd_dat(0b11110110);
}

void Lcd_clearsum2 (void)
{
   //"_"
   Lcd_dat(0b00100000);
   Lcd_dat(0b00100000);
   //gr SYNOLIKOY
   Lcd_dat(0b11110110);
   Lcd_dat(0b01011001);
   Lcd_dat(0b01001110);
   Lcd_dat(0b01001111);
   Lcd_dat(1);
   Lcd_dat(0b01001001);
   Lcd_dat(0b01001011);
   Lcd_dat(0b01001111);
   Lcd_dat(0b01011001);
   //"_"
   Lcd_dat(0b00100000);
   //gr BAROS
   Lcd_dat(0b01000010);
   Lcd_dat(0b01000001);
   Lcd_dat(0b01010000);
   Lcd_dat(0b01001111);
   Lcd_dat(0b01011001);
   Lcd_dat(0b11110110);
}

/**
 End of File
```

```
*/
```

lcd.h

```
#ifndef XC_HEADER_TEMPLATE_H
#define XC_HEADER_TEMPLATE_H

#include <xc.h>

void Lcd_initialize (void);
void Lcd_clear (void);
void Lcd_com (unsigned char i);
void Lcd_dat (unsigned char i);
void Lcd_line (unsigned char i);
void Lcd_str (const unsigned char *x);
void Lcd_custom (void);
void Lcd_barosk (void);
void Lcd_barossum1 (void);
void Lcd_barossum2 (void);
void Lcd_clearsum1 (void);
void Lcd_clearsum2 (void);

#ifdef  __cplusplus

#endif /* __cplusplus */

#ifdef  __cplusplus

#endif /* __cplusplus */

#endif  /* XC_HEADER_TEMPLATE_H */

/**
 End of File
*/
```

serial.c

```
#include "mcc.h"
#include <string.h>
#include <stdio.h>
#include "serial.h"

#define  EEAddr    0x0000      // EEPROM address

char StartMsgT[]="RS1-OK";
char StartMsgG[]="RS2-OK";
uint8_t MsgT[29];                       //Holds 28 sensor characters plus '\0'
uint8_t MsgTlast[29];                    //Holds last sensor message that was processed
uint8_t MsgG[29];                       //Holds 28 device characters plus '\0'
uint8_t MsgComp[29];                       //Compare string
uint8_t MsgOK[29];                      //Check device OK reply string
uint8_t MsgCon[29];                      //Check sensor CON string
char MsgZero[6];                       //Zero message when sum is zeroed
char Msgsum[10];                        //transformed sum int to ASCII
char Msgwgh[5];                        //transformed weight int to ASCII
char Msgid[5];                        //To display id of bin "info part"
char Msgidserial[5];                     //To display id of bin "serial part"
uint8_t BufferSizeT=0;                    //Serial Buffer size for incoming sensor message
uint8_t BufferSizeG=0;                     //Serial Buffer size for incoming device message
uint8_t k=0;                          //pointer for sensor start check
uint8_t t=0;                          //pointer for sensor message read-write
uint8_t g=0;                          //pointer for device message read-write
uint8_t d=0;                          //pointer for messages delete function
uint8_t sensorcon=0;                       //connection check for sensor
uint8_t devicecon=0;                       //connection check for device
uint8_t datalen=0;                        //sensor data length
uint8_t datas=0;                         //sensor data check for Sx message
uint8_t units=0;                         //units of weight message
uint8_t tens=0;                          //tens of weight message
uint8_t hundreds=0;                        //hundreds of weight message
uint16_t weight=0;                        //weight of bin
```

```
unsigned int lower_serialid=0;                  //id of bin first 8bits
unsigned int upper_serialid=0;                  //id of bin first 8bits
unsigned int serialid=0;                       //id of bin
unsigned int sumweight=0;                       //weight sum of all bins
unsigned int lower_sum=0;                       //first 8bits of sum to store
unsigned int upper_sum=0;                       //last 8bits of sum to store
uint8_t zerosent=1;

//Msg LCD       "12345678901234567890"
char Msgcl[]    ="                  ";
char Msgtok[]   ="Sensor OK";
char Msggok[]   ="-Device OK";
char Msgtnok[]  ="   SENSOR PROBLEM   ";
char Msggnok[]  ="   DEVICE PROBLEM   ";
char Msgserv[]  ="  Contact  Service  ";
char Msgnoid[]  ="No ID Tag";                       //when no id tag is present
char Msgyesid[] ="ID: ";                            //when id tag is present

void Lcd_str (const unsigned char *x);
void Lcd_line (unsigned char i);
void Lcd_dat (unsigned char i);
void Lcd_clear (void);
void Lcd_barosk (void);
void Lcd_barossum1 (void);
void Lcd_barossum2 (void);
void Lcd_clearsum1 (void);
void Lcd_clearsum2 (void);

void Serial_Initialize(void)
{
   lower_sum = DATAEE_ReadByte(EEAddr);              //read sum from storage - first 8bits
   upper_sum = DATAEE_ReadByte(EEAddr+1);            //read sum from storage - last 8bits
   NVMCON1=0x80;                                  //for ERRATA correction
   sumweight = (upper_sum << 8) | lower_sum;         //construct sum from stored values
   //try for 10s to get sensor message
   for (k=0 ; k < 10 ; k++)
   {
```

```
    __delay_ms(1000);
    Serial1_input();       //read if there is message
    sensorcon = MsgT_con(); //check if message is CON
    if (sensorcon==1)
    {
      break;            //if you get CON break "for loop"
    }
}
//if sensor dead then display problem and blink error LED
if (sensorcon==0)
{
    Lcd_clear();
    Lcd_line(1);
    Lcd_str(Msgtnok);
    Lcd_line(3);
    Lcd_str(Msgserv);
    while (1)
    {
      lederror_SetHigh();
      __delay_ms(1000);
      lederror_SetLow();
    }
}
//if sensor OK display OK and check device
else
{
   Lcd_line(4);
   Lcd_str(Msgtok);
}
__delay_ms(500);
Serial2_input();                    //read if there is message
devicecon=MsgG_con();                    //check if message is OK
if (devicecon==1)                    //if device OK display OK and return to "main"
{
   Lcd_str(Msggok);
   return;
}
```

```
  //if device NOK then try 2 more times and check only device
  Serial1_input();
  __delay_ms(500);
  Serial2_input();                    //read if there is message
  devicecon=MsgG_con();                    //check if message is OK
  if (devicecon==1)                    //if device OK display OK and return to "main"
  {
    Lcd_str(Msggok);
    return;
  }
  //if device NOK then try 1 more time and check only device
  Serial1_input();
  __delay_ms(500);
  Serial2_input();                    //read if there is message
  devicecon=MsgG_con();                    //check if message is OK
  if (devicecon==1)                    //if device OK display OK and return to "main"
  {
    Lcd_str(Msggok);
    return;
  }
  else
  {
     //if device dead then display problem and blink error LED
     Lcd_clear();
     Lcd_line(1);
     Lcd_str(Msggnok);
     Lcd_line(3);
     Lcd_str(Msgserv);
     while (1)
     {
        lederror_SetHigh();
        __delay_ms(1000);
        lederror_SetHigh();
     }
  }
}
```

```
void Serial1_input(void)
{
  if(eusart1RxCount!=0) //if sensor port buffer not empty read message
  {
    BufferSizeT=eusart1RxCount;
    for (t = 0 ; t < BufferSizeT ; t++)
    {
      MsgT[t] = EUSART1_Read();
    }
    for (t = 0 ; t < BufferSizeT ; t++)
    {
      putch2(MsgT[t]);
    }
  }
}

void Serial2_input(void)
{
  if(eusart2RxCount!=0) //if device port buffer not empty read message
  {
    BufferSizeG=eusart2RxCount;
    for (g = 0 ; g < BufferSizeG ; g++)
    {
      MsgG[g] = EUSART2_Read();
    }
    for (g = 0 ; g < BufferSizeG ; g++)
    {
      putch1(MsgG[g]);
    }
  }
}

void MsgT_delete(void)
{
  //write null in all sensor message array
  for (d = 0 ; d < 29 ; d++)
  {
```

