ΑΛΕΞΑΝΔΡΕΙΟ Τ. Ε. Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή εργασία

Έλεγχος – Καταγραφή Περιβαλλοντολογικών Συνθηκών Μηχανογραφικού Κέντρου

Των φοιτητών:

Βαφειάδη - Σίνογλου Αλέξανδρου

Κόκκου Αθανάσιου

(022035 – 032172)

Επιβλέπων Καθηγητής:

Μπόζιος Ελευθέριος

Θεσσαλονίκη 2009

Πρόλογος

Στον 21^{ο} αιώνα, η πληροφορία ως έννοια, απέκτησε τεράστια σημασία και αξία για τους ανθρώπους, με αποτέλεσμα να αποτελεί πλέον, τον ακρογωνιαίο λίθο, της λειτουργίας όλων των οικονομικών δομών, σε όλες τις τεχνολογικά εξελιγμένες κοινωνίες. Η πληροφορία που παράγουν ή διαχειρίζονται σήμερα όλοι οι οργανισμοί και οι υποδομές που την υποστηρίζουν είναι αντικείμενα ζωτικής σημασίας για την λειτουργία τους. Η πληθώρα των υπολογιστικών συστημάτων και των τηλεπικοινωνιακών διατάξεων ενός δωματίου υπολογιστών, αποτελούν την καρδιά και τον εγκέφαλο των πληροφοριακών συστημάτων των οργανισμών και καθιστούν το δωμάτιο υπολογιστών τον πιο κρίσιμο παράγοντα για την εύρυθμη λειτουργία του μηχανογραφικού κέντρου.

Όσο οφείλουμε να ελέγχουμε και να προστατεύουμε την πληροφορία, άλλο τόσο οφείλουμε να προστατεύουμε τις υποδομές που την υποστηρίζουν, συμπεριλαμβάνοντας σ' αυτές από τους υπολογιστές και τις τηλεπικοινωνιακές συσκευές μέχρι και τη δομημένη καλωδίωση των δικτύων και τις συσκευές αδιάλειπτης παροχής ρεύματος. Καθοριστικός παράγοντας για την προστασία των υποδομών είναι και ο έλεγχος των περιβαλλοντολογικών παραμέτρων στο δωμάτιο διαχείρισης και διοίκησης των υπολογιστικών δικτύων, ούτως ώστε να μπορεί ο οργανισμός να ζει, δηλαδή να λειτουργεί απρόσκοπτα, να εξυπηρετεί τις ανάγκες του και να εξελίσσεται.

Για τον σκοπό της προστασίας του δωματίου διαχείρισης και διοίκησης των πληροφοριακών συστημάτων, απαιτείται η ενσωμάτωση στην λειτουργία του οργανισμού, ενός ολοκληρωμένου συστήματος παρακολούθησης των περιβαλλοντολογικών συνθηκών, που σε έκτακτες καταστάσεις θα προειδοποιεί τον διαχειριστή και θα παρέχει μηχανισμούς αποτροπής των απειλών καθώς και επανάκαμψης μετά από καταστροφή.

Στα πλαίσια της εργασίας πραγματοποιείται η σχεδίαση και ανάπτυξη ενός ολοκληρωμένου συστήματος επιτήρησης το οποίο παρακολουθεί τις περιβαλλοντολογικές συνθήκες του δωματίου υπολογιστών (Computer/Server Room) με σκοπό την άμεση παρέμβαση όταν αυτές δεν είναι κανονικές.

Το σύστημα διαθέτει τις ακόλουθες δυνατότητες:

- Παρακολούθηση και καταγραφή των περιβαλλοντολογικών συνθηκών
- Παρουσίαση των μετρήσεων σε πραγματικό χρόνο.
- Ειδοποίηση του διαχειριστή σε συνθήκες απειλής των υποδομών.
- Προβολή στατιστικών διαγραμμάτων των μετρήσεων και

- Αποστολή ημερήσιας αναφοράς.

Η παρακολούθηση των περιβαλλοντολογικών συνθηκών επιτυγχάνεται μέσω μιας διάταξης η οποία διαθέτει αισθητήρια όργανα τοποθετημένα σε επιλεγμένα σημεία του δωματίου υπολογιστών. Η συσκευή δέχεται τις μετρήσεις από τους αισθητήρες και τις αποστέλλει σε έναν ηλεκτρονικό υπολογιστή.

Ο υπολογιστής αναλαμβάνει την παραλαβή των μετρήσεων από την εξωτερική συσκευή, την παρουσίαση διαγραμματικά των τιμών αυτών, τον έλεγχο τους και τέλος, αν συντρέχουν συνθήκες απειλής των υποδομών, την ειδοποίηση του διαχειριστή με μήνυμα ηλεκτρονικού ταχυδρομείου (email) και γραπτού μηνύματος στο κινητό του τηλέφωνο (SMS).

Ο υπολογιστής, πέραν της βασικής του αρμοδιότητας, παρέχει τη δυνατότητα παρουσίασης στατιστικών διαγραμμάτων για όλες τις καταγεγραμμένες μετρήσεις. Επιπλέον αποστέλλει στο διαχειριστή ημερήσια αναφορά μέσω ηλεκτρονικού ταχυδρομείου η οποία περιέχει τα στατιστικά διαγράμματα των μετρήσεων σε αρχείο PDF (Portable Document Format).

Η εργασία περιλαμβάνει δέκα κεφάλαια και δύο παραρτήματα που διαρθρώνονται σε τέσσερις ενότητες. Στην Πρώτη Ενότητα (Κεφάλαια 1 έως 4) περιγράφεται ο σχεδιασμός και η αναπτυξη της εξωτερικής συσκευης. Στη Δεύτερη Ενότητα (Κεφάλαια 5 και 6) αναλύεται το σύστημα διαχείρισης βάσεων δεδομένων. Στην Τρίτη Ενότητα (Κεφάλαια 7 και 8) αναπτύσσεται η κύρια εφαρμογή και στην Τέταρτη Ενότητα (Κεφάλαια 9 και 10) αναπτύσσεται η διαδικτυακή εφαρμογή. Στα παραρτήματα περιλαμβάνονται τα δύο εγχειρίδια διάδρασης, στο πρώτο η εφαρμογή καταγραφής και ειδοποίησης και στο δεύτερο η διαδικτυακή εφαρμογή

Η παρούσα εργασία εκπονήθηκε από τον Ιανουάριο έως τον Σεπτέμβριο του 2009 με επιβλέποντα καθηγητή τον κ. Ελευθέριο Μπόζιο. Τον ευχαριστούμε ιδιαίτερα για τη θερμή συμπαράσταση και υποστήριξη που μας πρόσφερε σε όλη τη διάρκεια της συνεργασίας μας καθώς και για την επιστημονική καθοδήγηση για την ολοκλήρωση της πτυχιακής εργασίας.

Περίληψη

Στα πλαίσια της προστασίας των υποδομών ενός μηχανογραφικού κέντρου και κατά επέκταση της ίδιας της πληροφορίας που διαχειρίζεται, κρίνεται αναγκαία η σχεδίαση και ανάπτυξη ενός ολοκληρωμένου συστήματος επιτήρησης το οποίο θα παρακολουθεί τις περιβαλλοντολογικές συνθήκες του δωματίου υπολογιστών (Computer/Server Room) με σκοπό την άμεση παρέμβαση όταν αυτές δεν είναι κανονικές.

Το σύστημα θα υποστηρίζει τις εξής δυνατότητες:

- Παρακολούθηση των περιβαλλοντολογικών συνθηκών
- Παρουσίαση των μετρήσεων σε πραγματικό χρόνο
- Ειδοποίηση του διαχειριστή σε συνθήκες απειλής των υποδομών
- Καταγραφή των περιβαλλοντολογικών συνθηκών σε βάση δεδομένων
- Προβολή στατιστικών διαγραμμάτων των μετρήσεων
- Αποστολή ημερήσιας αναφοράς

Η παρακολούθηση των περιβαλλοντολογικών συνθηκών επιτυγχάνεται μέσω μιας διάταξης η οποία διαθέτει αισθητήρια όργανα τοποθετημένα στα κρίσιμα σημεία του δωματίου υπολογιστών. Η συσκευή λαμβάνει τις μετρήσεις από τους αισθητήρες και τις αποστέλλει σε έναν ηλεκτρονικό υπολογιστή.

Ο υπολογιστής αποτελεί το θεμέλιο λίθο του συστήματος. Αναλαμβάνει την παραλαβή των μετρήσεων από την εξωτερική συσκευή, την παρουσίαση διαγραμματικά των τιμών αυτών, τον έλεγχο τους και τέλος την ειδοποίηση του διαχειριστή με μήνυμα ηλεκτρονικού ταχυδρομείου (email) και γραπτού μηνύματος στο κινητό του τηλέφωνο (SMS) σε συνθήκες απειλής των υποδομών.

Ο υπολογιστής, πέραν της βασικής του αρμοδιότητας, δίνει τη δυνατότητα παρουσίασης στατιστικών διαγραμμάτων για όλες τις καταγεγραμμένες μετρήσεις κατά το παρελθόν. Επιπλέον αποστέλλει στο διαχειριστή ημερήσια αναφορά μέσω ηλεκτρονικού ταχυδρομείου η οποία περιλαμβάνει τα στατιστικά διαγράμματα των μετρήσεων σε ανεξάρτητα πλατφόρμας έγγραφα (Portable Document Format - PDF).

Summary

Concerning the protection of the infrastructures of a computer centre and by extension the protection of the so – called information it supports, it is regarded not just considerable but also necessary the designing and development of a complete system which will be able to monitor the environmental conditions of the Computer/Server Room in order to intervene on time when these conditions are not normal.

The system is going to have the following capabilities:

- Monitorship of the environmental conditions.
- Real time presentation of the measurements.
- Warning at the administrator that the infrastructures are under threatening conditions.
- Record of the environmental conditions into a database.
- Presentation of the statistical diagrams of the measurements.
- Daily report via email.

The monitorship of the environmental conditions is achieved through a device which has sensors positioned at crucial spots in the computer room. The device receives the measurements from the sensors and sends them to a personal computer.

The personal computer represents the base of the system. It undertakes the reception of the measurements from the external device, their presentation in form of diagrams, their check and finally the notification to the administrator via email and SMS when the infrastructures are under threatening situations.

The personal computer, except for its main “duty”, has the ability to present statistical diagrams for all the past-recorded measurements. Furthermore it sends to the administrator a daily report via email, which constitutes of the statistical diagrams of the measurements in Portable Document Format – PDF.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Εισαγωγή

Η ασφάλεια του δωματίου υπολογιστών καθορίζεται από δυο παράγοντες. Τον φυσικό και τον ανθρώπινο. Ο φυσικός παράγοντας περιλαμβάνει τους κινδύνους από φυσικά αίτια, όπως φωτιά, πλημμύρα, υψηλή θερμοκρασία, υγρασία, σεισμός κτλ, ενώ ο ανθρώπινος παράγοντας περιλαμβάνει την κλοπή, την βομβιστική ή εμπρηστική επίθεση, την ηλεκτρονική υφαρπαγή και τη δολιοφθορά. Το σύνολο αυτών των κινδύνων προσδιορίζει και τα συστήματα αποτροπής και επανάκαμψης που θα πρέπει να διαθέτει κάθε δωμάτιο υπολογιστών, εάν θέλει ο διαχειριστής να μην χάσει τον ύπνο του και την δουλειά του.

Ο Στόχος της Πτυχιακής Εργασίας

Ο στόχος της δικής μας υλοποίησης είναι η καταγραφή, η παρουσίαση σε πραγματικό χρόνο και η ειδοποίηση του διαχειριστή σε συνθήκες απειλής και κινδύνου των υποδομών του δωματίου υπολογιστών. Τη στιγμή που οι μετρήσεις των αισθητήρων θα έχουν ξεπεράσει τα αποδεκτά όρια που έχει ορίσει ο διαχειριστής, θα υπάρξει άμεση ενημέρωση του για κάθε έναν από τους αισθητήρες, μέσω μηνυμάτων ηλεκτρονικού ταχυδρομείου, αλλά και μηνυμάτων στο κινητό του τηλέφωνο. Επισημαίνεται ότι η παρούσα υλοποίηση δεν διαθέτει δυνατότητα ενεργοποίησης μηχανισμών αποτροπής κινδύνων που ενδέχεται να υπάρχουν στο δωμάτιο υπολογιστών, όπως π.χ. τερματισμό της λειτουργίας ενός υπολογιστή-διακομιστή που έχει ξεπεράσει το όριο θερμοκρασίας του, ενεργοποίηση του συστήματος πυρόσβεσης ή του κλιματισμού, ή την επανάκαμψη μετά από καταστροφή.

Η υλοποίηση του συστήματος μας, αποτελείται από τρία διαφορετικά μέρη:

I. Σχεδιασμό της συσκευής των αισθητήρων (συσκευή επιτήρησης) και προγραμματισμό του μικροελεγκτή της διάταξης σε γλώσσα μηχανής assembly . Το πρόγραμμα διαβάζει τις τιμές της τάσης (αναλογικό σήμα) από τους αισθητήρες, τις μετασχηματίζει σε ψηφιακή μορφή και στη

συνέχεια τις αποστέλλει μέσω της σειριακής σύνδεσης (RS-232) στον υπολογιστή.

II. Δημιουργία εφαρμογής α) για την καταγραφή των περιβαλλοντολογικών συνθηκών, β) την ειδοποίηση του διαχειριστή σε συνθήκες απειλής και γ) την ενημέρωσή του, μέσω ημερήσιας αναφοράς με τα στατιστικά των μετρήσεων. Η εφαρμογή τρέχει στον υπολογιστή όπου είναι συνδεδεμένη η συσκευή των αισθητήρων και δίνει τη δυνατότητα στον διαχειριστή α) να εκκινήσει τον διακομιστή Βάσεων Δεδομένων MySQL που χρησιμοποιείται στο σύστημα μας για την καταγραφή των μετρήσεων από τους αισθητήρες, β) να ορίσει τις παραμέτρους της σειριακής επικοινωνίας για την σύνδεση με την συσκευή, και γ) να ορίσει τις παραμέτρους της σειριακής επικοινωνίας για την σύνδεση με το κινητό τηλέφωνο (GSM Modem), μέσω του οποίου θα πραγματοποιηθεί η αποστολή μηνύματος (sms), στο κινητό τηλέφωνο του διαχειριστή σε συνθήκες κινδύνου.

III. Δημιουργία διαδικτυακής εφαρμογής η οποία δίνει τη δυνατότητα παρουσίασης σε πραγματικό χρόνο των μετρήσεων από τους αισθητήρες, αλλά και προβολής ημερήσιων, εβδομαδιαίων, μηνιαίων και ετήσιων στατιστικών διαγραμμάτων για τις καταγεγραμμένες μετρήσεις κάθε αισθητήρα. Μέσω της εφαρμογής αυτής, ο διαχειριστής μπορεί να παραμετροποιήσει το σύστημα ορίζοντας την περιγραφή της τοποθεσίας του κάθε αισθητήρα, μεταβάλλοντας τις τιμές-όρια και να αλλάξει τα μηνύματα ειδοποίησης που θα παραλάβει εάν ξεπεραστούν αυτά τα όρια, να καθορίσει το κινητό τηλέφωνο και τον λογαριασμό ηλεκτρονικού ταχυδρομείου στα οποία θα ειδοποιηθεί, καθώς και να τροποποιήσει τον λογαριασμό ηλεκτρονικού ταχυδρομείου που χρησιμοποιεί η ίδια η εφαρμογή, για να πραγματοποιήσει την ηλεκτρονική ενημέρωση.

Οι Απαιτήσεις του Συστήματος

Το λογισμικό καταγραφής των μετρήσεων και ειδοποίησης σε συνθήκες κινδύνου, αλλά και η διαδικτυακή εφαρμογή παρουσίασης των μετρήσεων και διαμόρφωσης των επιλογών του συστήματος, σχεδιάστηκε και υλοποιήθηκε στην γλώσσα προγραμματισμού java, πράγμα που καθιστά τις δύο αυτές

εφαρμογές ανεξάρτητες πλατφόρμας. Αυτό σημαίνει ότι μπορούν να χρησιμοποιηθούν σε οποιοδήποτε λειτουργικό σύστημα (Windows, Linux, Mac, Netware, Unix, κτλ) έχει εγκατεστημένη κάποια έκδοση του διερμηνευτή της java (java runtime environment), χωρίς να απαιτείται η επαναμεταγλώττιση του πηγαίου κώδικά μας.

Οι εξωτερικές βιβλιοθήκες ανοιχτού κώδικα (Open Source) που χρησιμοποιούν οι δυο εφαρμογές μας, ακόμα και ο οδηγός σύνδεσης με τη βάση δεδομένων MySQL, είναι ενσωματωμένες στις εφαρμογές και έτσι δεν απαιτείται ο χρήστης να τις προσθέσει στον κατάλογο των εξωτερικών βιβλιοθηκών του java runtime environment. Το μόνο που απαιτείται από μέρους του χρήστη, είναι να τοποθετήσει το αρχείο rxtxSerial.dll στον κατάλογο jre/bin/.

Το σύστημα μας, απαιτεί για την λειτουργία του την ύπαρξη σειριακής θύρας στον υπολογιστή για να συνδεθεί η εφαρμογή καταγραφής και ειδοποίησης με τη συσκευή επιτήρησης ώστε να παραλαμβάνει τα δεδομένα. Επίσης απαιτείται η ύπαρξη USB θύρας για να συνδεθεί το κινητό τηλέφωνο, μέσω του οποίου θα αποστείλει η εφαρμογή γραπτό μήνυμα στο κινητό τηλέφωνο του διαχειριστή όταν παραστεί ανάγκη. Εναλλακτικά η εφαρμογή μπορεί να επικοινωνήσει με το κινητό τηλέφωνο (GSM modem) μέσω Bluetooth εάν αυτό είναι διαθέσιμο και επιθυμητό, διότι η επικοινωνία ακολουθεί και στις δύο περιπτώσεις το σειριακό μοντέλο μετάδοσης δεδομένων.

Το πρόγραμμα για τον έλεγχο της συσκευής επιτήρησης έχει γραφεί σε γλώσσα μηχανής assembly για το μικροελεγκτή PIC16F877-A της MICROCHIP και χρειάζεται να φορτωθεί μέσω του υπολογιστή στη μνήμη του μικροελεγκτή μέσω κάποιας κατάλληλης διασύνδεσης.

Η δομή της τεκμηρίωσης

Η οργάνωση της πτυχιακής εργασίας βασίζεται σε τέσσερις κύριες θεματικές ενότητες. Η πρώτη, η τρίτη και η τέταρτη ενότητα αναφέρονται στα τρία μέρη τα οποία αποτελούν την υλοποίηση του συστήματός μας, ενώ η

δεύτερη ενότητα αφιερώθηκε αποκλειστικά στο διακομιστή βάσεων δεδομένων MySQL.

Η **πρώτη ενότητα** διαπραγματεύεται το σχεδιασμό και την ανάπτυξη της εξωτερικής συσκευής (συσκευή επιτήρησης) και περιλαμβάνει τέσσερα κεφάλαια. Το πρώτο κεφάλαιο περιέχει αρχικά μία εισαγωγή στους μικροελεγκτές και συνεχίζεται με μία πιο λεπτομερή περιγραφή της οικογένειας των μικροελεγκτών την οποία και χρησιμοποιήσαμε στο σύστημά μας (PIC Microchip). Στο κεφάλαιο αυτό αναλύεται η αρχιτεκτονική του μικροελεγκτή, η οργάνωση της μνήμης του, η διεπαφή των εισόδων και εξόδων του και η γενικότερη λειτουργία του. Το δεύτερο κεφάλαιο αφορά στο περιβάλλον ανάπτυξης λογισμικού για τον προγραμματισμό του μικροελεγκτή. Το περιβάλλον προγραμματισμού στο οποίο εργαστήκαμε, είναι το MPLAB IDE v.8.20a της Microchip το οποίο και διατίθεται δωρεάν στην ιστοσελίδα της[1]. Το περιβάλλον αυτό δίνει τη δυνατότητα μεταγλώττισης, αποσφαλμάτωσης και φόρτωσης του προγράμματος στη μνήμη του μικροελεγκτή με τη βοήθεια πρόσθετου εξοπλισμού. Για τη δημιουργία του προγράμματος επιλέξαμε την γλώσσα μηχανής assembly και όχι κάποια γλώσσα υψηλότερου επιπέδου. Επεξηγείται ο γενικός τρόπος λειτουργίας της γλώσσας και η αλληλεπίδρασή της με το υλικό. Το τρίτο κεφάλαιο αφορά στη σχεδίαση και προσομοίωση σε ηλεκτρονικό υπολογιστή του κυκλώματος της συσκευής με τους αισθητήρες (συσκευής επιτήρησης). Αυτό επιτυγχάνεται με τη βοήθεια του προγράμματος μοντελοποίησης και εικονικής προσομοίωσης ηλεκτρονικών κυκλωμάτων Proteus VSM της Labcenter Electronics. Μια περιορισμένη έκδοση του προγράμματος διατίθεται στην ιστοσελίδα της εταιρείας[2]. Η πρώτη ενότητα ολοκληρώνεται με το τέταρτο κεφάλαιο στο οποίο παρουσιάζεται η υλοποίηση της συσκευής στο Proteus. Εδώ αναλύεται βήμα προς βήμα η διαδικασία παραλαβής των μετρήσεων από τους αισθητήρες και η αποστολή τους στον ηλεκτρονικό υπολογιστή.

---

1 http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002#P186_6049

2 http://www.labcenter.co.uk/index.cfm

Η **δεύτερη ενότητα** περιλαμβάνει δύο κεφάλαια και διαπραγματεύεται το Σύστημα Διαχείρισης Βάσεων Δεδομένων που χρησιμοποιείται στο σύστημά μας. Η βάση δεδομένων αξιοποιείται στην αποθήκευση των ρυθμίσεων που αφορούν στο σύστημα και για την καταγραφή των μετρήσεων των αισθητήρων. Αποτελεί επίσης τον συνδετικό κρίκο ανάμεσα στην κύρια και στη διαδικτυακή εφαρμογή του συστήματος, αφού αυτή χρησιμοποιείται ως όχημα ανταλλαγής πληροφοριών ανάμεσά τους. Έτσι, στο πέμπτο κεφάλαιο αναλύεται η λειτουργία του διακομιστή βάσεων δεδομένων MySQL 6 και περιγράφεται η παραμετροποίησή του. Το έκτο κεφάλαιο περιλαμβάνει την υλοποίηση της βάσης δεδομένων “sensingsystem” και των πινάκων της που χρησιμοποιούν οι δύο εφαρμογές του συστήματος για την καταγραφή και παρουσίαση των μετρήσεων αλλά και για την παραμετροποίηση του όλου συστήματος.

Η **τρίτη ενότητα** διαπραγματεύεται την κύρια εφαρμογή του συστήματος μας για την παραλαβή και την καταγραφή των μετρήσεων από την εξωτερική συσκευή, τον έλεγχο τους και την ειδοποίηση του διαχειριστή σε συνθήκες κινδύνου για τις υποδομές, καθώς και τις τεχνολογίες που αυτό χρησιμοποιεί για να υποστηρίξει τις λειτουργίες του. Το έβδομο κεφάλαιο αφορά στις τεχνολογίες που χρησιμοποιήσαμε για να εξασφαλίσουμε τη λειτουργικότητα της εφαρμογής. Περιλαμβάνει, δηλαδή, το περιβάλλον ανάπτυξης του λογισμικού στη γλώσσα προγραμματισμού Java, τις εξωτερικές βιβλιοθήκες ανοικτού κώδικα που χρησιμοποιεί το πρόγραμμα, ενώ το όγδοο κεφάλαιο ολοκληρώνει την τρίτη ενότητα με την υλοποίηση και παρουσίαση της κύριας εφαρμογής του συστήματος μας.

Η **τέταρτη ενότητα** περιλαμβάνει δύο κεφάλαια και διαπραγματεύεται τη διαδικτυακή εφαρμογή, η οποία αναλαμβάνει την παρουσίαση των καταγεγραμμένων μετρήσεων σε πραγματικό χρόνο, την προβολή στατιστικών διαγραμμάτων για αυτές και τη διαμόρφωση των ρυθμίσεων του συστήματος. Στο ένατο κεφάλαιο περιγράφονται οι τεχνολογίες που χρησιμοποιούμε για τη δημιουργία και υποστήριξη των βασικών λειτουργιών της. Η ενότητα ολοκληρώνεται με το δέκατο κεφάλαιο παρουσιάζοντας την υλοποίηση της διαδικτυακής εφαρμογής και των λειτουργιών της.

Στο τέλος, παρατίθεται η **βιβλιογραφία** και τα δύο **παραρτήματα** της τεκμηρίωσης που περιλαμβάνουν τα εγχειρίδια χρήσης των εφαρμογών με σχηματικές αναπαραστάσεις της αλληλεπίδρασης.

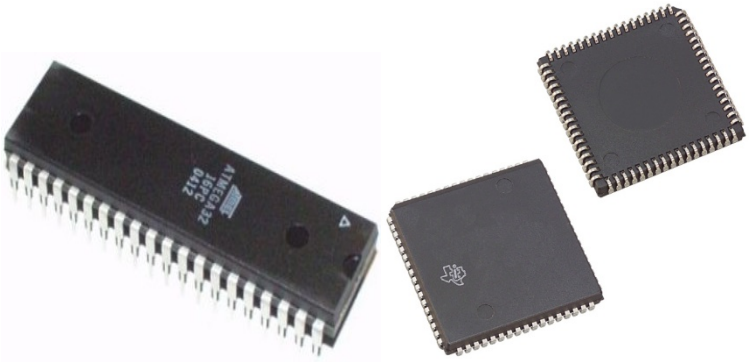
Πρώτη Ενότητα -

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ΤΗΣ ΕΞΩΤΕΡΙΚΗΣ ΣΥΣΚΕΥΗΣ

ΚΕΦΑΛΑΙΟ 1
Οι Μικροελεγκτές και οι Αισθητήρες

1.1. - Εισαγωγή στους μικροελεγκτές

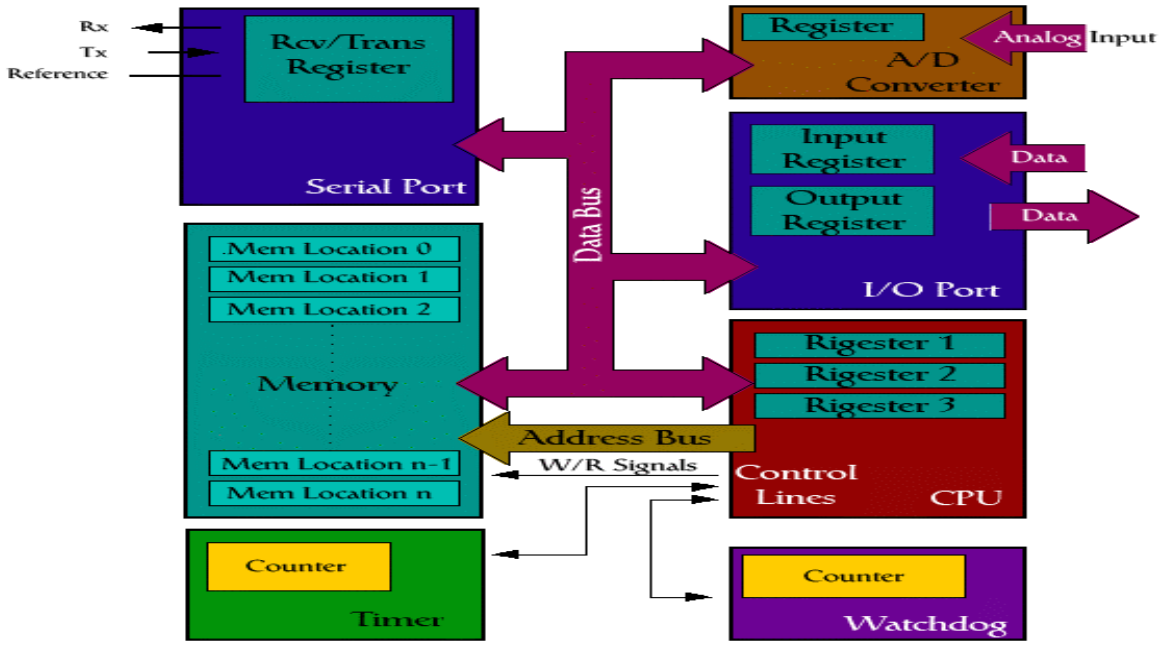
Ο μικροελεγκτής είναι ένα ολοκληρωμένο ηλεκτρονικό κύκλωμα (*Σχήμα 1.1*) που χωρίς το λογισμικό δεν είναι σε θέση να κάνει οτιδήποτε. Όταν όμως ένας μικροελεγκτής ελέγχεται από κάποιο λογισμικό, οι πρακτικές εφαρμογές που μπορεί να χρησιμοποιηθεί είναι θεωρητικά απεριόριστες. Οι μικροελεγκτές, είναι από τις μεγαλύτερες εφευρέσεις μετά την δημιουργία των μικροεπεξεργαστών. Είναι ένας ολοκληρωμένος αυτοδύναμος υπολογιστής μέσα σ' ένα μοναδικό ολοκληρωμένο κύκλωμα, ο οποίος έχει τη δυνατότητα αποθήκευσης προγραμμάτων και εκτέλεσης σύνθετων μαθηματικών πράξεων.



Σχήμα 1.1 - Μικροελεγκτές

Όπως κάθε υπολογιστικό σύστημα, ο μικροελεγκτής περιέχει τον επεξεργαστή του, το σύστημα εισόδου/εξόδου, την μνήμη και τον ταλαντωτή και όλα αυτά μέσα στο ίδιο chip, σε ένα μικρό κομμάτι πυριτίου. Αυτό εξασφαλίζει σημαντική μείωση χώρου, εξοικονόμηση του χρόνου σχεδίασης, απουσία ανάγκης εξωτερικού κυκλώματος χρονισμού, αποφυγή προβλημάτων συμβατότητας, αλλά σε ορισμένες περιπτώσεις παρουσιάζονται περιορισμοί στην ποσότητα μνήμης και στις δυνατότητες εισόδου και εξόδου.

Οι μικροελεγκτές έχουν ένα ευρύ φάσμα εφαρμογών και χρησιμοποιούνται σε πληθώρα ψηφιακών συσκευών και διατάξεων - π.χ. συσκευές αναπαραγωγής εικόνας και ήχου, κινητά τηλέφωνα, ψηφιακές φωτογραφικές μηχανές, οχήματα πάσης φύσης κ.α. - και έτσι κυκλοφορούν πολλά διαφορετικά μοντέλα, με διαφορετικές αρχιτεκτονικές και δυνατότητες, που εξυπηρετούν και διαφορετικές απαιτήσεις. Για την επιλογή του κατάλληλου μικροελεγκτή θα πρέπει να εξετασθούν οι απαιτήσεις και οι ανάγκες της εφαρμογή μας. Τα χαρακτηριστικά του μικροελεγκτή - η συχνότητα ρολογιού του, ο χρόνος εκτέλεσης των εντολών, η χωρητικότητα της μνήμης προγράμματος, ο αριθμός των θυρών εισόδου και εξόδου που διαθέτει και ο τύπος τους (αναλογικές – ψηφιακές), εάν θα υποστηρίζει κάποιου τύπου σειριακής επικοινωνίας, εάν θα διαθέτει μετατροπέα αναλογικού σήματος σε ψηφιακό και πολλά άλλα χαρακτηριστικά - θα εξαρτηθούν από τις απαιτήσεις του συστήματος το οποίο θα υποστηρίξουν. Ένας υποτυπώδης μικροελεγκτής μπορεί να περιγραφεί σχηματικά με την παρακάτω εικόνα (*Σχήμα 1.2*).



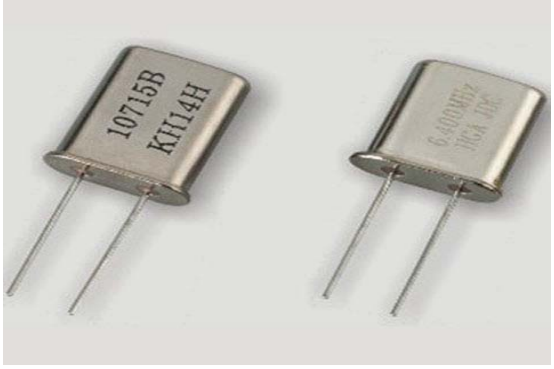
Σχήμα 1.2 – Η εσωτερική δομή ενός μικροελεγκτή PIC

Θα πρέπει να αναφέρουμε επίσης ότι και οι μικροελεγκτές, όπως άλλωστε και όλα τα υπολογιστικά συστήματα, λειτουργούν με βάση τη δυαδική λογική, όπου με τη βοήθεια δύο μόνο λογικών καταστάσεων (λογικό 0 και λογικό 1), που αναπαρίστανται με επίπεδα ηλεκτρικών τάσεων, εκτελούν όλες τις αριθμητικές πράξεις. Έτσι οι πληροφορίες που ανταλλάσσει ο επεξεργαστής με τη μνήμη και

τις μονάδες εισόδου/εξόδου είναι συνδυασμοί δυαδικών ψηφίων. Η μεταφορά της δυαδικής πληροφορίας ανάμεσα στις διάφορες μονάδες του μικροελεγκτή γίνεται παράλληλα από ένα σύνολο γραμμών, που αναφέρονται σαν *γραμμές δεδομένων* (data lines). Το σύνολο αυτών των γραμμών αποτελεί το *δίαυλο δεδομένων* (data bus).

Ο δίαυλος δεδομένων δεν λύνει όλα τα προβλήματα μεταφοράς της πληροφορίας. Ο επεξεργαστής θα πρέπει να έχει τη δυνατότητα επιλογής της μονάδας, με την οποία θα επικοινωνήσει, ώστε να μπορεί να την ειδοποιήσει ότι θα της στείλει ή ότι θα πάρει δεδομένα από αυτή. Για το λόγο αυτό χρησιμοποιεί δύο ακόμα διαύλους, το *δίαυλο διευθύνσεων* (address bus) και το *δίαυλο ελέγχου* (control bus). Οι γραμμές των διαύλων αυτών λέγονται αντίστοιχα *γραμμές διευθύνσεων* (address lines) και *γραμμές ελέγχου* (control lines). Με τις γραμμές διευθύνσεων ο επεξεργαστής στέλνει τη δυαδική διεύθυνση της θέσης μνήμης ή της μονάδας εισόδου/εξόδου, με την οποία θέλει να επικοινωνήσει, και με τις γραμμές ελέγχου τα κατάλληλα ηλεκτρικά σήματα για την ενεργοποίηση των επιθυμητών λειτουργιών της μνήμης ή των μονάδων εισόδου/εξόδου.

Η απαίτηση της ενεργοποίησης στοιχειωδών λειτουργιών σε προκαθορισμένα χρονικά διαστήματα, δημιουργεί την ανάγκη ύπαρξης μιας βάσης χρόνου, που αναφέρεται ως *κύκλωμα χρονισμού* (clock). Το κύκλωμα χρονισμού αποτελείται συνήθως από ένα κρυσταλλικό ταλαντωτή, που παράγει τετραγωνικούς παλμούς σταθερής συχνότητας χρησιμοποιώντας κατάλληλο κρύσταλλο χαλαζία (quartz) (*Σχήμα 1.3*). Η συχνότητα αυτή του ταλαντωτή καθορίζει και τη συχνότητα λειτουργίας του επεξεργαστή.



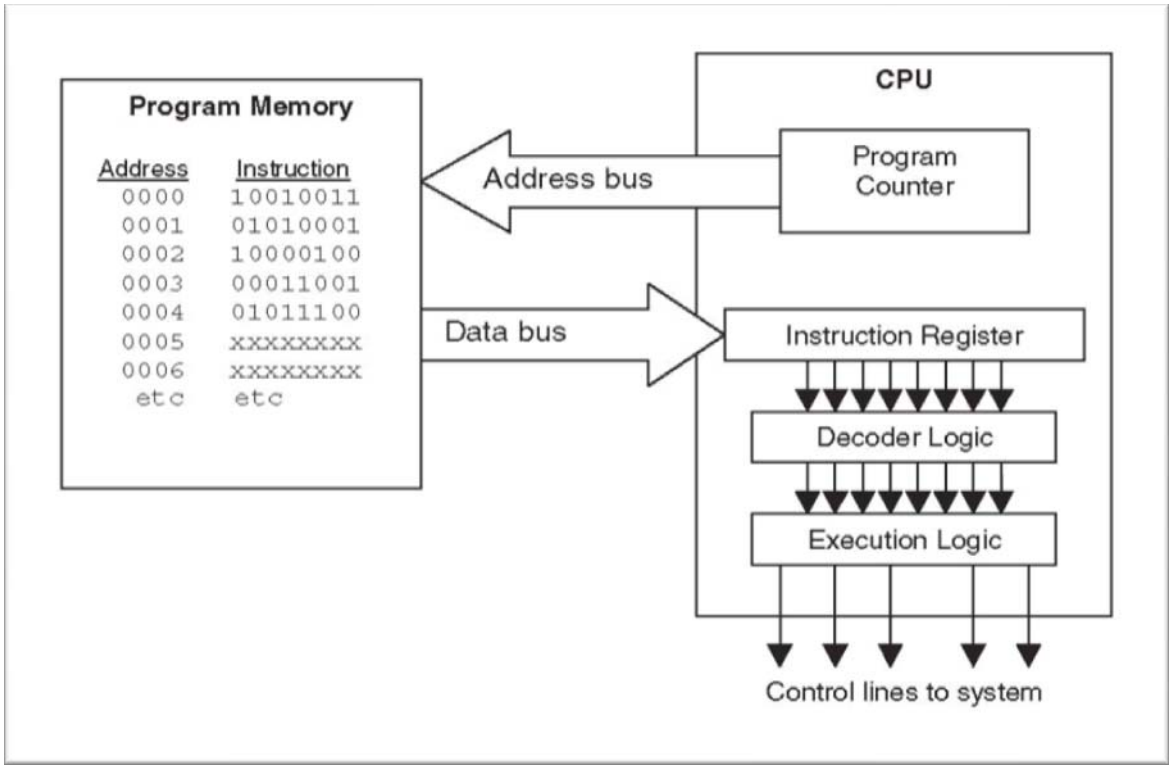
Σχήμα 1.3 – Κρυσταλλικός ταλαντωτής

Σε κάθε υπολογιστή η *κύρια μνήμη* αποτελείται από ένα σύνολο θέσεων, κάθε μία από τις οποίες περιλαμβάνει ένα ή περισσότερα αποθηκευτικά κύτταρα

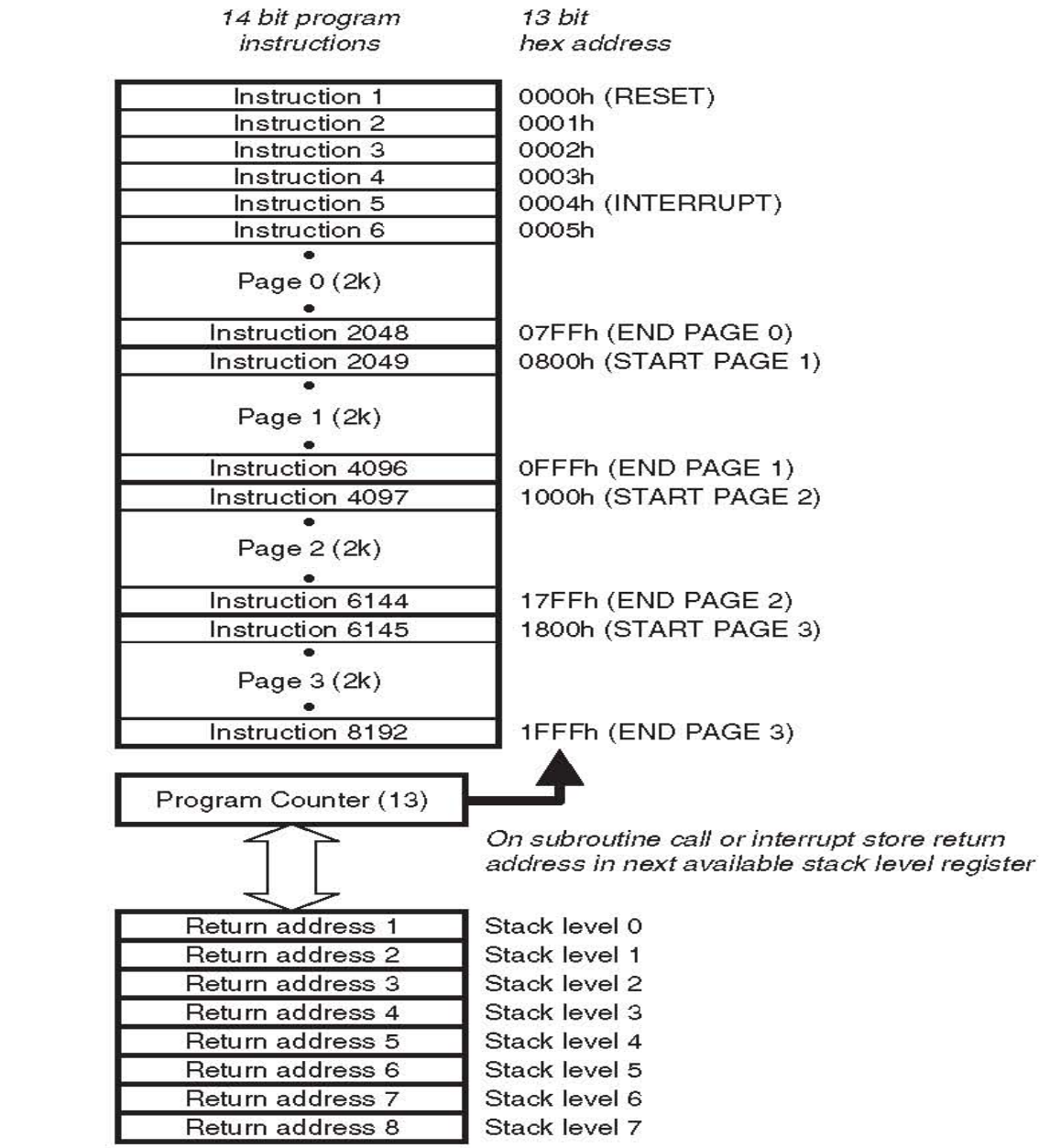
(storage cells) ικανά να αποθηκεύσουν πληροφορία ενός δυαδικού ψηφίου 0 ή 1 (binary digit – bit). Η διάκριση μεταξύ δύο θέσεων γίνεται από ένα χαρακτηριστικό αριθμό, που ονομάζεται διεύθυνση. Η αρίθμηση των θέσεων μνήμης αρχίζει από τον αριθμό 0 και είναι συνεχής. Ο αριθμός, που αντιστοιχεί στην τελευταία θέση μνήμης, το πλήθος δηλαδή των θέσεων μνήμης, διαφέρει από σύστημα σε σύστημα και εξαρτάται αποκλειστικά από την κατασκευή της κεντρικής μονάδας επεξεργασίας και τον τρόπο σύνδεσής της με την κύρια μνήμη. Στην περίπτωση που κάθε θέση μνήμης περιέχει ένα μόνο αποθηκευτικό κύτταρο, λέμε ότι είναι οργανωμένη σε δυαδικά ψηφία, ενώ, όταν περιέχει περισσότερα από ένα αποθηκευτικά κύτταρα, λέμε ότι είναι οργανωμένη σε λέξεις.

Το πλήθος των αποθηκευτικών κυττάρων της λέξης είναι δύναμη του δύο. Μια ομάδα από 8 δυαδικά ψηφία ονομάζεται ψηφιολέξη (byte). Συνήθως η ψηφιολέξη χωρίζεται σε δύο ομάδες των 4 δυαδικών ψηφίων, καθεμία από τις οποίες ονομάζεται ημι-ψηφιολέξη (nibble). Η πρώτη από τα αριστερά συμβολίζεται με NBH (Nibble Byte High) και η δεύτερη με NBL (Nibble Byte Low).

Όπως είπαμε και νωρίτερα, ο μικροεπεξεργαστής δέχεται και εκτελεί οδηγίες. Οι οδηγίες αυτές βρίσκονται γραμμένες ως ένα πρόγραμμα που βρίσκεται αποθηκευμένο στη μνήμη. Οι εντολές μεταφέρονται από τον δίαυλο δεδομένων στον καταχωρητή εντολών και πηγαίνουν προς αποκωδικοποίηση και εκτέλεση (Σχήμα 1.4).



Σχήμα 1.4.α – Εκτέλεση εντολών προγράμματος



Σχήμα 1.4.b – Εκτέλεση εντολών προγράμματος

Ο μικροελεγκτής που επιλέχθηκε ως κατάλληλος για τις απαιτήσεις του συστήματος επιτήρησης είναι ο PIC16F877 της Microchip. Αρχικά προσανατολιστήκαμε στην οικογένεια μικροελεγκτών PIC διότι θεωρείται φιλική για τους αρχάριους και επιλέξαμε το συγκεκριμένο μικροελεγκτή καθώς ενσωματώνει αναλογικές εισόδους και σειριακή διεπαφή για σύνδεση με ηλεκτρονικό υπολογιστή. Έτσι μειώσαμε τις απαιτήσεις για επιπλέον περιφερειακά εξαρτήματα.

1.2. - Ο μικροελεγκτής PIC16F877

Επεξεργαστής Υψηλών επιδόσεων Αρχιτεκτονικής RISC:

- Διαθέτει μόνο 35 εντολές μιας λέξης (single-word) που πρέπει να μάθουμε για να τον προγραμματίσουμε
- Όλες οι εντολές είναι ενός απλού κύκλου, εκτός αυτών που χρησιμοποιούνται για την αλλαγή ροής του το προγράμματος, που απαιτούν δύο κύκλους
- Ταχύτητα Υπολογισμού: Με συχνότητα 20 MHz – 200 ns περίοδο κύκλου
- Μνήμη προγράμματος (Flash) με μέγεθος που φτάνει τις 8192 θέσεις των 14 bits, μνήμη δεδομένων (RAM) με μέγεθος που φτάνει τις 368 θέσεις των 8 bits, EEPROM για αποθήκευση δεδομένων
- Σχήμα Ακροδεκτών συμβατό με αυτό άλλων μικροελεγκτών με 28 ή 40/44 ακροδέκτες (PIC16CXXX και PIC16FXXX)

Χαρακτηριστικά περιφερειακών:

- Timer0: 8-bit χρονόμετρο/μετρητής με προδιαιρέτη (prescaler) 8-bit
- Timer1: 16-bit χρονόμετρο/μετρητής με προδιαιρέτη, ο οποίος μπορεί να προσαυξηθεί σε κατάσταση ύπνωσης με τη βοήθεια εξωτερικού ταλαντωτή/παλμού
- Timer2: 8-bit χρονόμετρο/μετρητής με καταχωρητή περιόδου (period register), προδιαιρέτη και μεταδιαιρέτη (postscaler)
- Σύγχρονη σειριακή θύρα (Synchronous Serial Port - SSP) με SPI™ (Master mode) και I2C™ (Master/Slave) πρωτόκολλα
- Σύγχρονη/Ασύγχρονη σειριακή θύρα (Universal Synchronous Asynchronous Receiver Transmitter - USART/SCI) με ανίχνευση διευθύνσεων των 9-bit
- Παράλληλη θύρα (Parallel Slave Port - PSP) πλάτους 8 bits με εξωτερικά RD, WR και CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Αναλογικά Χαρακτηριστικά:

- Ενσωματωμένος Μετατροπέας σήματος από αναλογικό σε ψηφιακό, ανάλυσης των 10 bits, μέχρι 8 εισόδων
- Brown-out Reset (BOR)
- Μονάδα αναλογικού συγκριτή με:
  - Δύο αναλογικούς συγκριτές
  - Ενσωματωμένη μονάδα για τον καθορισμό της τάσης αναφοράς - προγραμματιστικά

- Ενσωματωμένος πολυπλέκτης εισόδων με καθορισμό προγραμματιστικά της κάθε εισόδου και ενσωματωμένη τάση αναφοράς
- Εξωτερικά προσβάσιμες έξοδοι των συγκριτών

Πρόσθετα χαρακτηριστικά του μικροελεγκτή:

- 100,000 κύκλοι διαγραφών/εγγραφών της μνήμης Flash
- 1,000,000 κύκλοι αναγνώσεων/εγγραφών δεδομένων της μνήμης EEPROM
- Διατήρηση των δεδομένων στη μνήμη EEPROM > 40 χρόνια
- Σειριακός προγραμματισμός εντός συστήματος (In-Circuit Serial Programming™ - ICSP™), μέσω δύο ακροδεκτών
- Χρονομετρητής επαγρύπνησης - Επιτηρητής ο οποίος ενσωματώνει ταλαντωτή, με χρήση πυκνωτών και αντιστάσεων, για μεγαλύτερη αξιοπιστία
- Προστασία (κλείδωμα) κώδικα προγράμματος
- Κατάσταση ύπνωσης για εξοικονόμηση ενέργειας
- Δυνατότητα επιλογής εξωτερικού ταλαντωτή
- Αποσφαλμάτωση του προγράμματος, ενώ ο μικροελεγκτής βρίσκεται εντός του συστήματος, μέσω δύο ακροδεκτών.

Τεχνολογία CMOS:

- Χαμηλής κατανάλωσης, υψηλής ταχύτητας τεχνολογία Flash/EEPROM
- Πλήρης στατική σχεδίαση
- Μεγάλο εύρος τάσης λειτουργίας (2.0V έως 5.5V)
- Εύρος θερμοκρασίας – για εμπορική και βιομηχανική χρήση
- Χαμηλή κατανάλωση ενέργειας

Μνήμη Προγράμματος: 14.3 Kbytes μίας λέξης εντολές: 8192 θέσεις
Μνήμη Δεδομένων: Στατική RAM: 368 bytes
EEPROM: 256 bytes
I\O: 33
10-BIT A/D: 8 είσοδοι
USART: Ναι
TIMERS: 8/16-BIT: 2/1

Ακροδέκτες του PIC16F877

Οι περισσότεροι ακροδέκτες χρησιμοποιούνται είτε ως είσοδοι είτε ως έξοδοι (*Σχήμα 1.5*). Συνολικά αριθμούνται πέντε θύρες: A(5), B(8), C(8), D(8) και E(3), άρα συνολικά 32 ακροδέκτες εισόδου/εξόδου (I/O pins). Όλες τους μπορούν να λειτουργήσουν ως ψηφιακοί ακροδέκτες εισόδου ή εξόδου, αλλά οι περισσότερες διαθέτουν περισσότερες της μιας λειτουργίες. Η επιθυμητή λειτουργία επιλέγεται κατά την αρχικοποίηση των καταχωρητών ελέγχου του μικροελεγκτή. Η πόρτα A και η E χρησιμοποιούνται προεπιλεγμένα ως αναλογικοί.

Η πόρτα B χρησιμοποιείται για τη φόρτωση του προγράμματος στην μνήμη flash του μικροελεγκτή (RB6 και RB7). Η πόρτα C μας παρέχει τη πρόσβαση στους timers και τις σειριακές πόρτες του μικροελεγκτή, ενώ η πόρτα D, μπορεί να χρησιμοποιηθεί ως slave θύρα.

Reset = 0, Run = 1	**MCLR**	1	40	**RB7**	Port B, Bit 7 (Prog. Data, Interrupt)
Port A, Bit 0 (Analogue AN0)	**RA0**	2	39	**RB6**	Port B, Bit 6 (Prog. Clock, Interrupt))
Port A, Bit 1 (Analogue AN1)	**RA1**	3	38	**RB5**	Port B, Bit 5 (Interrupt)
Port A, Bit 2 (Analogue AN2)	**RA2**	4	37	**RB4**	Port B, Bit 4 (Interrupt)
Port A, Bit 3 (Analogue AN3)	**RA3**	5	36	**RB3**	Port B, Bit 3 (LV Program)
Port A, Bit 4 (Timer 0)	**RA4**	6	35	**RB2**	Port B, Bit 2
Port A, Bit 5 (Analogue AN4)	**RA5**	7	34	**RB1**	Port B, Bit 1
Port E, Bit 0 (AN5, Slave control)	**RE0**	8	33	**RB0**	Port B, Bit 0 (Interrupt)
Port E, Bit 1 (AN6, Slave control)	**RE1**	9	32	**V_{DD}**	+5V Power Supply
Port E, Bit 2 (AN7, Slave control)	**RE2**	10	31	**Vss**	0V Power Supply
+5V Power Supply	**V_{DD}**	11	30	**RD7**	Port D, Bit 7 (Slave Port)
0V Power Supply	**Vss**	12	29	**RD6**	Port D, Bit 6 (Slave Port)
(CR clock) XTAL circuit	**CLKIN**	13	28	**RD5**	Port D, Bit 5 (Slave Port)
XTAL circuit	**CLKOUT**	14	27	**RD4**	Port D, Bit 4 (Slave Port)
Port C, Bit 0 (Timer 1)	**RC0**	15	26	**RC7**	Port C, Bit 7 (Serial Ports)
Port C, Bit 1 (Timer 1)	**RC1**	16	25	**RC6**	Port C, Bit 6 (Serial Ports)
Port C, Bit 2 (Timer 1)	**RC2**	17	24	**RC5**	Port C, Bit 5 (Serial Ports)
Port C, Bit 3 (Serial Clocks)	**RC3**	18	23	**RC4**	Port C, Bit 4 (Serial Ports)
Port D, Bit 0 (Slave Port)	**RD0**	19	22	**RD3**	Port D, Bit 3 (Slave Port)
Port D, Bit 1 (Slave Port)	**RD1**	20	21	**RD2**	Port D, Bit 2 (Slave Port)

Σχήμα 1.5 – Σχήμα ακροδεκτών PIC16F877

Ο μικροελεγκτής διαθέτει δυο ζευγάρια ακροδεκτών τροφοδοσίας (*V*DD_5V nominal and *V*ss_0V) και μπορεί να χρησιμοποιηθεί είτε το ένα, είτε το άλλο. Ο μικροελεγκτής μπορεί να λειτουργήσει ακόμα και με 2V. Μπορούμε να συνδέσουμε στους ακροδέκτες CLKIN, ένα χαμηλής συχνότητας κύκλωμα ρολογιού που θα αποτελείται από μία αντίσταση και έναν πυκνωτή, είτε να συνδέσουμε ένα κρυσταλλικό ταλαντωτή στους ακροδέκτες CLKIN και CLKOUT. Ο ακροδέκτης

MCLR αποτελεί την είσοδο του μηδενισμού (RESET). Όταν γίνεται 0, ο επεξεργαστής (MCU), σταματάει και επανεκκινεί όταν γίνεται 1. Η είσοδος στο MCLR πρέπει να είναι 1, αν δεν συνδέσουμε εξωτερικό κύκλωμα μηδενισμού, ώστε να λειτουργεί ο μικροελεγκτής. Είναι όμως καλή πρακτική να ενσωματώνουμε ένα διακόπτη για χειροκίνητο μηδενισμό στις περισσότερες, αν όχι σε όλες, τις εφαρμογές.

1.2.1. - Ο Επεξεργαστής και η Αρχιτεκτονική του

Η καρδιά του επεξεργαστή, είναι φυσικά, η Αριθμητική και Λογική Μονάδα (ALU) που περιέχει μονάχα έναν καταχωρητή, τον W, ή Working Register. Ο επεξεργαστής του PIC, διαφέρει από άλλους μικροεπεξεργαστές στον αριθμό των καταχωρητών που χρησιμοποιεί για τις αριθμητικές και λογικές πράξεις. Ο καταχωρητής W είναι 8-bits και χειρίζεται όλα τα δεδομένα μέσα στον επεξεργαστή. Μέσα στον επεξεργαστή βρίσκονται και οι καταχωρητές δεδομένων, οι οποίοι χωρίζονται σε δύο κατηγορίες: αυτούς που αφορούν τις διαδικασίες εισόδου/εξόδου και σε αυτούς που χρησιμοποιούνται σαν την μνήμη RAM του χρήστη. Να αναφέρουμε επίσης ότι ο επεξεργαστής διαθέτει τρεις επιπλέον καταχωρητές (θέσεις μνήμης) οι οποίοι ενσωματώνουν τις δυνατότητες του πολλαπλασιασμού, της διαίρεσης, της αφαίρεσης και της μετακίνησης περιεχομένου από μία θέση μνήμης σε μία άλλη.

Πρόσθετα Χαρακτηριστικά του επεξεργαστή

Όλοι οι μικροελεγκτές της σειράς PIC16F87XA διαθέτουν μία σειρά από χαρακτηριστικά τα οποία:

- Αυξάνουν την αξιοπιστία του συστήματος
- Μειώνουν το κόστος - μειώνοντας τον αριθμό των απαιτούμενων περιφερειακών εξαρτημάτων
- Εφοδιάζονται με λειτουργίες εξοικονόμησης ενέργειας
- Παρέχουν προστασία (κλείδωμα) του κώδικα του προγράμματος

και είναι οι εξής:

- Δυνατότητα επιλογής του εξωτερικού ταλαντωτή (oscillator)
- Μηδενισμός (Reset)
- Power-on Reset (POR)

- Χρονόμετρο για τη σταθεροποίηση της παροχής ενέργειας (Power-up Timer - PWRT)
- Χρονόμετρο για τη σταθεροποίηση του ταλαντωτή (Oscillator Start-up Timer - OST)
- Brown-out Reset (BOR)
- Διακοπές (Interrupts)
- Χρονομετρητής επαγρύπνησης - Επιτηρητής (Watchdog Timer - WDT)
- Λειτουργία Ύπνωσης (Sleep)
- Προστασία (κλείδωμα) κώδικα προγράμματος (Code Protection)
- Σειριακός προγραμματισμός του μικροελεγκτή ενώ βρίσκεται εντός του συστήματος (In-Circuit Serial Programming)
- Αποσφαλμάτωση του προγράμματος ενώ βρίσκεται εντός του συστήματος (In-Circuit Debugger)

Η δυνατότητα επιλογής διαφορετικών εξωτερικών ταλαντωτών διευκολύνει τη χρήση του μικροελεγκτή σε διαφορετικές εφαρμογές. Ο ταλαντωτής με χρήση πυκνωτών και αντιστάσεων (RC oscillator) μειώνει το κόστος, ενώ ο κρυσταλλικός ταλαντωτής (LP – Low Power crystal) προσφέρει μεγαλύτερη ακρίβεια και απαιτεί λιγότερη ενέργεια. Ένα σύνολο από bits διαμόρφωσης χρησιμοποιούνται για να κάνουμε τις απαραίτητες ρυθμίσεις (*Σχήμα 1.6*).

Κατά την έναρξη λειτουργίας του μικροελεγκτή αυτός διατηρείται σε κατάσταση μηδενισμού μέχρι να εξασφαλιστούν οι κατάλληλες συνθήκες λειτουργίας (σταθεροποίηση τάσης τροφοδοσίας, σταθεροποίηση συχνότητας ταλαντωτή, κατάλληλη θερμοκρασία κ.α). Αυτή η διαδικασία περιγράφεται με τον όρο “Power-on Reset” (POR).

Δύο χρονόμετρα παρέχουν τις απαραίτητες καθυστερήσεις κατά την έναρξη λειτουργίας. Το ένα διατηρεί το μικροελεγκτή σε κατάσταση μηδενισμού (Reset) ώστε να σταθεροποιηθεί ο ταλαντωτής (Oscillator Start-up Timer - OST). Το δεύτερο (Power-up Timer - PWRT), προσφέρει μία καθυστέρηση προκαθορισμένης διάρκειας (72 ms) μόνο κατά την κανονική έναρξη λειτουργίας. Σχεδιάστηκε για να διατηρεί τη συσκευή σε κατάσταση μηδενισμού μέχρι να σταθεροποιηθεί η παροχή ενέργειας. Με αυτά τα δύο χρονόμετρα ενσωματωμένα στο μικροελεγκτή, οι περισσότερες εφαρμογές δεν χρειάζονται επιπλέον εξωτερικό

κύκλωμα μηδενισμού. Η συσκευή επανέρχεται από την κατάσταση ύπνωσης μέσω του μηδενισμού, της αφύπνισης από τον επιτηρητή, ή λόγω κάποιας διακοπής.

Ανάλογα με τον “Power-on Reset” (POR), αν κατά τη λειτουργία του μικροελεγκτή, διαπιστωθεί μια πτώση τάσης εκτός κάποιου ορίου για χρονικό διάστημα μεγαλύτερο των 100 μs ο μικροελεγκτής διατηρείται σε κατάσταση μηδενισμού μέχρι η τάση να επανέλθει σε φυσιολογικά επίπεδα. Αυτή η διαδικασία περιγράφεται με τον όρο “Brown-out Reset” (BOR).

Ο μικροελεγκτής υποστηρίζει και την εκτέλεση του προγράμματος καθοδηγούμενο από εξωτερικές διακοπές, αν αυτό είναι επιθυμητό. Αν ανιχνευθεί σήμα διακοπής, ολοκληρώνεται η εντολή που εκτελείται εκείνη τη στιγμή και την εκτέλεση ακολουθεί το μπλοκ κώδικα που σχετίζεται με τη συγκεκριμένη διακοπή. Έπειτα ο επεξεργαστής συνεχίζει την κανονική ροή εκτέλεσης του προγράμματος από το σημείο που είχε διακοπεί.

Οι μικροελεγκτές της σειράς PIC16F87XA διαθέτουν έναν επιτηρητή (Watchdog Timer – WDT) ο οποίος απενεργοποιείτε μόνο μέσω των bit διαμόρφωσης (configuration bits). Ο επιτηρητής διαθέτει δικό του ταλαντωτή (RC Oscillator) και είναι ανεξάρτητος από το χρονισμό του μικροελεγκτή, για μεγαλύτερη αξιοπιστία. Όταν ο επιτηρητής είναι ενεργοποιημένος, ένας μετρητής ξεκινάει από το “00” και προσαυξάνεται μέχρι να πάρει την τιμή “FF” οπότε και μηδενίζει τη λειτουργία του μικροελεγκτή. Αυτό συμβαίνει ανά 18ms και ο λόγος που γίνεται είναι για να αποφύγουμε καταστάσεις που το πρόγραμμα έχει κολλήσει, έχει βρεθεί σε ατέρμονη βρόγχο και γενικά δεν εκτελείται φυσιολογικά. Κατά τη φυσιολογική εκτέλεση, όμως, του προγράμματος πρέπει να προβλέψουμε και να μηδενίζουμε το μετρητή ανά διαστήματα μικρότερα των 18ms ώστε να αποτραπεί να πάρει την τιμή “FF” με συνέπεια το μηδενισμό της λειτουργίας.

Ο κώδικας μηχανής (το πρόγραμμα) που έχει φορτωθεί στον μικροελεγκτή μπορεί στη συνέχεια να ανακτηθεί από κάποιον υπολογιστή με τον οποίο είναι συνδεδεμένος και να πάρουμε την πηγαία του μορφή. Μπορούμε πολύ εύκολα αποτρέψουμε αυτό το ενδεχόμενο θέτοντας την κατάλληλη τιμή στα bit CP1:CP0 του καταχωρητή διαμόρφωσης του μικροελεγκτή (*Σχήμα 1.6*). Ανάλογα μπορούμε να ρυθμίσουμε και την πρόσβαση στη μνήμη αποθήκευσης δεδομένων EEPROM.

Ο μικροελεγκτής μπορεί να προγραμματιστεί και εντός του συστήματος, δηλαδή εφόσον αυτό έχει τοποθετηθεί στο τελικό κύκλωμα της εφαρμογής. Αυτό επιτυγχάνεται με τη βοήθεια επιπλέον εξωτερικού εξοπλισμού, ο οποίος συνδέεται σε έναν ηλεκτρονικό υπολογιστή και στον μικροελεγκτή (ακροδέκτες MCLR, RB3, RB6, RB7 – ICD Connector). Με τον ίδιο εξοπλισμό έχουμε τη δυνατότητα να κάνουμε αποσφαλμάτωση του προγράμματος στις κανονικές συνθήκες λειτουργίας με όλα τα περιφερειακά εξαρτήματα και τις περιφερειακές συσκευές συνδεδεμένες.

Όλα τα παραπάνω παρουσιάζονται σχηματικά παρακάτω (*Σχήμα 1.7*).

Bits Διαμόρφωσης του μικροελεγκτή (Configuration Bits)

Τα bits διαμόρφωσης του μικροελεγκτή βρίσκονται στον “καταχωρητή διαμόρφωσης” (Configuration Word register) στη θέση μνήμης 2007h η οποία δεν εντάσσεται στην περιοχή μνήμης που μπορούμε να προσπελάσουμε από τον κώδικα του προγράμματος. Με τον καταχωρητή διαμόρφωσης ρυθμίζουμε συγκεκριμένες λειτουργίες του μικροελεγκτή (*Σχήμα 1.6*). Ο καθορισμός της τιμής τους είναι εφικτός μόνο κατά την φάση του προγραμματισμού του μικροελεγκτή, ενώ η προκαθορισμένη τιμή του καταχωρητή, αν δεν την προσδιορίσουμε εμείς, είναι η ‘3FFFh’.

1.2.2. - Οργάνωση Μνήμης

Σε κάθε μικροελεγκτή της σειράς PIC16F87XA υπάρχουν τριών τύπων μνήμης, η μνήμη του προγράμματος (μνήμη Flash), η μνήμη των δεδομένων (στατική μνήμη RAM), και η EEPROM. Η μνήμη του προγράμματος και η μνήμη των δεδομένων διαθέτουν ξεχωριστούς διαύλους ώστε να είναι εφικτή η ταυτόχρονη προσπέλασή τους.

Η μνήμη Flash χρησιμοποιείται για την αποθήκευση του προγράμματος (κώδικας). Αποτελείται από 8192 θέσεις μνήμης μεγέθους 14 bit η κάθε μία. Είναι αμετάβλητου τύπου (Non Volatile) οπότε τα περιεχόμενα της διατηρούνται αναλλοίωτα ακόμα και μετά την διακοπή της τάσης τροφοδοσίας και μπορεί να επανεγγραφεί περίπου 100,000 φορές μέσω της διαδικασίας του προγραμματισμού του μικροελεγκτή.

Η EEPROM χρησιμεύει για την αποθήκευση σημαντικών δεδομένων όπως κωδικοί αριθμοί, τιμές αρχικοποίησης, παραμέτρους διεργασιών κ.α. Το σημαντικότερο χαρακτηριστικό είναι ότι διατηρεί το περιεχόμενο της και μετά την

CONFIGURATION WORD (ADDRESS 2007h)(1)

R/P-1	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1
CP	—	DEBUG	WRT1	WRT0	CPD	LVP	BOREN	—	—	$\overline{\text{PWRTEN}}$	WDTEN	Fosc1	Fosc0
bit 13													bit0

bit 13 **CP:** Flash Program Memory Code Protection bit
1 = Code protection off
0 = All program memory code-protected

bit 12 **Unimplemented:** Read as '1'

bit 11 **DEBUG:** In-Circuit Debugger Mode bit
1 = In-Circuit Debugger disabled, RB6 and RB7 are general purpose I/O pins
0 = In-Circuit Debugger enabled, RB6 and RB7 are dedicated to the debugger

bit 10-9 **WRT1:WRT0** Flash Program Memory Write Enable bits
For PIC16F876A/877A:
11 = Write protection off; all program memory may be written to by EECON control
10 = 0000h to 00FFh write-protected; 0100h to 1FFFh may be written to by EECON control
01 = 0000h to 07FFh write-protected; 0800h to 1FFFh may be written to by EECON control
00 = 0000h to 0FFFh write-protected; 1000h to 1FFFh may be written to by EECON control
For PIC16F873A/874A:
11 = Write protection off; all program memory may be written to by EECON control
10 = 0000h to 00FFh write-protected; 0100h to 0FFFh may be written to by EECON control
01 = 0000h to 03FFh write-protected; 0400h to 0FFFh may be written to by EECON control
00 = 0000h to 07FFh write-protected; 0800h to 0FFFh may be written to by EECON control

bit 8 **CPD:** Data EEPROM Memory Code Protection bit
1 = Data EEPROM code protection off
0 = Data EEPROM code-protected

bit 7 **LVP**: Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit
1 = RB3/PGM pin has PGM function; low-voltage programming enabled
0 = RB3 is digital I/O, HV on $\overline{\text{MCLR}}$ must be used for programming

bit 6 **BOREN**: Brown-out Reset Enable bit
1 = BOR enabled
0 = BOR disabled

bit 5-4 **Unimplemented:** Read as '1'

bit 3 **$\overline{\text{PWRTEN}}$**: Power-up Timer Enable bit
1 = PWRT disabled
0 = PWRT enabled

bit 2 **WDTEN**: Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

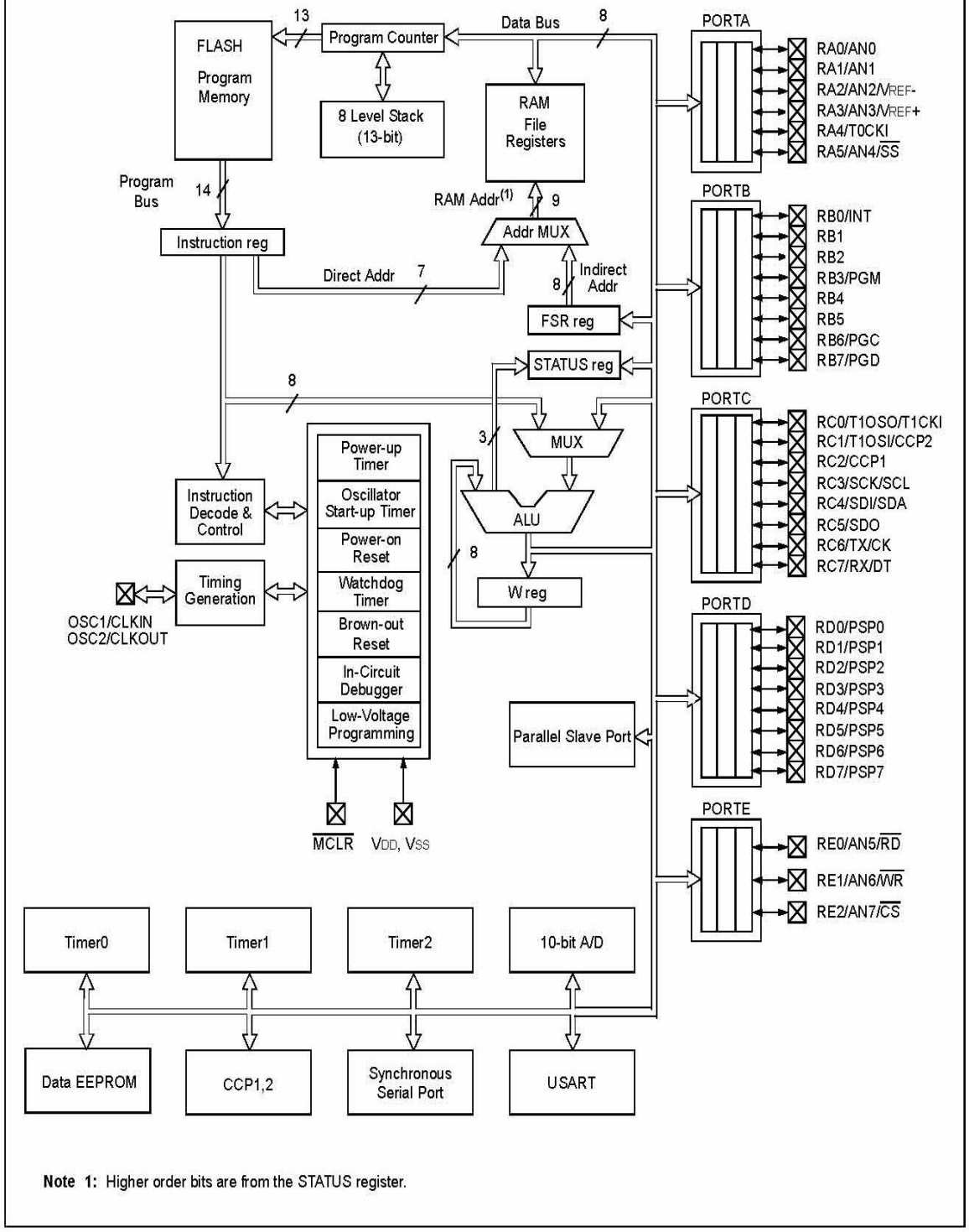
bit 1-0 **Fosc1:Fosc0**: Oscillator Selection bits
11 = RC oscillator
10 = HS oscillator
01 = XT oscillator
00 = LP oscillator

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

Note 1: The erased (unprogrammed) value of the Configuration Word is 3FFFh.

Σχήμα 1.6 – Ο καταχωρητής διαμόρφωσης του μικροελεγκτή

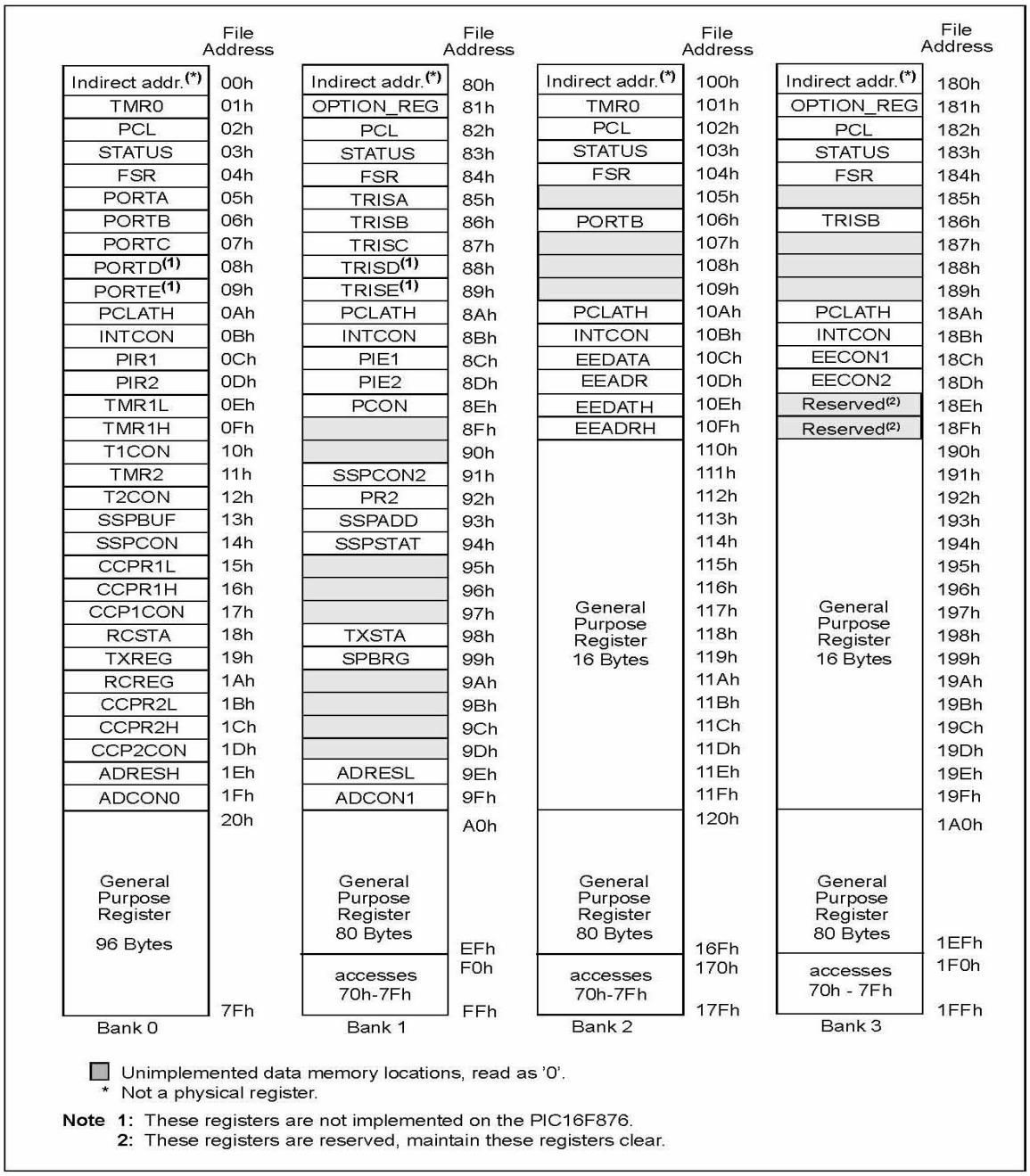
Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

Σχήμα 1.7 – Η εσωτερική δομή του PIC16F877

διακοπή της τάσης τροφοδοσίας, για διάστημα που φτάνει τα 40 χρόνια (σύμφωνα με την Microchip).Τέλος μπορεί να επανεγγραφεί μέσα από το πρόγραμμα περίπου 1.000.000 φορές. Η κύρια μνήμη είναι χωρισμένη σε πολλαπλές τράπεζες (4 στον PIC16F877) που περιλαμβάνουν καταχωρητές γενικού σκοπού (General Purpose Registers - GPRs) και καταχωρητές ειδικών λειτουργιών (Special Function Registers - SFRs). Κάθε τράπεζα έχει 128 θέσεις μνήμης. Οι πρώτες θέσεις είναι αφιερωμένες για τους καταχωρητές ειδικών λειτουργιών, ενώ οι επόμενες για τους καταχωρητές γενικού σκοπού. Εάν αφαιρέσουμε τους ειδικούς καταχωρητές συναρτήσεων από το συνολικό αριθμό των διευθύνσεων της RAM και πέρα κάποιων καταχωρητών που επαναλαμβάνονται σε περισσότερες της μιας τράπεζες μνήμης, υπολείπονται 368 bytes τα οποία μπορούν να χρησιμοποιηθούν ως καταχωρητές δεδομένων (γενικού σκοπού). Η προεπιλεγμένη τράπεζα κατά την έναρξη λειτουργίας του μικροελεγκτή είναι η πρώτη (Bank_0) και οι θέσεις μνήμης της αριθμούνται από το 0 έως τη θέση 0Fh, της δεύτερης τράπεζας (Bank_1) αριθμούνται από το 80h μέχρι τη FFh κτλ (*Σχήμα 1.8*).



Σχήμα 1.8 – Η οργάνωση της μνήμης

1.2.3. - Το Βασικό Σύνολο Εντολών των PIC

Το σύνολο εντολών των PIC16 αποτελείται από τρεις βασικές κατηγορίες:

- **Εντολές χειρισμού ψηφιολέξης (byte)**
- **Εντολές χειρισμού ψηφίου (bit)**
- **Εντολές πράξεων με σταθερές τιμές και ελέγχου**

Κάθε σύνολο εντολών των PIC16 είναι μια λέξη με μήκος 14 bit χωρισμένη σε έναν **κωδικό-διεργασίας** και σε έναν ή περισσότερους **τελεστές**. Ο κωδικός

διεργασίας καθορίζει το είδος της εντολής και ο/οι τελεστής/ές ορίζουν λεπτομερέστερα τη λειτουργία της. Η μορφή της κάθε εντολής παρουσιάζεται στο *Πίνακα 1.1*, ενώ συνοψίζονται και τα πεδία των διαφόρων κωδικών διεργασίας.

Στις **εντολές χειρισμού byte**, το 'f' προσδιορίζει τον καταχωρητή που θα χρησιμοποιηθεί από την εντολή και το 'd' τον προορισμό. Αν το 'd' είναι 0, το αποτέλεσμα τοποθετείται στον καταχωρητή W, ενώ αν είναι 1, το αποτέλεσμα τοποθετείται στον καταχωρητή που έχει χρησιμοποιηθεί από την εντολή.

Στις **εντολές χειρισμού bit**, το 'b' αναπαριστά τη θέση του bit που θα τροποποιηθεί από την εντολή, ενώ το 'f' αναπαριστά τη διεύθυνση/καταχωρητή όπου το bit είναι τοποθετημένο.

Στις **εντολές πράξεων με σταθερές τιμές και ελέγχου** το 'k' αναπαριστά μία σταθερά 8 ή 11 ψηφίων.

Πίνακας 1.1 – Το σύνολο εντολών του PIC16F877

PIC16F87XA INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add Literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND Literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call Subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to Address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR Literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move Literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from Interrupt	2	00 0000 0000 1001		
RETLW	k	Return with Literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into Standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from Literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR Literal with W	1	11 1010 kkkk kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 module.

3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

Ένας κύκλος εντολής αποτελείται από 4 περιόδους του ταλαντωτή. Για έναν ταλαντωτή με συχνότητα 4MHz η διάρκεια εκτέλεσης μίας εντολής είναι 1μs. Όλες οι εντολές εκτελούνται σε έναν κύκλο εντολής, εκτός αν κάποια συνθήκη είναι αληθινή ή ο μετρητής προγράμματος τροποποιείται σαν αποτέλεσμα μιας εντολής. Σε αυτές τις περιπτώσεις, η εκτέλεση διαρκεί δύο κύκλους (στον δεύτερο κύκλο εκτελείται η εντολή NOP).

Εντολές που χρησιμοποιήθηκαν στο πρόγραμμα του μικροελεγκτή

Εντολές χειρισμού byte:

NOP: Με την εντολή NOP εισάγουμε έναν νεκρό κύκλο. Η εντολή αυτή δεν κάνει κάποια εργασία, απλά καθυστερεί την εκτέλεση των υπόλοιπων κατά ένα κύκλο.

CLRF "f": μηδενίζει όλα τα bits του καταχωρητή "f".

MOVWF "f": Μεταφέρει το περιεχόμενο του W στον καταχωρητή "f".

DECFSZ "f": Μειώνει την τιμή του καταχωρητή "f" κατά 1, ενώ αν αυτή είναι 0, υπερπηδά την επόμενη εντολή.

Εντολές χειρισμού bit:

BCF "f", "d": Μηδενίζει το bit που βρίσκεται στη θέση "d" στον καταχωρητή "f".

BSF "f", "d": Δίνει την τιμή 1 στο bit που βρίσκεται στη θέση "d" στον καταχωρητή "f".

BTFSC "f", "d": Ελέγχει την τιμή του bit που βρίσκεται στη θέση "d" στον καταχωρητή "f" και εάν αυτή είναι 0 υπερπηδά την επόμενη εντολή.

BTFSS "f", "d": Ελέγχει την τιμή του bit που βρίσκεται στη θέση "d" στον καταχωρητή "f" και εάν αυτή είναι 1 υπερπηδά την επόμενη εντολή.

Εντολές πράξεων με σταθερές τιμές και ελέγχου:

MOVLW "k": Μεταφέρει τη σταθερά "k" στον καταχωρητή W.

GOTO "k": Μεταφέρει την εκτέλεση του προγράμματος στη θέση μνήμης "k" (ο μετρητής εντολών δείχνει στη θέση μνήμης "k").

CALL "k": Καλεί την υπορουτίνα που βρίσκεται στη θέση μνήμης "k" (η εκτέλεση του προγράμματος συνεχίζει από τη θέση μνήμης "k").

RETURN: Επιστροφή από την υπορουτίνα. Μετά την εκτέλεση της υπορουτίνας, η εκτέλεση του προγράμματος συνεχίζεται από την εντολή που βρίσκεται αμέσως μετά την κλήση της.

1.2.4. - Καταχωρητές ειδικών λειτουργιών

Όπως είδαμε το σύνολο των καταχωρητών αποτελείται από τους καταχωρητές ειδικών λειτουργιών (SFRs) και τους καταχωρητές γενικού σκοπού (GPRs). Οι SFRs έχουν προκαθορισμένες λειτουργίες και βρίσκονται στις θέσεις μνήμης 00-1F (Τράπεζα 0), 80-9F (Τράπεζα 1), 100-10F (Τράπεζα 2), 180-18F (Τράπεζα 3). Πολλοί από αυτούς επαναλαμβάνονται σε περισσότερες της μίας τράπεζες. Οι λειτουργία μερικών από αυτούς θα εξηγηθούν παρακάτω.

Μετρητής Προγράμματος (PCL)

Ο μετρητής προγράμματος δείχνει την εντολή που πρέπει να εκτελέσει ο επεξεργαστής. Κατά την έναρξη ή το μηδενισμό λειτουργίας παίρνει την τιμή 0. Για την διευθυνσιοδότηση των 8192 θέσεων της μνήμης προγράμματος απαιτείται ένας μετρητής με μήκος 13 bits (0000-1FFF, 0-8191). Ο καταχωρητής PCL (Program Counter Low, SFR 02h) περιέχει τα πρώτα 8 bits του μετρητή προγράμματος και μπορεί να εγγραφεί και να διαβαστεί σαν ένας οποιοσδήποτε καταχωρητής. Τα bit 8-12 του μετρητή προγράμματος είναι έμμεσα προσπελάσιμα μέσω του καταχωρητή PCLATH (Program Counter Latch High, SFR 0Ah).

Καταχωρητής κατάστασης (STATUS)

Ο καταχωρητής κατάστασης διατηρεί το αποτέλεσμα συγκεκριμένων λειτουργιών, την κατάσταση της τάσης τροφοδοσίας του μικροελεγκτή και περιλαμβάνει τα bits με τα οποία επιλέγουμε μία τράπεζα μνήμης (Σχήμα).

Σηματοδότης Μηδέν *(Z)*

Ο Σηματοδότης παίρνει την τιμή 1 όταν το αποτέλεσμα μίας λειτουργίας είναι 0, διαφορετικά παίρνει την τιμή 0. Για να δούμε ποιες λειτουργίες επηρεάζουν το σηματοδότη πρέπει να συμβουλευτούμε όλο το σύνολο εντολών.

Οι εντολές ελέγχου bit (Bit test) και υπερπήδησης (skip) χρησιμοποιούν το σηματοδότη για αλλαγή της ροής υπό συνθήκη, αλλά εντολές που χρησιμοποιούνται για την προσαύξηση/μείωση και υπερπήδηση αν γίνει 0, όλως περιέργως, δεν επηρεάζουν το σηματοδότη.

Μία τυπική χρήση είναι ο έλεγχος της ισότητας δύο αριθμών, αφαιρώντας τους, ελέγχοντας το bit Z και έπειτα υπερπηδώντας.

Σηματοδότης Κρατούμενου (C)

Ο σηματοδότης επηρεάζεται μόνο από τις εντολές πρόσθεσης, αφαίρεσης και περιστροφής. Αν κατά την πρόσθεση υπάρχει κρατούμενο ο σηματοδότης παίρνει την τιμή 1. Αυτό συμβαίνει αν κατά την πρόσθεση αριθμών των 8 bits το αποτέλεσμα είναι ένας αριθμός με μήκος 9 bits. Σε αυτήν την περίπτωση για το τελικό αποτέλεσμα πρέπει να συνυπολογίσουμε και το κρατούμενο. Κατά την αφαίρεση πρέπει πρώτα να δώσουμε στο σηματοδότη την τιμή 1 γιατί θα αποτελέσει το δανεικό ψηφίο στην περίπτωση που χρειαστεί. Οπότε αν μετά την αφαίρεση ο σηματοδότης είναι 0 σημαίνει ότι το αποτέλεσμα ήταν αρνητικό.

Λαμβάνοντας υπ όψη τους σηματοδότες μηδέν και κρατούμενο στις αριθμητικές πράξεις διαπιστώνουμε αν το αποτέλεσμα ήταν θετικό, αρνητικό ή μηδέν.

Bits Επιλογής τράπεζας μνήμης (IRP:RP1:RP0)

Η επιλογή μίας τράπεζας μνήμης γίνεται θέτοντας την κατάλληλη τιμή στα bits IRP:RP1:RP2 του καταχωρητή κατάστασης, όπως φαίνεται στο *Σχήμα 1.9*.

Bits Κατάστασης λειτουργίας

Ο καταχωρητής κατάστασης διαθέτει δύο bits μόνο ανάγνωσης που προσδιορίζουν την κατάσταση λειτουργίας του μικροελεγκτή. Το Power Down (PD) bit γίνεται 0 αν ο μικροελεγκτής βρίσκεται σε κατάσταση ύπνωσης (Sleep), ενώ το Time Out (TO) bit γίνεται 0 αν έχει εξαντληθεί ο μετρητής επαγρύπνησης (επιτηρητής).

STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7 **IRP:** Register Bank Select bit (used for indirect addressing)
1 = Bank 2, 3 (100h-1FFh)
0 = Bank 0, 1 (00h-FFh)

bit 6-5 **RP1:RP0**: Register Bank Select bits (used for direct addressing)
11 = Bank 3 (180h-1FFh)
10 = Bank 2 (100h-17Fh)
01 = Bank 1 (80h-FFh)
00 = Bank 0 (00h-7Fh)
Each bank is 128 bytes.

bit 4 **$\overline{TO}$**: Time-out bit
1 = After power-up, CLRWDT instruction or SLEEP instruction
0 = A WDT time-out occurred

bit 3 **$\overline{PD}$**: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2 **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)
(for $\overline{borrow}$, the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{borrow}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high, or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Σχήμα 1.9 – Ο Καταχωρητής Κατάστασης (Status Register)

Καταχωρητές TRISx – PORTx

Το ζεύγος αυτών των καταχωρητών χρησιμοποιείται για τη διαχείριση της "x" πόρτας (ψηφιακή διεπαφή επικοινωνίας) και έχουν μήκος ίσο με τους αντίστοιχους ακροδέκτες που διαθέτει η κάθε πόρτα *(Σχήμα 1.10)*. Ο TRISx ορίζει τον κάθε ακροδέκτη ως ψηφιακή είσοδο ή έξοδο, ενώ με τον PORTx διαβάζουμε από τις εισόδους της πόρτας ή γράφουμε στις εξόδους αυτής. Περισσότερες λεπτομέρειες για τη χρήση των πορτών θα δούμε στην αντίστοιχη παράγραφο *(1.2.7)*.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111

Legend: x = unknown, u = unchanged

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
88h	TRISD	PORTD Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged

Σχήμα 1.10 – Καταχωρητές PORTx-TRISx

Καταχωρητές ADCON0 και ADCON1, ADRESH και ADRESL

Οι δύο πρώτοι καταχωρητές χρησιμοποιούνται για τη διαχείριση των αναλογικών εισόδων και του μετατροπέα σήματος από αναλογικό σε ψηφιακό (Analog-to-Digital Converter - ADC). Ο καταχωρητής ADCON0 ελέγχει τη λειτουργία του μετατροπέα, ενώ ο ADCON1 ρυθμίζει τη λειτουργία των αναλογικών εισόδων *(Σχήματα 1.11, 1.12)*. Οι δύο επόμενοι καταχωρητές χρησιμοποιούνται για την τοποθέτηση του αποτελέσματος της μετατροπής.

Λεπτομέρειες για τη χρήση του μετατροπέα σήματος από αναλογικό σε ψηφιακό (ADC) θα δούμε στην αντίστοιχη παράγραφο *(1.2.8).*

ADCON0 REGISTER (ADDRESS 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7-6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits (ADCON0 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	**00**	Fosc/2
0	**01**	Fosc/8
0	**10**	Fosc/32
0	**11**	FRC (clock derived from the internal A/D RC oscillator)
1	**00**	Fosc/4
1	**01**	Fosc/16
1	**10**	Fosc/64
1	**11**	FRC (clock derived from the internal A/D RC oscillator)

bit 5-3 **CHS2:CHS0:** Analog Channel Select bits

000 = Channel 0 (AN0)
001 = Channel 1 (AN1)
010 = Channel 2 (AN2)
011 = Channel 3 (AN3)
100 = Channel 4 (AN4)
101 = Channel 5 (AN5)
110 = Channel 6 (AN6)
111 = Channel 7 (AN7)

Note: The PIC16F873A/876A devices only implement A/D channels 0 through 4; the unimplemented selections are reserved. Do not select any unimplemented channels with these devices.

bit 2 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

When ADON = 1:
1 = A/D conversion in progress (setting this bit starts the A/D conversion which is automatically cleared by hardware when the A/D conversion is complete)
0 = A/D conversion not in progress

bit 1 **Unimplemented:** Read as '0'

bit 0 **ADON:** A/D On bit

1 = A/D converter module is powered up
0 = A/D converter module is shut-off and consumes no operating current

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Σχήμα 1.11 – Καταχωρητής ADCON0

ADCON1 REGISTER (ADDRESS 9Fh)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified. Six (6) Most Significant bits of ADRESH are read as '0'.
0 = Left justified. Six (6) Least Significant bits of ADRESL are read as '0'.

bit 6 **ADCS2:** A/D Conversion Clock Select bit (ADCON1 bits in shaded area and in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	FOSC/2
0	01	FOSC/8
0	10	FOSC/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	FOSC/4
1	01	FOSC/16
1	10	FOSC/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

A = Analog input D = Digital I/O
C/R = # of analog input channels/# of A/D voltage references

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note: On any device Reset, the port pins that are multiplexed with analog functions (ANx) are forced to be an analog input.

Σχήμα 1.12 - Καταχωρητής ADCON1

Καταχωρητές TXSTA, RCSTA, SPBRG, TXREG και PIR1

Οι καταχωρητές αυτοί χρησιμοποιούνται για τη διαχείριση της σειριακής επικοινωνίας. Ο TXSTA ρυθμίζει τις παραμέτρους για τη σειριακή αποστολή

(Σχήμα 1.13), ο RCSTA ρυθμίζει τις παραμέτρους για τη σειριακή παραλαβή και θέτει σε λειτουργία τη σειριακή πόρτα *(Σχήμα 1.14)*, ο SPBRG περιέχει μία τιμή που καθορίζει το ρυθμό μετάδοσης, στον TXREG τοποθετούμε τα δεδομένα που θέλουμε να αποστείλουμε (αντίστοιχα υπάρχει και ο RCREG για την παραλαβή δεδομένων αλλά δε θα ασχοληθούμε με αυτήν), ενώ το bit TXIF του καταχωρητή PIR1 *(Σχήμα 1.15)* σηματοδοτεί αν έχει ολοκληρωθεί η αποστολή (TXIF=1). Η διαδικασία της σειριακής επικοινωνίας θα περιγραφεί αναλυτικά σε επόμενη παράγραφο *(1.2.9)*.

TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care.
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled
Note: SREN/CREN overrides TXEN in Sync mode.

bit 4 **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **Unimplemented:** Read as '0'

bit 2 **BRGH**: High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode.

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data, can be Parity bit

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Σχήμα 1.13 – Καταχωρητής TXSTA

RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)
0 = Serial port disabled

bit 6 **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave:
Don't care.

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit

bit 2 **FERR**: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data (can be parity bit but must be calculated by user firmware)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Σχήμα 1.14 – Καταχωρητής RCSTA

PIR1 REGISTER (ADDRESS 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **PSPIF:** Parallel Slave Port Read/Write Interrupt Flag bit(1)
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred
Note 1: PSPIF is reserved on PIC16F873A/876A devices; always maintain this bit clear.

bit 6 **ADIF**: A/D Converter Interrupt Flag bit
1 = An A/D conversion completed
0 = The A/D conversion is not complete

bit 5 **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer is full
0 = The USART receive buffer is empty

bit 4 **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer is empty
0 = The USART transmit buffer is full

bit 3 **SSPIF**: Synchronous Serial Port (SSP) Interrupt Flag bit
1 = The SSP interrupt condition has occurred and must be cleared in software before returning from the Interrupt Service Routine. The conditions that will set this bit are:
- SPI – A transmission/reception has taken place.
- I^2C Slave – A transmission/reception has taken place.
- I^2C Master
  - A transmission/reception has taken place.
  - The initiated Start condition was completed by the SSP module.
  - The initiated Stop condition was completed by the SSP module.
  - The initiated Restart condition was completed by the SSP module.
  - The initiated Acknowledge condition was completed by the SSP module.
  - A Start condition occurred while the SSP module was Idle (multi-master system).
  - A Stop condition occurred while the SSP module was Idle (multi-master system).