```
      MsgT[d]='\0';
   }
}

void MsgG_delete(void)
{
   //write null in all device message array
   for (d = 0 ; d < 29 ; d++)
   {
      MsgG[d]='\0';
   }
}

void Msg_clear(void)
{
   //check if sensor message not null to clear it
   if (strcmp( MsgT,MsgComp))
   {
      MsgT_delete();
   }
   //check if device message not null to clear it
   if (strcmp( MsgG,MsgComp))
   {
      MsgG_delete();
   }
   for (d = 0 ; d < 5 ; d++)
   {
      Msgid[d]='\0';
      Msgidserial[d]='\0';
   }
}
void MsgG_check(void)
{
   MsgOK[0]='O';
   MsgOK[1]='K';
   MsgOK[2]=MsgT[BufferSizeT-3];
   MsgOK[3]='\r';
```

```
MsgOK[4]='\n';
datalen=0;
datas=0;
datalen=MsgT[1];
datas=MsgT[2];
if (datalen==2 && datas==0x53)
{
   Lcd_clear();
   Lcd_line(1);
   Lcd_str(Msgtnok);
   Lcd_line(3);
   Lcd_str(Msgserv);
   lederror_SetHigh();
}
else
{
   datalen=0;
   datas=0;
}
if (!strcmp(MsgG,MsgOK))
{
   //parsing of sensor message and display W:, ID: and SUM
   if (strcmp(MsgT,MsgTlast))
   {
      datalen=MsgT[1];
      //#lW:xxxID:hhhhhhhhhhhh
      //012345678901234567890
      switch (datalen)
      {
         case 5:
         {
            //calculate weight
            units=tens=hundreds=weight=0;
            units=MsgT[6] - 0x30;
            tens=MsgT[5] - 0x30;
            hundreds=MsgT[4] - 0x30;
            weight=(hundreds*100)+(tens*10)+units;
```

```
   sprintf (Msgwgh,"%u",weight);              //export weight to ascii
   sumweight=sumweight+weight;                  //calculate sum
   lower_sum = sumweight & 0xff;               //split sum to store - first 8bits
   upper_sum = (sumweight >> 8) & 0xff;        //split sum to store - last 8bits
   DATAEE_WriteByte(EEAddr, lower_sum);        //store sum - first 8bits
   DATAEE_WriteByte(EEAddr+1, upper_sum);      //store sum - last 8bits
   NVMCON1=0x80;                             //for ERRATA correction
   sprintf (Msgsum,"%u",sumweight);           //export sum to ascii
   //copy sensor displayed message to last message
   for (d = 0 ; d < 29 ; d++)
   {
      MsgTlast[d] = MsgT[d];
   }
   //print weight
   Lcd_line(1);
   Lcd_str(Msgcl);
   Lcd_line(1);
   Lcd_barosk();
   Lcd_str(Msgwgh);
   //print no id msg
   Lcd_line(2);
   Lcd_str(Msgcl);
   Lcd_line(2);
   Lcd_str(Msgnoid);
   //print sum
   Lcd_line(3);
   Lcd_str(Msgcl);
   Lcd_line(3);
   Lcd_barossum1();
   Lcd_line(4);
   Lcd_str(Msgcl);
   Lcd_line(4);
   Lcd_barossum2();
   Lcd_str(Msgsum);
   break;
}
case 20:
```

```
{
  //calculate weight
  units=tens=hundreds=weight=0;
  units=MsgT[6] - 0x30;
  tens=MsgT[5] - 0x30;
  hundreds=MsgT[4] - 0x30;
  weight=(hundreds*100)+(tens*10)+units;
  sprintf(Msgwgh,"%u",weight);              //export weight to ascii
  //export ID
  Msgid[0]=MsgT[10];
  Msgid[1]=MsgT[11];
  Msgid[2]=MsgT[13];
  Msgid[3]=MsgT[14];
  upper_serialid=MsgT[20];
  lower_serialid=MsgT[21];
  serialid = (upper_serialid << 8) | lower_serialid; //construct id from lower and upper part
  sprintf(Msgidserial,"%u",serialid);              //export id serial to ascii
  sumweight=sumweight+weight;               //calculate sum
  lower_sum = sumweight & 0xff;             //split sum to store - first 8bits
  upper_sum = (sumweight >> 8) & 0xff;      //split sum to store - last 8bits
  DATAEE_WriteByte(EEAddr, lower_sum);      //store sum - first 8bits
  DATAEE_WriteByte(EEAddr+1, upper_sum);    //store sum - last 8bits
  NVMCON1=0x80;                             //for ERRATA correction
  sprintf(Msgsum,"%u",sumweight);           //export sum to ascii
  //copy sensor displayed message to last message
  for (d = 0 ; d < 29 ; d++)
  {
    MsgTlast[d] = MsgT[d];
  }
  //print weight
  Lcd_line(1);
  Lcd_str(Msgcl);
  Lcd_line(1);
  Lcd_barosk ();
  Lcd_str(Msgwgh);
  //print id
  Lcd_line(2);
```

```
                Lcd_str(Msgcl);
                Lcd_line(2);
                Lcd_str(Msgyesid);
                Lcd_str(Msgid);
                Lcd_dat(0b10110000);
                Lcd_str(Msgidserial);
                //print sum
                Lcd_line(3);
                Lcd_str(Msgcl);
                Lcd_line(3);
                Lcd_barossum1();
                Lcd_line(4);
                Lcd_str(Msgcl);
                Lcd_line(4);
                Lcd_barossum2();
                Lcd_str(Msgsum);
                break;
              }
            }
          }
        }
}
uint8_t MsgT_con(void)
{
   uint8_t c = 0;
   MsgCon[0]=0x23;
   MsgCon[1]=0x03;
   MsgCon[2]=0x43;
   MsgCon[3]=0x4F;
   MsgCon[4]=0x4E;
   MsgCon[5]=0xE3;
   MsgCon[6]='\r';
   MsgCon[7]='\n';
   if (!strcmp(MsgT,MsgCon))
   {
      c=1;
      return c;
```

```
    }
    else
    {
      c=0;
      return c;
    }
}
uint8_t MsgG_con(void)
{
    uint8_t c = 0;
    MsgOK[0]='O';
    MsgOK[1]='K';
    MsgOK[2]=MsgT[BufferSizeT-3];
    MsgOK[3]='\r';
    MsgOK[4]='\n';
    if (!strcmp(MsgG,MsgOK))
    {
      c=1;
      return c;
    }
    else
    {
      c=0;
      return c;
    }
}

void Check_unload (void)
{
    if (resetsum_PORT>0)
    {
      __delay_ms(500);
      if (resetsum_PORT>0)
      {
        while (resetsum_PORT>0)
        {
```

```
        }
        sumweight=0;                        //zero sum
        DATAEE_WriteByte(EEAddr, sumweight);        //store sum
        NVMCON1=0x80;                       //for ERRATA correction
        //send clear sum message to server code "Z0", '23025A308C'
        while (zerosent>0)
        {
           zerosent=Send_zero();
           __delay_ms(1000);
        }
        //clear screen and print clear sum
        Lcd_line(1);
        Lcd_str(Msgcl);
        Lcd_line(1);
        Lcd_clearsum1();
        Lcd_line(2);
        Lcd_str(Msgcl);
        Lcd_line(2);
        Lcd_clearsum2();
        Lcd_line(3);
        Lcd_str(Msgcl);
        Lcd_line(4);
        Lcd_str(Msgcl);
        zerosent=1;
        return;
      }
    }
    else
    {
      return;
    }
}
uint8_t Send_zero (void)
{
    uint8_t z=0;
    MsgZero[0]=0x23;
    MsgZero[1]=0x02;
```

```
    MsgZero[2]=0x5A;
    MsgZero[3]=0x30;
    MsgZero[4]=0x8C;
    for (t = 0 ; t < 6 ; t++)
    {
      putch2(MsgZero[t]);
    }
    __delay_ms(800);
    if(eusart2RxCount!=0) //if device port buffer not empty read message
    {
      BufferSizeG=eusart2RxCount;
      for (g = 0 ; g < BufferSizeG ; g++)
      {
        MsgG[g] = EUSART2_Read();
      }
    }
    MsgOK[0]='O';
    MsgOK[1]='K';
    MsgOK[2]=0x8C;
    MsgOK[3]='\r';
    MsgOK[4]='\n';
    if (!strcmp(MsgG,MsgOK))
    {
      z=0;
      return z;
    }
    else
    {
      z=1;
      return z;
    }
}