0 = No SSP interrupt condition has occurred

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode.

bit 1 **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Σχήμα 1.15 – Καταχωρητής PIR1

1.2.5. - Ο Χρονισμός του μικροελεγκτή

Για το χρονισμό του μικροελεγκτή αρκεί η χρήση ενός ταλαντωτή αποτελούμενος από ένα πυκνωτή και μία αντίσταση. Αυτή η επιλογή μειώνει το κόστος κατασκευής αλλά δεν είναι αξιόπιστη. Για μεγαλύτερη ακρίβεια, ειδικά στην περίπτωση που το πρόγραμμα χρησιμοποιεί τα ενσωματωμένα χρονόμετρα του

μικροελεγκτή για να κάνει ακριβείς μετρήσεις ή να παράγει ακριβή σήματα στην έξοδο, ένας κρυσταλλικός ταλαντωτής είναι απαραίτητος.

Ο κρυσταλλικός ταλαντωτής, συνήθως, συνδέεται στους ακροδέκτες χρονισμού με ένα ζεύγος πυκνωτών (15pF) για να σταθεροποιήσουν τη συχνότητα. Ο κρύσταλλος λειτουργεί σαν ένα αυτόνομο συντονισμένο κύκλωμα, όπου ο λίθος χαλαζία ή ο κεραμικός ταλαντωτής πάλλεται με μία συχνότητα υψηλής ακριβείας όταν υπόκειται σε ηλεκτρική διέγερση. Η ακρίβεια της συχνότητας του ταλαντωτή είναι πολύ υψηλή με πιθανότητα σφάλματος μόλις 0,005%.

Η μέγιστη συχνότητα που μπορούν να λειτουργήσουν οι μικροελεγκτές της σειράς PIC16FXXX αγγίζει τα 20MHz χρησιμοποιώντας κρύσταλλο υψηλής ταχύτητας (High-Speed Crystal).

1.2.6. - Η επεξεργασία των Δεδομένων

Τα περισσότερα προγράμματα μικροελεγκτών χρειάζονται να επεξεργαστούν δεδομένα, είτε με αριθμητικές, είτε με λογικές πράξεις.

Οι κύριοι τύποι δεδομένων στους γενικούς επεξεργαστές είναι αριθμοί και χαρακτήρες. Οι χαρακτήρες δεν είναι πρόβλημα, διότι είναι μόνο 26 γράμματα του αλφάβητου και 10 αριθμητικοί χαρακτήρες. Ακόμα και να συμπεριλάβουμε τα κεφαλαία και τα πεζά γράμματα, συνολικά απαιτούνται 62 κωδικοποιήσεις.

Το βασικό σύνολο χαρακτήρων ASCII απαιτεί μόνο 7- bit για τη κωδικοποίηση κάθε συμβόλου. Η χρήση του $8^{ου}$ bit κωδικοποίησης επιτρέπει να προστεθούν και άλλοι ειδικοί χαρακτήρες. Τα αριθμητικά δεδομένα αποτελούν συνήθως τη δύσκολη υπόθεση, καθώς μια κωδικοποίηση των 8 bit μας επιτρέπει να αναπαραστήσουμε αριθμούς μεταξύ του 0 και του 255. Άρα γίνεται κατανοητό πως απαιτούνται κάποιες επιπρόσθετες μεθόδους για τον χειρισμό μεγαλύτερων καθώς και μικρότερων αριθμών, ούτως ώστε να μπορούν να πραγματοποιηθούν υπολογισμοί με ένα μεγάλο βαθμό ακρίβειας.

Αριθμητικά Συστήματα

Οι υπολογιστές εργάζονται μόνο με δυαδικούς αριθμούς, αλλά οι άνθρωποι μαθαίνουν να εργάζονται με τους δεκαδικούς αριθμούς.

Η βάση ενός αριθμητικού συστήματος αντιστοιχεί στο πλήθος των ψηφίων που χρησιμοποιεί (το 10 είναι η βάση του δεκαδικού συστήματος που έχει 10

ψηφία, το 2 είναι η βάση του δυαδικού που έχει 2 ψηφία, το 8 είναι η βάση του οκταδικού και το 16 είναι η βάση του δεκαεξαδικού συστήματος)

Θεωρητικά, μπορεί να χρησιμοποιηθεί οποιαδήποτε βάση, αλλά στην πράξη κάποιες βάσεις είναι πιο χρήσιμες από κάποιες άλλες.

Ιστορικά, η βάση του 12 χρησιμοποιήθηκε ευρέως (ώρες, λεπτά, στα χαρτονομίσματα) διότι είναι πάρα πολύ χρήσιμη αφού διαιρείται με το 2, το 3, το 4 και το 6. Βέβαια δεν είναι αληθινό αριθμητικό σύστημα γιατί δεν υπάρχουν διακριτά σύμβολα για το 10 και το 11. Παρομοίως, το σύστημα BCD δεν είναι ένα κατάλληλο αριθμητικό σύστημα διότι η αρίθμηση σταματάει στον αριθμό 9. Το δεκαεξαδικό σύστημα είναι ένα αληθινό αριθμητικό σύστημα, διότι χρησιμοποιεί διακριτά σύμβολα για το 10(A) μέχρι και το 15(F). Πρόκειται για ένα πολύ χρήσιμο σύστημα αρίθμησης διότι παρέχει ένα συμπυκνωμένο τρόπο αναπαράστασης των δυαδικών αριθμών, του φυσικού αριθμητικού συστήματος των ψηφιακών υπολογιστών και ελεγκτών.

Δεκαδικό Σύστημα

Το 0 αποτέλεσε μια μεγάλη εφεύρεση και τα αριθμητικά συστήματα εξαρτώνται από αυτό. Μια ακόμα πολύ σημαντική ιδέα, αποτέλεσε η βαρύτητα της θέσης του κάθε συμβόλου. Ένας αριθμός υπολογίζεται κατά τα ακόλουθα:

Τελική Τιμή = **Σ** (βαρύτητα θέσης ψηφίου **x** τιμή ψηφίου)

Column weight	$1000 = 10^3$	$100 = 10^2$	$10 = 10^1$	$1 = 10^0$
Example	**7**	**3**	**9**	**5**
Value	7×10^3	3×10^2	9×10^1	5×10^0
Total value	$7000 + 300 + 90 + 5 = \mathbf{7395}$			

Δυαδικό Σύστημα

Τα υπολογιστικά συστήματα εργάζονται στο δυαδικό σύστημα διότι είναι ο πιο αποτελεσματικός και ακριβής τρόπος στην ηλεκτρονική αναπαράσταση των αριθμών. Προτού αναπτυχθεί το κατάλληλο ψηφιακό υλικό, οι υπολογιστές λειτουργούσαν με βάση την αναλογική τάση που χρειάζονταν για να αναπαραστήσουν τις δεκαδικές τιμές, αλλά η ακρίβεια περιοριζόταν από την ποιότητα των σημάτων.

Από την άλλη μεριά, η ακρίβεια το ψηφιακού υπολογιστή μπορεί να αυξηθεί θεωρητικά σε οποιοδήποτε βαθμό ακρίβειας, απλώς αυξάνοντας τον αριθμό των

δυαδικών ψηφίων. Εάν, για παράδειγμα η επεξεργασία γίνεται με 32 bits, εν δυνάμει μας παρέχεται ένας βαθμός ακρίβειας 1 προς 232 (4294967296), που ακόμα και τα 8 bit μας δίνουν την ακρίβεια 1 προς 256, περισσότερο από 0.5% πιθανότητα σφάλματος.

Column weight	$8 = 2^3$	$4 = 2^2$	$2 = 2^1$	$1 = 2^0$
Example	**1**	**0**	**0**	**1**
Value	$1 \times 8 = 8$	$0 \times 4 = 0$	$0 \times 2 = 0$	$1 \times 1 = 1$
Total value	$8 + 0 + 0 + 1 = 9$			

Το αποτέλεσμα της ανάλυσης της δομής ενός τυπικού δυαδικού αριθμού μας δείχνει ότι η δεκαδική τιμή μπορεί εύκολα να υπολογισθεί ως εξής:

Τελική Τιμή = **Σ** (Βαρύτητα θέσης των μη-μηδενικών ψηφίων)

Το πρότυπο της βαρύτητας των θέσεων των ψηφίων, είναι η βάση του συστήματος υψωμένη στην δύναμη του 0, του 1, του 2, του 3, κτλ. Αυτή είναι και η σημασία του αριθμού βάσης ενός αριθμητικού συστήματος.

Ο μέγιστος αριθμός για ένα πλήθος ψηφίων εξαντλείται όταν όλα τα ψηφία είναι 1.Σε ένα αριθμό των 8-bit, η μέγιστη τιμή είναι $11111111_2 = 255_{10}$ (ο αριθμός 2 αποτελεί τη βάση του συστήματος). Αυτό υπολογίζεται κάνοντας τη πράξη 2^8 - 1, το 2 υψωμένο στη δύναμη της βάσης μείον το 1. Ο αριθμός των διαφορετικών κωδικοποιήσεων, μαζί με το 0, είναι 2^8 = 256.

Αυτό είναι σημαντικό στον ορισμό της χωρητικότητας της μνήμης, όπου ένα έξτρα bit στη διεύθυνση, διπλασιάζει τη μνήμη. Τα σημαντικά σημεία αναφοράς είναι 2^10 = 1024 bytes (1 kB), 2^16 (64 kB), 2^20 (1 MB) και 2^30 (1 GB).Η υψηλότερη διεύθυνση σε μια μνήμη 1 kB, για παράδειγμα, είναι η 1023.

Δεκαεξαδικό Σύστημα

Το όνομα του αριθμητικού συστήματος προσδιορίζει και την αριθμητική βάση του 16 .Το πρόβλημα με αυτό το σύστημα ήταν τα έξτρα αριθμητικά σύμβολα που απαιτούνται για την κωδικοποίηση και έτσι μερικά σύμβολα που χρησιμοποιούνται ως χαρακτήρες, εδώ συμπεριφέρονται ως αριθμητικά:από το $A_{16} = 10_{(10)}$ έως το $F_{16} = 15_{(10)}$.

Example	9	B	0	F
Column weight	$1000_{16} = 4096 = 16^3$	$100_{16} = 256 = 16^2$	$10_{16} = 16 = 16^1$	$1_{16} = 1 = 16^0$
Value	$9 \times 4096 = 36864$	$B = 11 \times 256 = 2816$	$0 \times 16 = 0$	$F = 15 \times 1 = 15$
Total value	$36864 + 2816 + 0 + 15 =$ **39695**			

Η βαρύτητα κάθε ψηφίου στην μετατροπή σε δεκαεξαδικό σύστημα αρίθμησης, είναι 1_{16}, 10_{16}, 100_{16}, 1000_{16}, κτλ. Αυτό εφαρμόζεται σε όλα τα αριθμητικά συστήματα, όπου η βαρύτητα του ψηφίου προσδιορίζεται προοδευτικά ως 1x*n*, 10x*n*, 100x*n*, 1000x*n*, κτλ, όπου το *n* είναι η βάση του συστήματος. Εύκολα γίνεται αντιληπτό ότι η μετατροπή από το δεκαεξαδικό σύστημα στο δεκαδικό δεν είναι τόσο απλή υπόθεση, όσο η μετατροπή από δεκαεξαδικό στο δυαδικό σύστημα.

Μετατροπές μεταξύ συστημάτων αρίθμησης

Οι μετατροπές μεταξύ αριθμητικών συστημάτων απαιτούνται συχνά στη λειτουργία των συστημάτων μικροϋπολογιστών. Τα δεδομένα μπορεί να εισάγονται σε μορφή ASCII, να γίνει επεξεργασία τους στο δυαδικό σύστημα, ή στο FP format και η έξοδος μπορεί να είναι με σύστημα κωδικοποίησης BCD. Ο κώδικας μηχανής συνήθως αναπαρίσταται σε δεκαεξαδική μορφή, καθώς πρόκειται για μια πιο συμπυκνωμένη αναπαράσταση και άρα πιο αποδοτική, για αυτό και πρέπει να γνωρίζουμε την μετατροπή από τον ένα σύστημα στο άλλο.

1.2.7. - Διεπαφές Επικοινωνίας (Πόρτες)

Ο μικροελεγκτής PIC16F877 διαθέτει 5 παράλληλες πόρτες, με ετικέτες A-E. Οι ακροδέκτες της κάθε πόρτας μπορούν να χρησιμοποιηθούν όλοι μαζί ως ψηφιακή έισοδος/έξοδος ενός byte, ή ο κάθε ένας ξεχωριστά ως ψηφιακή είσοδος/έξοδος ενός bit. Οι πόρτες μπορεί να έχουν εναλλακτικές λειτουργίες, εκτός από ψηφιακοί είσοδοι/έξοδοι, οι οποίες φαίνονται στον Πίνακα 1.2. Η λειτουργία του κάθε ακροδέκτη ορίζεται μέσω αντίστοιχων καταχωρητών ελέγχου, ενώ στον πίνακα αναφέρεται η προκαθορισμένη λειτουργία τους κατά την εκκίνηση ή το μηδενισμό του μικροελεγκτή.

Πίνακας 1.2 – Λειτουργίες των διεπαφών επικονωνίας του PIC16F877

	Bits	Pins	Alternate function/s	Bit	Default
Port A	6	RA0–RA5	Analogue inputs Timer0 clock input Serial port slave select input	0,1,2,3,5 4 5	Analogue Input
Port B	8	RB0–RB7	External interrupt Low-voltage programming input Serial programming In-circuit debugging	0 3 6,7 6,7	Digital I/O
Port C	8	RC0–RC7	Timer1 clock input/output Capture/Compare/PWM SPI, I^2C synchronous clock/data USART asynchronous clock/data	0,1 1,2 3,4,5 6,7	Digital I/O
Port D	8	RD0–RD7	Parallel slave port data I/O	0–7	Digital I/O
Port E	3	RE0–RE2	Analogue inputs Parallel slave port control bits	0,1,2 0,1,2	Analogue Input

Χρήση των πορτών ως ψηφιακές εισόδους/εξόδους

Οι πόρτες B, C και D είναι προρυθμισμένες ως ψηφιακές διεπαφές και διαθέτουν 8 εισόδους/εξόδους. Οι πόρτες A και E με 6 και 3 εισόδους/εξόδους αντίστοιχα είναι προρυθμισμένες ως αναλογικές είσοδοι και για τη ρύθμισή τους ως ψηφιακές ακολουθείται μία συγκεκριμένη διαδικασία που περιγράφεται στην παράγραφο *1.2.8*.

Η χρήση των πορτών ως ψηφιακές εισόδους/εξόδους χωρίζεται σε δύο σκέλη. Το πρώτο σκέλος έχει να κάνει με τον ορισμό των ακροδεκτών της πόρτας ως εισόδους ή εξόδους και συμμετέχουν οι καταχωρητές TRISx, ενώ το δεύτερο με την ανάγνωση των εισόδων ή την εγγραφή στις εξόδους και συμμετέχουν οι καταχωρητές PORTx.

Στο πρώτο σκέλος πρέπει να καθορίσουμε τη χρήση του κάθε ακροδέκτη της πόρτας. Για να ορίσουμε έναν ακροδέκτη της πόρτας “x” ως είσοδο αρκεί να δώσουμε την τιμή 1 στο αντίστοιχο bit του καταχωρητή TRISx, ανάλογα, δίνοντας την τιμή 0 ο αντίστοιχος ακροδέκτης ορίζεται ως έξοδος. Π.χ. Αν θέλουμε να χρησιμοποιήσουμε τους ακροδέκτες 0, 4, 7 και 8 της πόρτας B σαν εισόδους ενώ τους υπόλοιπους σαν εξόδους, θα δώσουμε την τιμή 1 στα bits 0, 4, 7 και 8 του καταχωρητή TRISB, ενώ στα υπόλοιπα bits την τιμή 0. Εδώ να αναφέρουμε ότι για να τροποποιήσουμε τον καταχωρητή TRISx πρέπει πρώτα να μεταβούμε στην

τράπεζα μνήμης 1 (BANK_1) θέτοντας την κατάλληλη τιμή στα bits IRP:RP1:RP2 του καταχωρητή κατάστασης (STATUS Register) όπως φαίνεται στο *Σχήμα 1.9*.

Στο δεύτερο σκέλος περιγράφουμε την ανάγνωση/εγγραφή από/προς τις εισόδους/εξόδους, μία διαδικασία το ίδιο απλή. Για την ανάγνωση των εισόδων της πόρτας "x" αρκεί να διαβάσουμε τις τιμές των αντίστοιχων bits του καταχωρητή PORTx, ενώ για την εγγραφή προς τις εξόδους της πόρτας "x" αρκεί να δώσουμε την κατάλληλη τιμή στα αντίστοιχα bits του καταχωρητή PORTx. Π.χ. με βάση τις ρυθμίσεις που κάναμε στην πόρτα B του προηγούμενου παραδείγματος, έστω ότι έχουμε ένα διακόπτη στον ακροδέκτη 0 της πόρτας B (είσοδος) και ένα LED στον ακροδέκτη 6 (έξοδος). Θέλουμε όταν πατήσουμε το διακόπτη να ανάψουμε το LED, ενώ όταν τον αφήσουμε να σβήσει το LED. Για να το πετύχουμε αυτό θα διαβάσουμε τον καταχωρητή PORTB, θα ελέγξουμε την τιμή του bit_0 και εάν αυτό είναι 1 (έχει πατηθεί ο διακόπτης) θα θέσουμε την τιμή 1 στο bit_6 (για να ανάψει το LED), διαφορετικά αν το bit_0 είναι 0 (διακόπτης ανοικτός) θα θέσουμε την τιμή 0 στο bit_6 (για να σβήσει το LED).

1.2.8. - Η Αναλογική διασύνδεση και η μετατροπή του αναλογικού σήματος σε ψηφιακό

Στο σύστημα μας απαιτείται η μέτρηση της θερμοκρασίας, της υγρασίας και της τάσης του δικτύου. Για να πάρουμε αυτές τις μετρήσεις χρησιμοποιούμε αναλογικούς αισθητήρες. Η έξοδος όμως από τους αναλογικούς αισθητήρες είναι μία αναλογική τάση η οποία δεν είναι αντιληπτή από τον μικροελεγκτή, διότι αυτός αντιλαμβάνεται μόνο τη δυαδική αναπαράσταση αριθμών όπως κάθε ψηφιακή συσκευή. Για αυτό το λόγο απαιτείται η μετατροπή της τάσης στη δυαδική μορφή. Τη μετατροπή την αναλαμβάνουν ειδικά ολοκληρωμένα κυκλώματα – μετατροπείς σήματος από αναλογικό σε ψηφιακό (ADC – Analog to Digital Converters).

Επιλεγμένοι μικροελεγκτές, όπως αυτός που χρησιμοποιούμε, ενσωματώνουν αναλογικές εισόδους (PORTA, PORTE) οι οποίες είναι απευθείας συνδεδεμένες σε ένα μετατροπέα αναλογικού σήματος σε ψηφιακό (ADC). Η λειτουργία του μετατροπέα ελέγχεται μέσω των καταχωρητών ειδικού σκοπού ADCON0 *(Σχήμα 1.11)* και ADCON1 *(Σχήμα 1.12)*. Το αποτέλεσμα της μετατροπής είναι η δυαδική αναπαράσταση της αναλογικής τάσης εισόδου, έχει μήκος 10 bit και αποθηκεύεται στους καταχωρητές ADRESH (υψηλό byte) και ADRESL (χαμηλό byte). Για να ενεργοποιήσουμε τη μονάδα του μετατροπέα ώστε

να μπορέσουμε έπειτα να τον χρησιμοποιήσουμε πρέπει να θέσουμε το bit ADON που βρίσκεται στον καταχωρητή ADCON0 σε 1.

Λειτουργία του Μετατροπέα

Όπως αναφέραμε σε προηγούμενη παράγραφο οι αναλογικές αυτές είσοδοι έχουν και εναλλακτικές λειτουργίες (π.χ. μπορούν να χρησιμοποιηθούν ως ψηφιακές είσοδοι/έξοδοι), οπότε πρέπει να καθορίσουμε, σε πρώτη φάση, τη λειτουργία αυτή. Οι είσοδοι είναι συνδεδεμένοι με έναν επιλογέα λειτουργίας ο οποίος καθορίζει αν ο κάθε ακροδέκτης θα χρησιμοποιηθεί ως αναλογική ή ψηφιακή είσοδος ανάλογα με τον συνδυασμό των 4 bit που βρίσκονται στις θέσεις PCFG3:PCFG0 στον καταχωρητή ADCON1.

Οι αναλογικοί είσοδοι έπειτα τροφοδοτούνται στον πολυπλέκτη ο οποίος επιτρέπει τη μετατροπή μόνο μίας από τις εισόδους σε οποιαδήποτε χρονική στιγμή. Η επιλογή της εισόδου καθορίζεται από τα 3 bit που βρίσκονται στις θέσεις CHS0:CHS2 στον καταχωρητή ADCON0. Οπότε αν επιθυμούμε τη μετατροπή παραπάνω της μίας εισόδων, πρέπει μεταξύ των λειτουργιών της μετατροπής πρώτα να καθορίζουμε την επιθυμητή είσοδο τροποποιώντας αυτά τα bits. Η έναρξη της λειτουργίας της μετατροπής ενεργοποιείται θέτοντας το GO/DONE bit που βρίσκεται στον καταχωρητή ADCON0 σε 1, το οποίο τίθεται αυτόματα σε 0 σηματοδοτώντας το πέρας της διαδικασίας.

Χρονισμός του μετατροπέα

Η μετατροπή γίνεται συγκρίνοντας την αναλογική τάση εισόδου με την τάση αναφοράς (μέγιστη τιμή αναλογικής τάσης εισόδου που μπορεί να μετατραπεί – άνω άκρο του εύρους τιμών που αναπαριστά η δυαδική αναπαράσταση). Ο ακροδέκτης για την τάση αναφοράς (VREF+) καθορίζεται με βάση το σχήμα που επιλέχθηκε κατά τη φάση επιλογής των λειτουργιών των αναλογικών εισόδων.

Για παράδειγμα αν εφαρμόσουμε στον ακροδέκτη VREF+ τάση 4,096V τότε αυτή θα είναι και η μέγιστη τάση εισόδου που θα μπορεί να μετατρέψει και να αναπαραστήσει σε δυαδική μορφή ο μετατροπέας. Οπότε, με τα 10 bits (0-1023) του αποτελέσματος της μετατροπής, μπορούμε να αναπαραστήσουμε τιμές από 0V έως 4,096V, δηλαδή 4mV / bit (για κάθε αύξηση των 4mV στην τάση εισόδου θα αυξάνεται κατά 1 το αποτέλεσμα της μετατροπής - η δυαδική αναπαράσταση).

Η ταχύτητα της μετατροπής καθορίζεται από τα bit ADSC2 του καταχωρητή ADCON1 και ADSC1:ADSC0 του καταχωρητή ADCON0. Η διαδικασία για την εξαγωγή του τελικού αποτελέσματος γίνεται κατά προσέγγιση και έχει ως εξής: η τάση εισόδου εισάγεται σε ένα συγκριτή και εάν αυτή είναι μεγαλύτερη από το 50% του εύρους τιμών που αντιστοιχεί στα 10 bits (εύρος τάση αναφοράς – [(VREF+) – (VREF-)]) τότε το MSB γίνεται 1. Στη συνέχεια αφαιρείται από την αρχική τάση αυτή που αναπαρίσταται με το MSB και το αποτέλεσμα συγκρίνεται πάλι με το 50% του εύρους των 9 πλέον bit και εφόσον είναι μεγαλύτερη το επόμενο bit γίνεται 1 διαφορετικά γίνεται 0. Το ίδιο συμβαίνει και για τα 10 bits. Αυτή η διαδικασία απαιτεί σημαντικό χρόνο. Ο ελάχιστος χρόνος που απαιτείται είναι 1,6μs για κάθε ένα bit οπότε για τη μετατροπή σε 10 bit απαιτείται συνολικά 16μs. Η ταχύτητα του μετατροπέα πρέπει να επιλεχθεί έτσι ώστε να τηρείται τουλάχιστον αυτό το χρονικό περιθώριο, διαιρώντας κατάλληλα με 2, 8 ή 32 το χρονισμό του μικροελεγκτή.

Ο μικροελεγκτής στο κύκλωμά μας είναι χρονισμένος στα 4MHz, που σημαίνει ότι η περίοδος κάθε κύκλου είναι 0,25μs. Εμείς χρειαζόμαστε ένα χρονικό διάστημα της τάξης του 1,6μs για τη μετατροπή, οπότε εάν διαιρέσουμε τη συχνότητα του μικροελεγκτή με 8 η περίοδος κάθε κύκλου του μετατροπέα θα είναι 8 * 0,25 = 2 μs, το οποίο καλύπτει την ελάχιστη απαίτηση που έχουμε. Οπότε θέτουμε τα bit ADSC2:ADSC1:ADSC0 σε 0:0:1 (Πίνακας 1.3).

Πίνακας 1.3 – Ρύθμιση χρονισμού μετατροπέα

TAD vs. MAXIMUM DEVICE OPERATING FREQUENCIES (STANDARD DEVICES (F))

AD Clock Source (TAD)		Maximum Device Frequency
Operation	ADCS2:ADCS1:ADCS0	
2 TOSC	000	1.25 MHz
4 TOSC	100	2.5 MHz
8 TOSC	001	5 MHz
16 TOSC	101	10 MHz
32 TOSC	010	20 MHz
64 TOSC	110	20 MHz
RC(1, 2, 3)	x11	**(Note 1)**

Note 1: The RC source has a typical TAD time of 4 µs but can vary between 2-6 µs.

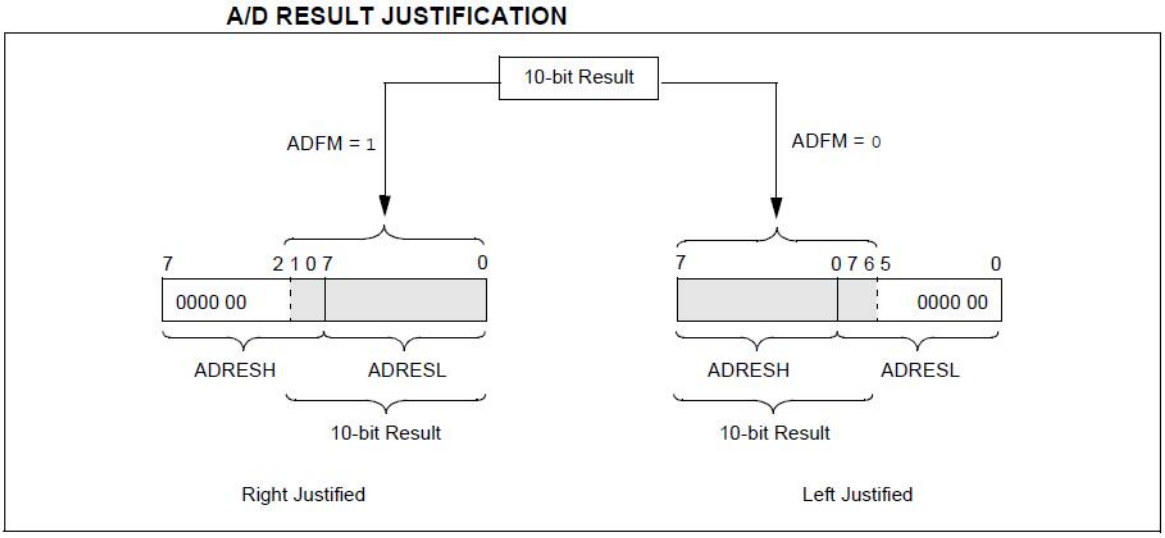
2: When the device frequencies are greater than 1 MHz, the RC A/D conversion clock source is only recommended for Sleep operation.

3: For extended voltage devices (LF), please refer to **Section 17.0 "Electrical Characteristics"**.

Καταχωρητές αποτελέσματος

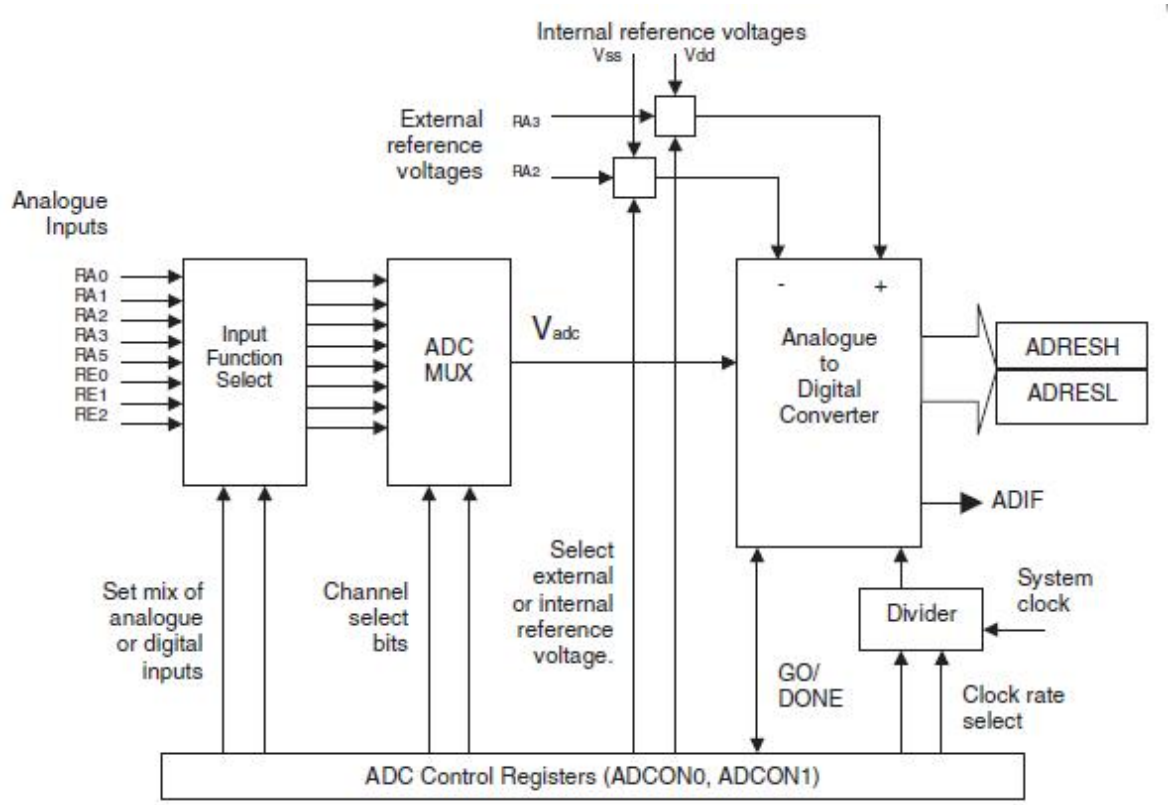
Μόλις ολοκληρωθεί η μετατροπή, το GO/DONE bit γίνεται 0 και το αποτέλεσμα τοποθετείται στο ζεύγος καταχωρητών ADRESH - ADRESL. Επειδή το αποτέλεσμα είναι μήκους 10 bit απαιτούνται δύο καταχωρητές (8 bits ο καθένας). Το πώς θα τοποθετηθεί το αποτέλεσμα στους 2 αυτούς καταχωρητές μπορεί να οριστεί με το bit ADFM που βρίσκεται στον καταχωρητή ADCON1 *(Σχήμα 1.16).*

Έχουμε δύο επιλογές την «αριστερή τοποθέτηση» (Left Justification - ADFM=0) και την «δεξιά τοποθέτηση» (Right Justification - ADFM=1). Στην $1^{η}$ περίπτωση το MSB του αποτελέσματος τοποθετείται στο MSB του ADRESH και δανειζόμαστε και τα 2 MSB bits από το ADRESL, ενώ στη $2^{η}$ το LSB του αποτελέσματος τοποθετείται στο LSB του ADRESL και δανειζόμαστε και τα 2 LSB του ADRESH. Για τη διατήρηση της ακριβέστερης προσέγγισης πρέπει να επεξεργαστούμε και τους 2 καταχωρητές. Γενικά η $2^{η}$ επιλογή φαίνεται η πιο κατάλληλη από άποψη πολυπλοκότητας.



Σχήμα 1.16 – Τοποθέτηση αποτελέσματος μετατροπής

Η διαδικασία μετατροπής του αναλογικού σήματος σε ψηφιακό παρουσιάζεται σχηματικά στο *Σχήμα 1.17*.



Σχήμα 1.17 – Διαδικασία μετατροπής αναλογικού σήματος σε ψηφιακό

1.2.9. - Η Σειριακή Επικοινωνία

Στη σειριακή επικοινωνία τα bits των κωδικοποιημένων χαρακτήρων αποστέλλονται το ένα μετά το άλλο, μέσα από ένα απλό φυσικό κανάλι μετάδοσης.

Στη σειριακή επικοινωνία, τις περισσότερες φορές εκπέμπεται πρώτο το λιγότερο σημαντικό bit (LSB - Least Significant Bit) του χαρακτήρα προς μετάδοση και σε μερικές περιπτώσεις προηγείται το πλέον σημαντικό bit του χαρακτήρα (MSB - Most Significant Bit). Η σειριακή επικοινωνία απαιτεί ένα καλώδιο για την αποστολή των δεδομένων, ένα καλώδιο για την παραλαβή των δεδομένων και ένα ακόμα για τη γείωση. Στην σύγχρονη σειριακή επικοινωνία υπάρχει και ένα σήμα χρονισμού για την μεταφορά των δεδομένων, ενώ στην ασύγχρονη δεν απαιτείται ο χρονισμός αυτός.

Η σειριακή επικοινωνία αρχικά αναπτύχθηκε για την επικοινωνία των «κουτών τερματικών» με τους κεντρικούς υπολογιστές (Mainframe computers) και μετέπειτα ακολούθησε η εφαρμογή της στις πόρτες COM των υπολογιστών για να παράσχει μια διεπαφή επικοινωνίας για το ποντίκι και άλλες περιφερειακές συσκευές. Το πιο γνωστό πρότυπο σειριακής επικοινωνίας είναι το RS232 το οποίο διαθέτει ενισχυμένη τάση για τη μετάδοση των δεδομένων ώστε να υποστηρίξει μεγαλύτερες αποστάσεις. Η σειριακή επικοινωνία χρησιμοποιείται κατά κόρον στην επικοινωνία των μικροελεγκτών PIC με τους ηλεκτρονικούς υπολογιστές. Επίσης χρησιμοποιείται για να συνδέσουμε ένα προγραμματιστή για τον PIC με τον υπολογιστή, ώστε να φορτώσουμε τον πηγαίο κώδικα στην μνήμη του μικροελεγκτή.

1.2.9.1. - Universal Synchronous Asynchronous Receiver Transmitter (USART)

Το USART μπορεί να ρυθμιστεί ως ένα αμφίδρομο ασύγχρονο σύστημα που μπορεί να επικοινωνήσει με τις διάφορες περιφερειακές συσκευές, όπως τα CRT τερματικά και τους προσωπικούς υπολογιστές, ή μπορεί να ρυθμιστεί να λειτουργήσει ως ένα ημι-αμφίδρομο σύγχρονο σύστημα που μπορεί να επικοινωνήσει με περιφερειακές συσκευές όπως κυκλώματα μετατροπής αναλογικού σήματος σε ψηφιακό και το αντίστροφο (A/D converters, ή D/A converters), σειριακές μνήμες EEPROM κτλ.

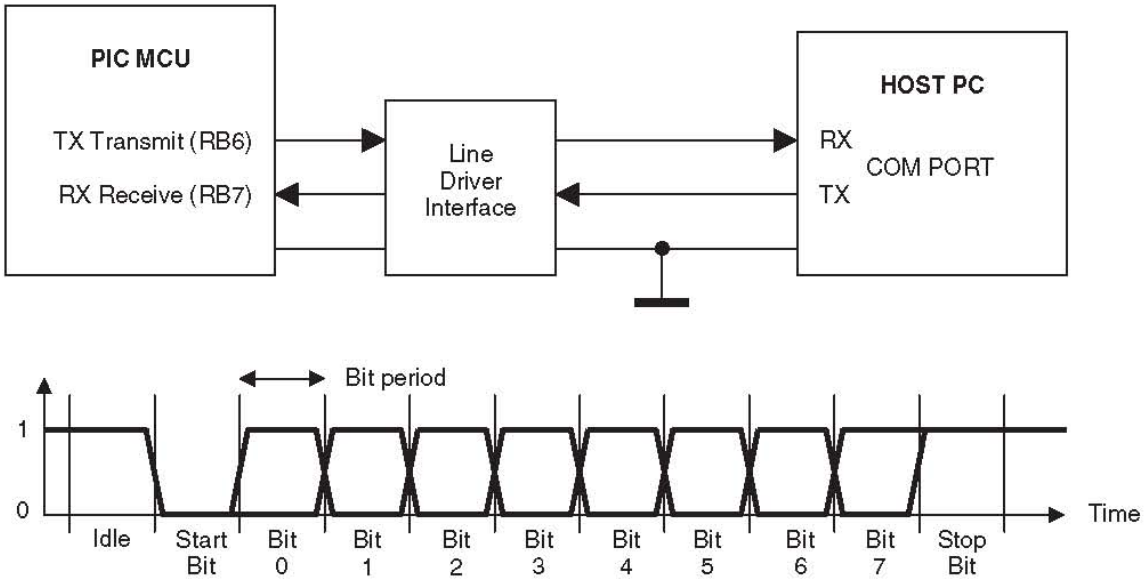
Το USART μπορεί να διαμορφωθεί να λειτουργήσει ως:

- Ασύγχρονο (πλήρως αμφίδρομη επικοινωνία)
- Σύγχρονο - Master (ημί αμφίδρομη επικοινωνία)
- Σύγχρονο - Slave (ημί αμφίδρομη επικοινωνία)

Στον μικροελεγκτή PIC 16F877, η σύγχρονη-ασύγχρονη επικοινωνία υποστηρίζεται διαμέσου των ακροδεκτών RB6 και RB7. Διαθέτει δύο καταστάσεις λειτουργίας: τη σύγχρονη (που χρησιμοποιεί ένα ξεχωριστό σήμα χρονισμού της μετάδοσης και λήψης δεδομένων) και την ασύγχρονη που δεν διαθέτει χρονισμό στην μετάδοση και λήψη δεδομένων. Η ασύγχρονη επικοινωνία είναι αυτή που χρησιμοποιείται συνήθως, αλλά βέβαια ο μικροελεγκτής μας υποστηρίζει και τη σύγχρονη επαρκώς. Στην ασύγχρονη επικοινωνία ο ακροδέκτης RB6 συμπεριφέρεται ως ο αποστολές των δεδομένων (TX), ενώ ο ακροδέκτης RB7 συμπεριφέρεται ως ο παραλήπτης των δεδομένων (RX). Τα δεδομένα συνήθως μεταδίδονται σε λέξεις των 8-bit, με πρώτο το ψηφίο του λιγότερου σημαντικού bit του χαρακτήρα που αποστέλλεται. Ο παραλήπτης θα πρέπει να διαβάζει την είσοδο του με τον ίδιο ρυθμό με τον οποίο αποστέλλονται τα δεδομένα και για αυτό χρησιμοποιούνται προκαθορισμένοι ρυθμοί επικοινωνίας. Φυσικά χρησιμοποιούνται διαφορετικές γραμμές για τη μετάδοση και την παραλαβή των δεδομένων, ούτως ώστε να μπορούν να πραγματοποιηθούν ταυτόχρονα οι δύο εργασίες της επικοινωνίας. Ο αποστολέας και ο παραλήπτης θα πρέπει να αρχικοποιηθούν στον ίδιο ρυθμό μετάδοσης, στον ίδιο αριθμό bits δεδομένων (το προεπιλεγμένο είναι 8 ψηφία) και στον ίδιο αριθμό από bits διακοπής (stop bits).

Για να συνδέσουμε τον μικροελεγκτή με έναν ηλεκτρονικό υπολογιστή, απαιτείται η μετατροπή των επιπέδων τάσης από 0V-5V σε +/-12V. Αυτό επιτυγχάνεται με μια εξωτερική συσκευή που μετατρέπει την έξοδο σε μια υψηλότερη, συμμετρική τάση *(Σχήμα 1.18)*. Αυτή η διαδικασία κρίνεται αναγκαία διότι το διαμορφωμένο σήμα εξασθενεί κατά τη μετάδοση του, λόγω της αντίστασης και της χωρητικότητας του καλωδίου.

Η βασική σειριακή διεπαφή RS232 λειτουργεί με υψηλότερη τάση ώστε το σήμα να μεταδοθεί σε μεγαλύτερες αποστάσεις προτού καλυφθεί από θόρυβο και παρεμβολές. Μια συνηθισμένη απόσταση που μπορεί να καλύψει η σειριακή διεπαφή RS232 είναι τα 10 μέτρα, με συμμετρική τάση που φτάνει τα +/-25V. Το σύνηθες βέβαια είναι τα +/-12V



Σχήμα 1.18 – Σύνδεση ηλεκτρονικού υπολογιστή με μικροελεγκτή

USART: Ασύγχρονη λειτουργία

Σε αυτή τη κατάσταση λειτουργίας η σειριακή επικοινωνία πραγματοποιείται βάση του γνωστού πρωτοκόλλου (ένα Start bit, 8 ή 9 data bits και ένα Stop bit). Η πλέον συνηθισμένη μορφή είναι αυτή με τα 8 data bits και ορίζεται δίνοντας την τιμή 0 στο bit TX9 που βρίσκεται στον καταχωρητή TXSTA. Ο μικροελεγκτής ενσωματώνει μια γεννήτρια διαμόρφωσης σήματος των 8 bits για να παράγει τους τυπικούς ρυθμούς μετάδοσης χρησιμοποιώντας τον ταλαντωτή. Ο αποστολέας και ο παραλήπτης είναι ανεξάρτητοι αλλά χρησιμοποιούν την ίδια μορφή δεδομένων και τον ίδιο ρυθμό μετάδοσης. Εμείς επιλέξαμε τον ρυθμό μετάδοσης της τάξεως των 9600 kbps, που σημαίνει ότι τα bits μεταδίδονται με ταχύτητα περίπου 10 Kbytes ανά δευτερόλεπτο, θέτοντας την τιμή 1 στο bit BRGH που βρίσκεται στον καταχωρητή TXSTA και δίνοντας την τιμή 25 στον καταχωρητή SPBRG με βάση τον Πίνακα 1.4.

Η ασύγχρονη λειτουργία μπορεί να επιλεχθεί θέτοντας την τιμή 0 στο bit SYNC που βρίσκεται στον καταχωρητή TXSTA (Σχήμα 1.13), ενώ η ενεργοποίηση της σειριακής πόρτας πραγματοποιείται θέτοντας την τιμή 1 στο bit SPEN που βρίσκεται στον καταχωρητή RCSTA *(Σχήμα 1.14)*. Η μονάδα ασύγχρονης μετάδοσης και λήψης, αποτελείται από τα παρακάτω σημαντικά συστατικά:

- τη γεννήτρια διαμόρφωσης σήματος (Baud Rate Generator)
- το κύκλωμα δειγματοληψίας

- τον ασύγχρονο αποστολέα δεδομένων (Asynchronous Transmitter)
- τον ασύγχρονο παραλήπτη δεδομένων (Asynchronous Receiver)

Πίνακας 1.4 – Αντιστοιχία ρυθμών μετάδοσης και τιμών SPBRG για 4Mhz και BRGH =1

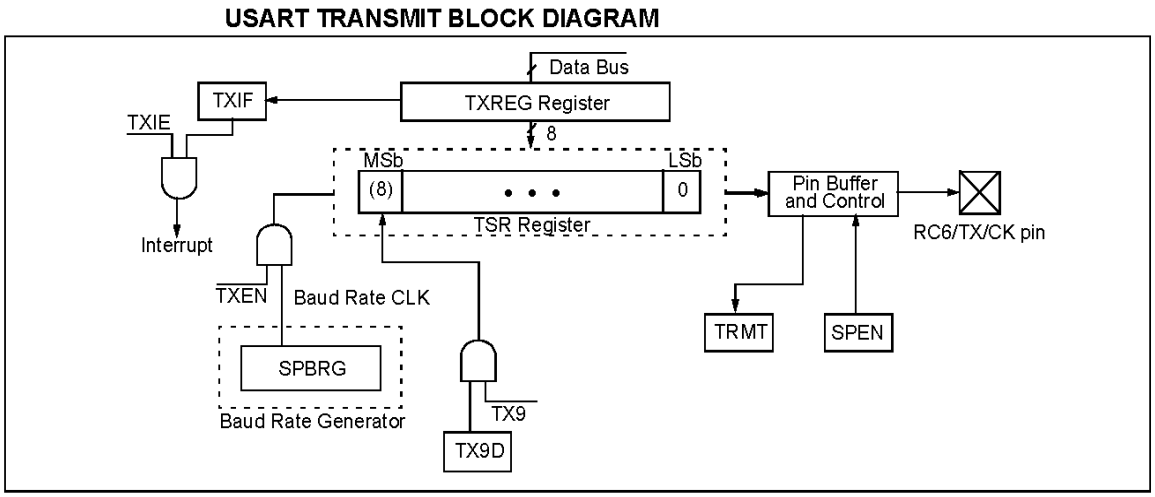
BAUD RATE (K)	Fosc = 4 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-
1.2	1.202	0.17	207
2.4	2.404	0.17	103
9.6	9.615	0.16	25
19.2	19.231	0.16	12
28.8	27.798	3.55	8
33.6	35.714	6.29	6
57.6	62.500	8.51	3
HIGH	0.977	-	255
LOW	250.000	-	0

Ασύγχρονη μετάδοση δεδομένων

Η μονάδα του USART που αφορά στη μετάδοση παρουσιάζεται σχηματικά στην παρακάτω εικόνα *(Σχήμα 1.19)*. Η καρδιά του αποστολέα είναι ο καταχωρητής TSR (Transmit Shift Register). Ο καταχωρητής αποκτά τα δεδομένα του από τον ενταμιευτή ανάγνωσης/εγγραφής, TXREG. Ο καταχωρητής αυτός αποκτά τα δεδομένα του προγραμματιστικά, ενώ ο καταχωρητής TSR δεν φορτώνει τα δεδομένα έως ότου το ψηφίο Stop μεταδοθεί από την προηγούμενη διαδικασία μετάδοσης. Μόλις μεταδοθεί το ψηφίο Stop, ο καταχωρητής TSR γεμίζει με νέα δεδομένα από τον καταχωρητή TXREG (εάν είναι διαθέσιμα). Από τη στιγμή που ο καταχωρητής TXREG μεταφέρει τα δεδομένα στον καταχωρητή TSR, ο καταχωρητής TXREG αδειάζει και σηματοδοτείται το flag bit, TXIF που βρίσκεται στον καταχωρητή PIR1 *(Σχήμα 1.15)*. Καθώς το flag bit, TXIF εντοπίζει τη κατάσταση του καταχωρητή TXREG, ένα άλλο bit, το TRMT που βρίσκεται στον καταχωρητή TXSTA, δείχνει την κατάσταση του καταχωρητή TSR. Το bit κατάστασης TRMT είναι ένα bit μόνο ανάγνωσης που παίρνει την τιμή 1 όταν αδειάσει ο καταχωρητής TSR.

Η μετάδοση ενεργοποιείται δίνοντας την τιμή 1 στο bit TXEN που βρίσκεται στον καταχωρητή TXSTA. Η πραγματική μετάδοση δεν θα εκκινήσει έως ότου ο καταχωρητής TXREG φορτώσει με τα δεδομένα και η γεννήτρια διαμόρφωσης σήματος (Baud Rate Generator - BRG) παράγει ένα «shift clock» *(Σχήμα 1.20)*. Η

μετάδοση μπορεί επίσης να ξεκινήσει εάν πρώτα φορτώσουμε τον καταχωρητή TXREG και μετά δώσουμε την τιμή 1 στο bit TXEN που βρίσκεται στον καταχωρητή TXSTA. Φυσιολογικά, όταν η μετάδοση πρωτοξεκινήσει ο καταχωρητής TSR είναι άδειος. Σε αυτό το σημείο η μεταφορά των δεδομένων στον καταχωρητή TXREG θα προκαλέσει την άμεση μεταφορά τους στον καταχωρητή TSR και το άδειασμα του καταχωρητή TXREG.

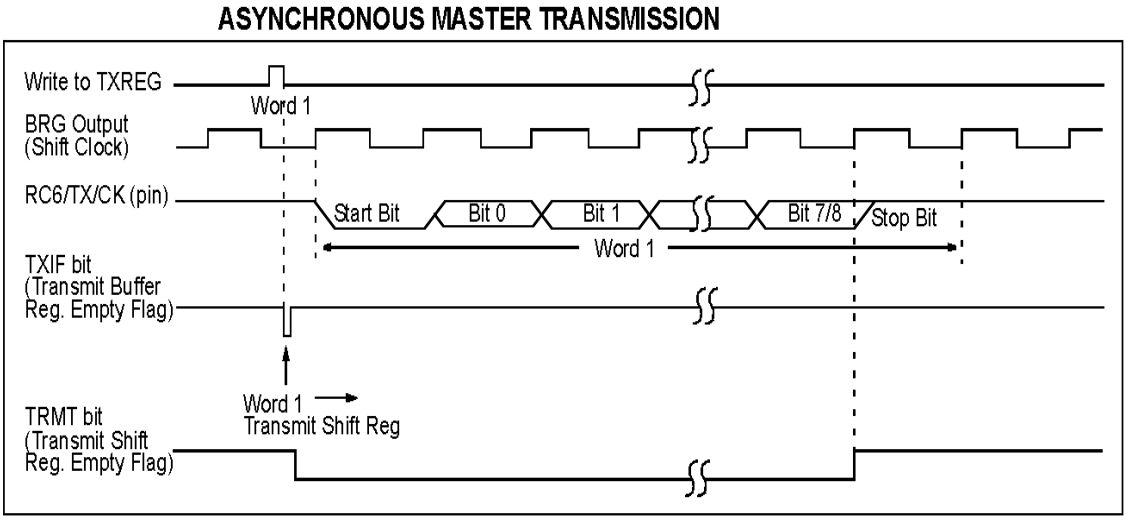


Σχήμα 1.19 – Διαδικασία σειριακής μετάδοσης

Για να πραγματοποιήσουμε την ασύγχρονη μετάδοση, θα πρέπει να ακολουθήσουμε τα παρακάτω βήματα:

1. Ενεργοποίηση της σειριακής πόρτας δίνοντας την τιμή 1 στο bit SPEN που βρίσκεται στον καταχωρητή RCSTA και επιλογή της ασύγχρονης λειτουργίας δίνοντας την τιμή 0 στο bit SYNC που βρίσκεται στον καταχωρητή TXSTA.
2. Εάν προτιμηθεί υψηλής ταχύτητας ρυθμός μετάδοσης, δίνουμε την τιμή 1 στο bit BRGH που βρίσκεται στον καταχωρητή TXSTA. Αρχικοποίηση του καταχωρητή SPBRG με την κατάλληλη τιμή βάση του Πίνακα 1.4.
3. Εάν επιθυμούμε μετάδοση 9 bits, θα πρέπει να δώσουμε την τιμή 1 στο bit TX9 που βρίσκεται στον καταχωρητή TXSTA.
4. Ενεργοποίηση της μετάδοσης δίνοντας την τιμή 1 στο bit TXEN που βρίσκεται στον καταχωρητή TXSTA
5. Εάν έχει επιλεχθεί η μετάδοση των 9-bit το ένατο bit θα πρέπει να φορτωθεί στο bit TX9D που βρίσκεται στον καταχωρητή TXSTA.

6. Φόρτωση των δεδομένων στον καταχωρητή TXREG (εκκίνηση της μετάδοσης).



Σχήμα 1.20 – Ασύγχρονη σειριακή μετάδοση

1.3. - Περιφερειακά εξαρτήματα

1.3.1. - Αναλογικοί και ψηφιακοί αισθητήρες

Οι Αισθητήρες χρησιμοποιούνται κατά κόρον σε ηλεκτρονικά συστήματα για να μετρούν θερμοκρασίες αλλά και για τη συλλογή ενός μεγάλου συνόλου άλλων αναλογικών περιβαλλοντολογικών πληροφοριών (ήχο, υγρασία, πίεση, κίνηση, τάση κτλ). Για αυτό το λόγο και υπάρχουν διάφορες κατηγορίες αισθητήρων, όπως: επαγωγικοί αισθητήρες, αισθητήρες χωρητικότητας, φωτοηλεκτρικοί αισθητήρες, αισθητήρες πίεσης, αισθητήρες μαγνητικών πεδίων, χημικοί αισθητήρες, υπερηχητικοί αισθητήρες κτλ.

1.4. - Χαρακτηριστικά και Κατηγορίες Αισθητήρων

Οι πιο γνωστές τεχνολογίες αισθητήρων, είναι:

Α) **Thermocouples (Θερμοζεύγη):** κατασκευάζονται από την ένωση δύο συρμάτων ανόμοιων μετάλλων. Στην επαφή παράγεται μια τάση που είναι περίπου ανάλογη προς τη θερμοκρασία. Υπάρχουν διάφοροι τύποι θερμοηλεκτρικών ζευγών. Τα ολοκληρωμένα κυκλώματα MAX6674 και MAX6675 της Maxim εκτελούν και ρυθμίζουν τις λειτουργίες των σημάτων για τα θερμοηλεκτρικά ζεύγη, απλοποιώντας σημαντικά τις μετρήσεις.

Β) **RTDs:** Τα RTDs είναι ουσιαστικά αντιστάτες (που συχνά κατασκευάζονται από λευκόχρυσο) των οποίων η αντίσταση ποικίλλει, ανάλογα με τη θερμοκρασία. Τα χαρακτηριστικά περιλαμβάνουν το μεγάλο εύρος μετρήσεων θερμοκρασίας (μέχρι 750°C), την άριστη ακρίβεια, τη λογική γραμμικότητα.

Γ) **Thermistors:** είναι αντιστάτες εξαρτώμενοι από τη θερμοκρασία των αγώγιμων υλικών που κατασκευάζονται. Στα χαρακτηριστικά τους περιλαμβάνονται το ικανοποιητικό εύρος μετρήσεων θερμοκρασίας (μέχρι 150°C), το χαμηλό-μέτριο κόστος (ανάλογα με την ακρίβεια) και η φτωχή αλλά προβλέψιμη γραμμικότητα.

Δ) **IC temperature sensors:** είναι ολοκληρωμένα κυκλώματα πλήρως βασισμένα στο πυρίτιο που ενσωματώνουν αισθητήρες και έχουν έξοδο είτε αναλογική ή ψηφιακή. Τα χαρακτηριστικά τους περιλαμβάνουν το ικανοποιητικό εύρος μετρήσεων θερμοκρασίας (μέχρι 150°C), το χαμηλό κόστος, την άριστη γραμμικότητα, καθώς και πρόσθετα χαρακτηριστικά γνωρίσματα όπως συγκριτές, και ψηφιακές διεπαφές. Τα ψηφιακά σχήματα είναι πολυάριθμα και περιλαμβάνουν διατάξεις με 1, 2, 3 ή 4 ακροδέκτες.

1.4.1. - Ο Αναλογικός Αισθητήρας LM35

Ο LM35 είναι ένα ολοκληρωμένο κύκλωμα ενός αισθητήρα θερμοκρασίας υψηλής ακρίβειας, με τάση εξόδου γραμμικά ανάλογη με τη θερμοκρασία σε βαθμούς Κελσίου. Ο LM35 είναι ιδανικός για θερμοκρασίες δωματίου (-55 °C έως +150 °C), δεν απαιτεί επιπλέον βαθμονόμηση (calibration) και έχει χαμηλό κόστος.

Χαρακτηριστικά:

- Διαβαθμισμένο σε °C (βαθμούς Κελσίου)
- Γραμμική έξοδος: 10mV/°C
- 0,5°C ακρίβεια (στους 25°C)
- Τάση λειτουργίας 4V - 30V
- Χαμηλή εκπομπή θερμότητας (0,008°C)
- Χαμηλή αντίσταση εξόδου

Η τάση εξόδου διαμορφώνεται ως εξής:

- +150°C: 1500mV
- +25°C: 250mV
- 0°C: 0mV

Λόγω της γραμμικότητας της εξόδου δεν απαιτείται κάποια επιπλέον πολύπλοκη ενέργεια για να υπολογίσουμε τη μέτρηση της θερμοκρασίας. Αρκεί να χρησιμοποιήσουμε τον τύπο:

“y” °C = “x” mV / 10

1.4.2. - Ο Μετατροπέας επιπέδων τάσης σειριακής διεπαφής MAX232CPE

Όπως έχει ήδη αναφερθεί στην παράγραφο 1.2.8.1 για την επικοινωνία του ηλεκτρονικού υπολογιστή με το μικροελεγκτή μέσω της σειριακής πόρτας, βάση του πρωτοκόλλου RS232, απαιτείται η μεσολάβηση ενός ολοκληρωμένου κυκλώματος το οποίο αναλαμβάνει το μετασχηματισμό των επιπέδων τάσης από 0V–5V σε +/-12V. Αυτό κρίνεται απαραίτητο διότι στο μικροελεγκτή το λογικό 0 αναπαρίσταται με τάση 0V και το λογικό 1 με τάση 5V, ενώ στο πρωτόκολλο RS232 των σειριακών θυρών του υπολογιστή το λογικό 0 αναπαρίσταται με τάση από -12V έως -5V και το λογικό 1 με τάση από +5V έως +12V. Ένα ενδεικτικό ολοκληρωμένο είναι και το MAX232CPE της MAXIM.

ΚΕΦΑΛΑΙΟ 2
Δημιουργία Λογισμικού

2.1. - Το περιβάλλον προγραμματισμού

Οι Μικροελεγκτές PICmicro® υποστηρίζονται από ένα ευρύ φάσμα εργαλείων ανάπτυξης υλικού και λογισμικού όπως:

- Ενοποιημένο Περιβάλλον Ανάπτυξης
  - MPLAB ® IDE Software
- Μεταφραστές/Μεταγλωττιστές/Linkers
  - MPASMTM Assembler
  - MPLAB C17 and MPLAB C18 C Compilers
  - MPLINKTM Object Linker/MPLIBTM Object Librarian
  - MPLAB C30 C Compiler
  - MPLAB ASM30 Assembler/Linker/Library
- Προσομοιωτές
  - MPLAB SIM Software Simulator
  - MPLAB dsPIC30 Software Simulator
- Εξομοιωτές
  - MPLAB ICE 2000 In-Circuit Emulator
  - MPLAB ICE 4000 In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD 2
  - Προγραμματιστές Συσκευών
  - PRO MATE® II Universal Device Programmer
  - PICSTART® Plus Development Programmer
  - Low Cost Demonstration Boards

- PICDEMTM 1 Demonstration Board
- PICDEM. net TM Demonstration Board
- PICDEM 2 Plus Demonstration Board
- PICDEM 3 Demonstration Board
- PICDEM 4 Demonstration Board
- PICDEM 17 Demonstration Board
- PICDEM 18R Demonstration Board
- PICDEM LIN Demonstration Board
- PICDEM USB Demonstration Board

- Evaluation Kits
  - KEELOQ®
  - PICDEM MSC
  - microID®
  - CAN
  - PowerSmart®
  - Analog

2.2. - MPLAB IDE 8.30

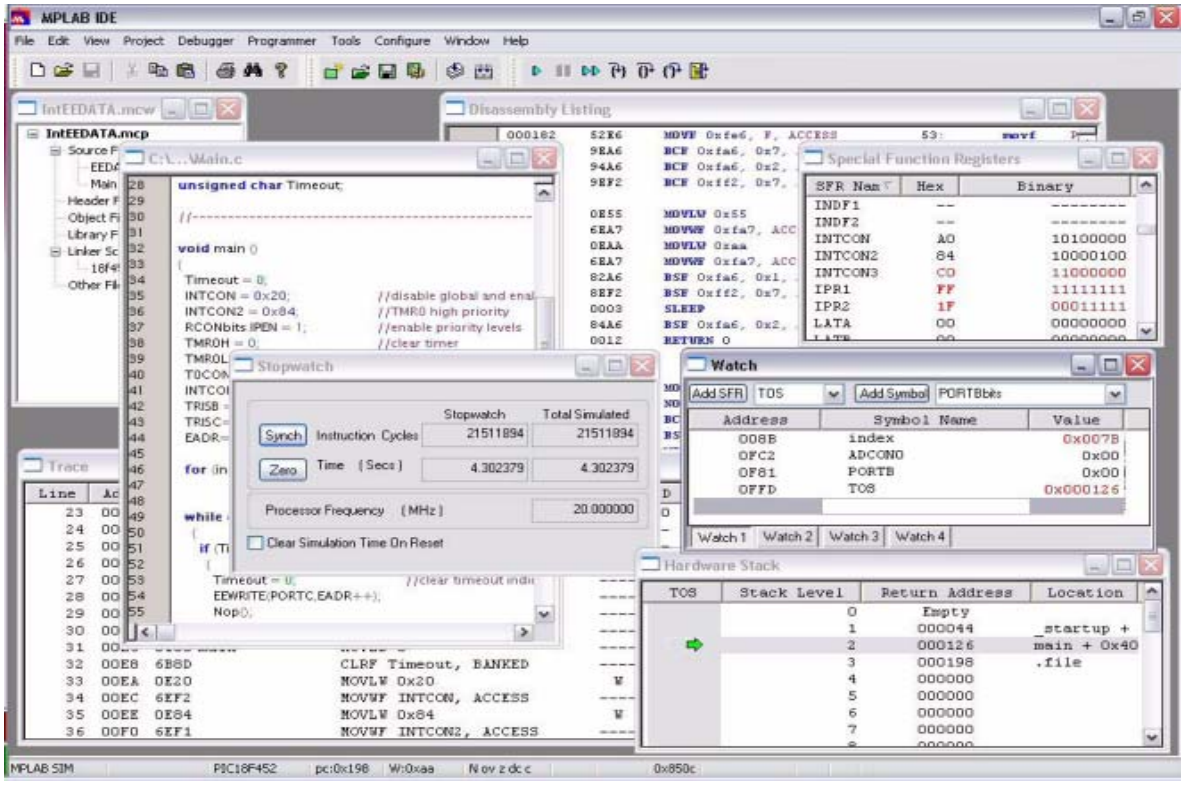
Το ολοκληρωμένο περιβάλλον ανάπτυξης MPLAB, είναι ένα ελεύθερο σύνολο υποστηρικτικών εργαλείων που υποστηρίζει την ανάπτυξη προγραμμάτων για τους μικροελεγκτές της Microchip, PIC® και dsPIC®. Το περιβάλλον MPLAB IDE τρέχει ως μια 32-bit εφαρμογή σε Λειτουργικό Σύστημα MS Windows®, είναι πολύ εύκολο στην χρήση του και περιλαμβάνει διάφορα ελεύθερα εργαλεία που διευκολύνουν την γρήγορη ανάπτυξη προγραμμάτων και την υψηλού επιπέδου αποσφαλμάτωση τους. Το περιβάλλον MPLAB IDE προσφέρεται επίσης ως ένα απλό γραφικό περιβάλλον χρήστη για την υποστήριξη λογισμικού και υλικού και άλλων κατασκευαστών πέραν της MicroChip.

Το Ολοκληρωμένο περιβάλλον προγραμματισμού MPLAB, μας παρέχει την δυνατότητα εύκολης ανάπτυξης λογισμικού που προηγουμένως δεν υπήρχε στην αγορά των μικροελεγκτών των 8/16-bit. Το Περιβάλλον MPLAB IDE περιέχει:

- Μια Διεπαφή για τα εργαλεία Αποσφαλμάτωσης:
  - Προσομοιωτής (simulator)
  - Προγραμματιστής (programmer- sold separately)
  - Εξομοιωτής (emulator)

- Αποσφαλμάτωση εντός συστήματος (in-circuit debugger)

- Έναν πλήρη συντάκτη με χρωματισμένο περιεχόμενο
- Διαχειριστή πολλαπλών projects
- Υψηλού επιπέδου αποσφαλμάτωση κώδικα



Σχήμα 2.1 – Το περιβάλλον προγραμματισμού

Το MPLAB υποστηρίζει πολλαπλά εργαλεία αποσφαλμάτωσης σε ένα περιβάλλον ανάπτυξης. Από τους χαμηλού κόστους προσομοιωτές, τους αποσφαλματωτές εντός συστήματος μέχρι και τους πλήρεις εξομοιωτές. Το περιβάλλον MPLAB περιλαμβάνει έναν ενσωματωμένο συντάκτη κειμένου, σχεδιασμένο για τη συγγραφή των προγραμμάτων για τους μικροελεγκτές PIC. Φυσικά για τον προγραμματισμό θα απαιτηθεί η γλώσσα προγραμματισμού assembly, χρησιμοποιώντας το σύνολο εντολών που αντιστοιχεί στον επιλεγμένο μικροελεγκτή που θα χρησιμοποιηθεί, συν τις έξτρα οδηγίες που θα απαιτηθούν προς τον μεταφραστή (assembler).

2.3. - Η Assembly

Μια τυπική εντολή αναλύεται στον Πίνακα 2.1. Η εντολή γλώσσας μηχανής παρέχει τις πληροφορίες προς την μονάδα εκτέλεσης του επεξεργαστή για να εκτελέσει τη κατάλληλη ενέργεια (move, calculate, test, κ.α). Θα μπορούσε να

εισαχθεί στο κοινό δυαδικό σύστημα αλλά αυτό θα απαιτούσε από εμάς να συμβουλευόμαστε τον κώδικα κάθε φορά. Μια πιο φιλική προς τον χρήστη επιλογή, είναι τα μνημονικά εντολών. Τα μνημονικά χρησιμοποιούνται για να αναπαραστήσουν τον κωδικό-διεργασίας και τους τελεστές. Ο μεταφραστής τα αντικαθιστά στο μεταφρασμένο αρχείο με τον κατάλληλο δυαδικό κώδικα για να παράγει το τελικό κώδικα μηχανής του προγράμματος (δεκαεξαδικό αρχείο). Στον Πίνακα 2.2 παρατηρούμε πως μια 14-bit εντολή μπορεί να διαιρεθεί σε τρία λειτουργικά μέρη. Τα πρώτα δύο bits είναι πάντα μηδέν, δηλαδή τα αχρησιμοποίητα bits 14 και 15, τα επόμενα τέσσερα δίνουν την εντολή BTFSS, τα επόμενα τρία προσδιορίζουν το bit που θα εξεταστεί (0) και τα τελευταία επτά τη διεύθυνση του καταχωρητή αρχείων (08).

Πίνακας 2.1 – Σχήμα εντολής

List File Column	Example Content	Meaning
0	000C	Memory location at which machine code instruction is stored
1	1C08	Machine code instruction, including op-code and operands
2	00065	Source code line number
3	start	Address label marking jump destination
4	BTFSS	Instruction mnemonic
5	PORTD,Inres	Instruction operand labels
6	;Test reset button	Comment delimited by semi-colon

Πίνακας 2.2 – Δομή μιας εντολής

Label	Hex	Binary	Meaning	Range
start	000C	0000 0000 0000 1100	Program memory address label	0000 – 1FFF(8K)
BTFSS		00 01 11-- ---- ----	Op-code (bits 13,12,11,10 only)	-
PORTD	1C08	-- -- ---- -000 1000	File register address = 08	00 – 7F (128)
Inres		-- -- --00 0--- ----	File register bit = 0	0 – 7

Άλλες εντολές δεν έχουν απαραίτητα όλες αυτούς τους τελεστές. Η εντολή CLRW δεν έχει καθόλου τελεστές και η εντολή CLRF έχει έναν, τη διεύθυνση του

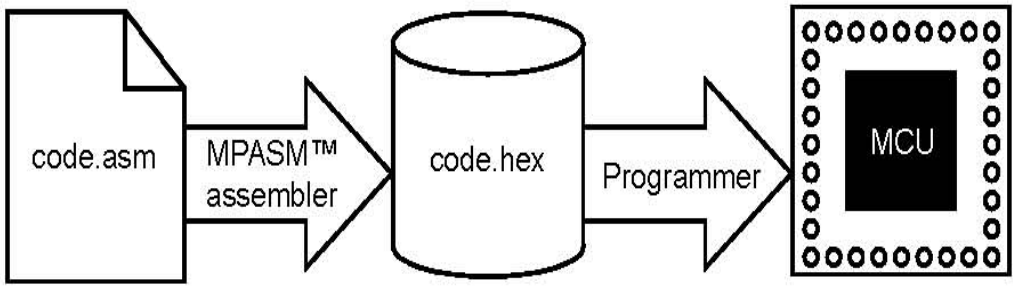
καταχωρητή. Στις περισσότερες περιπτώσεις χειρισμών τέτοιων καταχωρητών, ο προορισμός του αποτελέσματος επιστρέφεται στο W θέτοντας το bit 7=0.

Οι εντολές, στο αρχείο του πηγαίου κώδικα, τοποθετούνται στη δεύτερη και τρίτη στήλη, που για σαφήνεια διαχωρίζονται από ένα tab, αλλά και το κενό είναι αποδεκτό από τον μεταφραστή. Τα σχόλια είναι οριοθετημένα από ερωτηματικά και πρέπει να περιγράψουν την επίδραση των εντολών, παρά να επαναλαμβάνουμε τη σημασία των μνημονικών και των ετικετών. Επιθυμητό είναι ένα μεγάλο μπλοκ από σχόλια στην κορυφή του προγράμματος, αλλά αυτό πρέπει να είναι σε αναλογία με την πολυπλοκότητα του προγράμματος. Το όνομα του πηγαίου κώδικα, ο συγγραφέας και η ημερομηνία ή ο αριθμός έκδοσης είναι ουσιαστικά. Ακολουθεί μια περιγραφή του προγράμματος και οι λεπτομέρειες του επεξεργαστή που θα χρησιμοποιηθεί. Πρέπει να συμπεριλάβουμε επίσης τις λεπτομέρειες του αντικειμενικού σκοπού του υλικού, ειδικά τις διαδικασίες I/O (εισόδου-εξόδου). Έπειτα ακολουθείται από την οδηγία επιλογής του επεξεργαστή (MCU) και την τιμή του καταχωρητή διαμόρφωσης. Αυτά είναι οδηγίες του μεταφραστή και δεν μετατρέπονται σε εντολές γλώσσας μηχανής, όπως μπορούμε να δούμε στην αριστερή στήλη του αρχείου. Η οδηγία του PROCESSOR (επεξεργαστή) λέει στον μεταφραστή ποιος επεξεργαστής θα χρησιμοποιηθεί, προφανώς γιατί υπάρχουν κάποιες παραλλαγές στον αριθμό των πορτών και των διευθύνσεων. Η σειρά μικροελεγκτών 18FXXX έχουν ένα πιο εκτενές σύνολο εντολών. Η οδηγία __CONFIG θέτει το «προγραμματιστικό φυτίλι» στον μικροελεγκτή, το οποίο δεν μπορεί να αλλάξει εκτός με τον επαναπρογραμματισμό του. Η τιμή του καταχωρητή διαμόρφωσης (3733h) και η θέση μνήμης του (2007h) εμφανίζονται στην αριστερή στήλη του αρχείου καταλόγων. Η μόνη οδηγία που είναι απολύτως απαραίτητη είναι η END, που προσδιορίζει το τέλος του προγράμματος στον μεταφραστή.

Ο μεταφραστής MPASM μπορεί να χρησιμοποιηθεί με 2 τρόπους:

- Για να παράγει τον απόλυτο κώδικα που θα εκτελεστεί απευθείας από τον μικροελεγκτή.
- Για να παράγει τον κώδικα (*relocatable code*) που θα συνδεθεί (linked) μετέπειτα με άλλα μεταφρασμένα, ή μεταγλωττισμένα πηγαία αρχεία.

Ο απόλυτος κώδικας είναι η προεπιλεγμένη έξοδος του μεταφραστή MPASM. Αυτή η διαδικασία φαίνεται στην παρακάτω εικόνα *(Σχήμα 2.2)*.



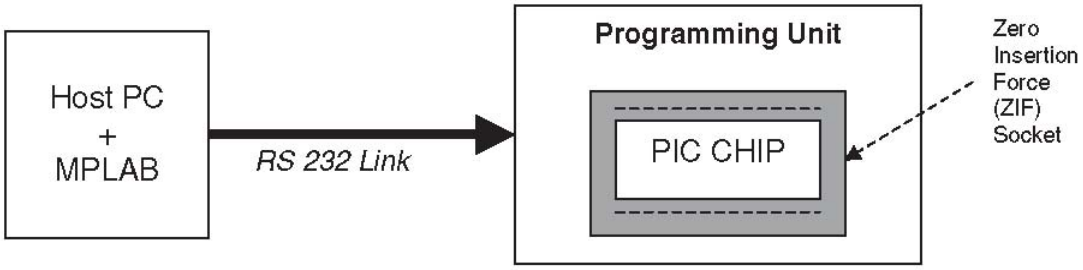
Σχήμα 2.2 – Παραγωγή απόλυτου κώδικα από το μεταφραστή

Όταν ένα πηγαίο αρχείο μεταφράζεται κατά αυτό τον τρόπο, όλες οι μεταβλητές και οι ρουτίνες που χρησιμοποιούνται από το πηγαίο κώδικα, θα πρέπει να δηλωθούν μέσα στο ίδιο το πηγαίο αρχείο, ή σε άλλα αρχεία που δηλώνονται στο πηγαίο αρχείο. Εάν η assembly μεταφραστεί επιτυχώς, παράγεται ένα δεκαεξαδικό αρχείο που περιέχει τον εκτελέσιμο κώδικα μηχανής για την προοριζόμενη συσκευή. Το αρχείο μπορεί να χρησιμοποιηθεί σε ένα αποσφαλματωτή για τον έλεγχο της εκτέλεσης του κώδικα, ή με ένα προγραμματιστή συσκευής για τον προγραμματισμό του μικροελεγκτή.

2.4. - Το κύκλωμα του προγραμματιστή και η φόρτωση του προγράμματος στη μνήμη του μικροελεγκτή

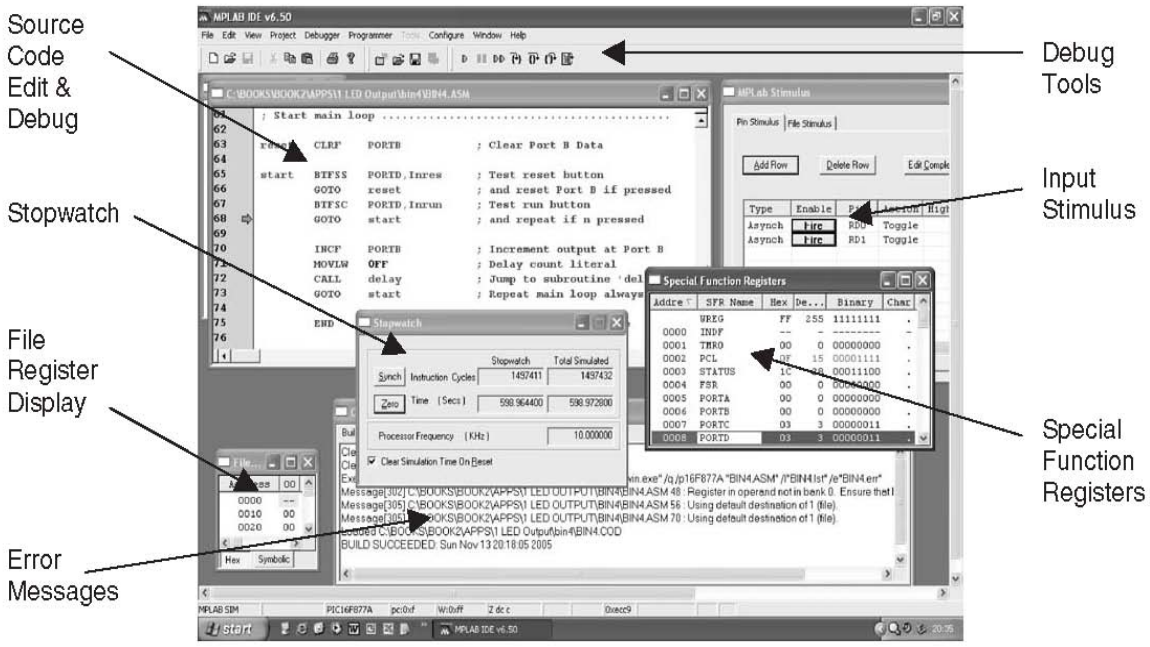
Ο πηγαίος κώδικας μπορεί να δημιουργηθεί σε οποιοδήποτε υπολογιστή, χρησιμοποιώντας έναν απλό κειμενογράφο, ή μέσω ενός παραθύρου επεξεργασίας του MPLAB (the standard development system), έπειτα μεταφράζεται και φορτώνεται στο chip με τη βοήθεια του κατάλληλου εξοπλισμού. Το υλικό (hardware) σύστημα ανάπτυξης που συνήθως χρησιμοποιείται, φαίνεται στο *Σχήμα 2.3*. Περιέχει έναν υπολογιστή και μια μονάδα προγραμματισμού που συνδέεται με τον υπολογιστή με σειριακή σύνδεση RS232. Ο υπολογιστής χρησιμοποιεί το λογισμικό MPLAB και αφού γραφεί ο κώδικας και μεταφραστεί στο περιβάλλον του, έπειτα το πρόγραμμα μπορεί να φορτωθεί, αφού βέβαια τοποθετηθεί ο μικροελεγκτής στην μονάδα προγραμματισμού. Το πρωτόκολλο RS232 είναι το πιο απλό σειριακό μοντέλο δεδομένων. Ο μικροελεγκτής προγραμματίζεται διαμέσου των pin (ακροδεκτών), RB6 (clock) και RB7 (δεδομένα).

Η Μονάδα προγραμματισμού διαθέτει μια υποδοχή (socket) που τοποθετείται ο μικροελεγκτής και περιέχει επίσης έναν ακόμα μικροελεγκτή PIC για τον έλεγχο του προγραμματισμού. Ωστόσο, ο μικροελεγκτής μπορεί να



Σχήμα 2.3 – Σύστημα ανάπτυξης

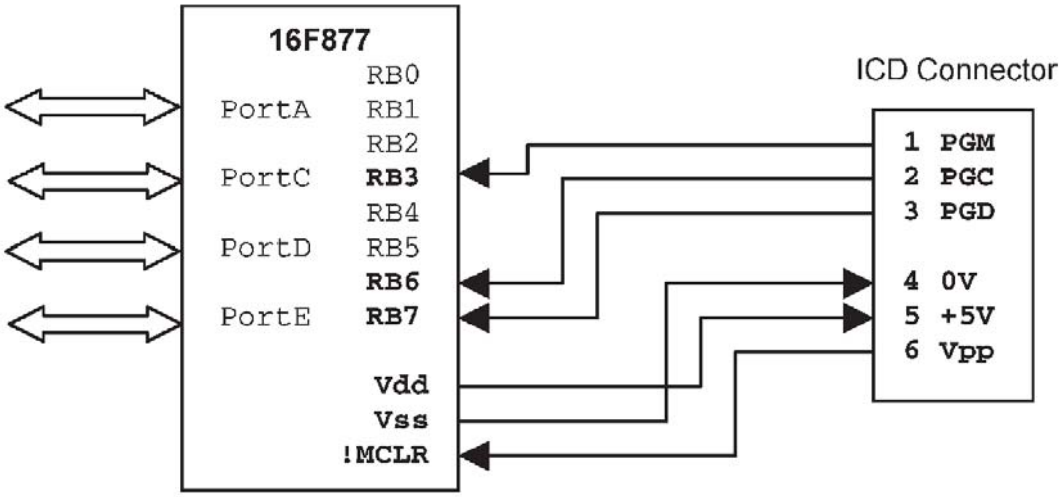
προγραμματιστεί να τρέξει σε λειτουργία ICD που σημαίνει ότι η εκτέλεση του προγράμματος από τον μικροελεγκτή, μπορεί να ελεγχθεί μέσα από το περιβάλλον του MPLAB *(Σχήμα 2.4)*. Το πρόγραμμα μπορεί να σταματήσει, να επανεκκινήσει και να προχωρήσει βήμα προς βήμα την εκτέλεση του, όπως γίνεται όταν προσομοιώνεται η λειτουργία του κώδικα. Πλέον το πρόγραμμα μπορεί να ελεγχθεί μέσα στο ίδιο το υλικό για το οποίο προορίζεται, μια ρεαλιστική και πανίσχυρη επιλογή. Αφού δημιουργηθεί το πρόγραμμα θα πρέπει να φορτωθεί στην μνήμη του μικροελεγκτή. Ο παραδοσιακός τρόπος είναι να συνδέσουμε τον μικροελεγκτή σε μια ξεχωριστή μονάδα προγραμματισμού. Έπειτα το πρόγραμμα φορτώνεται μέσω του εργαλείου προγραμματισμού του MPLAB. Βέβαια αυτό έχει και τα μειονεκτήματα του, καθώς η μεταφορά του μικροελεγκτή μπορεί να προκαλέσει φυσική, ή ηλεκτρική ζημιά. Για αυτό το λόγο η Microchip παρουσίασε μια «εντός του συστήματος» (in-circuit) μέθοδο προγραμματισμού, η οποία είναι οικονομική και παρέχει την δυνατότητα αποσφαλμάτωσης του κώδικα καθώς αυτός τρέχει στο αληθινό υλικό.



Σχήμα 2.4 – Έλεγχος εκτέλεσης προγράμματος

Η «εντός του συστήματος» αποσφαλμάτωση (In-Circuit Debugging) χρησιμοποιεί τους ίδιους ακροδέκτες προγραμματισμού του μικροελεγκτή, αλλά συνδέονται με ένα ξέχωρο κομμάτι, ένα προγραμματιστή των 6 ακροδεκτών. Έπειτα αυτό το κομμάτι συνδέεται ανάμεσα στον υπολογιστή και την συσκευή και τέλος το πρόγραμμα φορτώνεται στον μικροελεγκτή.

Η αποσφαλμάτωση «εντός του συστήματος» επιτρέπει σε ένα πρόγραμμα να τρέξει μέσα από το περιβάλλον του MPLAB, να αποσφαλματωθεί μέσα στο υλικό με απλό βηματισμό εκτέλεσης και σημεία ελέγχου. Η εντολή NOP (No Operation) θα πρέπει να τοποθετηθεί στην θέση 0000 ($1^{η}$ εντολή του προγράμματος), για να επιτρέψει τη πρόσβαση στην αποσφαλμάτωση του κώδικα, προτού το πρόγραμμα ξεκινήσει την εκτέλεση του.

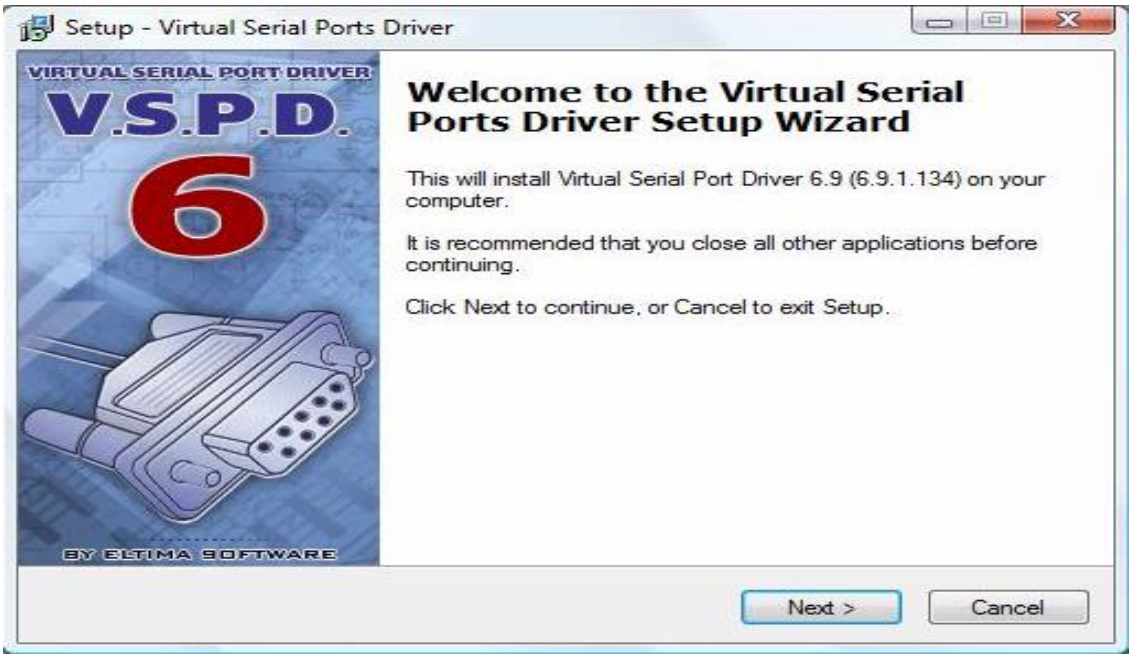


Σχήμα 2.5 – Συνδεσμολογία ICD

2.5. - Eltima Virtual Serial Port Driver 6.9.1.134

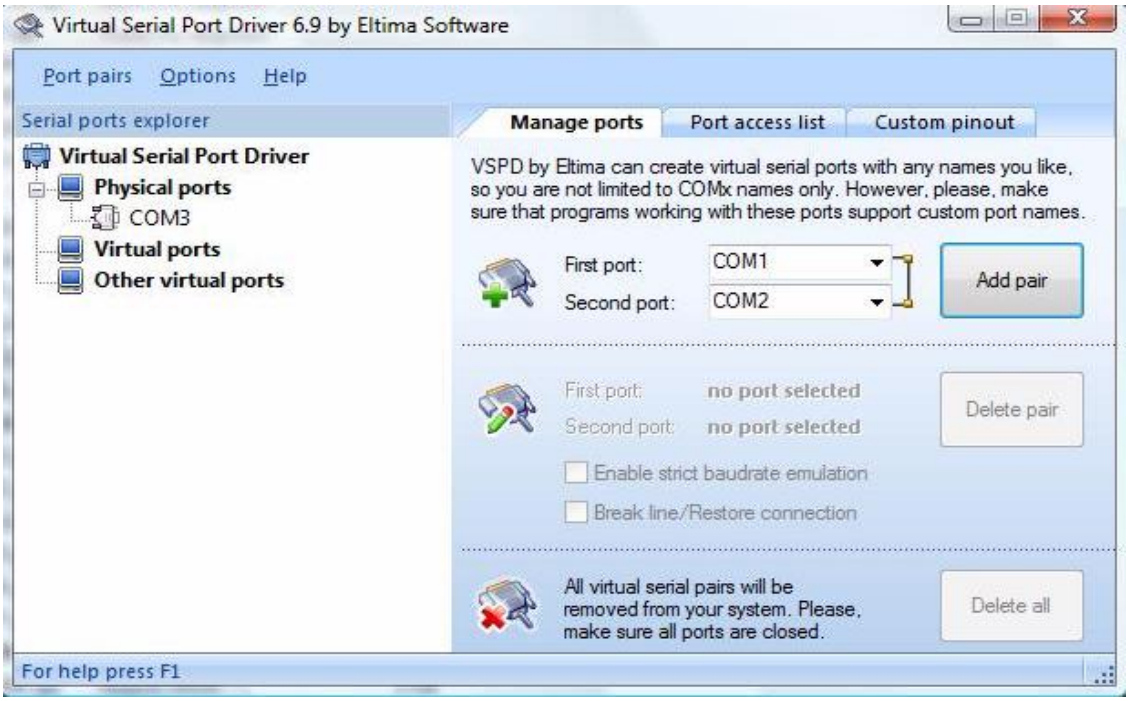
Το Virtual Serial Port Driver (VSPD), πρόκειται για ένα εμπορικό λογισμικό της εταιρίας Eltima, που δίνει τη δυνατότητα σε ένα υπολογιστή που δεν διαθέτει φυσικές σειριακές θύρες επικοινωνίας, να τις αποκτήσει, εγκαθιστώντας ένα, ή περισσότερα εικονικά ζευγάρια σειριακών θυρών στη πλατφόρμα μας και τον οδηγό για την επικοινωνία μαζί τους.

Αφού προχωρήσουμε αποδεχόμενοι τους όρους χρήσης του προγράμματος και επιλέγοντας τον κατάλογο εγκατάστασης του και ολοκληρώσουμε επιτυχώς την εγκατάσταση μας, ξεκινάμε το πρόγραμμα μας για τι πρέπει να ορίσουμε το ζευγάρι σειριακών θυρών που θέλουμε να ορίσουμε στο λειτουργικό σύστημα μας.



Σχήμα 2.6.a – Εγκατάσταση VSPD

Παρατηρούμε στα αριστερά του πάνελ *(Σχήμα 2.6.b)*, στην στήλη “serial port explorer”, πως το πρόγραμμα εντόπισε ότι υπάρχει στο σύστημα μας, μόνο η φυσική COM3 σειριακή θύρα, που όντως υπάρχει και χρησιμοποιείται από το Data Fax Modem που προϋπάρχει στο σύστημα μας και έτσι μας προτείνει να ορίσουμε ως νέο ζεύγος την COM1]--[COM2.



Σχήμα 2.6.b – Εγκατάσταση VSPD

Εάν συμφωνούμε και δεν θέλουμε να τις αλλάξουμε, κάνουμε “Add pair” *(Σχήμα 2.6.c)*. Έπειτα το πρόγραμμα θα πρέπει να μας δείχνει στην στήλη “serial port explorer”, πως υπάρχουν και virtual, εικονικές σειριακές πόρτες στο σύστημα μας.

Σχήμα 2.6.c – Εγκατάσταση VSPD

Εάν όλα πήγαν κατ'ευχήν, κλείνουμε το πρόγραμμα και πλέον δύναται να χρησιμοποιήσουμε τις δύο αυτές σειριακές θύρες επικοινωνίας για να εξυπηρετήσουμε τις ανάγκες του δικού μας συστήματος.

ΚΕΦΑΛΑΙΟ 3
Τεχνολογίες και Λογισμικό Υποστήριξης

3.1. - Σχεδίαση και προσομοίωση κυκλωμάτων στον Η/Υ

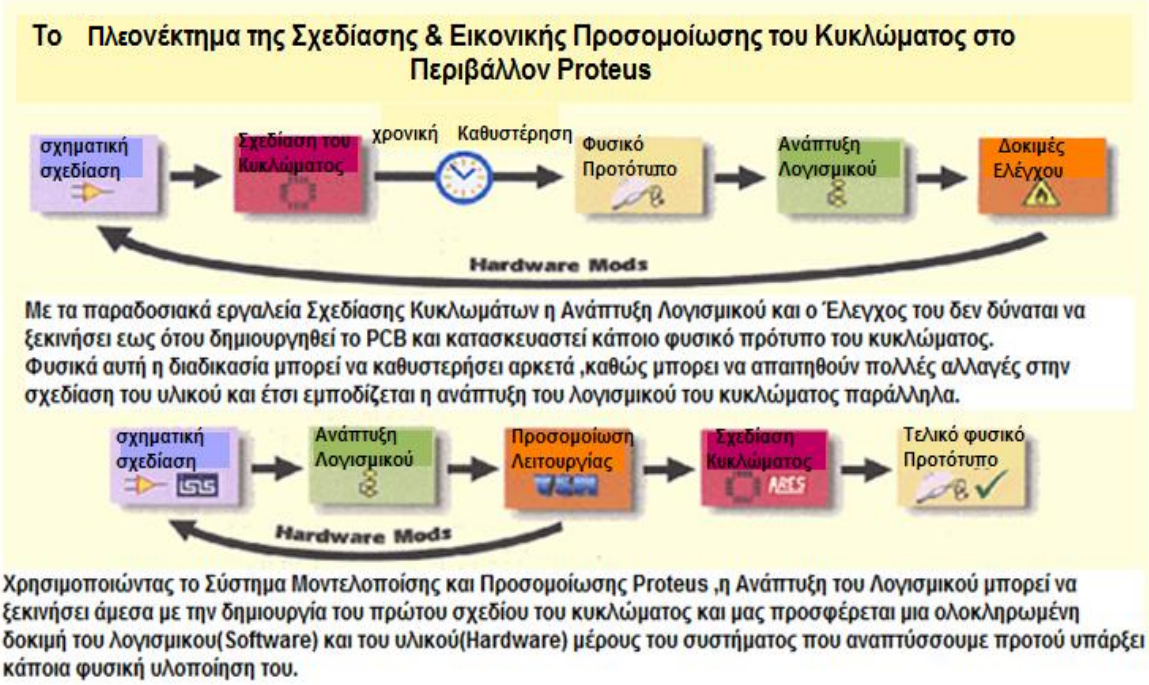
Στο παρελθόν οι ηλεκτρονικοί μηχανικοί και οι μηχανικοί υπολογιστών, έπρεπε να έχουν μια πλήρη και εκτεταμένη γνώση πάνω στις λειτουργίες των ηλεκτρονικών στοιχείων και την ανάλυση κυκλωμάτων, προτού σχεδιάσουν και αναπτύξουν ηλεκτρονικές εφαρμογές. Το κύκλωμα έπρεπε να σχεδιαστεί στο χαρτί και έπειτα να κατασκευαστεί το πρωτότυπο με βάση κάποια τεχνική προτυποποίησης υλικού, όπως η διάτρητη κάρτα (stripboard), για να ελεγχθεί η ορθότητα του κυκλώματος. Έως τώρα η σχεδίαση κυκλωμάτων ποτέ δεν ήταν αρκετή. Μόνο όταν το κύκλωμα θα ήταν πλήρως λειτουργικό, μπορούσε να κυκλοφορήσει το τελικό τυπωμένο κύκλωμα. Η εκμάθηση για το πώς λειτουργούν τα ηλεκτρονικά συστήματα θέλει μεγάλη φαντασία! Σε αντίθεση με τα μηχανικά συστήματα, δεν είναι ορατό το πώς λειτουργεί το κύκλωμα με την απλή παρατήρηση. Πρέπει να χρησιμοποιηθούν διάφορα όργανα (βολτόμετρα, παλμογράφοι, κτλ.) για να δούμε τι συμβαίνει και βέβαια απαιτεί ειδικές δεξιότητες και γνώσεις ώστε να τα χρησιμοποιήσεις αποτελεσματικά.

Πλέον διαθέτουμε εργαλεία-προγράμματα στον ηλεκτρονικό υπολογιστή που κάνουν τα πράγματα ακόμα πιο εύκολα και ενδεχομένως και πιο διασκεδαστικά. Ένα παλαιότερο λογισμικό που ονομαζόταν ECAD (Electronic Computer-Aided Design) ήταν ένα σύστημα μαθηματικής μοντελοποίησης για την «πρόβλεψη» της συμπεριφοράς ενός ηλεκτρονικού κυκλώματος. Έπειτα αναπτύχθηκε από το πανεπιστήμιο του Berkeley της Καλιφόρνιας, το πρόγραμμα SPICE για να παρέχει ένα πρακτικό και απλό στην κατανόηση σύνολο μοντέλων για τα κυκλώματα και τα σήματα. Αυτό το σύστημα πραγματοποιεί ανάλυση της

ροής του σήματος ανάμεσα σε οποιοδήποτε σημείο σε ένα κύκλωμα, βασιζόμενο στις συνδέσεις μεταξύ των συστατικών του στοιχείων.

3.2. - Το πλεονέκτημα του συστήματος μοντελοποίησης και προσομοίωσης Proteus

Η σουίτα προγραμμάτων σχεδίασης Proteus © της Labcenter Electronics © είναι μοναδική στην παροχή της δυνατότητας προσομοίωσης του κώδικα χαμηλού και υψηλού επιπέδου ενός οποιουδήποτε μικροελεγκτή σε ένα περιβάλλον απόλυτης προσομοίωσης του κυκλώματος. Χάρις σε αυτή την δυνατότητα μοντελοποίησης και εικονικής (virtual) προσομοίωσης, μπορούμε να ελέγξουμε πλήρως τον κύκλο σχεδίασης ενός προϊόντος και με χαμηλό κόστος να σχεδιάσουμε και να ελέγξουμε την σωστή λειτουργία του κώδικα του προγράμματος μας, πριν αυτό φορτωθεί για πρώτη φορά στον μικροελεγκτή. Εάν το ίδιο πρόσωπο σχεδιάζει και το υλικό και το λογισμικό, τότε πλεονεκτεί, αφού η σχεδίαση του υλικού μέρους, μπορεί να αλλάξει τόσο εύκολα, όσο και η σχεδίαση του κώδικα του λογισμικού. Σε μεγάλους οργανισμούς και εταιρίες, που οι δύο αυτοί ρόλοι είναι διαχωρισμένοι, οι προγραμματιστές, μπορούν να ξεκινήσουν την δημιουργία του λογισμικού μόλις ολοκληρωθεί η σχεδίαση του κυκλώματος χωρίς να απαιτείται να περιμένουν την φυσική υλοποίηση του πρωτοτύπου κυκλώματος. Το περιβάλλον εικονικής σχεδίασης και προσομοίωσης, Proteus, βελτιώνει την αποδοτικότητα, την ποιότητα και την προσαρμοστικότητα κατά την διαδικασία της σχεδίασης κυκλωμάτων



3.2.1. - Τι είναι το Proteus VSM

Το σύστημα εικονικής μοντελοποίησης Proteus συνδυάζει την δυνατότητα προσομοίωσης κυκλωμάτων και διαφόρων μοντέλων μικρό-επεξεργαστών για να διευκολύνει την προσομοίωση σχεδίων με βάση το μικροελεγκτή. Για πρώτη φορά, είναι εφικτή η ανάπτυξη και ο έλεγχος του κυκλώματος προτού αυτό κατασκευαστεί πραγματικά. Αυτό είναι δυνατό, γιατί μπορούμε να αλληλεπιδράμε με το σχέδιο μας μέσω τον στοιχείων που έχουμε τοποθετήσει π.χ. κουμπιά, διακόπτες κ.α. Η προσομοίωση λαμβάνει χώρα σε πραγματικό χρόνο, ή αρκετά κοντά σε αυτόν: ένας Pentium III με συχνότητα ρολογιού 1GHz μπορεί να προσομοιώσει ένα βασικό σύστημα χρονισμού 8051 στη συχνότητα των 12 MHz. Το περιβάλλον Proteus προσφέρει επίσης εκτεταμένες δυνατότητες αποσφαλμάτωσης, με σημεία ελέγχου (breakpoints), με «απλό βηματισμό» (single stepping) και προβολής του περιεχομένου των μεταβλητών, τόσο για τον κώδικα σε assembly, όσο και για κώδικα γλώσσας υψηλού επιπέδου(C).

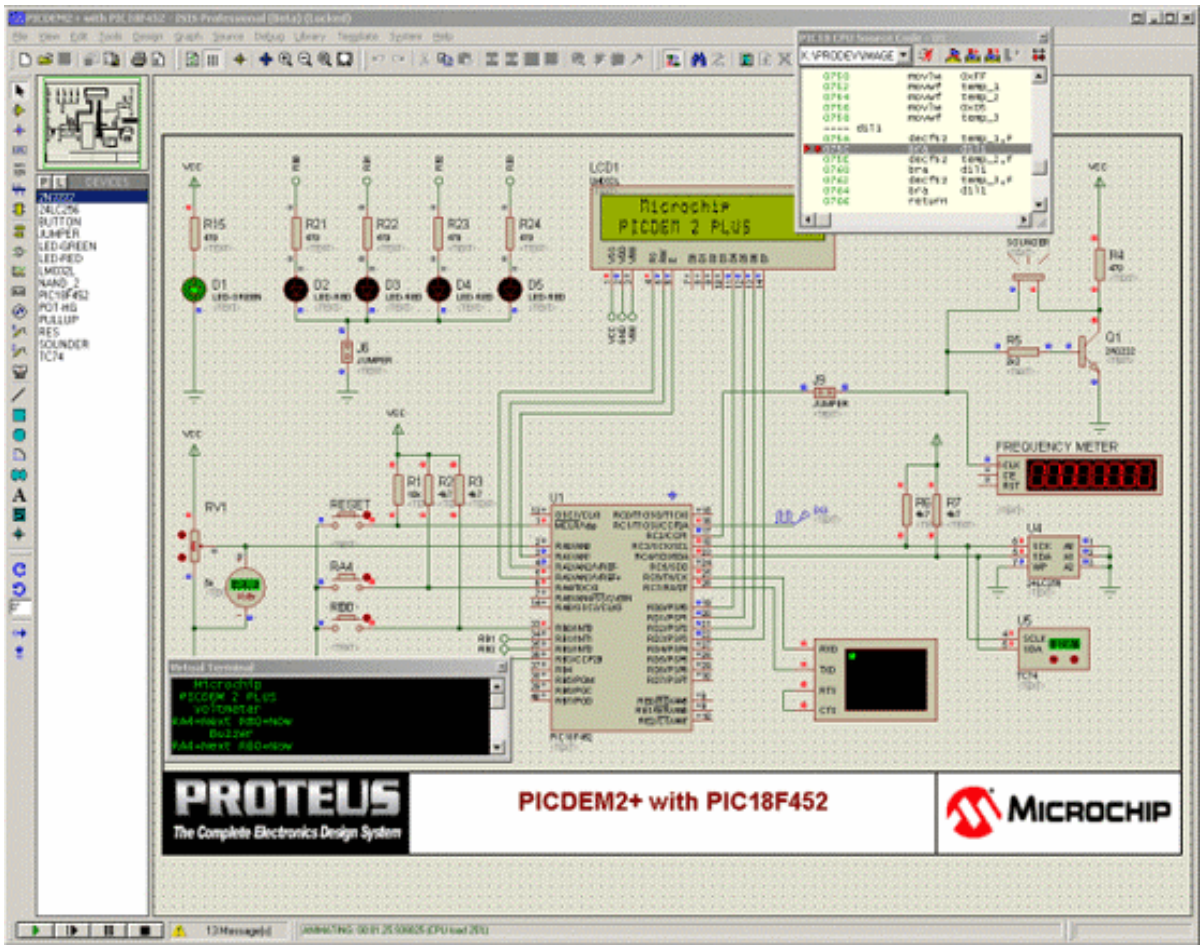
Στο *Σχήμα 3.1* βλέπουμε το σχεδιαστικό εργαλείο ISIS και παρατηρούμε μια πλήρης εικονική παρουσίαση της πλακέτας ανάπτυξης (evaluation board) PICDEM2+ της Microchip™ που περιέχει τον μικροελεγκτή PIC18F452, μια οθόνη LCD, έναν αισθητήρα θερμοκρασίας TC74, μια μνήμη EEPROM 24LC256 I2C, ένα σειριακό(RS232) τερματικό και διάφορα κουμπιά και LED's.

3.2.2. - Σχεδίαση

Το Proteus VSM χρησιμοποιεί το πρόγραμμα σχηματικής αναπαράστασης ISIS για να παράσχει το περιβάλλον σχεδίασης και ανάπτυξης κυκλωμάτων. Το πρόγραμμα αυτό είναι ικανό στο να υποστηρίζει τη σχηματική αναπαράσταση, την προσομοίωση και τη σχεδίαση του PCB. Η σχεδίαση εισάγεται στο σύστημα εικονικής μοντελοποίησης και προσομοίωσης Proteus για έλεγχο, ενώ το PCB μπορεί να δημιουργηθεί είτε από το proteus, είτε από άλλα εργαλεία σχεδίασης PCB. Το λογισμικό ISIS παρέχει επίσης έναν υψηλού επιπέδου έλεγχο πέρα από την σχεδιαστική εμφάνιση.

3.2.3. - Προσομοίωση Κυκλώματος

Η καρδιά του Συστήματος Μοντελοποίησης και Προσομοίωσης Proteus είναι το ProSPICE. Πρόκειται για ένα μαθηματικό σύστημα μοντελοποίησης κυκλωμάτων που αναπτύχθηκε πριν πολλά χρόνια και πλέον αυτό το μοντέλο μπορεί να χρησιμοποιηθεί για να δώσει ζωή στη σχεδίαση των κυκλωμάτων. Κουμπιά και εικονικές πηγές σήματος μπορούν να δημιουργούν τις εισόδους στο κύκλωμα. Οι έξοδοι του κυκλώματος μπορούν να προβληθούν σε ένα βολτόμετρο, ή σε έναν εικονικό παλμογράφο. Διαθέτοντας τη δυνατότητα προσομοίωσης των



Σχήμα 3.1 – Το σχεδιαστικό εργαλείο ISIS

μικροελεγκτών μπορούμε απλά να τοποθετήσουμε τον μικροελεγκτή που θα χρησιμοποιήσουμε στο σχέδιο του κυκλώματος μας, να του επισυνάψουμε το αρχείο με τον μεταφρασμένο κώδικα γλώσσας μηχανής και να προχωρήσουμε σε αποσφαλμάτωση του. Η ηλεκτρονική σχεδίαση κυκλωμάτων ποτέ δεν υπήρξε τόσο εύκολη υπόθεση. Το λογισμικό SPICE, συνδυάζει έναν πυρήνα αναλογικής προσομοίωσης SPICE3F5 μαζί με ένα γρήγορο, καθοδηγούμενο από τα συμβάντα ψηφιακό προσομοιωτή, για να παράσχει μια σύνθετη προσομοίωση. Η χρήση του πυρήνα SPICE μας επιτρέπει να χρησιμοποιήσουμε οποιοδήποτε από τα μοντέλα διαφόρων κατασκευαστών που σήμερα αγγίζουν τα 6000 και περιλαμβάνονται στο πακέτο λογισμικού Proteus.

Το Σύστημα Μοντελοποίησης και Προσομοίωσης Proteus περιλαμβάνει έναν μεγάλο αριθμό εικονικών (virtual) οργάνων, όπως έναν παλμογράφο (Oscilloscope), ένα αναλυτή λογικής (Logic Analyser), μια γεννήτρια συναρτήσεων (Function Generator), μια γεννήτρια προτύπων (Pattern Generator), έναν μετρητή

χρονισμού (Counter Timer) και ένα εικονικό τερματικό (Virtual Terminal), όπως και επίσης απλά βολτόμετρα και αμπερόμετρα. Προαιρετικά μας προσφέρονται αποκλειστικά πρωτόκολλα ανάλυσης Master/Slave/Monitor mode για τους τύπους σειριακής μετάδοσης SPI και I2C – πολύ απλά τα καλωδιώνουμε στις σειριακές γραμμές επικοινωνίας και παρακολουθούμε, ή αλληλεπιδρούμε με τα δεδομένα ζωντανά, κατά την διάρκεια της προσομοίωσης του κυκλώματος. Ένας αληθινά εξαιρετικά πολύτιμος και μη δαπανηρός τρόπος για τη υλοποίηση της επικοινωνίας προτού υπάρξει η φυσική υλοποίηση του πρωτοτύπου της κατασκευής. Μπορούμε να πάρουμε αναλυτικές μετρήσεις σε διάφορες μορφές διαγραμμάτων, ή ακόμα να πραγματοποιήσουμε διαφόρων ειδών αναλύσεις, όπως της συχνότητας του σήματος, της παραμόρφωσης του σήματος, του θορύβου κα.

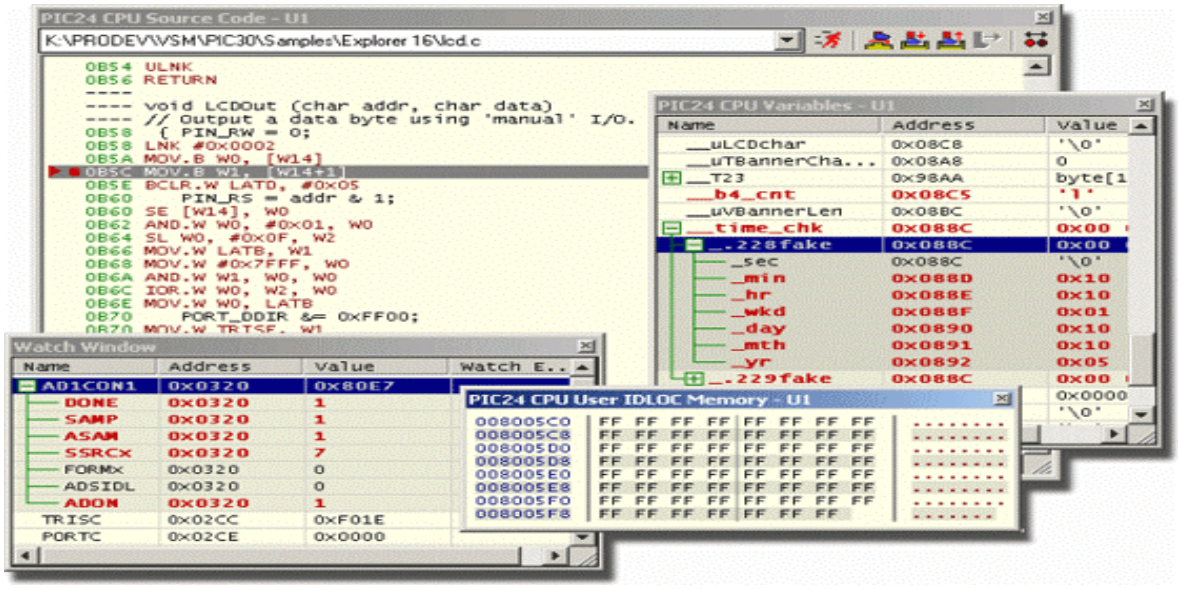
3.2.4. - Προσομοίωση λογισμικού μικροελεγκτή

Η πιο εντυπωσιακή και χρήσιμη δυνατότητα που μας παρέχει το Σύστημα Μοντελοποίησης και Προσομοίωσης Proteus, είναι η ικανότητα προσομοίωσης της διάδρασης ανάμεσα στο πρόγραμμα που τρέχει στον μικροελεγκτή και οποιοδήποτε άλλων αναλογικών, ή ψηφιακών ηλεκτρονικών συσκευών που είναι εικονικά (virtual) συνδεδεμένα σε αυτόν. Το μοντέλο του μικροελεγκτή «κάθεται» στην σχηματική αναπαράσταση μαζί με τα άλλα ηλεκτρονικά μέρη που αποτελούν την σχεδίαση του συστήματος που θέλουμε να υλοποιήσουμε. Έτσι προσομοιώνουμε την εκτέλεση του κώδικα αντικειμένου (γλώσσα μηχανής), όπως ακριβώς μέσα στο ίδιο το chip. Εάν το πρόγραμμα γράφει σε μια πόρτα, το επίπεδο λογικής αλλάζει αυτόματα και εάν το κύκλωμα αλλάζει την κατάσταση των ακροδεκτών (pins) του επεξεργαστή, από το πρόγραμμα που έχουμε γράψει, αυτό θα υλοποιηθεί όπως στην πραγματική ζωή. Η Μοντελοποίηση του επεξεργαστή, προσομοιώνει πλήρως τις πόρτες εισόδου/εξόδου του, τις διακοπές του, τον χρονισμό του (timers), τις επικοινωνίες τύπου USART και όλα τα περιφερειακά που υποστηρίζει ο κάθε επεξεργαστής. Το Proteus VSM δύναται ακόμα να προσομοιώσει σχέδια κυκλωμάτων που περιλαμβάνουν πολλαπλούς επεξεργαστές και έτσι μπορούμε πάρα πολύ απλά να τους τοποθετήσουμε στο σχέδιο μας και να τους συνδέσουμε μεταξύ τους. Φυσικά στις προσομοιώσεις μας μπορούμε να χρησιμοποιήσουμε οποιοδήποτε από τα 8000 ηλεκτρονικά συστατικά που περιλαμβάνονται στο PROSPICE

3.2.5. - Αποσφαλμάτωση πηγαίου κώδικα

Το Σύστημα Μοντελοποίησης και Προσομοίωσης Proteus, είναι ήδη μοναδικό στην ικανότητα του να προσομοιώνει πολύ κοντά στον πραγματικό χρόνο όλα τα συστήματα μικροελεγκτών, η πραγματική του δύναμη όμως έγκειται στο ότι πραγματοποιεί αυτή την προσομοίωση βήμα προς βήμα και έτσι χάρις αυτού μπορούμε να βλέπουμε γραμμή προς γραμμή την εκτέλεση του κώδικά μας και την επίδραση του, όπως και τις αλλαγές καταστάσεων και στα υπόλοιπα εξωτερικά ηλεκτρονικά μέρη του συστήματος μας που επικοινωνούν με τον μικροελεγκτή. Το πώς το VSM τα πραγματοποιεί όλα αυτά, εξαρτάται από την οικογένεια που ανήκει ο επεξεργαστής που επιλέγεται και τα υπόλοιπα εργαλεία τα οποία θα χρησιμοποιήσουμε για την δημιουργία του κώδικα του προγράμματος μας.

Στο *Σχήμα 3.2* παρατηρούμε το παράθυρο αποσφαλμάτωσης για κάποιο σχέδιο που αφορά το PIC24 Virtual Explorer16. Παρατηρούμε ότι ο πηγαίος κώδικας διαθέτει σημεία ελέγχου (breakpoints) και στις χαμηλού επιπέδου και στις υψηλού επιπέδου εντολές και πως διαθέτουμε και ένα παράθυρο παρακολούθησης των διευθύνσεων των εντολών του προγράμματος μας.



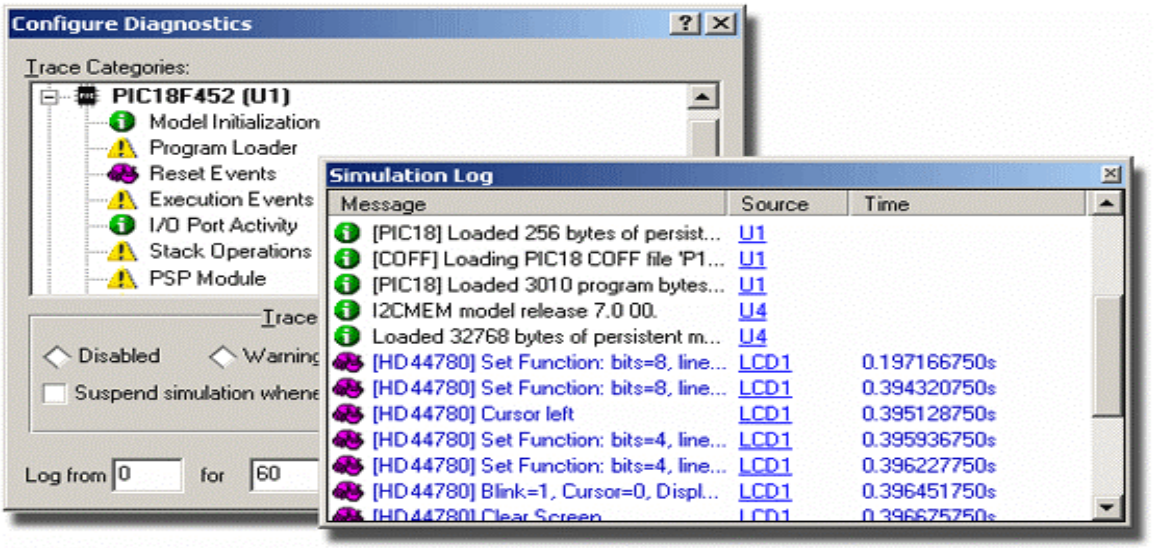
Σχήμα 3.2 – Αποσφαλμάτωση πηγαίου κώδικα

3.2.6. - Διαγνωστικά Σημεία

Το Proteus είναι εφοδιασμένο με έναν ολοκληρωμένο μηχανισμό διαγνωστικών μηνυμάτων. Αυτό μας επιτρέπει να προσδιορίσουμε σε όποια στοιχεία του κυκλώματος μας επιθυμούμε, ή σε περιφερειακά ηλεκτρονικά μέρη

που χρησιμοποιούμε, να μας ενημερώνουν σε οποιαδήποτε χρονική στιγμή με αναλυτικό γραπτό κείμενο για την δραστηριότητα του και την αντίδραση του συστήματος. Αυτό είναι μια αξιοθαύμαστη τεχνική αποσφαλμάτωσης αφού μας επιτρέπει να προσδιορίσουμε απόλυτα την πηγή του οποιουδήποτε προβλήματος παρουσιαστεί και στο λογισμικό και στο υλικό μέρος του υπό ανάπτυξη συστήματός μας, πολύ πιο γρήγορα και εύκολα από ότι θα μπορούσαμε εάν δουλεύαμε απευθείας με την φυσική διάταξη.

Στο *Σχήμα 3.3* παρατηρούμε την διαμόρφωση του διαχειριστή διαγνωστικών και προσομοίωσης, όπου προβάλλονται τα μηνύματα από το μοντέλο της LCD οθόνης αλφαριθμητικών



Σχήμα 3.3 - Διαχειριστής διαγνωστικών και προσομοίωσης.

3.2.7. - Βιβλιοθήκες Περιφερειακών Μοντέλων

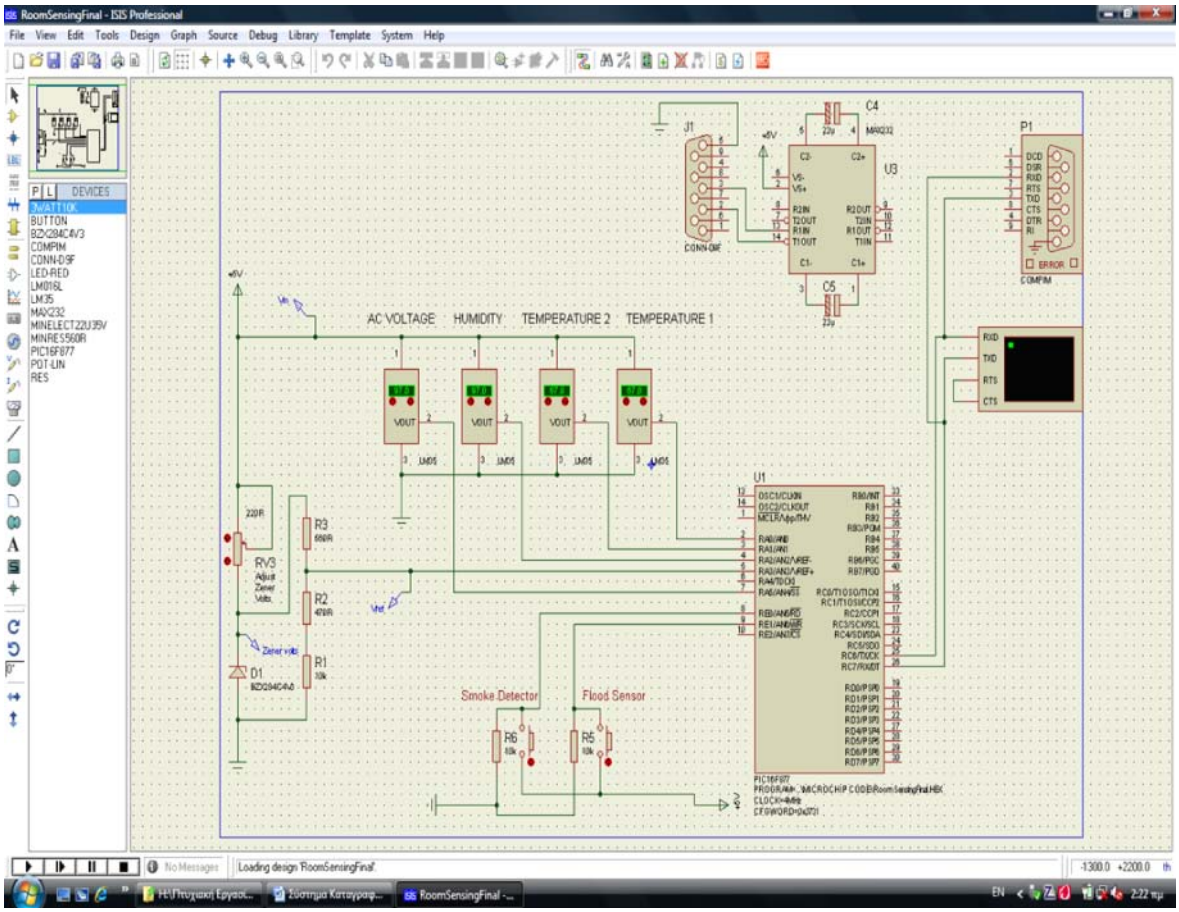
Η αληθινή δύναμη του περιβάλλοντος μοντελοποίησης και προσομοίωσης Proteus VSM ξεχωρίζει από το γεγονός ότι μπορούμε να προσομοιώσουμε το λογισμικό που δημιουργούμε, καθώς επικοινωνεί με την προσομοίωση ενός ολόκληρου υλικού συστήματος (hardware system). Αναλόγως την κατηγορία και την οικογένεια του κάθε μικροελεγκτή που χρησιμοποιείται, υπάρχουν αμέτρητα πρότυπα και μοντέλα για τα TTL/CMOS, μνήμες κτλ. Το Σύστημα Μοντελοποίησης και Προσομοίωσης Proteus είναι εμπλουτισμένο με μια τεράστια βιβλιοθήκη με ενσωματωμένα περιφερειακά μοντέλα, από οθόνες αλφαριθμητικών και γραφικών LCD, μέχρι τον ελεγκτή της διεπαφής Ethernet. Μια Αναλυτική λίστα με τα περιφερειακά που υποστηρίζει το Proteus, μπορεί να βρεθεί στη ιστοσελίδα www.labcenter.co.uk/vmodels/peripherals.cfm. Το Proteus περιέχει χιλιάδες

μοντέλα CMOS/TTL, μνήμες RAM, ROM και μνήμες I2C EEPROM, opamps, τρανζίστορς, διόδους, passives, κουμπιά, διακόπτες κτλ.

ΚΕΦΑΛΑΙΟ 4
Η Υλοποίηση

4.1. - Σχεδίαση και προσομοίωση του κυκλώματος

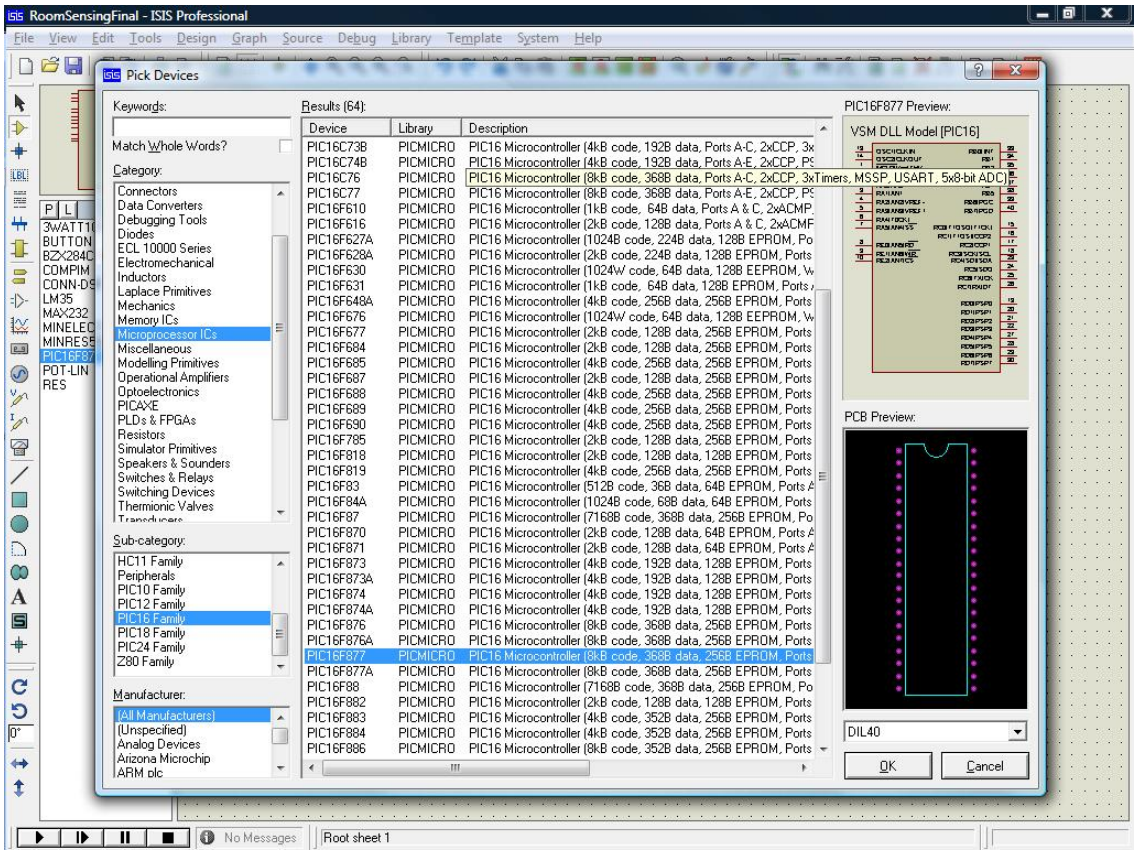
Το κύκλωμα που σχεδιάσαμε για να εξυπηρετήσει τις ανάγκες του συστήματος επιτήρησης των περιβαλλοντολογικών συνθηκών του μηχανογραφικού κέντρου παρουσιάζεται στο *Σχήμα 4.1*.



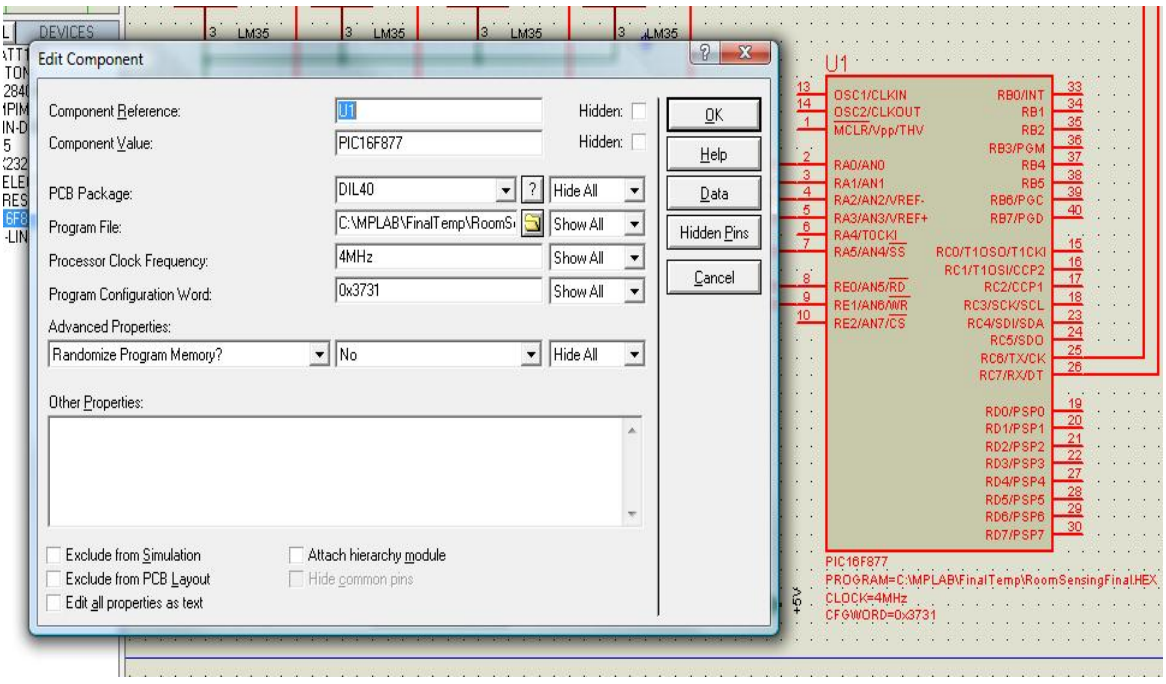
Σχήμα 4.1 – Η συσκευή επιτήρησης των περιβαλλοντολογικών συνθηκών

Στα αριστερά της εικόνας παρατηρούμε μία λίστα με όλα τα συστατικά (φυσικά και λογικά), που επιλέχθηκαν από τη βιβλιοθήκη εξαρτημάτων του Proteus *(Σχήμα 4.2)*, από τα οποία αποτελείται το κύκλωμα της συσκευής:

- **PIC16F877:** Ο μικροελεγκτής που χρησιμοποιήσαμε στο σύστημά μας. Είναι το εξάρτημα με ετικέτα “U1”. Στο *Σχήμα 4.2.a* παρατηρούμε το μικροελεγκτή και το παράθυρο ιδιοτήτων του μέσω του οποίου ορίζουμε:
  - τη συχνότητα του μικροελεγκτή
  - την τιμή του καταχωρητή διαμόρφωσης
  - το αρχείο του απόλυτου κώδικα (λογισμικό μικροελεγκτή)



Σχήμα 4.2 – Βιβλιοθήκη εξαρτημάτων του Proteus

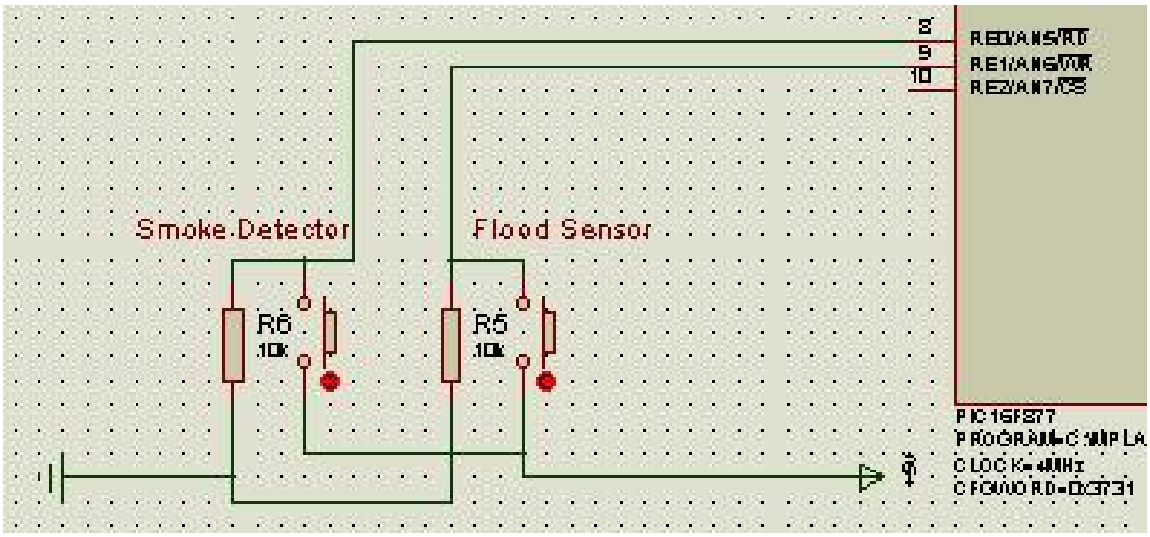


Σχήμα 4.2.α – Ο μικροελεγκτής PIC16F877

- **3WATT10K:** Αντίσταση των 10ΚΩ. Είναι τα εξαρτήματα με ετικέτες R5 και R6

- **BUTTON:** 2 κουμπιά/διακόπτες που προσομοιώνουν τη λειτουργία των αισθητήρων καπνού και πλημμύρας με ετικέτες Flood Sensor και Smoke Detector.

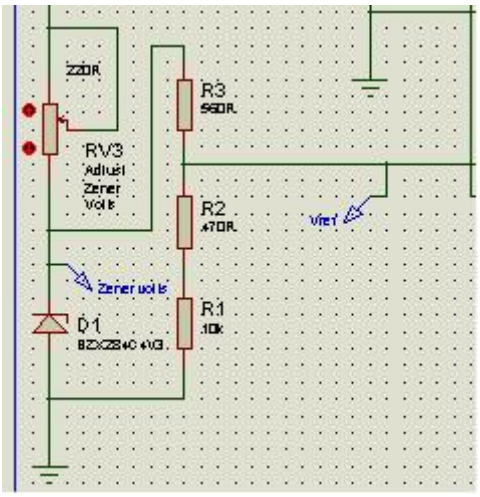
Τα τέσσερα παραπάνω εξαρτήματα παρουσιάζονται στο *Σχήμα 4.2.b*. Οι ψηφιακές είσοδοι RE0 (αισθητήρας καπνού) και RE1 (αισθητήρας πλημμύρας) είναι μονίμως γειωμένοι (λογικό 0). Τα δύο κουμπιά είναι συνδεδεμένα με την τάση τροφοδοσίας +5V (λογικό 1). Μόλις πατήσουμε το αντίστοιχο κουμπί της κάθε εισόδου αλλάζει η κατάσταση της σε λογικό 1 (true – ύπαρξη καπνού/πλημμύρας). Οι αντιστάσεις χρησιμοποιούνται για να περιορίσουν τη ροή του ρεύματος από τα +5V στη γείωση για να μη βραχυκυκλώσει η συσκευή.



Σχήμα 4.2.b – Ψηφιακοί αισθητήρες

- **BZX284C4V3:** Δίοδος Zener. Το εξάρτημα με ετικέτα D1. Χρησιμοποιείται για τη σταθεροποίηση της τάσης.
- **POT-LIN:** Ροοστάτης (μεταβλητή αντίσταση). Το εξάρτημα με ετικέτα RV3. Χρησιμοποιείται για να μεταβάλλουμε την τάση που εφαρμόζεται στη δίοδο Zener.
- **RES:** Αντιστάσεις με ετικέτες R1, R2, R3 και αντιστάσεις 10KΩ, 470Ω και 560Ω αντίστοιχα.

Τα παραπάνω 5 εξαρτήματα παρουσιάζονται στο *Σχήμα 4.2.c* και αποτελούν ένα ενιαίο τμήμα με το οποίο καθορίζουμε την τάση αναφοράς που εφαρμόζεται στον κατάλληλο ακροδέκτη του μικροελεγκτή (VREF+).



Σχήμα 4.2.c – Τμήμα καθορισμού τάσης αναφοράς

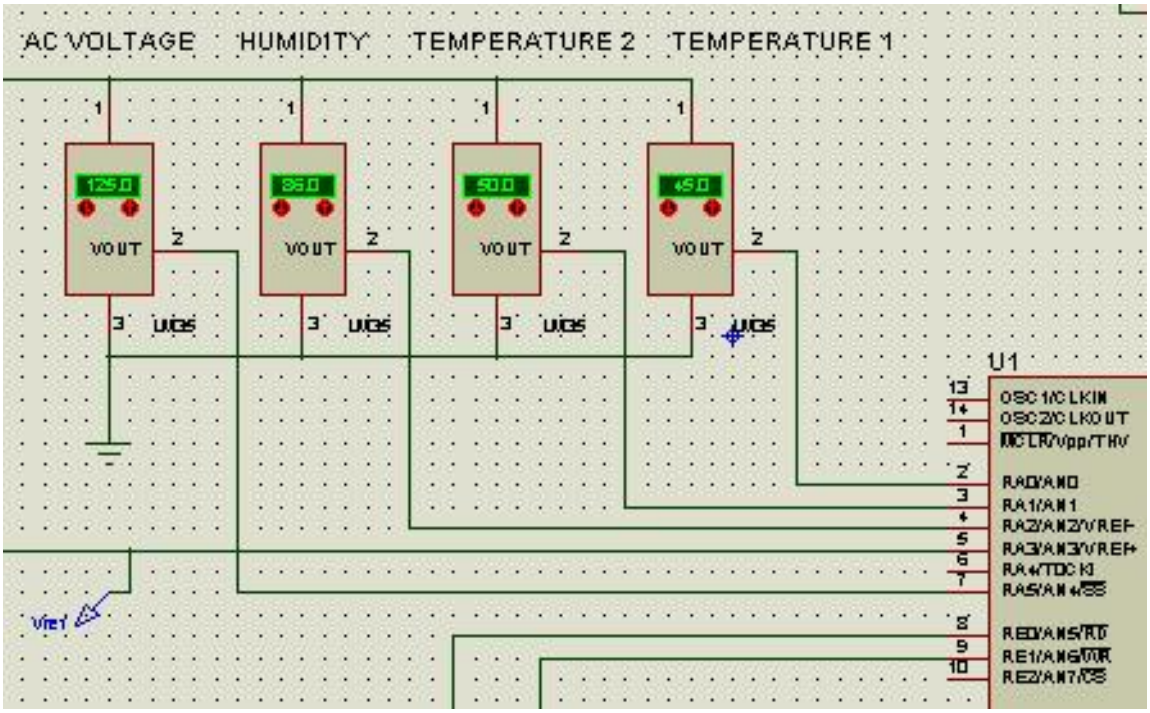
- **LM35:** Αναλογικός αισθητήρας θερμοκρασίας. Είναι τα εξαρτήματα με ετικέτες Temperature1, Temperature2, Humidity και AC Voltage.

Όπως παρατηρούμε ο LM35, πέρα από αισθητήρας θερμοκρασίας (Temperature1 και Temperature2), χρησιμοποιείται και ως αισθητήρας υγρασίας και τάσης δικτύου. Ακολουθήσαμε αυτήν την πρακτική, διότι το περιβάλλον Proteus δεν συμπεριλαμβάνει κάποιον αναλογικό αισθητήρα υγρασίας/τάσης στη βιβλιοθήκη εξαρτημάτων του. Οι αναλογικοί αισθητήρες και η διασύνδεσή τους με το μικροελεγκτή παρουσιάζονται στο *Σχήμα 4.2.d.* Όπως μπορούμε να διακρίνουμε ο LM35 μας παρέχει τη δυνατότητα να αυξομειώσουμε την τιμή εισόδου (μέτρηση θερμοκρασίας), ώστε να προσομοιώσουμε τη λειτουργία του.

Η γραμμικότητα που παρουσιάζει η έξοδος του LM35 (10 mV/°C) παρέχει ευελιξία στη χρήση του και μας επιτρέπει να το χρησιμοποιήσουμε, **στην προσομοίωση**, σαν αισθητήρα μέτρησης υγρασίας και τάσης. *Για παράδειγμα:*

Humidity: Είσοδος: 80 (στην πραγματικότητα °C) – Έξοδος: 800mV.
Εμείς θεωρούμε 80% υγρασία. ***(Έξοδος/10) % υγρασία***

AC Voltage: Είσοδος: 112 (στην πραγματικότητα °C) – Έξοδος: 1120mV.
Εμείς θεωρούμε 224V. ***(Έξοδος/10)*2 Volts***



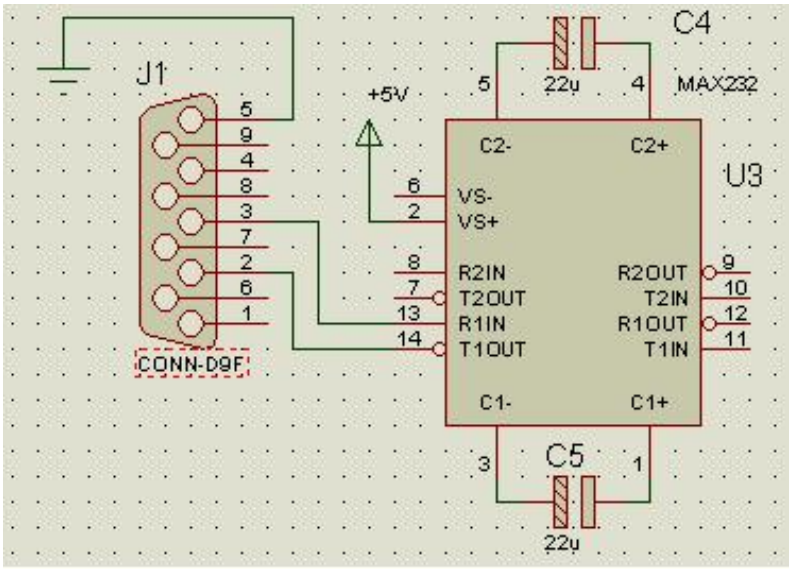
Σχήμα 4.2.d – Αναλογικοί αισθητήρες

- **MAX232:** Ολοκληρωμένο κύκλωμα για το μετασχηματισμό των σταθμών της τάσης της σειριακής διασύνδεσης (λογικό 0 και λογικό 1). Εξασφαλίζει την ορθή επικοινωνία μεταξύ μικροελεγκτή και ηλεκτρονικού υπολογιστή.
- **MINELECT22U35V:** Πυκνωτής χωρητικότητας 22μF. Είναι τα εξαρτήματα με ετικέτες C4 και C5 και χρειάζονται για την εξουδετέρωση του θορύβου στα κανάλια μετάδοσης της σειριακής διασύνδεσης.
- **CONN-D-9F:** Συνδετήρας σειριακής θύρας (connector DB9) 9 ακροδεκτών θηλυκού τύπου για τη διασύνδεση της συσκευής με τον ηλεκτρονικό υπολογιστή.

Τα παραπάνω 4 εξαρτήματα παρουσιάζονται στο *σχήμα 4.2.e*. Στην εικόνα φαίνεται πως πραγματοποιείται η διασύνδεση του μικροελεγκτή με το ολοκληρωμένο MAX232. Στον ακροδέκτη T1IN του MAX232 συνδέεται ο ακροδέκτης RC6/Tx για την αποστολή δεδομένων από το μικροελεγκτή προς τον Η/Υ. Ο MAX232 κάνει το μετασχηματισμό και μέσω του ακροδέκτη T1OUT, ο οποίος στη συνέχεια συνδέεται με τον ακροδέκτη No2 του συνδετήρα σειριακής διασύνδεσης, αποστέλλει τα δεδομένα στον Η/Υ.

Η αντίστροφη διαδικασία – για την παραλαβή δεδομένων από τον Η/Υ - γίνεται με τους ακροδέκτες No3 του συνδετήρα σειριακής διασύνδεσης, R1IN και

R1OUT του MAX232 για να καταλήξει στον ακροδέκτη του μικροελεγκτή RC7/Rx. Να σημειώσουμε ότι στο σύστημά μας δεν απαιτείται η παραλαβή δεδομένων από τον Η/Υ. Στο *Σχήμα 4.2.f.* παρουσιάζεται ο φυσικός συνδετήρας (θηλυκός DB9) για τη σειριακή διασύνδεση με τον ηλεκτρονικό υπολογιστή Παρατηρούμε την ύπαρξη μόνο δύο καλωδίων, ένα για τη γείωση (ακροδέκτης No5) και ένα για την αποστολή δεδομένων από το μικροελεγκτή προς τον Η/Υ (ακροδέκτης No2).



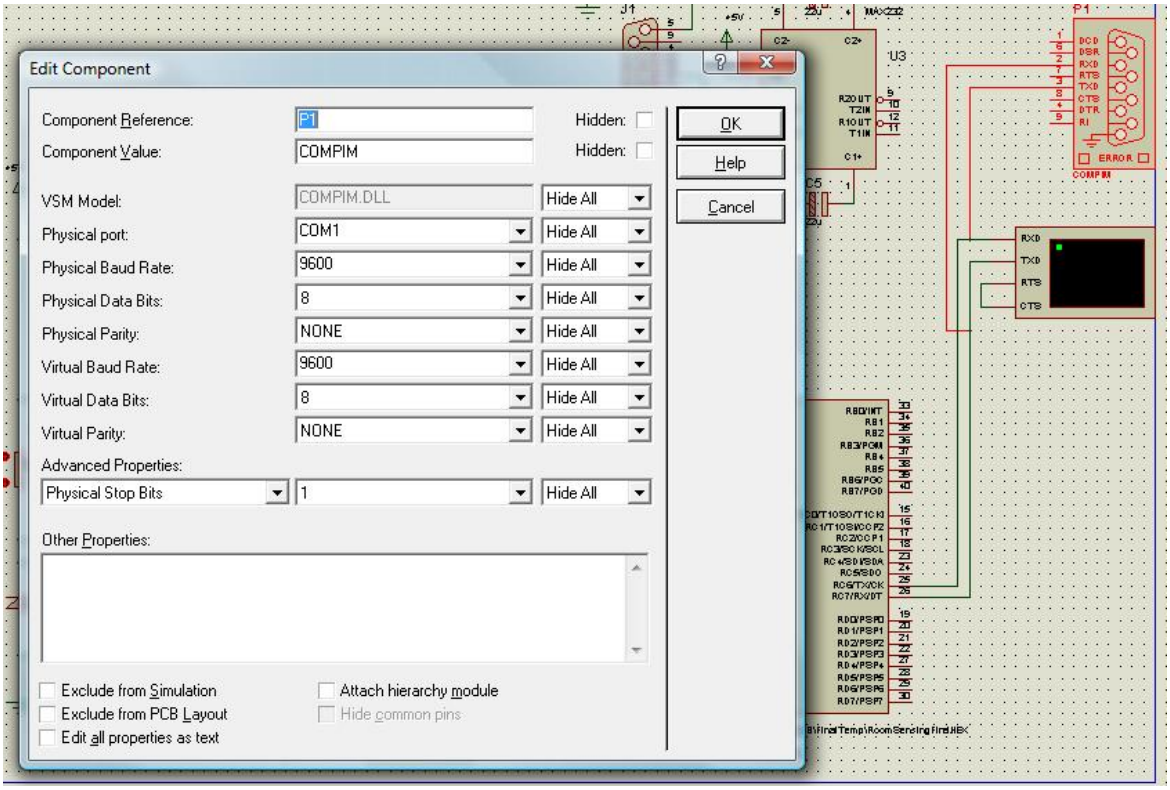
Σχήμα 4.2.e – Διάταξη φυσικής σειριακής διασύνδεσης

Σχήμα 4.2.f – Θηλυκός συνδετήρας DB9

- **COMPIM:** Εικονική σειριακή διασύνδεση. Χρησιμοποιείται (μόνο) στην προσομοίωση και αντικαθιστά τη διάταξη που περιγράφηκε παραπάνω, η οποία αφορούσε τη **φυσική** διασύνδεση του Η/Υ με το μικροελεγκτή.

Οι δυνατότητες που μας προσφέρει είναι αξιοσημείωτες. Γεφυρώνει τη προσομοίωση με το φυσικό κόσμο. Χρησιμοποιώντας μία από τις υπαρκτές ελεύθερες σειριακές θύρες του Η/Υ στον οποίο γίνεται η προσομοίωση, καταφέρνει και διασυνδέει την «εικονική συσκευή» με μία δεύτερη σειριακή θύρα. Η δεύτερη σειριακή θύρα μπορεί να ανήκει είτε στον ίδιο Η/Υ (όπου γίνεται η προσομοίωση), είτε σε κάποιον άλλον. Έτσι επιτυγχάνεται η παραλαβή των δεδομένων που αποστέλλει ο μικροελεγκτής μέσω της σειριακής διασύνδεσης.

Στο *Σχήμα 4.2.g* βλέπουμε το παράθυρο ιδιοτήτων του αντικειμένου COMPIM, στο οποίο ορίζουμε τη σειριακή θύρα που θα χρησιμοποιήσει για την διασύνδεση καθώς και τις παραμέτρους επικοινωνίας.



Σχήμα 4.2.g – Αντικείμενο COMPIM

Να αναφέρουμε επίσης ότι στην περίπτωση που δε διαθέτουμε φυσικές σειριακές θύρες, η παραλαβή δεδομένων από το μικροελεγκτή είναι εφικτή χρησιμοποιώντας ένα ζεύγος εικονικών σειριακών θυρών συνδεδεμένων μεταξύ τους. Ένα από τα λογισμικά που μας παρέχει αυτήν τη δυνατότητα είναι και το Eltima Virtual Serial Port Driver.
Εν κατακλείδι, να επισημάνουμε ότι μπορούμε να παρακολουθήσουμε τα δεδομένα που αποστέλλονται/παραλαμβάνονται από/προς τη σειριακή διασύνδεση τοποθετώντας ένα εικονικό τερματικό στους ακροδέκτες Rx, Tx (Σχήμα 2.12.g).

4.2. - Το πρόγραμμα Assembly

Εισαγωγή

Το πρόγραμμα που αναπτύξαμε για το μικροελεγκτή της συσκευής επιτήρησης, αναλαμβάνει την ανάγνωση των μετρήσεων των αναλογικών αισθητήρων, τη μετατροπή τους στην αντίστοιχη δυαδική αναπαράσταση και τέλος την κατασκευή ενός πακέτου που περιλαμβάνει τις μετρήσεις αυτές, την κατάσταση των ψηφιακών αισθητήρων καθώς και τους απαραίτητους χαρακτήρες ελέγχου για τη διασφάλιση της ορθής επικοινωνίας.

Οι κύριες λειτουργίες καθώς και οι εντολές του προγράμματος, έχουν αναλυθεί σε μεγάλο βαθμό στην αρχή της ενότητας. Ιδιαίτερη βαρύτητα δίνεται στη σειριακή μετάδοση, καθώς δε χρησιμοποιεί κάποιο προκαθορισμένο πρωτόκολλο.

Για τη μετάδοση της απαιτούμενης πληροφορίας στον Η/Υ δημιουργήσαμε ένα πακέτο καθορισμένου μεγέθους των 19 bytes *(Σχήμα 4.3)* που έχει πάντα την ίδια δομή και αποτελείται από τα εξής τμήματα:

- Μία λέξη 5 χαρακτήρων που σηματοδοτεί την αρχή του πακέτου: **START**.
- Το σύνολο των μετρήσεων όλων των αισθητήρων (αναλογικοί και ψηφιακοί).
- Μία λέξη 3 χαρακτήρων που σηματοδοτεί το τέλος του πακέτου: **END,** ακολουθούμενη από τους χαρακτήρες **"carriage return" (CR)** και την **αλλαγή γραμμής (LF).**

	Header					Analog Sensor 1		Analog Sensor 2		Analog Sensor 3		Analog Sensor 4		Digital Sensors	Footer				
	S	T	A	R	T	ADRESL	ADRESH	ADRESL	ADRESH	ADRESL	ADRESH	ADRESL	ADRESH	DIGRES	E	N	D	CR	LF
bit:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Σχήμα 4.3 – Πακέτο αποστολής δεδομένων

Η ωφέλιμη πληροφορία έχει μέγεθος στο σύνολο 9 bytes και βρίσκεται ανάμεσα στη λέξη που σηματοδοτεί την αρχή του πακέτου (START) και στη λέξη που σηματοδοτεί το τέλος του (END). Οπότε ένας τρόπος για να διασφαλίσουμε ότι διατηρήθηκε η ακεραιότητα της ωφέλιμης πληροφορίας, μετά την παραλαβή του πακέτου στον Η/Υ, είναι να ελέγξουμε:

- αν το πακέτο περιλαμβάνει έστω κάποια αλληλουχία των τελευταίων χαρακτήρων της λέξης START (π.χ A,R,T),
- αν το πακέτο περιλαμβάνει έστω κάποια αλληλουχία των πρώτων χαρακτήρων της λέξης END (π.χ. E,N)

- και αν ο αριθμός των bytes του πακέτου ανάμεσα στην αλληλουχία χαρακτήρων A,R,T και στην αλληλουχία χαρακτήρων E,N είναι ίσος με 9.

Στη συνέχεια μπορούμε με μεγαλύτερη σιγουριά να εξάγουμε την ωφέλιμη πληροφορία από το πακέτο (bytes 5-13).

- Τα bytes 5 και 6 περιλαμβάνουν τη μέτρηση του 1^{ου} αναλογικού αισθητήρα.
- Τα bytes 7 και 8 περιλαμβάνουν τη μέτρηση του 2^{ου} αναλογικού αισθητήρα.
- Τα bytes 9 και 10 περιλαμβάνουν τη μέτρηση του 3^{ου} αναλογικού αισθητήρα.
- Τα bytes 11 και 12 περιλαμβάνουν τη μέτρηση του 4^{ου} αναλογικού αισθητήρα.
- Το 13^{ο} byte - και συγκεκριμένα τα 2 λιγότερο σημαντικά bits του - περιλαμβάνει την κατάσταση των δύο ψηφιακών αισθητήρων.

Τα 2 bytes που περιλαμβάνουν τη μέτρηση κάθε αναλογικού αισθητήρα αποτελούνται από τα ADRESH και ADRESL όπου αποθηκεύεται το αποτέλεσμα της μετατροπής του ψηφιακού σήματος σε αναλογικό.

Τελική μορφή της μέτρησης

Για την τοποθέτηση του αποτελέσματος επιλέξαμε την «δεξιά τοποθέτηση» *(Σχήμα 1.16)*, οπότε στον τελικό υπολογισμό θα συμμετέχουν: ο ADRESL ως έχει και τα 2 bits που τοποθετήσαμε στον ADRESH στις 2 λιγότερο σημαντικές θέσεις (το LSB και το επόμενό του). Να σημειώσουμε ότι τα 2 αυτά bits θα συμμετέχουν στον υπολογισμό σα να βρίσκονταν στις θέσεις 8 και 9 της δυαδικής αναπαράστασης των 10 bits – αντί των θέσεων 0 και 1 του ADRESH.

Π.χ. Έστω ότι ο ADRESH έχει την τιμή xxxx xx11$_{(2)}$ (x: δε μας ενδιαφέρει η τιμή). Αυτή είναι ίση με 3$_{(10)}$. Όταν όμως τη θεωρήσουμε ως μέρος της δυαδικής αναπαράστασης 11 yyyy yyyy$_{(2)}$ (y: τα bits του ADRESL) αυτή είναι ίση με

$$1*2^9 + 1*2^8 = 768_{(10)}.$$

Παρατηρούμε ότι πολλαπλασιάζοντας κάθε φορά την τιμή των 2 λιγότερο σημαντικών bits του ADRESH (όπως είναι στις θέσεις 0 και 1) με το 256, παίρνουμε την πραγματική τιμή όταν αυτά συμμετέχουν στη δυαδική αναπαράσταση των 10 bits στις θέσεις 8 και 9.

$$3_{(10)}*256_{(10)}=768_{(10)}$$

Η τελική τιμή που αναπαριστούν τα 2 bytes (ADRESH-ADRESL) οπότε υπολογίζεται ως εξής:

(2 bits του ADRESH) * 256 + ADRESH (1).

Βέβαια αυτό μας δίνει ένα καθαρό αριθμό με εύρος τιμών 0-1023 (10 bits). Όπως έχει ήδη αναφερθεί κάθε μονάδα του αριθμού αυτού αντιστοιχεί σε 4mV της αναλογικής τάσης εξόδου του αισθητήρα, οπότε για να αναπαραστήσουμε τη τάση εξόδου του αισθητήρα σε mV πολλαπλασιάζουμε το (1) με 4.

Οπότε ο τύπος για τον υπολογισμό της αναλογικής τάσης εξόδου του αισθητήρα είναι: **[(2 bits του ADRESH) * 256 + ADRESH] * 4 (mV).**

Το κυρίως πρόγραμμα

Ξεκινάμε δηλώνοντας το μοντέλο του μικροελεγκτή για τον οποίο προορίζεται το πρόγραμμα assembly που θα ακολουθήσει, διαμορφώνουμε το μικροελεγκτή θέτοντας την κατάλληλη τιμή στον καταχωρητή διαμόρφωσης και εισάγουμε το header αρχείο για το συγκεκριμένο μοντέλο.

PROCESSOR 16F877	
__CONFIG B'11011100110001'	• Ενεργοποίηση της αποσφαλμάτωσης εντός του συστήματος • Απενεργοποίηση BOR, WDT και Low Voltage Programming • Ενεργοποίηση PWRT • Επιλογή εξωτερικού ταλαντωτή XT.
INCLUDE "P16F877A.INC"	

Συνεχίζουμε με την αντιστοίχιση των ονομάτων με τις διευθύνσεις μνήμης των καταχωρητών ειδικών λειτουργιών που θα χρησιμοποιήσουμε στο πρόγραμμά μας. Οι εντολές αυτές δεν είναι οδηγίες προς τον επεξεργαστή, αλλά οδηγίες για το μεταφραστή.

STATUS EQU 0x03

ADCON0 EQU 0x1F
ADCON1 EQU 0x9F

ADRESH EQU 0x1E
ADRESL EQU 0x9E

RCSTA EQU 0x18
TXSTA EQU 0x98
SPBRG EQU 0x99
TXREG EQU 0x19
PIR1 EQU 0x0C

TRISB EQU 0x86
TRISC EQU 0x87
TRISD EQU 0x88
TRISE EQU 0x89

PORTB EQU 0x06
PORTD EQU 0x08
PORTE EQU 0x09

Έπειτα συνεχίζουμε με την αντιστοίχιση των ονομάτων με τις διευθύνσεις μνήμης των καταχωρητών γενικού σκοπού που θα χρησιμοποιήσουμε στο πρόγραμμά μας.

COUNT EQU 0x30 ;Μετρητής.

ADRUHI1 EQU 0x31 ;Αποθήκευση ADRESH 1ου αισθητήρα (RA0)
ADRULO1 EQU 0x32 ;Αποθήκευση ADRESL 1ου αισθητήρα (RA0)

ADRUHI2 EQU 0x33 ;Αποθήκευση ADRESH 2ου αισθητήρα (RA1)
ADRULO2 EQU 0x34 ;Αποθήκευση ADRESL 2ου αισθητήρα (RA1)

ADRUHI3 EQU 0x35 ;Αποθήκευση ADRESH 3ου αισθητήρα (RA2)
ADRULO3 EQU 0x36 ;Αποθήκευση ADRESL 3ου αισθητήρα (RA2)

ADRUHI4 EQU 0x37 ;Αποθήκευση ADRESH 4ου αισθητήρα (RA3)
ADRULO4 EQU 0x38 ;Αποθήκευση ADRESL 4ου αισθητήρα (RA3)

DIGRES EQU 0x39 ; ;Αποθήκευση κατάστασης ψηφιακών εισόδων
DEL1 EQU 0x3A ;Αυτοί οι 3 καταχωρητές χρησιμοποιούνται στη
DEL2 EQU 0x3B ;ρουτίνα del1sec. για την καθυστέρηση ενός
DEL3 EQU 0x3C ;δευτερολέπτου

Κατόπιν ακολουθούν δύο εντολές που δεν προσφέρουν κάποια λειτουργικότητα στο πρόγραμμα.

```
ORG 0          ;Δείχνει τη διεύθυνση εκκίνησης του προγράμματος (0)
               ;και απαιτείται για τη διαχείριση των διακοπών
NOP            ;Εισαγωγή ενός νεκρού κύκλου. Απαιτείται από τον
               αποσφαλματωτή εντός του συστήματος.
```

4.2.1. - Διαμόρφωση μετατροπέα αναλογικού σήματος σε ψηφιακό

```
; Διαμόρφωση Μετατροπέα Αναλογικού Σήματος σε Ψηφιακό

; Βρισκόμαστε στο Bank 0 της μνήμης, οπότε διαμορφώνουμε τον καταχωρητή
  ADCON0

MOVLW B'01000001'   ;BITS:7,6=0,1: Fosc/8, Χρονισμός μετατροπέα
                    ;4MHz / 8 (Για να καλύψουμε την ελάχιστη
                    ;απαίτηση του 1,6μs για τη μετατροπή του κάθε
                    ;bit)
                    ;5,4,3=0,0,0:RA0, που σημαίνει επιλογή της
                    εισόδου RA0 προς μετατροπή
MOVWF ADCON0        ;2=0:GO=FALSE
                    ;0=1: ενεργοποίηση μετατροπέα A/D

;Πάμε στο Bank 1 για να διαμορφώσουμε τον καταχωρητή ADCON1

BCF STATUS,7        ;IRP=0 (BANK:0,1)
BCF STATUS,6        ;
BSF STATUS,5        ;RP1:RP0=0,1 (BANK:1)

MOVLW B'10000011'   ;BITS:7=1: Right Justification
                    ;6=0: Fosc/8
                    ;5,4=0,0: Δε χρησιμοποιούνται
MOVWF ADCON1        ;3,2,1,0=0,0,1,1:
                    ;AN0-2: Αναλογικές είσοδοι,
                    ;AN3: Vref+ (τάση αναφοράς),
                    ;AN4: Αναλογική είσοδος,
                    ;AN5-7: ;Ψηφιακές είσοδοι

;Τέλος Διαμόρφωσης Μετατροπέα Αναλογικού Σήματος σε Ψηφιακό
```

4.2.2. - Διαμόρφωση εισόδων/εξόδων

;Διαμόρφωση Πόρτας PORTB (TRISB)

BSF TRISE,0	;θέτουμε τους ακροδέκτες της PORTB
BSF TRISE,1	;(0,1,2) ως ψηφιακές εισόδους.
BSF TRISE,2	;

; Τέλος Διαμόρφωσης Πόρτας PORTB

4.2.3. - Διαμόρφωση παραμέτρων σειριακής πόρτας

;Διαμόρφωση Σειριακής Επικοινωνίας

;USART: Ασύγχρονη Σειριακή Μετάδοση Δεδομένων

;9600 Baud rate, 8 Data bit, 1 start bit, 1 stop bit, NO hardware control

;Είμαστε στο Bank 1, οπότε και διαμορφώνουμε τον καταχωρητή TXSTA

BCF TXSTA,6	;8-bit Μετάδοση
BCF TXSTA,5	;Απενεργοποίηση μετάδοσης (αρχικά)
BCF TXSTA,4	;Επιλογή ασύγχρονης μετάδοσης
BSF TXSTA,2	;Επιλογή υψηλού ρυθμού μετάδοσης

;Διαμορφώνουμε τον καταχωρητή SPBRG στο Bank1 για να ρυθμίσουμε το ρυθμό μετάδοσης

MOVLW D'25'	;Από τον Πίνακα 1.4, με χρονισμό μικροελεγκτή ;4MHz και επιλογή υψηλού ρυθμού μετάδοσης ;(BRGH=1) για ρυθμό μετάδοσης 9600bps
MOVWF SPBRG	;Κατάλληλη τιμή για τον SPBRG είναι η '25'
BSF TXSTA,5	;αρχικοποίηση της μετάδοσης

```
;Τώρα πηγαίνουμε στο Bank 0 για να διαμορφώσουμε τον καταχωρητή RCSTA

                        ;IRP=0 (BANK:0,1)
BCF STATUS,5            ;RP1:RP0=00 (BANK:0)

BSF RCSTA,7             ;Ενεργοποίηση της σειριακής πόρτας.
                        ;Οι ακροδέκτες RC7,RC6 χρησιμοποιούνται για
                        ;παραλαβή/αποστολή (Rx,Tx)

;Διαμορφώσαμε την σειριακή μας πόρτα, μόνο για μετάδοση δεδομένων
;δεν χρειάζεται να διαμορφωθούν άλλα bits στον καταχωρητή RCSTA.
```

4.2.4. - Ανάγνωση αναλογικών τάσεων από τους αισθητήρες

```
;Είμαστε στο Bank 0 και διαμορφώνουμε τον καταχωρητή ADCON0 για να
;επιλέξουμε την είσοδο προς μετατροπή.

start   MOVLW B'01000001'   ;5,4,3=0,0,0: RA0, που σημαίνει επιλογή της
                            ;εισόδου RA0 προς μετατροπή
        MOVWF ADCON0        ;
        CALL getIn          ;Καλούμε την υπορουτίνα για τη μετατροπή

;Πηγαίνουμε στο Bank 1 για να πάρουμε το ADRESL

        BSF STATUS,5        ;RP1:RP0=01 (BANK:1)

                            ;Παίρνουμε το ADRESL του αποτελέσματος
                            ;της μετατροπής από τον 1ο αναλογικό
        MOVF ADRESL,0       ;αισθητήρα (RA0) και το μετακινούμε στο W

;Επιστρέφουμε στο Bank 0 για να αποθηκεύσουμε τους ADRESH και ADRESL

BCF STATUS,5            ;RP1:RP0=0,0 (BANK:0)

MOVWF ADRULO1           ;Αποθήκευση του ADRESL του 1ου αισθητήρα στον
                        ;καταχωρητή ADRULO1
MOVF ADRESH,0           ;Μετακινούμε τον ADRESH του 1ου αισθητήρα στο W
MOVWF ADRUHI1           ;Αποθήκευση του ADRESH του 1ου αισθητήρα στον
                        ;καταχωρητή ADRUHI1
;Επιλογή της επόμενης αναλογικής εισόδου προς μετατροπή
BSF ADCON0,3            ;5,4,3=0,0,1, που σημαίνει επιλογή της
```

```
                        ;εισόδου RA1 προς μετατροπή
CALL getIn              ;Καλούμε την υπορουτίνα για τη μετατροπή

;Πηγαίνουμε στο Bank 1 για να πάρουμε το ADRESL  και να το αποθηκεύσουμε

BSF STATUS,5            ;RP1:RP0=0,1 (BANK:1)
                        ;Παίρνουμε το ADRESL του αποτελέσματος
                        ;της μετατροπής από τον 2ο αναλογικό
MOVF ADRESL,0           ;αισθητήρα (RA1) και το μετακινούμε στο W

;Επιστρέφουμε στο Bank 0 για να αποθηκεύσουμε τους ADRESH και ADRESL

BCF STATUS,5            ;RP1:RP0=0,0 (BANK:0)

MOVWF ADRULO2           ;Αποθήκευση του ADRESL του 2ου αισθητήρα στον
                        ;καταχωρητή ADRULO2
MOVF ADRESH,0           ;Μετακινούμε τον ADRESH του 2ου αισθητήρα στο W
MOVWF ADRUHI2           ;Αποθήκευση του ADRESH του 2ου αισθητήρα στον
                        ;καταχωρητή ADRUHI2

;Επιλογή της επόμενης αναλογικής εισόδου προς μετατροπή

BCF ADCON0,3            ;5,4,3=0,1,0: που σημαίνει επιλογή της
BSF ADCON0,4            ;εισόδου RA2 προς μετατροπή

CALL getIn              ;Καλούμε την υπορουτίνα για τη μετατροπή

;Πηγαίνουμε στο Bank 1 για να πάρουμε το ADRESL  και να το αποθηκεύσουμε

BSF STATUS,5            ;RP1:RP0=0,1 (BANK:1)
                        ;Παίρνουμε το ADRESL του αποτελέσματος
                        ;της μετατροπής από τον 3ο αναλογικό
MOVF ADRESL,0           ;αισθητήρα (RA2) και το μετακινούμε στο W

;Επιστρέφουμε στο Bank 0 για να αποθηκεύσουμε τους ADRESH και ADRESL

BCF STATUS,5            ;RP1:RP0=0,0 (BANK:0)

MOVWF ADRULO3           ;Αποθήκευση του ADRESL του 3ου αισθητήρα στον
                        ;καταχωρητή ADRULO3
MOVF ADRESH,0           ;Μετακινούμε τον ADRESH του 3ου αισθητήρα στο W
MOVWF ADRUHI3           ;Αποθήκευση του ADRESH του 3ου αισθητήρα στον
                        ;καταχωρητή ADRUHI3
```

```
BSF ADCON0,5        ;5,4,3=1,0,0: που σημαίνει επιλογή της
BCF ADCON0,4        ; ;εισόδου RA3 προς μετατροπή

CALL getIn          ;Καλούμε την υποροutίνα για τη μετατροπή

;Πηγαίνουμε στο Bank 1 για να πάρουμε το ADRESL  και να το αποθηκεύσουμε

BSF STATUS,5        ;RP1:RP0=0,1 (BANK:1)
                    ;Παίρνουμε το ADRESL του αποτελέσματος
                    ;της μετατροπής από τον 3ο αναλογικό
MOVF ADRESL,0       ;αισθητήρα (RA2) και το μετακινούμε στο W

;Επιστρέφουμε στο Bank 0 για να αποθηκεύσουμε τους ADRESH και ADRESL

BCF STATUS,5        ;RP1:RP0=0,0 (BANK:0)

MOVWF ADRULO4       ;Αποθήκευση του ADRESL του 4ου αισθητήρα στον
                    ;καταχωρητή ADRULO4
MOVF ADRESH,0       ;Μετακινούμε τον ADRESH του 4ου αισθητήρα στο W
MOVWF ADRUHI4       ;Αποθήκευση του ADRESH του 4ου αισθητήρα στον
                    ;καταχωρητή ADRUHI4

;Αποθηκεύουμε τις τιμές των ψηφιακών αισθητήρων στον καταχωρητή DIGRES

MOVF PORTE,0        ;Μετακινούμε το PORTB στο W
MOVWF DIGRES        ;Μετακινούμε το W στο DIGRES
```

4.2.5. - Ρουτίνα μετατροπής αναλογικού σήματος σε ψηφιακό

```
;Ρουτίνα μετασχηματισμού αναλογικού σήματος σε ψηφιακό
getIn   MOVLW D'7'      ;Καθυστέρηση 20μs για να επιτρέψουμε τη
                        ;σταθεροποίηση της τάσης από τις αναλογικές
        MOVWF COUNT     ;εισόδους μεταξύ των μετατροπών
loop    DECFSZ COUNT    ;
        GOTO loop       ;
        BSF ADCON0,2    ; BIT:3=1: Go/Done=True που σημαίνει έναρξη της
                        ;διαδικασίας μετατροπής
wait    BTFSC ADCON0,2  ;Περιμένουμε μέχρι το bit Go/Done να γίνει 0
        GOTO wait       ;που σηματοδοτεί και το πέρας της μετατροπής
        RETURN          ;επιστρέφουμε από την υποροutίνα στο κυρίως
```

```
                                    ;πρόγραμμα.
;Τέλος ρουτίνας μετασχηματισμού αναλογικού σήματος σε ψηφιακό
;Το αποτέλεσμα της μετατροπής είναι μεγέθους 10-Bit και βρίσκεται στο ζεύγος
;καταχωρητών ADRESH-ADRESL.
```

4.2.6. - Ρουτίνα σειριακής μετάδοσης δεδομένων

```
;Ρουτίνα σειριακής μετάδοσης των αποτελεσμάτων

;Βρισκόμαστε στο Bank 0

;Μετάδοση πρώτου χαρακτήρα της κεφαλίδας του πακέτου

serSend     MOVLW 0x53      ;Μετακίνηση του "S" στο W
            MOVWF TXREG     ;Μετάδοση  S (από το START)
wtH1        BTFSS PIR1,4    ;Έλεγχος εάν ο ενταμιευτής της
                            ;μετάδοσης έχει αδειάσει-TXREG
            GOTO wtH1       ;TRANSMIT INTERRUPT FLAG = 1.

;Μετάδοση επόμενου χαρακτήρα

            MOVLW 0x54      ;Μετακίνηση του "T" στο W
            MOVWF TXREG     ;Μετάδοση του T
wtH2        BTFSS PIR1,4    ;Έλεγχος εάν ο ενταμιευτής της
                            ;μετάδοσης έχει αδειάσει-TXREG
            GOTO wtH2       ;TRANSMIT INTERRUPT FLAG = 1.

;Μετάδοση επόμενου χαρακτήρα

            MOVLW 0x41      ;Μετακίνηση του "A" στο W
            MOVWF TXREG     ;Μετάδοση του A
wtH3        BTFSS PIR1,4    ;Έλεγχος εάν ο ενταμιευτής της
                            ;μετάδοσης έχει αδειάσει-TXREG
            GOTO wtH3       ;TRANSMIT INTERRUPT FLAG = 1.

;Μετάδοση επόμενου χαρακτήρα

            MOVLW 0x52      ;Μετακίνηση του "R" στο W
            MOVWF TXREG     ;Μετάδοση του R
wtH4        BTFSS PIR1,4    ;Έλεγχος εάν ο ενταμιευτής της
                            ;μετάδοσης έχει αδειάσει-TXREG
```

```
                GOTO wtH4       ;TRANSMIT INTERRUPT FLAG = 1.

;Μετάδοση επόμενου χαρακτήρα

                MOVLW 0x54      ;Μετακίνηση του "T" στο W
                MOVWF TXREG     ;Μετάδοση του T
wtH5            BTFSS PIR1,4    ;Έλεγχος εάν ο ενταμιευτής της
                                ;μετάδοσης έχει αδειάσει-TXREG
                GOTO wtH5       ;TRANSMIT INTERRUPT FLAG = 1.

;Μετάδοση αποτελέσματος μετατροπής 1ου αναλογικού αισθητήρα

                MOVF ADRULO1,0  ;Μετακίνηση του SENSOR1
                                ;LO-Αποτελέσματος στο W
                MOVWF TXREG     ;Μετάδοση ADDRULO1
wt1             BTFSS PIR1,4    ;Έλεγχος εάν ο ενταμιευτής της
                                ;μετάδοσης έχει αδειάσει-TXREG
                GOTO wt1        ;TRANSMIT INTERRUPT FLAG = 1.

                MOVF ADRUHI1,0  ;Μετακίνηση του SENSOR1
                                ;HI-Αποτελέσματος στο W
                MOVWF TXREG     ;Μετάδοση ADDRUHI1
wt2             BTFSS PIR1,4    ;Έλεγχος εάν ο ενταμιευτής της
                                ;μετάδοσης έχει αδειάσει-TXREG
                GOTO wt2        ;TRANSMIT INTERRUPT FLAG = 1.

;Μετάδοση αποτελέσματος μετατροπής 2ου αναλογικού αισθητήρα

                MOVF ADRULO2,0  ; Μετακίνηση του SENSOR2
                                ;LO-Αποτελέσματος στο W
                MOVWF TXREG     ;Μετάδοση ADDRULO2
wt3             BTFSS PIR1,4    ;Έλεγχος εάν ο ενταμιευτής της
                                ;μετάδοσης έχει αδειάσει-TXREG
                GOTO wt3        ;TRANSMIT INTERRUPT FLAG = 1.

                MOVF ADRUHI2,0  ;Μετακίνηση του SENSOR2
                                ;HI-Αποτελέσματος στο W
                MOVWF TXREG     ;Μετάδοση ADDRUHI2
wt4             BTFSS PIR1,4    ;Έλεγχος εάν ο ενταμιευτής της
```

```
                                ;μετάδοσης έχει αδειάσει-TXREG
                GOTO wt4        ;TRANSMIT INTERRUPT FLAG = 1.
```

;Μετάδοση αποτελέσματος μετατροπής 3ου αναλογικού αισθητήρα

```
                MOVF ADRULO3,0  ;Μετακίνηση του SENSOR3
                                ;LO-Αποτελέσματος στο W
                MOVWF TXREG     ;Μετάδοση ADDRULO3
wt5             BTFSS PIR1,4    ;Έλεγχος εάν ο ενταμιευτής της
                                ;μετάδοσης έχει αδειάσει-TXREG
                GOTO wt5        ;TRANSMIT INTERRUPT FLAG = 1.

                MOVF ADRUHI3,0  ;Μετακίνηση του SENSOR3
                                ;HI-Αποτελέσματος στο W
                MOVWF TXREG     ;Μετάδοση του ADDRUHI3
wt6             BTFSS PIR1,4    ;Έλεγχος εάν ο ενταμιευτής της
                                ;μετάδοσης έχει αδειάσει-TXREG
                GOTO wt6        ;TRANSMIT INTERRUPT FLAG = 1.
```

;Μετάδοση αποτελέσματος μετατροπής 4ου αναλογικού αισθητήρα

```
                MOVF ADRULO4,0  ;Μετακίνηση του SENSOR4
                                ;LO-Αποτελέσματος στο W
                MOVWF TXREG     ;Μετάδοση του ADDRULO4
wt7             BTFSS PIR1,4    ;Έλεγχος εάν ο ενταμιευτής της
                                ;μετάδοσης έχει αδειάσει-TXREG
                GOTO wt7        ;TRANSMIT INTERRUPT FLAG = 1.

                MOVF ADRUHI4,0  ;Μετακίνηση του SENSOR4
                                ;HI-Αποτελέσματος στο W
                MOVWF TXREG     ;Μετάδοση του ADDRUHI4
wt8             BTFSS PIR1,4    ;Έλεγχος εάν ο ενταμιευτής της
                                ;έχει αδειάσει-TXREG
                GOTO wt8        ;TRANSMIT INTERRUPT FLAG = 1.
```

;Μετάδοση ψηφιακών εισόδων

```
                MOVF DIGRES,0   ;Μετακίνηση DIGITAL SENSORS
                                ;Αποτελέσματος στο W
                MOVWF TXREG     ;Μετάδοση του DIGRES
```

```
wt9         BTFSS PIR1,4      ;Έλεγχος εάν ο ενταμιευτής της
                              ;μετάδοσης έχει αδειάσει-TXREG
            GOTO wt9          ;TRANSMIT INTERRUPT FLAG = 1.

;Μετάδοση πρώτου χαρακτήρα της λέξης που σηματοδοτεί το τέλος του πακέτου

            MOVLW 0x45        ;Μετακίνηση του "E" στο W
            MOVWF TXREG       ;Μετάδοση E (END δηλαδή)
wtF1        BTFSS PIR1,4      ;Έλεγχος εάν ο ενταμιευτής της
                              ;μετάδοσης έχει αδειάσει-TXREG
            GOTO wtF1         ;TRANSMIT INTERRUPT FLAG = 1.

;Μετάδοση επόμενου χαρακτήρα

            MOVLW 0x4E        ;Μετακίνηση του "N" στο W
            MOVWF TXREG       ;Μετάδοση του N
wtF2        BTFSS PIR1,4      ;Έλεγχος εάν ο ενταμιευτής της
                              ;μετάδοσης έχει αδειάσει -TXREG
            GOTO wtF2         ;TRANSMIT INTERRUPT FLAG = 1.

;Μετάδοση επόμενου χαρακτήρα

            MOVLW 0x44        ;Μετακίνηση του "D" στο W
            MOVWF TXREG       ;Μετάδοση του D
wtF3        BTFSS PIR1,4      ;Έλεγχος εάν ο ενταμιευτής της
                              ;μετάδοσης έχει αδειάσει-TXREG
            GOTO wtF3         ;TRANSMIT INTERRUPT FLAG = 1.

;Μετάδοση επόμενου χαρακτήρα

            MOVLW 0x0D        ;Μετακίνηση χαρακτήρα
                              ; "CARRIAGE RETURN" στο W
            MOVWF TXREG       ;Μετάδοση του carriage_return
wt10        BTFSS PIR1,4      ;Έλεγχος εάν ο ενταμιευτής της
                              ;μετάδοσης έχει αδειάσει-TXREG
            GOTO wt10         ;TRANSMIT INTERRUPT FLAG = 1.

;Μετάδοση επόμενου χαρακτήρα

            MOVLW 0x0A        ;Μετακίνηση του χαρακτήρα
                              ;"NEW LINE FEED" στο W
```

```
                MOVWF TXREG     ;Μετάδοση του new_line
wt11            BTFSS PIR1,4    ;Έλεγχος εάν ο ενταμιευτής της
                                ;μετάδοσης έχει αδειάσει-TXREG
                GOTO wt11       ;TRANSMIT INTERRUPT FLAG = 1.

                RETURN

;ΤΕΛΟΣ ΡΟΥΤΙΝΑΣ ΣΕΙΡΙΑΚΗΣ ΜΕΤΑΔΟΣΗΣ
```

4.2.7. - Ρουτίνα καθυστέρησης ενός δευτερόλεπτου

```
;ΡΟΥΤΙΝΑ ΚΑΘΥΣΤΕΡΗΣΗΣ ΕΝΟΣ ΔΕΥΤΕΡΟΛΕΠΤΟΥ
;999997 ΚΥΚΛΟΙ ΜΗΧΑΝΉΣ

;Αυτή η ρουτίνα δημιουργήθηκε αυτόματα από την ιστοσελίδα
;http://www.piclist.com/techref/piclist/codegen/delay.htm

del1sec         movlw   0x08
                movwf   DEL1
                movlw   0x2F
                movwf   DEL2
                movlw   0x03
                movwf   DEL3

del1sec_0       decfsz  DEL1, f
                goto    $+2
                decfsz  DEL2, f
                goto    $+2
                decfsz  DEL3, f
                goto    del1sec_0

;3 ΚΥΚΛΟΙ ΜΗΧΑΝΗΣ
                goto    $+1
                nop

                RETURN

; ΤΕΛΟΣ ΡΟΥΤΙΝΑΣ ΚΑΘΥΣΤΕΡΗΣΗΣ ΕΝΟΣ ΔΕΥΤΕΡΟΛΕΠΤΟΥ

;Αυτή η ρουτίνα δημιουργήθηκε αυτόματα από την ιστοσελίδα
;http://www.piclist.com/techref/piclist/codegen/delay.htm
```

Δεύτερη Ενότητα

ΤΟ ΣΥΣΤΗΜΑ ΔΙΑΧΕΙΡΙΣΗΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

ΚΕΦΑΛΑΙΟ 5
Ο Διακομιστής Βάσεων Δεδομένων MySQL 6

5.1. - Εισαγωγή

Η MySQL είναι το πιο δημοφιλές, ανοιχτού κώδικα λογισμικό διαχείρισης σχεσιακών βάσεων δεδομένων.

- **Η MySQL είναι ένα Σύστημα Διαχείρισης Βάσεων Δεδομένων.**

Μια βάση Δεδομένων, είναι μια δομημένη συλλογή δεδομένων. Μπορεί να είναι οτιδήποτε, από μια λίστα αγορών, μία συλλογή εικόνων, μέχρι τεράστιες ποσότητες δεδομένων μεγάλων εταιρικών δικτύων. Για να εισάγουμε, να προσπελάσουμε και να επεξεργαστούμε τα δεδομένα που είναι αποθηκευμένα σε ένα σύστημα βάσεων δεδομένων, χρειαζόμαστε ένα σύστημα διαχείρισης όπως ο διακομιστής MySQL. Από τότε που οι υπολογιστές χειρίζονται μεγάλες ποσότητες δεδομένων, τα συστήματα διαχείρισης βάσεων δεδομένων παίζουν έναν κεντρικό ρόλο στην «ζωή» τους, είτε ως αυτόνομα εργαλεία, είτε ως τμήματα άλλων μεγάλων εφαρμογών.

- **Η MySQL είναι ένα Σχεσιακό Σύστημα Διαχείρισης Βάσεων Δεδομένων.**

Μια σχεσιακή βάση δεδομένων, αποθηκεύει τα δεδομένα σε χωριστούς πίνακες και όχι σε ένα ενιαίο χώρο δεδομένων. Αυτό προσδίδει ευελιξία και ταχύτητα. Το SQL τμήμα της “MySQL” αναφέρεται στην γνωστή “Structured Query Language”. Η SQL είναι η βασική τυποποιημένη γλώσσα που χρησιμοποιείται για την προσπέλαση Βάσεων Δεδομένων και ορίζεται από το πρότυπο ANSI/ISO SQL. Το πρότυπο SQL επινοήθηκε το 1986 και υπάρχουν διάφορες εκδόσεις. Η σημερινή έκδοση που χρησιμοποιείται, είναι η “SQL:2003”.

- **Η MySQL είναι Λογισμικό Ανοιχτού Κώδικα.**

Όταν το λογισμικό χαρακτηρίζεται ως «ανοιχτού κώδικα», σημαίνει ότι δίνεται ελεύθερα στον κάθε ένα, για χρήση και τροποποίηση του. Ο καθένας μπορεί να κατεβάσει το λογισμικό και να το χρησιμοποιήσει χωρίς να πληρώσει κάποιο αντίτιμο. Εάν το επιθυμούμε, μπορούμε να μελετήσουμε τον πηγαίο κώδικα και να τον αλλάξουμε για να ανταποκρίνεται στις δικές μας απαιτήσεις. Η MySQL χρησιμοποιεί την άδεια χρήσης GPL (GNU General Public License), http://www.fsf.org/licenses/, για να προσδιορίσει τι μπορούμε και τι δεν μπορούμε να κάνουμε με το λογισμικό σε διάφορες συνθήκες. Εάν δεν μας ικανοποιεί η άδεια GPL, ή εάν θελήσουμε να προσθέσουμε τον πηγαίο κώδικα της MySQL σε κάποιο άλλο εμπορικό λογισμικό, θα πρέπει να αγοράσουμε άδεια για να το κάνουμε αυτό. Για περισσότερες πληροφορίες, θα απευθυνθούμε στην ιστοσελίδα (http://www.mysql.com/company/legal/licensing/).

- **Ο Διακομιστής Βάσεων Δεδομένων MySQL είναι πολύ γρήγορος, αξιόπιστος και εύκολος στην χρήση του.**

Ο Διακομιστής MySQL αναπτύχθηκε για να χειρίζεται τεράστιες βάσεις δεδομένων πολύ πιο γρήγορα από ότι οι άλλες υπάρχουσες λύσεις και έχει δοκιμαστεί η επιτυχία του σε προηγμένα προϊόντα λογισμικού και περιβάλλοντα για πολλά χρόνια. Σήμερα, ο διακομιστής MySQL, προσφέρει ένα εκτεταμένο σύνολο χρήσιμων συναρτήσεων. Η συνδεσιμότητα του, η ταχύτητα του και η ασφάλεια του, ενδείκνυται για την προσπέλαση του μέσω του διαδικτύου.

- **Ο Διακομιστής MySQL λειτουργεί σε περιβάλλοντα πελάτη/εξυπηρετητή (client/server) ή σε ενσωματωμένα σύστημα (embedded systems).**

Ο διακομιστής βάσεων δεδομένων MySQL, είναι ένα σύστημα client/server που αποτελείται από ένα πολυνηματικό διακομιστή SQL που υποστηρίζει διαφορετικά «άκρα επικοινωνίας», διαφορετικά προγράμματα πελάτη και βιβλιοθήκες, εργαλεία διαχείρισης και ένα ευρύ πεδίο από διεπαφές προγραμματισμού εφαρμογών (application programming interfaces (APIs)).

Ο διακομιστής MySQL προσφέρεται και ως ενσωματωμένη (embedded), πολυνηματική βιβλιοθήκη που μπορούμε να «συνδέσουμε» με τις εφαρμογές μας

για να έχουμε μια μικρότερη, γρηγορότερη και πιο εύκολη στη διαχείριση υπηρεσία.

5.2. - 5.2 - mysql-noinstall-6.0.10-alpha

Ο διακομιστής βάσεων δεδομένων MySQL 6.0.10, διανέμεται και σε έκδοση που μπορεί να χρησιμοποιηθεί χωρίς εγκατάσταση στο λειτουργικό μας σύστημα. Αυτό είναι ένα πολύ σημαντικό πλεονέκτημα, ιδιαίτερα στο λειτουργικό σύστημα των windows, στο οποίο παρουσιάζονται συχνά προβλήματα με τα εγκατεστημένα προγράμματα και τη registry (μητρώο εγγραφών). Για αυτό το λόγο και χρησιμοποιήσαμε αυτή την έκδοση-χωρίς εγκατάσταση για να αποφύγουμε τα ενδεχόμενα προβλήματα που μπορεί να προέκυπταν στο λειτουργικό μας σύστημα.

5.2.1. - Η Υπηρεσία mysqld

Το πρόγραμμα **mysqld.exe** αποτελεί τον διακομιστή βάσεων δεδομένων MySQL. Ο διακομιστής MySQL μπορεί να εκκινείται χειροκίνητα από τον χρήστη, μέσω της γραμμής εντολών σε οποιαδήποτε έκδοση των windows. Για να εκκινήσουμε την υπηρεσία-δαίμονα του διακομιστή, ανοίγουμε ένα παράθυρο κονσόλας (DOS window) και δίνουμε την εντολή:

γραμμή εντολών > ***mysqld***

Εάν θέλουμε να υποστηρίξει κάποια άλλη γλώσσα (π.χ. ελληνικά) από την προεπιλεγμένη η υπηρεσία του διακομιστή βάσεων δεδομένων μας (αγγλικά), τότε πρέπει να καλέσουμε την υπηρεσία με κάποιες συμπληρωματικές παραμέτρους:

γραμμή εντολών > ***mysqld --default-character-set=greek***
--default-collation=greek_general_ci

Για να σταματήσουμε τον διακομιστή MySQL, μπορούμε να χρησιμοποιήσουμε το εργαλείο διαχείρισης mysqladmin δίνοντας την εντολή:

γραμμή εντολών > ***mysqladmin -u root shutdown***

Βέβαια, αντί να εκκινούμε κάθε φορά χειροκίνητα την υπηρεσία-δαίμονα του διακομιστή, μπορούμε να εγκαταστήσουμε μόνο την υπηρεσία αυτή και όχι ολόκληρο το σύστημα διαχείρισης βάσεων δεδομένων, καλώντας την υπηρεσία ως εξής:

Γραμμή εντολών > ***mysqld --install***

Εάν η υπηρεσία **mysqld** δεν εκκινείται, μπορούμε να ελέγξουμε το αρχείο καταγραφής σφαλμάτων για να δούμε τα μηνύματα που ο διακομιστής έχει γράψει, για να προσδιορίσουμε τη φύση του σφάλματος εκκίνησης του δαίμονα. Το αρχείο καταγραφής σφαλμάτων, εντοπίζεται στον κατάλογο *C:\Program Files\MySQL\MySQL Server 6.0\data*. Το αρχείο καταγραφής σφαλμάτων έχει κατάληξη *.err*. Μπορούμε βέβαια να καλέσουμε τον δαίμονα-υπηρεσία με την παράμετρο **--console**, όπου σε αυτή τη περίπτωση τα μηνύματα σφάλματος θα προβληθούν κατευθείαν στον οθόνη μας, διευκολύνοντας στον πιο γρήγορο εντοπισμό του σφάλματος.

5.2.2. - Μεταβλητές συστήματος διακομιστή

Ο διακομιστής βάσεων δεδομένων MySQL διαθέτει πολλές μεταβλητές συστήματος για την παραμετροποίηση του. Κάθε μεταβλητή συστήματος διαθέτει μια προεπιλεγμένη τιμή. Οι μεταβλητές συστήματος μπορούν να διαμορφωθούν κατά την εκκίνηση του διακομιστή χρησιμοποιώντας τις επιλογές που δίνονται είτε από τη γραμμή εντολών, είτε από κάποιο αρχείο. Πολλές από αυτές τις παραμέτρους μπορούν να αλλάξουν δυναμικά, κατά τη διάρκεια που τρέχει ο διακομιστής, χρησιμοποιώντας την εντολή SET, που επιτρέπει την τροποποίηση των παραμέτρων, χωρίς να απαιτείται η διακοπή του διακομιστή και η επανεκκίνηση του. Υπάρχουν διάφοροι τρόποι για να δούμε τα ονόματα και τις τιμές των μεταβλητών του συστήματος:

- Μπορούμε να χρησιμοποιήσουμε την εντολή :

  mysqld --verbose --help

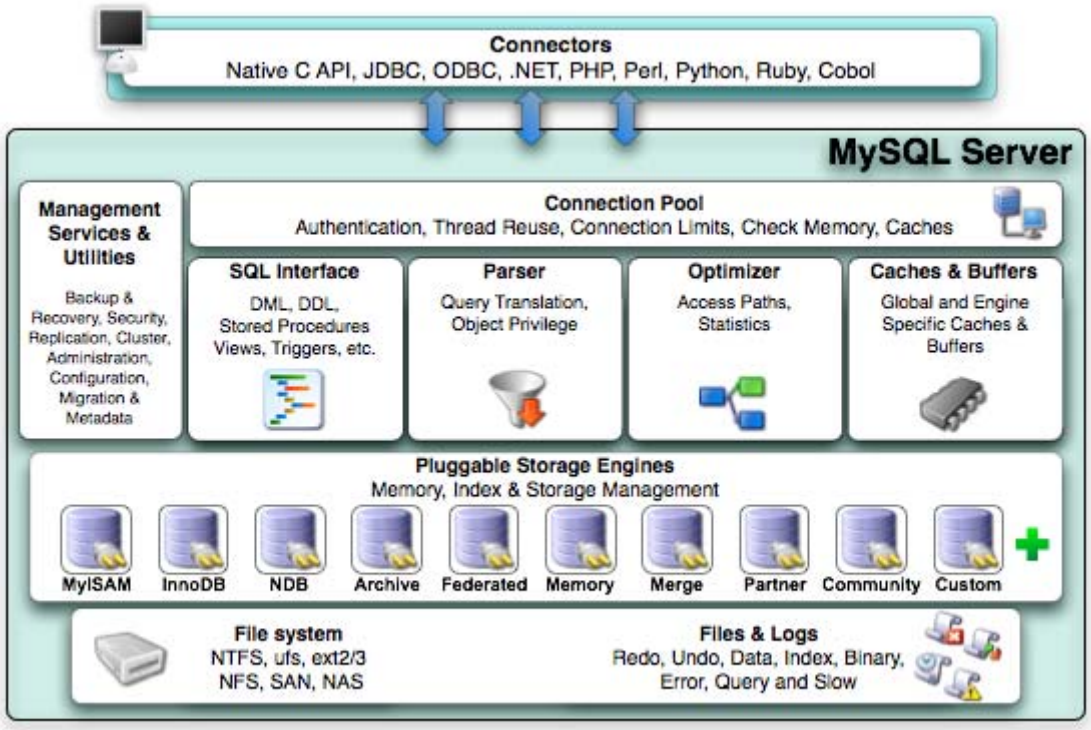
- Για να δούμε τις τρέχουσες τιμές των μεταβλητών συστήματος που χρησιμοποιεί ο διακομιστής, μπορούμε να χρησιμοποιήσουμε την εντολή

  SHOW VARIABLES

5.2.3. - Μηχανές αποθήκευσης δεδομένων

Η MySQL διαθέτει μια επεκτάσιμη (pluggable) αρχιτεκτονική μηχανών αποθήκευσης δεδομένων *(Σχήμα 5.1)* που επιτρέπει σε έναν επαγγελματία διαχειριστή βάσεων δεδομένων να επιλέξει μια ειδική μηχανή αποθήκευσης δεδομένων για κάποια συγκεκριμένη εφαρμογή που έχει εξεζητημένες απαιτήσεις

αποθήκευσης δεδομένων. Η αρχιτεκτονική του διακομιστή MySQL απομονώνει τον προγραμματισμό εφαρμογών και την διαχείριση της βάσης δεδομένων από όλες τις χαμηλού επιπέδου λεπτομέρειες υλοποίησης σε επίπεδο αποθήκευσης, παρέχοντας ένα συνεκτικό και εύκολο μοντέλο αποθήκευσης και μιας διεπαφής προγραμματισμού. Παρόλο που υπάρχουν πολλές και διαφορετικές δυνατότητες ανάμεσα στις διάφορες μηχανές αποθήκευσης η εφαρμογή είναι ανεξάρτητη αυτών.