/**
 End of File
*/
```

serial.h

```
#ifndef XC_HEADER_TEMPLATE_H
#define XC_HEADER_TEMPLATE_H

#include <xc.h>

extern uint8_t MsgT[29];                    //Holds 28 sensor characters plus '\0'
extern uint8_t MsgTlast[29];                 //Holds last sensor message that was prosseced
extern uint8_t MsgG[29];                    //Holds 28 device characters plus '\0'
extern uint8_t MsgComp[29];                    //Compare string
extern uint8_t MsgOK[29];                   //Check device reply string
extern uint8_t MsgCon[29];                   //Check device reply string

void Serial_Initialize(void);
void Serial1_input(void); //read sensor message
void Serial2_input(void); //read device message
void Msg_clear(void);    //check sensor and device messages and if not null clear them
void MsgT_delete(void);  //delete sensor message
void MsgG_delete(void);  //delete device message
void MsgG_check(void);   //check device message and if "OK" then display W: and add SUM
uint8_t MsgT_con(void);  //check sensor message and if "CON"
uint8_t MsgG_con(void);  //check device message and if "OK"
void Check_unload (void);   //check if truck is unloading to zero sum
uint8_t Send_zero (void);

#ifdef  __cplusplus
extern "C" {
#endif /* __cplusplus */

#ifdef  __cplusplus

#endif /* __cplusplus */

#endif  /* XC_HEADER_TEMPLATE_H */
```

```
/**
 End of File
*/
```

powerup.as

```
#include <xc.inc>
   GLOBAL  powerup, start
   PSECT   powerup, class=CODE, delta=1,reloc=2

powerup:
   BSF     NVMCON1, 7
   GOTO    start
   end

/**
 End of File
*/
```

device_config.c

```
// CONFIG1L
#pragma config FEXTOSC = OFF    // External Oscillator mode Selection bits->Oscillator not enabled
#pragma config RSTOSC = HFINTOSC_64MHZ    // Power-up default value for COSC bits->HFINTOSC with HFFRQ = 64 MHz and CDIV = 1:1

// CONFIG1H
#pragma config CLKOUTEN = OFF    // Clock Out Enable bit->CLKOUT function is disabled
#pragma config CSWEN = ON    // Clock Switch Enable bit->Writing to NOSC and NDIV is allowed
#pragma config FCMEN = ON    // Fail-Safe Clock Monitor Enable bit->Fail-Safe Clock Monitor enabled

// CONFIG2L
#pragma config MCLRE = EXTMCLR    // Master Clear Enable bit->If LVP = 0, MCLR pin is MCLR; If LVP = 1, RE3 pin function is MCLR
#pragma config PWRTE = OFF    // Power-up Timer Enable bit->Power up timer disabled
#pragma config LPBOREN = OFF    // Low-power BOR enable bit->ULPBOR disabled
#pragma config BOREN = SBORDIS    // Brown-out Reset Enable bits->Brown-out Reset enabled , SBOREN bit is ignored

// CONFIG2H
#pragma config BORV = VBOR_2P45    // Brown Out Reset Voltage selection bits->Brown-out Reset Voltage (VBOR) set to 2.45V
#pragma config ZCD = OFF    // ZCD Disable bit->ZCD disabled. ZCD can be enabled by setting the ZCDSEN bit of ZCDCON
#pragma config PPS1WAY = ON    // PPSLOCK bit One-Way Set Enable bit->PPSLOCK bit can be cleared and set only once; PPS registers remain locked after one clear/set
cycle
#pragma config STVREN = ON    // Stack Full/Underflow Reset Enable bit->Stack full/underflow will cause Reset
#pragma config DEBUG = OFF    // Debugger Enable bit->Background debugger disabled
#pragma config XINST = OFF    // Extended Instruction Set Enable bit->Extended Instruction Set and Indexed Addressing Mode disabled

// CONFIG3L
#pragma config WDTCPS = WDTCPS_31    // WDT Period Select bits->Divider ratio 1:65536; software control of WDTPS
#pragma config WDTE = OFF    // WDT operating mode->WDT Disabled

// CONFIG3H
#pragma config WDTCWS = WDTCWS_7    // WDT Window Select bits->window always open (100%); software control; keyed access not required
#pragma config WDTCCS = SC    // WDT input clock selector->Software Control

// CONFIG4L
#pragma config WRT0 = OFF    // Write Protection Block 0->Block 0 (000800-003FFFh) not write-protected
```

```
#pragma config WRT1 = OFF    // Write Protection Block 1->Block 1 (004000-007FFFh) not write-protected
#pragma config WRT2 = OFF    // Write Protection Block 2->Block 2 (008000-00BFFFh) not write-protected
#pragma config WRT3 = OFF    // Write Protection Block 3->Block 3 (00C000-00FFFFh) not write-protected
#pragma config WRT4 = OFF    // Write Protection Block 3->Block 4 (010000-013FFFh) not write-protected
#pragma config WRT5 = OFF    // Write Protection Block 3->Block 5 (014000-017FFFh) not write-protected
#pragma config WRT6 = OFF    // Write Protection Block 3->Block 6 (018000-01BFFFh) not write-protected
#pragma config WRT7 = OFF    // Write Protection Block 3->Block 7 (01C000-01FFFFh) not write-protected

// CONFIG4H
#pragma config WRTC = OFF    // Configuration Register Write Protection bit->Configuration registers (300000-30000Bh) not write-protected
#pragma config WRTB = OFF    // Boot Block Write Protection bit->Boot Block (000000-0007FFh) not write-protected
#pragma config WRTD = OFF    // Data EEPROM Write Protection bit->Data EEPROM not write-protected
#pragma config SCANE = ON    // Scanner Enable bit->Scanner module is available for use, SCANMD bit can control the module
#pragma config LVP = ON    // Low Voltage Programming Enable bit->Low voltage programming enabled. MCLR/VPP pin function is MCLR. MCLRE configuration bit is
ignored

// CONFIG5L
#pragma config CP = OFF    // UserNVM Program Memory Code Protection bit->UserNVM code protection disabled
#pragma config CPD = OFF    // DataNVM Memory Code Protection bit->DataNVM code protection disabled

// CONFIG6L
#pragma config EBTR0 = OFF    // Table Read Protection Block 0->Block 0 (000800-003FFFh) not protected from table reads executed in other blocks
#pragma config EBTR1 = OFF    // Table Read Protection Block 1->Block 1 (004000-007FFFh) not protected from table reads executed in other blocks
#pragma config EBTR2 = OFF    // Table Read Protection Block 2->Block 2 (008000-00BFFFh) not protected from table reads executed in other blocks
#pragma config EBTR3 = OFF    // Table Read Protection Block 3->Block 3 (00C000-00FFFFh) not protected from table reads executed in other blocks
#pragma config EBTR4 = OFF    // Table Read Protection Block 4->Block 4 (010000-013FFFh) not protected from table reads executed in other blocks
#pragma config EBTR5 = OFF    // Table Read Protection Block 5->Block 5 (014000-017FFFh) not protected from table reads executed in other blocks
#pragma config EBTR6 = OFF    // Table Read Protection Block 6->Block 6 (018000-01BFFFh) not protected from table reads executed in other blocks
#pragma config EBTR7 = OFF    // Table Read Protection Block 7->Block 7 (01C000-01FFFFh) not protected from table reads executed in other blocks

// CONFIG6H
#pragma config EBTRB = OFF    // Boot Block Table Read Protection bit->Boot Block (000000-0007FFh) not protected from table reads executed in other blocks

/**
 End of File
*/
```

device_config.h

```
#ifndef DEVICE_CONFIG_H
#define DEVICE_CONFIG_H

#define _XTAL_FREQ 48000000

#endif  /* DEVICE_CONFIG_H */

/**
 End of File
*/
```

eusart1.c

```
#include "eusart1.h"

#define EUSART1_TX_BUFFER_SIZE 32
#define EUSART1_RX_BUFFER_SIZE 32

volatile uint8_t eusart1TxHead = 0;
volatile uint8_t eusart1TxTail = 0;
volatile uint8_t eusart1TxBuffer[EUSART1_TX_BUFFER_SIZE];
volatile uint8_t eusart1TxBufferRemaining;

volatile uint8_t eusart1RxHead = 0;
volatile uint8_t eusart1RxTail = 0;
volatile uint8_t eusart1RxBuffer[EUSART1_RX_BUFFER_SIZE];
volatile uint8_t eusart1RxCount;

void EUSART1_Initialize(void)
{
    // disable interrupts before changing states
    PIE3bits.RC1IE = 0;
    EUSART1_SetRxInterruptHandler(EUSART1_Receive_ISR);
    PIE3bits.TX1IE = 0;
    EUSART1_SetTxInterruptHandler(EUSART1_Transmit_ISR);
    // Set the EUSART1 module to the options selected in the user interface.