Σχήμα 5.1 – Η αρχιτεκτονική του διακομιστή MySQL

Η επεκτάσιμη αρχιτεκτονική μηχανών αποθήκευσης παρέχει ένα βασικό σύνολο υπηρεσιών για τη διαχείριση και την υποστήριξη όλων των κοινών σημείων στις μηχανές αποθήκευσης. Οι ίδιες οι μηχανές αποθήκευσης είναι τα συστατικά του διακομιστή βάσεων δεδομένων που συνήθως εκτελούν ενέργειες στα θεμελιώδη δεδομένα που υποστηρίζονται στο φυσικό επίπεδο του διακομιστή. Η αποδοτικότητα και η προσαρμοστικότητα της αρχικτεκτονικής αυτής παρέχει τεράστια πλεονεκτήματα για εκείνους που επιθυμούν να εξυπηρετήσουν εξειδικευμένες ανάγκες των εφαρμογών τους. Ο προγραμματιστής εφαρμογών και ο διαχειριστής βάσεων δεδομένων, επικοινωνούν με τον διακομιστή βάσεων

δεδομένων διαμέσου των διεπαφών σύνδεσης και των επιπέδων υπηρεσιών που βρίσκονται πάνω από το επίπεδο των μηχανών αποθήκευσης.

Εάν μια εφαρμογή μεταβληθεί για να εξυπηρετήσει περαιτέρω ανάγκες και προστεθεί μια μηχανή, ή μηχανές αποθήκευσης για να ικανοποιήσουν τις ανάγκες αυτές, δεν θα απαιτηθεί να προστεθεί νέος κώδικας, ή να τροποποιηθούν οι υπάρχουσες διεργασίες για να λειτουργήσουν αυτές οι αλλαγές. Η αρχιτεκτονική του διακομιστή MySQL, απαλλάσσει την εφαρμογή από την πολυπλοκότητα που υπεισέρχεται με την αλλαγή μιας μηχανής αποθήκευσης, δίνοντας μας μια έξυπνη και εύκολη στη χρήση διεπαφή-περιβάλλον που υλοποιείται ανεξάρτητα της μηχανής αποθήκευσης που χρησιμοποιούμε.

5.2.3.1. - Προσθέτοντας μια νέα επέκταση μηχανής αποθήκευσης

Προτού μπορέσουμε να χρησιμοποιήσουμε μια μηχανή αποθήκευσης, η διαμοιραζόμενη επέκταση-βιβλιοθήκη της μηχανής αποθήκευσης, οφείλει να φορτωθεί στην MySQL, χρησιμοποιώντας την εντολή INSTALL PLUGIN. Για παράδειγμα, εάν η επέκταση της μηχανής αποθήκευσης λέγεται ha_example και η διαμοιραζόμενη βιβλιοθήκη λέγεται ha_example.so, θα τη φορτώσουμε με την ακόλουθη εντολή:

***INSTALL PLUGIN** ha_example **SONAME** 'ha_example.so';*

Η διαμοιραζόμενη βιβλιοθήκη θα πρέπει να βρίσκεται στον κατάλογο με τις επεκτάσεις των μηχανών αποθήκευσης, που προσδιορίζεται από την μεταβλητή συστήματος plugindir.

5.2.3.2. - Αφαιρώντας μια επέκταση μηχανής αποθήκευσης

Για να αφαιρέσουμε μια επέκταση μηχανής αποθήκευσης θα χρησιμοποιήσουμε την εντολή UNINSTALL PLUGIN:

***UNINSTALL PLUGIN** ha_example;*

Εάν αφαιρέσουμε μια επέκταση κάποιας μηχανής αποθήκευσης που απαιτούνταν από υπάρχοντες πίνακες της βάσης δεδομένων, αυτοί οι πίνακες καθίστανται μη-προσβάσιμοι, αλλά θα συνεχίσουν να υφίστανται στον σκληρό μας δίσκο. Για αυτό και πρέπει να διασφαλίζουμε ότι κανένας πίνακας δεν χρησιμοποιεί την μηχανή αποθήκευσης που θέλουμε να απομακρύνουμε.

5.2.3.3. - Οι μηχανές αποθήκευσης InnoDB και MyISAM

Η InnoDB είναι μια ασφαλής μηχανή αποθήκευσης δεδομένων για τη MySQL που διαθέτει commit, rollback αλλά και δυνατότητες ανάκτησης μετά από καταστροφή για την προστασία των δεδομένων του χρήστη. Η ποσότητα δεδομένων που κλειδώνει λόγω συγχρονισμού στους πίνακες, στη μηχανή αποθήκευσης InnoDB, γίνεται σε επίπεδο σειράς – ένα στυλ με τα μικρότερα δυνατά κλειδώματα - που οδηγεί σε αύξηση του συγχρονισμού πολλών χρηστών και της απόδοσης του διακομιστή. Η μηχανή αποθήκευσης InnoDB κρατά τα δεδομένα του χρήστη σε ομάδες δεικτών για να μειώσει τις διαδικασίες I/O για κοινά αιτήματα που βασίζονται σε κύρια κλειδιά. Για να διατηρηθεί η ακεραιότητα των δεδομένων, η InnoDB υποστηρίζει επίσης και τους περιορισμούς ακεραιότητας ξένου κλειδιού. Μπορείτε ελεύθερα να συνδυάσετε InnoDB πίνακες με άλλους πίνακες, διαφορετικών μηχανών αποθήκευσης της MySQL, ακόμη και μέσα στην ίδια εντολή.

Πίνακας 5.1 - Χαρακτηριστικά της μηχανής αποθήκευσης InnoDB

Όρια Αποθήκευσης	**64TB**	Συναλλαγές	**Ναι**	Ποσότητα Δεδομένων που κλειδώνεται λόγω συγχρονισμού	**Σε επίπεδο σειράς**
MVCC	**Ναι**	Υποστήριξη τύπων δεδομένων που χρησιμοποιούνται από Γεωγραφικά Συστήματα Πληροφοριών (G.I.S)	**Ναι**	Υποστήριξη δεικτών που χρησιμοποιούνται από Γεωγραφικά Συστήματα Πληροφοριών (G.I.S)	**Όχι**
Ευρετήρια B-δέντρο (BTree Indexes)	**Ναι**	Ευρετήρια κατακερματισμού	**Όχι**	Ευρετήρια αναζήτησης πλήρους κειμένου	**Όχι**
Συγκεντρωτικά Ευρετήρια	**Ναι**	Caches δεδομένων	**Ναι**	Caches δεικτών	**Ναι**
Συμπίεση δεδομένων	**Ναι**	Κρυπτογραφημένα δεδομένα	**Ναι**	Υποστήριξη Cluster βάσης	**'Όχι**
Υποστήριξη υλοποίησης σε server παρά στη μηχανή αποθήκευσης συναλλαγών	**Ναι**	Υποστήριξη Ξένου Κλειδιού	**Ναι**	Backup / ανάκτηση από τον server	**Ναι**

Πίνακας 5.1(Συνέχεια) - Χαρακτηριστικά της μηχανής αποθήκευσης InnoDB

Υποστήριξη προσωρινής αποθήκευσης ερωτημάτων	**Ναι**	Ενημέρωση στατιστικών για τα δεδομένα του λεξικού	**Ναι**		

Η InnoDB έχει σχεδιαστεί για υψηλή απόδοση κατά την επεξεργασία τεράστιου όγκου δεδομένων.

Η MyISAM είναι η προεπιλεγμένη μηχανή αποθήκευσης δεδομένων. Είναι βασισμένη στη αρχική ISAM μηχανή αποθήκευσης. Στην ISAM, κάθε πίνακας είναι αποθηκευμένος στο δίσκο σε τρία αρχεία, τα ονόματα των οποίων ξεκινούν με το όνομα του πίνακα και μια επέκταση που αναφέρει το τύπο αρχείου. Τα .frm αρχεία αποθηκεύουν τον ορισμό του πίνακα. Τα αρχεία που αποθηκεύουν τα δεδομένα έχουν .ISD επέκταση, ενώ τα αρχεία των δεικτών έχουν κατάληξη .ISM.

Πίνακας 5.2 - Χαρακτηριστικά της μηχανής αποθήκευσης MyISAM

Όρια Αποθήκευσης	**256TB**	Συναλλαγές	**Δεν**	Ποσότητα Δεδομένων που κλειδώνεται λόγω συγχρονισμού	**Σε επίπεδο Πίνακα**
MVCC	**Δεν**	Υποστήριξη τύπων δεδομένων που υποστηρίζονται από Γεωγραφικό σύστημα πληροφοριών	**Ναι**	Υποστήριξη δεικτών που υποστηρίζονται από Γεωγραφικό σύστημα πληροφοριών	**Ναι**
Ευρετήρια B-δέντρο (BTree Indexes)	**Ναι**	Ευρετήρια κατακερματισμού	**Δεν**	Ευρετήρια αναζήτηση πλήρους κειμένου	**Ναι**
Συγκεντρωτικά Ευρετήρια	**Δεν**	Caches δεδομένων	**Δεν**	Caches δεικτών	**Ναι**
Συμπίεση δεδομένων	**Ναι**	Κρυπτογραφημένα δεδομένα	**Ναι**	Υποστήριξη Cluster βάσης	**Δεν**

Πίνακας 5.2 (Συνέχεια) - Χαρακτηριστικά της μηχανής αποθήκευσης MyISAM

Υποστήριξη υλοποίησης σε server παρά στη μηχανή αποθήκευσης συναλλαγών	**Ναι**	Υποστήριξη Ξένου Κλειδιού	**Δεν**	Backup / ανάκτηση από τον server	**Ναι**
Υποστήριξη προσωρινής αποθήκευσης ερωτημάτων	**Ναι**	Ενημέρωση στατιστικών για τα δεδομένα του λεξικού	**Ναι**		

Οι ISAM μηχανές αποθήκευσης χρησιμοποιούν ευρετήρια B-Tree. Έχουν τις εξής ιδιότητες:

- Συμπιεσμένα και ορισμένου μήκους κλειδιά
- Σταθερό και δυναμικό μήκος εγγραφής
- 16 δείκτες ανά πίνακα, με 16 μέρη ανά κύριο κλειδί
- Μέγιστο μήκος κλειδιού 256 bytes (προεπιλογή)
- Οι τιμές των δεδομένων αποθηκεύονται στο μηχάνημα. Αυτό είναι γρήγορο, αλλά εξαρτάται από το μηχάνημα και το λειτουργικό σύστημα

5.2.4. - Το εργαλείο σύνδεσης στο διακομιστή MySQL

Για να μπορέσει ένα πρόγραμμα πελάτη να συνδεθεί στον διακομιστή βάσεων δεδομένων MySQL, θα πρέπει να χρησιμοποιήσει τις κατάλληλες παραμέτρους σύνδεσης, όπως το όνομα του υπολογιστή, ή τη διεύθυνση διαδικτύου που τρέχει ο διακομιστής και φυσικά το όνομα χρήστη και τον κωδικό πρόσβασης του λογαριασμού χρήστη που θα χρησιμοποιήσει. Κάθε παράμετρος διαθέτει μια προεπιλεγμένη τιμή, αλλά μπορούμε να τις υπερβούμε χρησιμοποιώντας τις παραμέτρους του προγράμματος στη γραμμή εντολών, ή μέσω ενός αρχείου.

Εάν δεν υπάρχουν επιλογές παραμέτρων, τότε εφαρμόζονται οι προεπιλεγμένες τιμές:

- Το προεπιλεγμένο όνομα υπολογιστή είναι το τοπικό σύστημα (localhost).
- Το προεπιλεγμένο όνομα χρήστη είναι ODBC στα Windows, ή το όνομα εισόδου του χρήστη στο Unix.

- Δεν χρησιμοποιείται κανένα συνθηματικό εάν δεν δοθεί ούτε ο διακόπτης -p είτε ο διακόπτης --password.

Για να προσδιορίσουμε το όνομα υπολογιστή που τρέχει ο διακομιστής, αλλά και το όνομα χρήστη και τον κωδικό του λογαριασμού μας, θα δώσουμε την εντολή:

Γραμμή εντολών> ***mysql --host=****όνομα_υπολογιστή*
--user= *όνομα_χρήστη* ***--password=****συνθηματικό*

είτε την εντολή

Γραμμή εντολών> ***mysql -h*** *όνομα_υπολογιστή* ***-u*** *όνομα_χρήστη*
–p *συνθηματικό*

5.2.5. - Τροποποίηση δικαιωμάτων λογαριασμών χρηστών

Για να τροποποιήσουμε τα δικαιώματα των λογαριασμών των χρηστών, να δημιουργήσουμε νέους, ή και να διαγράψουμε κάποιους άλλους, θα πρέπει να τροποποιήσουμε τον πίνακα user της βάσης δεδομένων mysql. Σε αυτό τον πίνακα βρίσκονται όλοι οι λογαριασμοί χρηστών του διακομιστή βάσεων δεδομένων. Κάθε εγγραφή του πίνακα αυτού περιέχει το όνομα, το συνθηματικό, τα δικαιώματα του χρήστη για κάθε ενέργεια προς τη βάση δεδομένων (εισαγωγή, ενημέρωση, δημιουργία, διαγραφή, επιλογή κτλ), αλλά και άλλες παράμετροι, όπως π.χ. ο μέγιστος αριθμός συνδέσεων για τον κάθε χρήστη.

Εάν επιθυμούμε π.χ. να τροποποιήσουμε το συνθηματικό ενός υπάρχοντος χρήστη, θα δώσουμε την εντολή:

Γραμμή εντολών> ***UPDATE mysql.user***
SET Password=PASSWORD("*10022910****")***
WHERE User="*root****";***

Και φυσικά την εντολή για να λάβουν χώρα άμεσα οι αλλαγές μας:

Γραμμή εντολών> ***FLUSH PRIVLEGES;***

5.2.6. - Ορισμός δικαιωμάτων απομακρυσμένης σύνδεσης χρήστη

Εάν τώρα θελήσουμε να δώσουμε δικαίωμα απομακρυσμένης πρόσβασης σε κάποιο χρήστη του διακομιστή μας, θα πρέπει να

παραχωρήσουμε το δικαίωμα αυτό για την βάση δεδομένων, ή τις βάσεις δεδομένων που θα χρησιμοποιήσει. Αλλά επίσης θα πρέπει και να ορίσουμε την διεύθυνση διαδικτύου από την οποία θα συνδεθεί αυτός ο χρήστης και να θέσουμε ρητώς τις επιτρεπτές ενέργειες για αυτό τον λογαριασμό προς την, ή τις βάσεις δεδομένων του διακομιστή μας. Η εντολή που θα χρησιμοποιήσουμε για την απομακρυσμένη πρόσβαση σε όλους τους πίνακες μιας συγκεκριμένης βάσης δεδομένων, είναι η:

Γραμμή εντολών> ***GRANT SELECT,INSERT,UPDATE,DELETE,***
ALTER,CREATE,DROP
ON *sensingsystem.** ***to*** *root@192.168.1.64*
IDENTIFIED BY "*10022910****";***

Εάν τώρα θελήσουμε να δώσουμε δικαιώματα σε ορισμένους πίνακες κάποιας συγκεκριμένης βάσης δεδομένων, αυτό που θα αλλάξει θα είναι ο χαρακτήρας μπαλαντέρ *, ο οποίος θα αντικατασταθεί με το/τα όνομα/τα του/των συγκεκριμένου/ων πίνακα/άκων που επιθυμούμε. Εάν θέλουμε να παραχωρήσουμε δικαιώματα για όλες τις βάσεις δεδομένων του διακομιστή μας, η εντολή θα πάρει τη μορφή:

Γραμμή εντολών> ***GRANT SELECT,INSERT,UPDATE,DELETE,***
ALTER,CREATE,DROP
ON *.* to *root@192.168.1.64*
IDENTIFIED BY "10022910";

5.2.7. - Συνοπτικός οδηγός αντιγράφων ασφάλειας (Backup)

Δημιουργία αντιγράφων ασφαλείας με τα εργαλεία *mysqldump* ή *mysqlhotcopy*

Μια τεχνική για την δημιουργία αντιγράφων ασφαλείας της βάσης δεδομένων, είναι να χρησιμοποιήσουμε το πρόγραμμα **mysqldump**, ή το script **mysqlhotcopy**. Το **mysqldump** είναι πιο γενικό εργαλείο, διότι μπορεί να δημιουργήσει αντίγραφα ασφαλείας όλων των ειδών των πινάκων. Το script **mysqlhotcopy** δουλεύει μόνο με ορισμένες μηχανές αποθήκευσης της MySQL. Για τη δημιουργία πλήρους αντίγραφου ασφάλειας της βάσης δεδομένων μας, δίνουμε την εντολή:

Γραμμή εντολών > ***mysqldump –tab =*** */διαδρομή/προς/κάποιο/φάκελο*

***--opt** Όνομα_Βάσης_Δεδομένων*

Μπορούμε να χρησιμοποιήσουμε τις εντολές BACKUP DATABASE και RESTORE όπως παρακάτω:

- Μέσω της εντολής BACKUP DATABASE δημιουργούμε αντίγραφα ασφάλειας μίας, ή περισσότερων βάσεων δεδομένων σε ένα αρχείο:

  ***BACKUP DATABASE** ονομα_1ης_βάσης_δεδομένων **TO** '/tmp/backupfile';*

  Για να δημιουργήσουμε αντίγραφα ασφάλειας περισσότερων της μιας βάσης δεδομένων, μπορούμε να διαχωρίσουμε τα ονόματα των βάσεων δεδομένων με κόμματα:

  BACKUP DATABASE
  ονομα_1ης_βάσης_δεδομένων, όνομα_2ης_βάσης_δεδομένων
  ***TO** '/tmp/backupfile';*

  Εάν θέλουμε να φτιάξουμε αντίγραφο ασφάλειας όλων των βάσεων δεδομένων του διακομιστή βάσεων δεδομένων μας, θα χρησιμοποιήσουμε τον χαρακτήρα-μπαλαντέρ * ως συντόμευση:

  BACKUP DATABASE * TO '/tmp/mybackupfile';

- Μέσω της εντολής RESTORE μπορούμε να ανακτήσουμε βάσεις δεδομένων χρησιμοποιώντας το αρχείο αντίγραφο ασφαλείας τους:

  ***RESTORE FROM** '/tmp/backupfile';*

  Η εντολή BACKUP DATABASE δημιουργεί αντίγραφα ασφάλειας μιας βάσης δεδομένων, των ορισμών των πινάκων, των δεδομένων των πινάκων, των αποθηκευμένων συναρτήσεων, των triggers, των γεγονότων και των όψεων. Οι προσωρινοί πίνακες φυσικά δεν συμπεριλαμβάνονται.

  Από την έκδοση MySQL 6.0.7, τα δικαιώματα της βάσης δεδομένων, αποθηκεύονται και ανακτούνται σύμφωνα με τους εξής κανόνες:

- Η εντολή BACKUP DATABASE αποθηκεύει τα δικαιώματα για τη βάση, ή τις βάσεις δεδομένων στο αρχείο εικόνα του αντιγράφου ασφάλειας.

- Αποθηκεύονται μόνο τα δικαιώματα σε επίπεδο πίνακα, στήλης, ή ρουτίνας. Τα Global δικαιώματα δεν αποθηκεύονται διότι δεν σχετίζονται με κάποια συγκεκριμένη βάση δεδομένων που δημιουργούμε αντίγραφο ασφάλειας της.

Η ανάκτηση των δικαιωμάτων για λογαριασμούς που δεν υφίστανται πλέον, δεν επιτρέπεται, γιατί κάτι τέτοιο θα δημιουργούσε λογαριασμούς που δεν θα διέθεταν κωδικό πρόσβασης και αυτό θα αποτελούσε ένα κενό ασφάλειας της βάσης δεδομένων.

Η εντολή BACKUP DATABASE δεν δημιουργεί αντίγραφο ασφαλείας της βάσης δεδομένων «mysql». Αυτή η βάση δεδομένων περιέχει τους λογαριασμούς χρηστών του διακομιστή βάσεων δεδομένων, τα δικαιώματα τους και φυσικά άλλες πληροφορίες του συστήματος. Για να δημιουργήσουμε ένα πλήρες αντίγραφο ασφάλειας του διακομιστή βάσεων δεδομένων, το οποίο θα περιλαμβάνει και τις πληροφορίες των λογαριασμών επιπροσθέτως των δεδομένων των πινάκων, θα πρέπει να χρησιμοποιήσουμε την εντολή BACKUP DATABASE μαζί με το εργαλείο δημιουργίας αντιγράφων ασφάλειας **mysqldump**. Στις παρακάτω οδηγίες η λέξη μονοπάτι προσδιορίζει τον κατάλογο στον οποίο αποθηκεύουμε τα αντίγραφα ασφάλειας μας.

Μέσω του εργαλείου **mysqldump**, μπορούμε να φτιάξουμε αντίγραφο ασφάλειας της βάσης δεδομένων «mysql».

1. *Γραμμή εντολών>* ***mysqldump --databases mysql >*** *μονοπάτι/mysql-db.sql*
2. Χρησιμοποιούμε έπειτα με την εντολή BACKUP DATABASE παίρνουμε ασφαλείας των δεδομένων των υπολοίπων βάσεων δεδομένων.

   Γραμμή εντολών> ***BACKUP DATABASE * TO*** *'μονοπάτι/other-dbs.bak****';***

Αργότερα, για να ανακτήσουμε από το αντίγραφο ασφαλείας το στιγμιότυπο του διακομιστή μας, ακολουθούμε τα παρακάτω βήματα:

1. Για να ανακτήσουμε τους λογαριασμούς χρηστών, φορτώνουμε το αρχείο που δημιουργήσαμε μέσω του προγράμματος πελάτη **mysql** :

   Γραμμή εντολών> ***mysql -u root -p <*** *μονοπάτι/mysql-db.sql*
2. Για να ανακτήσουμε τα δεδομένα άλλων βάσεων δεδομένων, πέραν της «mysql», θα χρησιμοποιήσουμε την εντολή RESTORE δίνοντάς της ως όρισμα το αρχείο εικόνα που δημιουργήσαμε με την εντολή BACKUP DATABASE:

   Γραμμή εντολών> ***RESTORE FROM*** *'μονοπάτι/other-dbs.bak****';***

ΚΕΦΑΛΑΙΟ 6
Η Βάση Δεδομένων sensingsystem

6.1. - Η δημιουργία της βάσης δεδομένων

Η δημιουργία μιας νέας βάσης δεδομένων, ενός νέου σχήματος δηλαδή στη MySQL, είναι δυνατή με την εντολή create database. Εάν επιθυμούμε η βάση δεδομένων μας να υποστηρίζει κάποια άλλη γλώσσα (π.χ. ελληνικά) από την προεπιλεγμένη (αγγλικά), θα πρέπει να χρησιμοποιήσουμε και κάποιες συμπληρωματικές παραμέτρους στην κλήση της εντολής δημιουργίας της βάσης δεδομένων μας.

Γραμμή εντολών> ***CREATE DATABASE*** *sensingsystem*
character set greek collate greek_general_ci;

6.2. - Οι πίνακες της βάσης δεδομένων

Οι πίνακες της βάσης δεδομένων εξυπηρετούν τις πληροφοριακές ανάγκες και της κύριας εφαρμογής, αλλά και της διαδικτυακής εφαρμογής του συστήματος μας. Οι πίνακες αποτελούν τις πληροφοριακές δεξαμενές των δυο εφαρμογών, καθώς τα προγράμματα στηρίζουν την λειτουργικότητα τους στην εγγραφή και στην ανάγνωση των πληροφοριών αυτών. Υπάρχουν συνολικά δέκα πίνακες: ο πίνακας με την ταυτότητα του διαχειριστή της εφαρμογής, ο πίνακας με την ταυτότητα του λογαριασμού ηλεκτρονικής αλληλογραφίας του ίδιου του συστήματος επιτήρησης, ο πίνακας των συσκευών με τις οποίες συνδέεται η εφαρμογή, ο πίνακας με τις παραμετροποιήσεις των σειριακών συνδέσεων που πραγματοποιεί η εφαρμογή για να επικοινωνήσει με τις συσκευές, ο πίνακας συμβάντων στον οποίο αποθηκεύονται τα συμβάντα στο περιβάλλον της διαδικτυακής εφαρμογής του συστήματος μας, ο πίνακας των αισθητήρων του συστήματος, ο πίνακας των αναλογικών αισθητήρων, ο πίνακας των ψηφιακών αισθητήρων, ο πίνακας καταγραφής των αναλογικών μετρήσεων, ο πίνακας καταγραφής των ψηφιακών μετρήσεων.

6.2.1. - Ο πίνακας applicationowner

Ο πίνακας αυτός, αφορά στο λογαριασμό του διαχειριστή της διαδικτυακής εφαρμογής μας. Τα πεδία του είναι τα εξής: το όνομα χρήστη και το συνθηματικό με τα οποία συνδέεται στην διαδικτυακή εφαρμογή αλλά και το κινητό τηλέφωνο, όπως και τον λογαριασμό ηλεκτρονικού ταχυδρομείου, στα οποία θα ειδοποιηθεί όταν θα παραστεί η ανάγκη.

Γραμμή εντολών > ***CREATE TABLE*** *applicationowner(*

webusername ***VARCHAR (20) NOT NULL****,*

webpassword ***VARCHAR (30) NOT NULL****,*

mobilenumber ***VARCHAR (15) NOT NULL****,*

emailaddress ***VARCHAR (100) NOT NULL****,*

PRIMARY KEY*(webusername))*

character set greek collate greek_general_ci;

6.2.2. - Ο πίνακας mailparameters

Ο πίνακας αυτός, αφορά στο λογαριασμό ηλεκτρονικού ταχυδρομείου που χρησιμοποιεί η κύρια εφαρμογή για την πραγματοποίηση της αποστολής μηνύματος στο λογαριασμό ηλεκτρονικού ταχυδρομείου του διαχειριστή. Τα πεδία του πίνακα είναι τα εξής: η διεύθυνση ηλεκτρονικού ταχυδρομείου, το όνομα χρήστη, ο κωδικός πρόσβασης, ο διακομιστής ηλεκτρονικής αλληλογραφίας και η πόρτα για την σύνδεση με τον διακομιστή μέσω SSL (Secure Socket Layer). Ο διαχειριστής μέσω της διαδικτυακής εφαρμογής, μπορεί να τροποποιήσει οποιοδήποτε στιγμή θελήσει, οποιοδήποτε πεδία, ή και ολόκληρη την εγγραφή.

Γραμμή εντολών > ***CREATE TABLE*** *mailparameters(*

emailaddress ***VARCHAR (100) NOT NULL****,*

smtpserver ***VARCHAR (40) NOT NULL****,*

mailuser ***VARCHAR (50) NOT NULL****,*

mailpassword ***VARCHAR (40) NOT NULL****,*

sslport ***INTEGER NOT NULL****,*

PRIMARY KEY*(emailaddress))*

character set greek collate greek_general_ci;

6.2.3. - Ο πίνακας sensors

Ο πίνακας αυτός, αφορά στο σύνολο των αισθητήρων: αναλογικών και ψηφιακών. Περιλαμβάνει, για κάθε έναν από τους 6 αισθητήρες, ένα προσδιοριστικό όνομα, ένα μήνυμα περιγραφής της τοποθεσίας που είναι τοποθετημένος, αλλά και επίσης το μήνυμα ειδοποίησης που θα λάβει ο διαχειριστής στο κινητό του τηλέφωνο και στο λογαριασμό ηλεκτρονικού ταχυδρομείου όταν ξεπεραστούν τα όρια-τιμές που έχει θέσει για κάθε έναν αισθητήρα. Ο διαχειριστής μέσω της διαδικτυακής εφαρμογής μπορεί να τροποποιήσει για κάθε αισθητήρα το πεδίο περιγραφής της τοποθεσίας του και το πεδίο που αφορά στο μήνυμα το οποίο θα λάβει στο κινητό του τηλέφωνο και στο λογαριασμό ηλεκτρονικής αλληλογραφίας του.

Γραμμή εντολών > ***CREATE TABLE*** *sensors(*
sensorid ***VARCHAR (10) NOT NULL****,*
info ***VARCHAR (100) NOT NULL****,*
alertmessage ***VARCHAR (200) NOT NULL****,*
PRIMARY KEY*(sensorid))*
character set greek collate greek_general_ci;

6.2.4. - Ο πίνακας analogsensors

Ο πίνακας αυτός αφορά μόνο στους τέσσερις από τους έξι αισθητήρες, τους αναλογικούς. Τα πεδία που περιέχει είναι τα εξής: το προσδιοριστικό όνομα καθενός από τους αισθητήρες και φυσικά την τιμή όριο του καθενός (θερμοκρασία Κελσίου για τους δυο πρώτους, ποσοστό επί τοις εκατό για τον αισθητήρα υγρασίας και Volts για τον αισθητήρα τάσης δικτύου). Ο διαχειριστής μέσω της διαδικτυακής εφαρμογής, μπορεί να τροποποιήσει οποιαδήποτε στιγμή θελήσει την τιμή του ορίου για κάθε έναν από τους αναλογικούς αισθητήρες.

Γραμμή εντολών > ***CREATE TABLE*** *analogsensors(*
analogsensorid ***VARCHAR (10) NOT NULL****,*
limitvalue ***DOUBLE****,*
FOREIGN KEY*(analogsensorid)*
REFERENCES *sensors(sensorid)* ***ON UPDATE NO ACTION****,*
PRIMARY KEY*(analogsensorid))*
character set greek collate greek_general_ci;

6.2.5. - Ο πίνακας digitalsensors

Ο πίνακας αυτός αφορά μόνο στους δύο από τους έξι αισθητήρες, τους ψηφιακούς. Τα πεδία που περιέχει είναι τα εξής: το προσδιοριστικό όνομα καθενός από τους αισθητήρες και φυσικά την τιμή όριο του καθενός (1 - true).

Γραμμή εντολών > ***CREATE TABLE*** *digitalsensors(*

digitalsensorid ***VARCHAR (10) NOT NULL****,*

limitvalue ***INTEGER NOT NULL****,*

FOREIGN KEY *(digitalsensorid)*

REFERENCES *sensors(sensorid)*

ON UPDATE NO ACTION*,*

PRIMARY KEY*(digitalsensorid))*

character set greek collate greek_general_ci;

6.2.6. - Ο πίνακας analogmeasurement

Ο πίνακας αυτός αφορά μόνο στις μετρήσεις των αναλογικών αισθητήρων. Τα πεδία που περιέχει κάθε εγγραφή του πίνακα είναι το προσδιοριστικό όνομα για κάθε έναν από τους τέσσερις αναλογικούς αισθητήρες, την χρονική στιγμή λήψης της τιμής της μέτρησης από την εξωτερική συσκευή και φυσικά την ίδια την τιμή για κάθε έναν αισθητήρα. Αυτός ο πίνακας χρησιμοποιείται από την εφαρμογή καταγραφής, για την αποθήκευση των μετρήσεων των αισθητήρων και από την διαδικτυακή εφαρμογή για την παρουσίαση των μετρήσεων σε πραγματικό χρόνο, κάθε 10 δευτερόλεπτα.

Γραμμή εντολών > ***CREATE TABLE*** *analogmeasurement (*

analogsensorid ***VARCHAR(10) NOT NULL****,*

measurementtime ***DATETIME NOT NULL****,*

captureddata ***DOUBLE****,*

PRIMARY KEY*(analogsensorid, measurementtime),*

FOREIGN KEY *(analogsensorid)*

REFERENCES *analogsensors(analogsensorid)*

ON UPDATE NO ACTION*)*

character set greek collate greek_general_ci;

6.2.7. - Ο πίνακας digitalmeasurement

Ο πίνακας αυτός αφορά μόνο την κατάσταση των ψηφιακών αισθητήρων. Τα πεδία που περιέχει κάθε εγγραφή του πίνακα είναι το προσδιοριστικό όνομα για κάθε έναν από τους δύο ψηφιακούς αισθητήρες, την χρονική στιγμή λήψης της κατάστασής τους από την εξωτερική συσκευή και φυσικά την ίδια την κατάσταση του κάθε αισθητήρα (1 – true, 0 - false). Αυτός ο πίνακας χρησιμοποιείται από την εφαρμογή καταγραφής, για την αποθήκευση της κατάστασης των αισθητήρων.

Γραμμή εντολών >CREATE TABLE *analogmeasurement* ***(***

analogsensorid ***VARCHAR(10) NOT NULL,***

measurementtime ***DATETIME NOT NULL,***

captureddata DOUBLE,

PRIMARY KEY *(analogsensorid, measurementtime),*

FOREIGN KEY *(analogsensorid)*

REFERENCES *analogsensors(analogsensorid)*

ON UPDATE NO ACTION)

character set greek collate greek_general_ci;

6.2.8. - Ο πίνακας logtable

Ο πίνακας αυτός, αφορά στην καταγραφή των γεγονότων/συμβάντων μέσα στο περιβάλλον της διαδικτυακής εφαρμογής. Όλα τα γεγονότα επιτυχίας και αποτυχίας καταγράφονται σε αυτό τον πίνακα: η επιτυχής σύνδεση του χρήστη στην εφαρμογή, η αλλαγή της ταυτότητας του, η αλλαγή του αριθμού κινητού τηλεφώνου και του λογαριασμού ηλεκτρονικής αλληλογραφίας που θα ειδοποιηθεί σε περίπτωση υπέρβασης των τιμών ορίων των αισθητήρων, η αλλαγή του λογαριασμού ηλεκτρονικής αλληλογραφίας του συστήματος επιτήρησης, όπως και επίσης η αλλαγή των ορίων για τις τιμές των αισθητήρων και των μηνυμάτων ειδοποίησης του διαχειριστή είναι μερικά από τα συμβάντα που καταγράφονται μαζί με την χρονική στιγμή την οποία συντελέστηκαν, αλλά και την διαδικτυακή διεύθυνση του υπολογιστή από τον οποίο προήλθαν οι ενέργειες προς την εφαρμογή.

Γραμμή εντολών > ***CREATE TABLE*** *logtable(*

timestamp ***VARCHAR(40) NOT NULL****,*

event ***VARCHAR(100) NOT NULL****,*

ipaddress ***VARCHAR(30) NOT NULL****,*

PRIMARY KEY*(timestamp))*

character set greek collate greek_general_ci;

6.2.9. - Ο πίνακας devices

Ο πίνακας αυτός αφορά στις δύο συσκευές με τις οποίες επικοινωνεί το πρόγραμμα καταγραφής και ειδοποίησης. Η μια είναι η εξωτερική συσκευή με τους αισθητήρες και η άλλη το κινητό τηλέφωνο που χρησιμοποιείται ως GSM μόντεμ για την αποστολή του μηνύματος στο κινητό του διαχειριστή. Οι εγγραφές του πίνακα αυτού είναι δύο, μία για κάθε συσκευή: τα πεδία κάθε εγγραφής, περιέχουν το προσδιοριστικό όνομα της συσκευής π.χ. PIC_KIT-Board, την σειριακή διεπαφή που θα χρησιμοποιηθεί για τη πραγματοποίηση της σύνδεσης με την συσκευή π.χ. COM2 και τέλος μια περιγραφή του τύπου της συσκευής π.χ. gsm_modem_device. Όταν ο χρήστης της εφαρμογής επιλέξει “FLUSH DEVICES”, αυτομάτως διαγράφονται οι εγγραφές του πίνακα αυτού και μπορεί μέσω των επιλογών “SENSING DEVICE SETUP” και “MODEM DEVICE SETUP”, να ορίσει νέες παραμέτρους για τις δύο συσκευές και να προστεθούν νέες εγγραφές σε αυτόν.

Γραμμή εντολών > ***CREATE TABLE*** *devices (*

devicename ***VARCHAR(50) NOT NULL****,*

interface ***VARCHAR(10) NOT NULL****,*

devicetype ***VARCHAR(50) NOT NULL****,*

PRIMARY KEY*(devicename),*

FOREIGN KEY*(interface)*

REFERENCES *serialportparameters(interfacename)*

ON UPDATE CASCADE ON DELETE CASCADE*)*

character set greek collate greek_general_ci engine=InnoDB;

6.2.10. - Ο πίνακας serialportparameters

Ο πίνακας αυτός αφορά στις παραμέτρους για την σειριακή επικοινωνία του προγράμματος με τις δύο συσκευές που χρησιμοποιεί (την εξωτερική συσκευή με τους αισθητήρες και το κινητό τηλέφωνο). Οι εγγραφές αυτού του πίνακα είναι τόσες, όσες και οι συσκευές που είναι αποθηκευμένες στον πίνακα devices. Τα πεδία κάθε εγγραφής περιέχουν την σειριακή διεπαφή (πόρτα) του υπολογιστή, που θα συνδεθεί το πρόγραμμα, ώστε να επικοινωνήσει με την συσκευή, τον

ρυθμό μετάδοσης δεδομένων μεταξύ του υπολογιστή και της συσκευής, τον αριθμό των bits που χρησιμοποιούνται για την μετάδοση των δεδομένων, τον αριθμό των bits που χρησιμοποιούνται για την σήμανση του stop στην μετάδοση των δεδομένων, ο τύπος του «ελέγχου ισοτιμίας» που θα χρησιμοποιηθεί για την μετάδοση των δεδομένων και τέλος, ο τύπος ελέγχου που θα χρησιμοποιηθεί, για την αποστολή των δεδομένων. Ο πίνακας αυτός χρησιμοποιείται από την εφαρμογή καταγραφής

Γραμμή εντολών > ***CREATE TABLE*** *serialportparameters(*

interfacename ***varchar(10) NOT NULL,***
baudrate ***INTEGER NOT NULL****,*
databits ***INTEGER NOT NULL****,*
stopbits ***INTEGER NOT NULL****,*
paritybits ***INTEGER NOT NULL****,*
flowcontrol ***INTEGER NOT NULL****,*
PRIMARY KEY*(interfacename))*
character set greek collate greek_general_ci engine=InnoDB;

Τρίτη Ενότητα

Η ΚΥΡΙΑ ΕΦΑΡΜΟΓΗ

(Έλεγχος – Καταγραφή - Ειδοποίηση)

ΚΕΦΑΛΑΙΟ 7
Οι τεχνολογίες και το περιβάλλον προγραμματισμού

7.1. - GSM MODEM & AT ΕΝΤΟΛΕΣ

Οι συσκευές που χρησιμοποιούν τα δίκτυα GSM είναι και τα GSM μόντεμ. Το GSM μόντεμ είναι μια επέκταση ενός κλασσικού ενσύρματου μόντεμ. Η διαφορά τους είναι ότι ένα GSM μόντεμ συνδέεται σ' ένα δίκτυο ασύρματα μέσω ενός δικτύου κινητής τηλεφωνίας της επιλογής μας. Για να συνδεθεί σε ένα δίκτυο χρειάζεται η χρήση μιας ενεργής κάρτας SIM. Τα μόντεμ δέχονται ορισμένες AT εντολές. Ένα GSM μόντεμ μπορεί να δεχθεί όλες τις AT εντολές των κλασσικών μόντεμ, αλλά λόγω της εξελιγμένης τεχνολογίας του δέχεται περισσότερες και μεταγενέστερες AT εντολές. Με τη βοήθεια των AT εντολών μπορούμε να διαχειριστούμε τη μνήμη της κάρτας SIM και του μόντεμ και να εκτελέσουμε διάφορες λειτουργίες μέσω του GSM μόντεμ.

7.1.1. - Οι AT Εντολές που χρησιμοποιήθηκαν στην εφαρμογή μας

AT+CMGF=1 :Ορίζει το format του SMS σε txt mode.

AT+CMGS="+30xxxxxxxxxx" :Ορίζει τον παραλήπτη του γραπτού μηνύματος

Μετά από αυτή την εντολή το μόντεμ μας επιστρέφει τον χαρακτήρα **>**. Μετά από αυτόν το χαρακτήρα μπορούμε να γράψουμε το κείμενο ενός SMS. Το SMS αυτό μπορεί να αποτελείται από 160 χαρακτήρες το μέγιστο. Η εντολή τερματίζεται με **Ctrl-Z** και αποστέλλεται το SMS.

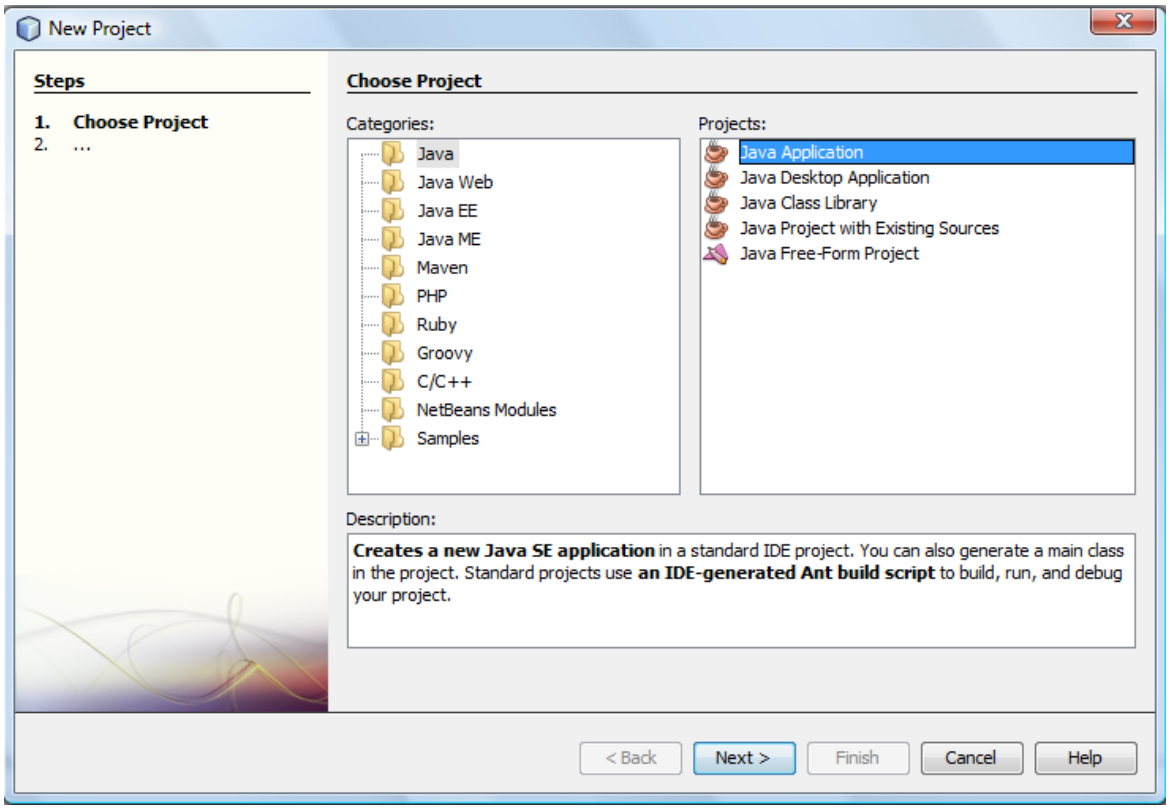
7.2. - NetBeans IDE 6.7

Το NetBeans αποτελεί ένα ελεύθερο, ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού σε Java. Μας δίνει τη δυνατότητα να δημιουργήσουμε αυτόνομες εφαρμογές, εφαρμογές ιστού, εφαρμογές κινητών τηλεφώνων και πολλά άλλα.

Διαθέτει ενσωματωμένους διακομιστές εφαρμογών ιστού, αλλά και εργαλεία για τη σύνδεση σε βάσεις δεδομένων και την εκτέλεση ερωτημάτων πάνω στους πίνακες της. Το NetBeans αποτελεί το πλέον ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού που προσφέρει μοναδική και γρήγορη σχεδίαση των γραφικών περιβαλλόντων χρήστη των εφαρμογών μας και συμβάλει στη ορθότερη ανάπτυξη ποιοτικών προγραμμάτων για τους υπολογιστές και άλλες συσκευές.

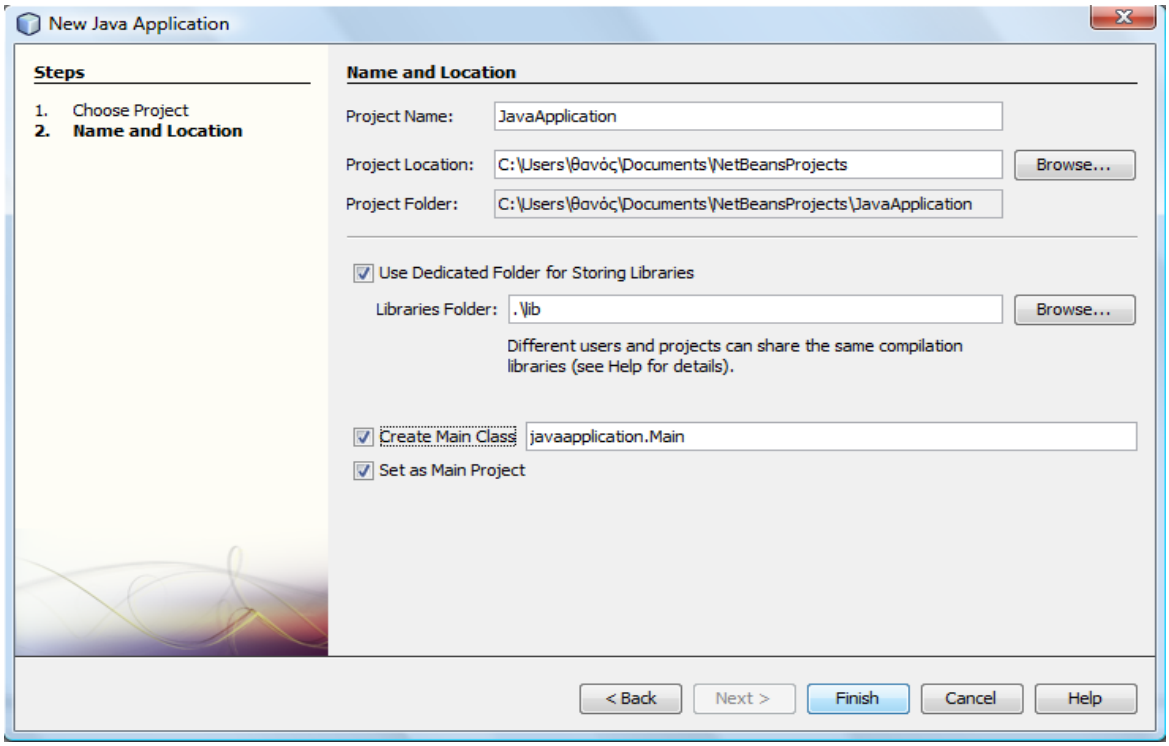
7.2.1. - Δημιουργία Application

Μπορούμε εύκολα να ξεκινήσουμε τη δημιουργία αυτόνομης εφαρμογής java, επιλέγοντας File → New Project, έπειτα επιλέγουμε την κατηγορία java στο αριστερό πλαίσιο και τελικώς επιλέγουμε στο δεξιό πλαίσιο των project, την επιλογή «Java Application».



Σχήμα 7.1 – Δημιουργία Project

Έπειτα επιλέγουμε ένα όνομα για το project μας, την διαδρομή στο τοπικό σύστημα αρχείων που θα αποθηκευτεί το project αυτό, τον κατάλογο με τις εξωτερικές βιβλιοθήκες που ενδέχεται να χρησιμοποιήσει εφαρμογή μας και τέλος την κύρια κλάση της.



Σχήμα 7.2 – Δημιουργία Application

7.3. - Εξωτερικές βιβλιοθήκες ανοιχτού κώδικα (Open Source) Java

Οι εξωτερικές βιβλιοθήκες ανοιχτού κώδικα που χρησιμοποιεί η εφαρμογή μας, ενσωματώνονται στο τελικό εκτελέσιμο αρχείο της (.jar) και έτσι δεν απαιτείται από μεριάς του χρήστη της εφαρμογής να τις εισάγει στον κατάλογο εξωτερικών βιβλιοθηκών του περιβάλλοντος της εικονικής μηχανής της java.

7.3.1. - RXTXcomm.jar

Η βιβλιοθήκη RxTx είναι μια «φυσική υλοποίηση» που υποστηρίζει την σειριακή και παράλληλη επικοινωνία για την εργαλειοθήκη ανάπτυξης λογισμικού της java (JDK). Όλοι οι όροι χρήσης της βρίσκονται κάτω από τους όρους χρήσης της άδειας gnu LGPL. Την εποχή που παρουσιάστηκε η βιβλιοθήκη comm της sun, δημοσιοποιήθηκε και η βιβλιοθήκη RXTXcomm από τον Trent Jarvi. Η βιβλιοθήκη αυτή δουλεύει αποδοτικότερα με τις θύρες επικοινωνίας από ότι αυτή της sun αλλά ιδίως στην περίπτωση μας, κρινόταν αναγκαία η χρήση της, διότι ο υπολογιστής που γράψαμε τον κώδικα δεν διέθετε καμία φυσική σειριακή πόρτα, παρά μόνο εικονική (Virtual) πόρτα και έτσι με την βιβλιοθήκη της sun δεν θα μπορούσε να επικοινωνήσει το πρόγραμμα μας μαζί της.

Για την χρήση της, πρέπει να τοποθετήσουμε το αρχείο **rxtxSerial.dll** στους καταλόγους C:\Program Files\Java\jdk1.6.0_xx\jre\bin και C:\Program Files\Java\jre1.6.0_xx\bin. Η διεπαφή προγραμματισμού **RXTXcomm.jar** πρέπει να τοποθετηθεί στους καταλόγους C:\Program Files\Java\jdk1.6.0_xx\jre\lib\ext και C:\Program Files\Java\jre1.6.0_xx\lib\ext, ή στον κατάλογο lib που θα πρέπει να δημιουργηθεί κάτω από τον κατάλογο \lib της εφαρμογή μας.

7.3.2. - jSMSEngine.jar

Η βιβλιοθήκη jSMSEngine αποτελεί μια ανοιχτού κώδικα διεπαφή προγραμματισμού για την αποστολή και λήψη γραπτών μηνυμάτων μέσω οποιουδήποτε GSM μόντεμ. Η βιβλιοθήκη αυτή, ενσωματώνοντας επίσης την βιβλιοθήκη RXTXcomm, παρέχει την δυνατότητα επικοινωνίας με ένα κινητό τηλέφωνο (GSM μόντεμ), οποιασδήποτε κατασκευάστριας εταιρίας, ώστε να υποστηρίξει την δυνατότητα αποστολής και παραλαβής γραπτών μηνυμάτων και λήψης πληροφοριών από την συσκευή. Η βιβλιοθήκη διανέμεται υπό τους όρους της άδειας gnu GPL. Το μοναδικό αρχείο που την αποτελεί είναι το **jSMSEngine.jar** το οποίο ενσωματώνουμε από τον κατάλογο \lib της εφαρμογής μας.

Ιστοσελίδα: http://www.jsmsengine.org

7.3.3. - mail.jar

Η Java και το Ηλεκτρονικό Ταχυδρομείο

Με την Java, η αποστολή και η παραλαβή μηνυμάτων ηλεκτρονικής αλληλογραφίας μέσα από κάποιο πρόγραμμα, είναι εύκολη υπόθεση. Εάν δημιουργήσουμε κάποια μικροεφαρμογή (applet), μπορούμε να χρησιμοποιήσουμε το ίδιο το πρόγραμμα του περιηγητή ιστού (browser) για στείλουμε ένα μήνυμα ηλεκτρονικής αλληλογραφίας. Ειδάλλως μπορούμε να χρησιμοποιήσουμε την βιβλιοθήκη ηλεκτρονικής αλληλογραφίας της Java **mail.jar**, τόσο για την αποστολή, όσο και για την λήψη μηνυμάτων ηλεκτρονικού ταχυδρομείου. Η βιβλιοθήκη JavaMail παρέχει τρεις κατηγορίες κλάσεων, γνωστές και ως: Messages, Transports, και Stores. Ένα Message, αναπαριστά ένα μήνυμα ηλεκτρονικής αλληλογραφίας. Ένα αντικείμενο Transport αναπαριστά την ίδια την διαδικασία της αποστολής ενός Message από την εφαρμογή μας προς το δίκτυο, ή το διαδίκτυο. Το αντικείμενο Store αναπαριστά τα αποθηκευμένα ηλεκτρονικά

μηνύματα και πρέπει να χρησιμοποιηθεί για να τα χειριστούμε ως αντικείμενα Message. Άρα η κλάση Transport είναι για την αποστολή μηνυμάτων ηλεκτρονικού ταχυδρομείου και η κλάση Store είναι για την ανάγνωσή τους.

Μια άλλη κλάση τώρα, αυτή της συνεδρίας, Session, χρησιμοποιείται για να παράσχει μια πιο εξειδικευμένη προδιαγραφή χρήσης των αντικειμένων Store και Transport.

Το πακέτο JavaMail πλέον συμπεριλαμβάνεται στην έκδοση Java 2 Enterprise Edition (J2EE) και έτσι δεν απαιτείται να το κατεβάσουμε και να το ενσωματώσουμε στον κατάλογο της java.

7.3.4. - jfreeChart 1.0.13

Η JFreeChart είναι μια ελεύθερη, ανοιχτού κώδικα βιβλιοθήκη για την πλατφόρμα της Java(tm). Σχεδιάστηκε ώστε να μπορεί εύκολα να χρησιμοποιηθεί σε εφαρμογές, σε μικροεφαρμογές, στις μικροϋπηρεσίες και στις σελίδες JSP. Η βιβλιοθήκη JFreeChart διανέμεται με όλο τον πηγαίο κώδικα της, κάτω από τους όρους χρήσης της άδειας GNU Lesser General Public Licence.

Χαρακτηριστικά

Με την JFreeChart, μπορούμε να δημιουργήσουμε διαγράμματα πίτας, οριζόντια και κάθετα ραβδογράμματα, ιστογράμματα, διαγράμματα διασποράς, στατιστικά γραφήματα, τρισδιάστατα και μη, χρονοδιαγράμματα, διαγράμματα Gantt, διαγράμματα μετρήσεων και μέσων όρων (θερμόμετρα και κοντέρ) και πολλά άλλα.

Επίσης η βιβλιοθήκη αυτή, ενσωματώνοντας και χρησιμοποιώντας και άλλες ανοιχτού κώδικα βιβλιοθήκες, μας παρέχει τις εξής δυνατότητες:

- Δυνατότητα αποθήκευσης των Διαγραμμάτων σε μορφή PNG και JPEG
- Δυνατότητα αποθήκευσης σε μορφή:
  - PDF χρησιμοποιώντας την βιβλιοθήκη iText (http://www.lowagie.com/iText/)
  - SVG χρησιμοποιώντας την βιβλιοθήκη Batik (http://xml.apache.org/batik/)
- Διαδραστικό zooming με τα διαγράμματα
- Δυναμική δημιουργία HTML image map
- Λειτουργία σε εφαρμογές, μικροεφαρμογές, με μικροϋπηρεσίες και σελίδες JSP

- Διανέμεται με ολόκληρο τον πηγαίο κώδικα υπό τους όρους της άδειας χρήσης GNU Lesser General Public License (LGPL).

Η βιβλιοθήκη JFreeChart είναι εξ' ολοκλήρου γραμμένη σε Java, και δουλεύει σε οποιαδήποτε υλοποίηση της πλατφόρμας Java 2 platform (JDK 1.2.2 και κατοπινές εκδόσεις).

Συνολικά η βιβλιοθήκη περιλαμβάνει τα αρχεία: **jfreechart-1.0.13.jar**, **jfreechart-1.0.13-swt.jar**, **jfreechart-1.0.13-experimental.jar**, **junit.jar**, **jcommon-1.0.16.jar**, **gnujaxp.jar**, **swtgraphics2d.jar,** τα οποία πρέπει να τοποθετηθούν στον κατάλογο των εξωτερικών βιβλιοθηκών της java, κάτω από τον κατάλογο C:\Program Files\Java\jdk1.6.0_xx\jre\lib\ext, ή στον κατάλογο lib της εφαρμογή μας.

7.3.5. - iText-2.1.5.jar

Η βιβλιοθήκη iText είναι μια βιβλιοθήκη που δίνει την δυνατότητα στους δημιουργούς λογισμικού να εμπλουτίσουν τον διακομιστή ιστού, ή άλλες εφαρμογές, να δημιουργούν δυναμικά αρχεία τύπου PDF.

Η βιβλιοθήκη iText μας επιτρέπει να δημιουργούμε αρχεία PDF «εν πτήση». Η βιβλιοθήκη αυτή δεν είναι ένα πρόγραμμα χρήστη που δίνει την δυνατότητα δημιουργίας αρχείων pdf, αλλά μια διεπαφή προγραμματισμού που χρησιμοποιείται για την δυναμική δημιουργία εγγράφων τύπου pdf μέσα από εφαρμογές της java.

Μπορούμε να χρησιμοποιήσουμε την βιβλιοθήκη iText για τους παρακάτω λόγους:

- Για να παρουσιάσουμε ένα αρχείο PDF σε ένα πρόγραμμα περιήγησης ιστού.
- Για να δημιουργήσουμε δυναμικά έγγραφα από αρχεία XML ή από βάσεις δεδομένων
- Για να προσθέσουμε ψηφιακή υπογραφή στα αρχεία μας
- Και πολλά άλλα.

Κοντολογίς: οι κλάσεις της βιβλιοθήκης iText είναι απαραίτητη για εκείνους που θέλουν να δημιουργήσουν δυναμικά, μόνο-ανάγνωσης, έγγραφα που είναι ανεξάρτητα πλατφόρμας και που περιέχουν κείμενο, λίστες, πίνακες και εικόνες, ή που επιθυμούν να εκτελέσουν εξειδικευμένες επεξεργασίες σε υπάρχοντα αρχεία

PDF εγγράφων. Η βιβλιοθήκη είναι πολύ χρήσιμη σε μικροϋπηρεσίες που βασίζονται στην τεχνολογία Java(TM). Η βιβλιοθήκη iText απαιτεί γα την λειτουργία της την έκδοση 1.4 του JDK και διανέμεται δωρεάν υπό τους όρους των αδειών χρήσης: MPL και LGPL.

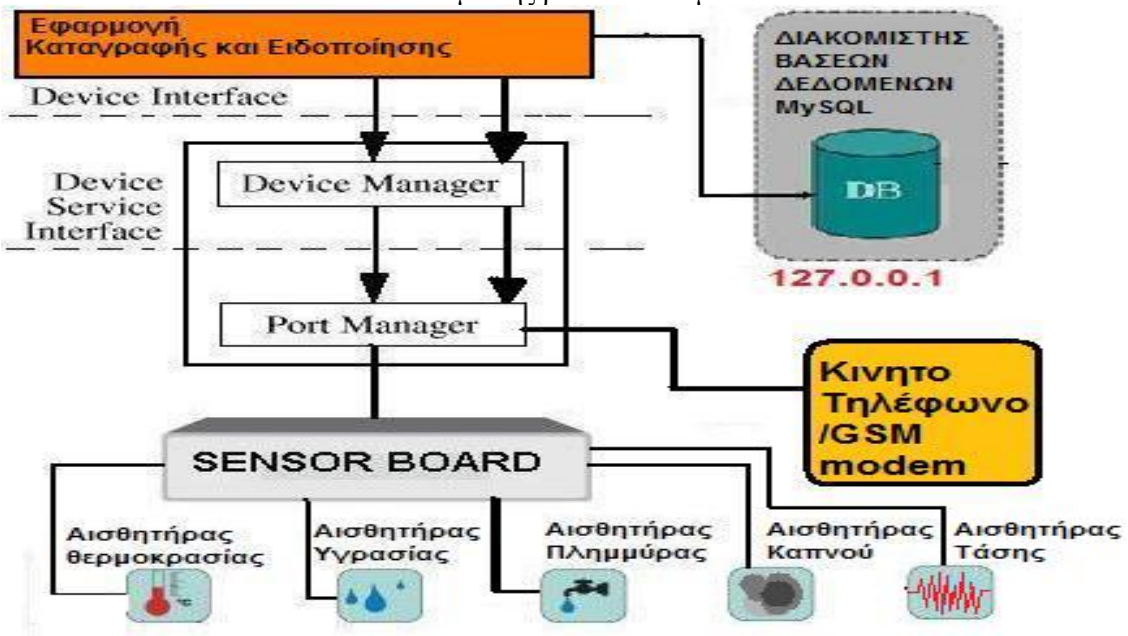
Η βιβλιοθήκη αποτελείται από το αρχείο: **iText-2.1.5.jar** το οποίο θα πρέπει να τοποθετηθεί στον κατάλογο των εξωτερικών βιβλιοθηκών της java, δηλαδή κάτω από τον κατάλογο C:\Program Files\Java\jdk1.6.0_xx\jre\lib\ext, ή στον κατάλογο lib της εφαρμογή μας.

ΚΕΦΑΛΑΙΟ 8
Η υλοποίηση της Εφαρμογής

8.1. - Περίληψη

Πρόκειται για μια εφαρμογή που χρησιμοποιεί διαφορετικά νήματα για τις εργασίες που έχει να επιτελέσει ούτως ώστε να επιταχύνει την λειτουργία της. Η εφαρμογή μας, αποτελείται από δύο πακέτα κλάσεων: το SensingSystem και το Utilities. Το πρώτο περιέχει τη βασική κλάση της εφαρμογής που παράγει το γραφικό περιβάλλον αλληλεπίδρασης και δύο ακόμα υποστηρικτικές κλάσεις που παράγουν μέρος του γραφικού περιβάλλοντος, ενώ το δεύτερο πακέτο, περιέχει όλες εκείνες τις κλάσεις που αφορούν τις διεργασίες που εκτελεί η κύρια εφαρμογή και περιλαμβάνουν τις μεθόδους για τις ενέργειες από και προς την βάση δεδομένων (σύνδεση, εκτέλεση επιλογών και προβολών, ενημερώσεων και τερματισμού της σύνδεσης), τις μεθόδους για τις σειριακές συνδέσεις με τις δύο συσκευές της εφαρμογής, τις μεθόδους για την αποστολή γραπτού μηνύματος στο κινητό του διαχειριστή όταν ξεπεραστούν τα όρια που έχει θέσει για κάθε έναν από τους αναλογικούς αισθητήρες, αλλά και τις μεθόδους για την δημιουργία αρχείων pdf με διαγράμματα των ημερήσιων, εβδομαδιαίων, μηνιαίων και ετήσιων στατιστικών των μετρήσεων και έπειτα την αποστολή τους ως συνημμένα αρχεία στο ηλεκτρονικό ταχυδρομείο του διαχειριστή. Στο *Σχήμα 8.1* παρουσιάζεται η εφαρμογή και τα τρία εξωτερικά μέρη με τα οποία συνεργάζεται και επικοινωνεί.

Η εξωτερική συσκευή με τους αισθητήρες
Το κινητό τηλέφωνο (GSM modem)
Ο διακομιστής βάσεων δεδομένων



Σχήμα 8.1 – Η εφαρμογή καταγραφής και ειδοποίησης

Αναλυτικά οι κλάσεις της εφαρμογής, είναι οι παρακάτω:

- Η **Recorder** η οποία αποτελεί και την κεντρική κλάση της εφαρμογής μας. Η κλάση αυτή είναι υπεύθυνη για τη δημιουργία του γραφικού περιβάλλοντος χρήστη (GUI) της εφαρμογής αλλά και την υποστήριξη όλων των λειτουργιών της. Η κλάση αυτή, ως κεντρικό νήμα της εφαρμογής μας, είναι υπεύθυνη και για την εκκίνηση των υπολοίπων νημάτων της: το νήμα σύνδεσης με τη συσκευή των αισθητήρων για τη παραλαβή των δεδομένων από αυτή, το νήμα σύνδεσης με το μόντεμ (κινητό τηλέφωνο), το νήμα που είναι υπεύθυνο για την άμεση ενημέρωση του διαχειριστή με γραπτό μήνυμα στο κινητό του και με μήνυμα στο ηλεκτρονικό ταχυδρομείο του (εάν ξεπεραστούν τα όρια που έχει θέσει για κάθε αισθητήρα), το νήμα ημερήσιας αναφοράς που αποστέλλει στο διαχειριστή μήνυμα ηλεκτρονικού ταχυδρομείου με τα διαγράμματα των μετρήσεων όλων των αισθητήρων. Η κλάση αυτή διαθέτει μεθόδους για τον διαχωρισμό της πληροφορίας που παραλήφθηκε με τις μετρήσεις των αισθητήρων (ως ένα πακέτο των 19 bytes), μεθόδους για την απόσπαση συγκεκριμένων bits στα bytes του πακέτου που παραλήφθηκε, αλλά και επίσης μέθοδο για το διαχωρισμό της πληροφορίας που αφορά τους ψηφιακούς αισθητήρες.

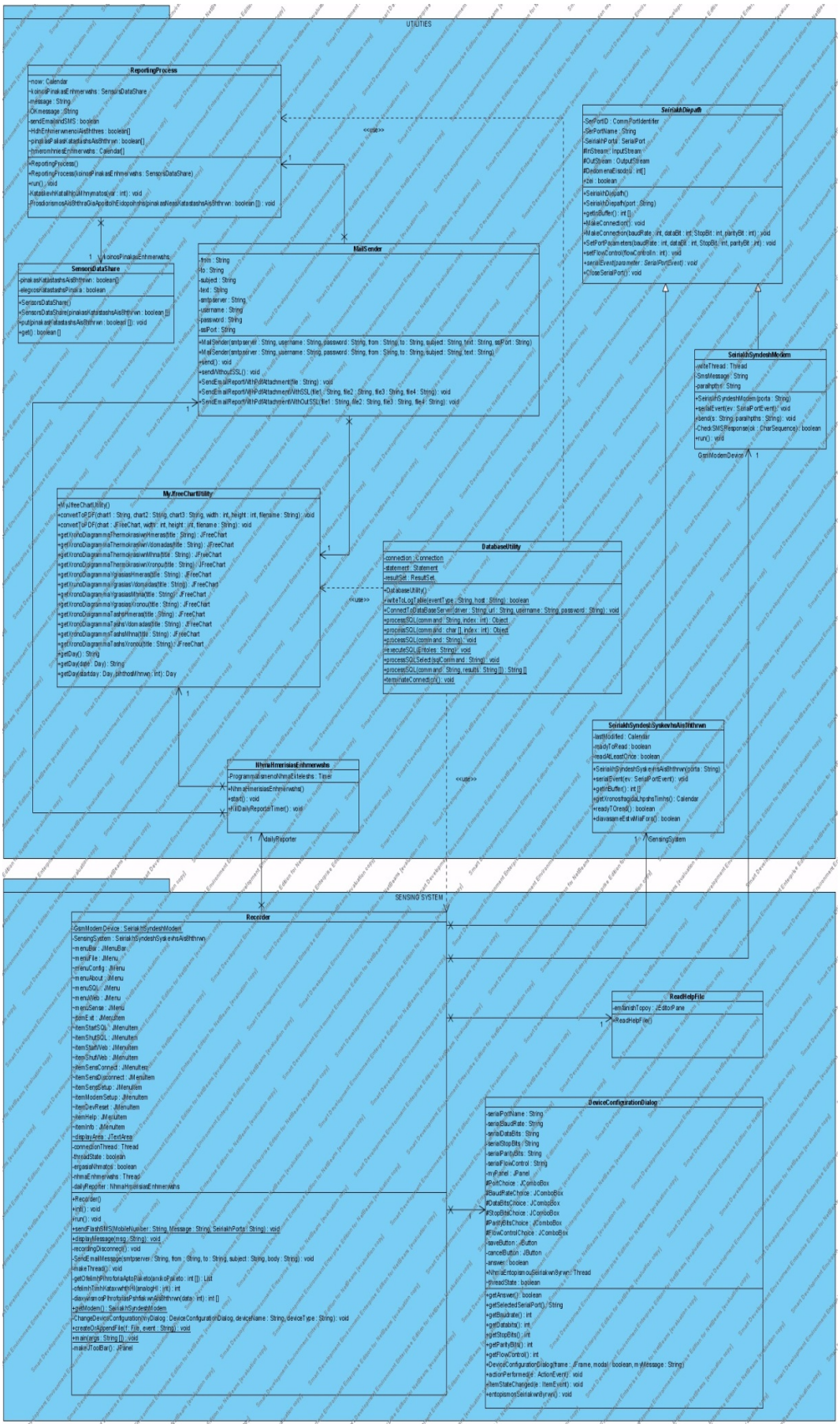
- Η **DeviceConfigurationDialog** η οποία χρησιμοποιείται για να παράγει το πλαίσιο επιλογής και αποθήκευσης των παραμέτρων επικοινωνίας για κάθε μια από τις δύο σειριακές θύρες στις οποίες συνδέεται η εφαρμογή.
- Η **SeiriakhDiepafh** η οποία δίνει μια σαφή περιγραφή για τη σύνδεση και αποσύνδεση με κάποια σειριακή θύρα και μια αφηρημένη περιγραφή (ανυλοποίητη) της μεθόδου της ασύγχρονης ανάγνωσης δεδομένων από την σειριακή θύρα επικοινωνίας. Η κλάση αυτή κληρονομείται από τις κλάσεις των δυο συσκευών, οι οποίες και υλοποιούν την μέθοδο της ασύγχρονης σειριακής ανάγνωσης δεδομένων.
- Η **SeiriakhSyndeshSyskevhsAis8hthrwn** η οποία εξειδικεύει την επικοινωνία με τη σειριακή θύρα που είναι συνδεδεμένη η εξωτερική συσκευή με τους αισθητήρες, για να καταστεί δυνατή η παραλαβή και ανάγνωση των μετρήσεων των αισθητήρων από αυτήν. Η κλάση αυτή διαθέτει μεθόδους για την ανάγνωση των δεδομένων που καταφθάνουν στη σειριακή πόρτα, αλλά και επίσης μέθοδο προσδιορισμού της χρονικής στιγμής λήψης των τιμών από τους αισθητήρες, ώστε να πραγματοποιηθούν κάποιοι έλεγχοι αργότερα και μέθοδο προσπέλασης του χώρου μνήμης που χρησιμοποιείται για την προσωρινή αποθήκευση των δεδομένων των μετρήσεων.
- Η **SeiriakhSyndeshModem** η οποία εξειδικεύει την επικοινωνία με την σειριακή θύρα που είναι συνδεδεμένο το GSM μόντεμ (κινητό τηλέφωνο) για να καταστεί δυνατή η αποστολή γραπτού μηνύματος από την εφαρμογή στο κινητό τηλέφωνο του διαχειριστή. Η κλάση αυτή, διαθέτει μεθόδους για την αποστολή των AT εντολών προς το GSM μόντεμ, την σειριακή ανάγνωση δεδομένων από αυτό και μέθοδο για την πραγματοποίηση της αποστολής του μηνύματος.
- Η **ReportingProcess** η οποία είναι υπεύθυνη για την ειδοποίηση του διαχειριστή με μήνυμα ηλεκτρονικής αλληλογραφίας αλλά και γραπτό μήνυμα στο κινητό του τηλέφωνο για κάθε αισθητήρα που ξεπέρασε το όριο του. Μέσω αυτής της κλάσης, υλοποιούμε ένα ακόμη νήμα της εφαρμογής, το οποίο εκκινείται σχεδόν ταυτόχρονα με το κύριο νήμα της σύνδεσης με τη συσκευή των αισθητήρων και το οποίο διαβάζει τον πίνακα με τα status ειδοποίησης, του κοινόχρηστου αντικειμένου SensorsDataShare, για κάθε

έναν αισθητήρα και αποφασίζει για το εάν πρέπει να υπάρξει ειδοποίηση του χρήστη για κάποιον από τους έξι ξεχωριστά, ή ακόμα και για όλους ταυτόχρονα, οπότε και πραγματοποιεί την αποστολή του μηνύματος ηλεκτρονικού ταχυδρομείου, αλλά και του γραπτού μηνύματος στο κινητό τηλέφωνο του διαχειριστή.

- Η **SensorsDataShare** η οποία αποτελεί ένα κοινόχρηστο αντικείμενο που περιέχει έναν πίνακα με τα status αποστολής ειδοποίησης για κάθε αισθητήρα και χρησιμοποιείται ως όρισμα στην δημιουργία του νήματος της ReportingProcess. Αυτή η κλάση διαθέτει δυο μεθόδους synchronized μια για την εισαγωγή του πίνακα με τις νέες καταστάσεις ειδοποίησης (status), αλλά και μια ακόμη για την επιστροφή του. Το ότι είναι δηλωμένες ως συγχρονισμένες αυτές οι μέθοδοι, υποδηλώνει ότι δεν μπορούν δυο νήματα να μπουν ταυτόχρονα και να εκτελέσουν αυτό τον κώδικα που περιέχουν. Έτσι εξασφαλίζουμε στην ασύγχρονη επεξεργασία τη σειριακή εκτέλεση των λειτουργιών
- Η **NhmaHmerisiasEnhmerwshs** η οποία είναι υπεύθυνη για την ημερήσια αναφορά για την δημιουργία τεσσάρων αρχείων pdf με τα στατιστικά των μετρήσεων των αισθητήρων για την τρέχουσα ημέρα, εβδομάδα, μήνα και έτος και την αποστολή τους ως συνημμένα στον λογαριασμό ηλεκτρονικού ταχυδρομείου του διαχειριστή. Μέσω αυτής της κλάσης, υλοποιούμε ένα ακόμη νήμα της εφαρμογής μας το οποίο όμως πραγματοποιεί την εργασία του όχι κατά την σύνδεση με την συσκευή των αισθητήρων, αλλά μετά από τόσο χρόνο (σε χιλιοστά του δευτερολέπτου), όσο αυτός που απέχει κάθε φορά από τα μεσάνυχτα. Μετά από την πάροδο αυτού του χρονικού διαστήματος, δηλαδή κατά την αλλαγή της ημέρας, στα μεσάνυχτα, το νήμα πραγματοποιεί τις εργασίες του και έπειτα τις επαναλαμβάνει κάθε 24 ώρες, εάν φυσικά η εφαρμογή συνεχίσει τη λειτουργία καταγραφής.
- Η **MailSender** η οποία χρησιμοποιείται για την σύνδεση με τον λογαριασμό ηλεκτρονικού ταχυδρομείου της εφαρμογής, είτε μέσω SSL, είτε όχι και την αποστολή του μηνύματος στον λογαριασμό ηλεκτρονικού ταχυδρομείου του διαχειριστή.

- Η **DatabaseUtility** η οποία χρησιμοποιείται για την σύνδεση με την βάση δεδομένων, την εκτέλεση εισαγωγών, διαγραφών, την επιλογή και την επιστροφή αποτελεσμάτων και στις δύο εφαρμογές μας.
- **Η MyJfreeChartUtility** η οποία χρησιμοποιείται για να κατασκευάσουμε τα διαγράμματα της ημερήσιας ενημέρωσης αλλά και τα αρχεία pdf για αυτά τα διαγράμματα ώστε να αποσταλούν ως συνημμένα αρχεία στο ηλεκτρονικό ταχυδρομείο του διαχειριστή.

Ακολουθεί το διάγραμμα κλάσεων της εφαρμογής στη γλώσσα μοντελοποίησης UML *(Σχήμα 8.2)* και στη συνέχεια παρατίθεται ο κώδικας της υλοποίησης της.



Σχήμα 8.2 – Διάγραμμα κλάσεων

8.2. - Έναρξη της βάσης δεδομένων MySQL

Ο χρήστης επέλεξε να εκκινήσει το Διακομιστή Βάσεων Δεδομένων MySQL6

```
itemShutSQL.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
                try{
                        Runtime rt = Runtime.getRuntime();


                        Process proc = rt.exec("taskkill -f -im mysqld.exe");



                        displayMessage("MySql DataBase Server ShutDown");
                }
                catch(IOException sfalma1){
                        JOptionPane.showMessageDialog(null,
                                        "CANNOT SHUTDOWN MYSQL DEAMON");
                }
                catch(Exception sfalma2){
                        JOptionPane.showMessageDialog(null,
                                        "CANNOT SHUTDOWN MYSQL DEAMON");
                }
        }
});
```

8.3. - Έναρξη του Διαχειριστή Ιστού Apache-Tomcat

Ο χρήστης επέλεξε να εκκινήσει τον διακομιστή εφαρμογών ιστού Tomcat

```
itemStartSQL.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            try{
               Runtime rt = Runtime.getRuntime();


               Process proc = rt.exec("C:\\mysql-noinstall-6.0.10-alpha-win32
```

```
                                \\mysql-6.0.10-alpha-win32\\bin\\mysqld.exe");


                displayMessage("MySql DataBase Server Has Started");
                }
                catch(IOException sfalma1){
                        JOptionPane.showMessageDialog(null,
                                "CANNOT STARTUP MYSQL DEAMON");
                }
                catch(Exception sfalma2){
                        JOptionPane.showMessageDialog(null,
                                "CANNOT STARTUP MYSQL DEAMON");
                }
        }
});
```

8.4. - Ρύθμιση παραμέτρων σειριακής θύρας για την επικοινωνία με την εξωτερική συσκευή με τους αισθητήρες

Ο χρήστης επέλεξε να ρυθμίσει τις παραμέτρους της σειριακής πόρτας για τη σύνδεση με την εξωτερική συσκευή με τους αισθητήρες

```
itemSensSetup.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent e){
        DeviceConfigurationDialog myDialog = new
        DeviceConfigurationDialog(null, true,
                "SETUP SENSING DEVICE COMMUNICATION PARAMETERS");
```

//Προβάλλουμε το πλαίσιο επιλογής και αποθήκευσης των παραμέτρων

```
if(myDialog.getAnswer() ){ //ο χρήστης επέλεξε 'save'
        ChangeDeviceConfiguration(myDialog,"proteus","sensing board");
```

Καλούμε την μέθοδο αλλαγής των παραμέτρων επικοινωνίας με όρισμα το όνομα και τον τύπο της συσκευής που αφορούν οι παράμετροι αυτοί.

```
}
```

```
else{} //ο χρήστης επέλεξε cancel οπότε δεν γίνεται τίποτε απολύτως
}
});
```

Η μέθοδος αλλαγής των παραμέτρων επικοινωνίας:

```
private void ChangeDeviceConfiguration(DeviceConfigurationDialog
                                        myDialog,String deviceName,
                                        String deviceType){
String ok = null;
//ΣΥΝΔΕΣΗ ΜΕ ΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ
DatabaseUtility.ConnectToDataBaseServer("com.mysql.jdbc.Driver","jdbc:mysql://l
ocalhost:3306/sensingsystem","root","10022910");
ok = (String) DatabaseUtility.processSQL(
                                    "SELECT interface FROM devices
                                    where devicetype like '"+deviceType+"' ; ",1);
if(ok==null){ //δεν υπάρχει συσκευή αισθητήρων,οπότε την εισάγουμε
ok = (String)DatabaseUtility.processSQL(
                                        "select interfacename
                                        from serialportparameters
                                        where interfacename like
                                    '"+myDialog.getSelectedSerialPort()+"';",1);
```

Έλεγχος εάν υπάρχουν ρυθμίσεις για αυτή τη σειριακή πόρτα στον πίνακα που αφορά στις παραμέτρους επικοινωνίας και με τις δυο συσκευές (serialportparameters)

```
if(ok==null){
        //δεν υπάρχει ούτε συσκευή, ούτε κάποιο interface, οπότε τα
εισάγουμε
DatabaseUtility.processSQL("insert into
                                    serialportparameters(interfacename,baudrate,da
                                    tabits,stopbits,paritybits,flowcontrol)"
                                    +"values('"+myDialog.getSelectedSerialPort()+"',
                                    "+myDialog.getBaudrate()+","+myDialog.getData
                                    bits()+","+myDialog.getStopBits()+",
```

```
                                    "+myDialog.getParityBits()+","+
                                    myDialog.getFlowControl() +");");
DatabaseUtility.processSQL("insertinto
                                    devices(devicename,interface,devicetype)"+
                                    "values('"+deviceName+"','"+
                                    myDialog.getSelectedSerialPort()+ "',
                                    '"+deviceType+"');");
}
else{
```

δεν υπάρχει κάποια συσκευή αισθητήρων αλλά υπάρχει κάποιο interface

```
String deviceInterface = (String) DatabaseUtility.processSQL("
                                                SELECT devicename
                                                FROM devices
                                                where interface like '"+
                                                myDialog.getSelectedSerial
                                                Port()+"' ; ",1);
if(deviceInterface==null){
```

Έλεγχος για το εάν υπάρχει ήδη κάποια συσκευή που χρησιμοποιεί αυτή τη σειριακή πόρτα επικοινωνίας. Δεν υπάρχει συσκευή οπότε αλλάζουμε τις ρυθμίσεις και τις συσχετίζουμε με τη συσκευή

```
DatabaseUtility.processSQL("UPDATE serialportparameters
                                    SET baudrate="+myDialog.getBaudrate()+",
                                    databits="+myDialog.getDatabits()+
                                    ",stopbits="+myDialog.getParityBits()+
                                    ",paritybits="+myDialog.getParityBits()+
                                    "flowcontrol="+myDialog.getFlowControl()+
                                    " WHERE interfacename LIKE '"+
                                    myDialog.getSelectedSerialPort()+"'; ");
                                    DatabaseUtility.processSQL("insert into
                                    devices(devicename,interface,devicetype)" +
                                    " values('"+deviceName+"','"
                                    +myDialog.getSelectedSerialPort() +
                                    "','"+deviceType+"');");
```

```
}
else {
```

Υπάρχει ήδη συσκευή που χρησιμοποιεί την δοθείσα σειριακή θύρα. Οπότε ο διαχειριστής ενημερώνεται ώστε να επιλέξει κάποια άλλη σειριακή θύρα

```
JOptionPane.showMessageDialog(null,myDialog.getSelectedSerialPort()+
"IS IN USE BY ANOTHER DEVICE.PLEASE USE ANOTHER SERIAL
PORT OR FLUSH DEVICES CONFIGURATION AND TRY AGAIN!",
"USER ERROR",JOptionPane.ERROR_MESSAGE);
}
}
}
else {
```

Υπάρχει συσκευή αποθηκευμένη στον πίνακα devices οπότε λόγω εξάρτησης των πινάκων, υπάρχει και interface, οπότε ενημερώνουμε τον πίνακα των παραμέτρων επικοινωνίας και αυτόματα ενημερώνεται και το interface στον πίνακα devices

```
String deviceInterface = (String) DatabaseUtility.processSQL("
SELECT devicename FROM devices where interface like
'"+myDialog.getSelectedSerialPort()+"' ; ",1);
if(deviceInterface==null){
```

Ελέγχουμε εάν η επιλεγμένη σειριακή πόρτα υπάρχει ήδη και αντιστοιχεί σε κάποια συσκευή, εάν δεν υπάρχει, κάνουμε ενημέρωση

```
DatabaseUtility.processSQL("UPDATE serialportparameters SET
                        interfacename='"+myDialog.getSelectedSerialPort()+"',
                        baudrate="+myDialog.getBaudrate()+",
                        databits="+myDialog.getDatabits()+
                        ",stopbits="+myDialog.getParityBits()+
                        ",paritybits="+myDialog.getParityBits()+
                        "flowcontrol="+myDialog.getFlowControl()+
                        " WHERE interfacename LIKE '"+ok+"' ");
}
else{  //Διαφορετικά,επειδή η επιλεγμένη πόρτα χρησιμοποείται ήδη
```

```
if(ok.equals(myDialog.getSelectedSerialPort()) ){
```

ελέγχουμε εάν η επιλεγμένη πόρτα χρησιμοποιείται ήδη από τη συσκευή που τροποποιούμε, οπότε και κάνουμε ενημέρωση

```
DatabaseUtility.processSQL("UPDATE serialportparameters SET
                    interfacename='"+myDialog.getSelectedSerialPort()+"',
                    baudrate="+myDialog.getBaudrate()+",
                    databits="+myDialog.getDatabits()+
                    ",stopbits="+myDialog.getParityBits()+
                    ",paritybits="+myDialog.getParityBits()+
                    "flowcontrol="+myDialog.getFlowControl()+"
                    WHERE interfacename LIKE ' "+ok+" ' ");
}
else{
```

Διαφορετικά,η πόρτα που επιλέχθηκε χρησιμοποιείται από άλλη συσκευή και ενημερώνουμε τον χρήστη προκειμένου να κάνει εκκαθάριση (flush) των παραμετροποιήσεων

```
JOptionPane.showMessageDialog(null,myDialog.getSelectedSerialPort()+" IS IN
USE BY
ANOTHER DEVICE.PLEASE USE ANOTHER SERIAL PORT OR
FLUSH DEVICES CONFIGURATION AND TRY AGAIN!",
"USER ERROR",JOptionPane.ERROR_MESSAGE);
}
}
}
DatabaseUtility.terminateConnection();
```

//τερματισμός σύνδεσης με βάση δεδομένων

```
}
```

8.5. - Ρύθμιση παραμέτρων σειριακής θύρας για την επικοινωνία με το κινητό τηλέφωνο (GSM MODEM)

Ο χρήστης επέλεξε να παραμετροποιήσει την επικοινωνία με το GSM modem

```
itemModemSetup.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
                DeviceConfigurationDialog myDialog = new
                                                DeviceConfigurationDialog(
                                                null, true,"SETUP SENSING
                                                DEVICE COMMUNICATION
                                                PARAMETERS");
                if(myDialog.getAnswer() ){
                //o χρήστης πάτησε save στο πλαίσιο διαλόγου
```

Καλούμε την μέθοδο αλλαγής των παραμέτρων επικοινωνίας με όρισμα το όνομα και τον τύπο της συσκευής που αφορούν οι παράμετροι αυτοί.

```
                ChangeDeviceConfiguration(myDialog,"N6630","modem");
                }
                else{ }
        }
});
```

8.6. - Πραγματοποίηση σύνδεσης με την επιλεγμένη σειριακή θύρα για την παραλαβή δεδομένων από την συσκευή ελέγχου των αισθητήρων

```
DatabaseUtility.ConnectToDataBaseServer("com.mysql.jdbc.Driver",
                                                "jdbc:mysql://localhost:
                                                3306/sensingsystem",
                                                "root","10022910");
                        try {
                                String[] proteusParams = new String[6];
```

Ανάγνωση αποθηκευμένων παραμέτρων επικοινωνίας για τη συσκεύη των αισθητήρων

```
DatabaseUtility.processSQL("
SELECT * FROM serialportparameters WHERE interfacename
like (SELECT interface FROM devices WHERE devicetype like
'sensing board');",proteusParams);
if(proteusParams[0]==null){
```

Εάν δεν υπάρχει αποθηκευμένη κάποια διαμόρφωση για την επικοινωνία με τη συσκευή προτρέπουμε τον χρήστη να εισάγει κάποια παραμετροποίηση.

```
JOptionPane.showMessageDialog(null,
"YOU MUST SELECT DEVICES CONFIGURATION FIRST",
"USER ERROR",JOptionPane.ERROR_MESSAGE);
DatabaseUtility.terminateConnection();
recordingDisconnect();
}
else {
```

Υπάρχουν αποθηκευμένες παράμετροι επικοινωνίας και τις χρησιμοποιούμε για να συνδεθούμε με τη συσκευή των αισθητήρων

```
try{
```

Πραγματοποίηση της Σύνδεσης στην αποθηκευμένη σειριακή πόρτα

```
SensingSystem = new SeiriakhSyndeshSyskevhsAis8hthrwn( proteusParams[0] );
```

Παραμετροποίηση σειριακής επικοινωνίας με βάση τις αποθηκευμένες παραμέτρους επικοινωνίας

```
SensingSystem.MakeConnection(Integer.parseInt(proteusParams[1]),
Integer.parseInt(proteusParams[2]),
Integer.parseInt(proteusParams[3]),
Integer.parseInt(proteusParams[4]) );
SensingSystem.setFlowControl(Integer.parseInt(proteusParams[5]) );
}
catch(Exception sfalmasyndeshs){
JOptionPane.showMessageDialog(null,
"AN ERROR OCCURED ON ATTEMPT TO CONNECT WITH SENSORS
SYSTEM!!",
"CRITICAL ERROR",JOptionPane.ERROR_MESSAGE);
displayMessage("AN ERROR OCCURED ON ATTEMPT TO CONNECT
WITH SENSORS SYSTEM!!");
```

Τερματίζουμε τη σύνδεση με τη βάση δεδομένων

```
DatabaseUtility.terminateConnection();
```

Τερματίζουμε τα νήματα που έχουν εκκινήσει

```
recordingDisconnect();
return;
}
```

8.7. - Πραγματοποίηση σύνδεσης με την επιλεγμένη σειριακή θύρα για την αποστολή μηνύματος sms μέσω του GSM Modem στο κινητό του διαχειριστή

Για την διαδικασία αποστολής γραπτού μηνύματος στο κινητό τηλέφωνο παρέχουμε δύο υλοποιήσεις: μία εξ' ολοκλήρου δική μας βασισμένη μόνο στη σειριακή διασύνδεση και τις AT εντολές η οποία όμως υπολείπεται της δεύτερης διότι υποστηρίζει μόνο την αποστολή SMS σε Text-mode. Η δεύτερη βασίζεται στην ολοκληρωμένη βιβλιοθήκη jSMSEngine και υποστηρίζει όλες τις δυνατότητες αποστολής SMS.