    // ABDOVF no_overflow; SCKP Non-Inverted; BRG16 16bit_generator; WUE disabled; ABDEN disabled;
    BAUD1CON = 0x08;

    // SPEN enabled; RX9 8-bit; CREN enabled; ADDEN disabled; SREN disabled;
    RC1STA = 0x90;

    // TX9 8-bit; TX9D 0; SENDB sync_break_complete; TXEN enabled; SYNC asynchronous; BRGH hi_speed; CSRC slave;
    TX1STA = 0x24;

    // SP1BRGL 225;
```

```
    SP1BRGL = 0xE1;

    // SP1BRGH 4;
    SP1BRGH = 0x04;


    // initializing the driver state
    eusart1TxHead = 0;
    eusart1TxTail = 0;
    eusart1TxBufferRemaining = sizeof(eusart1TxBuffer);

    eusart1RxHead = 0;
    eusart1RxTail = 0;
    eusart1RxCount = 0;

    // enable receive interrupt
    PIE3bits.RC1IE = 1;
}

uint8_t EUSART1_is_tx_ready(void)
{
    return eusart1TxBufferRemaining;
}

uint8_t EUSART1_is_rx_ready(void)
{
    return eusart1RxCount;
}

bool EUSART1_is_tx_done(void)
{
    return TX1STAbits.TRMT;
}

uint8_t EUSART1_Read(void)
{
    uint8_t readValue  = 0;
```

```
    while(0 == eusart1RxCount)
    {
        break;
    }

    readValue = eusart1RxBuffer[eusart1RxTail++];
    if(sizeof(eusart1RxBuffer) <= eusart1RxTail)
    {
        eusart1RxTail = 0;
    }
    PIE3bits.RC1IE = 0;
    eusart1RxCount--;
    PIE3bits.RC1IE = 1;

    return readValue;
}

void EUSART1_Write(uint8_t txData)
{
    while(0 == eusart1TxBufferRemaining)
    {
    }

    if(0 == PIE3bits.TX1IE)
    {
        TX1REG = txData;
    }
    else
    {
        PIE3bits.TX1IE = 0;
        eusart1TxBuffer[eusart1TxHead++] = txData;
        if(sizeof(eusart1TxBuffer) <= eusart1TxHead)
        {
            eusart1TxHead = 0;
        }
        eusart1TxBufferRemaining--;
```

```
    }
    PIE3bits.TX1IE = 1;
}

void EUSART1_Transmit_ISR(void)
{

    // add your EUSART1 interrupt custom code
    if(sizeof(eusart1TxBuffer) > eusart1TxBufferRemaining)
    {
        TX1REG = eusart1TxBuffer[eusart1TxTail++];
        if(sizeof(eusart1TxBuffer) <= eusart1TxTail)
        {
            eusart1TxTail = 0;
        }
        eusart1TxBufferRemaining++;
    }
    else
    {
        PIE3bits.TX1IE = 0;
    }
}

void EUSART1_Receive_ISR(void)
{

    if(1 == RC1STAbits.OERR)
    {
        // EUSART1 error - restart

        RC1STAbits.CREN = 0;
        RC1STAbits.CREN = 1;
    }

    // buffer overruns are ignored
    eusart1RxBuffer[eusart1RxHead++] = RC1REG;
    if(sizeof(eusart1RxBuffer) <= eusart1RxHead)
```

```
    {
        eusart1RxHead = 0;
    }
    eusart1RxCount++;
}

void EUSART1_SetTxInterruptHandler(void (* interruptHandler)(void)){
    EUSART1_TxDefaultInterruptHandler = interruptHandler;
}

void EUSART1_SetRxInterruptHandler(void (* interruptHandler)(void)){
    EUSART1_RxDefaultInterruptHandler = interruptHandler;
}

//-------My code---------------------------------------------------------------

void putch1(const char txData)
{
    EUSART1_Write(txData);
}

void putstr1(const char *x)
{
    while(*x)
    {
        EUSART1_Write(*x++);
    }
}

//-----------------------------------------------------------------------------


/**
  End of File
*/
```

eusart1.h

```
#ifndef EUSART1_H
#define EUSART1_H

#include <xc.h>
#include <stdbool.h>
#include <stdint.h>

#ifdef __cplusplus  // Provide C++ Compatibility

#endif

#define EUSART1_DataReady  (EUSART1_is_rx_ready())

extern volatile uint8_t eusart1TxBufferRemaining;
extern volatile uint8_t eusart1RxCount;

void (*EUSART1_TxDefaultInterruptHandler)(void);
void (*EUSART1_RxDefaultInterruptHandler)(void);
void EUSART1_Initialize(void);
uint8_t EUSART1_is_tx_ready(void);
uint8_t EUSART1_is_rx_ready(void);
bool EUSART1_is_tx_done(void);
uint8_t EUSART1_Read(void);
void EUSART1_Write(uint8_t txData);
void EUSART1_Transmit_ISR(void);
void EUSART1_Receive_ISR(void);
void EUSART1_SetTxInterruptHandler(void (* interruptHandler)(void));
void EUSART1_SetRxInterruptHandler(void (* interruptHandler)(void));

#ifdef __cplusplus  // Provide C++ Compatibility

#endif

#endif  // EUSART1_H
```

```
//-------My code-----------------------------------------------------------------

//put one character to serial1
void putch1(char txData);

//put string of characters to serial1
void putstr1(const char *x);

//-------------------------------------------------------------------------

/**
 End of File
*/
```

eusart2.c

```
#include "eusart2.h"

#define EUSART2_TX_BUFFER_SIZE 32
#define EUSART2_RX_BUFFER_SIZE 32

volatile uint8_t eusart2TxHead = 0;
volatile uint8_t eusart2TxTail = 0;
volatile uint8_t eusart2TxBuffer[EUSART2_TX_BUFFER_SIZE];
volatile uint8_t eusart2TxBufferRemaining;

volatile uint8_t eusart2RxHead = 0;
volatile uint8_t eusart2RxTail = 0;
volatile uint8_t eusart2RxBuffer[EUSART2_RX_BUFFER_SIZE];
volatile uint8_t eusart2RxCount;

void EUSART2_Initialize(void)
{
    // disable interrupts before changing states
    PIE3bits.RC2IE = 0;
    EUSART2_SetRxInterruptHandler(EUSART2_Receive_ISR);
    PIE3bits.TX2IE = 0;
    EUSART2_SetTxInterruptHandler(EUSART2_Transmit_ISR);
    // Set the EUSART2 module to the options selected in the user interface.

    // ABDOVF no_overflow; SCKP Non-Inverted; BRG16 16bit_generator; WUE disabled; ABDEN disabled;
    BAUD2CON = 0x08;

    // SPEN enabled; RX9 8-bit; CREN enabled; ADDEN disabled; SREN disabled;
    RC2STA = 0x90;

    // TX9 8-bit; TX9D 0; SENDB sync_break_complete; TXEN enabled; SYNC asynchronous; BRGH hi_speed; CSRC slave;
    TX2STA = 0x24;

    // SP2BRGL 225;
```

```
    SP2BRGL = 0xE1;

    // SP2BRGH 4;
    SP2BRGH = 0x04;


    // initializing the driver state
    eusart2TxHead = 0;
    eusart2TxTail = 0;
    eusart2TxBufferRemaining = sizeof(eusart2TxBuffer);

    eusart2RxHead = 0;
    eusart2RxTail = 0;
    eusart2RxCount = 0;

    // enable receive interrupt
    PIE3bits.RC2IE = 1;
}

uint8_t EUSART2_is_tx_ready(void)
{
    return eusart2TxBufferRemaining;
}

uint8_t EUSART2_is_rx_ready(void)
{
    return eusart2RxCount;
}

bool EUSART2_is_tx_done(void)
{
    return TX2STAbits.TRMT;
}

uint8_t EUSART2_Read(void)
{
    uint8_t readValue  = 0;
```

```
    while(0 == eusart2RxCount)
    {
        break;
    }

    readValue = eusart2RxBuffer[eusart2RxTail++];
    if(sizeof(eusart2RxBuffer) <= eusart2RxTail)
    {
        eusart2RxTail = 0;
    }
    PIE3bits.RC2IE = 0;
    eusart2RxCount--;
    PIE3bits.RC2IE = 1;

    return readValue;
}

void EUSART2_Write(uint8_t txData)
{
    while(0 == eusart2TxBufferRemaining)
    {
    }

    if(0 == PIE3bits.TX2IE)
    {
        TX2REG = txData;
    }
    else
    {
        PIE3bits.TX2IE = 0;
        eusart2TxBuffer[eusart2TxHead++] = txData;
        if(sizeof(eusart2TxBuffer) <= eusart2TxHead)
        {
            eusart2TxHead = 0;
        }
        eusart2TxBufferRemaining--;
```

```
    }
    PIE3bits.TX2IE = 1;
}

void EUSART2_Transmit_ISR(void)
{

    // add your EUSART2 interrupt custom code
    if(sizeof(eusart2TxBuffer) > eusart2TxBufferRemaining)
    {
        TX2REG = eusart2TxBuffer[eusart2TxTail++];
        if(sizeof(eusart2TxBuffer) <= eusart2TxTail)
        {
            eusart2TxTail = 0;
        }
        eusart2TxBufferRemaining++;
    }
    else
    {
        PIE3bits.TX2IE = 0;
    }
}

void EUSART2_Receive_ISR(void)
{

    if(1 == RC2STAbits.OERR)
    {
        // EUSART2 error - restart

        RC2STAbits.CREN = 0;
        RC2STAbits.CREN = 1;
    }

    // buffer overruns are ignored
    eusart2RxBuffer[eusart2RxHead++] = RC2REG;
    if(sizeof(eusart2RxBuffer) <= eusart2RxHead)
```

```
    {
        eusart2RxHead = 0;
    }
    eusart2RxCount++;
}

void EUSART2_SetTxInterruptHandler(void (* interruptHandler)(void)){
    EUSART2_TxDefaultInterruptHandler = interruptHandler;
}

void EUSART2_SetRxInterruptHandler(void (* interruptHandler)(void)){
    EUSART2_RxDefaultInterruptHandler = interruptHandler;
}

//-------My code--------------------------------------------------------------

void putch2(const char txData)
{
    EUSART2_Write(txData);
}

void putstr2(const char *x)
{
    while(*x)
    {
        EUSART2_Write(*x++);
    }
}

//--------------------------------------------------------------------------

/**
  End of File
*/
```

eusart2.h

```
#ifndef EUSART2_H
#define EUSART2_H

#include <xc.h>
#include <stdbool.h>
#include <stdint.h>

#ifdef __cplusplus  // Provide C++ Compatibility

#endif

#define EUSART2_DataReady  (EUSART2_is_rx_ready())
extern volatile uint8_t eusart2TxBufferRemaining;
extern volatile uint8_t eusart2RxCount;
void (*EUSART2_TxDefaultInterruptHandler)(void);
void (*EUSART2_RxDefaultInterruptHandler)(void);
void EUSART2_Initialize(void);
uint8_t EUSART2_is_tx_ready(void);
uint8_t EUSART2_is_rx_ready(void);
bool EUSART2_is_tx_done(void);
uint8_t EUSART2_Read(void);
void EUSART2_Write(uint8_t txData);
void EUSART2_Transmit_ISR(void);
void EUSART2_Receive_ISR(void);
void EUSART2_SetTxInterruptHandler(void (* interruptHandler)(void));
void EUSART2_SetRxInterruptHandler(void (* interruptHandler)(void));

#ifdef __cplusplus  // Provide C++ Compatibility

#endif

#endif  // EUSART2_H

//-------My code---------------------------------------------------------------
```

```
//put one character to serial1
void putch2(char txData);

//put string of characters to serial1
void putstr2(const char *x);

//------------------------------------------------------------------------------

/**
 End of File
*/
```

interrupt_manager.c

```
#include "interrupt_manager.h"
#include "mcc.h"

void  INTERRUPT_Initialize (void)
{
   INTCONbits.IPEN = 0;
}

void __interrupt() INTERRUPT_InterruptManager (void)
{
   // interrupt handler
   if(INTCONbits.PEIE == 1)
   {
      if(PIE3bits.TX2IE == 1 && PIR3bits.TX2IF == 1)
      {
         EUSART2_TxDefaultInterruptHandler();
      }
      else if(PIE3bits.RC2IE == 1 && PIR3bits.RC2IF == 1)
      {
         EUSART2_RxDefaultInterruptHandler();
      }
      else if(PIE3bits.TX1IE == 1 && PIR3bits.TX1IF == 1)
      {
         EUSART1_TxDefaultInterruptHandler();
      }
      else if(PIE3bits.RC1IE == 1 && PIR3bits.RC1IF == 1)
      {
         EUSART1_RxDefaultInterruptHandler();
      }
      else
      {
         //Unhandled Interrupt
      }
   }
```

```
    else
    {
        //Unhandled Interrupt
    }
}

/**
 End of File
*/
```

interrupt_manager.h

```
#ifndef INTERRUPT_MANAGER_H
#define INTERRUPT_MANAGER_H
#define INTERRUPT_GlobalInterruptEnable() (INTCONbits.GIE = 1)
#define INTERRUPT_GlobalInterruptDisable() (INTCONbits.GIE = 0)
#define INTERRUPT_PeripheralInterruptEnable() (INTCONbits.PEIE = 1)
#define INTERRUPT_PeripheralInterruptDisable() (INTCONbits.PEIE = 0)
void INTERRUPT_Initialize (void);

#endif  // INTERRUPT_MANAGER_H

/**
 End of File
*/
```

mcc.c

```
#include "mcc.h"

void SYSTEM_Initialize(void)
{
    INTERRUPT_Initialize();
    PMD_Initialize();
    PIN_MANAGER_Initialize();
    OSCILLATOR_Initialize();
    EUSART1_Initialize();
    EUSART2_Initialize();
}

void OSCILLATOR_Initialize(void)
{
    // NOSC HFINTOSC; NDIV 1;
    OSCCON1 = 0x60;
    // CSWHOLD may proceed; SOSCPWR Low power;
    OSCCON3 = 0x00;
    // MFOEN disabled; LFOEN disabled; ADOEN disabled; SOSCEN disabled; EXTOEN disabled; HFOEN disabled;
    OSCEN = 0x00;
    // HFFRQ 48_MHz;
    OSCFRQ = 0x07;
    // TUN 0;
    OSCTUNE = 0x00;
}

void PMD_Initialize(void)
{
    // CLKRMD CLKR enabled; SYSCMD SYSCLK enabled; SCANMD SCANNER enabled; FVRMD FVR enabled; IOCMD IOC enabled; CRCMD CRC enabled; HLVDMD HLVD
enabled; NVMMD NVM enabled;
    PMD0 = 0x00;
    // TMR0MD TMR0 enabled; TMR1MD TMR1 enabled; TMR4MD TMR4 enabled; TMR5MD TMR5 enabled; TMR2MD TMR2 enabled; TMR3MD TMR3 enabled; TMR6MD
TMR6 enabled;
    PMD1 = 0x00;
```

```
    // ZCDMD ZCD enabled; DACMD DAC enabled; CMP1MD CMP1 enabled; ADCMD ADC enabled; CMP2MD CMP2 enabled;
    PMD2 = 0x00;
    // CCP2MD CCP2 enabled; CCP1MD CCP1 enabled; PWM4MD PWM4 enabled; PWM3MD PWM3 enabled;
    PMD3 = 0x00;
    // CWG1MD CWG1 enabled; UART2MD EUSART2 enabled; MSSP1MD MSSP1 enabled; UART1MD EUSART enabled; MSSP2MD MSSP2 enabled;
    PMD4 = 0x00;
    // DSMMD DSM enabled;
    PMD5 = 0x00;
}


/**
 End of File
*/
```

mcc.h

```
#ifndef MCC_H
#define MCC_H
#include <xc.h>
#include "device_config.h"
#include "pin_manager.h"
#include <stdint.h>
#include <stdbool.h>
#include "interrupt_manager.h"
#include "memory.h"
#include "eusart1.h"
#include "eusart2.h"

void SYSTEM_Initialize(void);
void OSCILLATOR_Initialize(void);
void PMD_Initialize(void);