8.7.1. - Υλοποίηση για αποστολή SMS σε Text-Mode

//ΣΥΝΔΕΣΗ ΜΕ ΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

```
DatabaseUtility.ConnectToDataBaseServer("com.mysql.jdbc.Driver","jdbc:mysql://l
ocalhost:3306/sensingsystem","root","10022910");
```

Ανάγνωση αποθηκευμένων παραμέτρων επικοινωνίας για τη συσκεύη GSM modem

```
DatabaseUtility.processStringSQL("
SELECT * FROM serialportparameters WHERE interfacename
like (SELECT interface FROM devices WHERE devicetype like
'mobile');",mobileParams);
if (mobileParams[0]==null ) {
```

Εάν δεν υπάρχει αποθηκευμένη κάποια διαμόρφωση για την επικοινωνία με το modem προτρέπουμε τον χρήστη να εισάγει κάποια παραμετροποίηση.

```
JOptionPane.showMessageDialog(null,"Select devices configuration first.","Error",
JOptionPane.ERROR_MESSAGE);
DatabaseUtility.terminateConnection();
recordingDisconnect();
}
else {
```

Υπάρχουν αποθηκευμένες παράμετροι επικοινωνίας και τις χρησιμοποιούμε για να συνδεθούμε με τη συσκευή των αισθητήρων

```
try {
```

Πραγματοποίηση της Σύνδεσης στην αποθηκευμένη σειριακή πόρτα

```
GsmModem = new SeiriakiSindesiModem(mobileParams[0]);
```

Παραμετροποίηση σειριακής επικοινωνίας με βάση τις αποθηκευμένες παραμέτρους επικοινωνίας

```
GsmModem.MakeConnection(Integer.parseInt(mobileParams[1]),
Integer.parseInt(mobileParams[2]),
Integer.parseInt(mobileParams[3]),
Integer.parseInt(mobileParams[4]));
GsmModem.setFlowControl(Integer.parseInt(mobileParams[5]));
}catch(Exception e) {
displayMessage("Sensor Data Capturing Process is being terminated!\n
Check device operation and restart manually!");
JOptionPane.showMessageDialog(null,
"Error on attempt to connect to GSM Modem.",
"Error", JOptionPane.ERROR_MESSAGE);
```

Τερματίζουμε τη σύνδεση με τη βάση δεδομένων

```
DatabaseUtility.terminateConnection();
```

Τερματίζουμε τα νήματα που έχουν εκκινήσει

```
recordingDisconnect();
return;
}
```

8.7.2. - Υλοποίηση για αποστολή SMS σε PDU-Mode με χρήση της βιβλιοθήκης jSMSEngine

Μέθοδος σύνδεσης και αποστολής στιγμιαίου μηνύματος στο κινητό τηλέφωνο του χρήστη της εφαρμογης,όταν παραστεί η ανάγκη.

```
public static void sendFlashSMS(String MobileNumber,String Message,String
SeiriakhPorta){
```

```
try {
```

ΣΥΝΔΕΣΗ ΜΕ ΤΗ ΣΥΣΚΕΥΗ ΤΟΥ ΚΙΝΗΤΟΥ ΤΗΛΕΦΩΝΟΥ ΣΤΗ ΑΠΟΘΗΚΕΥΜΕΝΗ ΣΕΙΡΙΑΚΗ ΘΥΡΑ

```
CService srv = new CService(SeiriakhPorta,9600,"SonyEricsson","w810i");
srv.setSimPin("0000");
srv.connect();
srv.setSmscNumber("");
COutgoingMessage msg = new COutgoingMessage(MobileNumber,Message);
msg.setMessageEncoding(CMessage.MESSAGE_ENCODING_7BIT);
msg.setFlashSms(true);
srv.sendMessage(msg);
srv.disconnect();//αποσύνδεση από το GSM MODEM
displayMessage("SMS ALERT MESSAGE HAS BEEN SENT SUCCESFULLY!");
}
catch (Exception e){
displayMessage("USER ERROR => THERE IS NO CONNECTION WITH THE
GSM MODEM!");
displayMessage("SENDING ALERT SMS MESSAGE PROCESS HAS FAILED!" );
try{ //ΕΓΓΡΑΦΗ ΤΟΥ ΓΕΓΟΝΟΤΟΣ ΣΤΟ ΑΡΧΕΙΟ ΣΦΑΛΜΑΤΩΝ
createOrAppendFile(new File("ErrorsLogFile.txt"),"USER ERROR => THERE IS
                    NO CONNECTION WITH THE GSM MODEM!");
createOrAppendFile(new File("ErrorsLogFile.txt"),"SENDING ALERT SMS
                    MESSAGE PROCESS HAS FAILED!");
}
catch(Exception lethos){
displayMessage("Application Hazard => createOrAppendFile() HAS
                    FAILED!".toUpperCase() );
return;
}
return;
}
}
```

8.8. - Παραλαβή δεδομένων μετρήσεων, διαχωρισμός της πληροφορίας, αποθήκευση μετρήσεων στη βάση δεδομένων

Το παρακάτω μπλοκ κώδικα είναι αυτό που πραγματοποιεί την διαδικασία της καταγραφής των δεδομένων που καταφθάνουν από την συδκευή των αισθητήρων που είναι συνδεδεμένη σε κάποια σειριακή θύρα του υπολογιστή

```
while(ergasiaNhmatos){
```

//ελέγχουμε εάν είμαστε έτοιμοι να διαβάσουμε τα δεδομένα (το μπλοκ των 19 byte)

```
if(SensingSystem.readyTOread() ){
```

Η παρακάτω πρόταση θα ισχύει την πρώτη φορά της εκτέλεσης του κώδικα, οπότε θα συνεχίσει με την αποθήκευση των μετρήσεων στη βάση δεδομένων

```
        if(lastDBinsertDate==null){
                shouldContinue = true;
```

Παίρνουμε τη χρονική στιγμή που λάβαμε τιμή

```
        newDBinsertDate = SensingSystem.getXronosfragidaLhpshsTimhs();
}
else {
        newDBinsertDate = SensingSystem.getXronosfragidaLhpshsTimhs();
```

συγκρίνουμε εάν η τωρινή χρονική στιγμή, είναι μεταγενέστερη της τελευταίας χρονικής στιγμής που λάβαμε δεδομένα

```
if( newDBinsertDate.after(lastDBinsertDate) ){
```

εάν η ημερομηνία είναι πιο πρόσφατη, τότε συνέχισε την διαδικασία καταγραφής

```
shouldContinue = true;
}
else{
        shouldContinue = false;
  }
```

}
if(shouldContinue){

εάν τώρα πρέπει να συνεχίσει διαβάζουμε τα δεδομένα από τον προσωρινό χώρο αποθήκευσής τους

sensorDataPacket = SensingSystem.getInBuffer();

Αναμενώμενη Μορφή Πακέτου:
('S','T','A','R','T', (AN1), (AN2), (AN3), (AN4), (DIG1, DIG2), 'E', 'N', 'D', '\r', '\n')
Κάθε τελική τιμή για κάθε αναλογικό αισθητήρα, αποτελείται από ένα ζεύγος τιμών που προκύπτει από τους καταχωρητές LO & HI.
Οι τιμές των δυο ψηφιακών αισθητήρων αποθηκεύονται μόνο σε ένα byte που αποτελείται από 8 bits απο τα 8 bit εμάς μας ενδιαφέρουν μόνο τα 2, τα λιγότερο σημαντικά bits (LSB), που βρισκονται στην 1η και τη 2η θέση. Το πρώτο bit αφορά στον αισθητήρα καπνού, ενώ το δεύτερο στον αισθητήρα πλημμύρας.
Από το πακέτο των 19 bytes εμείς χρειαζόμαστε μόνο τα 9 από αυτά, γιατί τα υπόλοιπα χρησιμοποιούνται για την αποστολή χαρακτήρων για τον έλεγχο και την διασφάλιση της ακεραιότητας της πληροφορίας. Η παρακάτω μέθοδος αποσπά την ωφέλιμη πληροφορία, δηλαδή αυτή που αφορά στις μετρήσεις των αισθητήρων

List listaOfelimhsPlhroforias = getOfelimhPlhroforiaAptoPaketo(sensorDataPacket);

Η ωφέλιμη πληροφορία, βρίσκεται ανάμεσα στο 'S,T,A,R,T' και στο 'E,N,D' και είναι τα 9 BYTES από τα 19 που περιέχει το πακέτο. κάθε αποτέλεσμα για τους αναλογικούς, δημιουργείται ως εξής:
(OFELIMO{ HI } * 256 +LO)* 4 MILLIVOLTS /10 = Δεκαδική Τιμή Μέτρησης

if(listaOfelimhsPlhroforias.size() ==9){

Έλεγχος παραλαβής δεδομένων για όλους τους αισθητήρες, δηλαδη 2*4 αριθμούς για τους αναλογικούς αισθητήρες και 1 αριθμό για τους δυο ψηφιακούς αισθητήρες.

for(int i=0;i<4;i++) {
int HI =

```
ofelimhTimhKataxwrhthHI((Integer)listaOfelimhsPlhroforias.get( 2 * i + 1) );
int LO = (Integer) listaOfelimhsPlhroforias.get(2*i) ;
AnalogSensors[i]=((HI*256.0) + LO)*4.0/10.0;
```

Πολλαπλασιάζουμε την τιμή του καταχωρητή HI με 256 και προσθέτουμε την τιμή του LO έπειτα πολλαπλασιάζουμε με 4 γιατι θέλουμε να αναπαριστήσουμε τιμές μέχρι 4096, όσο είναι και το εύρος των milliVolts μέσα στο οποίο μας στέλνονται μετρήσεις από τους αισθητήρες και τέλος διαιρούμε με 10 για να βγάλουμε το επιθυμητό νούμερο-μέτρηση.

```
if(!(AnalogSensors[i]>0.0 && AnalogSensors[i]<151.0) )
```

Ελέγχουμε εάν οι τιμές είναι άκυρες,ώστε να μην αποθηκευτούν στη βάση δεδομένων. Εάν οι τιμή είναι κάτω από 0.0, είτε πάνω από 150.0, τότε τοποθετούνται μηδενικά στην θέση τους, ώστε να αποθηκευτούν και στην αναπαράσταση τους να φανερώσουν στον χρήστη ότι για κάποιο λόγο δεν αποστέλλονται σωστές τιμές από τους αισθητήρες.

```
AnalogSensors[i]=0.0;
}
```

επειδή ο αισθητήρας LM35 μπορεί να αναπαραστήσει τιμές μέχρι και 150.0, τον διπλασιάζουμε για να μπορεί να αναπαραστήσει μέχρι και τα 300 volts.

```
AnalogSensors[3] = AnalogSensors[3]*2;
```

Δίνουμε στη μέθοδο το 8ο byte από το πακέτο των 19 bytes, το οποίο είναι εκείνο που περιλαμβάνει την κατάσταση των 2 ψηφιακών αισθητήρων

```
int[]digitalSensors=
diaxwrismosPlhroforiasPshfiakwnAis8hthrwn((Integer)
listaOfelimhsPlhroforias.get(8) );
for (int i=0;i<4;i++){
Calendar hmerologio = newDBinsertDate;


String hmeromhnia = ""+hmerologio.get(Calendar.YEAR)+"-" +
```

```
(hmerologio.get(Calendar.MONTH)+1)+"-"+



hmerologio.get(Calendar.DAY_OF_MONTH)+" "+
hmerologio.get(Calendar.HOUR_OF_DAY)+ ":" +
hmerologio.get(Calendar.MINUTE)+ ":" +
hmerologio.get(Calendar.SECOND)+"";
String command2 = "insert into
analogmeasurement(analogsensorid,measurementtim
e,captureddata)
values('AN"+(i+1)+"','"+hmeromhnia+"',"+AnalogSenso
rs[i]+");";
DatabaseUtility.processSQL(command2);
}
for (int i=0;i<2;i++) {
Calendar hmerologio = newDBinsertDate;

String hmeromhnia = ""+hmerologio.get(Calendar.YEAR)+"-" +



(hmerologio.get(Calendar.MONTH)+1)+"-"+



hmerologio.get(Calendar.DAY_OF_MONTH)+" "+
hmerologio.get(Calendar.HOUR_OF_DAY)+ ":" +
hmerologio.get(Calendar.MINUTE)+ ":" +
hmerologio.get(Calendar.SECOND)+"";
String command2 =
"insert into
```

```
digitalmeasurement(digitalsensorid,measurementtime,captureddata)
values('DIG"+(i+1)+"','"+hmeromhnia+"',"+digitalSensors[i]+");";
DatabaseUtility.processSQL(command2);
}
```

Αφού ενημερώσουμε τη βάση δεδομένων με τις νέες τιμές των αισθητήρων, κρατάμε τη νέα ημερομηνία παραλαβής των δεδομένων για μελλοντική σύγκριση.

```
lastDBinsertDate = newDBinsertDate ;
```

Μηδενίζουμε τον μετρητή σφαλμάτων παραλαβής δεδομένων ενώ η συσκευή λειτουργούσε κανονικά

```
metrhthsApotyxhmenwnApostolwnDedomenwn = 0;
for(int i=0;i<4;i++){
```

Παίρνουμε από τη βάση δεδομένων το τρέχον όριο που έχει ορίσει ο χρήστης για κάθε έναν από τους αναλογικούς αισθητήρες

```
double limitValue = (Double) DatabaseUtility.processSQL("SELECT limitvalue
FROM
analogsensors WHERE
analogsensorid like
'AN"+(i+1)+"';",1);
if(AnalogSensors[i]>limitValue){
pinakasKatastashsAis8hthrwn[i]=true;
}
else{
pinakasKatastashsAis8hthrwn[i]=false;
}
}
for(int i=0;i<2;i++){
```

Έλεγχος εάν έχουν ενεργοποιηθεί οι ψηφιακοί αισθητήρες

```
if(digitalSensors[i]==1){
pinakasKatastashsAis8hthrwn[i+4]=true;
}
```

```
else{
pinakasKatastashsAis8hthrwn[i+4]=false;
}
}
```

τοποθετούμε τον ενημερωμένο πίνακα στο κοινόχρηστο αντικείμενο μας

```
sdshare.put( pinakasKatastashsAis8hthrwn );
try{
Thread.sleep(4000); //4 δευτερόλεπτα ύπνου για το τρέχον νήμα
}
catch(InterruptedException ee){}
```

} **//τέλος if το πακέτο size==9**

```
else{
```

αυτό το μπλόκ κώδικα εκτελείται εάν το πακέτο που παραλήφθηκε από τη συσκεύη των αισθητήρων δεν έχει έγκυρο μέγεθος(9 bytes).

```
System.out.println( ( j++) + " CORRUPTED DATA PACKET!") );
try{
Thread.sleep(500); //ύπνος μισού χιλιοστού του δευτερολέπτου για το νήμα μας
}
catch(InterruptedException ee){}
}
}
else{
```

εάν οι τιμές δεν είναι καινούργιες, τότε κοιμήσου για ένα δευτερόλεπτο

```
metrhthsApotyxhmenwnApostolwnDedomenwn++;
System.out.println((metrhthsApotyxhmenwnApostolwnDedomenwn+"
T HERE ARE NO NEW DATA!") );
try{
Thread.sleep(1000); //ύπνος ενός δευτερολέπτου για το νήμα μας
}
catch(InterruptedException ee){}
```

Ελέγχουμε εάν έχουν συμβεί 60 αποτυχημένες προσπάθειες ανάγνωσης από την συσκευή

```
if( metrhthsApotyxhmenwnApostolwnDedomenwn>60 ){
displayMessage(

"FATAL ERROR --> IT SEEMS LIKE THE SENSORS DEVICE



COLLAPSED!!");
recordingDisconnect(); //Διακοπή Καταγραφής
String mailAccount[] = new String[5];
```

Παίρνουμε από τη βάση δεδομένων τα απόθηκευμένα στοιχεία για τον παραλήπτη του μηνύματος ειδοποίησης (σε κινητό και ηλεκτρονικό ταχυδρομείο)

```
DatabaseUtility.processSQL("SELECT * FROM mailparameters;",mailAccount);
String seiriakhPorta = DatabaseUtility.processSQL(
"SELECT interface FROM devices WHERE
devicetype like 'modem';",1).toString();
String ArithmosKinhtouThlefwnou = DatabaseUtility.processSQL(
"SELECT mobilenumber FROM
applicationowner;",1).toString();
String paralhpths = DatabaseUtility.processSQL("
SELECT emailaddress FROM
applicationowner;",1).toString();
String subject = "CRITICAL SITUATION!!!!";
Recorder.sendFlashSMS(ArithmosKinhtouThlefwnou,
"FATAL ERROR:IT SEEMS LIKE THE SENSORS DEVICE COLLAPSED!!",
seiriakhPorta);
try{
```

//ΕΓΓΡΑΦΗ ΤΟΥ ΓΕΓΟΝΟΤΟΣ ΣΤΟ ΑΡΧΕΙΟ ΣΦΑΛΜΑΤΩΝ

```
createOrAppendFile(new File("ErrorsLogFile.txt"),"Daily Report
```

```
Process Has Failed!".toUpperCase() );
createOrAppendFile(new File("ErrorsLogFile.txt"),"IT SEEMS LIKE
THE SENSORS DEVICE COLLAPSED!!");
}
catch(Exception sfalma){
displayMessage("Application Hazard : createOrAppendFile() HAS
FAILED!");
return;
}
if(Integer.parseInt(mailAccount[4])==0){
```

Εάν δεν υπάρχει αποθηκευμένη κάποια πόρτα για SSL τη σύνδεση στο λογαριασμό ηλεκτρονικής αλληλογραφίας, τότε είναι 0 και η εφαρμογή θα συνδεθεί χωρίς SSL για την αποστολή του ηλεκτρονικου μηνύματος

```
try{
MailSender sendMail=
New MailSender(mailAccount[1],mailAccount[2],mailAccount[3],
mailAccount[0],paralhpths,subject,"FATAL ERROR:IT SEEMS LIKE THE
SENSORS DEVICE COLLAPSED!!");
sendMail.sendWithoutSSL(); //αποστολή email
Recorder.sendFlashSMS(ArithmosKinhtouThlefwnou,
"FATAL_ERROR:IT SEEMS LIKE THE SENSORS DEVICE COLLAPSED!!",
seiriakhPorta); //αποστολή sms
}
catch(Exception e){ }
}
else{
```

χρησιμοποιείται κάποια SSL πόρτα για τη σύνδεση με τον λογαριασμό ηλεκτρονικήε αλληλογραφίας της εφαρμογής μας.

```
try{
MailSendersendMail=
new MailSender(mailAccount[1],mailAccount[2],mailAccount[3],
mailAccount[0],paralhpths,subject,
```

"FATAL ERROR:IT SEEMS LIKE THE SENSORS DEVICE
COLLAPSED!!",mailAccount[4]);
sendMail.send(); **//αποστολή email**
Recorder.sendFlashSMS(ArithmosKinhtouThlefwnou,
"FATAL ERROR:IT SEEMS LIKE THE
SENSORS DEVICE COLLAPSED!!",
seiriakhPorta); **//αποστολή sms**
}
catch(Exception e){}
}
JOptionPane.showMessageDialog(null,
"IT SEEMS LIKE THE SENSORS
DEVICE COLLAPSED!!","FATAL ERROR",
JOptionPane.ERROR_MESSAGE);
}
}
} **//τέλος του if readyToRead()**
else {

ΕΑΝ ΔΕΝ ΕΧΟΥΜΕ ΠΑΡΑΛΑΒΕΙ ΤΙΜΕΣ ΓΙΑ ΚΑΠΟΙΟ ΛΟΓΟ, ΣΗΜΑΙΝΕΙ ΕΙΤΕ ΟΤΙ ΠΑΡΑΛΑΜΒΑΝΕΙ ΑΥΤΗ ΤΗ ΣΤΙΓΜΗ, ΕΙΤΕ ΟΤΙ ΔΕΝ ΕΧΕΙ ΛΑΒΕΙ ΠΟΤΕ ΕΩΣ ΤΩΡΑ. ΒΑΖΟΥΜΕ ΓΙΑ ΥΠΝΟ ΕΝΟΣ ΔΕΥΤΕΡΟΛΕΠΤΟΥ ΤΟ ΝΗΜΑ ΩΣΤΕ ΝΑ ΞΑΝΑΔΟΚΙΜΑΣΕΙ ΝΑ ΠΑΡΑΛΑΒΕΙ ΔΕΔΟΜΕΝΑ ΑΠΟ ΤΗ ΣΥΣΚΕΥΗ

if(!(SensingSystem.diavasameEstwMiaFora())){

εάν δεν έχουμε διαβάσει ούτε μια φορά, σημαίνει ότι δεν υπάρχει σύνδεδεμένη κάποια συσκευή στην σειριακή πορτα που έχει συνδεθεί το πρόγραμμα, οπότε καταμετράμε τις αποτυχημένες αυτές απόπειρες, ώστε να ειδοποιήθεί ο χρήστης της εφαρμογής.

metrhthsApotyxhmenwnProspa8eiwnEpikoinwnias++;
try{
Thread.sleep(1000);
}
catch(InterruptedException ee){}

```
if(metrhthsApotyxhmenwnProspa8eiwnEpikoinwnias>60){
```

Ελέγχουμε εάν έχουν συμβεί 60 αποτυχημένες προσπάθειες επικοινωνίας

```
displayMessage("FATAL ERROR --> IT SEEMS LIKE THERE IN NONE


SENSORS
DEVICE COONECTED ON THIS SERIAL PORT!!");
recordingDisconnect();
try{ //ΕΓΓΡΑΦΗ ΤΟΥ ΓΕΓΟΝΟΤΟΣ ΣΤΟ ΑΡΧΕΙΟ ΣΦΑΛΜΑΤΩΝ
createOrAppendFile(new File("ErrorsLogFile.txt"),
                        "FATAL ERROR :IT SEEMS LIKE THERE
                        IN NONE SENSORS DEVICE COONECTED
                        ON THIS SERIAL PORT!!");
}
catch(Exception sfalma){
        displayMessage("Application Hazard => createOrAppendFile() HAS
                                FAILED!".toUpperCase() );
return;
}
try{
createOrAppendFile(new File("ErrorsLogFile.txt"),
                        "FATAL ERROR :IT SEEMS LIKE THERE
                        IN NONE SENSORS DEVICE COONECTED
                        ON THIS SERIAL PORT!!");
}
catch(Exception sfalma){
        displayMessage("Application Hazard => createOrAppendFile() HAS
                        FAILED!".toUpperCase() );
        return;
}
JOptionPane.showMessageDialog(null,"IT SEEMS LIKE THERE IN NONE
```

```
                                    SENSORS DEVICE COONECTED ON
                                    THIS SERIAL PORT!!!!",
                                    "FATAL ERROR",
                                    JOptionPane.ERROR_MESSAGE);
}
}
}
} //τελος while
} //τελος else
} //τελος try
catch(Exception sfalma){
                displayMessage("CAPTURING SENSORS DATA PROCESS HAS
                                    FAILED!!");
String mailAccount[] = new String[5];
String seiriakhPorta = null;
String ArithmosKinhtouThlefwnou = null;
String paralhpths = null;
DatabaseUtility.processSQL(
                            "SELECT * FROM mailparameters;",mailAccount);
try{
seiriakhPorta = DatabaseUtility.processSQL("SELECT interface
                                                FROM devices
                                                WHERE devicetype
                                                Like 'modem';",1).toString();
ArithmosKinhtouThlefwnou = DatabaseUtility.processSQL("
                                            SELECT mobilenumber
                                            FROM applicationowner;",1).toString();
paralhpths = DatabaseUtility.processSQL("SELECT emailaddress
                                            FROM applicationowner;",1).toString();
}
catch(Exception eksaireshSQL){}
String subject = "CRITICAL SITUATION!!!!";
if(Integer.parseInt(mailAccount[4])==0){
try{
```

```
MailSender sendMail =
new MailSender(mailAccount[1],mailAccount[2],mailAccount[3],
                mailAccount[0],paralhpths,subject,"FATAL ERROR:IT SEEMS
                LIKE THE SENSORS DEVICE COLLAPSED!!");
```

αποστολή ηλεκτρονικού μηνύματος (email)

```
sendMail.sendWithoutSSL();
```

αποστολή γραπτού μηνύματος μέσω GSM modem (sms)

```
Recorder.sendFlashSMS(ArithmosKinhtouThlefwnou,
                        "FATAL ERROR:IT SEEMS LIKE THE SENSORS DEVICE
                        COLLAPSED!!",seiriakhPorta);
}
catch(Exception e){}
}
else{
try{
MailSender sendMail =
new MailSender(mailAccount[1],mailAccount[2],mailAccount[3],
mailAccount[0],paralhpths,subject,
"FATAL ERROR:IT SEEMS LIKE THE SENSORS DEVICE
COLLAPSED!!",mailAccount[4]);
sendMail.send(); //αποστολή email
Recorder.sendFlashSMS(ArithmosKinhtouThlefwnou,
"FATAL ERROR:IT SEEMS LIKE THE SENSORS DEVICE
COLLAPSED!!",seiriakhPorta);//αποστολή sms
}
catch(Exception e){}
}
DatabaseUtility.terminateConnection();
recordingDisconnect();
try{ //ΕΓΓΡΑΦΗ ΤΟΥ ΓΕΓΟΝΟΤΟΣ ΣΤΟ ΑΡΧΕΙΟ ΣΦΑΛΜΑΤΩΝ
createOrAppendFile(new File("ErrorsLogFile.txt"),
                        "APPLICATION HAZARD :MAIN CAPTURING SENSORS
                        DATA PROCESS HAS FAILED!!");
```

```
}
catch(Exception sfalmaaaaaaa){
        displayMessage("Application Hazard => createOrAppendFile() HAS
                                FAILED!".toUpperCase() );
return;
}
}
```
}/ /τελος μεθόδου RUN

μέθοδος για την απόσπαση της χρήσιμης πληροφορίας από το πακέτο δεδομένων με τις μετρήσεις

```
private List getOfelimhPlhroforiaAptoPaketo(int[] arxikoPaketo){
List ofelimoPaketo = new ArrayList();
int theshStart = 0;
int theshEnd = 0;
        for(int i=0;i<arxikoPaketo.length;i++){
                if(i>=2){

if(arxikoPaketo[i]==84 && arxikoPaketo[i-1]==82 && arxikoPaketo[i-2]==65)
```

ελέγχουμε εάν μας ήρθε έστω το ‘A’, ’R’, ’T’ από το ‘S’, ’T’, ’A’, ’R’, ’T’

```
theshStart = i;
}
}
for(int i=0;i<arxikoPaketo.length;i++){

        if(i<=arxikoPaketo.length-2){

if(arxikoPaketo[i]==69 && arxikoPaketo[i+1]==78 && arxikoPaketo[i+2]==68)
```

ελέγχουμε εάν ήρθε ‘E’, ’N’, ’D’

```
theshEnd = i;
}
}
for(int i=(theshStart+1);i<theshEnd;i++){
        ofelimoPaketo.add( arxikoPaketo[i] );
}
return ofelimoPaketo;
}
```

μέθοδος για να πάρουμε την τιμή των δυο πρώτων bit απο το byte HI για κάθε αισθητήρα

```
private int ofelimhTimhKataxwrhthHI(int analogHI) {
// 00000011 ή 00000000
String HI = Integer.toBinaryString( analogHI ); // Μετατροπή του αριθμού HI σε δυαδική μορφή
int apotelesma = 0;
if(HI.length()==1){


char thesh0 = (char) HI.charAt( HI.length()-1);
```

παίρνουμε τον τελευταίο χαρακτήρα της συμβολοσειράς (1ο LSB BIT από τον HI)

```
if(thesh0 == '0'){
        apotelesma = 0; // 0^1 + 0^0 = 0+0
}
else if(thesh0 == '1'){
        apotelesma = 1; // 0^1 + 2^0 = 0+1
}
}
else{
```

```
char thesh0 = (char) HI.charAt(HI.length()-1);
```

παίρνουμε τους δυο τελευταίους χαρακτήρες από τη συμβολοσειρά (LSB BITS του HI)

```
char thesh1 = (char) HI.charAt(HI.length()-2);// >>
```

```
if(thesh1 == '1' && thesh0 == '1'){
apotelesma = 2; // 2^1 + 0^0 = 2+0
}
else if(thesh1 == '1' && thesh0 == '1'){
apotelesma = 3; // 2^1 + 2^0 = 2+1
}
}
return apotelesma;
}
```

Μέθοδος για να αποσπάσουμε την τιμή του 8ου byte του πακέτου των μετρήσεων αλλά και για να πάρουμε έναν πίνακα με τις τιμές τους (αλήθεια, ή ψέμα)

```
private int[] diaxwrismosPlhroforiasPshfiakwnAis8hthrwn(int data){
int digital[]={0,0};
switch ( data ) {
        case 0:
        digital[0] = 0; // FALSE
        digital[1] = 0; // FALSE
        break;
        case 1:
        digital[0] = 1; // TRUE
```

```
        digital[1] = 0; // FALSE
        break;
        case 2:
        digital[0] = 0; // FALSE
        digital[1] = 1; // TRUE
        break;
        case 3:
        digital[0] = 1; // TRUE
        digital[1] = 1; // TRUE
        break;
  }//τελος switch
return digital;
}
```

8.9. - Παραλαβή Δεδομένων από τους αισθητήρες στην σειριακή πόρτα

```
private Calendar lastModified;
```

μεταβλητή που δείχνει την τελευταία φορά που πήραμε πακέτο τιμών από τη συσκευή με τους αισθητήρες

```
private boolean readyToRead =false;
```

Μεταβλητή που ελέγχουμε για το εάν είναι έτοιμα τα δεδομένα προς ανάγνωση

```
private boolean readAtLeastOnce=false;
```

Μεταβλητή που δείχνει εάν έχουμε λάβει δεδομένα έστω μία φορά

```
public SeiriakhSyndeshSyskevhsAis8hthrwn(String porta){
super(porta);
}
```

Μέθοδος για την ανάγνωση των δεδομένων από τη σειριακή πόρτα / χειρισμού των συμβάντων της

```
public void serialEvent(SerialPortEvent ev) {
switch (ev.getEventType()) {
case SerialPortEvent.DATA_AVAILABLE:
```

readAtLeastOnce = true;
readyToRead =false;

τα δεδομένα δεν είναι διαθέσιμα για ανάγνωση μέχρι να ολοκληρωθεί η μέθοδος

int data; **//μεταβλητή που αναπαριστά το byte που μόλις διαβάσαμε**
try{
int len=0; **//μέγεθος του ενταμιευτή που αποθηκεύονται τα bytes που παραλαμβάνουμε**

while((data = InStream.read()) >-1) {

για όσο υπάρχουν δεδομένα στο ρεύμα εισόδου από τη σειριακή πόρτα, τα τοποθετούμε στο χώρο αποθήκευσης και συνεχίζουμε να διαβάζουμε έως ότου συναντήσουμε τον χαρακτήρα αλλαγής γραμμής

DedomenaEisodou[len++] = data;
if (data == '\n') {

μόλις τον συναντήσουμε, κάνουμε break ώστε να πακεταριστούν τα δεδομένα και να καταστούν διαθέσιμα για αποστολή.

break;
}
}
τα δεδομένα είναι πλέον έτοιμα για ανάγνωση

readyToRead = true;

κρατάμε ένα στιγμιότυπο της χρονικής στιγμής παραλαβής των δεδομένων

lastModified = Calendar.getInstance();
}
catch (IOException e) {
Recorder.displayMessage(e.toString());
}

```
break;
default:
break;
} //telos switch
}
```

μέθοδος προσπέλασης του χώρου αποθήκευσης των μετρήσεων που παρελήφθησαν

```
public int[] getInBuffer(){
return this.DedomenaEisodou;
}
```

μέθοδος προσπέλασης της χρονικής στιγμής παραλαβής των δεδομένων

```
public Calendar getXronosfragidaLhpshsTimhs(){
return this.lastModified;
}
```

μέθοδος που ενημερώνει εάν είναι έτοιμα για ανάγνωση τα δεδομένα

```
public boolean readyTOread(){
return readyToRead;
}
```

μέθοδος που ενημερώνει εάν έχουμε διαβάσει έστω μια φορά

```
public boolean diavasameEstwMiaFora(){
return readAtLeastOnce;
}
```

8.10. - Αποστολή SMS Text-Mode

Μέθοδος για την ανάγνωση των δεδομένων από τη σειριακή πόρτα / χειρισμού των συμβάντων της

```
public void serialEvent(SerialPortEvent ev) {
switch (ev.getEventType()) {
                        case SerialPortEvent.DATA_AVAILABLE: {
            readyToRead = false;
```

```
                int data;
                try{
                        int len=0;
                        while( (data = InStream.read()) >-1) {
                                this.InBuffer[len++] = data;
                                if (data == '\n') {
                                        break;
                                }
                        }
        readyToRead = true;
                }catch (IOException e) {
                        Recorder.displayMessage(e.toString());
                }
                break;
            }
            default: {
                break;
            }
        }
    }
```

Μέθοδος για την εκκίνηση του νήματος που αναλαμβάνει την αποστολή του SMS

```
    public void send(String s,String recipient){
```

```
            this.SmsMessage = s;
        this.recipient = recipient;
            this.writeThread = new Thread(this);
            writeThread.start();
        }
```

Μέθοδος για τον έλεγχο της απάντησης του modem/κινητού στις AT εντολές

```
        private boolean CheckSMSResponse(CharSequence ok ){
            try{
                Thread.sleep(2000);
            }catch(InterruptedException e){
                Recorder.displayMessage("Diakopika!");
            }
            String s = new String(InBuffer,0,3);
            Recorder.displayMessage(s);
            if (s.contains(ok)){
                Recorder.displayMessage("OK!");
                return true;
            }
            else return false;
        }
```

Νήμα για τη διαδικασία αποστολής του SMS

```
public void run(){
PrintWriter pw = new PrintWriter(this.OutStream,true);
```

```
CharSequence response1 = "K";
CharSequence response2 = ">";
char cntrlZ = (char)26;
pw.println("AT+CMGF=1"); //"AT+CMGF=?"
if (CheckSMSResponse(response1)){
    pw.println("AT+CMGF=1");
    if (CheckSMSResponse(response1)){
        pw.println("AT+CMGS=\""+recipient+"\"");
        if (CheckSMSResponse(response2)){
            pw.println(SmsMessage+" "+cntrlZ);
            if (CheckSMSResponse(response1)){
    Recorder.displayMessage("To Minima Estali Epitixws!");
            }
            else {
    Recorder.displayMessage(new String(InBuffer,0,InBuffer.length));
Recorder.displayMessage("Apotixia stin 4i fasi (Apostoli Minimatos)");
            }
        }
        else{
Recorder.displayMessage(new String(InBuffer,0,InBuffer.length));
Recorder.displayMessage("Apotixia stin 3i fasi (Eisagwgi Arithmou)");
        }
    }
```

```
                    else {
Recorder.displayMessage(new String(InBuffer,0,InBuffer.length));
Recorder.displayMessage("Apotixia stin 2i fasi (Perasma se Text Mode)");
                    }
                }
                else {
        Recorder.displayMessage(new String(InBuffer,0,InBuffer.length));
Recorder.displayMessage("I Apostoli Minimatos Apetixe! 1i fasi (Elegxos Modem)");
                }
        }
```

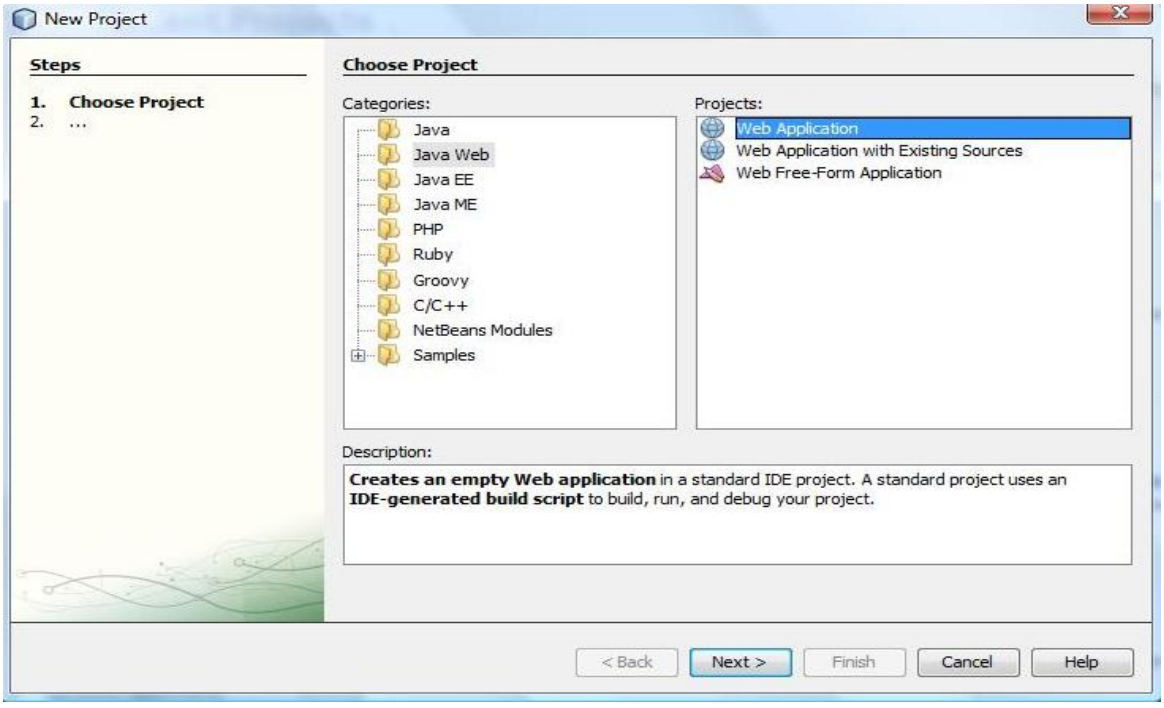
Τέταρτη Ενότητα

Η ΔΙΑΔΙΚΤΥΑΚΗ ΕΦΑΡΜΟΓΗ

ΚΕΦΑΛΑΙΟ 9
Οι τεχνολογίες και το περιβάλλον προγραμματισμού

9.1. - Δημιουργία Web Application μέσω NetBeans IDE 6.7

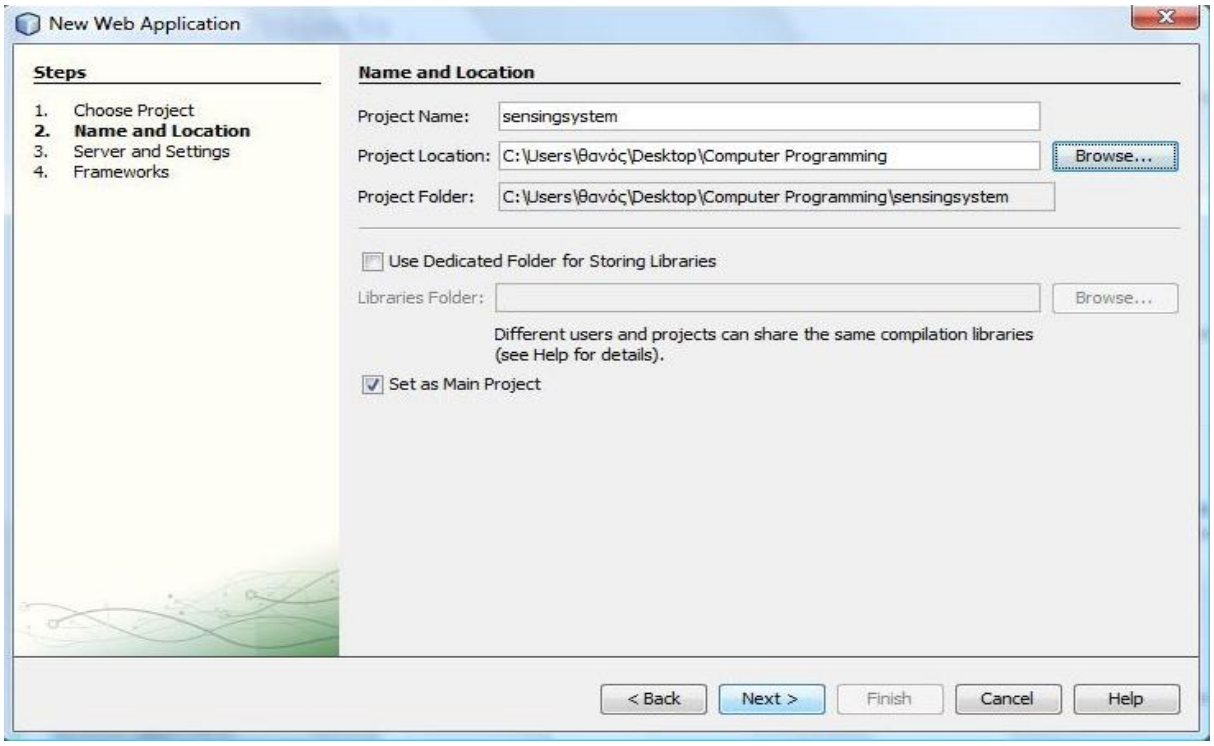
Το NetBeans, είναι ένα ελεύθερο ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού που μας δίνει την δυνατότητα να κατασκευάσουμε γρήγορα και εύκολα, μια εφαρμογή ιστού java, επιλέγοντας Αρχείο -> New Project -> Web Application. *(Σχήμα 9.1)*



Σχήμα 9.1 – Δημιουργία Project

Έπειτα πρέπει να επιλέξουμε το όνομα του project μας, τον κατάλογο που θα αποθηκευτούν τα αρχεία του project, αλλά και τον κατάλογο που θα

χρησιμοποιηθεί για την αποθήκευση άλλων βιβλιοθηκών που θα χρησιμοποιηθούν στο project *(Σχήμα 9.2)*.

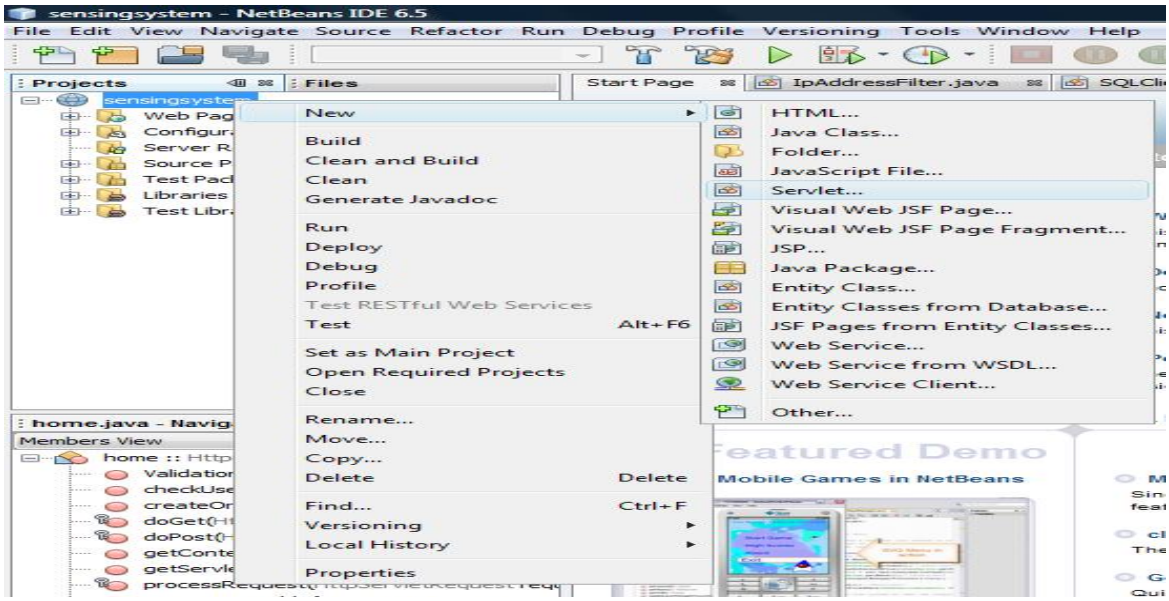


Σχήμα 9.2 – Δημιουργία Web Application

Έπειτα, επιλέγουμε τον web server που θα χρησιμοποιεί το application, την έκδοση της Enterprise Edition έχουμε ήδη στην πλατφόρμα μας και την παράμετρο Context Path την αφήνουμε ως έχει, με το όνομα του project μας *(Σχήμα 9.3)*.

Σχήμα 9.3 – Επιλογή διακομιστή Web εφαρμογών

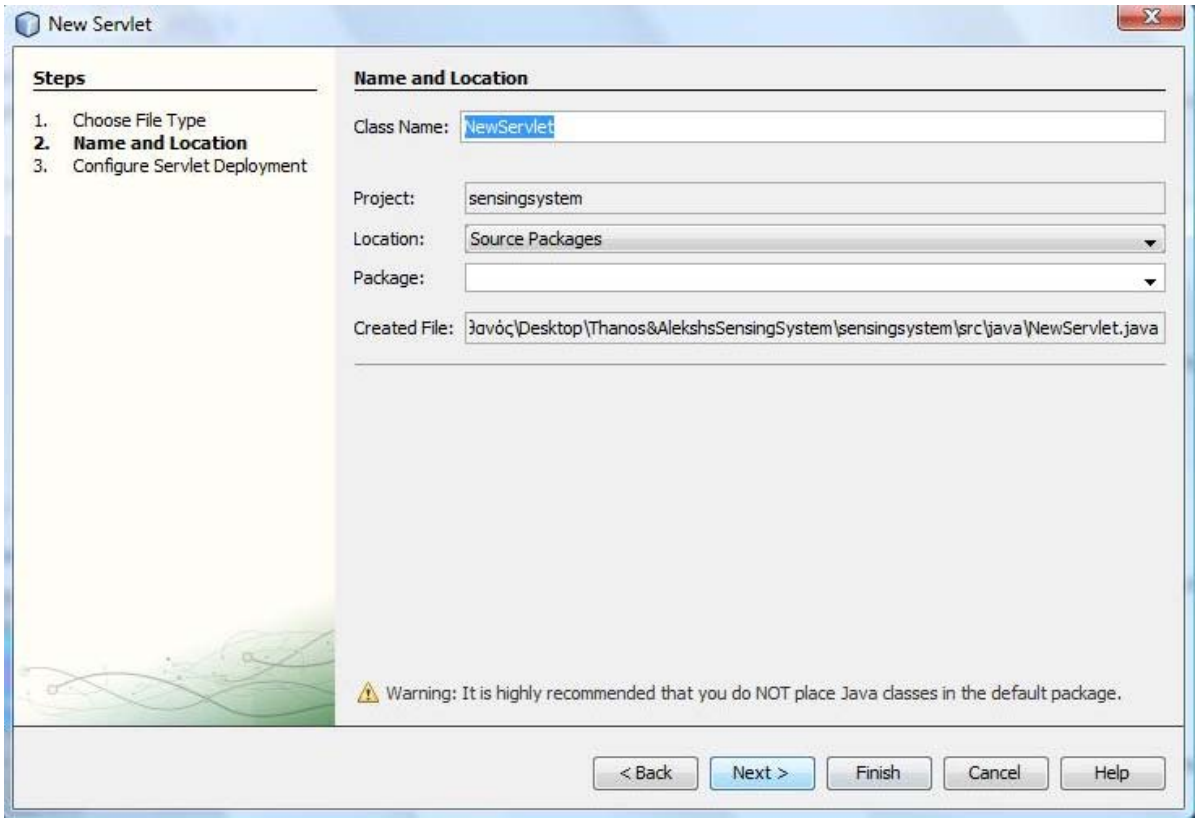
Έπειτα, αφού δημιουργηθεί το project, στην αριστερή στήλη με τίτλο projects, παρατηρούμε πως έχει δημιουργηθεί μια δενδρική δομή καταλόγων με ρίζα το όνομα του project μας, που περιλαμβάνει τους παρακάτω καταλόγους: «Web Pages», «Configuration Files», «Server Resources», «Source Packages», «Test Packages», «Libraries», «Test Libraries».



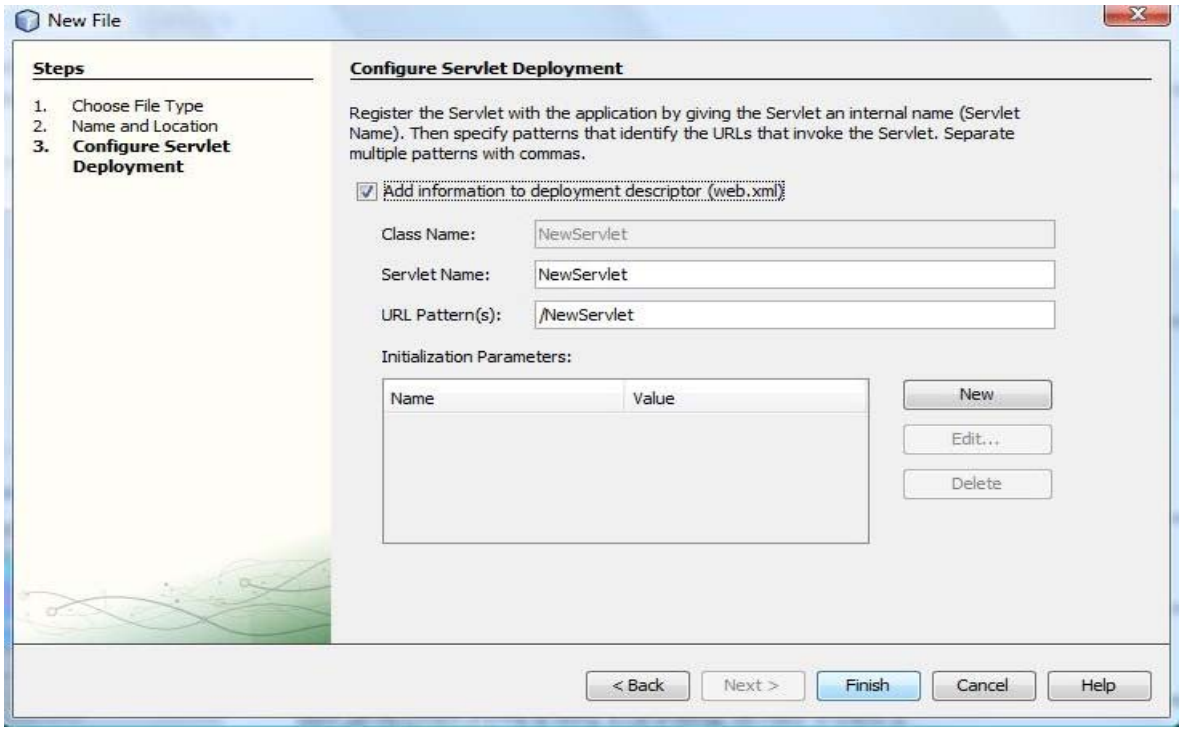
Σχήμα 9.4.α – Δημιουργία μικροϋπηρεσίας

Έπειτα, για να κατασκευάσουμε μια μικροϋπηρεσία (servlet), κάνουμε δεξί κλικ πάνω στη ρίζα κατάλογο, επιλέγουμε New -> Servlet και επιλέγουμε όνομα για το αρχείο μας, το όνομα του project, το όνομα του πακέτου που ενδεχομένως να έχουμε φτιάξει *(Σχήμα 9.4)*.

Τέλος, επιλέγουμε εάν επιθυμούμε να προστεθεί μια νέα εγγραφή για αυτό το servlet στο αρχείο web.xml και ορίζουμε, εάν το επιθυμούμε, και παραμέτρους αρχικοποίησης για αυτό το servlet *(Σχήμα 9.5)*. Ακολουθώντας την ίδια διαδικασία, δημιουργούμε αναλόγως όλα τα servlet, ή jsp αρχεία της εφαρμογής ιστού μας και στο τέλος κάνουμε Build για να μεταγλωττίσουμε το project μας, ώστε να παραχθεί το εκτελέσιμο αρχείο, το web archive.



Σχήμα 9.4.b – Δημιουργία μικροϋπηρεσίας



Σχήμα 9.5 – Προσθήκη παραμέτρων μικροϋπηρεσίας στο web.xml

9.2. - Apache Tomcat Web Server 6.0.18

Περιγραφή του Apache Tomcat

Ο Apache Tomcat αποτελεί τον πιο διαδεδομένο διακομιστή ιστού (Web Server) αυτή τη στιγμή στο διαδίκτυο, περίπου το 80% των διακομιστών εφαρμογών στηρίζονται σε αυτόν. Αφού εγκατασταθεί σε έναν υπολογιστή, του δίνει την δυνατότητα να εξυπηρετήσει αιτήσεις ενός υπολογιστή πελάτη που αφορούν τον ιστό. Τέτοιες αιτήσεις φυσικά περιλαμβάνουν την παροχή μιας ιστοσελίδας, το ανέβασμα, ή το κατέβασμα αρχείων, την αποστολή emai κτλ. Στην πραγματικότητα η μηχανή μικροϋπηρεσιών (Servlet Engine) χειρίζεται αιτήσεις μικροϋπηρεσιών, ιστοσελίδων και σελίδων JSP.

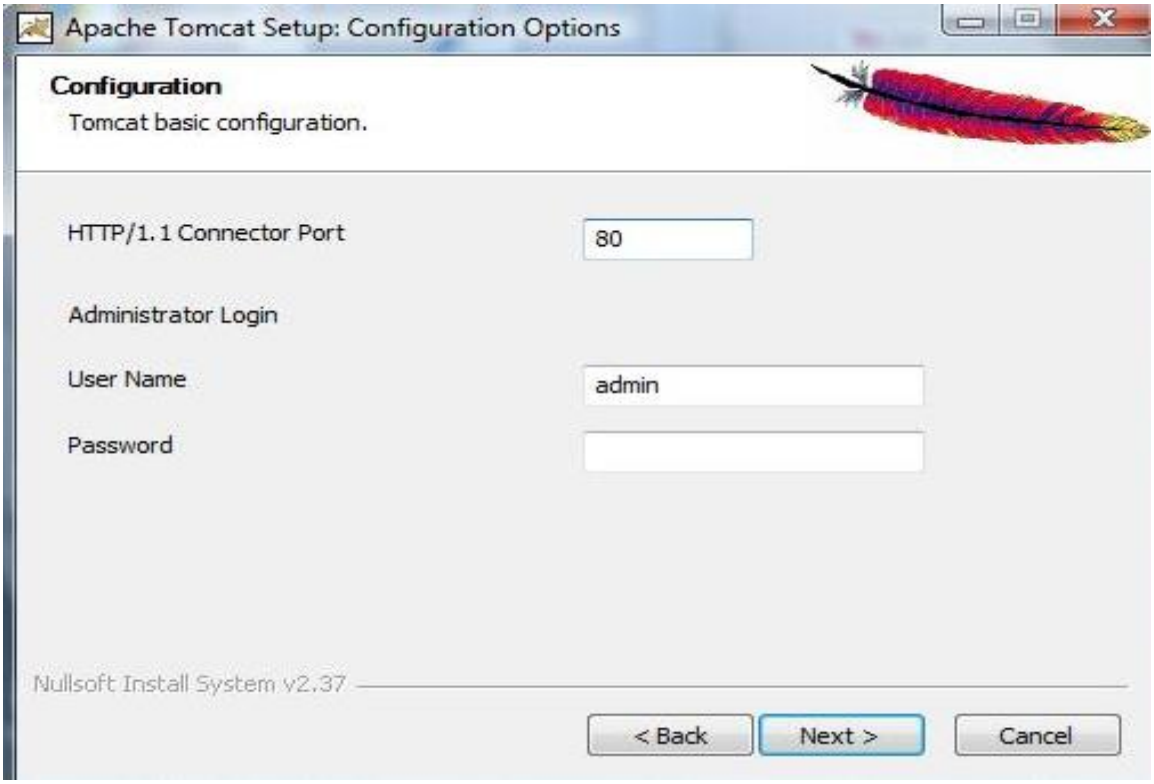
9.2.1. - Εγκατάσταση και Διαμόρφωση του Web Server

Στο διαδίκτυο υπάρχουν σήμερα πολλές εκδόσεις του διακομιστή ιστού apache-tomcat για διάφορα λειτουργικά συστήματα. Τα στατιστικά λένε ότι πάνω του 80% των ιστοτόπων βασίζονται σε αυτόν. Εμείς επιλέξαμε την τελευταία έκδοση 6.0.18 που παρέχεται σε ένα μόνο εκτελέσιμο αρχείο. Αρχικά, αυτό που θα πρέπει να διασφαλίσουμε, είναι η δημιουργία της μεταβλητής περιβάλλοντος $JAVA_HOME, προσδιορίζοντας τον κατάλογο εγκατάστασης του Sun Development Kit.

Σχήμα 9.6.α – Εγκατάσταση Apache Tomcat

Αφού προχωρήσουμε στην διαδικασία, αποδεχόμενοι τους όρους και επιλέγοντας εκείνα τα έξτρα στοιχεία που μπορεί να επιθυμούμε να

εγκατασταθούν, όπως παραδείγματα κώδικα, έπειτα πρέπει να επιλέξουμε την πόρτα που θα τρέχει η υπηρεσία του apache-tomcat αλλά και να ορίσουμε το όνομα και των κωδικό του διαχειριστή του διακομιστή ιστού.



Σχήμα 9.6.b – Εγκατάσταση Apache Tomcat

Αφού συνεχίσουμε την διαδικασία, ο διακομιστής ιστού έχει εγκατασταθεί στην πλατφόρμα μας. Πλέον, μπορούμε να συνεχίσουμε ορίζοντας τις μεταβλητές περιβάλλοντος: τη μεταβλητή χρήστη **CATALINA_HOME**, δίνοντας της ως τιμή το μονοπάτι στο τοπικό σύστημα αρχείων, που εγκαταστήσαμε την δική μας έκδοση του Apache-Tomcat.

Π.χ. **C:\Program Files\Apache Software Foundation\Tomcat 6. 0**

Μέσα σε αυτό τον κατάλογο, βρίσκονται οι Βασικοί φάκελοι λειτουργίας του διακομιστή ιστού Apache-Tomcat:

- **/bin** – Εδώ βρίσκονται τα προγράμματα εκκίνησης και τερματισμού του διακομιστή μας, αλλά και άλλα. Τα αρχεία με κατάληξη .sh είναι για τα Unix συστήματα, ενώ για τα Windows είναι τα .bat αρχεία.

- **/conf** – Εδώ βρίσκονται τα κύρια αρχεία διαμόρφωσης του διακομιστή ιστού. Το πιο σημαντικό αρχείο εδώ μέσα, είναι το server.xml. Είναι το βασικό αρχείο διαμόρφωσης και παραμετροποίησης του υποδοχέα ιστού (Web Container).
- **/logs** – Τα αρχεία log τοποθετούνται εδώ αυτόματα.
- **/webapps** – Εδώ τοποθετούνται οι εφαρμογές ιστού μας, δηλαδή όλα τα αρχεία με κατάληξη .war
- **/lib** –Εδώ βρίσκονται οι βιβλιοθήκες του Apache-tomcat διακομιστή ιστού, αλλά και επίσης εδώ τοποθετούνται οι drivers (οδηγοί) των διαφόρων βάσεων δεδομένων που ενδεχομένως να χρησιμοποιούν οι εφαρμογές ιστού μας. Ανάλογα με το εάν χρησιμοποιούμε Oracle, Mysql, Postegres, Derby κτλ, τοποθετούμε τον κατάλληλο connector, ένα αρχείο .jar μέσα σε αυτόν τον κατάλογο.

9.2.2. - Εκκίνηση web server από την γραμμή εντολών, ή μέσω της υπηρεσίας tomcat6

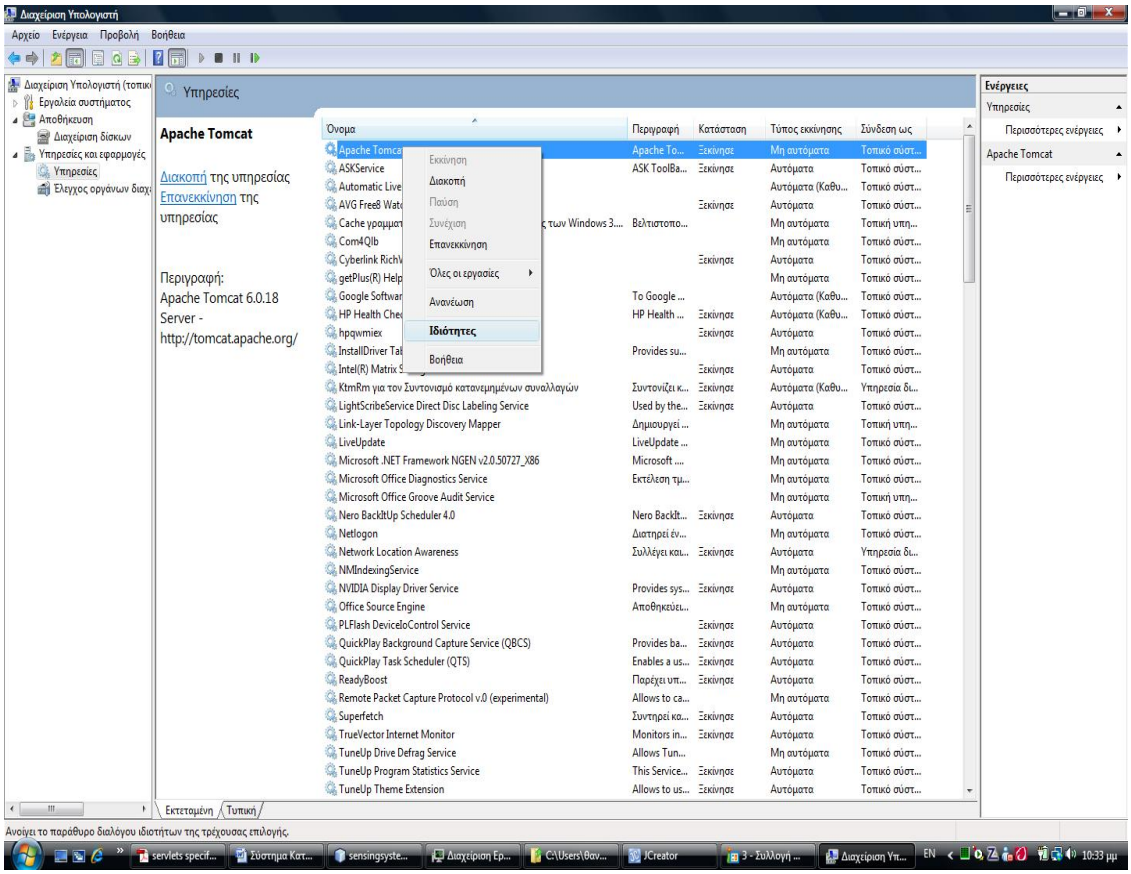
Δεξί κλικ στο εικονίδιο Ο Υπολογιστής μου -> ΔΙΑΧΕΙΡΙΣΗ -> Υπηρεσίες και Εφαρμογές, δεξί κλικ πάνω στην υπηρεσία Apache tomcat 6 -> Ιδιότητες *(Σχήμα 9.7)*

- Από την πρώτη καρτέλα, «Γενικά», μπορούμε να επιλέξουμε εάν θα εκκινεί αυτόματα η υπηρεσία με την εισαγωγή στο περιβάλλον χρήστη, ή εάν θα εκκινεί χειροκίνητα. Επίσης από εδώ ξεκινάμε και σταματάμε την υπηρεσία του διακομιστή ιστού *(Σχήμα 9.8)*.
- Στην δεύτερη καρτέλα, «Σύνδεση», μπορούμε να ορίσουμε ότι θα συνδεθούμε με τον τοπικό λογαριασμό, ή να ορίσουμε κάποιον καινούργιο λογαριασμό *(Σχήμα 9.9)*.
- Στην τρίτη καρτέλα, «Αποκατάσταση», μπορούμε να ορίσουμε την απόκριση του υπολογιστή μετά από μία, ή περισσότερες διαδοχικές αποτυχίες της υπηρεσίας του διακομιστή ιστού *(Σχήμα 9.10)*.

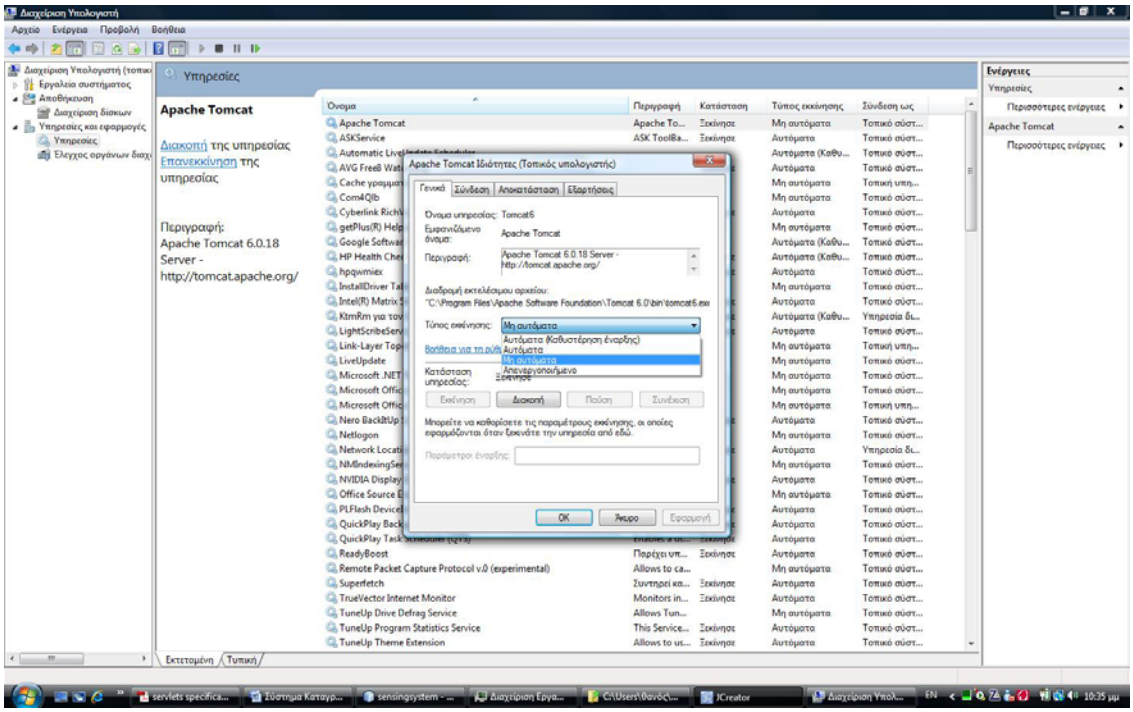
Ειδάλλως, μπορούμε να πάμε από την ΕΝΑΡΞΗ -> ΠΡΟΓΡΑΜΜΑΤΑ -> APACHE TOMCAT 6. 0 -> Monitor Tomcat, αφού το ανοίξουμε, κάνουμε δεξί κλικ στο εικονίδιο apache tomcat από το launch bar και επιλέγουμε “start service” για

να εκκινήσουμε τον διακομιστή ιστού μας και “stop service” για να τον σταματήσουμε.

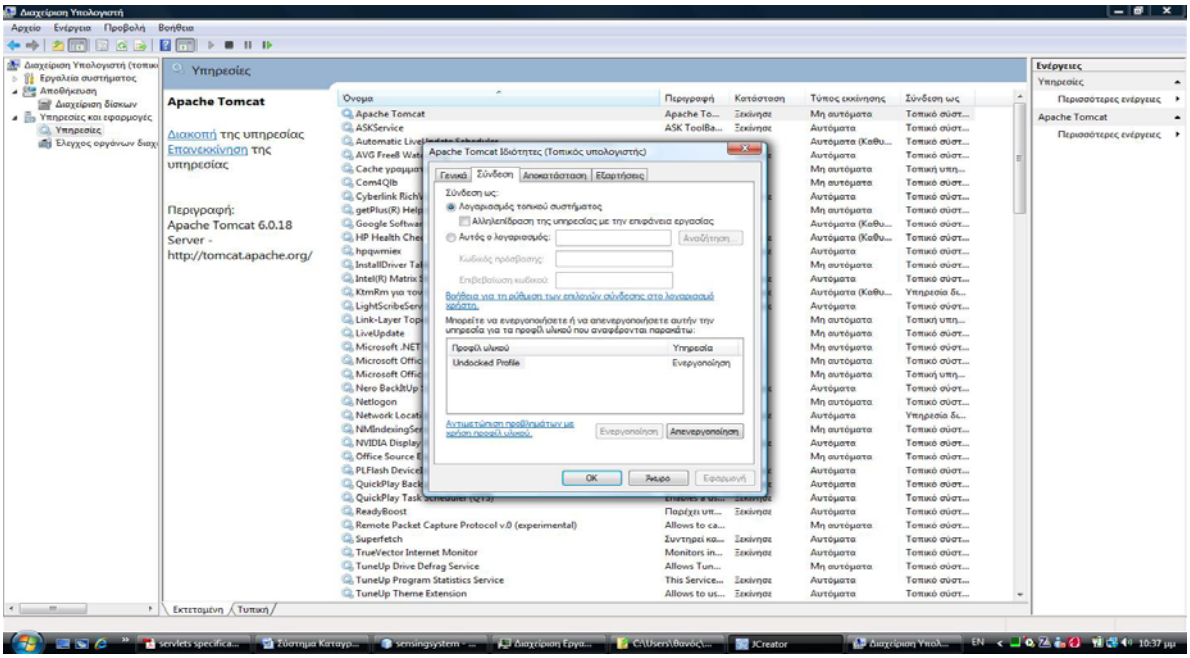
Για να δουλέψουμε από την γραμμή εντολών, θα πρέπει να έχουμε ορίσει επιτυχώς την μεταβλητή χρήστη $CATALINA_HOME ώστε να μπορεί η γραμμή εντολών να αναγνωρίζει τα προγράμματα του διακομιστή apache-tomcat. Πηγαίνουμε στον κατάλογο που έχουμε τοποθετήσει τον κατάλογο apache-tomcat-6.0.xx, πηγαίνουμε στον κατάλογο \bin και καλούμε το startup.bat για να εκκινήσουμε την υπηρεσία μας, ενώ το shutdown.bat, όποτε θελήσουμε να σταματήσουμε την υπηρεσία του διακομιστή ιστού apache tomcat.



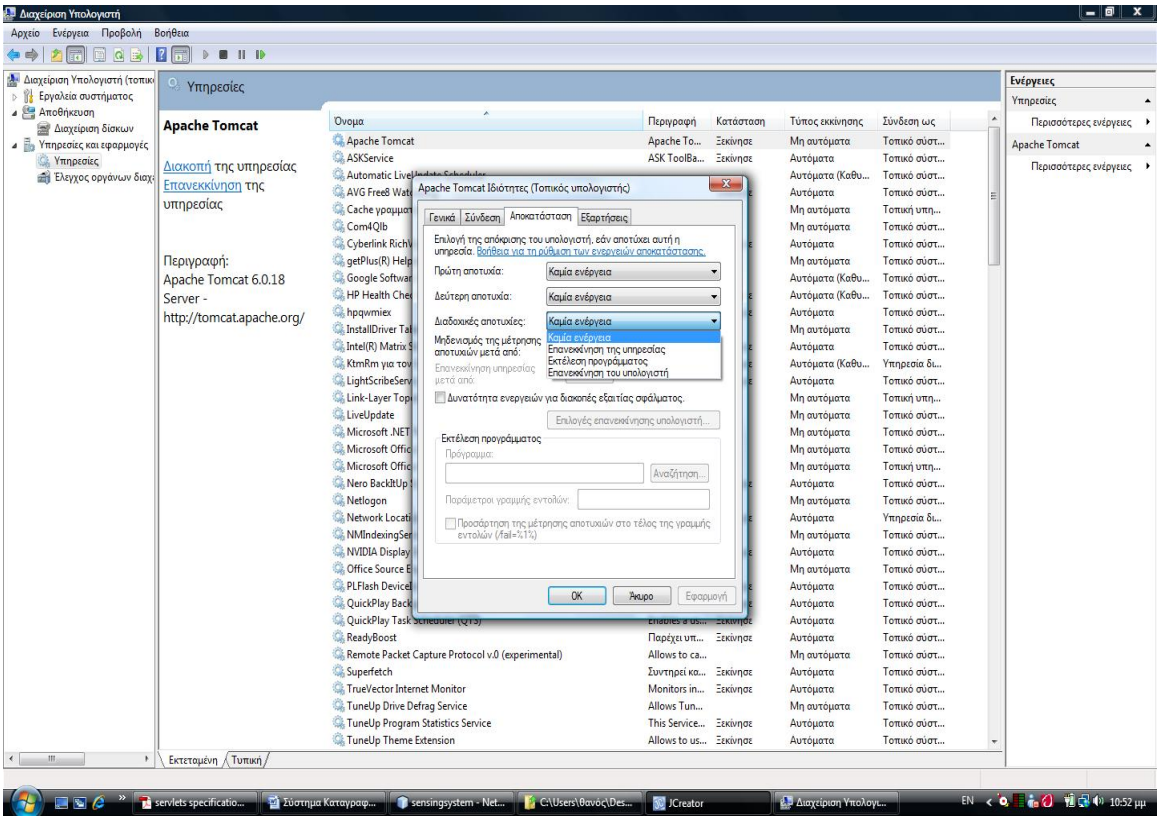
Σχήμα 9.7 – Άνοιγμα παραθύρου ιδιοτήτων



Σχήμα 9.8 – Ορισμός αυτόματης εκκίνησης



Σχήμα 9.9 – Ορισμός είδους λογαριασμού

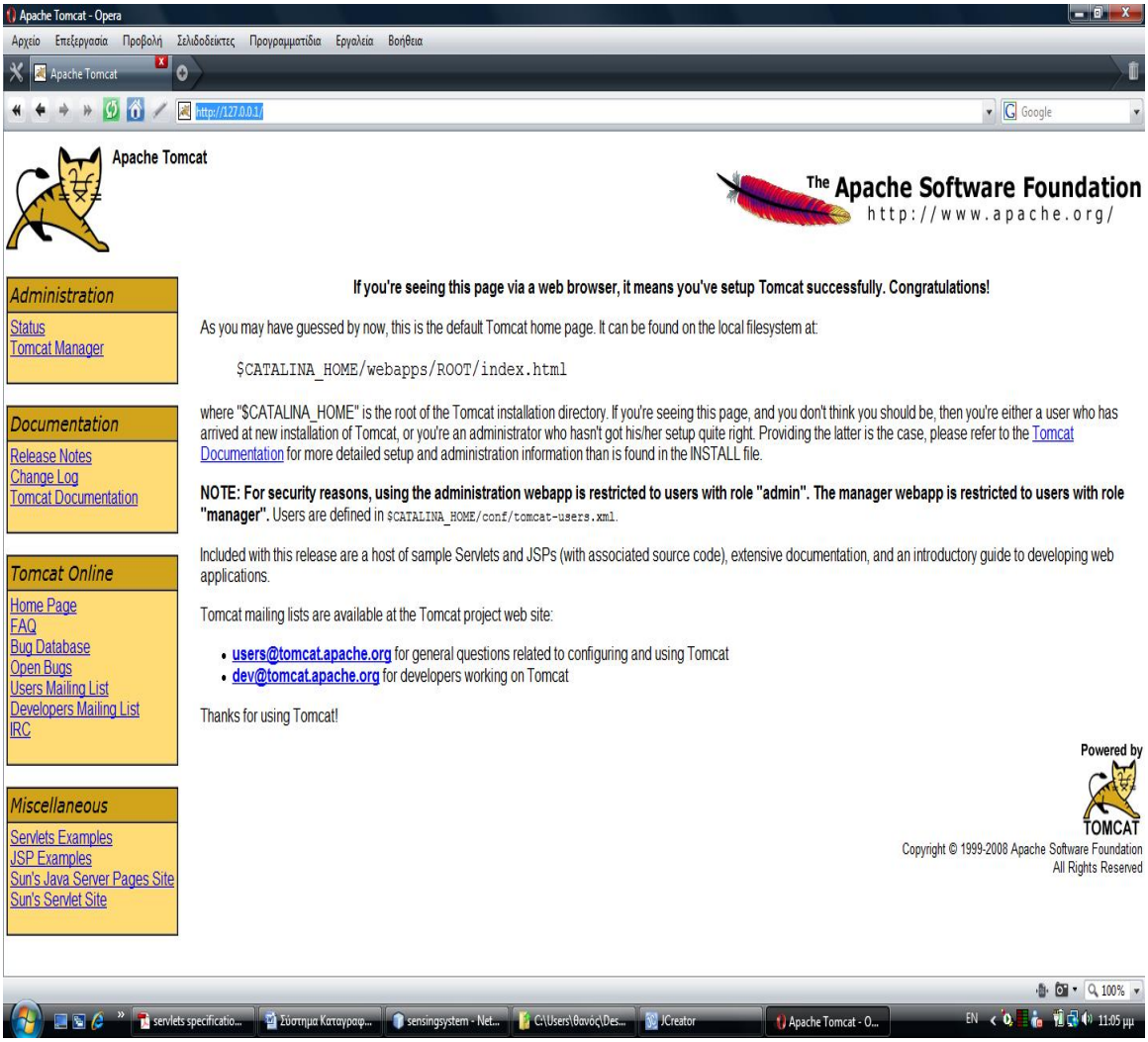


Σχήμα 9.10 – Ορισμός ενεργειών σε αποτυχία του διακομιστή

9.2.3. - Διαχείριση του διακομιστή ιστού και των Web Applications

Αφού έχουμε ολοκληρώσει με επιτυχία την εγκατάσταση και την διαμόρφωση του περιβάλλοντος μας και αφού έχουμε εκκινήσει με τον έναν, ή με

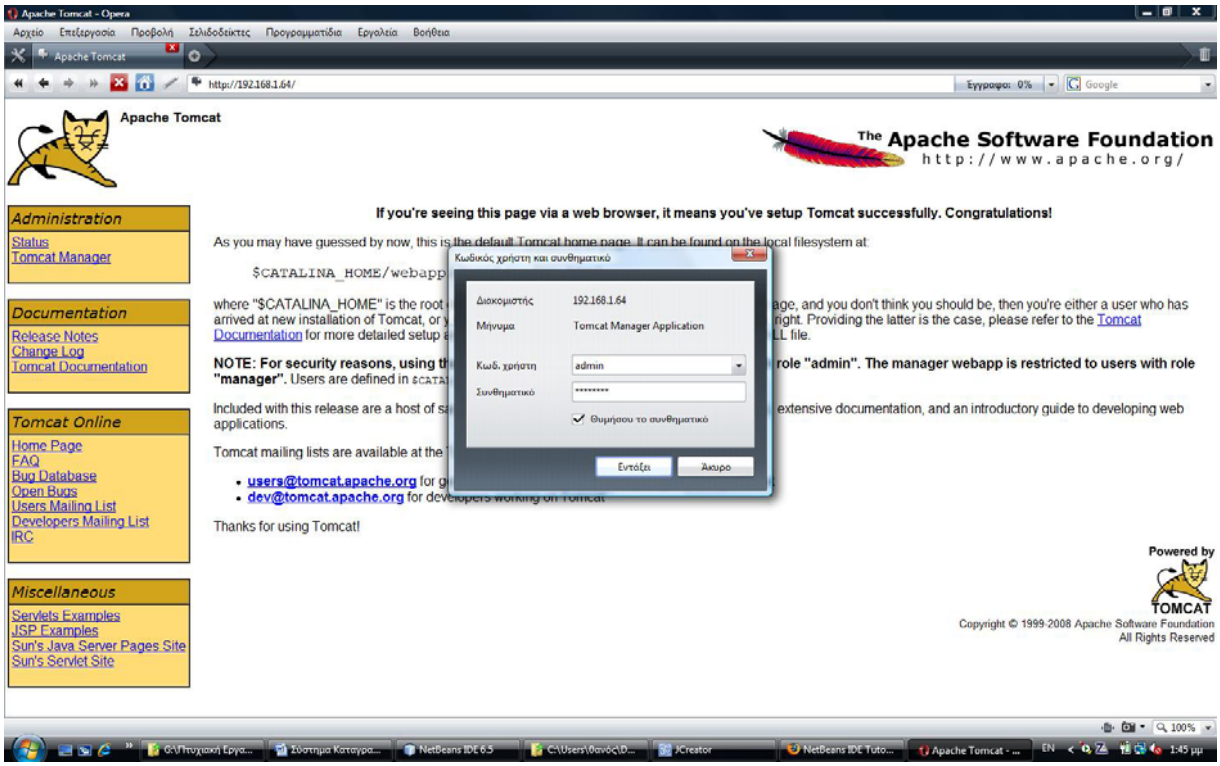
τον άλλο τρόπο την υπηρεσία του διακομιστή ιστού μας, ανοίγουμε ένα παράθυρο οποιουδήποτε φυλλομετρητή (web client) και γράφουμε http://127.0.0.1/ εάν δεν έχουμε κάποια σύνδεση με το διαδίκτυο, ή εάν έχουμε σύνδεση δίνουμε τη διεύθυνση διαδικτύου μας. Εάν όλα έχουν εξελιχθεί καλά, τότε θα πρέπει να πάρουμε ως απάντηση την ιστοσελίδα του *Σχήματος 9.11*, που είναι και το βασικό περιβάλλον υποδοχής του διακομιστή Apache-Tomcat. Από εδώ μέσα, ο διαχειριστής του διακομιστή, έχει την δυνατότητα να διαχειριστεί τον διακομιστή ιστού tomcat, να μελετήσει ολόκληρη την τεκμηρίωση του λογισμικού αυτού, να εκτελέσει έτοιμα παραδείγματα μικροϋπηρεσιών, να εντοπίσει τις αιτίες των σφαλμάτων κατά την διάρκεια εκτέλεσης των εφαρμογών ιστού, από την on-line βάση δεδομένων του Apache-tomcat κ.α.



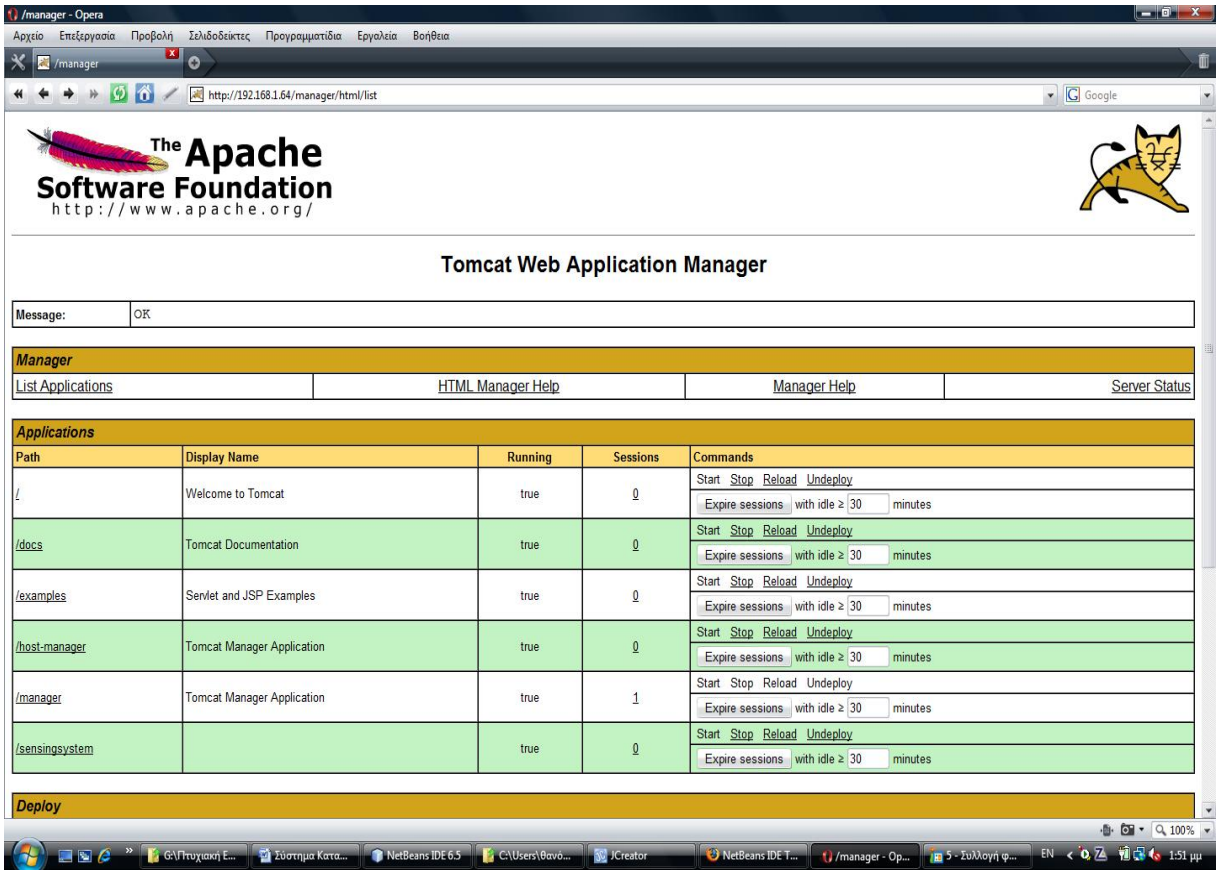
Σχήμα 9.11 – Το περιβάλλον του διακομιστή Tomcat

Εμείς επιλέγουμε “Tomcat Manager” και παρατηρούμε ότι μας ζητείται όνομα χρήστη και κωδικός *(Σχήμα 9.12)*, οπότε δίνουμε το όνομα και των κωδικό

του διαχειριστή που ορίσαμε κατά την εγκατάσταση του Apache-Tomcat *(Σχήμα 9.13)*.



Σχήμα 9.12 – Εισαγωγή όνομα χρήστη και κωδικού πρόσβασης



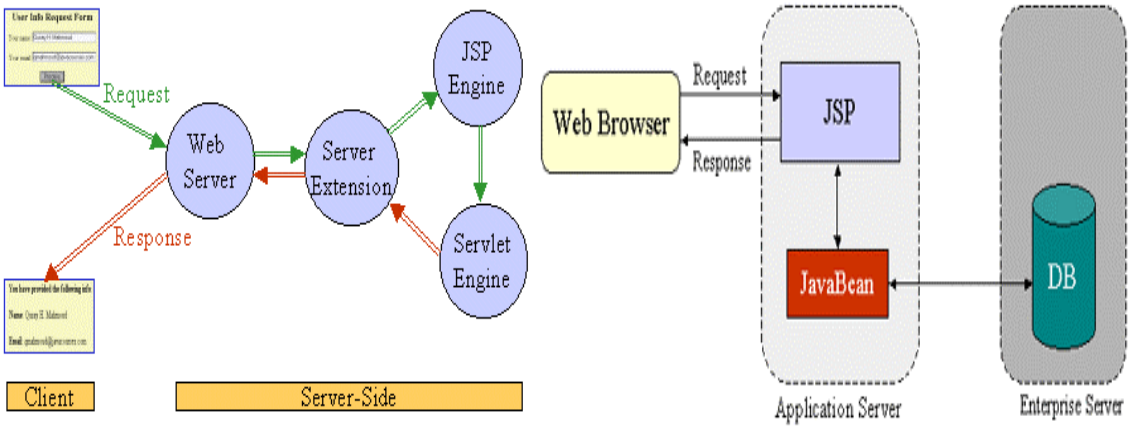
Σχήμα 9.13 – Διαχειριστής εφαρμογών ιστού

9.3. - Οι μικροϋπηρεσίες και οι σελίδες JSP

Η μικροϋπηρεσία είναι κώδικας java ο οποίος εκτελείται μόνο κάτω από το περιβάλλον ενός διακομιστή ιστού. Ο αγγλικός όρος προέρχεται από το συνδυασμό των λέξεων applet (μικρό-εφαρμογή) και server (διακομιστής). Συνεπώς, το servlet είναι η μικρό-εφαρμογή που λειτουργεί κάτω από έναν διακομιστή εφαρμογών. Η μικροϋπηρεσία «εκπέμπεται» μέσα από ένα http://url για να παράσχει υπηρεσίες σε έναν υπολογιστή-πελάτη. Η μικροϋπηρεσία μπορεί να λειτουργήσει και ως «ενδιάμεσος», ανάμεσα σε μια HTTP αίτηση προερχόμενη από έναν web browser ενός πελάτη-υπολογιστή και τον αποδέκτη της αίτησης, που μπορεί να είναι μια βάση δεδομένων, ή ένας άλλος διακομιστής ιστού (http).

Οι σελίδες JSP, είναι ένας τύπος μικροϋπηρεσίας java που σχεδιάστηκε για να ικανοποιήσει το ρόλο του γραφικού περιβάλλοντος αλληλεπίδρασης χρήστη για τις εφαρμογές ιστού. Οι προγραμματιστές εφαρμογών ιστού, δημιουργούν σελίδες JSP ως αρχεία κειμένου που συνδυάζουν περιεχόμενο HTML, ή κώδικα XHTML, στοιχεία XML και οδηγίες JSP. Οι σελίδες JSP σχεδιάστηκαν πάνω στα πρότυπα των τεχνολογιών από την πλευρά του διακομιστή, όπως η ASP της Microsoft.

Ο μεταφραστής σελίδων JSP, όπως ο Jasper του Tomcat, μετατρέπει αυτόματα το βασισμένο σε κείμενο έγγραφο x.jsp, σε μια μικροϋπηρεσία. Ο υποδοχέας ιστού, δημιουργεί ένα στιγμιότυπο της μικροϋπηρεσίας και την καθιστά διαθέσιμη προς εξυπηρέτηση αιτήσεων. Η δημιουργία της σελίδας JSP ως απλό έγγραφο κειμένου, παρά ως αρχείο java, επιτρέπει τον διαχωρισμό της σχεδίασης του γραφικού περιβάλλοντος της εφαρμογής μας, από την υπόλοιπη υλοποίηση της.



Σχήμα 9.14 – Οι μικροϋπηρεσίες και οι σελίδες jsp

9.3.1. - Τι είναι ο υποδοχέας ιστού (Web Container)

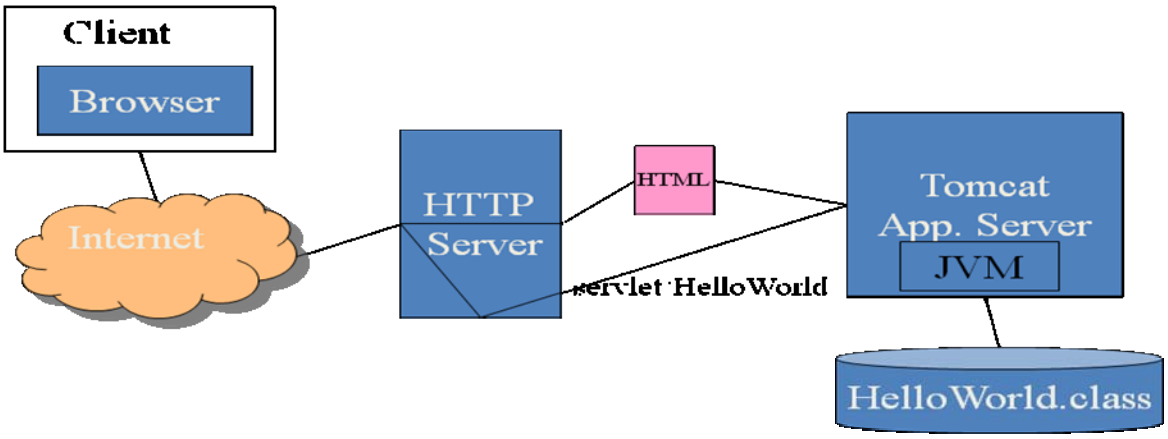
Ο υποδοχέας μικροϋπηρεσιών (servlet) είναι μέρος ενός διακομιστή ιστού, ή ενός διακομιστή εφαρμογών και παρέχει τις υπηρεσίες δικτύου, από τις οποίες στέλνονται αιτήσεις και απαντήσεις, αποκωδικοποιούνται τα MIME αιτήματα και μορφοποιούνται οι MIME απαντήσεις. Ένας υποδοχέας μικροϋπηρεσιών περιέχει και διαχειρίζεται τα servlets μέσω του κύκλου της ζωής τους. Ένας υποδοχέας servlet μπορεί να «χτιστεί» σε έναν κεντρικό υπολογιστή δικτύου, ή να εγκατασταθεί ως πρόσθετο συστατικό σε έναν υπολογιστή δικτύου μέσω της επέκτασης της εγγενούς διεπαφής προγραμματισμού (API), εκείνου του υπολογιστή. Οι υποδοχείς Servlet μπορούν επίσης να εγκατασταθούν στους διακομιστές εφαρμογών. Όλοι οι υποδοχείς μικροϋπηρεσιών πρέπει να υποστηρίζουν το HTTP ως βασικό πρωτόκολλο για τα αιτήματα και τις απαντήσεις, αλλά και επίσης πρέπει να υποστηρίζουν και πρωτόκολλα όπως το HTTPS (HTTP διαμέσου SSL) που μας εξασφαλίζει ασφαλείς κρυπτογραφημένες συνεδρίες.

Μια μικροϋπηρεσία (servlet), καλείται στην ζωή, όταν ο υποδοχέας βρίσκει την κλάση της. Αυτό συμβαίνει συνήθως όταν εκκινούμε τον υποδοχέα ιστού (web container), δηλαδη όταν εκκινούμε τον Tomcat. Όταν ο υποδοχέας ξεκινάει, αναζητάει δημιουργημένες εφαρμογές ιστού και έπειτα ξεκινάει την αναζήτηση των αρχείων των μικροϋπηρεσιών. Το να βρεί τις κλάσεις είναι το πρώτο βήμα. Το φόρτωμα της κλάσης είναι η δεύτερη ενέργεια και αυτό συμβαίνει είτε στην εκκίνηση του υποδοχέα ιστού, είτε στην πρώτη κλήση του πελάτη. Η μέθοδος service() δεν θα κληθεί εως ότου η μικροϋπηρεσία αρχικοποιηθεί πλήρως.

9.3.2. - Η Αλληλεπίδραση με τον Client

Τα βήματα που ακολουθούνται για την εξυπηρέτηση μιας αίτησης από ένα servlet, είναι:

1. Ο πελάτης προσπελαύνει έναν διακομιστή ιστού πραγματοποιώντας μια http αίτηση προς αυτόν.
2. Η αίτηση παραλαμβάνεται από το διακομιστή ιστού και τη διαχειρίζεται ο υποδοχέας servlet. Ο υποδοχέας servlet με την βοήθεια των κεφαλίδων της αίτησης, διαβάζει τις παραμέτρους που αφορούν την αίτηση.
3. Ο υποδοχέας servlet καλεί την κατάλληλη μικροϋπηρεσία, η οποία μπορεί με την σειρά της να χρησιμοποιεί το jdbc, ή άλλα API's, για να ικανοποιήσει την αίτηση.
4. Η μικροϋπηρεσία χρησιμοποιώντας το αντικείμενο της αίτησης του πελάτη, μπορεί να προσδιορίσει ποιος είναι ο απομακρυσμένος πελάτης υπολογιστής, ποιες http POST παράμετροι στάλθηκαν ως μέρος της αίτησης του πελάτη και άλλα σχετικά δεδομένα. Η μικροϋπηρεσία, διαμορφώνει και παράγει δυναμικά δεδομένα, σύμφωνα με την φύση της εφαρμογής και τα αποστέλλει στον web client, στον φυλλομετρητή ιστού του υπολογιστή του πελάτη, διαμέσου του αντικειμένου http απάντησης.
5. Μόλις ολοκληρώσει το servlet, η μικροϋπηρεσία, το αίτημα, ο υποδοχέας servlet εξασφαλίζει ότι η απάντηση ορίστηκε κατάλληλα και επιστρέφει τον έλεγχο πίσω στον διακομιστή ιστού ώστε να στείλει το έγγραφο πίσω στον πελάτη, συνήθως σε μορφή html *(Σχήμα 9.15)*.



Σχήμα 9.15 – Η αλληλεπίδραση με τον πελάτη

9.3.3. - Σύγκριση των μικροϋπηρεσιών με άλλες τεχνολογίες

Τα πλεονεκτήματα των μικροϋπηρεσιών της java, έναντι άλλων τεχνολογιών, όπως των CGI Scripts, εντοπίζονται στα εξής σημεία:

1. Είναι ταχύτερα των CGI Scripts διότι υλοποιούνται βάση ενός εντελώς διαφορετικού μοντέλου επεξεργασίας. Στα σενάρια CGI, δημιουργείται κάθε φορά μια νέα διεργασία για κάθε αίτηση http. Η εκτέλεση μιας μικροϋπηρεσίας, δεν απαιτεί την δημιουργία μιας νέας διεργασίας κάθε φορά που εκτελείται ο κώδικας της, αντιθέτως η μικροϋπηρεσία παραμένει στην μνήμη, καθώς τρέχει η εικονική μηχανή της java και συνεπώς δεν χρειάζεται να ξαναφορτώνεται κάθε φορά που ζητείται, αλλά μονάχα εάν αλλάξουν τα στοιχεία σε αυτήν. Η εικονική μηχανή της java χειρίζεται την κάθε αίτηση, με ένα ελαφρύ νήμα (java thread) και όχι με μια βαριά διεργασία (process) του λειτουργικού συστήματος. Στο μοντέλο επεξεργασίας των CGI Scripts, εάν υπάρξουν n αιτήσεις, θα φορτωθεί ο κώδικας στην μνήμη n φορές. Στα java servlets, θα δημιουργηθούν n νήματα για n αιτήσεις, αλλά θα φορτωθούν μονάχα μια φορά οι κλάσεις στην μνήμη.
2. Έχουν το σημαντικό πλεονέκτημα της ανεξαρτησίας πλατφόρμας εκτέλεσης που προσφέρει η εικονική μηχανή της java. Έτσι πολύ απλά, οι μικροϋπηρεσίες μπορούν να μεταφερθούν σε διαφορετικά λειτουργικά συστήματα χωρίς να απαιτείται να ξανά μεταγλωττίσουν τον πηγαίο κώδικα.
3. Ακόμα και να μην υλοποιείται προγραμματιστικά κάποια πολιτική ασφάλειας, κάποιος έλεγχος, οι μικροϋπηρεσίες εκτελούνται στο ασφαλές περιβάλλον της εικονικής μηχανής της java. Έτσι, αποτρέπονται οι απευθείας επιθέσεις στην

μνήμη RAM, αλλά και οι επιθέσεις υπερχείλισης μνήμης που μπορούν εύκολα να υλοποιηθούν με γλώσσες όπως C και C++. Ακόμα και εάν η μικροϋπηρεσία πραγματοποιήσει μια κλήση συστήματος (Runtime.exec or JNI), για να καλέσει κάποιο άλλο πρόγραμμα στο τοπικό σύστημα, αυτό δεν θα πραγματοποιηθεί απευθείας, αλλά διαμέσου της εικονικής μηχανής της java προς το λειτουργικό, οπότε εξασφαλίζεται ότι δεν μπορεί να εκτελεστεί κάποιος κακόβουλος κώδικας προς το λειτουργικό σύστημα.

9.3.4. - Μέθοδοι χειρισμού των αιτήσεων HTTP

Η κλάση HttpServlet, μας παρέχει τις μεθόδους εκείνες που καλούνται αυτόματα από την μέθοδο service για τον χειρισμό και την επεξεργασία των http αιτήσεων. Αυτές ο μέθοδοι είναι:

doGet: για τον χειρισμό των Http GET αιτήσεων
doPost: για τον χειρισμό των Http POST αιτήσεων
doPut: για τον χειρισμό των Http PUT αιτήσεων
doDelete: για τον χειρισμό των Http DELETE αιτήσεων
doHead: για τον χειρισμό των Http HEAD αιτήσεων
doOptions: για τον χειρισμό των Http OPTIONS αιτήσεων
doTrace: για τον χειρισμό των Http TRACE αιτήσεων

9.3.5. - Ο χειρισμός της Αίτησης

Η παρακάτω ακολουθία γεγονότων συμβαίνει όταν ένας υποδοχέας ιστού δημιουργεί ένα στιγμιότυπο μιας μικροϋπηρεσίας:

1. Ο υποδοχέας μικροϋπηρεσιών καλεί την μέθοδο init() της μικροϋπηρεσίας, η οποία είναι σχεδιασμένη ώστε να αρχικοποιεί τους πόρους που θα χρησιμοποιήσει η μικροϋπηρεσία. Η μέθοδος αυτή καλείται μόνο μια φορά στο κύκλο ζωής της μικροϋπηρεσίας.
2. Η init() μέθοδος, αρχικοποιεί ένα αντικείμενο που υλοποιεί τη διεπαφή javax.servlet.ServletConfig. Αυτό το αντικείμενο, δίνει στη μικροϋπηρεσία πρόσβαση στις παραμέτρους αρχικοποιήσης που έχουν δηλωθεί στην περιγραφή της εφαρμογής ιστού. Επίσης, αυτό το αντικείμενο παρέχει στην μικροϋπηρεσία πρόσβαση στο αντικείμενο javax.servlet.ServletContext, μέσω του οποίου μπορούμε να διατηρήσουμε ένα log αρχείο με μηνύματα, να ανακατευθύνουμε τις αιτήσεις προς άλλα αντικείμενα ιστού και να

αποκτήσουμε πρόσβαση προς άλλα αντικείμενα-στοιχεία της ίδιας της εφαρμογής.