#endif  /* MCC_H */

/**
 End of File
*/
```

memory.c

```
#include <xc.h>
#include "memory.h"

uint8_t FLASH_ReadByte(uint32_t flashAddr)
{
   NVMCON1bits.NVMREG = 2;
   TBLPTRU = (uint8_t)((flashAddr & 0x00FF0000) >> 16);
   TBLPTRH = (uint8_t)((flashAddr & 0x0000FF00)>> 8);
   TBLPTRL = (uint8_t)(flashAddr & 0x000000FF);

   asm("TBLRD");

   return (TABLAT);
}

uint16_t FLASH_ReadWord(uint32_t flashAddr)
{
   return ((((uint16_t)FLASH_ReadByte(flashAddr+1))<<8)|(FLASH_ReadByte(flashAddr)));
}

void FLASH_WriteByte(uint32_t flashAddr, uint8_t *flashRdBufPtr, uint8_t byte)
{
   uint32_t blockStartAddr = (uint32_t)(flashAddr & ((END_FLASH-1) ^ (ERASE_FLASH_BLOCKSIZE-1)));
   uint8_t offset = (uint8_t)(flashAddr & (ERASE_FLASH_BLOCKSIZE-1));
   uint8_t i;

   // Entire row will be erased, read and save the existing data
   for (i=0; i<ERASE_FLASH_BLOCKSIZE; i++)
   {
      flashRdBufPtr[i] = FLASH_ReadByte((blockStartAddr+i));
   }

   // Load byte at offset
   flashRdBufPtr[offset] = byte;
```

```
    // Writes buffer contents to current block
    FLASH_WriteBlock(blockStartAddr, flashRdBufPtr);
}

int8_t FLASH_WriteBlock(uint32_t writeAddr, uint8_t *flashWrBufPtr)
{
    uint32_t blockStartAddr  = (uint32_t )(writeAddr & ((END_FLASH-1) ^ (ERASE_FLASH_BLOCKSIZE-1)));
    uint8_t GIEBitValue = INTCONbits.GIE;     // Save interrupt enable
    uint8_t i;

    // Flash write must start at the beginning of a row
    if( writeAddr != blockStartAddr )
    {
        return -1;
    }

    // Block erase sequence
    FLASH_EraseBlock(writeAddr);

    // Block write sequence
    TBLPTRU = (uint8_t)((writeAddr & 0x00FF0000) >> 16);    // Load Table point register
    TBLPTRH = (uint8_t)((writeAddr & 0x0000FF00)>> 8);
    TBLPTRL = (uint8_t)(writeAddr & 0x000000FF);

    // Write block of data
    for (i=0; i<WRITE_FLASH_BLOCKSIZE; i++)
    {
        TABLAT = flashWrBufPtr[i];  // Load data byte

        if (i == (WRITE_FLASH_BLOCKSIZE-1))
        {
            asm("TBLWT");
        }
        else
        {
            asm("TBLWTPOSTINC");
```

```
    }
  }

  NVMCON1bits.NVMREG = 2;
  NVMCON1bits.WREN = 1;
  INTCONbits.GIE = 0; // Disable interrupts
  NVMCON2 = 0x55;
  NVMCON2 = 0xAA;
  NVMCON1bits.WR = 1;  // Start program

  NVMCON1bits.WREN = 0;   // Disable writes to memory
  INTCONbits.GIE = GIEBitValue;   // Restore interrupt enable

  return 0;
}

void FLASH_EraseBlock(uint32_t baseAddr)
{
  uint8_t GIEBitValue = INTCONbits.GIE;   // Save interrupt enable

  TBLPTRU = (uint8_t)((baseAddr & 0x00FF0000) >> 16);
  TBLPTRH = (uint8_t)((baseAddr & 0x0000FF00)>> 8);
  TBLPTRL = (uint8_t)(baseAddr & 0x000000FF);

  NVMCON1bits.NVMREG = 2;
  NVMCON1bits.WREN = 1;
  NVMCON1bits.FREE = 1;
  INTCONbits.GIE = 0; // Disable interrupts
  NVMCON2 = 0x55;
  NVMCON2 = 0xAA;
  NVMCON1bits.WR = 1;
  INTCONbits.GIE = GIEBitValue;   // Restore interrupt enable
}

void DATAEE_WriteByte(uint16_t bAdd, uint8_t bData)
{
  uint8_t GIEBitValue = INTCONbits.GIE;   // Save interrupt enable
```

```
    NVMADRH = ((bAdd >> 8) & 0x03);
    NVMADRL = (bAdd & 0xFF);
    NVMDAT = bData;
    NVMCON1bits.NVMREG = 0;
    NVMCON1bits.WREN = 1;
    INTCONbits.GIE = 0;     // Disable interrupts
    NVMCON2 = 0x55;
    NVMCON2 = 0xAA;
    NVMCON1bits.WR = 1;
    // Wait for write to complete
    while (NVMCON1bits.WR)
    {
    }

    NVMCON1bits.WREN = 0;
    INTCONbits.GIE = GIEBitValue;   // Restore interrupt enable
}

uint8_t DATAEE_ReadByte(uint16_t bAdd)
{
    NVMADRH = ((bAdd >> 8) & 0x03);
    NVMADRL = (bAdd & 0xFF);
    NVMCON1bits.NVMREG = 0;
    NVMCON1bits.RD = 1;
    NOP();  // NOPs may be required for latency at high frequencies
    NOP();

    return (NVMDAT);
}

void MEMORY_Tasks( void )
{
    PIR7bits.NVMIF = 0;
}

/**
```

```
 End of File
*/
```

memory.h

```
#ifndef MEMORY_H
#define MEMORY_H

#include <stdbool.h>
#include <stdint.h>

#ifdef __cplusplus  // Provide C++ Compatibility

#endif

#define WRITE_FLASH_BLOCKSIZE    128
#define ERASE_FLASH_BLOCKSIZE    128
#define END_FLASH              0x020000
uint8_t FLASH_ReadByte(uint32_t flashAddr);
uint16_t FLASH_ReadWord(uint32_t flashAddr);
void FLASH_WriteByte(uint32_t flashAddr, uint8_t *flashRdBufPtr, uint8_t byte);
int8_t FLASH_WriteBlock(uint32_t writeAddr, uint8_t *flashWrBufPtr);
void FLASH_EraseBlock(uint32_t baseAddr);
void DATAEE_WriteByte(uint16_t bAdd, uint8_t bData);
uint8_t DATAEE_ReadByte(uint16_t bAdd);
void MEMORY_Tasks(void);

#ifdef __cplusplus  // Provide C++ Compatibility

#endif

#endif // MEMORY_H

/**
 End of File
*/
```

pin_manager.c

```
#include <xc.h>
#include "pin_manager.h"
#include "stdbool.h"

void PIN_MANAGER_Initialize(void)
{
   /**LATx registers*/
   LATE = 0x00;
   LATD = 0x00;
   LATA = 0x00;
   LATB = 0x00;
   LATC = 0x00;

   /**TRISx registers*/
   TRISE = 0x07;
   TRISA = 0x0F;
   TRISB = 0xFE;
   TRISC = 0xBF;
   TRISD = 0xFF;

   /**ANSELx registers*/
   //ANSELD = 0xFF;
   ANSELC = 0x7F;
   ANSELB = 0xFD;
   //ANSELE = 0x07;
   ANSELA = 0xFE;

   /*My digital input-output*/
   ANSELE = 0x00;
   ANSELD = 0x00;

   /**WPUx registers*/
   WPUD = 0x00;
   WPUE = 0x00;
```

```
    WPUB = 0x00;
    WPUA = 0x00;
    WPUC = 0x00;

    /**ODx registers*/
    ODCONE = 0x00;
    ODCONA = 0x00;
    ODCONB = 0x00;
    ODCONC = 0x00;
    ODCOND = 0x00;

    /**SLRCONx registers*/
    SLRCONA = 0xFF;
    SLRCONB = 0xFF;
    SLRCONC = 0xFF;
    SLRCOND = 0xFF;
    SLRCONE = 0x07;