3. Ο υποδοχέας μικροϋπηρεσιών, καλεί την μέθοδο service() της μικροϋπηρεσίας ως απόκριση στις αιτήσεις προς τη μικροϋπηρεσία. Η μέθοδος service(), εμπλέκει τη κατάλληλη μέθοδο HTTP για να χειριστεί την αίτηση, καλώντας ή την μέθοδο doGet(), ή τη μέθοδο doPost(). Για παράδειγμα, η μικροϋπηρεσία αποκρίνεται σε μια αίτηση HTTP Post, με την εκτέλεση της μεθόδου doPost().

4. Όταν καλούνται οι μέθοδοι doGet() και doPost(), ο υποδοχέας μικροϋπηρεσιών δημιουργεί ένα αντικείμενο javax.servlet.http.HttpServletRequest και ένα αντικείμενο HttpServletResponse και τα προωθεί ως παραμέτρους στις μεθόδους χειρισμού της αίτησης. Το αντικείμενο HttpServletRequest αντιπροσωπεύει την αίτηση προς τη μικροϋπηρεσία, ενώ το αντικείμενο HttpServletResponse, τη απόκριση της μικροϋπηρεσίας προς την αίτηση αυτή.

5. Ο υποδοχέας ιστου είναι αυτός που ελέγχει το κύκλο ζωής μιας μικροϋπηρεσίας, δηλαδή το χρόνο παραμονής ένος στιγμιοτύπου μικροϋπηρεσίας στην εικονική μηχανή της java. Όταν ο υποδοχέας ρυθμιστεί να απομακρύνει την μικροϋπηρεσία, καλεί τη μέθοδο destroy(), μέσω της οποίας απελευθερώνονται οι πόροι που είχε δεσμεύσει, όπως π.χ. τη σύνδεση με κάποια βάση δεδομένων.

9.3.5.1. - Ζητήματα Πολυνημάτωσης

Η βασική διεπαφή Servlet, ορίζει μια μέθοδο service() για το χειρισμό των αιτήσεων του πελάτη και των απαντήσεων προς αυτόν. Αυτή η μέθοδος καλείται για κάθε μια αίτηση που δέχεται ένα στιγμιότυπο κάποιας μικροϋπηρεσίας από τον υποδοχέα μικροϋπηρεσιών. Ο χειρισμός των αιτήσεων ταυτόχρονα (concurrently) σε μια εφαρμογή ιστού, απαιτεί από μεριάς του προγραμματιστή, τη σχεδίαση των μικροϋπηρεσιών με τρόπο που να καθίσταται εφικτή η εκτέλεση της μεθόδου service() μέσω πολλαπλών νημάτων την ίδια χρονική στιγμή. Γενικά ο υποδοχέας ιστού χειρίζεται τις ταυτόχρονες αιτήσεις προς την ίδια μικροϋπηρεσία, με ταυτόχρονη εκτέλεση της μεθόδου service() από διαφορετικά νήματα.

ΑΡΙΘΜΟΣ ΣΤΙΓΜΙΟΤΥΠΩΝ

Εάν μια μικροϋπηρεσία υλοποιεί τη διεπαφή **SingleThreadModel**, ο υποδοχέας θα στιγμιοτυποποιήσει πολλαπλά στιγμιότυπα για να χειριστεί μια «βαριά αίτηση» και στη συνέχεια να φορτώθούν και να ακολουθήσει σειριακή εκτέλεση των αιτήσεων σε ένα στιγμιότυπο. Εάν η μικροϋπηρεσία υλοποιεί τη διεπαφή SingleThreadModel ο υποδοχέας θα στιγμιοτυποποιήσει πολλαπλά στιγμιότυπα αυτής της μικροϋπηρεσίας σε κάθε JVM του υποδοχέα. Η χρήση της διεπαφής SingleThreadModel εξασφαλίζει πως μόνο ένα νήμα θα εκτελέσει την μέθοδο service() του στιγμιοτύπου της μικροϋπηρεσίας μια δεδομένη χρονική στιγμή.

Τα αντικείμενα τα οποία είναι προσπελάσιμα από περισσότερα του ενός στιγμιότυπα μίας μικροϋπηρεσίας, όπως τα στιγμιότυπα του HttpSession, μπορούν να είναι διαθέσιμα οποιαδήποτε χρονική στιγμή σε πολλαπλές μικροϋπηρεσίες που υλοποιούν τη διεπαφή SingleThreadModel. Φυσικά απαιτείται από μεριάς του προγραμματιστή εφαρμογών ιστού να επιλύσει τα ζητήματα που προκύπτουν λόγω υλοποίησης της διεπαφής αυτής, όπως να αποφύγει τη χρήση ενός στιγμιοτύπου μιας μεταβλητής, ή τον συγχρονισμό των μεθόδων που προσπελαύνουν κοινούς πόρους. Ένας υποδοχέας μικροϋπηρεσιών μπορεί να προωθήσει πολλαπλές αιτήσεις διαμέσου της μεθόδου service() της μικροϋπηρεσίας. Για το χειρισμό των αιτήσεων, ο προγραμματιστής πρέπει να λάβει επαρκή μέτρα για την ταυτόχρονη επεξεργασία πολλαπλών νημάτων στη μέθοδο service(). Παρ'ολαυτά, εάν υλοποιηθεί η διεπαφή SingleThreadModel ο υποδοχέας μπορεί να διασφαλίσει ότι μόνο ένα νήμα-αίτηση θα βρίσκεται μια δεδομένη χρονική στιγμή στη μέθοδο service(). Ο υποδοχέας εξασφαλίζει αυτή την απαίτηση, προωθόντας σειριακά τις αιτήσεις προς τη μικροϋπηρεσία, ή μέσω της διατήρησης μιας «πισίνας στιγμιοτύπων» της μικροϋπηρεσίας. Εάν η μικροϋπηρεσία είναι μέρος μιας εφαρμογής ιστού που έχει προσδιορίστεί ως διαμοιραζόμενη, ο υποδοχέας μπορεί να διατηρήσει μια πισίνα στιγμιοτύπων της σε κάθε JVM από όπου προσπελαύνεται η εφαρμογή.

Για τις μικροϋπηρεσίες που δεν υλοποιούν τη διεπαφή SingleThreadModel, εάν η μέθοδος service (ή οι μέθοδοι όπως η doGet, ή η doPost πού διεκπεραιώνονται στη μέθοδο service() της αφηρημένης κλάσης HttpServlet) έχει δηλωθεί ως synchronized, ο υποδοχέας δεν θα μπορεί να χρησιμοποιήσει το στιγμιότυπο της πισίνας που προαναφέρθηκε, αλλά πρέπει να προωθήσει

σειριακά τις αιτήσεις διαμέσου αυτού. Θα πρέπει οι προγραμματιστές να μην συγχρονίσουν τη μέθοδο service(), ή τις μεθόδους που διεκπεραιώνονται σε αυτή, λόγω τις δραματικής μείωσης της απόδοσης της εκτέλεσης της μικροϋπηρεσίας.

9.3.5.2. - Σφάλματα κατά των χειρισμό αιτήσεων

Μια μικροϋπηρεσία, μπορεί να προκαλέσει είτε μια εξαίρεση ServletException, είτε μια UnavailableException κατά τη εξυπηρέτηση μιας αίτησης. Μια εξαίρεση σφάλματος ServletException μας ενημερώνει ότι κάποιο σφάλμα συνέβει κατά τη διάρκεια της επεξεργασίας της αίτησης και ότι ο υποδοχέας πρέπει να λάβει τα κατάλληλα μέτρα για να «καθαρίσει» την αίτηση. Μια εξαίρεση σφάλματος UnavailableException, μας ενημερώνει ότι η μικροϋπηρεσία αδυνατεί να χειριστεί αιτήσεις, είτε προσωρινα, είτε μόνιμα. Εάν παρουσιαστεί μια μόνιμη αδυναμία εξυπηρέτησης μέσω της εξαίρεσης UnavailableException, ο υποδοχέας οφείλει να απομακρύνει την μικροϋπηρεσία από τη μέθοδο service(), να καλέσει τη μέθοδο destroy() και να απελευθερώσει το στιγμιότυπο της μικροϋπηρεσίας. Οποιοδήποτε αίτηση απορριφθεί από τον υποδοχέα, λόγω αυτής της αιτίας, επιστρέφει μια απάντηση SC_NOT_FOUND και κωδικό σφάλματος 404. Εάν παρουσιαστεί μια προσωρινή αδυναμία μέσω της εξαίρεσης UnavailableException, ο υποδοχέας μπορεί να επιλέξει να μην δρομολογήσει καμία αίτηση προς την μικροϋπηρεσία καθόλη τη διάρκεια της μη διαθεσιμότητας της. Οποιαδήποτε αίτηση αυτή τη χρονική περίοδο απορρίπτεται από τον υποδοχέα επιστρέφει μια απάντηση SC_SERVICE_UNAVAILABLE με κωδικό σφάλματος 503.

Ο υποδοχέας μπορεί να αγνοήσει οποιαδήποτε διάκριση ανάμεσα στην προσωρινή και την μόνιμη μη διαθεσιμότητα και να αντιμετωπίσει όλες τι εξαιρέσεις UnavailableException ως μόνιμη μη διαθεσιμότητα και να απομακρύνει αυτομάτως μια μικροϋπηρεσία όταν προκαλεί τέτοιες εξαιρέσεις σφάλματων.

9.3.5.3. - Ασύγχρονη Επεξεργασία

Μερικές φορές μια μικροϋπηρεσία είναι αδύνατο να ολοκληρώσει την επεξεργασία μιας αίτησης, χωρίς να περιμένει κάποιον άλλο πόρο, ή κάποιο συμβάν, να παράγει την δική του απόκριση. Για παράδειγμα μια μικροϋπηρεσία πρέπει να περιμένει, είτε για μια ενεργή σύνδεση προς κάποια βάση δεδομένων μέσω του JDBC, είτε την απόκριση κάποιας απομακρυσμένης διαδικτυακής

υπηρεσίας, είτε για κάποιο μήνυμα JMS, ή για κάποιο συμβάν κάποιας άλλης εφαρμογής, προτού τελικά παράγει την απόκριση στην αίτηση του πελάτη. Όμως η αναμονή μιας μικροϋπηρεσίας είναι ατελέσφορη ενέργεια καθώς είναι μια διαδικασία μπλοκαρίσματος που αναλώνει το νήμα και άλλους κρίσιμους και περιορισμένους πόρους. Συνήθως ένας «αργός» πόρος, όπως μια βάση δεδομένων μπορεί να έχει πολλά νήματα μπλοκαρισμένα να περιμένουν για πρόσβαση και αυτό μπορεί να προκαλέσει «παρατεταμένη στέρηση» κάποιων νημάτων και χαμηλή απόδοση και ποιότητα της υπηρεσίας ολόκληρου του υποδοχέα ιστού.

Η διεπαφή προγραμματισμού Servlet 3.0 εισήγαγε την δυνατότητα για ασύγχρονη επεξεργασία των αιτημάτων και έτσι το νήμα μορεί να επιστρέψει στον υποδοχέα και να πραγματοποιήσει και άλλες εργασίες εως ότου ολοκληρωθεί η ενέργεια – το γεγονός που περίμενε για να παράγει την απόκριση της η μικροϋπηρεσία. Όταν ξεκινάει η ασύγχρονη επεξεργασία της αίτησης, ένα άλλο νήμα μπορεί να παράγει τη τελική απάντηση-απόκριση, ή να την προωθήσει, ώστε να μπορεί να αναπαραχθεί στο περιεχόμενο του υποδοχεά χρησιμοποιώντας το αντικείμενο AsyncContext.

Η τυπική ακολουθία συμβάντων της ασύγχρονης επεξεργασίας είναι:

1. Η αίτηση παραλαμβάνεται και προωθείται διαμέσου των συνηθισμένων φίλτρων για αυθεντικοποίηση κτλ.προς τη μικροϋπηρεσία.
2. Η μικροϋπηρεσία επεξεργάζεται τις παραμέτρους και/ή το περιεχόμενο της αίτησης για να προσδιορίσει την «φύση» της.
3. Η μικροϋπηρεσία «κοινοποιεί» τις αιτήσεις για πόρους, ή δεδομένα, για παράδειγμα στέλνει μια αίτηση σε κάποια απομακρυσμένη υπηρεσία ιστού, ή κάνει join σε κάποια ουρά περιμένοντας κάποια σύνδεση μέσω JDBC σε κάποια βάση δεδομένων.
4. Η μικροϋπηρεσία επιστρέφει χωρίς να παράγει κάποια απόκριση.
5. Μετα από κάποιο χρονικό διάστημα που καταστεί διαθέσιμος ένας πόρος που έχει ζητηθεί και ήταν δεσμευμένος, το νήμα χειρίζεται το συμβάν και συνεχίζεται η επεξεργασία της αίτησης, είτε από το ίδιο νήμα, είτε προωθόντας την προς κάποιο άλλο πόρο του υποδοχέα χρησιμοποιώντας το αντικείμενο AsyncContext.

Όταν η τιμή της παράμετρου asyncSupported γίνει true, η εφαρμογή μπορεί να ξεκινήσει την ασύγχρονη επεξεργασία με ένα ξεχωριστό νήμα, καλώντας την μέθοδο startAsync, περνώντας της μια αναφορά στα αντικείμενα της αίτησης και απάντησης και τέλος βγαίνοντας από τον υποδοχέα του αρχικού νήματος. Αυτό σημαίνει πως η απάντηση θα διατρέξει, ανάποδα αυτή τη φορά τα ίδια φίλτρα που διέτρεξε και κατά τη διαδικασία της αρχικής εισόδου της.

Η προώθηση μιας αίτησης από μια μικροϋπηρεσία που έχει την παράμετρο asyncSupported=true, σε μια άλλη μικροϋπηρεσία που έχει την παράμετρο asyncSupported=false, είναι αποδεκτή και επιτρέπεται. Σε αυτή τη περίπτωση η απάντηση θα υπάρξει μονάχα όταν η μέθοδος service() της μικροϋπηρεσίας που δεν υποστηρίζει την ασύγχρονη επεξεργασία, τερματίσει και είναι ευθύνη του υποδοχέα να αποδεσμεύσει το AsyncContext, ώστε οποιαδήποτε στιγμιότυπα του AsycListener να ενημερωθούν.

Η προώθηση μιας αίτησης από μια μικροϋπηρεσία που λειτουργεί σύγχρονα σε μια μικροϋπηρεσία που λειτουργεί ασύγχρονα δεν είναι αποδεκτή και έγκυρη. Παρολαυτά η απόφαση για το εάν θα συμβεί μια εξαίρεση σφάλματος IllegalStateException, εξαρτάται από τη χρονική στιγμή που η εφαρμογή καλεί την μέθοδο startAsync. Αυτό μπορεί να επιτρέψει σε μια εφαρμογή να συμπεριφερθεί είτε με σύγχρονο τρόπο, είτε με ασύγχρονο.

9.3.6. - Ο τερματισμός της μικροϋπηρεσίας

Ο υποδοχέας μικροϋπηρεσίων δεν απαιτείται να κρατάει φορτωμένη μια μικροϋπηρεσία για μια μακρά χρονική περίοδο. Ένα στιγμιότυπο μιας μικροϋπηρεσίας, μπορεί να παραμείνει ενεργό μέσα στον υποδοχέα μικροϋπηρεσιών για μια περίοδο μόνο μερικών χιλιοστών του δευτερολέπτου μέσα στον κύκλο ζωής του υποδοχέα (που μπορεί να είναι ημέρες, μήνες ή ακόμα και χρόνια). Όταν ο υποδοχέας των μικροϋπηρεσιών αποφασίσει να απομακρύνει μια μικροϋπηρεσία, καλέι τη μέθοδο destroy() της διεπαφής servlet, για να επιτρέψει στην μικροϋπηρεσία να απελευθερώσει οποιουσδήποτε πόρους χρησιμοποίησε και ενδεχομένως να αποθηκεύσει κάποια κατάσταση. Για παράδειγμα, ο υποδοχέας μπορεί να πραγματοποιήσει μια τέτοια ενέργεια, όταν θελήσει να διατηρήσει τους πόρους της μνήμης, ή όταν τερματίζεται.

Προτού ο υποδεχέας μικροϋπηρεσιών καλέσει τη μέθοδο destroy(), πρέπει να επιτρέψει σε οποιοδήποτε νήμα τρέχει εκείνη τη στιγμή να ολοκληρώσει την εκτέλεση της μεθόδου service() της μικροϋπηρεσίας, ή να υπερβεί το χρονικό όριο εκτέλεσης που έχει οριστεί από τη μικροϋπηρεσία. Άπαξ και κληθεί η μέθοδος destroy() ενός στιγμιοτύπου της μικροϋπηρεσίας, ο υποδοχέας δεν θα κατευθύνει άλλες αιτήσεις προς αυτό το στιγμιότυπο της μικροϋπηρεσίας. Εάν ξανα απαιτηθεί ο υποδεχέας να ενεργοποιήσει μια μικροϋπηρεσία, αυτό θα πραγματοποιηθεί από ένα νέο στιγμιότυπο της κλάσης της μικροϋπηρεσίας. Αφού ολοκληρωθεί η εκτέλεση της μεθόδου destroy(), ο υποδοχέας απελευθερώνει το στιγμιότυπο της μικροϋπηρεσίας και πλεον αυτό καθίσταται έτοιμο για την παραλαβή του από τη διαδικασία «περισσυλογής απορριμάτων» (garbage collection)

9.3.7. - Οι web εφαρμογές

Προτού συνεχίσουμε με το πώς οργανώνεται η δομή των καταλόγων είναι χρήσιμο να μελετήσουμε την run-time οργάνωση μιας εφαρμογής ιστού. Μια εφαρμογή ιστού ορίζεται ως μια ιεραρχία καταλόγων και αρχείων. Αυτή η ιεραρχική δομή καταλόγων μπορεί να οργανωθεί είτε ως ένα αρχείο WAR (Web ARchive), είτε ως έχει. Η δεύτερη μορφή είναι πιο εύχρηστη κατά την διαδικασία της ανάπτυξης της εφαρμογής, ενώ η πρώτη είναι πιο χρήσιμη όταν θελήσουμε να διανείμουμε την εφαρμογή μας σε τρίτους.

Ο πρώτος κατάλογος στην ιεραρχία της εφαρμογής ιστού, είναι και ο κατάλογος ρίζα των εγγράφων της εφαρμογής. Εδώ τοποθετούνται τα αρχεία HTML και οι JSP σελίδες της εφαρμογής μας που απαρτίζουν το γραφικό περιβάλλον αλληλεπίδρασης με το χρήστη.

Όταν η εφαρμογή αναπτύσσεται στο σύστημα αρχείων ενός διακομιστή εφαρμογών, ο διαχειριστής ορίζει τη διαδρομή περιεχομένου για αυτή την εφαρμογή. Οπότε εάν ο διαχειριστής ορίσει για την εφαρμογή αυτή τη διαδρομή /catalog, εάν πραγματοποιηθεί μια αίτηση URI η οποία αναφέρεται στο /catalog/index.html, το αρχείο index.html θα ανακτηθεί από τον κατάλογο ρίζα των εγγράφων.

Για τη διευκόλυνση της δημιουργίας μιας εφαρμογής ιστού στην κατάλληλη μορφή, είναι προσφορό να οργανώσουμε όλα τα «εκτελέσιμα αρχεία» της εφαρμογής μας (πρόκειται για τα αρχεία που συνήθως χρησιμοποιεί ο διακομιστής

Tomcat, όταν εκτελεί την εφαρμογή μας) στο ίδιο WAR αρχείο. Για να το πράξουμε αυτό, στο τέλος θα πρέπει να έχουμε τα ακόλουθα στοιχεία στον κατάλογο ρίζα της εφαρμογής μας:

/WEB-INF/web.xml – Ο περιγραφέας ανάπτυξης της εφαρμογής ιστού. Πρόκειται για ένα αρχείο XML που περιγράφει τα servlets και τα άλλα συστατικά (components) που αποτελούν την εφαρμογή ιστού μας, μαζί με τις όποιες παραμέτρους αρχικοποίησης τους, αλλά και τους περιορισμούς ασφάλειας για τον υποδοχέα ιστού που θέλουμε να παρέχει ο διακομιστής για εμάς.

/WEB-INF/classes/ - Αυτός ο κατάλογος, περιέχει όλα τα εκτελέσιμα αρχεία (.class) που συνθέτουν την εφαρμογή μας, περιλαμβάνοντας και τα servlet και τα μή-servlet αρχεία class, που δεν έχουν ομαδοποιήθει σε αρχεία jar (java archive). Εάν τα αρχεία class είναι οργανωμένα σε πακέτα, θα πρέπει να τα τοποθετήσουμε στην ιεραρχία κατάλογο κάτω από την δομή /WEB-INF/classes/. Για παράδειγμα, ένα αρχείο class που λέγεται com.mycompany.mypackage.MyServlet, θα πρέπει να αποθηκευτεί σε ένα αρχείο που λέγεται /WEB INF/classes/com/mycompany/mypackage/MyServlet.class.

/WEB-INF/lib/ - Αυτός ο κατάλογος περιέχει όλα τα αρχεία jar που αποτελούν τις βιβλιοθήκες κλάσεων και τους οδηγούς JDBC για την σύνδεση στην βάση δεδομένων που χρησιμοποιείται και που απαιτούνται για την λειτουργία της εφαρμογής.

Όταν εγκαθιστούμε μια εφαρμογή στον διακομιστή Tomcat, οι κλάσεις του καταλόγου WEB-INF/classes/, όπως και όλες οι κλάσεις των αρχείων JAR που βρίσκονται στον κατάλογο WEB-INF/lib/, καθίστανται ορατές και στις υπόλοιπες κλάσεις της εφαρμογής. Οπότε εάν εισάγουμε όλες τις απαιτούμενες κλάσεις βιβλιοθηκών σε ένα από αυτά τα μέρη, θα απλοποιήσουμε την εγκατάσταση της εφαρμογής μας και δεν θα απαιτείται η τοποθέτηση των εξωτερικών κλάσεων στον κατάλογο εξωτερικών βιβλιοθηκών του java runtime environment του διακομιστή που θα τρέχει η εφαρμογή ιστού.

Όπως οι περισσότεροι υποδοχείς μικροϋπηρεσιών, ο διακομιστής Tomcat 6, υποστηρίζει επίσης μηχανισμούς για την εγκατάσταση των εξωτερικών βιβλιοθηκών, ή των απλών κλάσεων, ούτως ώστε να είναι ορατές σε όλες τις εγκατεστημένες εφαρμογές ιστού, χωρίς να απαιτείται κάθε φορά να

ενσωματόνονται σε κάθε εφαρμογή ιστού. Η περιοχή που συνήθως χρησιμοποιείται μέσα σον διακομιστή Tomcat 6 για τον διαμοιραζόμενο κώδικα, είναι ο κατάλογος $CATALINA_HOME/lib. Τα αρχεία JAR που τοποθετούνται εδώ πέρα είναι ορατά από όλες τις εφαρμογές ιστού και τον κώδικα του ίδιου του Tomcat. Μια καλή πρακτική είναι η τοποθέτηση των οδηγών JDBC, που απαιτούνται για τη σύνδεση της εφαρμογής ιστού με κάποια βάση δεδομένων, μέσα σε αυτό τον κατάλογο. Η βασική έκδοση εγκατάστασης του Tomcat 6 περιέχει ένα σύνολο προ-εγκατεστημένων διαμοιραζόμενων αρχείων βιβλιοθηκών που περιλαμβάνουν:

- Τη διεπαφή προγραμματισμού Servlet 2.5 και τη διεπαφή προγραμματισμού JSP 2.1 τα οποία είναι θεμελιώδη για τη δημιουργία των μικροϋπηρεσιών και των σελίδων JSP.
- Έναν αναλυτή XML συγγενικό της διεπαφής προγραμματισμού JAXP (έκδοση 1.2), οπότε η εφαρμογή μας μπορεί να επιτελέσει επεξεργασίες βασισμένες στο μοντέλο DOM, ή επεξεργασίες βασισμένες στο SAX των XML εγγράφων.

Το αρχείο /META-INF/context.xml μπορεί να χρησιμοποιηθεί για να ορίσει συγκεκριμένες παραμέτρους-επιλογές για τον διακομιστή Tomcat, όπως loggers, data sources, παραμετροποίηση του διαχειριστή συνδεριών και άλλα.

Προκειμένου να εκτελεστεί μια εφαρμογή ιστού, πρέπει να γίνει deployed σε έναν υποδοχέα μικροϋπηρεσιών. Μια εφαρμογή ιστού μπορεί να γίνει deployed στον Tomcat με έναν από τους παρακάτω τρόπους:

1ος Τρόπος: Αντιγράφουμε την ιεραρχία καταλόγων σε ένα υποκατάλογο του καταλόγου $CATALINA_BASE/webapps/. Ο Tomcat θα προσθέσει ένα μονοπάτι περιεχομένου (context path) για την εφαρμογή μας, που βασίζεται στο όνομα του υποκαταλόγου που έχουμε διαλέξει. Θα χρησιμοποιήσουμε αυτή την τακτική στο αρχείο build.xml που θα δημιουργήσουμε γιατί είναι ο πιο γρήγορος και εύκολος τρόπος κατά τη διαδικασία της ανάπτυξης. Οφείλουμε βέβαια πάντα να επανεκκινήσουμε το διακομιστή εφαρμογών ιστού Tomcat, μετά την εγκατάσταση, ή την ενημέρωση μιας εφαρμογής μας προκειμένου να λάβουν χώρα οι αλλάγες.

2ος Τρόπος: Αντιγράφουμε το αρχείο της εφαρμογής ιστου μας (WAR) στον κατάλογο $CATALINA_BASE/webapps/. Όταν εκκινηθεί ο Tomcat, αυτόματα το

αρχείο war θα ανοίξει και θα βρεθεί στην μη πακεταρισμένη του μορφή και θα εκτελεστεί. Αυτή η τακτική χρησιμοποιείται για την εγκατάσταση εφαρμογών όταν χρησιμοποιήθηκε περιβάλλον ανάπτυξης λογισμικού άλλου κατασκευαστή, ή ακομά και δικό μας, σε μία εγκατεστημένη έκδοση του Tomcat. Εάν χρησιμοποιήσουμε αυτή την τακτική και αργότερα θελήσουμε να ενημερώσουμε την εφαρμογή μας, πρέπει να αντικαταστήσουμε και το αρχείο war και να διαγράψουμε τον κατάλογο που δημιούργησε ο Tomcat και έπειτα να επανεκκινήσουμε τον διακομιστή για να εφαρμοστούν οι αλλαγές μας.

9.4. - JavaScript Client/Side Scripting Language

Η γλώσσα προγραμματισμού JavaScript (πρώην LiveScript της NetScape), είναι μια αντικειμενοστρεφής γλώσσα scripting για την δημιουργία σεναρίων μέσα σε αρχεία html, είτε JSP που θα διερμηνευτούν και θα εκτελεστούν στην πλευρά (στον υπολογιστή) του πελάτη και όχι στην πλευρα του διακομιστή εφαρμογών. Η διαδικτυακή εφαρμογή περιλαμβάνει τα σενάρια javascript είτε μέσα στα ίδια τα html και τα JSP αρχεία της, είτε σε ξεχωριστό αρχείο με κατάληξη js και διερμηνεύονται κατευθείαν από το πρόγραμμα περιήγησης ιστού του πελάτη, μόλις το περιεχόμενο της σελίδας φορτώθει σε αυτον. Με τα σεναρία της javascript, μπορούμε να προσπελάσουμε και να χειριστούμε όλα τα αντικείμενα του Domain Object Model, όλα τα στοιχεία-ετικέτες της HTML και να παράγουμε δυναμικό περιεχόμενο, από την μεριά του πελάτη, μέσα στις ιστοσελίδες και τις σελίδες JSP της διαδικτυακής εφαρμογής μας. Οι συναρτήσεις που θα δημιουργήσουμε, δεν χρειάζονται μεταγλώττιση, παρά μόνο, είτε να ενσωματώσουμε μια ετικετα με την σήμανση <script src=”όναμα αρχείου.js”> μέσα στα HTML και στα JSP αρχεία της εφαρμογής μας, εάν έχουμε βάλει τις συναρτήσεις μας σε ξεχωριστό αρχείο, είτε να ενσωματώσουμε μια ετικέτα με τη σήμανση <script language=”javascript”> στις σελίδες που έχουμε ενσωματώσει τις συναρτήσεις μας.

9.5. - Εξωτερικές βιβλιοθήκες ανοιχτού κώδικα (Open Source) Java

Οι εξωτερικές βιβλιοθληκες ανοιχτού κώδικα που χρησιμοποιεί η εφαρμογή μας, ενσωματώνονται στο τελικό εκτελέσιμο αρχείο της (.jar) και έτσι δεν απαιτείται από μεριάς του χρήστη της εφαρμογής να τις εισάγει στον κατάλογο εξωτερικών βιβλιοθηκών του περιβάλλοντος της εικονικής μηχανής της java.

9.5.1. - servlet.jar

Η προδιαγραφή προγραμματισμού μικροϋπηρεσιών Java Servlet API, ορίζει την διεπαφή Servlet την οποία πρέπει να υλοποιήσουν όλες οι μικροϋπηρεσίες, είτε απευθείας, είτε κληρονομώντας κάποια από τις κλάσεις GenericServlet και HttpServlet που υλοποιούν την διεπαφή Servlet. Η διεπαφή αυτή, ορίζει την μέθοδο service, η οποία είναι υπεύθυνη για τον χειρισμό των αιτήσεων http προς την μικροϋπηρεσία. Αυτή η μέθοδος καλείται για κάθε αίτηση που δέχεται ο υποδοχέας για ένα στιγμιότυπο κάποιας μικροϋπηρεσίας. Η ββιβλιοθήκη αποτελείται από το αρχείο **servlet.jar** το οποίο πρέπει να τοποθετηθεί στον κατάλογο των εξωτερικών βιβλιοθηκών της java, κάτω από τον κατάλογο **C:\Program Files\Java\ jdk1.6.0_xx\jre\lib\ext**, είτε στον κατάλογο **WEB-INF\lib** της εφαρμογής μας

ΚΕΦΑΛΑΙΟ 10
Η υλοποίηση της εφαρμογής

10.1. - Περίληψη

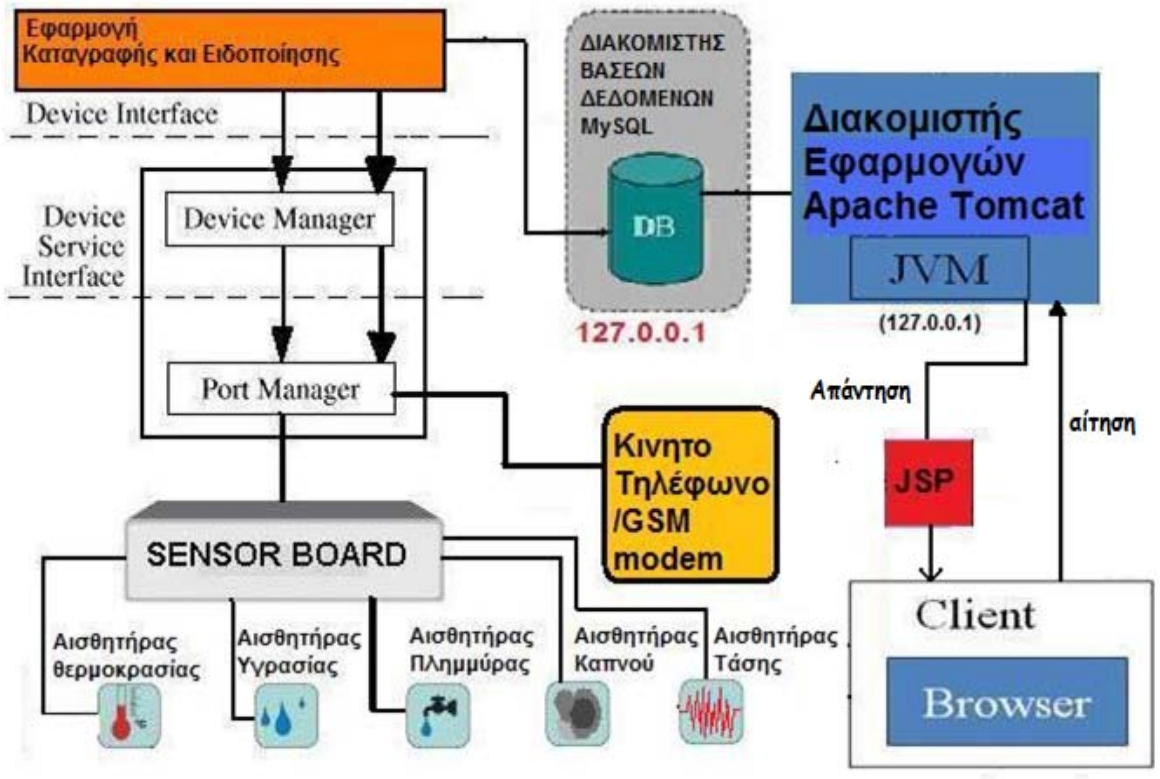
Η Εφαρμογή μας αποτελείται από δυο πακέτα: το web και το Utilities. Το πρώτο περιλαμβάνει τις μικροϋπηρεσίες που υποστηρίζουν την σύνδεση στην εφαρμογή, τη δυναμική αναπαράσταση των καταγεγραμμένων μετρήσεων από τους αισθητήρες σε πραγματικό χρόνο, την παρουσίαση στατιστικών διαγραμμάτων για τα δεδομένα κάθε αναλογικού αισθητήρα, τις τιμές όρια για κάθε αισθητήρα, τη διαμόρφωση των μηνυμάτων που θα λάβει ο διαχειριστής εάν ξεπεραστούν αυτά τα όρια, την αλλαγή των μηνυμάτων περιγραφής του κάθε αισθητήρα, την αλλαγή του ονόματος και του κωδικού με τον οποίο εισέρχεται στην δικτυακή εφαρμογή, αλλά και την αλλαγή του κινητού τηλεφώνου και του λογαριασμού ηλεκτρονικής αλληλογραφίας που θα ειδοποιήθει σε κατάσταση κινδύνου και τέλος την αποσύνδεση από την εφαρμογή.

Το δευτερο πακέτο περιλαμβάνει τις βοηθητικές κλάσεις της κύριας εφαρμογής: την κλάση που υλοποιεί τις εργασίες από και προς την βάση δεδομένων και την κλάση που υλοποιεί την κατασκεύη όλων των διαγραμμάτων της εφαρμογής, αυτών που αφορούν στην αναπαράσταση των τιμών των αισθητήρων σε πραγματικό χρόνο, αλλά και αυτών που αφορούν στην προβολή στατιστικών διαγραμμάτων των μετρήσεων για την τρέχουσα ημέρα, εβδομάδα, μήνα και έτος.

Η εφαρμογή μας, αποτελείται από τις μικροϋπηρεσίες που χειρίζονται τις αιτήσεις http και τις σελίδες JSP, οι οποίες χρησιμοποιούνται για την αλληλεπίδραση με τον client και την παρουσίαση των αποτελεσμάτων. Η αρχιτεκτονική που επιλέξαμε για την δημιουργία της διαδικτυακής εφαρμογής μας, βασίζεται στο μοντέλο εκείνο, όπου διαχωρίζεται το κομμάτι της επεξεργασίας των

δεδομένων, από αυτό της παρουσίασης τους στον χρήστη. Κάθε μικροϋπηρεσία λειτουργεί ως ένας «ελεγκτής», handler, που διεκπεραιώνει τις επεξεργασίες των αιτήσεων και αλληλεπιδρά με τις κατάλληλες σελίδες JSP, είτε για να πάρει παραμέτρους από τον χρήστη, είτε για να του παρουσιάσει αποτελέσματα. Έτσι, αντί η μικροϋπηρεσία να σηκώνει όλο το βάρος της επεξεργασίας αλλά και εκείνο της δυναμικής παρουσίασης των δεδομένων με δυναμικό html περιεχόμενο, στην δική μας υλοποίηση, η επεξεργασία γίνεται από την μικροϋπηρεσία και η παρουσίαση γινεται απτις σελίδες jsp οι οποίες δέχονται τις αιτήσεις με τις κατάλληλες παραμέτρους για να παρουσίασουν τα δεδομένα. Για κάθε μικροϋπηρεσία της εφαρμογής μας, υπάρχει και μια σελίδα JSP. Αντί η κάθε μικροϋπηρεσία να σηκώνει όλο το φόρτο εργασίας και να παράγει δυναμικά αποτελέσματα ως html περιεχόμενο, χρησιμοποιείται μια σελίδα JSP για την απόκριση σε κάθε αίτησης προς την εφαρμογή.

Στο *Σχήμα 10.1* βλέπουμε πλέον ολοκληρωμένο το Σύστημα Επιτήρησης Περιβαλλοντολλογικών συνθηκών, με την προσθήκη και της διαδικτυακής εφαρμογής να αλληλεπιδρά με τα υπόλοιπα μέρη του συστήματος.



Σχήμα 10.1 - Το Σύστημα Επιτήρησης Περιβαλλοντολογικών Συνθηκών

Αναλυτικά οι μικροϋπηρεσίες της εφαρμογής μας, είναι οι παρακάτω:

- Η **home** η οποία χρησιμοποιείται για την σύνδεση του διαχειριστή στην εφαρμογή και την ανακατεύθυνση του προς τη σελίδα central.jsp
- Η **livepresentation** η οποία χρησιμοποιείται για την δημιουργία δυναμικής παρουσίασης των δεδομένων καταγραφής των αισθητήρων σε πραγματικό χρόνο και για να τα προβάλλει στην σελίδα diagrams.jsp
- Η **history** η οποία χρησιμοποιείται για να δημιουργήσει δυναμικά, «εν πτήση», ημερίσια, εβδομαδιαία, μηνιαία και ετήσια στατιστικά διαγράμματα των μετρήσεων των τεσσάρων αισθητήρων και για να προβάλλει τα αποτελέσματα στη σελίδα statisticsdiagrams.jsp
- Η **adminsetup** η οποία χρησιμοποιείται για να χειριστεί τις αλλαγές του διαχειριστή που αφορούν στο λογαριασμό σύνδεσης στην εφαρμογή, στο κινητό και τον λογαριασμό ηλεκτρονικής αλληλογραφίας που θα ειδοποιηθεί σε περίπτωση κινδύνου στο δωμάτιο υπολογιστών.
- Η **mailsetup** η οποία χρησιμοποιείται για να χειριστεί τις αλλαγές του διαχειριστή που αφορούν στον λογαριασμό ηλεκτρονικής αλληλογραφίας του συστήματος επιτήρησης περιβαλλοντολογικών συνθηκών. Ο λογαριασμός αυτός χρησιμοποιείται για την αποστολή ηλεκτρονικού μηνύματος ειδοποίησης στον λογαριασμό ηλεκτρονικής αλληλογραφίας του διαχειριστή της εφαρμογής.
- Η **sensorsSetup** η οποία χρησιμοποιείται για να χειριστεί τις αλλαγές του διαχειριστή που αφορούν στις τιμές-όρια για τον καθένα από τους αναλογικούς αισθητήρες, στα μηνύματα ειδοποίησης για τα γραπτά και ηλεκτρονικά μηνύματα ειδοποίησης, αλλά και στα μηνύματα περιγραφής τοποθεσίας για όλους τους αισθητήρες, αναλογικούς και ψηφιακούς.
- Η **logout** η οποία χρησιμοποιείται για να χειριστεί την διαδικασία αποσύνδεσης του διαχειριστή από την διαδικτυακή εφαρμογή μας. Η μικροϋπηρεσία αυτή, τερματίζει την τρέχουσα συνεδρία http και στέλνει την αίτηση προς την αρχική μικροϋπηρεσία.

Οί βοηθητικές κλάσεις που χρησιμοποιούν οι περισσότερες μικροϋπηρεσίες, είναι οι:

- Η **DatabaseUtility** η οποία χρησιμοποιείται ως ξεχωριστό «εργαλείο» για την πραγματοποιήση ενεργειών από και προς τη βάση δεδομένων. Μέσω αυτής

της κλάσης, η εφαρμογή μας συνδέεται/αποσυνδέεται με/από τον διακομιστή βάσεων δεδομένων, εισάγει δεδομένα σε πίνακες και διαβάζει δεδομένα από αυτούς.

- Η **MyJfreeChartUtility** η οποία χρησιμοποιείται ως ξεχωριστό «εργαλείο» για την κατασκευή όλων των διαγραμμάτων που υποστηρίζει η εφαρμογή μας: αυτά που αφορούν στην παρουσίαση σε πραγματικό χρόνο των καταγεγραμμένων τιμών των αναλογικών αισθητήρων, αλλά και όλων των στατιστικών διαγραμμάτων για τη τρέχουσα ημέρα/εβδομάδα/μήνα/έτος.

Αναλυτικά οι σελίδες JSP της εφαρμογής μας, είναι οι παρακάτω:

⇒ Η **Index.jsp** η οποία αποτελεί την αρχική σελίδα της εφαρμογής μας. Μέσω αυτής της σελίδας ο διαχειριστής αποστέλλει το όνομα χρήστη και τον κωδικό του προς την μικροϋπηρεσία home η οποία, εάν τον αυθεντικοποιήσει επιτυχώς, τον ανακατευθύνει στην σελίδα central.jsp.

⇒ Η **central.jsp** η οποία αποτελεί την κεντρική σελίδα πλοήγησης της εφαρμογής που προβάλλεται μόνο εφόσον πιστοποιηθεί η ταυτότητα του χρήστη της εφαρμογής. Από αυτή τη σελίδα ο διαχειριστής μπορει να κατευθυνθεί προς οποιαδήποτε άλλη σελίδα της εφαρμογής.

⇒ Η **menu.jsp** η οποία αποτελέι το μενού πλοήγησης της εφαρμογής και που συμπεριλαμβάνεται σε όλες τις άλλες σελίδες JSP της εφαρμογής, εκτός της index.jsp.

⇒ Η **diagrams.jsp** η οποία αποτελεί την σελίδα που προβάλλονται τα δυναμικά διαγράμματα αναπαράστασης των μετρήσεων σε πραγματικό χρόνο.

⇒ Η **statisticsdiagrams.jsp** η οποία αποτελεί την σελίδα που προβάλλονται τα στατιστικά διαγράμματα των μετρήσεων είτε για την τρέχουσα ημέρα, είτε για την τρέχουσα εβδομάδα, είτε για τον τρέχον μήνα, είτε τέλος για το τρέχον έτος.

⇒ Η **adminIdentity.jsp** η οποία αποτελεί την σελίδα διαμέσου της οποίας ο διαχειριστής δύναται να αλλάξει την ταυτότητα σύνδεσης στην εφαρμογή, τον αριθμό του κινητού τηλεφώνου αλλά και τον λογαριασμό ηλεκτρονικού ταχυδρομείου, στα οποία και θα ειδοποιηθεί όταν ξεπεραστούν τα όρια που ο ίδιος έχει θέσει για τις τιμές κάθε αισθητήρα.

⇒ Η **mailIdentity.jsp** η οποία αποτελεί την σελίδα διαμέσου της οποίας ο διαχειριστής δύναται να τροποποιήσει τον λογαριασμό ηλεκτρονικής αλληλογραφίας που χρησιμοποιεί η εφαρμογή καταγραφής για να αποστείλλει μήνυμα στο ηλεκτρονικό ταχυδρομείο του, όταν κάποιος από τους αναλογικούς αισθητήρες ξεπεράσει το όριο τιμή του, αλλά και εάν κάποιος από τους ψηφιακούς αισθητήρες στείλει τιμή αλήθειας (true).

⇒ Η **sensorsIdentity.jsp** η οποία αποτελεί την σελίδα διαμέσου της οποίας ο διαχειριστής δύναται να αλλάξει τα όρια τιμές για κάθε έναν από τους τέσσερις αναλογικούς αισθητήρες, να τροποποιήσει τα μηνύματα ειδοποίησης που θα λάβει στο κινητό του και στο ηλεκτρονικό του ταχυδρομείου, όταν και εάν ξεπεραστούν τα όρια που έχει θέσει, αλλά και επίσης να τροποποιήσει το μήνυμα περιγραφής για κάθε έναν από τους αναλογικούς και ψηφιακούς αισθητήρες.

⇒ Η **messages.jsp** η οποία αποτελεί την σελίδα στην οποία ανακατευθύνονται τα μηνύματα, είτε σφάλματος είτε επιτυχίας, για κάθε μια από τις διαργασίες της εφαρμογής μας.

⇒ Η **log.jsp** η οποία αποτελεί την σελίδα στην οποία παρουσιάζονται όλες οι ενέργειες που εκτελούνται από την εφαρμογή, όλα οι τύποι γεγονότων καταγράφονται με βάση την ώρα που σενέβησαν, αλλά και με την διεύθυνση πρωτοκόλλου διαδικτύου του υπολογιστή από τον οποίο επιτελέστηκαν οι ενέργειες και τα συμβάντα αυτά.

10.2. - Η μικροϋπηρεσία home

```
package web;
import Utilities.DatabaseUtility;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.RequestDispatcher;
import java.io.*;
public class home extends HttpServlet {
```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
javax.servlet.http.HttpSession session = request.getSession(false);
String parametros = null;
```

εάν ο χρήστης προσπαθήσει να προσπελάσει την μικροϋπηρεσία χωρίς να έχει ανακατευθυνθεί από την κύρια διαδικασία εισόδου στην εφαρμογή,τότε επανακατευθύνεται στην κύρια σελίδα εισόδου index.jsp.

```
if(session==null) {
        response.sendRedirect("./index.jsp");
}
else {
        try{
                parametros = session.getValue("elegxos").toString();
        }
        catch(Exception e){
                response.sendRedirect("./index.jsp");
        }
if(parametros!=null) {
        if(parametros.equals("true")) {
                response.sendRedirect("./central.jsp");
        }
        else {
                response.sendRedirect("./index.jsp");
        }
}
}
}
```

η μέθοδος αυτή εκτελείται όταν ο χρήστης κάνει submit στη φόρμα που αποστέλλει τα στοιχεία της σε αυτή τη μικροϋπηρεσία

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
```

```
try {
javax.servlet.http.HttpSession session = request.getSession();
session.putValue("elegxos","true");
```
//παίρνουμε το όνομα χρήστη και τον κωδικό που μας αποστέλλονται από τη φόρμα που
//συμπλήρωσε ο χρήστης και μας απέστειλλε
```
String txtfld1 =(String)request.getParameter("txtfld1");
String txtfld2 =(String)request.getParameter("txtfld2");
boolean checkit = true;
DatabaseUtility.ConnectToDataBaseServer(
"com.mysql.jdbc.Driver","jdbc:mysql://localhost:3306/sensingsystem","root","10022
910");
checkit = DatabaseUtility.Validation(txtfld1,txtfld2);
```
Εάν το όνομα χρήστη και ο κωδικός ταιριάζουν με τα στοιχεία του λογαριασμό που είναι αποθηκευμένα, προχωρά η διαδικασία κανονικά
```
if(checkit) {
```
Γίνεται καταγραφή της επιτχής εισόδου της εφαρμογής στον πίνακα συμβάντων
```
DatabaseUtility.writeToLogTable("USER                    LOG-IN


SUCCESFULLY",request.getRemoteAddr() );
DatabaseUtility.terminateConnection();
RequestDispatcher diekpairewths =
getServletContext().getRequestDispatcher("/central.jsp");
//Ανακατεύθυνση της αίτησης http προς τη σελίδα central.jsp
diekpairewths.forward(request, response);
}
else {
```

Το όνομα χρήστη, ή/και ο Κωδικός χρήστη δεν είναι σωστά,οπότε και γίνεται καταγραφή της αποτυχημένης απόπειρας,στον πίνακα συμβάντων της εφαρμογής μας

```
DatabaseUtility.writeToLogTable(
"USERNAME & PASSWORD ARE NOT CORRECT!", request.getRemoteAddr() );
DatabaseUtility.terminateConnection();
request.setAttribute("msg", "USERNAME OR PASSWORD ARE NOT CORRECT");
request.setAttribute("topothesia","./index.jsp");
```

ανακατευθύνουμε την αίτηση με το κατάλληλο μήνυμα προς τη γενική σελίδα μηνυμάτων επιτυχίας και αποτυχίας

```
RequestDispatcher diekpairewths =
getServletContext().getRequestDispatcher("/messages.jsp");
diekpairewths.forward(request, response);
}
}
finally { }
}
}//τελος μικροϋπηρεσίας HOME
```

10.3. - Η μικροϋπηρεσία livepresentation

```
package web;
import Utilities.DatabaseUtility;
import java.awt.Color;
import java.awt.Font;
import java.awt.Paint;
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletContext;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.ThermometerPlot;
import org.jfree.data.general.DefaultValueDataset;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.axis.ValueAxis;
import org.jfree.chart.plot.MeterInterval;
import org.jfree.chart.plot.MeterPlot;
import org.jfree.chart.title.TextTitle;
import org.jfree.data.Range;
import org.jfree.data.general.ValueDataset;
/**
* @author ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ
*/
public class livepresentation extends HttpServlet {
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
javax.servlet.http.HttpSession session = request.getSession(false);
String parametros = null;
if(session==null) {
        response.sendRedirect("./index.jsp");
}
else {
        try{
                parametros = session.getValue("elegxos").toString();
        }
        catch(Exception e){
                response.sendRedirect("./index.jsp");
        }
if(parametros!=null) {
        if(!parametros.equals("true")) {
                response.sendRedirect("./index.jsp");
```

```
        }
        else {
                try {
//Κατασκευάζουμε τα διαγράμματα για τους 4 αναλογικούς αισθητήρες
JFreeChart jfc1 = getThermometer("AN1","TEMPERATURE SENSOR 1");
JFreeChart jfc2 = getThermometer("AN2","TEMPERATURE SENSOR 2");
JFreeChart jfc3 = getHumidityMeter("AN3","HUMIDITY SENSOR 3");
JFreeChart jfc4 = getVoltageMeter("AN4","VOLTAGE SENSOR 4");
ServletContext servletcontext = getServletContext();
String path = servletcontext.getRealPath("/");
//ΑΠΟΘΗΚΕΥΟΥΜΕ ΤΑ ΔΙΑΓΡΑΜΜΑΤΑ ΩΣ ΑΡΧΕΙΑ ΜΟΡΦΗΣ PNG
ChartUtilities.saveChartAsPNG(new
File(path+"images/livediagrams/LiveAnalogSensor1.png"),jfc1,450,400);
ChartUtilities.saveChartAsPNG(new
File(path+"images/livediagrams/LiveAnalogSensor2.png"),jfc2,450,400);
ChartUtilities.saveChartAsPNG(new
File(path+"images/livediagrams/LiveAnalogSensor3.png"),jfc3,450,400);
ChartUtilities.saveChartAsPNG(new
File(path+"images/livediagrams/LiveAnalogSensor4.png"),jfc4,450,400);
DatabaseUtility.ConnectToDataBaseServer(
"com.mysql.jdbc.Driver","jdbc:mysql://localhost:3306/sensingsystem","root","10022
910");
String measurementtime =DatabaseUtility.processSQL(
"SELECT max(measurementtime) from
analogmeasurement;",1).toString();
List paths = new ArrayList();
paths.add("images/livediagrams/LiveAnalogSensor1.png");
paths.add("images/livediagrams/LiveAnalogSensor2.png");
paths.add("images/livediagrams/LiveAnalogSensor3.png");
paths.add("images/livediagrams/LiveAnalogSensor4.png");
paths.add(measurementtime);
```

ΕΙΣΑΓΟΥΜΕ ΤΙΣ ΕΙΚΟΝΕΣ ΣΑΝ ΙΔΙΟΤΗΤΕΣ ΤΗΣ ΙΔΙΑΣ ΤΗΣ ΑΙΤΗΣΗΣ ΚΑΙ ΤΗΝ ΑΝΑΚΑΤΕΥΘΥΝΟΥΜΕ ΣΤΗΝ ΣΕΛΙΔΑ JSP ΠΟΥ ΘΑ ΠΡΟΒΛΗΘΟΥΝ ΑΥΤΕΣ ΟΙ ΕΙΚΟΝΕΣ

```
request.setAttribute("lista", paths);
RequestDispatcher diekpairewths =
servletcontext.getRequestDispatcher("/diagrams.jsp");
diekpairewths.forward(request, response);
}
catch(Exception eksaireshsfalmatos){}
}
}
}
}
}
```

10.4. - Η μικροϋπηρεσία history

```
package web;
import Utilities.MyJfreeChartUtility;
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.jfree.chart.JFreeChart;
import java.io.File;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletContext;
import org.jfree.chart.ChartUtilities;
public class history extends HttpServlet {
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException {
javax.servlet.http.HttpSession session = request.getSession(false);
String parametros = null;
```

```
if(session==null) {
	response.sendRedirect("./index.jsp");
}
else {
	try{
		parametros = session.getValue("elegxos").toString();
	}
	catch(Exception e){
		response.sendRedirect("./index.jsp");
	}
if(parametros!=null) {
	if(!parametros.equals("true")) {
		response.sendRedirect("./index.jsp");
	}
else {
try {
String query = request.getQueryString();
ServletContext servletcontext = getServletContext();
String path = servletcontext.getRealPath("/");
Calendar hmerologio = Calendar.getInstance();

String hmeromhnia = ""+hmerologio.get(Calendar.YEAR)+"-" +




(hmerologio.get(Calendar.MONTH)+1)+"-"+



hmerologio.get(Calendar.DAY_OF_MONTH);
String hmeromhniaANDhour = hmeromhnia +"_"+
hmerologio.get(Calendar.HOUR_OF_DAY);
if( !query.equals("") ){
```

```
        String diagram = query.substring(9);
        if(diagram.equals("daily") ){
                List Diagrammata = new ArrayList();

                String diagramma1 =path+"images/dailystatistics/
                DailyTemperaturesSensorsStatistics_"+hmeromhniaANDhour+".png"
                ;

                String diagramma2 =
                path+"images/dailystatistics/DailyHumiditySensorStatistics_"+
                hmeromhniaANDhour+".png";

                String diagramma3 =
                path+"images/dailystatistics/DailyVoltagesSensorsStatistics_"+
                hmeromhniaANDhour+".png";
        File arxeio1 = new File(diagramma1);
        File arxeio2 = new File(diagramma2);
        File arxeio3 = new File(diagramma3);

        if( (!arxeio1.exists())||(!arxeio2.exists())||(!arxeio3.exists() ) ) {
File katalogoshmerisiwndiagrammatwn =
                                        new File(path+"images/dailystatistics");
File [] pinakasarxeiwn = katalogoshmerisiwndiagrammatwn.listFiles();
for(int i=0;i<pinakasarxeiwn.length;i++){
        pinakasarxeiwn[i].delete();
}
JFreeChart statistikoXronodiagrammaThermokrasiwn =
MyJfreeChartUtility.getXronoDiagrammaThermokrasiwnHmeras("DAILY
TEMPERATURES STATISTICS");
ChartUtilities.saveChartAsPNG(arxeio1,statistikoXronodiagrammaThermokrasiwn,
750,400);
JFreeChart statistikoXronodiagrammaYgrasias =
MyJfreeChartUtility.getXronoDiagrammaYgrasiasHmeras(
                                                "DAILY HUMITDITY STATISTICS");
```

```
ChartUtilities.saveChartAsPNG(arxeio2,statistikoXronodiagrammaYgrasias,
                                        750,400);
JFreeChart statistikoXronodiagrammaTashs =
MyJfreeChartUtility.getXronoDiagrammaTashsHmeras(
                                                "DAILY AC VOLTAGES STATISTICS");
ChartUtilities.saveChartAsPNG(arxeio3,statistikoXronodiagrammaTashs,
                                        750,400);
}
Diagrammata.add("Daily Data Statistics");
Diagrammata.add("images/dailystatistics/DailyTemperaturesSensorsStatisti
                        cs_"+hmeromhniaANDhour+".png");
Diagrammata.add("images/dailystatistics/DailyHumiditySensorStatistics_"+
                        hmeromhniaANDhour+".png");
Diagrammata.add("images/dailystatistics/DailyVoltagesSensorsStatistics_"+
                        hmeromhniaANDhour+".png");

request.setAttribute("lista",Diagrammata);
RequestDispatcher diekpairewths =
                servletcontext.getRequestDispatcher("/statisticsdiagrams.jsp");
diekpairewths.forward( request,  response );
}
else if(diagram.equals("weekly") ){
        List Diagrammata = new ArrayList();
        String diagramma1 =path+"images/weeklystatistics/
                                WeeklyTemperaturesSensorsStatistics_"+
                                hmeromhniaANDhour+".png";
        String diagramma2 = path+"images/weeklystatistics/
                                WeeklyHumiditySensorStatistics_"+
                                hmeromhniaANDhour+".png";
        String diagramma3 = path+"images/weeklystatistics/
                                WeeklyVoltageSensorStatistics_"+
                                hmeromhniaANDhour+".png";
File arxeio1 = new File(diagramma1);
File arxeio2 = new File(diagramma2);
```

```
File arxeio3 = new File(diagramma3);
if( (!arxeio1.exists())||(!arxeio2.exists())||(!arxeio3.exists() ) ) {
File katalogoshmerisiwndiagrammatwn =
                                        new File(path+"images/weeklystatistics");
File [] pinakasarxeiwn = katalogoshmerisiwndiagrammatwn.listFiles();
for(int i=0;i<pinakasarxeiwn.length;i++){
        pinakasarxeiwn[i].delete();
}
JFreeChart statistikoXronodiagrammaThermokrasiwnVdomadas =
MyJfreeChartUtility.getXronoDiagrammaThermokrasiwnVdomadas(
                                        "WEEKLY TEMPERATURES STATISTICS");
JFreeChart statistikoXronodiagrammaYgrasiasVdomadas =
MyJfreeChartUtility.getXronoDiagrammaYgrasiasVdomadas(
                                        "WEEKLY HUMIDITY STATISTICS");
JFreeChart statistikoXronodiagrammaTashsVdomadas =
MyJfreeChartUtility.getXronoDiagrammaTashsVdomadas(
                                        "WEEKLY AC VOLTAGES STATISTICS");
ChartUtilities.saveChartAsPNG(arxeio1,
                statistikoXronodiagrammaThermokrasiwnVdomadas,750,400);
ChartUtilities.saveChartAsPNG(arxeio2,
                statistikoXronodiagrammaYgrasiasVdomadas,750,400);
ChartUtilities.saveChartAsPNG(arxeio3,
                statistikoXronodiagrammaTashsVdomadas,750,400);
}
Diagrammata.add("Weekly Data Statistics");
Diagrammata.add("images/weeklystatistics/
                                WeeklyTemperaturesSensorsStatistics_"+
                                hmeromhniaANDhour+".png");
Diagrammata.add("images/weeklystatistics/
                                WeeklyHumiditySensorStatistics_"
                                hmeromhniaANDhour+".png");
Diagrammata.add("images/weeklystatistics/
                                WeeklyVoltageSensorStatistics_"+
                                hmeromhniaANDhour+".png");
```

```
request.setAttribute("lista",Diagrammata);
RequestDispatcher diekpairewths =
                servletcontext.getRequestDispatcher("/statisticsdiagrams.jsp");

diekpairewths.forward(request, response);
}
else if(diagram.equals("monthly") ){
        List Diagrammata = new ArrayList();
JFreeChart statistikoXronodiagrammaThermokrasiwnMhna =
MyJfreeChartUtility.getXronoDiagrammaThermokrasiwnMhna(
                                "MONTHLY TEMPERATURES STATISTICS");
JFreeChart statistikoXronodiagrammaYgrasiasMhna =
MyJfreeChartUtility.getXronoDiagrammaYgrasiasMhna(
                                "MONTHLY HUMIDITY STATISTICS");
JFreeChart statistikoXronodiagrammaTashsMhna =
MyJfreeChartUtility.getXronoDiagrammaTashsMhna(
                                "MONTHLY AC VOLTAGES STATISTICS");
String diagramma1 = path+"images/monthlystatistics/
                        MonthlyTemperaturesSensorsStatistics_"+
                        hmeromhnia+".png";
String diagramma2 = path+"images/monthlystatistics/
                        MonthlyHumiditySensorStatistics_"+
                        hmeromhnia+".png";
String diagramma3 = path+"images/monthlystatistics/
                        MonthlyVoltageSensorStatistics_"+
                        hmeromhnia+".png";
File arxeio1 = new File(diagramma1);
File arxeio2 = new File(diagramma2);
File arxeio3 = new File(diagramma3);
if( (!arxeio1.exists())||(!arxeio2.exists())||(!arxeio3.exists() ) ) {
File katalogoshmerisiwndiagrammatwn =
                                        new File(path+"images/monthlystatistics");
File [] pinakasarxeiwn = katalogoshmerisiwndiagrammatwn.listFiles();
```

```
for(int i=0;i<pinakasarxeiwn.length;i++){
        pinakasarxeiwn[i].delete();
}
ChartUtilities.saveChartAsPNG(arxeio1,
                        statistikoXronodiagrammaThermokrasiwnMhna,750,400);
ChartUtilities.saveChartAsPNG(arxeio2,
                        statistikoXronodiagrammaYgrasiasMhna,750,400);
ChartUtilities.saveChartAsPNG(arxeio3,
                        statistikoXronodiagrammaTashsMhna,750,400);
}
Diagrammata.add("Monthly Data Statistics");
Diagrammata.add("images/monthlystatistics/
                        MonthlyTemperaturesSensorsStatistics_"+
                        hmeromhnia+".png");
Diagrammata.add("images/monthlystatistics/
                        MonthlyHumiditySensorStatistics_"+
                        hmeromhnia+".png");
Diagrammata.add("images/monthlystatistics/
                        MonthlyVoltageSensorStatistics_"+
                        hmeromhnia+".png");

request.setAttribute("lista",Diagrammata);
RequestDispatcher diekpairewths =
                servletcontext.getRequestDispatcher("/statisticsdiagrams.jsp");
diekpairewths.forward(request, response);
}
else if(diagram.equals("annually") ){
        List Diagrammata = new ArrayList();
                String diagramma1 = path+"images/annuallystatistics/
                                        AnnuallyTemperaturesSensorsStatistics_"+
                                        hmeromhnia+".png";
String diagramma2 = path+"images/annuallystatistics/
                                        AnnuallyHumiditySensorStatistics_"+
                                        hmeromhnia+".png";
```

```
String diagramma3 = path+"images/annuallystatistics/
                                    AnnuallyVoltageSensorStatistics_"+
                                    hmeromhnia+".png";
File arxeio1 = new File(diagramma1);
File arxeio2 = new File(diagramma2);
File arxeio3 = new File(diagramma3);
if( (!arxeio1.exists()) || (!arxeio2.exists()) || (!arxeio3.exists()) ) {
File katalogoshmerisiwndiagrammatwn =
                                new File(path+"images/annuallystatistics");
File [] pinakasarxeiwn = katalogoshmerisiwndiagrammatwn.listFiles();
for(int i=0;i<pinakasarxeiwn.length;i++){
        pinakasarxeiwn[i].delete();
}
JFreeChart statistikoXronodiagrammaThermokrasiwnXronou =
MyJfreeChartUtility.getXronoDiagrammaThermokrasiwnXronou(
                                    "ANNUALLYTEMPERATURES STATISTICS");
JFreeChart statistikoXronodiagrammaYgrasiasXronou =
MyJfreeChartUtility.getXronoDiagrammaYgrasiasXronou(
                                    "ANNUALLY HUMIDITY STATISTICS");
JFreeChart statistikoXronodiagrammaTashsXronou =
MyJfreeChartUtility.getXronoDiagrammaTashsXronou(
                                    "ANNUALLY AC VOLTAGES STATISTICS");
ChartUtilities.saveChartAsPNG(arxeio1,
                statistikoXronodiagrammaThermokrasiwnXronou,750,400);
ChartUtilities.saveChartAsPNG(arxeio2,
                statistikoXronodiagrammaYgrasiasXronou,750,400);
ChartUtilities.saveChartAsPNG(arxeio3,
                statistikoXronodiagrammaTashsXronou,750,400);
}
Diagrammata.add("Annualy Data Statistics");
Diagrammata.add("images/annuallystatistics/
        AnnuallyTemperaturesSensorsStatistics_"+hmeromhnia+".png");
Diagrammata.add("images/annuallystatistics/
        AnnuallyHumiditySensorStatistics_"+ hmeromhnia+".png");
```

```
Diagrammata.add("images/annuallystatistics/
        AnnuallyVoltageSensorStatistics_"+ hmeromhnia+".png");

request.setAttribute("lista",Diagrammata);
RequestDispatcher diekpairewths =
        servletcontext.getRequestDispatcher("/statisticsdiagrams.jsp");
diekpairewths.forward(request, response);
}
else{
        RequestDispatcher diekpairewths =
                                servletcontext.getRequestDispatcher("/home");
        diekpairewths.forward(request, response);
  }
}
else{
RequestDispatcher diekpairewths =
                                servletcontext.getRequestDispatcher("/home");
        diekpairewths.forward(request, response);
}
}
catch(Exception eksaireshsfalmatos){ }
}
}
}
}
}
```

10.5. - Η μικροϋπηρεσία adminsetup

```
package web;
import Utilities.DatabaseUtility;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.RequestDispatcher;
```

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
/**
* @author ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ
*/
public class adminsetup extends HttpServlet {
private String username;
private String password;
private String newwebusername;
private String newwebpassword ;
private String ipaddress;
private String mobilenumber;
private String email;
protected void doGet(HttpServletRequest request, HttpServletResponse
response)throws ServletException,IOException {
HttpSession session = request.getSession(false);
String parametros = null;
if(session==null) {
	response.sendRedirect("./index.jsp");
}
else {
	try{
		parametros = session.getValue("elegxos").toString();
	}
	catch(Exception e){
		response.sendRedirect("./index.jsp");
	}
if(parametros!=null) {
	if(!parametros.equals("true")) {
		response.sendRedirect("./index.jsp");
	}
```

```
        else {
        DatabaseUtility.ConnectToDataBaseServer("com.mysql.jdbc.Driver",
                                "jdbc:mysql://localhost:3306/sensingsystem",
                                "root","10022910");
        Object[] pinakas = new Object[4];
        pinakas = DatabaseUtility.processSQL(
                                        "select * from applicationowner;",pinakas);
        DatabaseUtility.terminateConnection();
        List lista = new ArrayList();
        lista.add(pinakas[0]);
        lista.add(pinakas[1]);
        lista.add(pinakas[2]);
        lista.add(pinakas[3]);

        request.setAttribute("LISTA",lista);
RequestDispatcher diekperaiwths =
                getServletContext().getRequestDispatcher("/adminIdentity.jsp");
diekperaiwths.forward(request, response);
}
}
}
}
protected void doPost(HttpServletRequest request, HttpServletResponse
response)throws ServletException,IOException {
username =          (String)request.getParameter("txtfld1");
password =          (String)request.getParameter("txtfld2");
String hidden =     (String)request.getParameter("hiddenField");
ipaddress =         request.getRemoteAddr();
boolean checkit = true;
DatabaseUtility.ConnectToDataBaseServer("com.mysql.jdbc.Driver",
                                        "jdbc:mysql://localhost:3306/sensingsystem","ro
                                        ot","10022910");
checkit = DatabaseUtility.Validation( username,  password );
if(checkit){
```

```
//ο χρήστης δεν θέλει να αλλάξει ουτε username, ούτε password
        if(hidden.equals("false")){
                mobilenumber =(String)request.getParameter("txtfld3");
                email =(String)request.getParameter("txtfld4");

                ChangeMailOrMobile(username);
                request.setAttribute("msg",
                "YOUR CHANGES HAVE BEEN APPLIED!
                NEW MOBILE NUMBER & MOBILE ADDRESS HAVE BEEN
                CHANGED!");

                request.setAttribute("topothesia","./adminsetup");
                RequestDispatcher diekpairewths =
                        getServletContext().getRequestDispatcher("/messages.jsp");
        diekpairewths.forward(request, response);
}
else{//ο χρήστης θέλει να αποθηκεύσει username,ή password
newwebusername =(String)request.getParameter("txtfld10");
newwebpassword =(String)request.getParameter("txtfld11");
mobilenumber =(String)request.getParameter("txtfld3");
email =(String)request.getParameter("txtfld4");

ChangeUsernameOrPassword();

if( checkFormParameters(newwebusername) )
        ChangeMailOrMobile(newwebusername);
else
        ChangeMailOrMobile(username);

request.setAttribute("msg",
                "YOUR CHANGES HAVE BEEN APPLIED!PLEASE LOGIN WITH
                YOUR NEW IDENTITY");
request.setAttribute("topothesia","./adminsetup");
RequestDispatcher diekpairewths =
```

```
                getServletContext().getRequestDispatcher("/messages.jsp");
diekpairewths.forward(request, response);
}
}//τελος if του checkit
else{
request.setAttribute("msg",
                        "YOUR USERNAME OR PASSWORD ARE NOT CORRECT
                        PLEASE TRY AGAIN!");
request.setAttribute("topothesia","./adminsetup");
RequestDispatcher diekpairewths =
                getServletContext().getRequestDispatcher("/messages.jsp");
diekpairewths.forward(request, response);
//ο χρήστης έδωσε λάθος username και password
}
}
private boolean checkFormParameters(String param){
        if((param!=null)&&(!param.equals("")))
                return true;
        else
        return false;
}
private void ChangeMailOrMobile(String user){
        if(!checkFormParameters(mobilenumber) ){
                if(checkFormParameters(email)){
                        DatabaseUtility.processSQL("
                        update applicationowner set emailaddress='"+email+"'
                        where webusername like '"+user+"';");
DatabaseUtility.writeToLogTable("USER :"+user.toUpperCase()+
                                        " CHANGED EMAIL ADDRESS",ipaddress );
DatabaseUtility.terminateConnection();
}
else{ DatabaseUtility.terminateConnection(); }
}
```

```
else{
//ο χρήστης έχει εισάγει κινητό τηλέφωνο
if(checkFormParameters(email)){
DatabaseUtility.processSQL("update applicationowner set
                                    emailaddress='"+email+"',mobilenumber='"+
                                    mobilenumber+"' where webusername
                                    like '"+user+"';");
DatabaseUtility.writeToLogTable("USER :"+user.toUpperCase()+
                                    "CHANGED EMAIL ADDRESS & MOBILE
                                    NUMBER",ipaddress );
DatabaseUtility.terminateConnection();
}
else{
        DatabaseUtility.processSQL("update applicationowner set
        mobilenumber='"+mobilenumber+"' where webusername
        like '"+user+"';");
DatabaseUtility.writeToLogTable("USER :"+user.toUpperCase()+
                                    " CHANGED MOBILE NUMBER",ipaddress );
DatabaseUtility.terminateConnection();
}
}
}
private void ChangeUsernameOrPassword(){
        if(!checkFormParameters(newwebusername) ){
                if(checkFormParameters(newwebpassword)){
                DatabaseUtility.processSQL("update applicationowner set
                                        webpassword='"+newwebpassword+"'
                                        where webusername like '
                                        "+username+"';");
                        DatabaseUtility.writeToLogTable("USER:"+
                                                username.toUpperCase()+
                                        " CHANGED WEB PASSWORD",
                                                ipaddress );
```

```
}
else{
        DatabaseUtility.terminateConnection();
}
}
else{
```

ο χρήστης θέλει να αλλάξει username

```
if(checkFormParameters(newwebpassword) ){
DatabaseUtility.processSQL("update applicationowner set
                                webusername='"+newwebusername+"',
                                webpassword='"+newwebpassword+"'
                                where webusername like '"+username+"';");
DatabaseUtility.writeToLogTable("USER :"+username.toUpperCase()+
                        " CHANGED USERNAME & PASSWORD",ipaddress );
}
else{//θέλει να αλλάξει μόνο username
DatabaseUtility.processSQL("update applicationowner set
                                webusername='"+newwebusername+"'
                                where webusername like '"+username+"';");
DatabaseUtility.writeToLogTable("USER :"+
                                username.toUpperCase()+
                                "CHANGED HIS MOBILE NUMBER",ipaddress );
}
}
}
}
```

10.6. - Η μικροϋπηρεσία mailsetup

```
package web;
import Utilities.DatabaseUtility;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
```

```
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
/**
*
* @ author ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ
*/
public class mailsetup extends HttpServlet {
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException {
HttpSession session = request.getSession(false);
String parametros = null;
if(session==null) {
        response.sendRedirect("./index.jsp");
}
else {
        try{
                parametros = session.getValue("elegxos").toString();
        }
        catch(Exception e){
                response.sendRedirect("./index.jsp");
        }
        if(parametros!=null) {
                if(!parametros.equals("true")) {
                        response.sendRedirect("./index.jsp");
                }
                else {
DatabaseUtility.ConnectToDataBaseServer("com.mysql.jdbc.Driver",
                                "jdbc:mysql://localhost:3306/sensingsystem",
                                "root","10022910");
```

```
Object[] pinakas = new Object[5];
List lista = new ArrayList();
pinakas = DatabaseUtility.processSQL(
                                        "select * from mailparameters;",pinakas);
DatabaseUtility.terminateConnection();
lista.add(pinakas[0]);
lista.add(pinakas[1]);
lista.add(pinakas[2]);
lista.add(pinakas[3]);
lista.add(pinakas[4]);

request.setAttribute("LISTA",lista);
RequestDispatcher diekperaiwths =
                getServletContext().getRequestDispatcher("/mailIdentity.jsp");
diekperaiwths.forward(request, response);
}
}
}
}
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
String email =(String)request.getParameter("txtfld1");
String username =(String)request.getParameter("txtfld2");
String password =(String)request.getParameter("txtfld3");
String smtp =(String)request.getParameter("txtfld4");
String ssl = request.getParameter("txtfld5");
String ipaddress = request.getRemoteAddr();
DatabaseUtility.ConnectToDataBaseServer("com.mysql.jdbc.Driver",
                                        "jdbc:mysql://localhost:3306/sensingsystem","ro
                                        ot","10022910");
String oldemailaddress = (String) DatabaseUtility.processSQL(
                                                "SELECT emailaddress
                                                FROM mailparameters;",1);
```

```
if(checkFormParameters( String.valueOf(ssl) ) ){
int sslport = Integer.parseInt( request.getParameter("txtfld5") );
DatabaseUtility.processSQL("update mailparameters set
                            emailaddress='"+email+"',smtpserver='"+smtp+"',
                            mailuser='"+username+"',mailpassword='"+
                            password+"',sslport="+sslport+"
                            where emailaddress like '"+oldemailaddress+"';");
DatabaseUtility.writeToLogTable("SENSING DEVICE EMAIL
                                    ACCOUNT HAS BEEN CHANGED!",
                                    ipaddress );
DatabaseUtility.terminateConnection();
request.setAttribute("msg", "YOUR CHANGES HAVE BEEN APPLIED!");
request.setAttribute("topothesia","./mailsetup");
RequestDispatcher diekpairewths =
            getServletContext().getRequestDispatcher("/messages.jsp");
diekpairewths.forward(request, response);
}
else{
if( !checkFormParameters(email) && !checkFormParameters(smtp) &&
!checkFormParameters(username) && !checkFormParameters(password)
&& !checkFormParameters(oldemailaddress) ){
        DatabaseUtility.processSQL("update mailparameters set
                                    emailaddress='"+email+"',smtpserver='"+smtp+"'
                                    ,mailuser='"+username+"',mailpassword='"+pass
                                    word+"',sslport=0
                                    where emailaddress
                                    like '"+oldemailaddress+"';");
DatabaseUtility.writeToLogTable("SENSING DEVICE EMAIL ACCOUNT
                                    HAS BEEN CHANGED!",ipaddress );
DatabaseUtility.terminateConnection();

request.setAttribute("msg", "YOUR CHANGES HAVE BEEN APPLIED!");
request.setAttribute("topothesia","./mailsetup");
RequestDispatcher diekpairewths =
```

```
                getServletContext().getRequestDispatcher("/messages.jsp");
diekpairewths.forward(request, response);
}
else {
        response.sendRedirect("./mailsetup");
}
}
}

private boolean checkFormParameters(String param){
        if((param!=null)&&(!param.equals("")))
                return true;
        else
                return false;
}
}
```

10.7. - Η μικροϋπηρεσία sensorsSetup

```
package web;
import Utilities.DatabaseUtility;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
/**
* @ author ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ
*/
public class sensorsSetup extends HttpServlet {
```

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
HttpSession session = request.getSession(false);
String parametros = null;
if(session==null) {
        response.sendRedirect("./index.jsp");
}
else {
        try{
                parametros = session.getValue("elegxos").toString();
        }
        catch(Exception e){
                response.sendRedirect("./index.jsp");
        }
        if(parametros!=null) {
                if(!parametros.equals("true")) {
                        response.sendRedirect("./index.jsp");
                }
                else {
        DatabaseUtility.ConnectToDataBaseServer("com.mysql.jdbc.Driver",
                                        "jdbc:mysql://localhost:3306/sensingsystem","ro
                                        ot","10022910");
List listaAN1 = new ArrayList();
List listaAN2 = new ArrayList();
List listaAN3 = new ArrayList();
List listaAN4 = new ArrayList();
List listaDIG1 = new ArrayList();
List listaDIG2 = new ArrayList();
GemismaListas("AN1", listaAN1);
GemismaListas("AN2", listaAN2);
GemismaListas("AN3",listaAN3);
GemismaListas("AN4",listaAN4);
GemismaListas("DIG1",listaDIG1);
```

```
GemismaListas("DIG2",listaDIG2);
Object LimitValue = DatabaseUtility.processSQL("select limitvalue
                                                from analogsensors
                                                where analogsensorid like
                                                'AN1';",1);
listaAN1.add(LimitValue);
LimitValue = DatabaseUtility.processSQL( "select limitvalue
                                          from analogsensors
                                          where analogsensorid
                                          like 'AN2';",1);
listaAN2.add(LimitValue);
LimitValue = DatabaseUtility.processSQL("select limitvalue
                                          from analogsensors
                                          where analogsensorid
                                          like 'AN3';",1);
listaAN3.add(LimitValue);
LimitValue = DatabaseUtility.processSQL("select limitvalue
                                          from analogsensors
                                          where analogsensorid
                                          like 'AN4';",1);
listaAN4.add(LimitValue);
DatabaseUtility.terminateConnection();

request.setAttribute("LISTA1",listaAN1);
request.setAttribute("LISTA2",listaAN2);
request.setAttribute("LISTA3",listaAN3);
request.setAttribute("LISTA4",listaAN4);
request.setAttribute("LISTA5",listaDIG1);
request.setAttribute("LISTA6",listaDIG2);

RequestDispatcher diekperaiwths =
        getServletContext().getRequestDispatcher("/sensorsIdentity.jsp");
diekperaiwths.forward(request, response);
}
```

```
}
}
}
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException {
String ipaddress = request.getRemoteAddr();
DatabaseUtility.ConnectToDataBaseServer("com.mysql.jdbc.Driver",
                                        "jdbc:mysql://localhost:3306/sensingsystem","ro
                                        ot","10022910");
String sensorMsg,info = null;
double lista = 0.0;
lista = Double.parseDouble( request.getParameter("lista1").toString() );
sensorMsg = (String)request.getParameter("txtfld2");
info = (String)request.getParameter("txtfld3");
        updateSensorsSetup("AN1",info,sensorMsg,lista);
lista = Double.parseDouble(request.getParameter("lista2").toString() );
sensorMsg = (String)request.getParameter("txtfld5");
info = (String)request.getParameter("txtfld6");
        updateSensorsSetup("AN2",info,sensorMsg,lista);
lista = Double.parseDouble( request.getParameter("lista3").toString() );
sensorMsg = (String)request.getParameter("txtfld8");
info = (String)request.getParameter("txtfld9");
        updateSensorsSetup("AN3",info,sensorMsg,lista);
lista = Double.parseDouble( request.getParameter("lista4").toString() );
sensorMsg = (String)request.getParameter("txtfld11");
info = (String)request.getParameter("txtfld12");
        updateSensorsSetup("AN4",info,sensorMsg,lista);
sensorMsg = (String)request.getParameter("txtfld13");
info = (String)request.getParameter("txtfld14");
updateSensorsSetup("DIG1",info,sensorMsg);
sensorMsg = (String)request.getParameter("txtfld15");
info = (String)request.getParameter("txtfld16");
        updateSensorsSetup("DIG2",info,sensorMsg);
```

```
DatabaseUtility.writeToLogTable("SENSORS PARAMETERS HAVE BEEN
CHANGED!", ipaddress);
DatabaseUtility.terminateConnection();

request.setAttribute("msg","YOUR CHANGES HAVE BEEN APPLIED!");
request.setAttribute("topothesia","./sensorsSetup");
RequestDispatcher diekperaiwths =
                getServletContext().getRequestDispatcher("/messages.jsp");
diekperaiwths.forward(request, response);
}

private void GemismaListas(String sensorid,List lista){
Object[] pinakas1 = new Object[2];
pinakas1 = DatabaseUtility.processSQL("select info,alertmessage
                                                from sensors
                                                where sensorid
                                                like '"+sensorid+"';",pinakas1);
lista.add(pinakas1[0] );
lista.add(pinakas1[1] );
}

private boolean checkFormParameters(String param){
        if( (param!=null) && (!param.equals("")) )
                return true;
        else
                return false;
}

private void updateSensorsSetup(String id,String info,String msg,
                                                                double limit){
if(checkFormParameters(info) ){
        if(checkFormParameters(msg) ){
                if(limit!=0 ){//θέλει να αλλάξει alert message,info και limit value
                DatabaseUtility.processSQL("UPDATE analogsensors
```

```
                                        SET limitvalue="+limit+
                                        " WHERE analogsensorid
                                        like '"+id+"';");
DatabaseUtility.processSQL("UPDATE sensors
                                SET info='"+info+"',alertmessage='"+msg+"'
                                WHERE sensorid like '"+id+"';");
}
else{//θέλει να αλλάξει alert message και info
DatabaseUtility.processSQL("UPDATE sensors
                                SET info='"+info+"',alertmessage='"+msg+"'
                                WHERE sensorid like '"+id+"';");
}
}
else{
```

δεν θέλει να αλλάξει alert message,αλλά θέλει να αλλάξει info και limit value

```
        if(limit!=0 ){
        DatabaseUtility.processSQL("UPDATE analogsensors
                                        SET limitvalue="+limit+
                                        "WHERE analogsensorid
                                        like '"+id+"';");
DatabaseUtility.processSQL("UPDATE sensors SET info='"+info+
                                        " WHERE sensorid like '"+id+"';");
}
else{
DatabaseUtility.processSQL("UPDATE sensors SET info='"+info+"'
                                        WHERE sensorid like '"+id+"';");
}
}
}
else{//δεν θέλουμε να αλλάξουμε info
if(checkFormParameters(msg) ){
        if(limit!=0 ){//θέλει να αλλάξει alert message και limit value
DatabaseUtility.processSQL("UPDATE analogsensors
```

```
                                    SET limitvalue="+limit+
                                    " WHERE analogsensorid like '"+id+"';");
DatabaseUtility.processSQL("UPDATE sensors
                                    SET alertmessage='"+msg+
                                    "' WHERE sensorid like '"+id+"';");
}
else{//θέλει να αλλάξει alert message
DatabaseUtility.processSQL("UPDATE sensors
                                    SET alertmessage='"+msg+
                                    "' WHERE sensorid like '"+id+"';");
}
}
else{//δεν θέλει να αλλάξει limit value
        if(limit!=0 ){
DatabaseUtility.processSQL("UPDATE analogsensors
                                    SET limitvalue="+limit+
                                    " WHERE analogsensorid like '"+id+"';");
}
else{
//ΚΑΜΙΑ ΑΛΛΑΓΗ
}
}
}
}

private void updateSensorsSetup(String id,String info,String msg){
if(checkFormParameters(info) ){
        if(checkFormParameters(msg) ){
DatabaseUtility.processSQL("UPDATE sensors
                                 SET info='"+info+"',alertmessage='"+msg+"'
                                WHERE sensorid like '"+id+"';");
}
else{//αλλαγή μόνο info μηνύματος
DatabaseUtility.processSQL("UPDATE sensors SET info='"+info+"'
```

```
WHERE sensorid like '"+id+"';");
}
}
else{
if(checkFormParameters(msg) ){
DatabaseUtility.processSQL("UPDATE sensors
                                    SET alertmessage='"+msg+"'
                                    WHERE sensorid like '"+id+"';");
}
else{ }
}
}
}
```

10.8. - Η μικροϋπηρεσία logOut

```
package web;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
/**
*
* @author ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ
*/
public class logOut extends HttpServlet {
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse
response)throws
ServletException, IOException {
HttpSession session = request.getSession(false);
```

```
String parametros = null;
if(session!=null) {
session.invalidate();
request.setAttribute("msg", "YOU ARE BEING LOGOUT FROM THE SYSTEM");
request.setAttribute("topothesia","./index.jsp");
RequestDispatcher diekpairewths =
getServletContext().getRequestDispatcher("/messages.jsp");
diekpairewths.forward(request, response);
}
else {
request.setAttribute("msg", "YOU ARE NOT LOGED IN!");
request.setAttribute("topothesia","./index.jsp");
RequestDispatcher diekpairewths =
getServletContext().getRequestDispatcher("/messages.jsp");
diekpairewths.forward(request, response);
}
}
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws
ServletException, IOException {
}
}
```

10.9. - Η σελίδα central.jsp

<%--

Document : central
Created on : 29 Μαϊ 2009, 8:43:53 μμ
Author : ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>SENSING SYSTEM CENTRAL APPLICATION MENU</title>
<%@page import="java.util.Properties" %>
<script src="./javascript/formValidation.js"> </script>
<%
HttpSession session2 = request.getSession(false);
String parametros = null;
if(session2==null) {
response.sendRedirect("./index.jsp");
}
else {
try{

```
parametros = session2.getValue("elegxos").toString();
}
catch(Exception e){
response.sendRedirect("./index.jsp");
}
if(parametros!=null) {
if(!parametros.equals("true")) {
response.sendRedirect("./index.jsp");
}
}
}
%>
</head>
<body bgcolor='black'>
<center>
<%@ include file="menu.jsp" %>
<img src="images/header.jpg" width="100%" height="10%"></img>
<br><br>
</center>
<%!
Runtime run1;
Properties pr;
long totalmem;
String stats;
%>
<%
try{
run1 = Runtime.getRuntime();
run1.gc();
totalmem = run1.totalMemory();
pr = System.getProperties();
totalmem = (long) (((float) totalmem) / 1000000.0);
stats ="Available Memory for Virtual Machine Process :"+totalmem+" MegaBytes";
}
```

```
catch(Exception e){}
%>
<b>
<font color="red">YOUR OPERATING SYSTEM IS :</font> <font color="white">
<%=pr.getProperty("os.name")%> </font> <br>
<font color="red">AVAILABLE CPU CORES :</font> <font color="white">
<%=run1.availableProcessors()%> </font> <br>
<font color="red">YOUR IP ADDRESS IS :</font> <font color="white">
<%=request.getRemoteAddr()%> </font> <br>
</b>
<center><font color="red">Environment Variables</font> <br>
<table border="1">
<tr>
<td><font color="red">Runtime Environment Name:</font> </td><td>
<font color="white"> <%=pr.getProperty("java.runtime.name")%></font> </td>
</tr>
<tr>
<td><font color="red">Virtual Machine Version:</font> </td> <td>
<font color="white"><%=pr.getProperty("java.vm.version")%></font> </td>
</tr>
<tr>
<td><font color="red">Virtual Machine Name:</font></td> <td>
<font color="white"><%=pr.getProperty("java.vm.name")%> </font> </td>
</tr>
<tr>
<td><font color="red">Your Country:</font> </td> <td>
<font color="white"><%=pr.getProperty("user.country")%> </font> </td>
</tr>
<tr>
<td><font color="red">Your Host Name:</font></td> <td>
<font color="white"><%=pr.getProperty("user.name")%> </font> </td>
</tr>
<tr>
<td><font color="red">Windows Service Pack:</font></td> <td>
```

```
<font color="white"><%=pr.getProperty("sun.os.patch.level")%> </font> </td>
</tr>
<tr>
<td><font color="red">Operating System Architecture:</font></td> <td>
<font color="white"><%=pr.getProperty("os.arch") %> </font> </td>
</tr>
<tr>
<td> <font color="red">Apache Tomcat Web Server Path:</font> </td><td>
<font color="white"><%=pr.getProperty("catalina.home")%> </font> </td>
</tr>
</table>
<br><br>
<B>
<font color="white"> <%=stats %> </font>
</B>
<br><br>
<img src="./images/tomcat.gif" width="50" height="50"
onclick="window.location='http://tomcat.apache.org/'; alt='http://tomcat.apache.org'
">
<img src="./images/java.png" width="50" height="50"
onclick="window.location='http://java.sun.com/'; alt='http://java.sun.com/' ">
</center>
</body>
</html>
```

10.10. - Η σελίδα index.jsp

```
<%--


Document : index
Created on : 8 Μαϊ 2009, 12:05:56 πμ
Author : ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ
```

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<script src="./javascript/formValidation.js"> </script>
<title>COMPUTER ROOM SENSING SYSTEM</title>
</head>
<body bgcolor="black"> <font color='white'>

<h1>COMPUTER ROOM SENSING SYSTEM</h1>
<img src='./images/Thanos&AlekshsSensingSystem.jpg' height='390'
width='600'>

<b>
<form action='./home' method='post' onSubmit='return
CheckParameters(this.txtfld1.value,this.txtfld2.value)'>

```
USERNAME : <input type="text" size='20' name="txtfld1" onblur='elegxos(this)'>
<br><br>
PASSWORD : <input type="password" size='20' name="txtfld2"
onblur='elegxos(this)'>
<br><br>
<input type="submit" value="Enter">
</form></b><br>
<div id='divobj'></div>
<script language='javascript'>
window.setInterval('RoloiThanoy()',1000);
window.status='YOUR COMPUTER TYPE IS: '+navigator.platform
function RoloiThanoy(){
var hmeromhnia=new Date();
var Defterolepta=hmeromhnia.getSeconds();
var Lepta=hmeromhnia.getMinutes();
var wres=hmeromhnia.getHours();
if(wres<10){
var wra="0"+wres;
}
else{
var wra=wres;
}
if(Lepta<10){
var lepta="0"+Lepta;
}
else{
var lepta=Lepta;
}
if(Defterolepta<10){
var seconds="0"+Defterolepta;
}
else{
var seconds=Defterolepta;
}
```

```
document.getElementById('divobj').innerHTML="<font color='red' size='24'>"
+wra+":"+lepta+":"+seconds+"</font>";
}
</script>
</center>
</font>
</body>
</html>
```

10.11. - Η σελίδα menu.jsp

```
<%--

Document : menu
Created on : 9 Ιουν 2009, 7:19:53 μμ
Author : ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01

Transitional//EN""http://www.w3.org/TR/html4/loose.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />



<link rel="stylesheet" href="CSSfiles/styles.css" type="text/css" />
</head>
<body>
<div id="menucase">
<div id="stylefour">
<ul>
<li><a onclick="window.location='./home'" class='current'>Home </a></li>
<li><a onclick="window.location='./livepresentation'" >Live Presentation</a></li>
<li><a onclick="window.location='./history?diagrams=daily'" >Daily
Statistics</a></li>
<li><a onclick="window.location='./history?diagrams=weekly'" >Weekly
Statistics</a></li>
<li><a onclick="window.location='./history?diagrams=monthly'" >Monthly
Statistics</a></li>
<li><a onclick="window.location='./history?diagrams=annually'" >Annually
Statistics</a></li>
<li><a onclick="window.location='./adminsetup'"> Admin Identity Setup</a></li>
<li><a onclick="window.location='./mailsetup'" >Email Account Setup</a></li>
<li><a onclick="window.location='./sensorsSetup'">Sensors Setup</a></li>
<li><a onclick="window.location='log.jsp'" >Logs</a></li>
<li><a onclick="window.location='./logOut'">Log Out</a></li>
</ul>
</div>
</div>
</body>
</html>
```

10.12. - Η σελίδα diagrams.jsp

<%--

Document : diagrams
Created on : 2 Ιουν 2009, 12:52:51 μμ
Author : ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>ANALOG & DIGITAL SENSORS LIVE PRESENTATION</title>
<script src="./javascript/formValidation.js"></script>

```
<%@ page import="java.util.*" %>
<%
HttpSession session2 = request.getSession(false);
String parametros = null;
if(session2==null) {
response.sendRedirect("./index.jsp");
}
else {
try{
parametros = session2.getValue("elegxos").toString();
}
catch(Exception e){
response.sendRedirect("./index.jsp");
}
if(parametros!=null) {
if(!parametros.equals("true")) {
response.sendRedirect("./index.jsp");
}
}
}
%>
<%
List listaparametrwn = (List)request.getAttribute("lista");
%>
</head>
<body bgcolor="black">
<center>
<%@ include file="menu.jsp" %>
<font color="red">
<b>Last Measurement At: <%=listaparametrwn.get(4).toString()%> </b>
<table>
<tr>
<td width="80" height="80"><img src="<%=listaparametrwn.get(0).toString()%>">
</td>
```

<td width="80" height="80"><img src="<%=listaparametrwn.get(1).toString()%>">
</td>
</tr>
<tr>
<td width="80" height="80"><img src="<%=listaparametrwn.get(2).toString()%>">
</td>
<td width="80" height="80"><img src="<%=listaparametrwn.get(3).toString()%>">
</td>
</tr>
</table>

<% response.addHeader("Refresh","5;livepresentation"); %>
</body>
</html>

10.13. - Η σελίδα statisticsdiagrams.jsp

<%--

Document : statisticsdiagrams
Created on : 2 Ιουν 2009, 7:35:29 μμ
Author : ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"



"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">



<%@ page import="java.util.*" %>
<title>ANALOG SENSORS STATISTICS</title>
<script src="./javascript/formValidation"> </script>
<%
HttpSession session2 = request.getSession(false);
String parametros = null;
if(session2==null) {
response.sendRedirect("./index.jsp");
}
else {
try{
parametros = session2.getValue("elegxos").toString();
}
catch(Exception e){
response.sendRedirect("./index.jsp");
}
if(parametros!=null) {
if(!parametros.equals("true")) {
response.sendRedirect("./index.jsp");
}
}
```

```
}
%>
<%
List orismata = (List) request.getAttribute("lista");
%>
</head>
<body bgcolor="black">
<center>
<%@ include file="menu.jsp" %>
<H1> <%= orismata.get(0).toString() %> </H1> <br><br>
<img src="<%= orismata.get(1).toString()%>"> <br><br>
<img src="<%= orismata.get(2).toString()%>"> <br><br>
<img src="<%= orismata.get(3).toString()%>" > <br><br>
</center>
</body>
</html>
```

10.14. - Η σελίδα adminIdentity.jsp

```
<%--

Document : adminIdentity
Created on : 5 Ιουν 2009, 3:51:50 μμ
Author : ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"



"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<%@ page import="java.util.*" %>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">



<title>SENSING SYSTEM ADMINISTARTOR IDENTITY SETUP</title>
<script src="./javascript/formValidation.js"> </script>
<%
HttpSession session2 = request.getSession(false);
String parametros = null;
if(session2==null) {
response.sendRedirect("./index.jsp");
}
else {
try{
parametros = session2.getValue("elegxos").toString();
}
catch(Exception e){
response.sendRedirect("./index.jsp");
}
if(parametros!=null) {
if(!parametros.equals("true")) {
response.sendRedirect("./index.jsp");
}
}
```

```
}
%>
<%
List list = (List) request.getAttribute("LISTA");
%>
</head>
<body bgcolor="black" color="white">
<%@ include file="menu.jsp" %>
<center>
<font color="red" size="8">Configure Administrator Identity</font>
<br><br>
<table border="1">
<tr>
<th align="center"><font color="red"> <b> CURRENT CONFIGURATION </b>
</font> <th>
</tr>
<tr><td>
<table cellpadding="5">
<tr>
<td> <font color="red"> USERNAME : </font></td>
<td> <font color="white"> <%= list.get(0) %> </font> </td>
</tr>
<tr>
<td> <font color="red"> PASSWORD : </font> </td>
<td> <font color="white"> <%= list.get(1) %> </font> </td>
</tr>
<tr>
<td> <font color="red"> MOBILE NUMBER : </font> </td>
<td> <font color="white"> <%= list.get(2) %> </font> </td>
</tr>
<tr>
<td> <font color="red"> EMAIL ADDRESS : </font> </td>
<td> <font color="white"> <%= list.get(3) %> </font> </td>
</tr>
```

```
</table>
</td></tr></table>
<br><br>
<form action='./adminsetup' method='post' onSubmit='return
CheckParameters2(this.txtfld1.value,this.txtfld2.value,this.txtfld3.value,
                                                this.txtfld4.value,this.hiddenField.value)'>
<table border='0' cellpadding="3">
<tr>
<td valign='top'><font color="orange">Web Username :</font></td>
<td valign='top'> <input type='text' size='20' name='txtfld1' onblur='elegxos(this)'>
</td>
</tr>
<tr>
<td valign='top'><font color="orange">Web Password :</font></td>
<td valign='top'> <input type='password' size='20' name='txtfld2'
onblur='elegxos(this)'> </td>
</tr>
<tr>
<td valign='top'><font color="orange">New Mobile Number :</font></td>
<td valign='top'> <input type='text' size='20' name='txtfld3' onblur='elegxos(this)'>
</td>
</tr>
<tr>
<td valign='top'><font color="orange">New Email Address :</font></td>
<td valign='top'> <input type='text' size='20' name='txtfld4' onblur='elegxos(this)'>
</td>
</tr>
</table>
<input type="hidden" id="hiddenField" name="hiddenField" value="false">
<br><br>
<a onclick="showUserPassFields()" style="color:orange">
click here to change username/password</a>
<div id="changeuserpass"></div><br>
<input type='submit' value='Save'>
```

<input type="button" onclick="javascript:history.go(-1)" value="CANCEL">

</form>

</body>
</html>

10.15. - Η σελίδα mailIdentity.jsp

<%--

Document : mailIdentity
Created on : 8 Ιουν 2009, 5:12:01 μμ
Author : ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

```
<html>
<head>
<%@ page import="java.util.*" %>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">



<title>SENSING SYSTEM MAIL ACCOUNT IDENTITY SETUP</title>
<script src="./javascript/formValidation.js"> </script>
<%
HttpSession session2 = request.getSession(false);
String parametros = null;
if(session2==null) {
response.sendRedirect("./index.jsp");
}
else {
try{
parametros = session2.getValue("elegxos").toString();
}
catch(Exception e){
response.sendRedirect("./index.jsp");
}
if(parametros!=null) {
if(!parametros.equals("true")) {
response.sendRedirect("./index.jsp");
}
}
}
%>
<%
List list = (List) request.getAttribute("LISTA");
%>
</head>
```

```
<body bgcolor="black" color="white">
<%@ include file="menu.jsp" %>
<center>
<font color="red">
<font color="red" size="8">Configure Mail Account Identity</font>
<br><br>
<table border="1">
<tr>
<th align="red"> <b> CURRENT CONFIGURATION </b> </th>
</tr>
<tr><td>
<table cellpadding="5">
<tr>
<td><font color="red">EMAIL ADDRESS :</font></td>
<td> <font color="white"><%= list.get(0) %> </font> </td>
</tr>
<tr>
<td><font color="red">USERNAME :</font></td>
<td> <font color="white"> <%= list.get(2) %> </font> </td>
</tr>
<tr>
<td><font color="red">PASSWORD :</font></td>
<td> <font color="white"> <%= list.get(3) %> </font> </td>
</tr>
<tr>
<td><font color="red">SMTP SERVER :</font></td>
<td> <font color="white"> <%= list.get(1) %> </font> </td>
</tr>
<tr>
<td><font color="red">SSL PORT :</font></td>
<td> <font color="white"> <%= list.get(4) %> </font> </td>
</tr>
</table>
</td></tr></table>
```

```
<br><br>
<form action='./mailsetup' method='post' onSubmit='return
CheckParameters3(this.txtfld1.value,this.txtfld2.value,this.txtfld3.value,this.txtfld4.v
alue)'>
<table border='0' cellpadding="3">
<tr>
<td valign='top'><font color="orange">EMAIL ADDRESS :</font></td>
<td valign='top'> <input type='text' size='20' name='txtfld1' onblur='elegxos(this)'>
</td>
</tr>
<tr>
<td valign='top'><font color="orange">USERNAME :</font></td>
<td valign='top'> <input type='text' size='20' name='txtfld2' onblur='elegxos(this)'>
</td>
</tr>
<tr>
<td valign='top'><font color="orange">PASSWORD :</font></td>
<td valign='top'> <input type='password' size='20' name='txtfld3'
onblur='elegxos(this)'>
</td>
</tr>
<tr>
<td valign='top'><font color="orange">SMTP SERVER :</font></td>
<td valign='top'> <input type='text' size='20' name='txtfld4' onblur='elegxos(this)'>
</td>
</tr>
<tr>
<td valign='top'><font color="orange">SSL PORT :</font></td>
<td valign='top'> <input type='text' size='20' name='txtfld5' onblur='elegxos(this)'>
</td>
</tr>
</table>
<input type='submit' value='Save'>
```

<input type="button" onclick="javascript:history.go(-1)" value="CANCEL">

</form>

</body>
</html>

10.16. - Η σελίδα sensorsIdentity.jsp

<%--

Document : sensorsIdentity
Created on : 8 Ιουν 2009, 7:41:08 μμ
Author : ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

```
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<%@ page import="java.util.*" %>
<title>SENSING SYSTEM SENSORS PARAMETERS SETUP</title>.
<%
HttpSession session2 = request.getSession(false);
String parametros = null;
if(session2==null) {
response.sendRedirect("./index.jsp");
}
else {
try{
parametros = session2.getValue("elegxos").toString();
}
catch(Exception e){
response.sendRedirect("./index.jsp");
}
if(parametros!=null) {
if(!parametros.equals("true")) {
response.sendRedirect("./index.jsp");
}
}
}
%>
<%
List sensor1 = (List) request.getAttribute("LISTA1");
List sensor2 = (List) request.getAttribute("LISTA2");
List sensor3 = (List) request.getAttribute("LISTA3");
List sensor4 = (List) request.getAttribute("LISTA4");
List sensor5 = (List) request.getAttribute("LISTA5");
List sensor6 = (List) request.getAttribute("LISTA6");
%>
<script src="javascript/formValidation.js"></script>
```

```
</head>
<body bgcolor="black" color="white" onload="gemismaListasLimits()">
<%@ include file="menu.jsp" %>
<center>
<font color="red" size="8">
Configure Sensors Parameters
</font>
<br><br>
<br><br>
<table border="1">
<tr>
<th align="center"> <font color="red"> <b> CURRENT CONFIGURATION </b>
</font> </th>
</tr>
<tr><td>
<table cellpadding="5">
<tr>
<th></th>
<th><font color="red">ANALOG SENSOR 1 </font></th>
<th><font color="red">ANALOG SENSOR 2 </font></th>
<th><font color="red">HUMIDITY SENSOR </font></th>
<th><font color="red">AC VOLTAGE SENSOR</font></th>
<th><font color="red">SMOKE DETECTION </font></th>
<th><font color="red">FLOOD DETECTION </font></th>
</tr>
<tr>
<th align="right"><font color="red">Sensor Info </font></th>
<td> <font color="white"> <%=sensor1.get(0).toString()%> </font> </td>
<td> <font color="white"> <%=sensor2.get(0).toString()%> </font> </td>
<td> <font color="white"> <%=sensor3.get(0).toString()%> </font> </td>
<td> <font color="white"> <%=sensor4.get(0).toString()%> </font> </td>
<td> <font color="white"> <%=sensor5.get(0).toString()%> </font> </td>
<td> <font color="white"> <%=sensor6.get(0).toString()%> </font> </td>
</tr>
```

```
<tr>
<th align="right"><font color="red">ALERT MESSAGE</font></th>
<td> <font color="white"> <%=sensor1.get(1).toString()%> </font> </td>
<td> <font color="white"> <%=sensor2.get(1).toString()%> </font> </td>
<td> <font color="white"> <%=sensor3.get(1).toString()%> </font> </td>
<td> <font color="white"> <%=sensor4.get(1).toString()%> </font> </td>
<td> <font color="white"> <%=sensor5.get(1).toString()%> </font> </td>
<td> <font color="white"> <%=sensor6.get(1).toString()%> </font> </td>
</tr>
<tr>
<th align="right"><font color="red">Limit Value</font></th>
<td> <font color="white"> <%=sensor1.get(2).toString()%> </font> </td>
<td> <font color="white"> <%=sensor2.get(2).toString()%> </font> </td>
<td> <font color="white"> <%=sensor3.get(2).toString()%> </font> </td>
<td> <font color="white"> <%=sensor4.get(2).toString()%> </font> </td>
</tr>
</table>
</td></tr></table>
<br><br>
<form action='./sensorsSetup' method='post' onSubmit='return
CheckParameters3(this.lista1.value,this.txtfld2.value,this.txtfld3.value,this.lista2.va
lue,this.txtfld5.valu
e,this.txtfld6.value,this.lista3.value,this.txtfld8.value,this.txtfld9.value,this.lista4.valu
e,this.txtfld11.val
ue,this.txtfld12.value,this.txtfld13.value,this.txtfld14.value,this.txtfld15.value,this.txtf
ld16.value)'>
<table border='0' cellpadding="5">
<tr>
<th><font color="red">Temperature Sensor 1</font></th>
<td><font color="orange">Limit Value :</font></td>
<td valign='top'>
<select name="lista1" id="lista1">
<option value="0">None</option>
</select><font color="red"> Celcious </font>
```

```
</td>
<td><font color="orange">Alert Message :</font></td>
<td valign='top'><input type='text' size='20' name='txtfld2'
onblur='elegxos(this)'></td>
<td><font color="orange">Info :</font></td>
<td valign='top'> <input type='text' size='20' name='txtfld3' onblur='elegxos(this)'>
</td>
</tr>
<tr>
<th><font color="red">Temperature Sensor 2</font></th>
<td><font color="orange">Limit Value :</font></td>
<td valign='top'>
<select name="lista2" id="lista2">
<option value="0">None</option>
</select>
<font color="red"> Celcious </font>
</td>
<td><font color="orange">Alert Message :</font></td>
<td valign='top'>
<input type='text' size='20' name='txtfld5' onblur='elegxos(this)'>
</td>
<td><font color="orange">Info :</font></td>
<td valign='top'> <input type='text' size='20' name='txtfld6' onblur='elegxos(this)'>
</td>
</tr>
<tr>
<th> <font color="red">Humidity</font> </th>
<td>
<font color="orange">Limit Value :</font>
</td>
<td valign='top'><select name="lista3" id="lista3">
<option value="0">None</option>
</select>
<font color="red"> % </font>
```

```
</td>
<td>
<font color="orange">Alert Message :</font>
</td>
<td valign='top'>
<input type='text' size='20' name='txtfld8' onblur='elegxos(this)'>
</td>
<td>
<font color="orange">Info :</font>
</td>
<td valign='top'>
<input type='text' size='20' name='txtfld9' onblur='elegxos(this)'>
</td>
</tr>
<tr>
<th> <font color="red">AC Voltage Sensor</font> </th>
<td><font color="orange">Limit Value :</font>
</td>
<td valign='top'>
<select name="lista4" id="lista4">
<option value="0">None</option>
</select>
<font color="red"> Voltages </font>
</td>
<td>
<font color="orange">Alert Message :</font>
</td><td valign='top'>
<input type='text' size='20' name='txtfld11' onblur='elegxos(this)'>
</td>
<td>
<font color="orange">Info :</td>
<td valign='top'>
<input type='text' size='20' name='txtfld12' onblur='elegxos(this)'>
</td>
```

```
</tr>
<tr>
<th> <font color="red">Digital Sensor 1</font> </th>
<td>
<font color="orange">Alert Message :</font>
</td>
<td valign='top'>
<input type='text' size='20' name='txtfld13' onblur='elegxos(this)'>
</td>
<td>
<font color="orange">Info :</font>
</td>
<td valign='top'>
<input type='text' size='20' name='txtfld14' onblur='elegxos(this)'>
</td>
</tr>
<tr>
<th> <font color="red">Digital Sensor 1</font> </th>
<td>
<font color="orange">Alert Message :</font>
</td>
<td valign='top'>
<input type='text' size='20' name='txtfld15' onblur='elegxos(this)'></td>
<td>
<font color="orange">Info :</font>
</td>
<td valign='top'>
<input type='text' size='20' name='txtfld16' onblur='elegxos(this)'> </td>
</tr>
</table>
<input type='submit' value='Save'>
<input type="button" onclick="window.location='./home'" value="CANCEL">
</form>
</center>
```

```
</body>
</html>
```

10.17. - Η σελίδα log.jsp

```
<%--

Document : log
Created on : 11 Ιουν 2009, 8:17:46 μμ
Author : ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
<title>JSP Page</title>
<%@ page import="java.sql.*,Utilities.DatabaseUtility"%>
</head>
<body bgcolor='black' font color='white'>
<%@ include file="menu.jsp"%>
<center><h1>LOG/EVENTS TABLE</h1></center>
<table border='1' height='100%' width='100%'>
<%
HttpSession session2 = request.getSession(false);
String parametros = null;
if(session2==null) {
response.sendRedirect("./index.jsp");
}
else {
try{
parametros = session2.getValue("elegxos").toString();
}
catch(Exception e){
response.sendRedirect("./index.jsp");
}
if(parametros!=null) {
if(!parametros.equals("true")) {
response.sendRedirect("./index.jsp");
}
}
}
%>
<%
ResultSet result=null;
ResultSetMetaData dbmetadata=null;
try {
DatabaseUtility.ConnectToDataBaseServer("com.mysql.jdbc.Driver",
"jdbc:mysql://localhost:3306/sensingsystem","root","10022910");
result = DatabaseUtility.processSQL();
```

```
dbmetadata = result.getMetaData();
int arithmoskeliwn = dbmetadata.getColumnCount();
out.println("<tr>");
for(int i=1;i<=arithmoskeliwn;++i){
out.println("<th><font color='red'>" + dbmetadata.getColumnName(i) +
"</font></th>");
}
out.println("</tr>");
while(result.next() ){
out.println("<tr>");
for(int i=1;i<=arithmoskeliwn;i++)
out.println("<td> <font color='white'>" + result.getString(i) + "</font></td>");
out.println("</tr>");
}
out.println("<tr>");
out.println("</table><br><br>");
}
catch(Exception e3){
return;
}
%>
</table>

<center><a href='javascript:history.go(-1)'>BACK</a></center>



</body>
</html>
```

10.18. - Η σελίδα messages.jsp

<%--

Document : messages
Created on : 8 Ιουν 2009, 4:01:43 μμ
Author : ΘΑΝΑΣΗΣ ΚΟΚΚΟΣ – ΑΛΕΞΗΣ ΒΑΦΕΙΑΔΗΣ

--%>

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<%
Object msg = request.getAttribute("msg");
Object topothesia = request.getAttribute("topothesia");
response.addHeader("Refresh","5;"+topothesia.toString() );
%>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">



<title>NOTIFICATIONS</title>
</head>
<body bgcolor="black">
<center>
<font color="red" size="18"> <%=msg%>
<br><br><br><br><br>
<h3>YOU WOULD AUTOMATICALLY REDIRECT <br>
INTO : <%=topothesia%> IN 5 SECONDS!!!!</h3>
IF YOUR BROWSER DOESNT SUPPORT REDIRECTION
<a href="<%=topothesia%>" style="color:orange"> CLICK HERE </a>
</font>
</center>
</body>
</html>
```

10.19. - Αρχείο με τις συναρτήσεις JAVASCRIPT που χρησιμοποιούν οι σελίδες JSP

```
/* Μέθοδος για χρήση στις φόρμες της διαδικτυακής εφαρμογής.
   Ελέγχουμε αν ο χρήστης έκανε focus σε ένα πεδίο και έπειτα έφυγε από
   αυτό χωρις να εισάγει κάτι.
   Σε αυτήν την περίπτωση το πεδίο παίρνει κόκκινο χρώμα.
*/
function elegxos(obj){
    if(obj.value=='')
        obj.style.backgroundColor='red';
     else
        obj.style.backgroundColor='white';
```

```
 }

/* Μέθοδος για τον έλεγχο του περιεχομένου σε πεδία τύπου email.
 * Το email πρέπει να είναι της μορφής κάτι@κάτι.κάτι.
*/
function elegxosEmail(emailTest){
    var protEmail = /^..*@..*\...*$/;

    return protEmail.test(emailTest);
}

/* Μέθοδος για τον έλεγχο του περιεχομένου σε πεδία τύπου mobile number
 * Ο αριθμός πρέπει να είναι της μορφής 69xxxxxxxx ή +3069xxxxxxxx
*/
function elegxosMobile(mobileTest){

    var protMobile = /^69\d{8}$/;
    var protMobile2 = /^\+3069\d{8}$/;

    return ( protMobile.test(mobileTest) || protMobile2.test(mobileTest) );
}

/* Μέθοδος για τον έλεγχο του περιεχομένου σε πεδία τύπου ssl πόρτας
 * Ο αριθμός της πόρτας πρέπει να είναι ακέραιος αριθμός από 1 έως 5
 * ψηφία.
*/
function elegxosSslPort(sslPortTest){
```

```
    var protSslPort = /^\d{1,5}$/;


    return ( protSslPort.test(sslPortTest) );
}
```

/* Μέθοδος για χρήση στην οθόνη σύνδεσης του χρήστη στη διαδικτυακή εφαρμογή.

Ελέγχουμε αν ο χρήστης έχει εισάγει στοιχεία (όνομα χρήστη, κωδικό

*** πρόσβασης) ώστε να συνδεθεί.**

Αν ο χρήστης άφησε κενό κάποιο από τα 2 πεδία εμφανίζεται ανάλογο

*** μήνυμα.**

***/**

```
function CheckParameters(webuser,webpass){
    if((webuser == '')||(webpass == '')){
        alert('Enter username and password');
        return false;
    }
    return true;
}
```

/* Μέθοδος για χρήση στην οθόνη διαχείρισης των ιδιοτήτων του διαχειριστή.

*** Αρχικά ελέγχουμε αν έχουν εισαχθεί το υπάρχων όνομα χρήστη και ο υπάρχων**

*** κωδικός πρόσβασης, ‘ώστε να προχωρήσουμε σε αλλαγές που αφορούν το**

*** λογαριασμό του διαχειριστή.**

*** Στη συνέχεια αν απαιτείται κάποια τροποποίηση του λογαριασμού του**

```
* διαχειριστή
*/
function CheckParametersAdmin(txtfld1,txtfld2,txtfld3,txtfld4,hiddenField){
   // Έλεγχος username kai password.
   if((txtfld1 == '')||(txtfld2 == '')){
      alert('Enter username and password');
      return false;
   }

   if (hiddenField=="false"){

 // Έλεγχος αν απαιτείται αλλαγή του αριθμού του κινητού τηλεφώνου ή του email.

      if((txtfld3 == '')&&(txtfld4 == '')){
         alert("You entered nothing to be saved!");
         return false;

      }else { // Έλεγχος των τιμών που εισήχθησαν

         var olaOK = true;
         var alertmsg="";

         if (txtfld3 != '') {
      // Έλεγχος αν το κινητό είναι της μορφής 69xxxxxxxx ή +3069xxxxxxxx
            if ( !elegxosMobile(txtfld3) ) {
               alertmsg = "Mobile number should be in format: \"69xxxxxxxx\" or \"+3069xxxxxxxx\"\n"
```

```
                olaOK=false;
            }
        }

        if (txtfld4 != '') {
            // Έλεγχος αν το email είναι της μορφής κάτι@κάτι.κάτι
            if ( !elegxosEmail(txtfld4) ) {
                alertmsg += "Email address should be in format: \"something@something.something\""
                olaOK=false;
            }
        }

        if (!olaOK) {
            alert(alertmsg);
            return false;
        }
        else return true;
    }

} else {

// Έλεγχος αν απαιτείται αλλαγή του username ή του password.

    var user = document.getElementById("txtfld5").value;
    var pass = document.getElementById("txtfld6").value;
```

```
 if((user == '')&&(pass == '')){
    alert("Enter new username or new password! Else close this Form");
    return false;
 }

 else {

/* Έλεγχος αν εκτός από την αλλαγή  username και password απαιτείται
  * και αλλαγή του email ή του κινητού τηλεφώνου.
 */

// έλεγχος των τιμών που εισήχθησαν.

    var olaOK2 = true;
    var alertmsg2="";

    if (txtfld3 != '') {
  // Έλεγχος αν το κινητό είναι της μορφής 69xxxxxxxx ή +3069xxxxxxxx
        if ( !elegxosMobile(txtfld3) ) {
            alertmsg2 = "Mobile number should be in format: \"69xxxxxxxx\" or \"+3069xxxxxxxx\"\n"
            olaOK2=false;
        }
    }

    if (txtfld4 != '') {
        // Έλεγχος αν το email είναι της μορφής κάτι@κάτι.κάτι
```

```
            if ( !elegxosEmail(txtfld4) ) {
                alertmsg2 += "Email address should be in format:
\"something@something.something\""
                olaOK2=false;
            }
        }

        if (!olaOK2) {
            alert(alertmsg2);
            return false;
        }
        else return true;
    }

  }
}
```

/* Μέθοδος για χρήση στην οθόνη διαχείρισης των ιδιοτήτων του

*** λογαριασμού**

*** ηλεκτρονικού ταχυδρομείου του συστήματος επιτήρησης**

*** Έλεγχος αν απαιτείται κάποια τροποποίηση των ιδιοτήτων αυτών.**

***/**

```
function
CheckParametersMail(emailField,smtpField,mailUserField,mailPassField,sslField){
  if((emailField == '')||(smtpField == '')||(mailUserField == '')||(mailPassField == '')){
    alert('You entered nothing to be saved!');
    return false;
```

```
    } else {


        var olaOK3 = true;

        var alertmsg3="";


        if (emailField != '') {

            // Έλεγχος αν το email είναι της μορφής κάτι@κάτι.κάτι

            if ( !elegxosEmail(emailField) ) {

                alertmsg3 += "Email address should be in format:
\"something@something.something\"\n"

                olaOK3=false;

            }

        }


        if (sslField != '') {

            // Έλεγχος αν η πόρτα είναι ακέραιος αριθμός από 1 έως 5 ψηφία

            if ( !elegxosSslPort(sslField) ) {

                alertmsg3 += "SSL Port should be in range 0-65535"

                olaOK3=false;

            }

            else {

                if ( sslField<0 || sslField>65535 ) {

                    alertmsg3 += "SSL Port should be in range 0-65535"

                    olaOK3=false;

                }

            }

        }
```

```
        if (!olaOK3) {
            alert(alertmsg3);
            return false;
        }
        else return true;
    }
}

/* Μέθοδος για χρήση στην οθόνη τροποποίησης των ιδιοτήτων των
*  αισθητήρων. Γεμίζει τα μενού με τις τιμές-όρια των αισθητήρων.
*/
function gemismaLimits() {
    for (j=1;j<=150;j++){
        var optionObj1 = document.createElement("OPTION");
        var optionObj2 = document.createElement("OPTION");
        var optionObj3 = document.createElement("OPTION");
        optionObj1.text=""+j;
        optionObj2.text=""+j;
        optionObj2.value=""+j;
        optionObj2.value=""+j;
        optionObj3.text=""+j;
        optionObj3.value=""+j;
        document.getElementById("an1LimitList").appendChild(optionObj1);
        document.getElementById("an2LimitList").appendChild(optionObj2);
        if ((j>=1)&&(j<=100)){
            document.getElementById("an3LimitList").appendChild(optionObj3);
```

```
        }


    }
    for (j=100;j<=250;j++){
        var optionObj4 = document.createElement("OPTION");
        optionObj4.text=""+j;
        optionObj4.value=""+j;
        document.getElementById("an4LimitList").appendChild(optionObj4);
        j++;
    }


}
```

/* Μέθοδος για χρήση στην οθόνη τροποποίησης των ιδιοτήτων των

*** αισθητήρων. Έλεγχος αν απαιτείται κάποια τροποποίηση των ιδιοτήτων του.**

***/**

```
function
CheckParametersSensors(an1InfoField,an1MessageField,an1LimitList,an2InfoFiel
d,an2MessageField,an2LimitList,


an3InfoField,an3MessageField,an3LimitList,an4InfoField,an4MessageField,an4Li
mitList,


dig1InfoField,dig1MessageField,dig2InfoField,dig2MessageField){


    if (
(an1InfoField=="")&&(an1MessageField=="")&&(an1LimitList=="0")&&(an2InfoFiel
d=="")&&(an2MessageField=="")&&(an2LimitList=="0")&&
```

```
(an3InfoField=="")&&(an3MessageField=="")&&(an3LimitList=="0")&&(an4InfoFiel
d=="")&&(an4MessageField=="")&&(an4LimitList=="0")&&

(dig1InfoField=="")&&(dig1MessageField=="")&&(dig2InfoField=="")&&(dig2Messa
geField=="") ) {

        alert('You entered nothing to be saved!');
        return false;
    }
    return true;
}
```

/* Μέθοδος για χρήση στην οθόνη διαχείρισης των ιδιοτήτων του διαχειριστή.

***Με αυτήν την μέθοδο εμφανίζονται επιπλέον επιλογές για την**

*** τροποποίηση του ονόματος χρήστη και του κωδικού πρόσβασης του**

*** διαχειριστή.**

***/**

```
function showUserPassFields() {
    var checkHidden = document.getElementById("hiddenField").value;
    if(checkHidden=="false"){
        document.getElementById("changeUserAndPass").innerHTML="<br>"+
            "<table border='0' cellpadding='3'>"+
                "<tr><td align='right'><font color='white'>New Web Username
:</font></td><td valign='top'> <input type='text' size='20' id='txtfld5' name='txtfld5'
onblur='elegxos(this)'> </td></tr>"+
                "<tr><td align='right'><font color='white'>New Web Password
:</font></td><td valign='top'> <input type='password' size='20' id='txtfld6'
name='txtfld6' onblur='elegxos(this)'> </td></tr>"+
            "</table><br>";
```

```
        document.getElementById("hiddenField").value="true";
    }else{
        document.getElementById("changeUserAndPass").innerHTML="";
        document.getElementById("hiddenField").value="false";
    }
}
```

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στόχος της εργασίας ήταν η σχεδίαση και ανάπτυξη ενός ολοκληρωμένου συστήματος επιτήρησης το οποίο παρακολουθεί τις περιβαλλοντολογικές συνθήκες ενός δωματίου υπολογιστών (Computer/Server Room) με σκοπό την άμεση παρέμβαση όταν αυτές δεν είναι κανονικές.

Το σύστημα επιτήρησης που υλοποιήθηκε διαθέτει χαρακτηριστικά που ακολουθούν τις αρχικές προδιαγραφές του σχεδιασμού και τις καλύπτουν πλήρως. Ειδικότερα, διαθέτει δυνατότητα παρακολούθησης και καταγραφής των περιβαλλοντολογικών συνθηκών και δυνατότητες ειδοποίησης του διαχειριστή σε συνθήκες απειλής των υποδομών καθώς και αποστολής ημερήσιας αναφοράς. Επίσης, έχει δυνατότητα παρουσίασης των μετρήσεων σε πραγματικό χρόνο καθώς και δυνατότητα προβολής στατιστικών διαγραμμάτων των μετρήσεων.

Στην εργασία αναπτύχθηκαν τα αναγκαία εργαλεία λογισμικού, έγινε προσομοίωση της λειτουργίας του συστήματος σε επίπεδο υλικού και λογισμικού και ολοκληρώθηκαν με επιτυχία οι λειτουργίες παρακολούθησης, καταγραφής και αποστολής αναφορών μέσω ηλεκτρονικού ταχυδρομείου και προειδοποιήσεων μέσω κινητού τηλεφώνου στον διαχειριστή.

Ολοκληρώθηκαν οι λειτουργίες του συστήματος που αφορούν στον υπολογιστή. Η παραλαβή των μετρήσεων από την εξωτερική συσκευή, η παρουσίαση διαγραμματικά των τιμών αυτών, ο έλεγχος τους και τέλος, η ειδοποίηση του διαχειριστή με μήνυμα ηλεκτρονικού ταχυδρομείου (email) και γραπτού μηνύματος στο κινητό του τηλέφωνο (SMS), υλοποιήθηκαν πλήρως. Επιπλέον ενσωματώθηκε στην εφαρμογή η δυνατότητα αποστολής μέσω ηλεκτρονικού ταχυδρομείου αρχείου pdf της ημερήσιας αναφοράς με τα στατιστικά διαγράμματα των μετρήσεων.

Η παρακολούθηση των περιβαλλοντολογικών συνθηκών επιτυγχάνεται μέσω διάταξης η οποία διαθέτει αισθητήρια όργανα τοποθετημένα σε επιλεγμένα σημεία του δωματίου υπολογιστών. Η συσκευή συλλέγει μετρήσεις από τους αισθητήρες και τις αποστέλλει στον ηλεκτρονικό υπολογιστή. Για τις ανάγκες της

παρούσας εργασίας η λειτουργία της συγκεκριμένης διάταξης προσομοιώθηκε με τη βοήθεια κατάλληλου λογισμικού στον Η/Υ. Η κατασκευή της διάταξης που αποτελεί μελλοντικό στόχο και ενδεχομένως αντικείμενο μιας άλλης εργασίας θα επιτρέψει την εγκατάσταση ενός πρωτοτύπου του συστήματος παρακολούθησης σε περιβάλλον δωματίου υπολογιστών και την αξιολόγηση του σε πραγματικές συνθήκες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία (Ηλεκτρονικά και Μη)

Αλεξόπουλος, Α. και Λαγογιάννης, Γ. (2003), Τηλεπικοινωνίες και Δίκτυα Υπολογιστών, *Αυτοέκδοση*, Αθήνα

Basham, B., Sierra, K. and Bates, B. (2008), Head First Servlets and JSP, Second Ediion, *O'Reilly Media Inc.*, USA

Bates, M. (2006), Interfacing PIC Microcontrollers Embedded Design by Interactive Simulation, *Newnes (Elsevier)*, Oxford, UK

Darwin. I. (2001), Java Cookbook, *O'Reilly Media Inc*., USA

Gardner, N. (1998), A Beginners Guide to the Microchip PIC, *Bluebird Technical Press,* Arizona, USA

Goodman, D. (2007), JavaScript and DHTML Cookbook, 2nd Edition, *OReilly Media Inc*., USA

Lowagie, B. (2007), iText in Action Creating and Manipulating pdf, *Manning Publications*, USA

Lowagie, B. and Soares, P. (2009), iText A free java-pdf library, on-line, Gent, BELGIUM http://www.lowagie.com/iText/docs.html [Σάββατο, 29 Αυγούστου 2009]

Loyd, I. (2008), The Ultimate HTML Reference, *Sitepoint*, USA

Matic, N. (2003), PIC microcontrollers, 3rd edition, *microEleKtronika*, book on-line http://www.mikroe.com/en/books/picbook/0_Uvod.htm [Σάββατο, 29 Αυγούστου 2009]

Mordani, R. (2008), Java™ Servlet Specification, Version 3.0, *Sun Microsystems,* USA

Perry, B.W. (2004), Java Servlet & JSP Cookbook, *O'Reilly Media Inc.,* USA

Διαδικτυακοί Τόποι [Σάββατο, 29 Αυγούστου 2009]

Παραδείγματα εφαρμογών με Pic
http://www.electronics-lab.com/pic-in-greek/samples/gallery/PIC_Circuits_Gallery.htm

Microchip Technology Inc
http://www.microchip.com

Maxim Integrated Products
http://www.maxim-ic.com

PIC16F87X Data Sheet, 28/40-Pin 8-Bit CMOS FLASHMicrocontrollers
http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

PIC16F87XA Data Sheet 28/40/44-Pin Enhanced Flash Microcontrollers
http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf

MPLAB ICD & PIC 16F877 tutorial
http://www.eng.uwi.tt/depts/elec/staff/feisal/ee25m/resources/ee25m-lect3b.pdf

Hall, M. and Brown, L., Core Servlets and JavaServer Pages
http://courses.coreservlets.com

Proteus VSM homepage
http://www.labcenter.co.uk/products/vsmoverview.cfm

The official API documentation for JFreeChart
http://www.jfree.org/jfreechart/api.html

Parekh, A., (2007), Temperature Logger using DS1820 and PIC 16L84 Microcontroller http://hackedgadgets.com/2007/04/02/temperature-logger-using-ds1820-and-pic-16l84-microcontroller/

A PIC-Based Temperature Alarm
http://192.197.62.35/staff/mcsele/TemperatureAlarm.htm

Quozl's Temperature Sensor Project
http://james.tooraweenah.com/moin/ts

Program in C for PIC
http://www.phanderson.com/PIC/PICC/ee_sv.html

Serial Port Temperature Sensors
http://martybugs.net/electronics/tempsensor/]

Introduction to Microcontrollers
http://www.mikroe.com/en/books/picbook/1_chapter.htm

PIC and PIC16 instruction
http://pic-in-greek.freewebpages.org/GUIDE/INSTRUCTIONS/Instruction%20spec%20(1)%20of%20PIC16%20series.htm

PIC Instruction
http://tutor.al-williams.com/pic-inst.html

PIC Tutorial
http://www.mstracey.btinternet.co.uk/pictutorial/picmain.htm

PIC Book
http://www.picbook.com/index.html

Delay code generator for PIC
http://www.piclist.com/techref/piclist/codegen/delay.htm

CSS Navigation Menu
http://www.jessett.com/web_sites/dhtml/create_dhtml_menu.shtml

Menu and Navigation
http://www.dynamicdrive.com/dynamicindex1/

How to Send SMS Messages from a Computer / PC
http://www.developershome.com/sms/howToSendSMSFromPC.asp

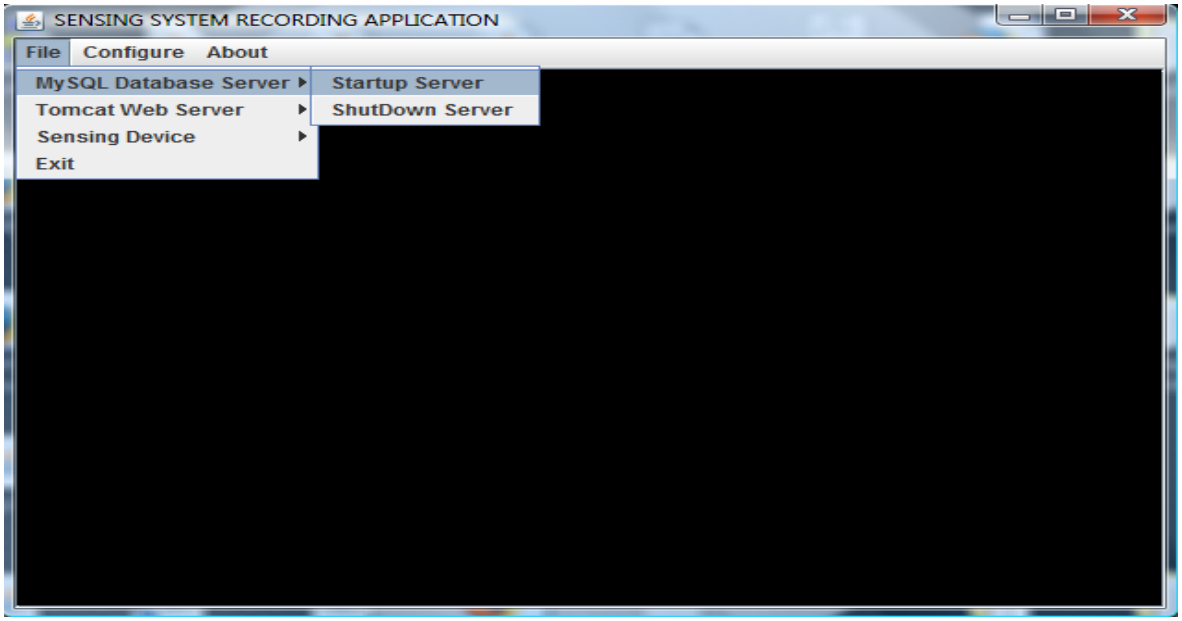
Using AT commands to send and read SMS
http://wiki.forum.nokia.com/index.php/Using_AT_commands_to_send_and_read_SMS

MySQL Creating an encrypted password field
http://lists.mysql.com/mysql/102722

Java Mail and SMTP over SSL
http://www.mail-archive.com/jdjlist@mailbox.sys-con.com/msg00347.html
http://forums.sun.com/thread.jspa?threadID=5267916

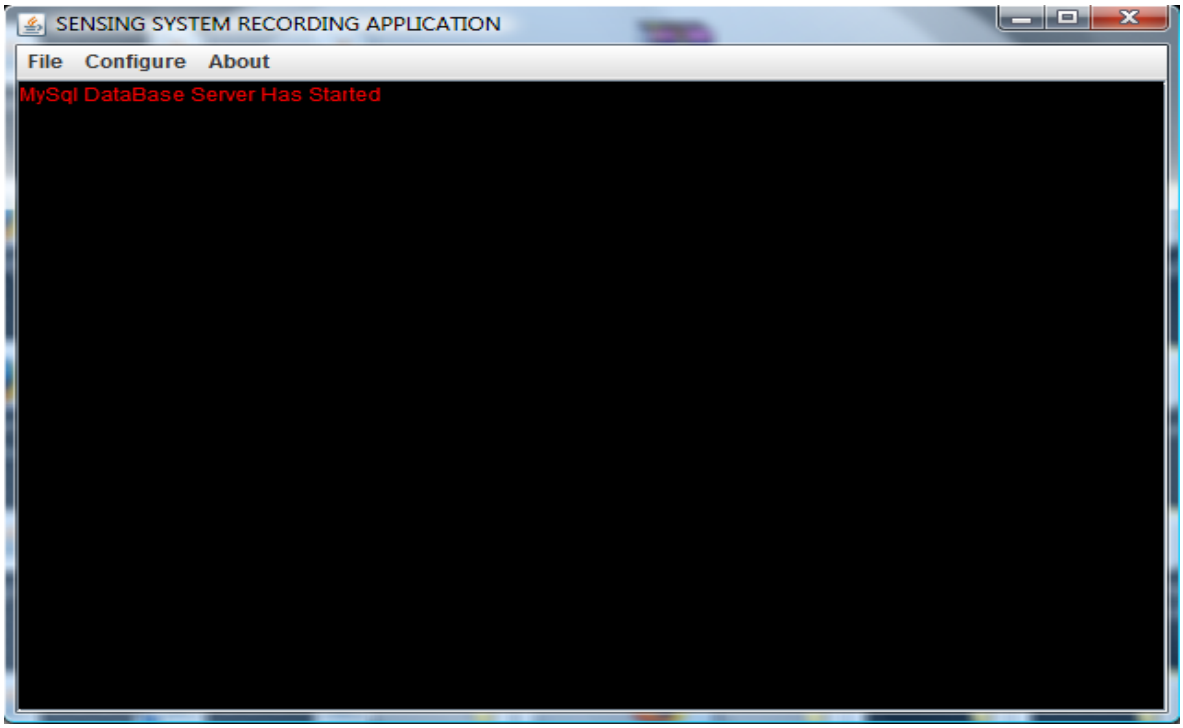
Παράρτημα Α
Εγχειρίδιο διάδρασης με την εφαρμογή καταγραφής

Η πολυνηματική εφαρμογή παρακολούθησης και καταγραφής των περιβαλλοντολογικών συνθηκών δίνει την δυνατότητα στον χρήστη της εφαρμογής, να εκκινήσει τον διακομιστή βάσεων δεδομένων MySQL, αλλά και τον διαχειριστή της υπηρεσίας του διακομιστή ιστού Apache-Tomcat, μέσα από το δικό της περιβάλλον, χωρίς να απαιτείται από μέρους του χρήστη η εκκίνηση των διακομιστών χειροκίνητα, είτε από κάποια συντόμευση, είτε από την γραμμή εντολών των windows. Ο χρήστης επιλέγει “File” → “MySQL Database Server” → “Startup Server”



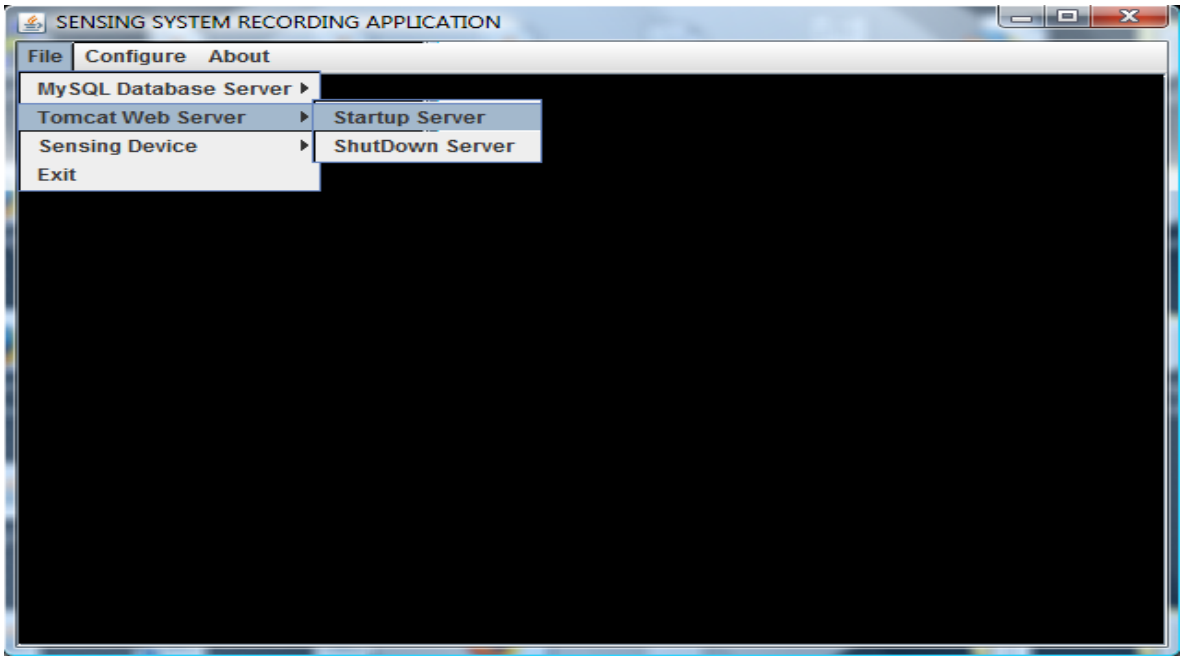
Σχήμα 11.1.α – Εκκίνηση διακομιστή βάσεων δεδομένων MySQL

για την εκκίνηση του διακομιστή βάσεων δεδομένων MySQL και

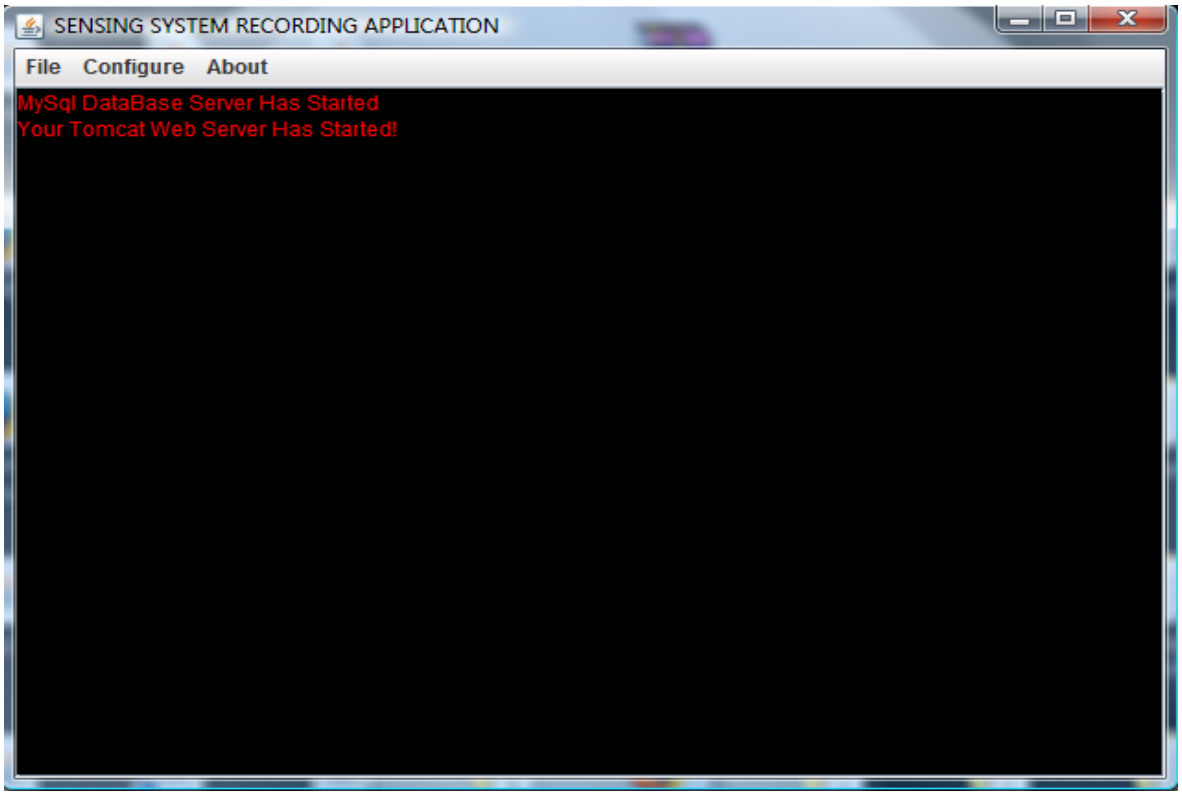


Σχήμα 11.1.b – Εκκίνηση διακομιστή βάσεων δεδομένων MySQL

“File” → “Tomcat Web Server ” → “Startup Server” για την εκκίνηση του διακομιστή ιστού Apache-Tomcat.



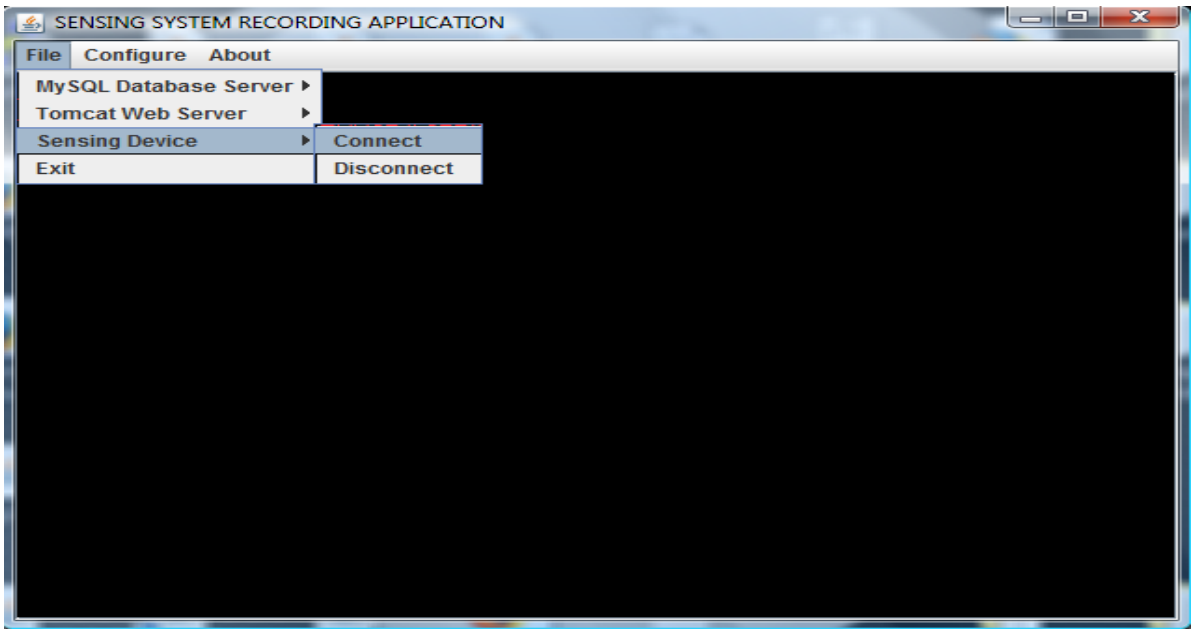
Σχήμα 11.2.a – Εκκίνηση διακομιστή εφαρμογών ιστού Tomcat



Σχήμα 11.2.b – Εκκίνηση διακομιστή εφαρμογών ιστού Tomcat

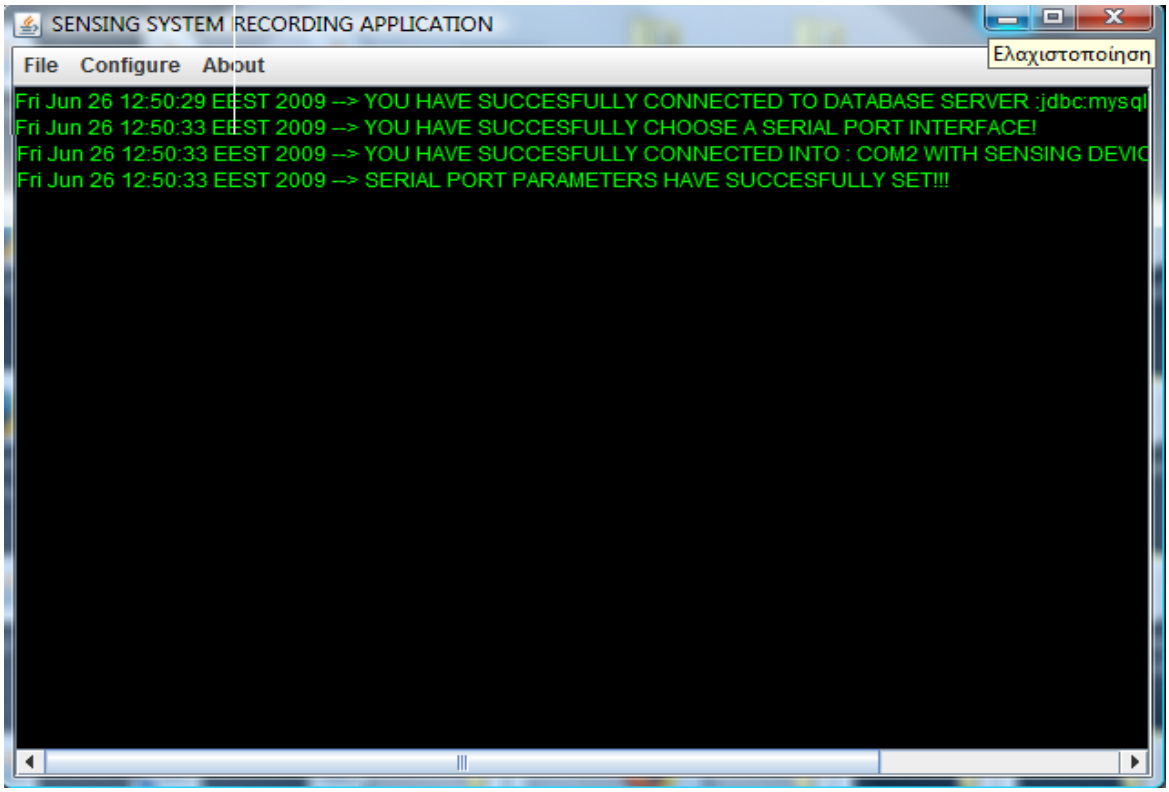
Έπειτα ο χρήστης μπορεί να συνδεθεί με τη συσκευή των αισθητήρων, διαμέσου οποιασδήποτε σειριακής διασύνδεσης, απλώς επιλέγοντας “File” → “Sensing Device” → “Connect” και η εφαρμογή θα συνδεθεί στη βάση

δεδομένων, θα διαβάσει τις αποθηκευμένες παραμέτρους επικοινωνίας για αυτή την συσκευή και θα εκκινήσει την διεργασία της επικοινωνίας μαζί της.



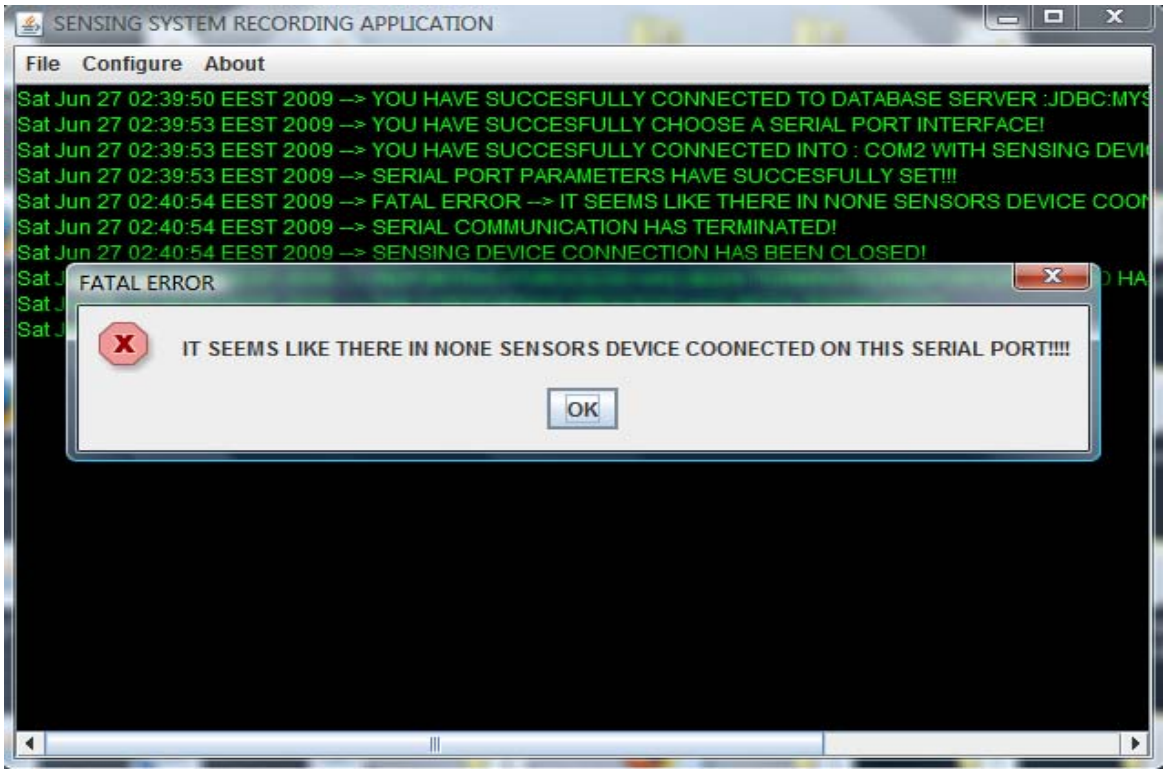
Σχήμα 11.3.α – Σύνδεση με την εξωτερική συσκευή

Εάν όντως υπάρχει κάποια συσκευή συνδεδεμένη με σειριακό καλώδιο RS-232 σε κάποια από τις σειριακές θύρες του υπολογιστή, τότε εκκινείται αυτομάτως η διεργασία ανάγνωσης των δεδομένων που στέλνει η συσκευή, με βάση τις αποθηκευμένες παραμέτρους επικοινωνίας για αυτή την συσκευή. Κάθε τέσσερα δευτερόλεπτα, διαβάζονται 19 bytes πληροφορίας, ελέγχονται για την ορθότητα τους και έπειτα συντελείται ο διαχωρισμός της πληροφορίας για κάθε αισθητήρα, αναλογικό, είτε ψηφιακό, αλλά και η αποθήκευση των μετρήσεων στη βάση δεδομένων, με βάση το προσδιοριστικό όνομα κάθε αισθητήρα και την χρονική στιγμή λήψης της τιμής από την συσκευή των αισθητήρων. Επίσης εκκινείται ένα ακόμη νήμα το οποίο είναι υπεύθυνο για να παρακολουθεί την κατάσταση (status) ειδοποίησης για τους αισθητηρές, ώστε να αποφασίζει εάν ειδοποιήθηκε ο χρήστης για κάποιο αισθητήρα και εάν όχι να του αποστέλλει ειδοποίηση, ενώ εάν έχει ήδη ειδοποιηθεί, δεν πραγματοποιεί ξανά τη διαδικασία.



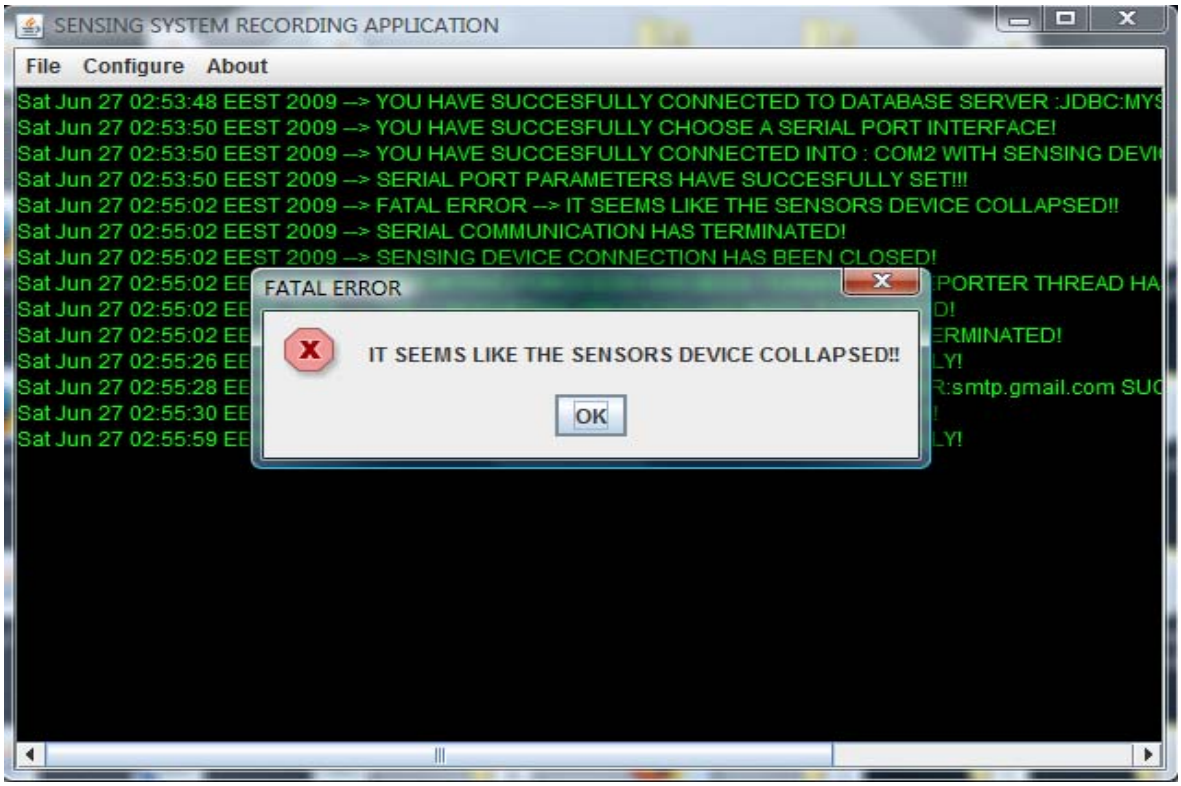
Σχήμα 11.3.b – Σύνδεση με την εξωτερική συσκευή

Εάν τώρα έχει επιλεχθεί κάποια σειριακή πόρτα στην οποία δεν υπάρχει κάποια συσκευή συνδεδεμένη μαζί της, τότε μετά την πάροδο ενός λεπτού, βγαίνει παράθυρο διαλόγου που ειδοποιεί τον χρήστη της εφαρμογής ότι δεν υπάρχει καμία συσκευή συνδεδεμένη στην θύρα που έχει επιλεχθεί και τερματίζονται τα νήματα που είχαν εκκινήσει.



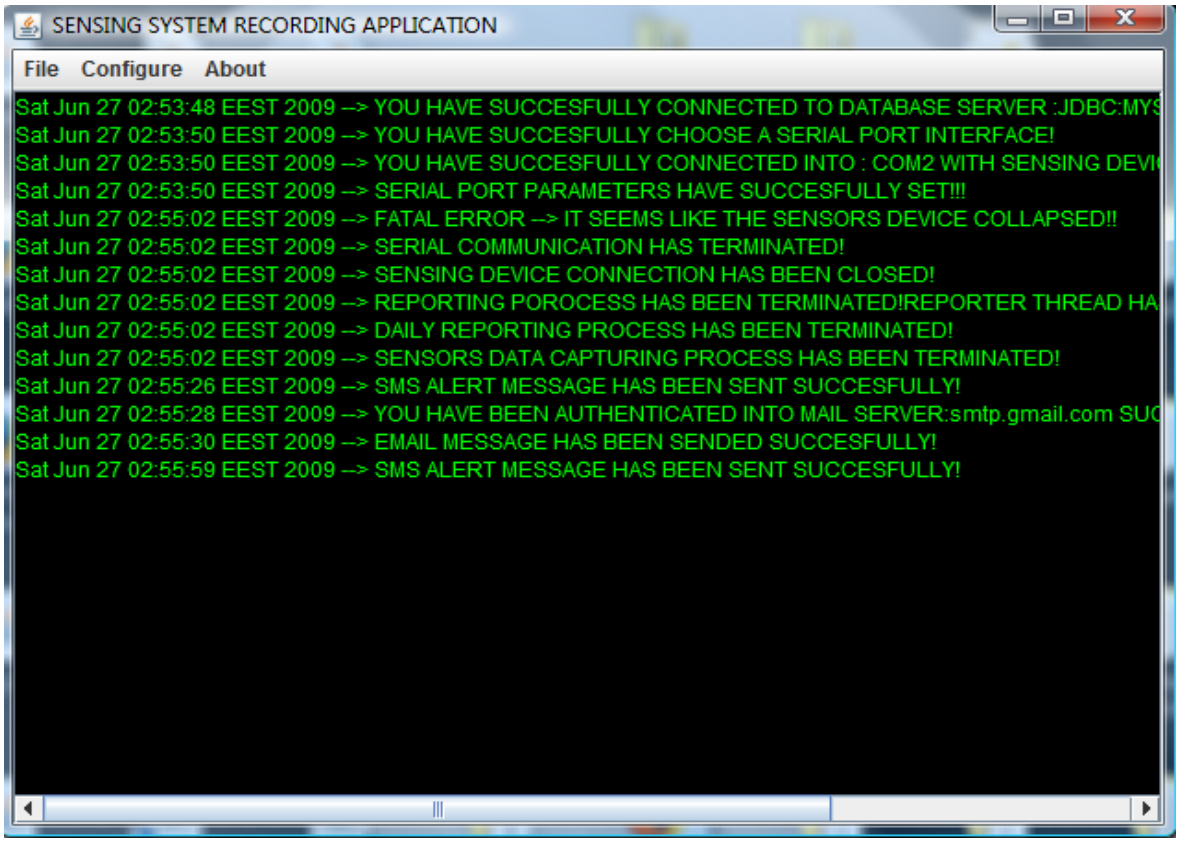
Σχήμα 11.4 – Αδυναμία επικοινωνίας με την εξωτερική συσκευή

Εάν η εφαρμογή συνδεθεί κανονικά με την εξωτερική συσκευή και εκκινήσει επιτυχώς και βρίσκεται σε πλήρη εξέλιξη η κύρια διεργασία της καταγραφής των μετρήσεων που στέλνονται από αυτή, αλλά για οποιοδήποτε λόγο διακοπεί η επικοινωνία, τότε ο διαχειριστής ειδοποιείται με παράθυρο διαλόγου, αλλά και επίσης με μήνυμα ηλεκτρονικής αλληλογραφίας και γραπτό μήνυμα στο κινητό του για το κρίσιμο αυτό σφάλμα.



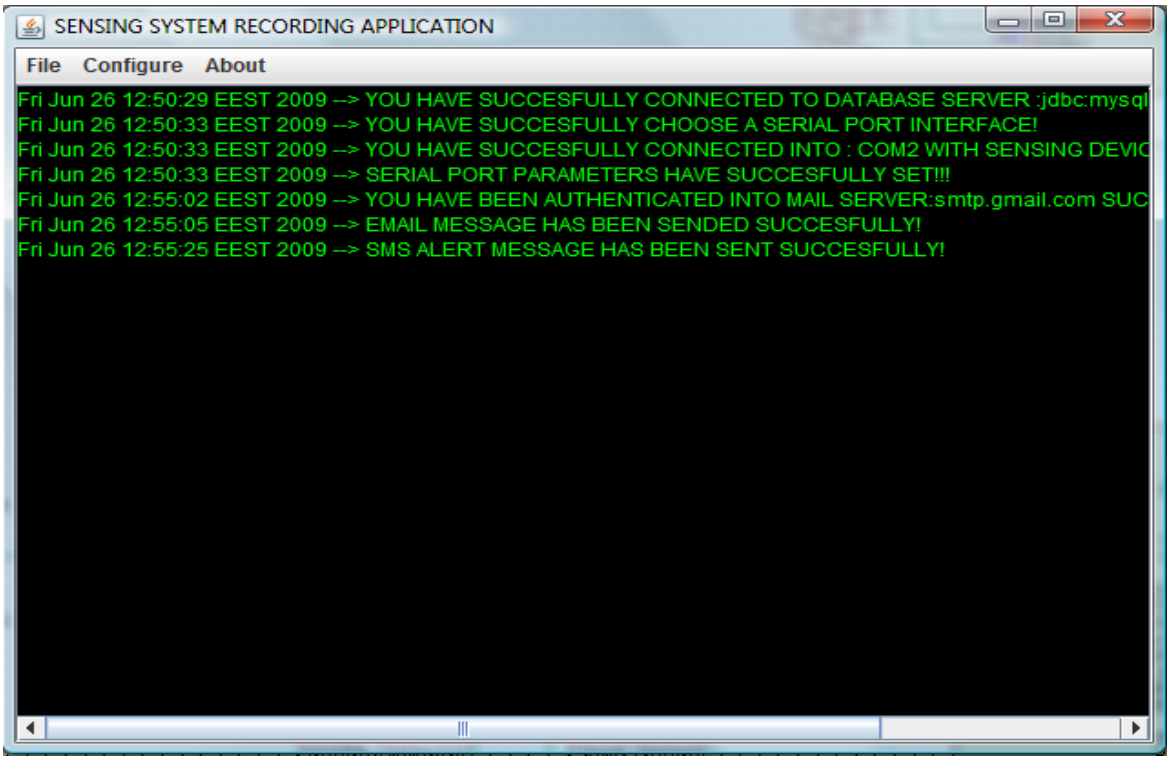
Σχήμα 11.5.a – Σφάλμα κατά την επικοινωνία με την εξωτερική συσκευή

Η συσκευή με τους αισθητήρες μπορεί να βραχυκύκλωσε, ακόμα και να κάηκε, για αυτό και η εφαρμογή τερματίζει τα νήματα που εκκίνησαν κατα τη σύνδεση με την συσκευή ώστε να αποφύγει το φαινόμενο του αδιεξόδου των νημάτων.



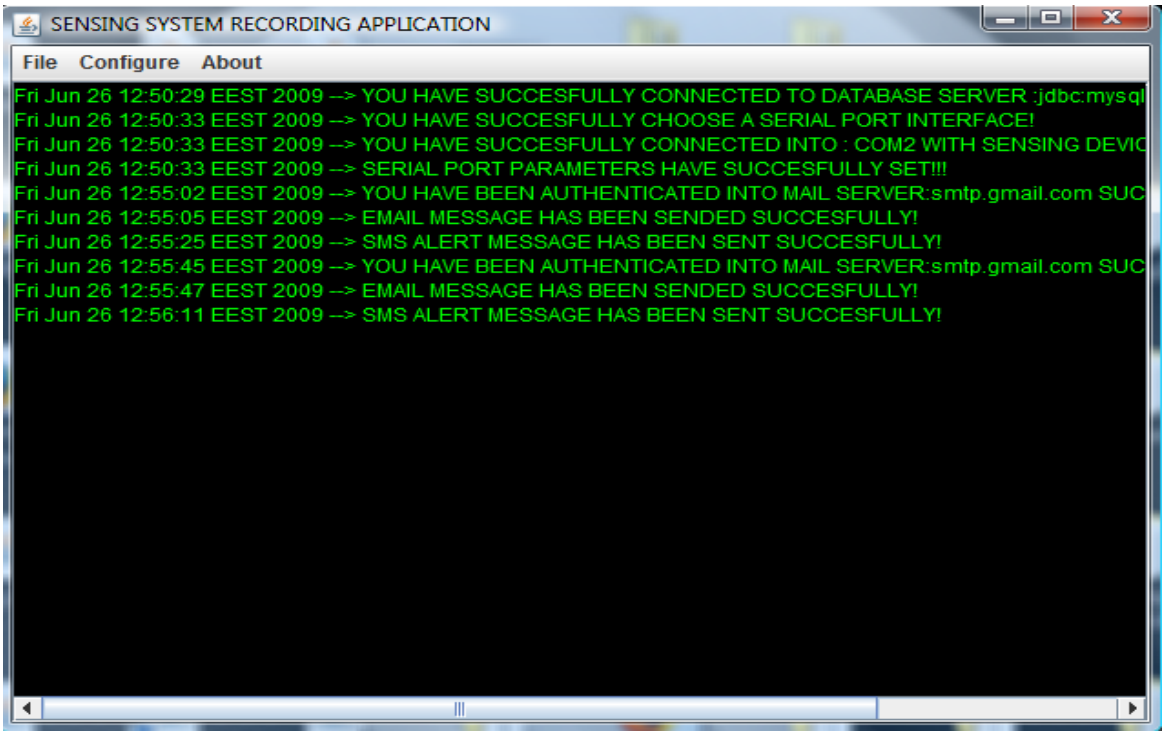
Σχήμα 11.5.b – Σφάλμα κατά την επικοινωνία με την εξωτερική συσκευή

Εάν οι τιμές αυτές που αφορούν τον κάθε αισθητήρα ξεπεράσουν το προβλεπόμενο και αποδεκτό όριο που έχει θέσει ο ίδιος ο διαχειριστής του συστήματος, μέσω της διαδικτυακής εφαρμογής μας, η εφαρμογή αναλαμβάνει να τον ενημερώσει άμεσα μέσω ηλεκτρονικού ταχυδρομείου αλλά και μέσω γραπτού μηνύματος, στο λογαριασμό και στο κινητό τηλέφωνο, που ο ίδιος έχει ορίσει μέσω της διαδικτακής εφαρμογής.



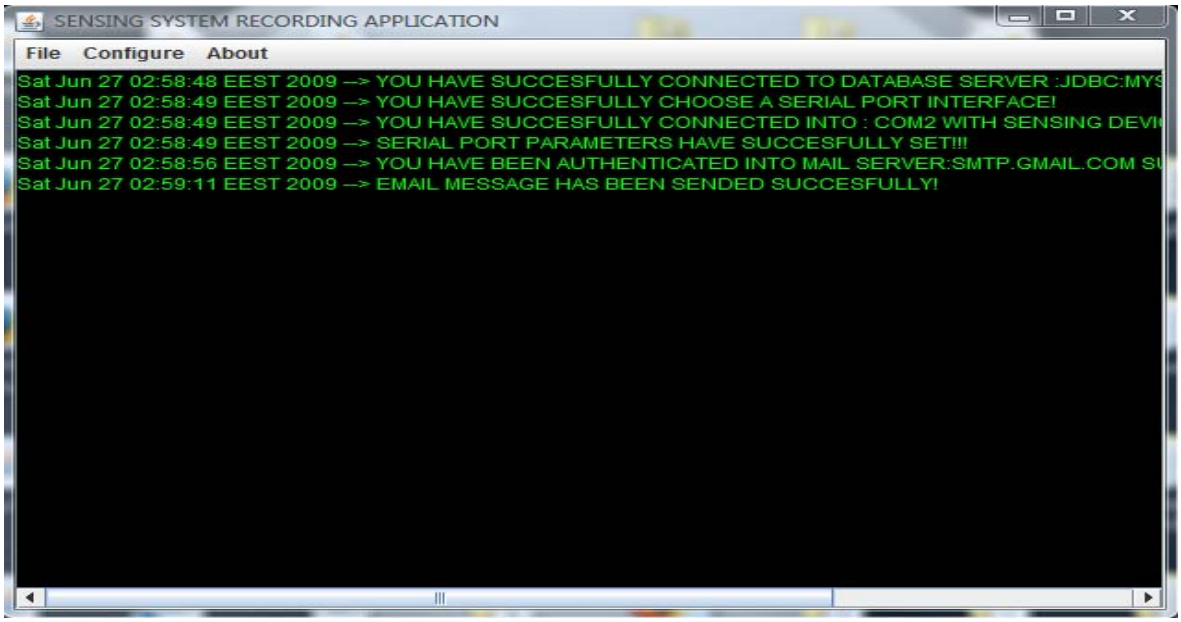
Σχήμα 11.6.a – Ειδοποίηση διαχειριστή με sms και email για κάποιον αισθητήρα

Εάν ο αισθητήρας στείλει μέτρηση στην αμέσως επόμενη χρονική στιγμή της μέτρησης (μετά από κάποια δευτερόλεπτα) η οποία βρισκόταν πάνω από όριο, που είναι κάτω από το όριο, τότε ο χρήστης ειδοποιείται εκ νέου με μήνυμα ηλεκτρονικού ταχυδρομείου, αλλά και με γραπτό μήνυμα στο κινητό του τηλέφωνο, αλλά αυτή τη φορά με μήνυμα ότι η μέτρηση αυτού/των του/των αισθητήρα/ρων επανήλθε/αν στο φυσιολογικό εύρος τιμών.



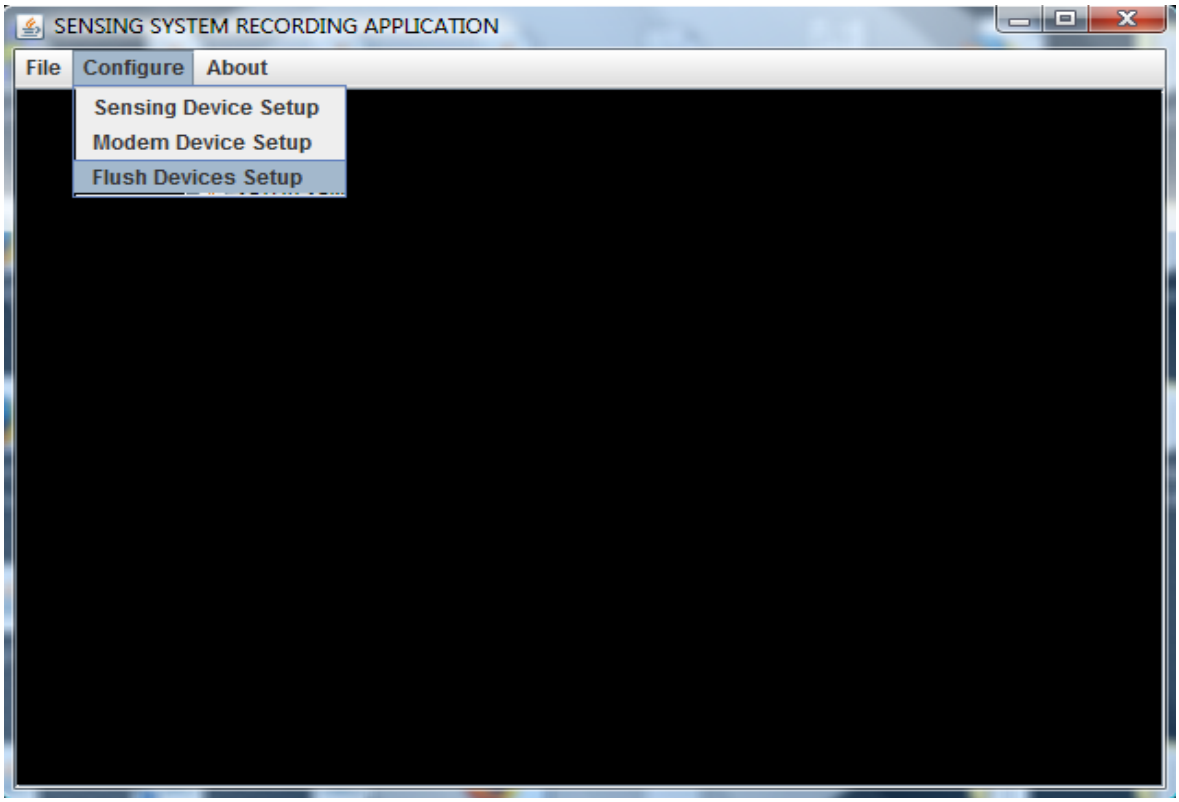
Σχήμα 11.6.b – Ειδοποίηση διαχειριστή με sms και email για κάποιον αισθητήρα

Ταυτόχρονα με την εκκίνηση του νήματος σύνδεσης με την συσκευή των αισθητήρων, αλλά και του νήματος παρακολούθησης των status ενημέρωσης και αποστολής ειδοποίησης για κάθε αισθητήρα, εκκινούμε και ένα ακόμα νήμα το οποίο πραγματοποιεί την εργασία του, μετά από τόσα χιλιοστά του δευτερολέπτου, όσα αυτά που απέχουν εως τα μεσάνυχτα. Το νήμα αυτό έχει την ευθύνη της δημιουργίας 12 στατιστικών διαγραμμάτων με όλες τις μετρήσεις και των τεσσάρων αισθητήρων, για την τρέχουσα ημέρα, εβδομάδα, μήνα και έτος, αλλά και την αποστολή τους ως συννημένα αρχεία σε ηλεκτρονικο μήνυμα στον λογαριασμό ηλεκτρονικού ταχυδρομείου του διαχειριστή της εφαρμογής.



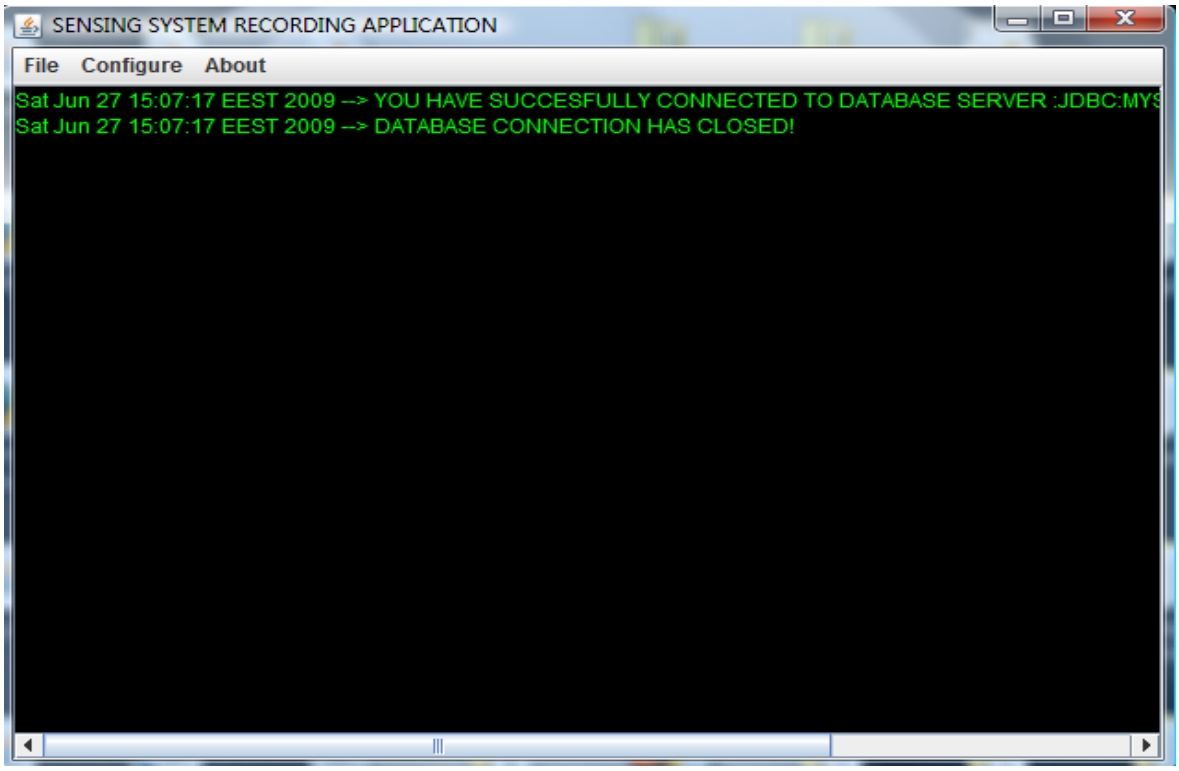
Σχήμα 11.7 – Αποστολή ημερήσιας αναφοράς με τα στατιστικά διαγράμματα ολων των μετρήσεων

Ο χρήστης δύναται να κάνει εκκαθάριση των αποθηκευμένων συσκευών και των παραμέτρων επικοινωνίας για αυτές επιλέγοντας “Configure” → “Flush Devices Setup” και να θέσει εκ νέου δυο νέες παραμετροποιήσεις για τις δύο συσκευές με τις οποίες επικοινωνεί η εφαρμογή.



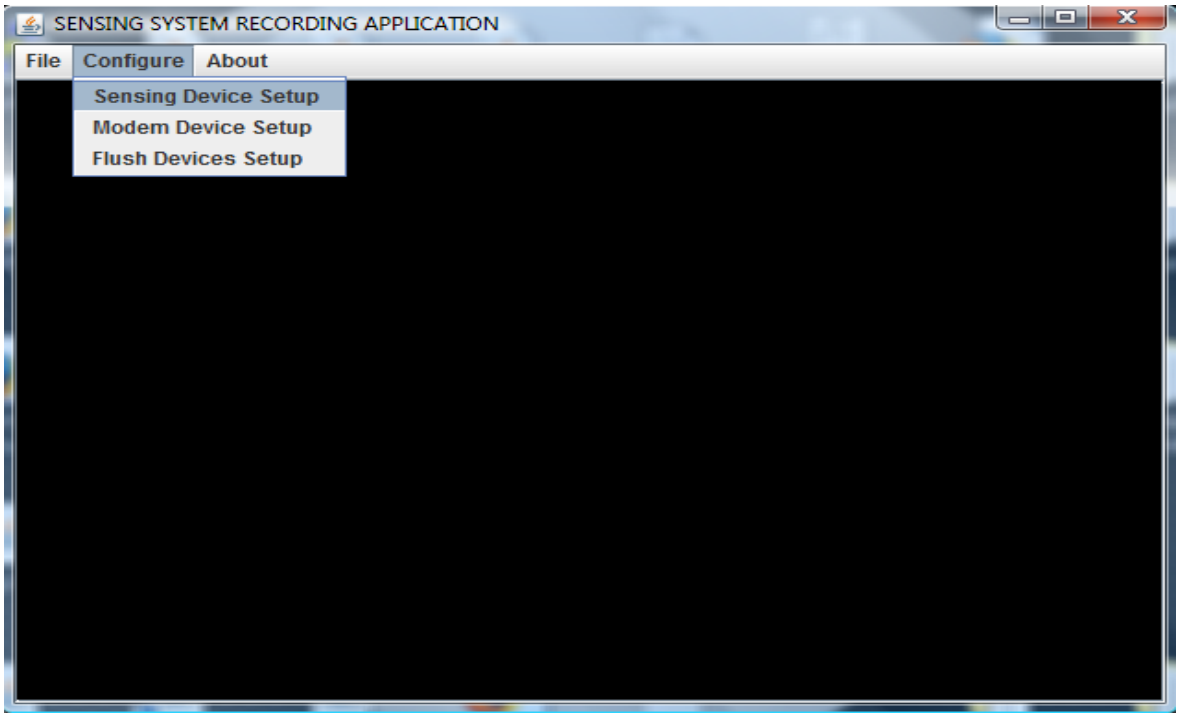
Σχήμα 11.8.a – Εκκαθάριση των αποθηκευμένων παραμετροποιήσεων

Μόλις ο χρήστης επιλέξει “Flush Devices Setup” το πρόγραμμα συνδέεται στην βάση δεδομένων διαγράφει τις αποθηκευμένες παραμετροποιήσεις και για τις δυο συσκευές και αποσυνδέεται από την βάση δεδομένων.



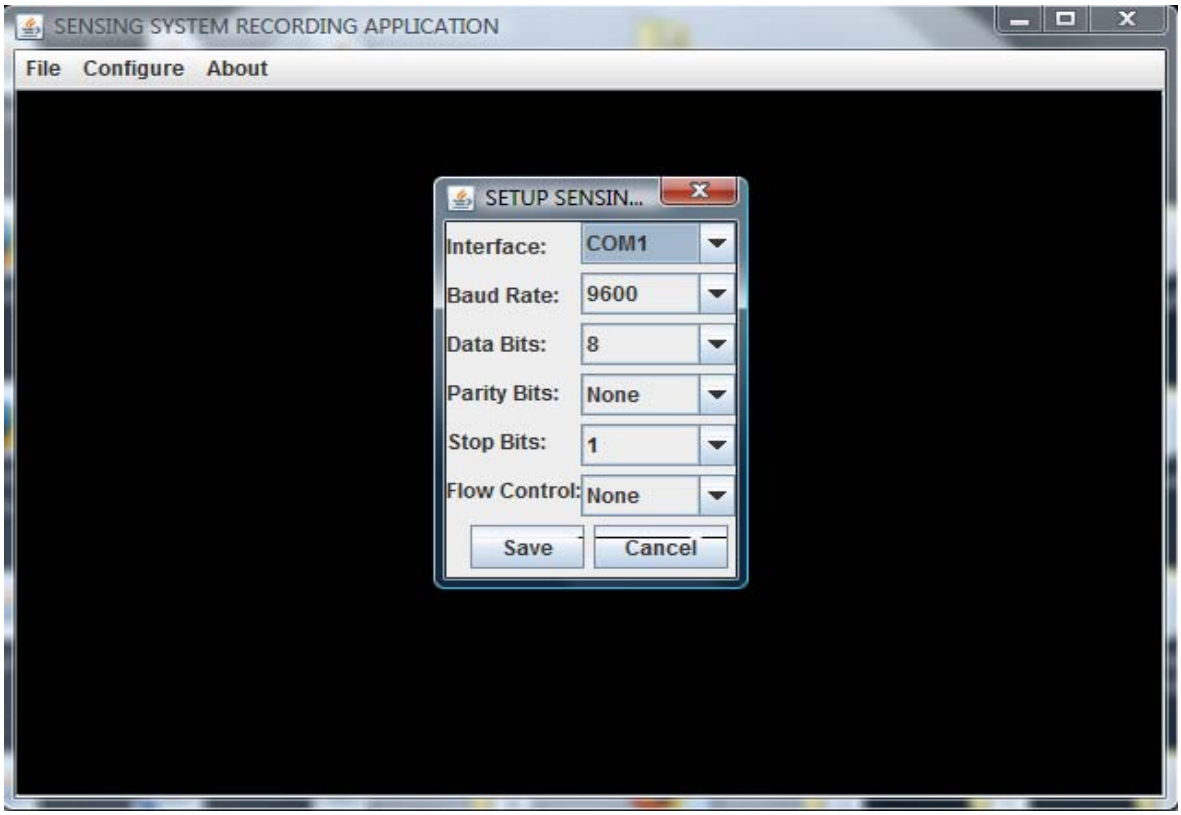
Σχήμα 11.8.b – Εκκαθάριση των αποθηκευμένων παραμετροποιήσεων

Έπειτα για να θέσει τις νέες παραμέτρους επικοινωνίας για την συσκευή των αισθητήρων, αλλά και για την συσκευή του κινητού τηλεφώνου που χρησιμοποιείται ως GSM Μόντεμ, μπορεί να επιλέξει “Configure” → “Sensing Device Setup” για τη συσκευή των αισθητήρων και αντιστοίχως “Configure” → “Modem Device Setup” για την συσκευή του κινητού τηλεφώνου.



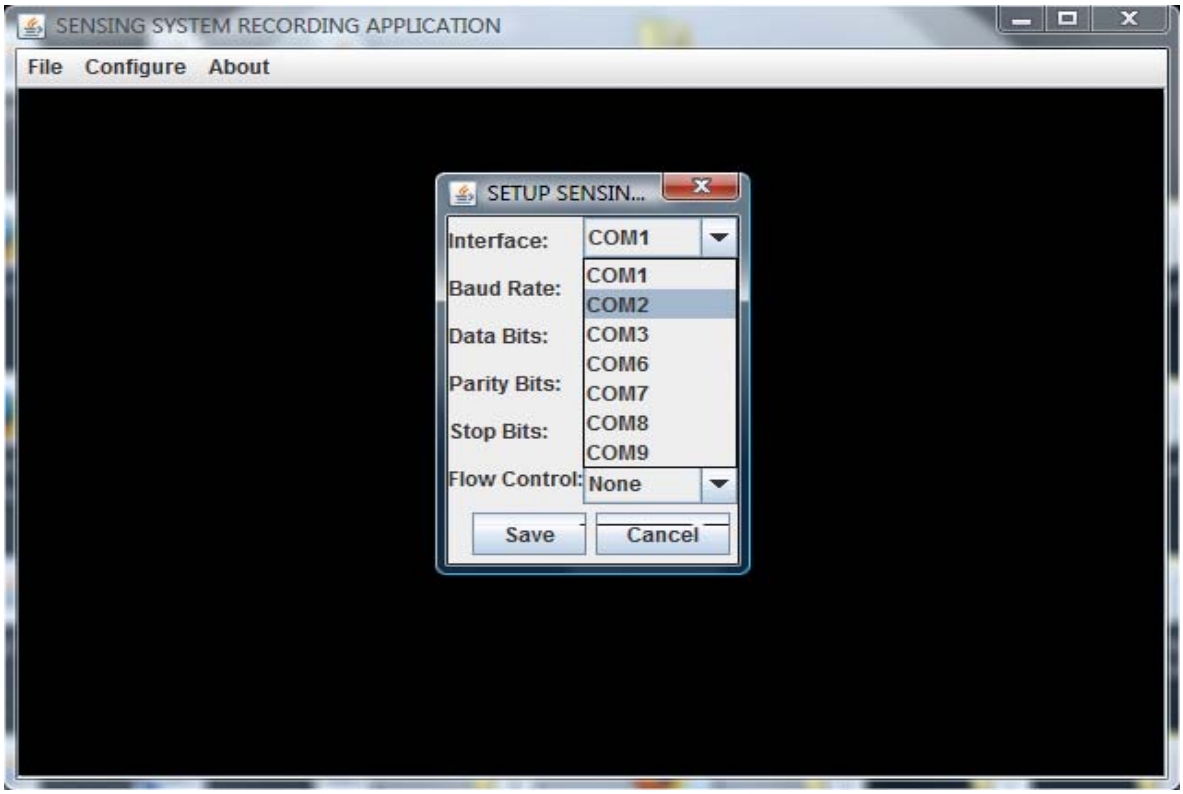
Σχήμα 11.9.α – Παραμετροποίηση σειριακής επικοινωνίας με την εξωτερική συσκευή

Ο διαχειριστής μπορεί να θέσει την σειριακή πόρτα στην οποία θα επικοινωνεί το πρόγραμμα με την εξωτερική συσκευή για να παραλαμβάνει τα δεδομένα των μετρήσεων από όλους τους αισθητήρες, να θέσει τον ρυθμό μετάδοσης δεδομένων, τον αριθμό των stop bits, τον τύπο του ελέγχου ισοτιμίας και τον τύπο του ελέγχου των δεδομένων της μετάδοσης.



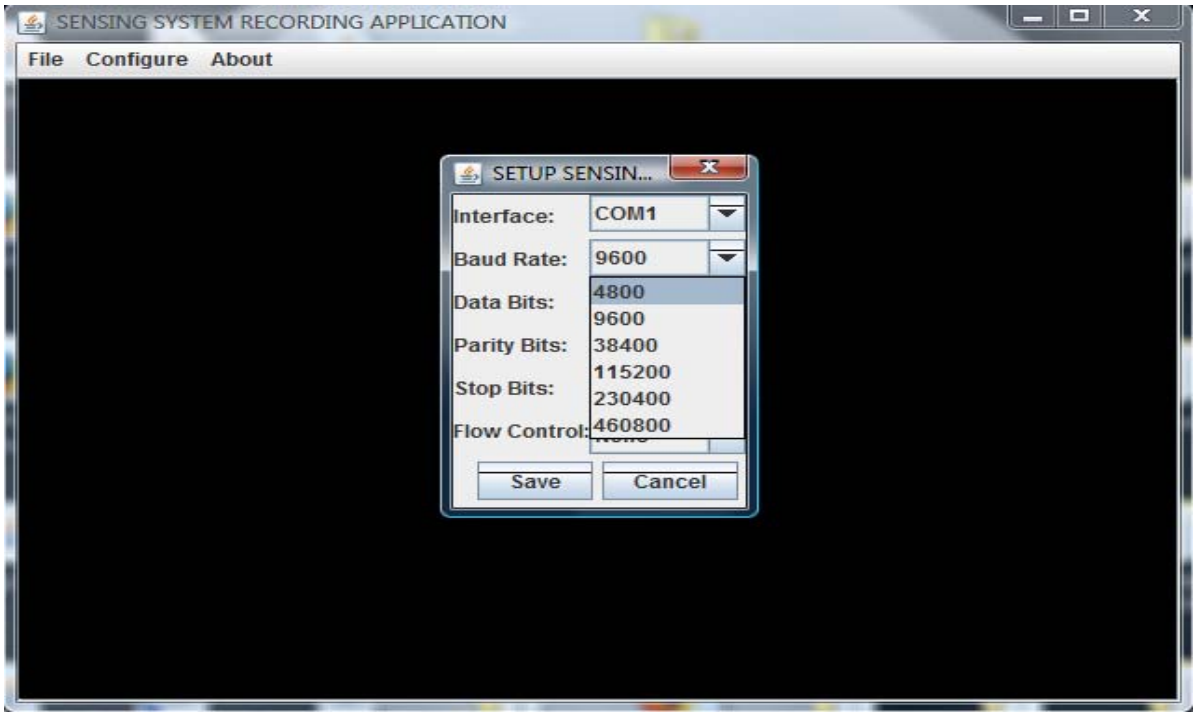
Σχήμα 11.9.b – Παραμετροποίηση σειριακής επικοινωνίας με την εξωτερική συσκευή

Η λίστα επιλογής των διαθέσιμων σειριακών θυρών του υπολογιστή, δημιουργείται δυναμικά. Κατά την φόρτωση του παραθύρου αυτού, σαρώνεται ο υπολογιστής που τρέχει η εφαρμογή και συμπληρώνεται η λίστα με όλες τις διαθέσιμες σειριακές και παράλληλες, φυσικές και εικονικές πόρτες του υπολογιστή. Στον υπολογιστή που γράφτηκε η εφαρμογή υπάρχουν από ότι φαίνεται και παρακάτω, 7 διαθέσιμες εικονικές και φυσικές σειριακές πόρτες: η COM1, η COM2, η COM3, η COM6, η COM7, η COM8 και η COM9.