    /*My digital input-output*/
    RX1PPS = 0x17;   //RC7->EUSART1:RX1;
    RB0PPS = 0x0B;   //RB0->EUSART2:TX2;
    RC6PPS = 0x09;   //RC6->EUSART1:TX1;
    RX2PPS = 0x09;   //RB1->EUSART2:RX2;
}

void PIN_MANAGER_IOC(void)
{
}

/**
 End of File
*/
```

pin_manager.h

```
#ifndef PIN_MANAGER_H
#define PIN_MANAGER_H

#define INPUT   1
#define OUTPUT  0

#define HIGH    1
#define LOW     0

#define ANALOG      1
#define DIGITAL     0

#define PULL_UP_ENABLED     1
#define PULL_UP_DISABLED    0

// get/set resetsum aliases
#define resetsum_TRIS               TRISAbits.TRISA0
#define resetsum_LAT                LATAbits.LATA0
#define resetsum_PORT                PORTAbits.RA0
#define resetsum_WPU                 WPUAbits.WPUA0
#define resetsum_OD                 ODCONAbits.ODCA0
#define resetsum_ANS                ANSELAbits.ANSELA0
#define resetsum_SetHigh()          do { LATAbits.LATA0 = 1; } while(0)
#define resetsum_SetLow()           do { LATAbits.LATA0 = 0; } while(0)
#define resetsum_Toggle()           do { LATAbits.LATA0 = ~LATAbits.LATA0; } while(0)
#define resetsum_GetValue()          PORTAbits.RA0
#define resetsum_SetDigitalInput()    do { TRISAbits.TRISA0 = 1; } while(0)
#define resetsum_SetDigitalOutput()   do { TRISAbits.TRISA0 = 0; } while(0)
#define resetsum_SetPullup()         do { WPUAbits.WPUA0 = 1; } while(0)
#define resetsum_ResetPullup()       do { WPUAbits.WPUA0 = 0; } while(0)
#define resetsum_SetPushPull()       do { ODCONAbits.ODCA0 = 0; } while(0)
#define resetsum_SetOpenDrain()      do { ODCONAbits.ODCA0 = 1; } while(0)
#define resetsum_SetAnalogMode()      do { ANSELAbits.ANSELA0 = 1; } while(0)
#define resetsum_SetDigitalMode()     do { ANSELAbits.ANSELA0 = 0; } while(0)
```

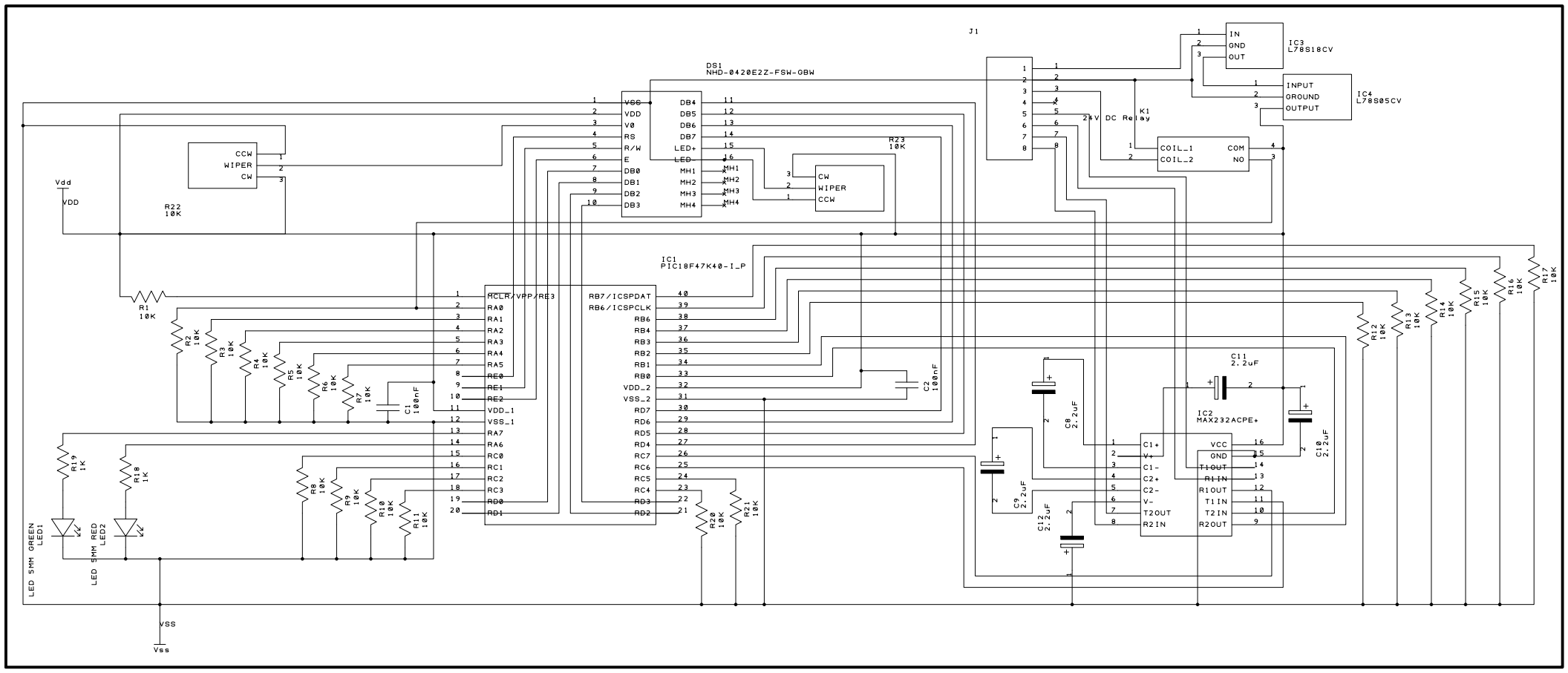
```
// get/set LTX aliases
#define LTX_TRIS                TRISAbits.TRISA4
#define LTX_LAT                 LATAbits.LATA4
#define LTX_PORT                PORTAbits.RA4
#define LTX_WPU                 WPUAbits.WPUA4
#define LTX_OD                  ODCONAbits.ODCA4
#define LTX_ANS                 ANSELAbits.ANSELA4
#define LTX_SetHigh()           do { LATAbits.LATA4 = 1; } while(0)
#define LTX_SetLow()            do { LATAbits.LATA4 = 0; } while(0)
#define LTX_Toggle()            do { LATAbits.LATA4 = ~LATAbits.LATA4; } while(0)
#define LTX_GetValue()          PORTAbits.RA4
#define LTX_SetDigitalInput()   do { TRISAbits.TRISA4 = 1; } while(0)
#define LTX_SetDigitalOutput()  do { TRISAbits.TRISA4 = 0; } while(0)
#define LTX_SetPullup()         do { WPUAbits.WPUA4 = 1; } while(0)
#define LTX_ResetPullup()       do { WPUAbits.WPUA4 = 0; } while(0)
#define LTX_SetPushPull()       do { ODCONAbits.ODCA4 = 0; } while(0)
#define LTX_SetOpenDrain()      do { ODCONAbits.ODCA4 = 1; } while(0)
#define LTX_SetAnalogMode()     do { ANSELAbits.ANSELA4 = 1; } while(0)
#define LTX_SetDigitalMode()    do { ANSELAbits.ANSELA4 = 0; } while(0)

// get/set LRX aliases
#define LRX_TRIS                TRISAbits.TRISA5
#define LRX_LAT                 LATAbits.LATA5
#define LRX_PORT                PORTAbits.RA5
#define LRX_WPU                 WPUAbits.WPUA5
#define LRX_OD                  ODCONAbits.ODCA5
#define LRX_ANS                 ANSELAbits.ANSELA5
#define LRX_SetHigh()           do { LATAbits.LATA5 = 1; } while(0)
#define LRX_SetLow()            do { LATAbits.LATA5 = 0; } while(0)
#define LRX_Toggle()            do { LATAbits.LATA5 = ~LATAbits.LATA5; } while(0)
#define LRX_GetValue()          PORTAbits.RA5
#define LRX_SetDigitalInput()   do { TRISAbits.TRISA5 = 1; } while(0)
#define LRX_SetDigitalOutput()  do { TRISAbits.TRISA5 = 0; } while(0)
#define LRX_SetPullup()         do { WPUAbits.WPUA5 = 1; } while(0)
#define LRX_ResetPullup()       do { WPUAbits.WPUA5 = 0; } while(0)
#define LRX_SetPushPull()       do { ODCONAbits.ODCA5 = 0; } while(0)
```

```
#define LRX_SetOpenDrain()      do { ODCONAbits.ODCA5 = 1; } while(0)
#define LRX_SetAnalogMode()     do { ANSELAbits.ANSELA5 = 1; } while(0)
#define LRX_SetDigitalMode()    do { ANSELAbits.ANSELA5 = 0; } while(0)

// get/set lederror aliases
#define lederror_TRIS               TRISAbits.TRISA6
#define lederror_LAT                LATAbits.LATA6
#define lederror_PORT               PORTAbits.RA6
#define lederror_WPU                WPUAbits.WPUA6
#define lederror_OD                 ODCONAbits.ODCA6
#define lederror_ANS                ANSELAbits.ANSELA6
#define lederror_SetHigh()          do { LATAbits.LATA6 = 1; } while(0)
#define lederror_SetLow()           do { LATAbits.LATA6 = 0; } while(0)
#define lederror_Toggle()           do { LATAbits.LATA6 = ~LATAbits.LATA6; } while(0)
#define lederror_GetValue()         PORTAbits.RA6
#define lederror_SetDigitalInput()    do { TRISAbits.TRISA6 = 1; } while(0)
#define lederror_SetDigitalOutput()   do { TRISAbits.TRISA6 = 0; } while(0)
#define lederror_SetPullup()        do { WPUAbits.WPUA6 = 1; } while(0)
#define lederror_ResetPullup()      do { WPUAbits.WPUA6 = 0; } while(0)
#define lederror_SetPushPull()      do { ODCONAbits.ODCA6 = 0; } while(0)
#define lederror_SetOpenDrain()     do { ODCONAbits.ODCA6 = 1; } while(0)
#define lederror_SetAnalogMode()    do { ANSELAbits.ANSELA6 = 1; } while(0)
#define lederror_SetDigitalMode()   do { ANSELAbits.ANSELA6 = 0; } while(0)

// get/set ledstatus aliases
#define ledstatus_TRIS              TRISAbits.TRISA7
#define ledstatus_LAT               LATAbits.LATA7
#define ledstatus_PORT              PORTAbits.RA7
#define ledstatus_WPU               WPUAbits.WPUA7
#define ledstatus_OD                ODCONAbits.ODCA7
#define ledstatus_ANS               ANSELAbits.ANSELA7
#define ledstatus_SetHigh()         do { LATAbits.LATA7 = 1; } while(0)
#define ledstatus_SetLow()          do { LATAbits.LATA7 = 0; } while(0)
#define ledstatus_Toggle()          do { LATAbits.LATA7 = ~LATAbits.LATA7; } while(0)
#define ledstatus_GetValue()        PORTAbits.RA7
#define ledstatus_SetDigitalInput()   do { TRISAbits.TRISA7 = 1; } while(0)
#define ledstatus_SetDigitalOutput()  do { TRISAbits.TRISA7 = 0; } while(0)
```

```
#define ledstatus_SetPullup()         do { WPUAbits.WPUA7 = 1; } while(0)
#define ledstatus_ResetPullup()       do { WPUAbits.WPUA7 = 0; } while(0)
#define ledstatus_SetPushPull()       do { ODCONAbits.ODCA7 = 0; } while(0)
#define ledstatus_SetOpenDrain()      do { ODCONAbits.ODCA7 = 1; } while(0)
#define ledstatus_SetAnalogMode()     do { ANSELAbits.ANSELA7 = 1; } while(0)
#define ledstatus_SetDigitalMode()    do { ANSELAbits.ANSELA7 = 0; } while(0)

// get/set RB0 procedures
#define RB0_SetHigh()          do { LATBbits.LATB0 = 1; } while(0)
#define RB0_SetLow()           do { LATBbits.LATB0 = 0; } while(0)
#define RB0_Toggle()           do { LATBbits.LATB0 = ~LATBbits.LATB0; } while(0)
#define RB0_GetValue()             PORTBbits.RB0
#define RB0_SetDigitalInput()    do { TRISBbits.TRISB0 = 1; } while(0)
#define RB0_SetDigitalOutput()   do { TRISBbits.TRISB0 = 0; } while(0)
#define RB0_SetPullup()          do { WPUBbits.WPUB0 = 1; } while(0)
#define RB0_ResetPullup()         do { WPUBbits.WPUB0 = 0; } while(0)
#define RB0_SetAnalogMode()       do { ANSELBbits.ANSELB0 = 1; } while(0)
#define RB0_SetDigitalMode()      do { ANSELBbits.ANSELB0 = 0; } while(0)

// get/set RB1 procedures
#define RB1_SetHigh()          do { LATBbits.LATB1 = 1; } while(0)
#define RB1_SetLow()           do { LATBbits.LATB1 = 0; } while(0)
#define RB1_Toggle()           do { LATBbits.LATB1 = ~LATBbits.LATB1; } while(0)
#define RB1_GetValue()             PORTBbits.RB1
#define RB1_SetDigitalInput()    do { TRISBbits.TRISB1 = 1; } while(0)
#define RB1_SetDigitalOutput()   do { TRISBbits.TRISB1 = 0; } while(0)
#define RB1_SetPullup()          do { WPUBbits.WPUB1 = 1; } while(0)
#define RB1_ResetPullup()         do { WPUBbits.WPUB1 = 0; } while(0)
#define RB1_SetAnalogMode()       do { ANSELBbits.ANSELB1 = 1; } while(0)
#define RB1_SetDigitalMode()      do { ANSELBbits.ANSELB1 = 0; } while(0)

// get/set RC6 procedures
#define RC6_SetHigh()          do { LATCbits.LATC6 = 1; } while(0)
#define RC6_SetLow()           do { LATCbits.LATC6 = 0; } while(0)
#define RC6_Toggle()           do { LATCbits.LATC6 = ~LATCbits.LATC6; } while(0)
#define RC6_GetValue()             PORTCbits.RC6
#define RC6_SetDigitalInput()    do { TRISCbits.TRISC6 = 1; } while(0)
```

```
#define RC6_SetDigitalOutput()   do { TRISCbits.TRISC6 = 0; } while(0)
#define RC6_SetPullup()          do { WPUCbits.WPUC6 = 1; } while(0)
#define RC6_ResetPullup()        do { WPUCbits.WPUC6 = 0; } while(0)
#define RC6_SetAnalogMode()      do { ANSELCbits.ANSELC6 = 1; } while(0)
#define RC6_SetDigitalMode()     do { ANSELCbits.ANSELC6 = 0; } while(0)

// get/set RC7 procedures
#define RC7_SetHigh()          do { LATCbits.LATC7 = 1; } while(0)
#define RC7_SetLow()           do { LATCbits.LATC7 = 0; } while(0)
#define RC7_Toggle()           do { LATCbits.LATC7 = ~LATCbits.LATC7; } while(0)
#define RC7_GetValue()           PORTCbits.RC7
#define RC7_SetDigitalInput()    do { TRISCbits.TRISC7 = 1; } while(0)
#define RC7_SetDigitalOutput()   do { TRISCbits.TRISC7 = 0; } while(0)
#define RC7_SetPullup()          do { WPUCbits.WPUC7 = 1; } while(0)
#define RC7_ResetPullup()        do { WPUCbits.WPUC7 = 0; } while(0)
#define RC7_SetAnalogMode()      do { ANSELCbits.ANSELC7 = 1; } while(0)
#define RC7_SetDigitalMode()     do { ANSELCbits.ANSELC7 = 0; } while(0)

void PIN_MANAGER_Initialize (void);
void PIN_MANAGER_IOC(void);

#endif // PIN_MANAGER_H

/**
 End of File
*/
```

ΠΑΡΑΡΤΗΜΑ Β – Σχηματικό


J1
DS1
NHD-0420E2Z-FSW-GBW
IC1
PIC18F47K40-I_P
IC2
MAX232ACPE+
IC3
L78S18CV
IC4
L78S05CV
K1
24V DC Relay
R23
10K
R22
10K
CCW
WIPER
CW
Vdd
VDD
Vss
VSS
LED 5MM GREEN
LED1
LED 5MM RED
LED2
MCLR/VPP/RE3
RA0
RA1
RA2
RA3
RA4
RA5
RE0
RE1
RE2
VDD_1
VSS_1
RA7
RA6
RC0
RC1
RC2
RC3
RD0
RD1
RB7/ICSPDAT
RB6/ICSPCLK
RB6
RB4
RB3
RB2
RB1
RB0
VDD_2
VSS_2
RD7
RD6
RD5
RD4
RC7
RC6
RC5
RC4
RD3
RD2
VSS
LED
RS
R/W
E
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
LED+
MH1
MH2
MH3
MH4
IN
GND
OUT
INPUT
GROUND
OUTPUT
COIL_1
COIL_2
COM
NO
C1+
V+
C1-
C2+
C2-
V-
T2OUT
R2IN
VCC
GND
T1OUT
R1IN
R1OUT
T1IN
T2IN
R2OUT
R1 10K
R2 10K
R3 10K
R4 10K
R5 10K
R6 10K
R7 10K
R8 10K
R9 10K
R10 10K
R11 10K
R12 10K
R13 10K
R14 10K
R15 10K
R16 10K
R17 10K
R18 1K
R19 1K
R20 10K
R21 10K
C1 100nF
C2 100nF
C8 2.2uF
C9 2.2uF
C10 2.2uF
C11 2.2uF
C12 2.2uF