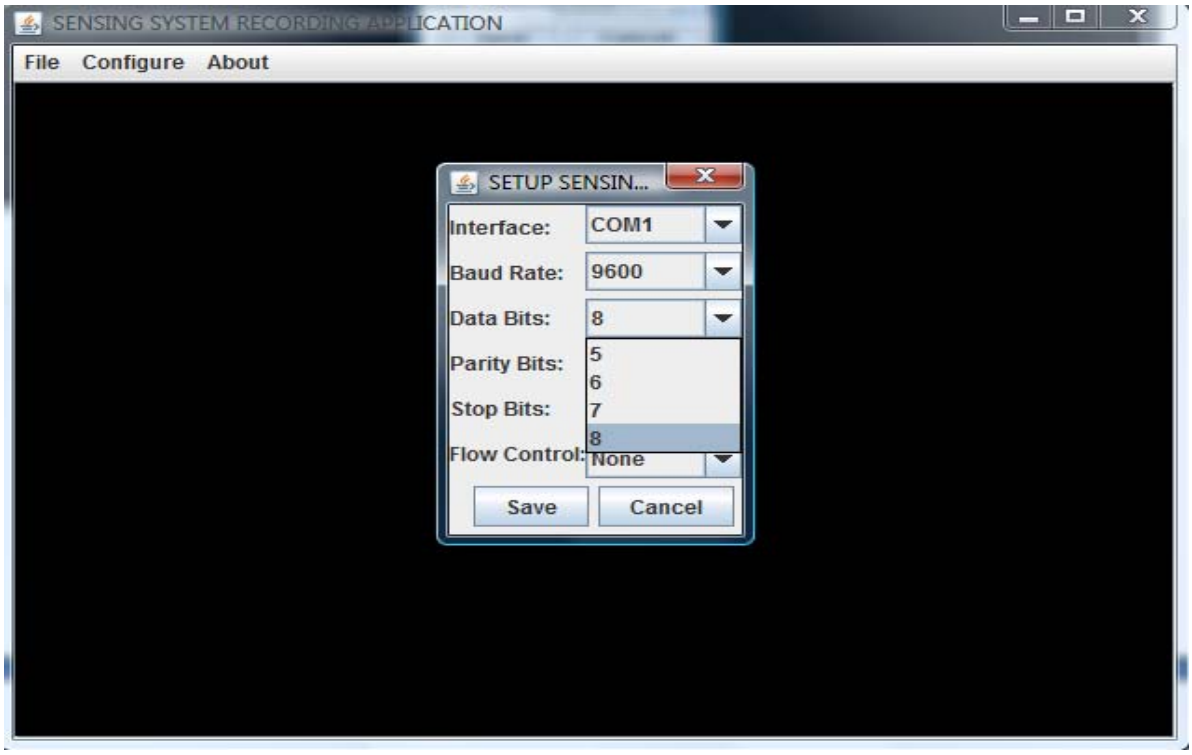
Σχήμα 11.9.c – Επιλογή σειριακής θύρας για την εξωτερική συσκευή

Ο διαχειριστής επιλέγει εκείνη την σειριακή πόρτα στην οποία έχει συνδεδεμένη την εξωτερική συσκευή με τους αισθητήρες και έπειτα μπορεί να θέσει και τις λοιπές παραμέτρους για την σειριακή επικοινωνία. Οι διαθέσιμες επιλογές που δίνονται από την εφαρμογή μας, εξόρισμού, είναι οι :4800 bytes, 9600 bytes, 38400 bytes, 115200 bytes, 230400 bytes και 460800 bytes, το δευτερόλεπτο.



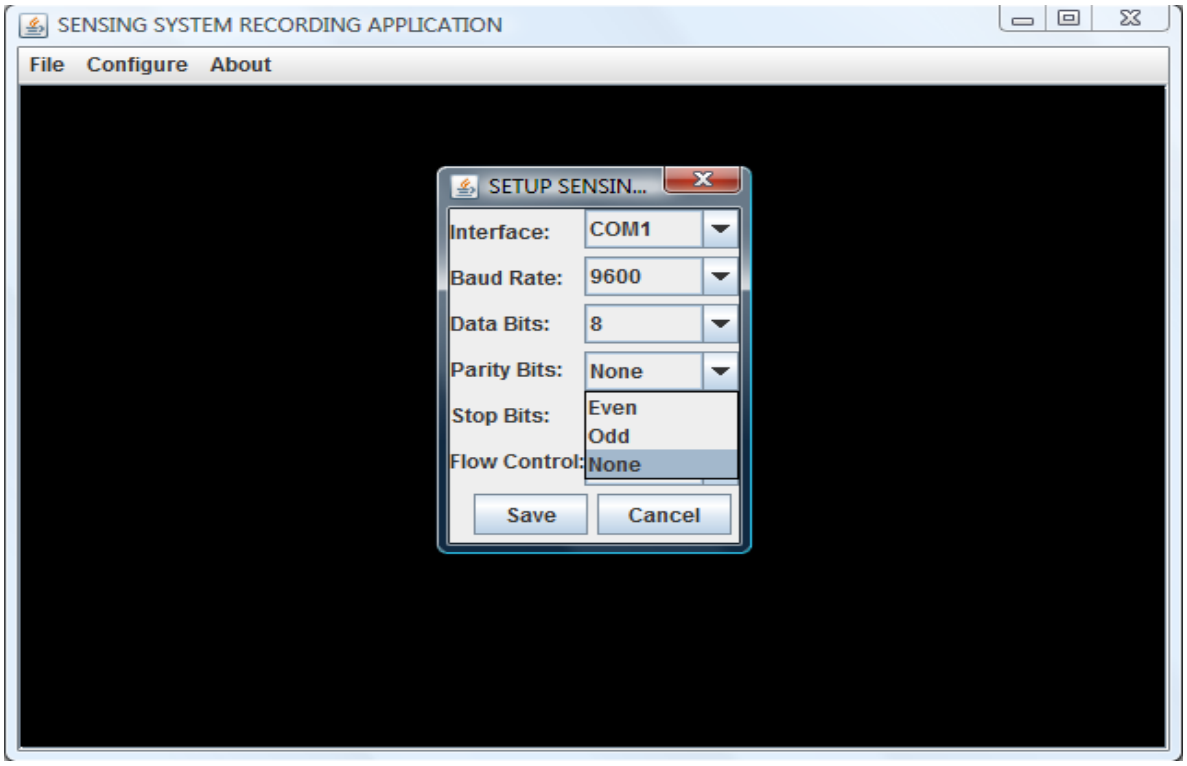
Σχήμα 11.9.d – Επιλογή ρυθμού μετάδοσης για την επικοινωνία

Ο διαχειριστής μπορεί να διαλέξει εάν θα χρησιμοποιηθούν είτε 5, είτε 6, είτε 7, είτε 8 ψηφία για την μετάδοση των των δεδομένων (data bits).



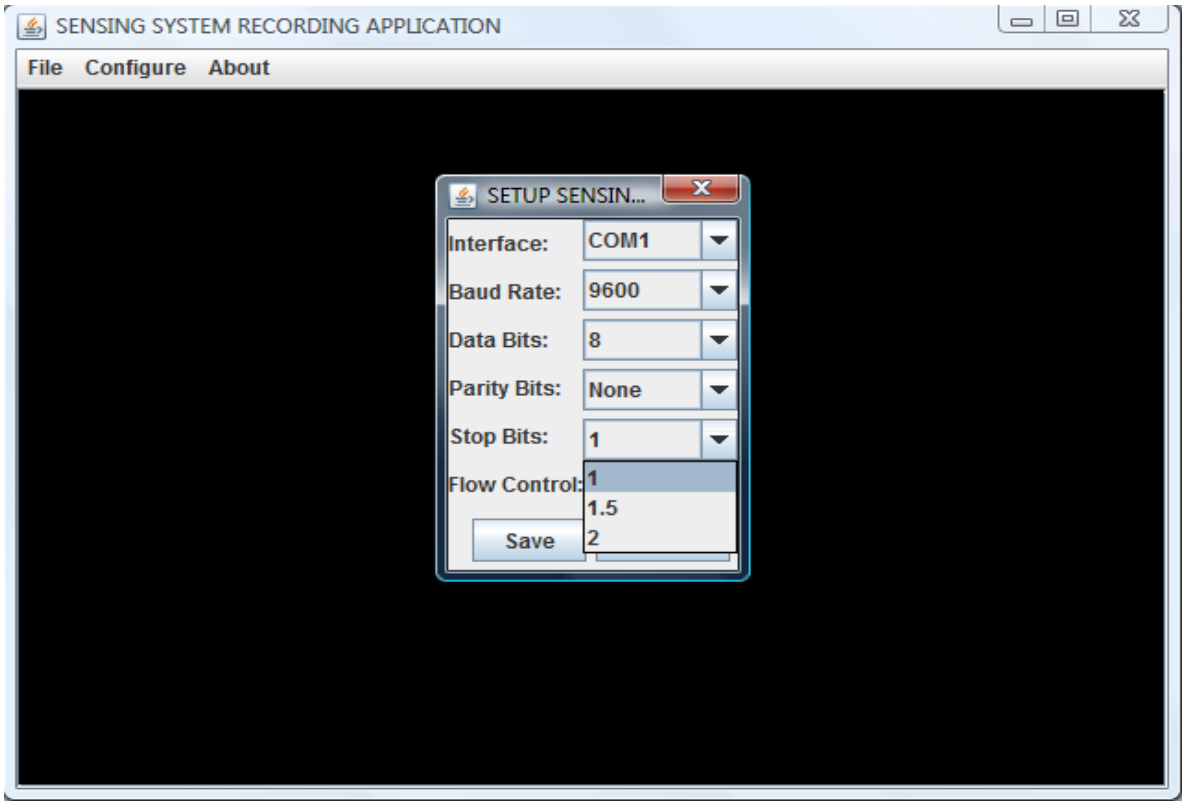
Σχήμα 11.9.e – Επιλογή αριθμού των bits μετάδοσης

‘Επειτα μπορεί να ορίσει τον τύπο του ελέγχου ισοτιμίας που θα χρησιμοποιηθεί στην μετάδοση των δεδομένων: οι διαθέσιμες επιλογές που δίνονται από την εφαρμογή μας, εξορισμού, είναι οι: Η «Άρτια Ισοτιμία», Η «Περιττή Ισοτιμία» και η επιλογή «ΚΑΝΕΝΑΣ».



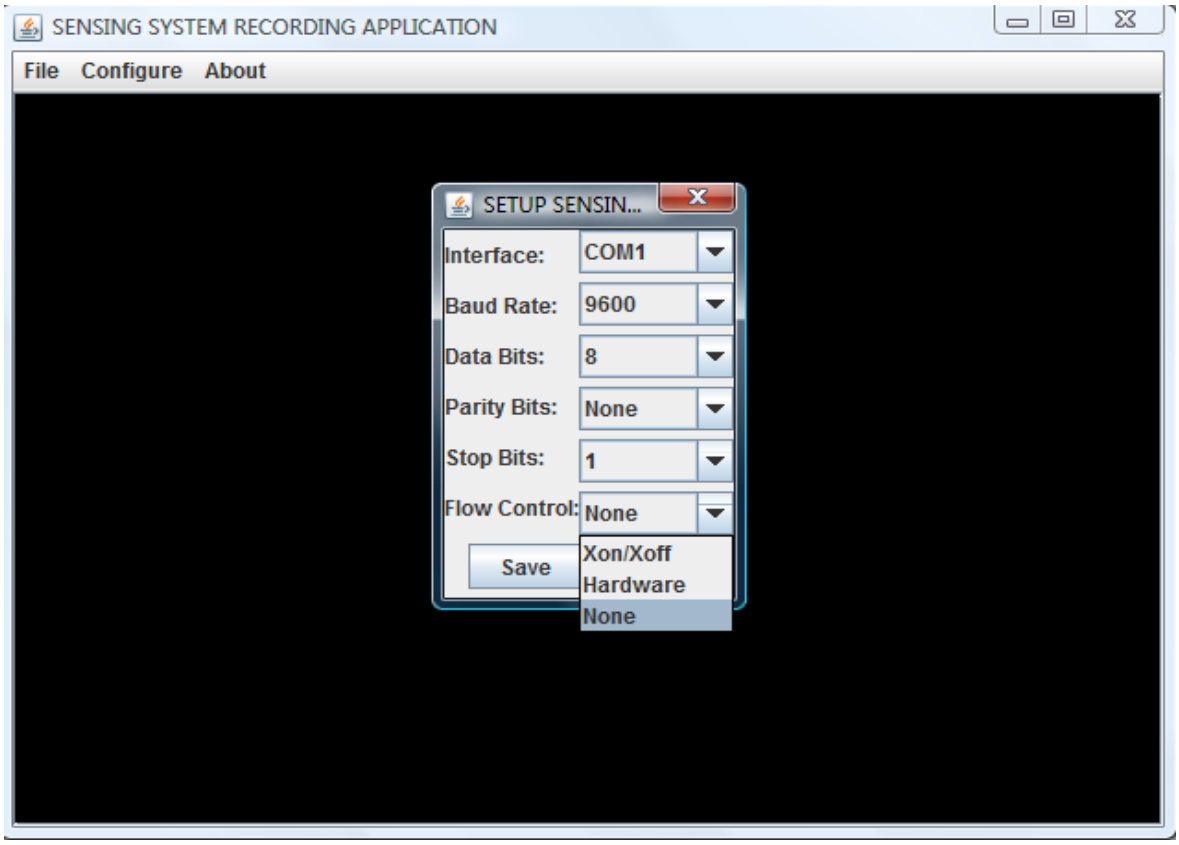
Σχήμα 11.9.f – Επιλογή τύπου ελέγχου της ισοτιμίας

Ο διαχειριστής μπορεί να θέσει και τον αριθμό των ψηφίων που θα χρησιμοποιηθούν για την σήμανση της διακοπής στη μετάδοση των δεδομένων (stop bit). Οι διαθέσιμες επιλογές που δίνονται από την εφαρμογή μας εξορισμού, είναι είτε 1 ψηφίο, είτε 1. 5 ψηφίο, είτε 2 ψηφία.



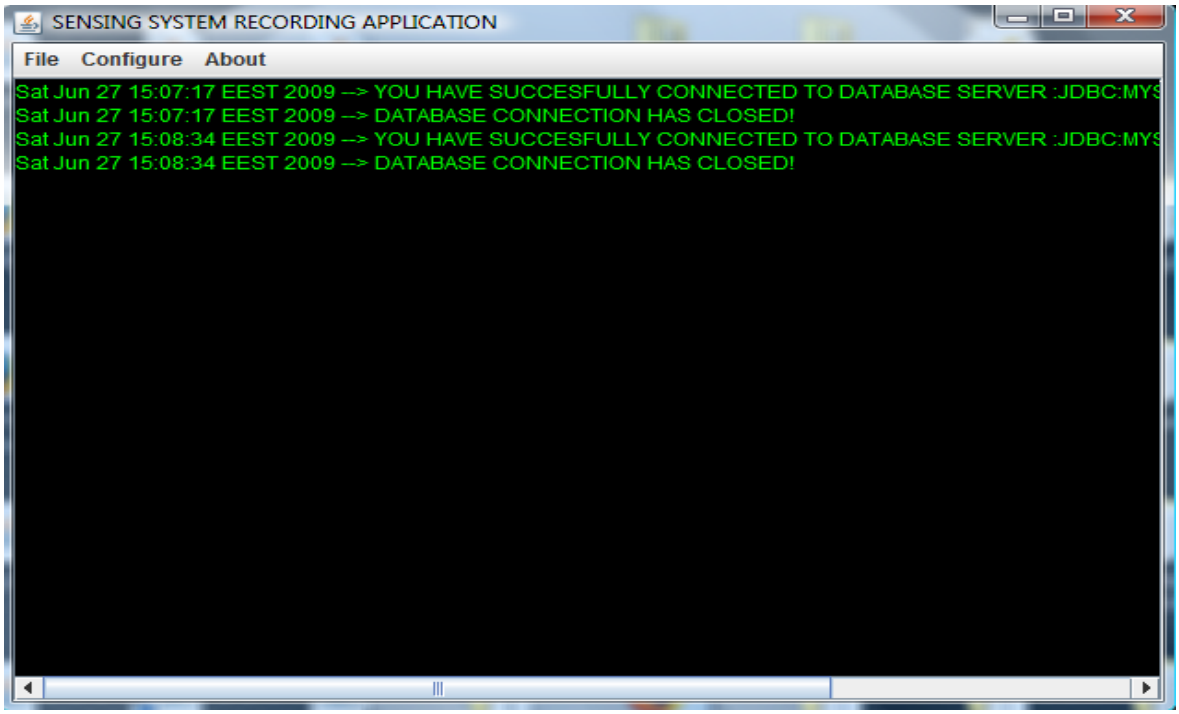
Σχήμα 11.9.g – Επιλογή του αριθμού των stop bits

Τέλος, η τελευταία παράμετρος που μπορεί να θέσει ο διαχειριστής για την επικοινωνία ανάμεσα στον υπολογιστή και την εξωτερική συσκευή με τους αισθητήρες, είναι ο τύπος του ελέγχου των δεδομένων της μετάδοσης. Οι διαθέσιμες επιλογές που δίνονται εξορισμού από την εφαρμογή μας, είναι είτε «on/off», είτε «hardware», είτε «κανένας».



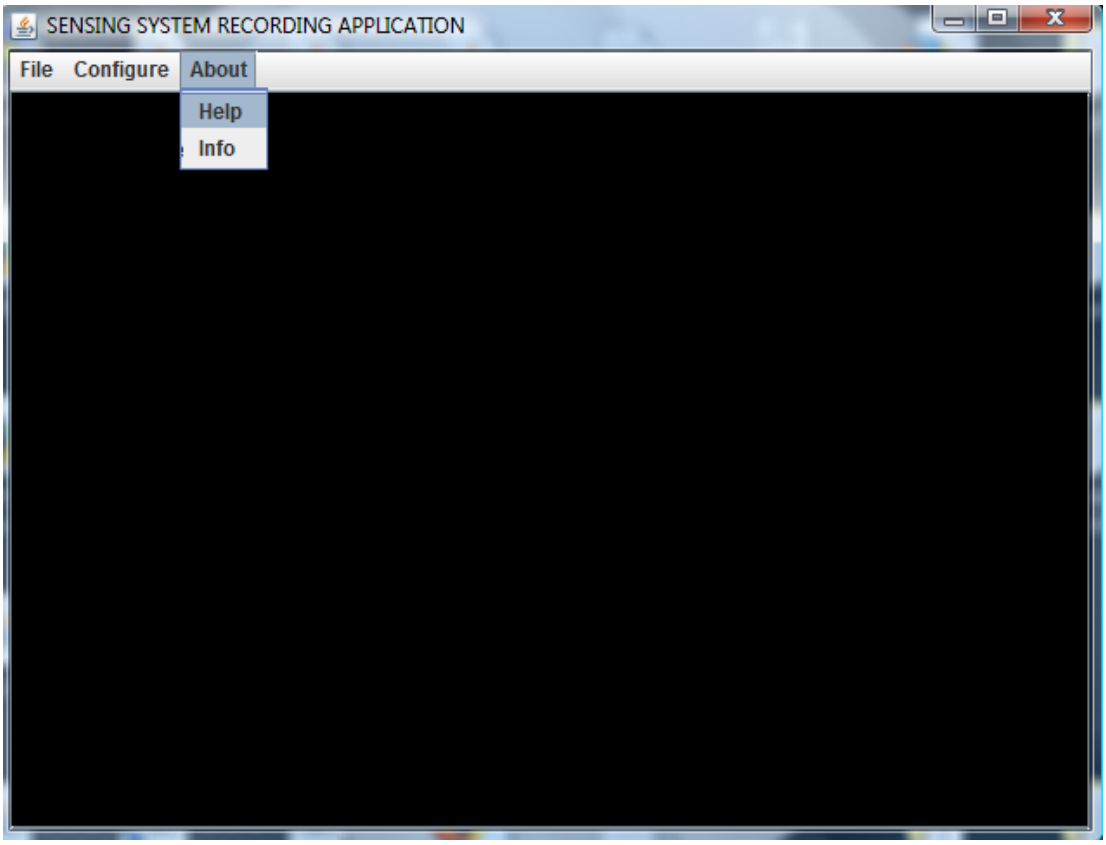
Σχήμα 11.9.h – Επιλογή τύπου ελέγχου της ροής δεδομένων

Για κάθε συσκευή, το πρόγραμμα συνδέεται στη βάση δεδομένων και αποθηκεύει την νέα διαμόρφωση των παραμέτρων επικοινωνίας που θα χρησιμοποιηθούν για την σύνδεση με κάθε μία από αυτές.

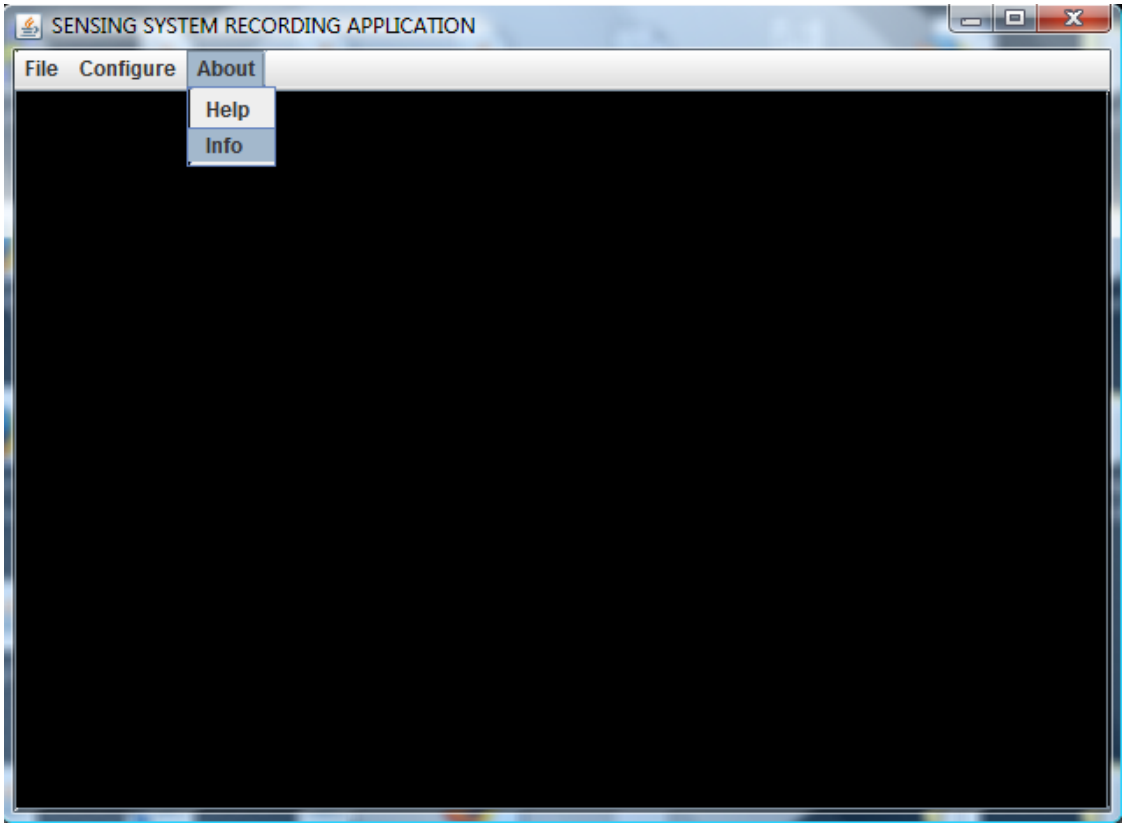


Σχήμα 11.10 – Αποθήκευση της παραμετροποίησης

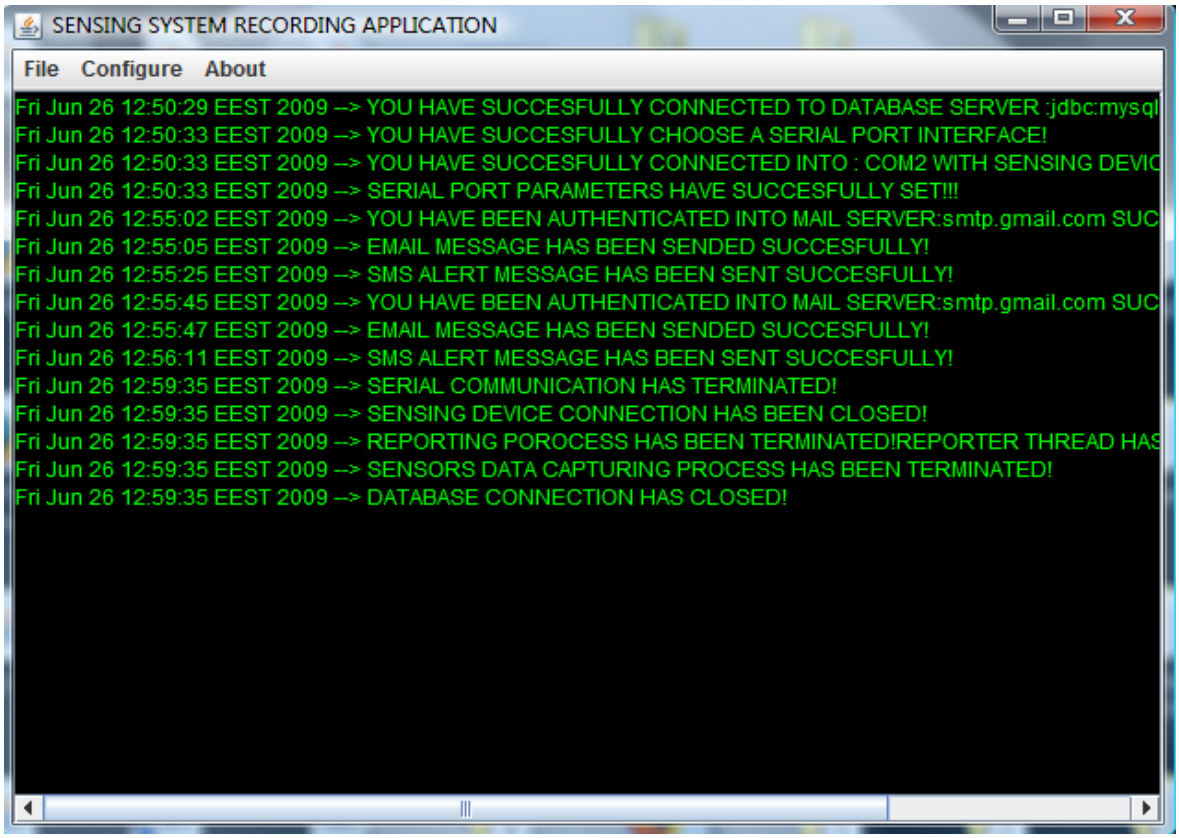
Από το menu “About” ο χρήστης μπορει να επιλέξει “Help” για να διαβάσει ένα σύντομο κείμενο βοήθειας και επεξήγησης της εφαρμογής μας, ενώ από την επιλογή “About” μπορεί να πληροφορηθεί για τους δημιουργούς της εφαρμογής (τα ονοματεπώνυμα μας) και για τους λογαριασμούς ηλεκτρονικού ταχυδρομείου που μπορεί να επικοινωνήσει για την υποστήριξη του.



Σχήμα 11.11 – Προβολή βοήθειας για τη χρήση της εφαρμογής

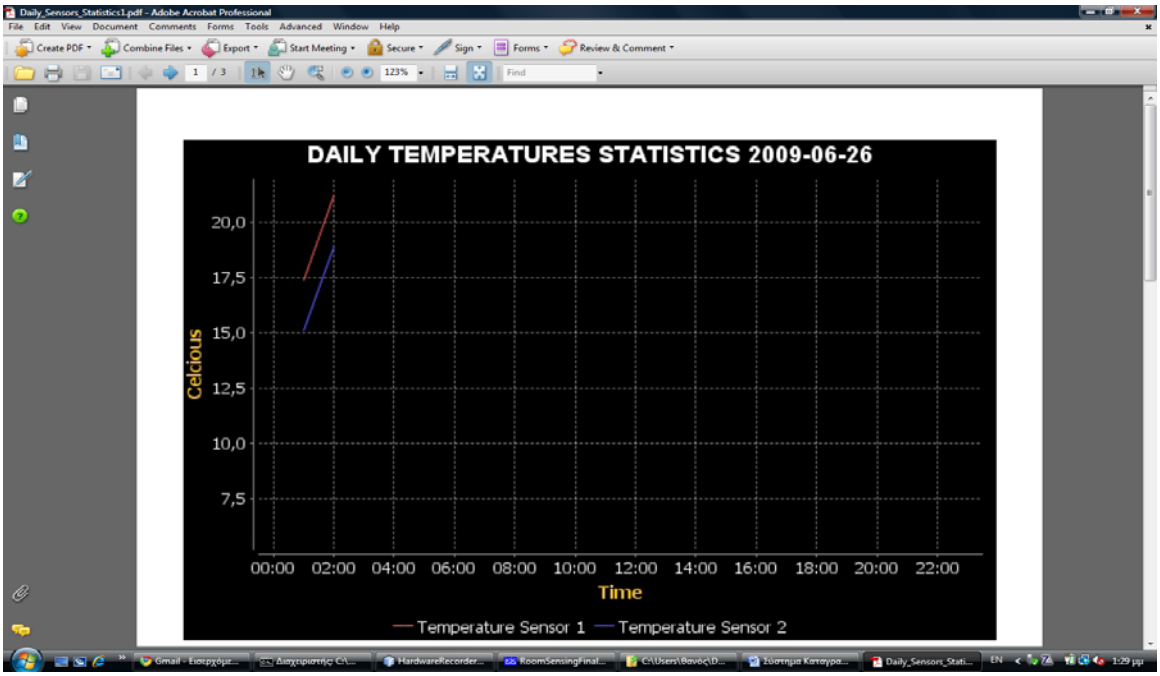


Σχήμα 11.12 – Προβολή πληροφοριών για τους δημιουργούς της εφαρμογής



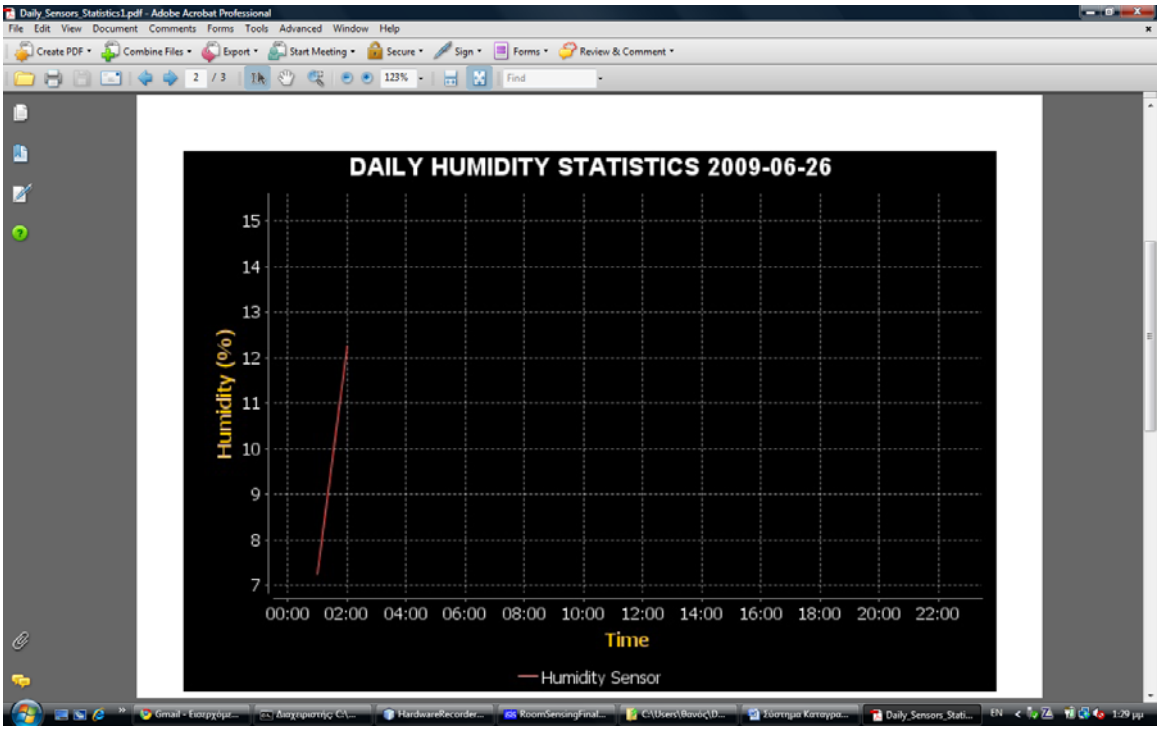
Σχήμα 11.13 – Αποσύνδεση από την εξωτερική συσκευή

Κατά την εκκίνηση του κύριου νήματος της εφαρμογής, εκκινείται ένα ακόμα νήμα, το οποίο όμως είναι προγραμματισμένο να πραγματοποιήσει την δική του εργασία, μετά από τόσα χιλιοστά του δευτερολέπτου, όσα εκείνα που προκύπτουν κάθε φορα από την αφαίρεση της χρονικής στιγμής που τρέχει η εφαρμογή από την τιμή που αντιστοιχεί στα μεσάνυχτα, δηλαδή την αλλαγή της ημέρας και έπειτα, να την επανα-πραγματοποιεί ανά 24 ώρες. Το νήμα αυτό είναι υπεύθυνο να δημιουργήσει δυναμικά τα αρχεία με τα στατιστικά διαγράμματα για κάθε αισθητήρα, να τα ενσωματώσει σε αρχεία pdf, ημέρας, εβδομάδας, μήνα και έτους και έπειτα να τα αποστείλει ως συννημένα αρχεία σε μήνυμα ηλεκτρονικής αλληλογρραφίας στον λογαριασμό ηλεκτρονικού ταχυδρομείου του χρηστή της εφαρμογής. Συνολικά δημιουργούνται τέσσερα αρχεία pdf (portable document format): το πρώτο αφορά στους



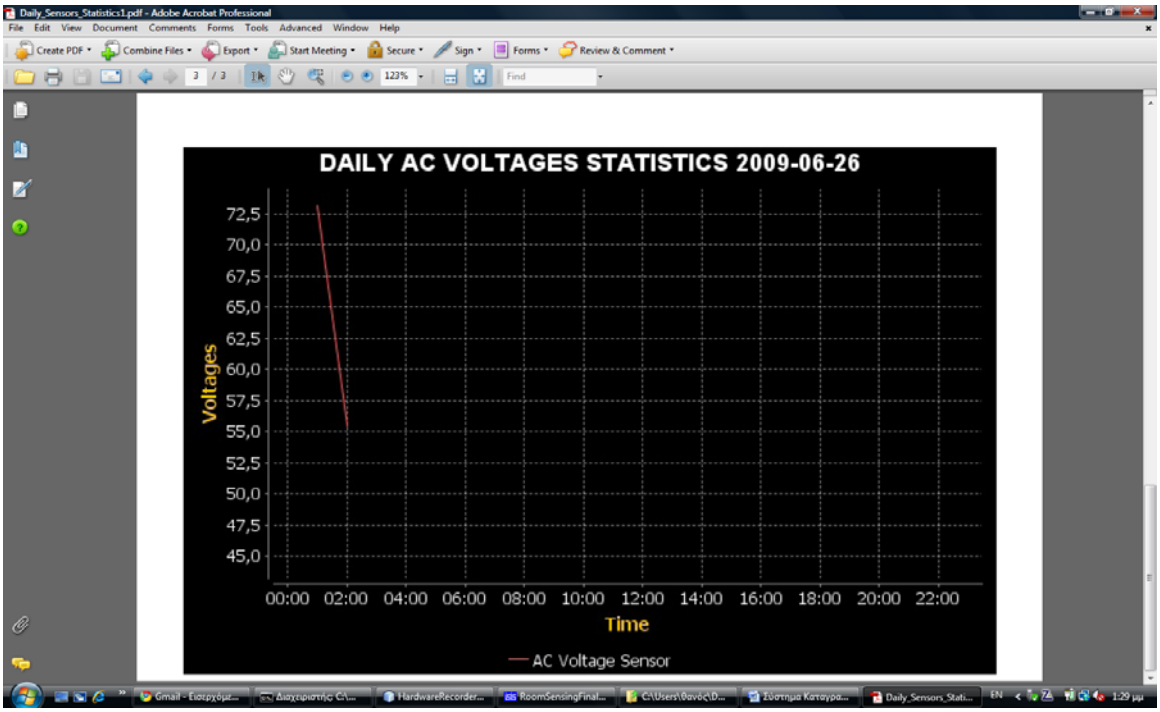
Σχήμα 11.14 – Στατιστικά ημέρας για τους αισθητήρες θερμοκρασίας

στις μετρήσεις των δυο αισθητήρων θερμοκρασίας, του αισθητήρα υγρασίας και του αισθητήρα τάσης δικτύου για την τρέχουσα ημέρα. Τα διαγράμματα αναπαριστούν τις μετρήσεις των αισθητήρων



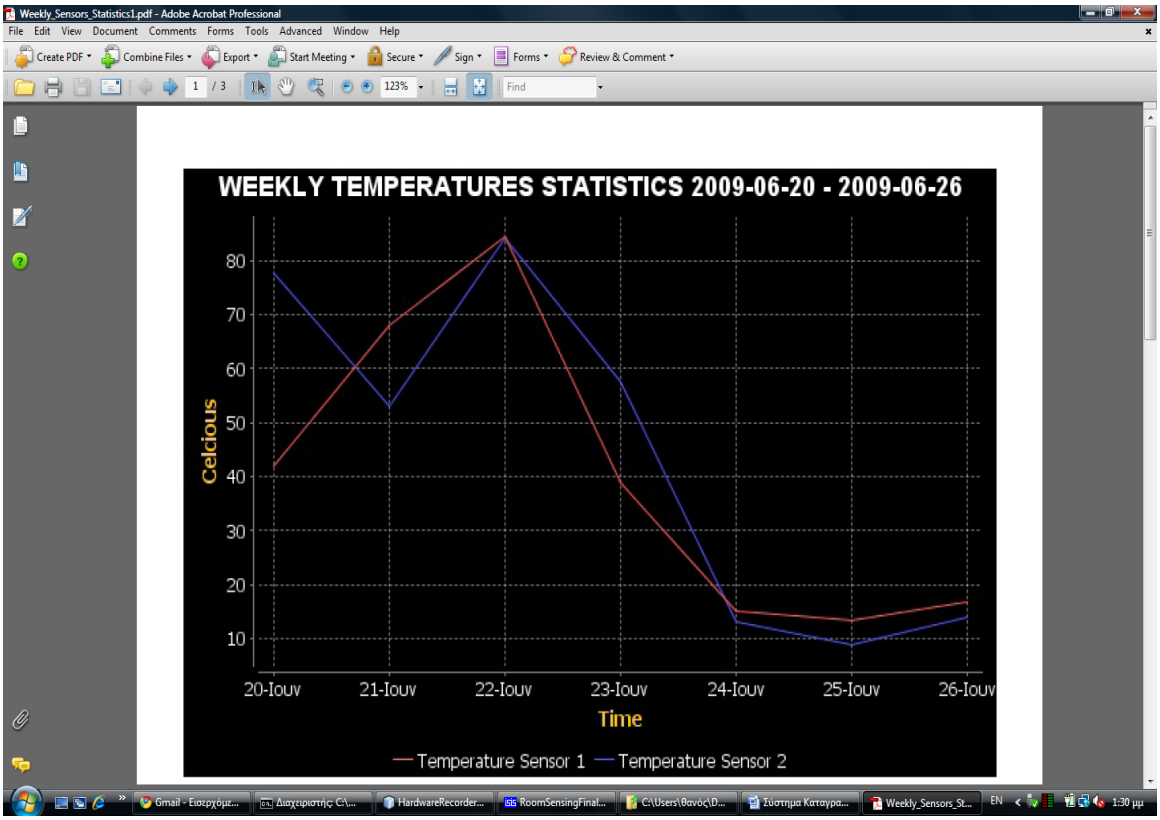
Σχήμα 11.15 – Στατιστικά ημέρας για τον αισθητήρα υγρασίας

για κάθε ώρα της τρέχουσας ημέρας ακόμα και εάν δεν υπάρχουν καταγεγραμμένες μετρήσεις για κάποιες από τις ώρες αυτές.



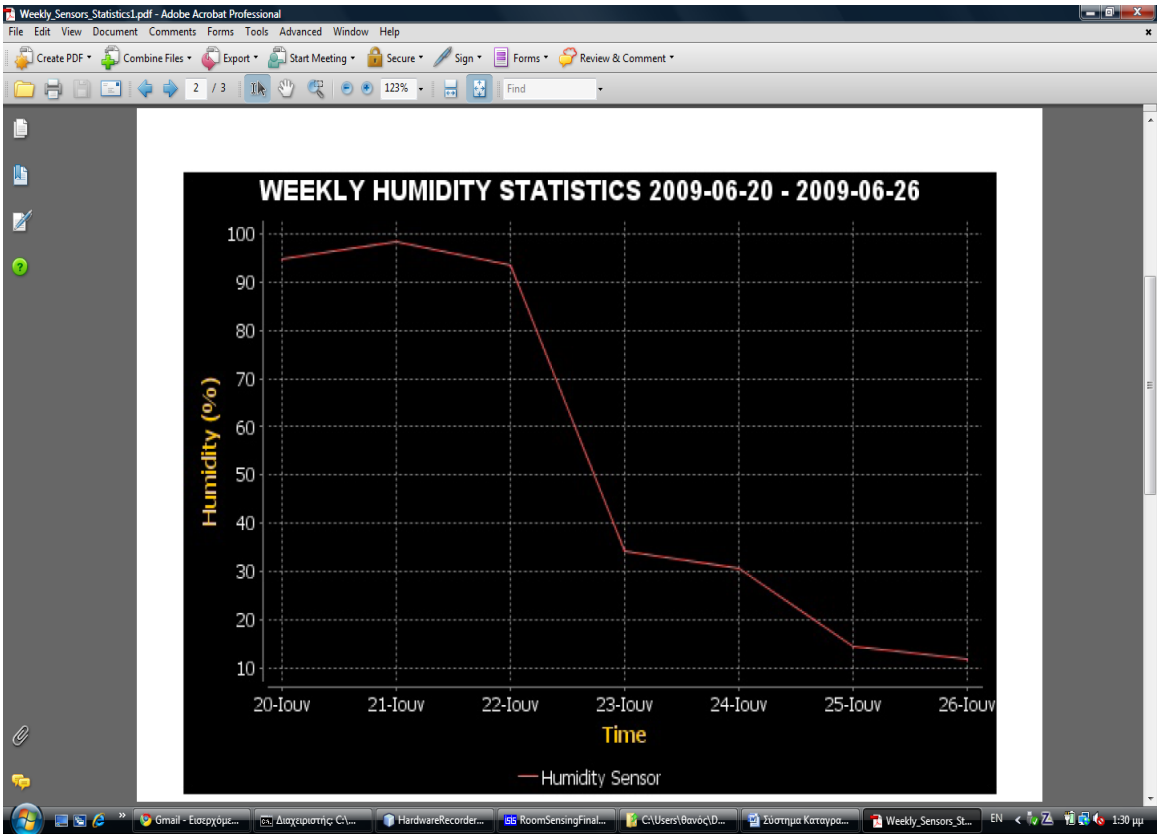
Σχήμα 11.16 – Στατιστικά ημέρας για τον αισθητήρα τάσης δικτύου

Το δεύτερο αρχείο pdf, αφορά στα διαγράμματα της τρέχουσας εβδομάδας για τους δυο αισθητήρες θερμοκρασίας, τον αισθητήρα υγρασίας και τον αισθητήρα τάσης δικτύου.

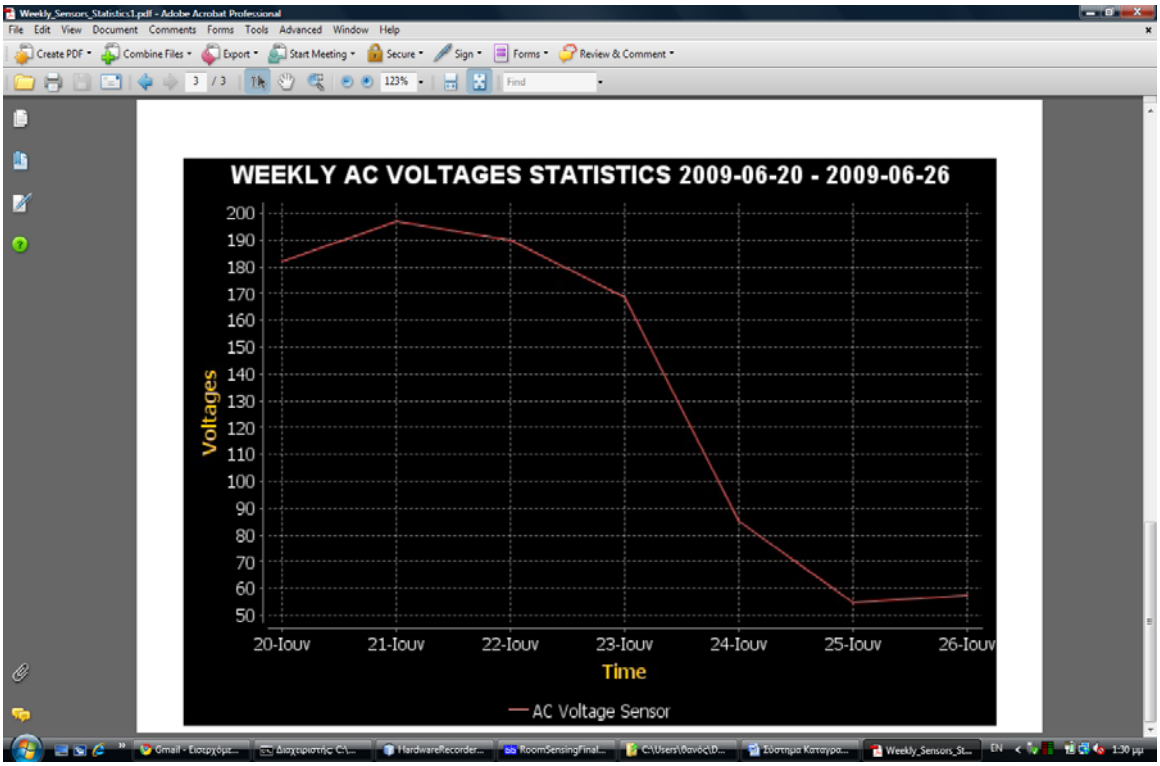


Σχήμα 11.17 – Στατιστικά εβδομάδας για τους αισθητήρες θερμοκρασίας

Σε κάθε σελίδα του εγγράφου, αναπαρίστανται όλες οι μετρήσεις για κάθε ημέρα της εβδομάδας, ακόμα και εάν δεν υπάρχουν καταγεγραμμένες μετρήσεις για κάποιες από τις ημέρες αυτές.

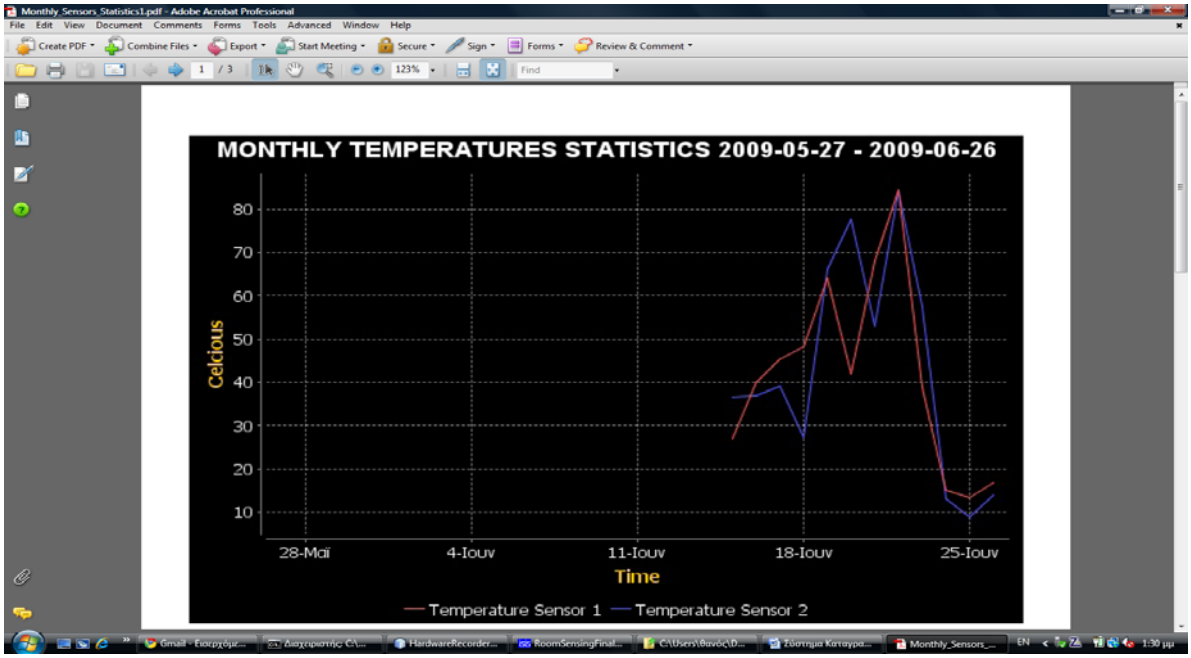


Σχήμα 11.18 – Στατιστικά εβδομάδας για τον αισθητήρα υγρασίας



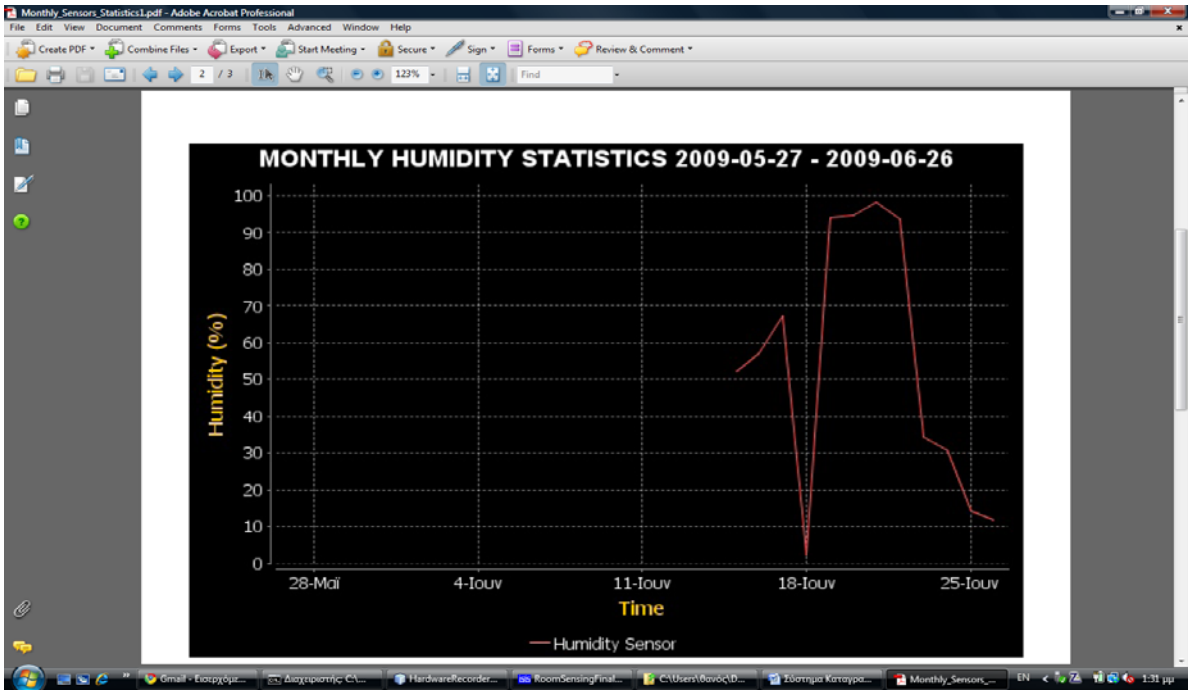
Σχήμα 11.19 – Στατιστικά εβδομάδας για τον αισθητήρα τάσης δικτύου

Το τρίτο αρχείο pdf, αφορά στα διαγράμματα του τρέχοντα μήνα για τους αισθητήρες θερμοκρασίας, τον αισθητήρα υγρασίας και τον αισθητήρα τάσης δικτύου.

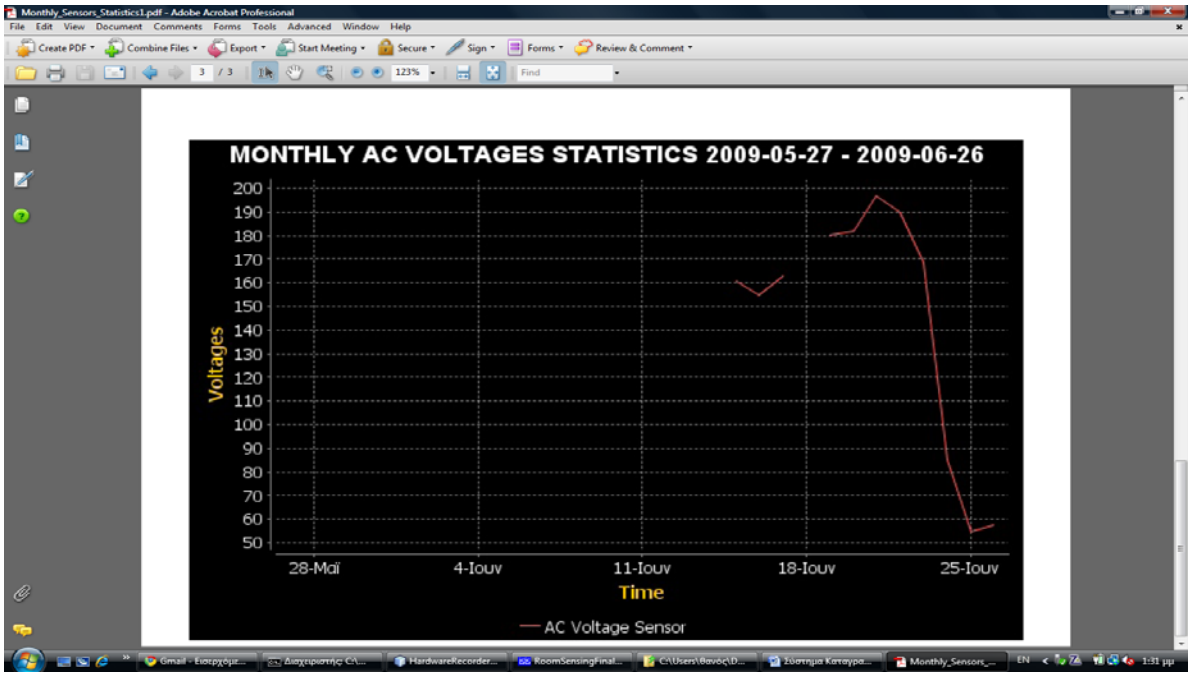


Σχήμα 11.20 – Στατιστικά μήνα για τους αισθητήρες θερμοκρασίας

Σε κάθε σελίδα του εγγράφου αυτού, αναπαρίστανται όλες οι μετρήσεις για κάθε ημέρα του μήνα αυτού ακόμα και αν δεν υπάρχουν καταγεγραμμένες μετρήσεις για κάποιες από τις ημέρες του μήνα αυτού.

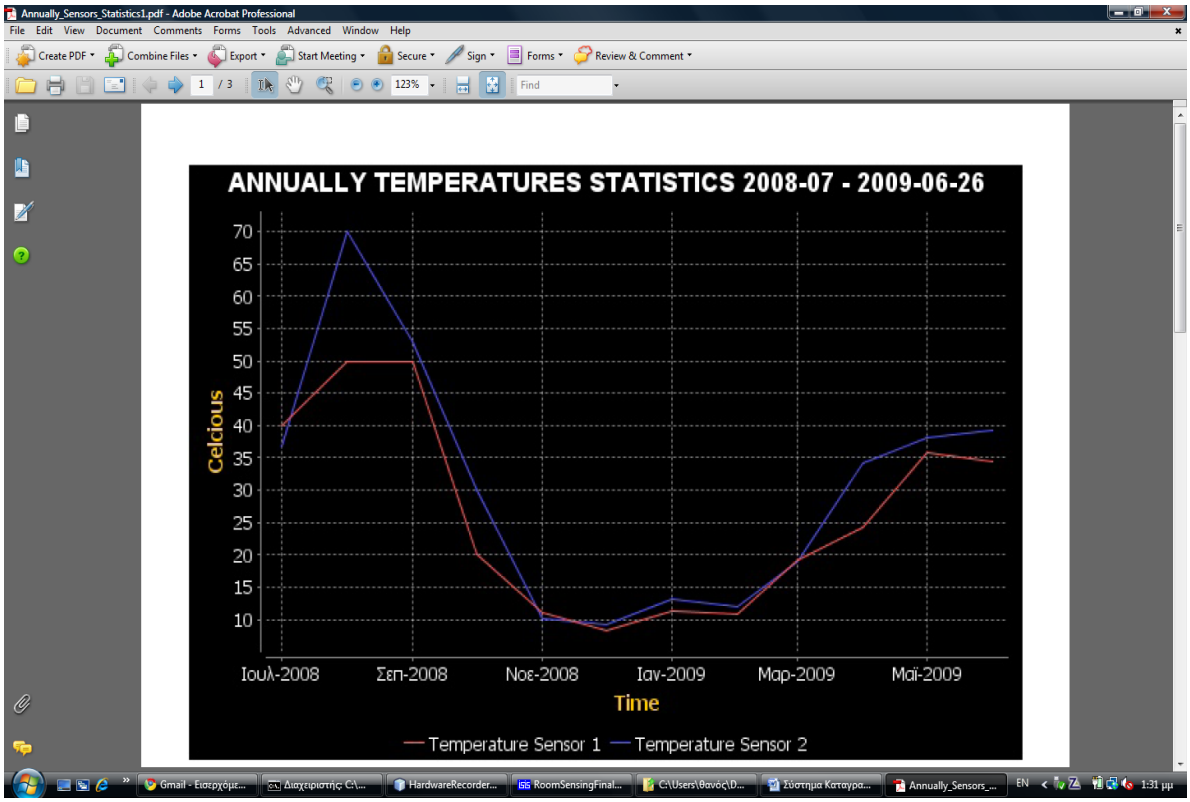


Σχήμα 11.21 – Στατιστικά μήνα για τον αισθητήρα υγρασίας



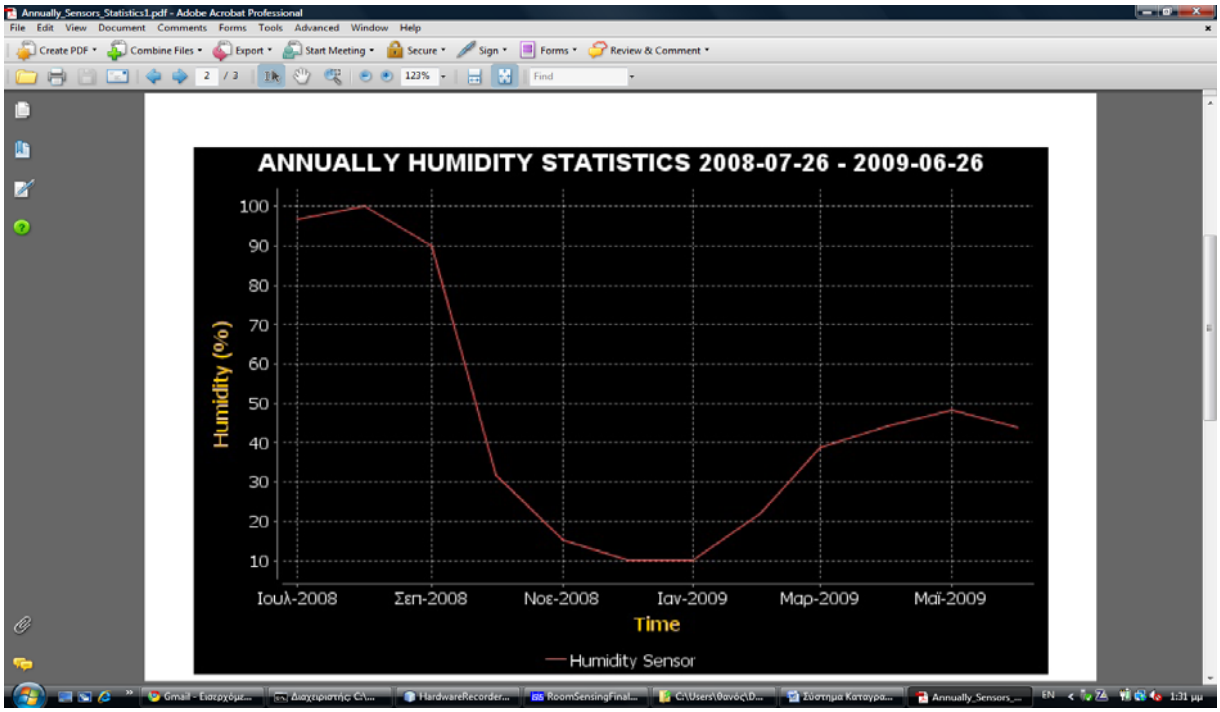
Σχήμα 11.21 – Στατιστικά μήνα για τον αισθητήρα τάσης δικτύου

Το τέταρτο και τελευταίο pdf που δημιουργείται, αφορα στα διαγράμματα του τρέχουν έτους για τους αισθητήρες θερμοκρασίας, τον αισθητήρα υγρασίας και τον αισθητήρα τάσης δικτύου.

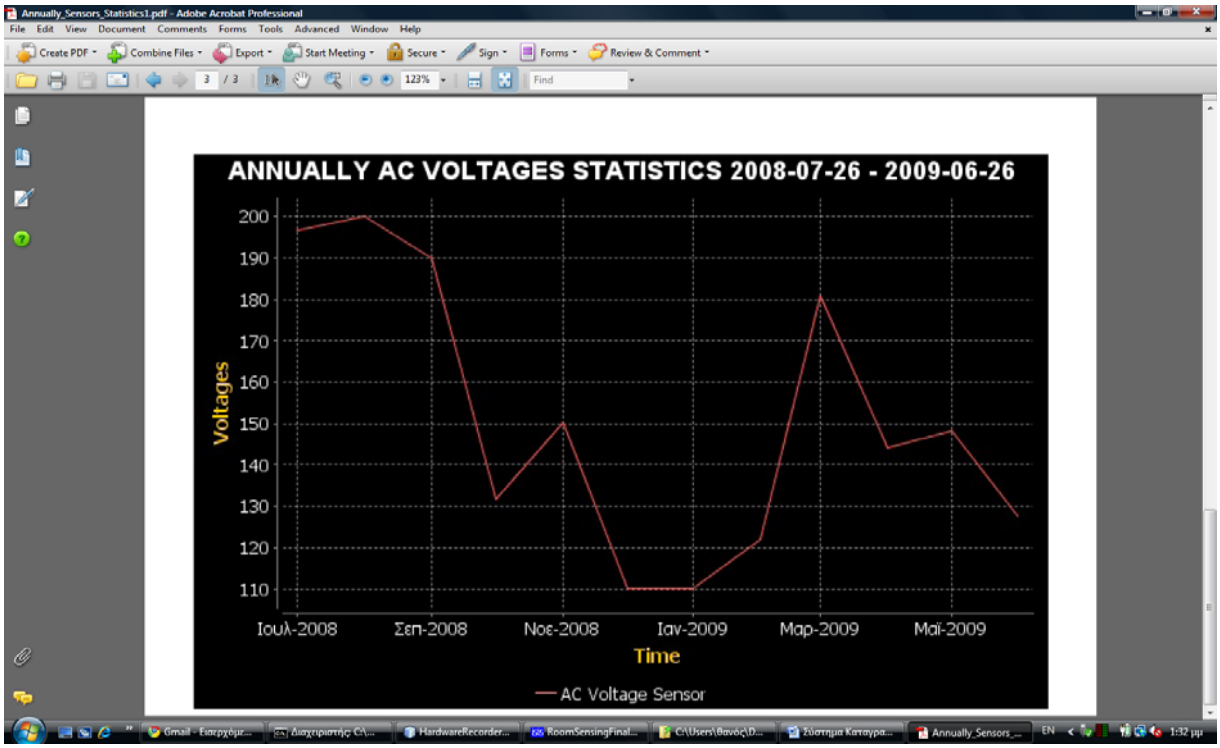


Σχήμα 11.22 – Στατιστικά έτους για τους αισθητήρες θερμοκρασίας

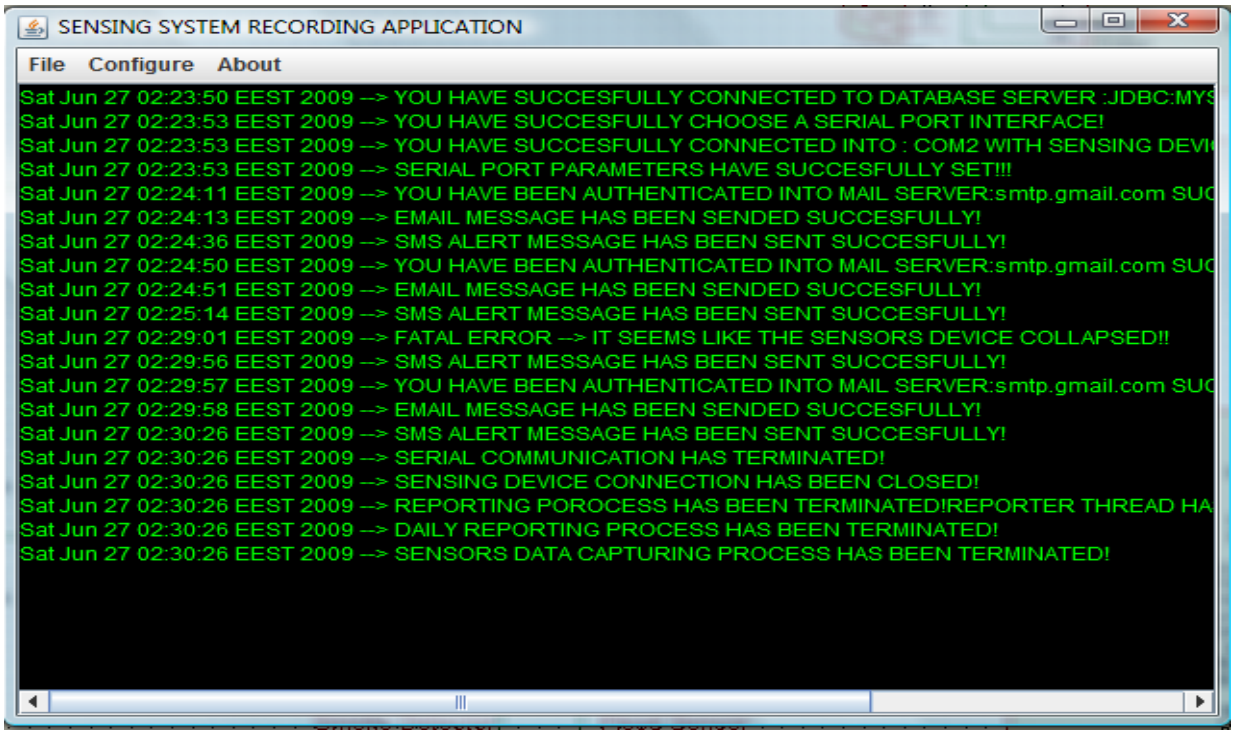
Σε κάθε σελίδα του εγγράφου αυτού, αναπαρίστανται οι μέσοι όροι των μετρήσεων για κάθε μήνα του τρέχοντος έτους ακόμα και αν δεν υπάρχουν καταγεγραμμένες μετρήσεις για κάποιους από τους μήνες αυτού.



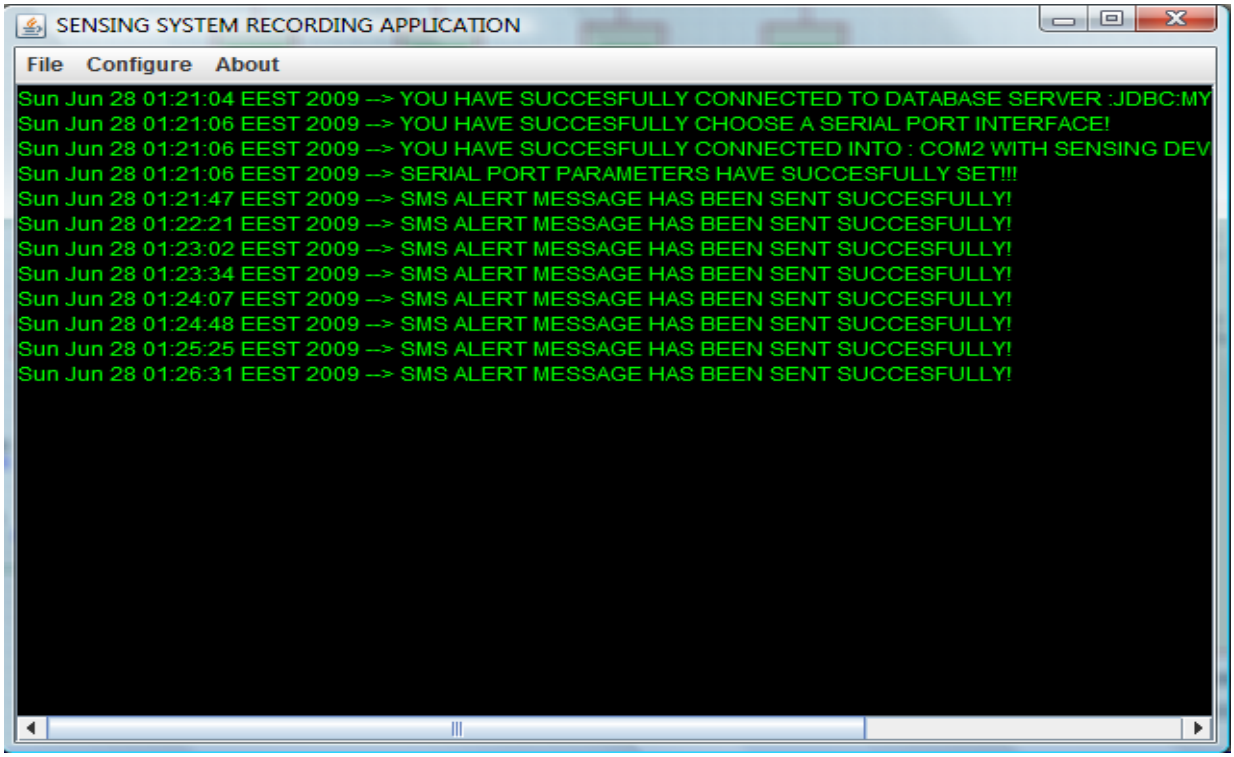
Σχήμα 11.23 – Στατιστικά έτους για τον αισθητήρα υγρασίας



Σχήμα 11.24 – Στατιστικά έτους για τον αισθητήρα τάσης δικτύου



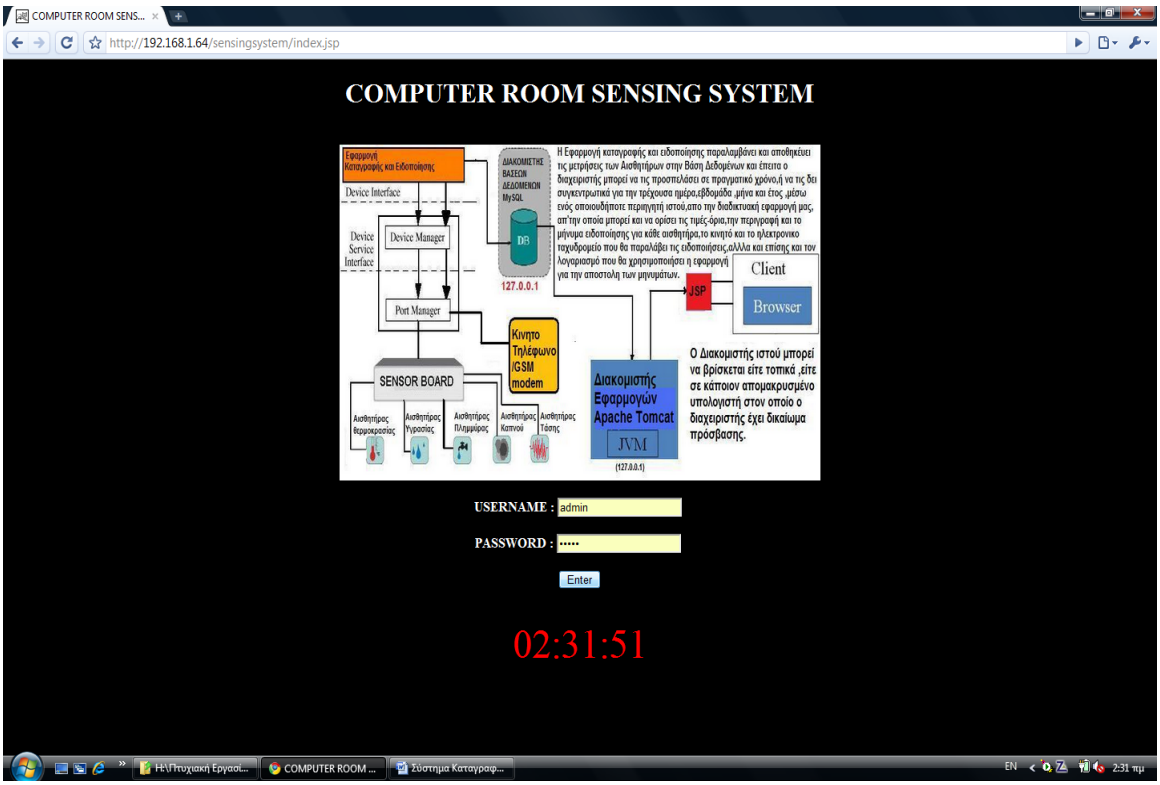
Σχήμα 11.25 – Καταγραφή λειτουργιών και αποστολή προειδοποίησης με sms και αναφοράς με email



Σχήμα 11.26 – Παρακολούθηση λειτουργιών για αποστολή sms

Παράρτημα Β
Εγχειρίδιο διάδρασης με την διαδικτυακή εφαρμογή παρουσίασης των καταγεγραμμένων μετρήσεων

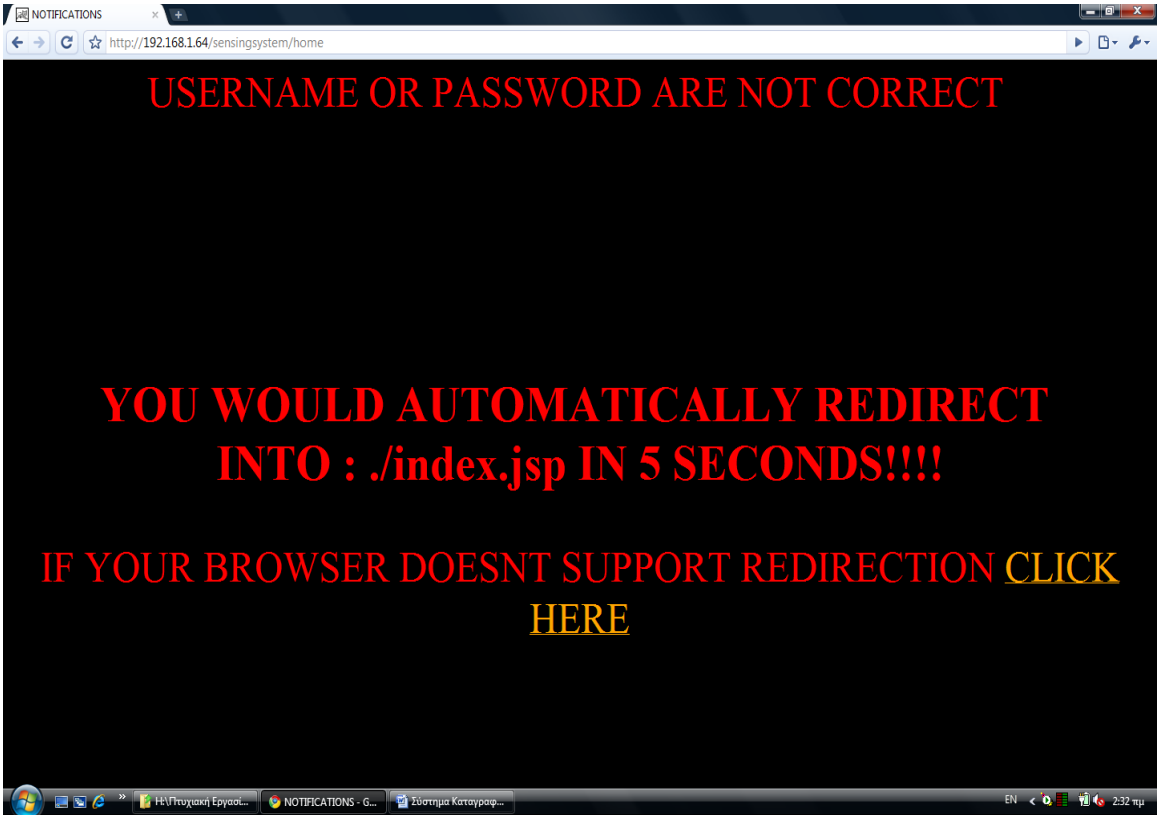
Η διαδικτυακή εφαρμογή μας, δίνει τη δυνατότητα παρουσίασης των μετρήσεων όλων των αισθητήρων σε πραγματικό χρόνο, αλλα και επίσης την δυνατότητα προβολής συγκεντρωτικών στατιστικών διαγραμμάτων των τιμών θερμοκρασίας, υγρασίας και τάσης δικτύου είτε για την τρέχουσα ημέρα, είτε για την τρέχουσα εβδομάδα, είτε για τον τρέχον μήνα, είτε για το τρέχον έτος.



Σχήμα 12.1 – Κύρια σελίδα προσπέλασης της διαδικτυακής εφαρμογής

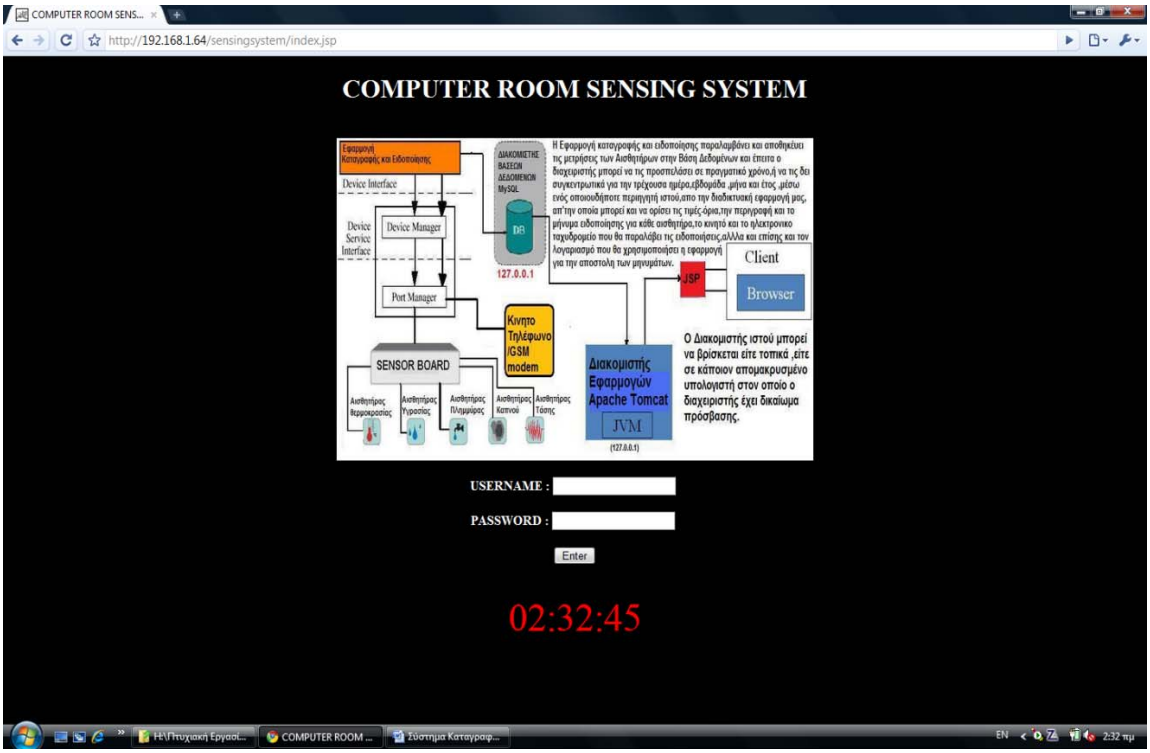
Στην παραπάνω εικόνα βλέπουμε την κύρια σελίδα της εφαρμογής στην οποία προβάλεται μια εικόνα σχηματικής αναπαράστασης του συστήματος μας, ένα ρολόι και τα πεδία του ονόματος και του κωδικού που οφείλει να συμπληρώσει ο διαχειριστής για να προσπελάσει την κύρια σελίδα της εφαρμογής μας. Εάν δοθεί είτε λανθασμένο όνομα χρήστη, είτε λανθασμένος κωδικός πρ'οσβασης, τότε ο χρήστης ανακατευθείνεται προς την σελίδα

μηνυμάτων επιτυχίας/αποτυχίας της εφαρμογής μας για να διαβάσει το κατάλληλο μήνυμα ειδοποίησης με τον λόγο για τον οποίο απέτυχε η διαδικασία εισόδου του στην εφαρμογή και ανακατευθύνεται στην αρχική σελίδα με τη φόρμα εισόδου για ξανα δώσει τα στοιχεία εισόδου του.



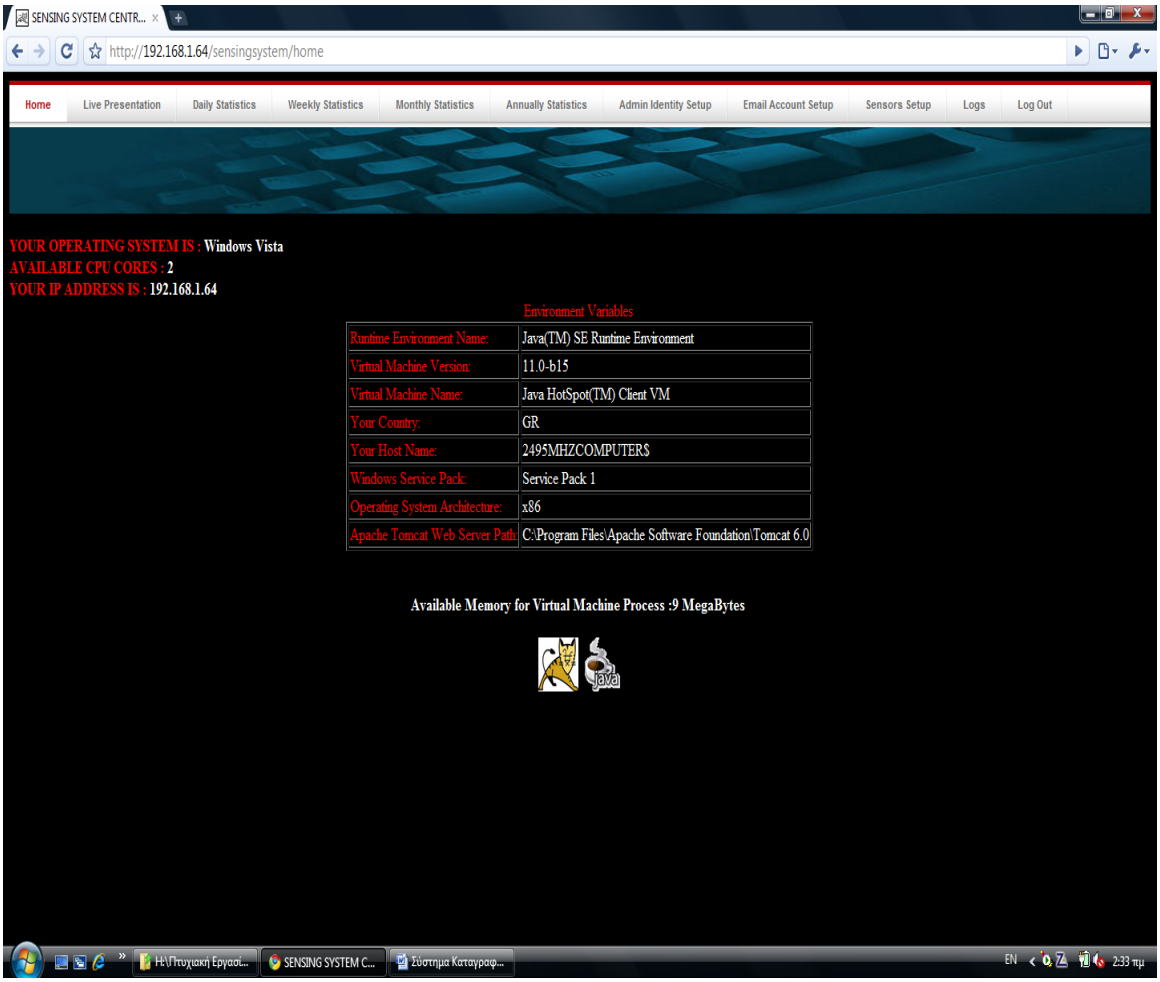
Σχήμα 12.2.α – Σφάλμα λόγω λανθασμένου ονόματος χρήστη, ή κωδικού προσβασης

Οι αποτυχημένες απόπειρες σύνδεσης στην εφαρμογή, αλλά και κάθε άλλη, είτε αποτυχημένη, είτε επιτυχημένη ενέργεια που επιτελείται μέσα στο περιβάλλον της εφαρμογής, καταγράφεται σε πίνακα της βάσης δεδομένων με την ακριβή χρονική στιγμή, τον τύπο της ενέργειας, αλλά και την διεύθυνση διαδικτύου του υπολογιστή από τον οποίο εκτελέστηκε η όποια ενέργεια προς την εφαρμογή.



Σχήμα 12.2.b – Επιστροφή στην αρχική σελίδα λόγω σφάλματος κατά την είσοδο

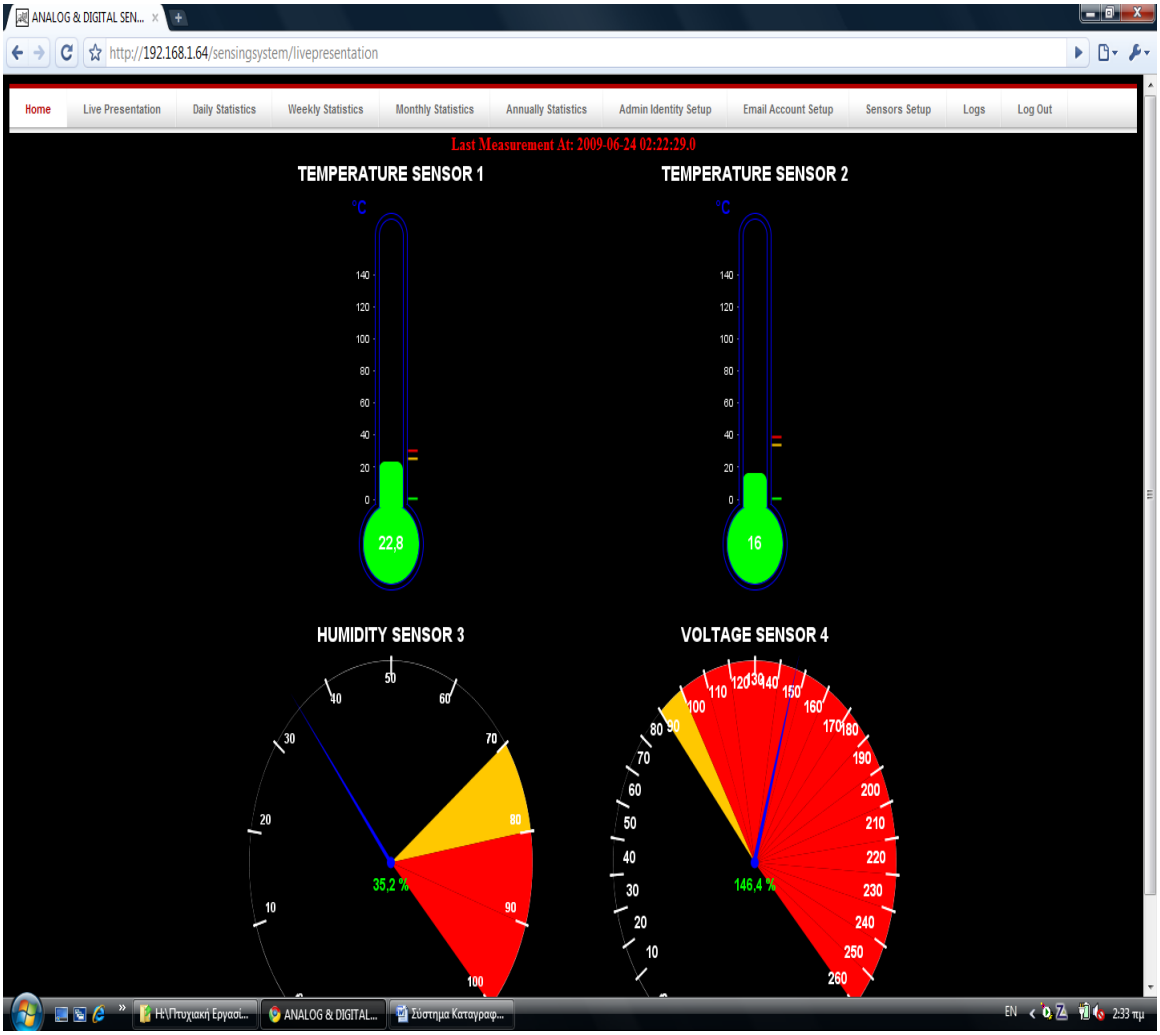
Εάν δόσει σωστά στοιχεία εισόδου, τότε ανακατευθύνεται στην σελίδα που βλέπουμε παρακάτω, στην home, η οποία περιέχει το κύριο μενού πλοήγησης της εφαρμογής μας. Στην σελίδα αυτή, παρατηρούμε πως προβάλλεται η διεύθυνση διαδικτύου του υπολογσιστή από τον οποίο προσπελαυνεταί η εφαρμογή μας, ο αριθμός των επεξεργαστών του υπολογιστή που τρέχει η εφαρμογή, αλλά και το λειτουργικό σύστημα το οποίο φιλοξενεί τον διακομιστή με την διαδικτυακή εφαρμογή.



Σχήμα 12.3 – Κύρια σελίδα πλοήγησης στην διαδικτυακή εφαρμογή

Ο διαχειριστής, επιλέγοντας “Live Presentation” ανακατευθύνεται στην κατάλληλη σελίδα jsp στην οποία προβάλλονται τα αποτελέσματα της καταγραφής των μετρήσεων και από τους τέσσερις αναλογικούς αισθητήρες σε δυο θερμόμετρα για τους αισθητήρες θερμοκρασίας και σε δυο κοντέρ για τον αισθητήρα υγρασίας και τον αισθητήρα τάσης δικτύου. Οι τιμές που αναπαρίστανται, διαβάζονται από τον πίνακα των αναλογικών μετρήσεων της βάσης δεδομένων, που τις αποθηκεύει το πρόγραμμα καταγραφής μας. Τα όρια που διακρίνονται στα δυο θερμόμετρα και στα δυο κοντέρ, διαμορφώνονται αναλόγως κάθε φορά, με βάση τα αποθηκευμένα όρια που δύναται να θέσει και να αλλάξει όποτε θελήσει, ο χρήστης της διαδικτυακής εφαρμογής μας. Για τους 2 ψηφιακούς αισθητήρες, καπνού και πλημμύρας, δεν υπάρχει αναπαράσταση διότι οι πιθανές τιμές του είναι είτε «Αλήθεια» (true), είτε «Ψέμα» (false). Εάν κάποιος, είτε και οι δύο, από τους αισθητήρες αποστείλλουν την τιμή της αλήθειας (Λογικό 1), τότε το πρόγραμμα

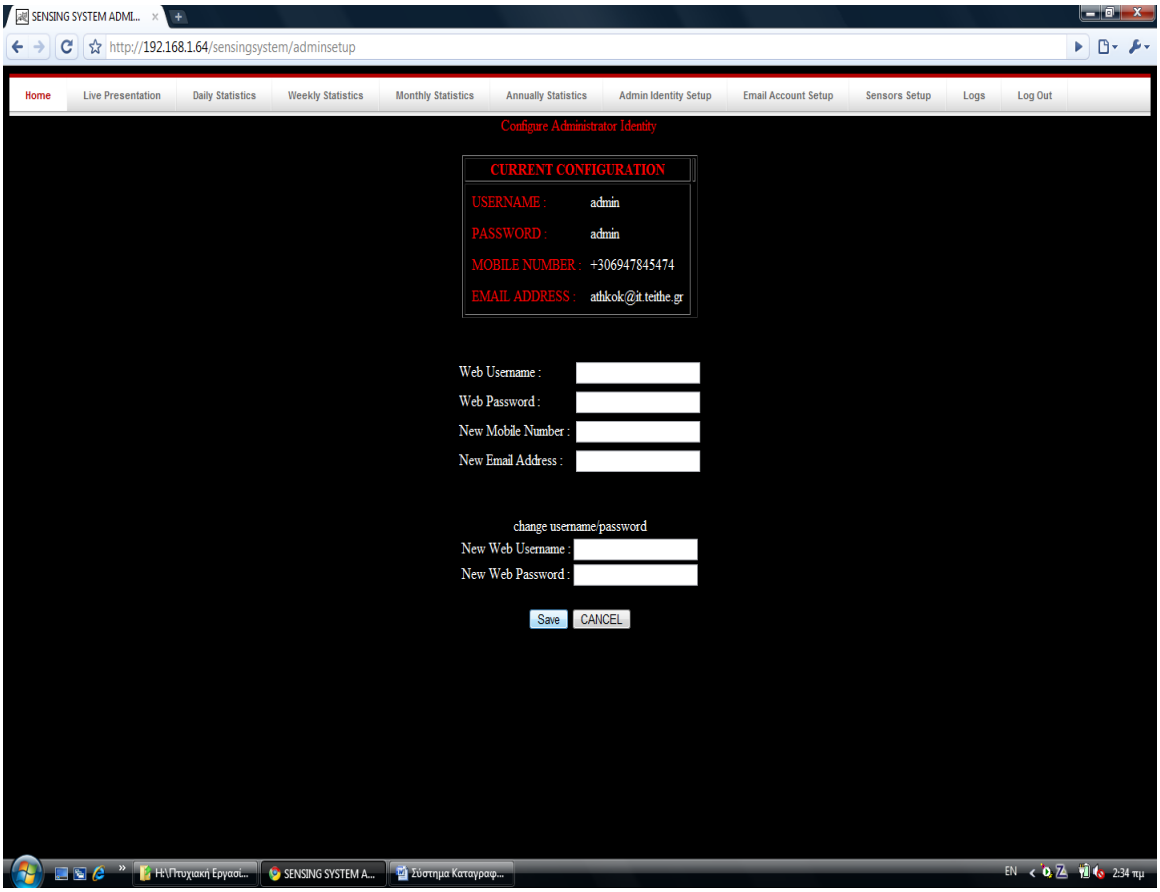
καταγραφής θα φροντήσει να ειδοποιήσει τον διαχειριστή με γραπτό μήνυμα στο κινητό του τηλέφωνο και με μήνυμα ηλεκτρονικού ταχυδρομείου ότι εντοπίστηκε είτε καπνός, είτε/και πλημμύρα. Η σελίδα κανει αυτόματα ανανέωση κάθε 5 δευτερόλεπτα, ουτως ώστε να ενημερώνεται με τις τελευταίες, πιο πρόσφατες τιμές που έχει καταγράψει η εφαργμογή μας.



Σχήμα 12.4 – Προβολή περιβαλλοντολογικών συνθηκών σε πραγματικό χρόνο

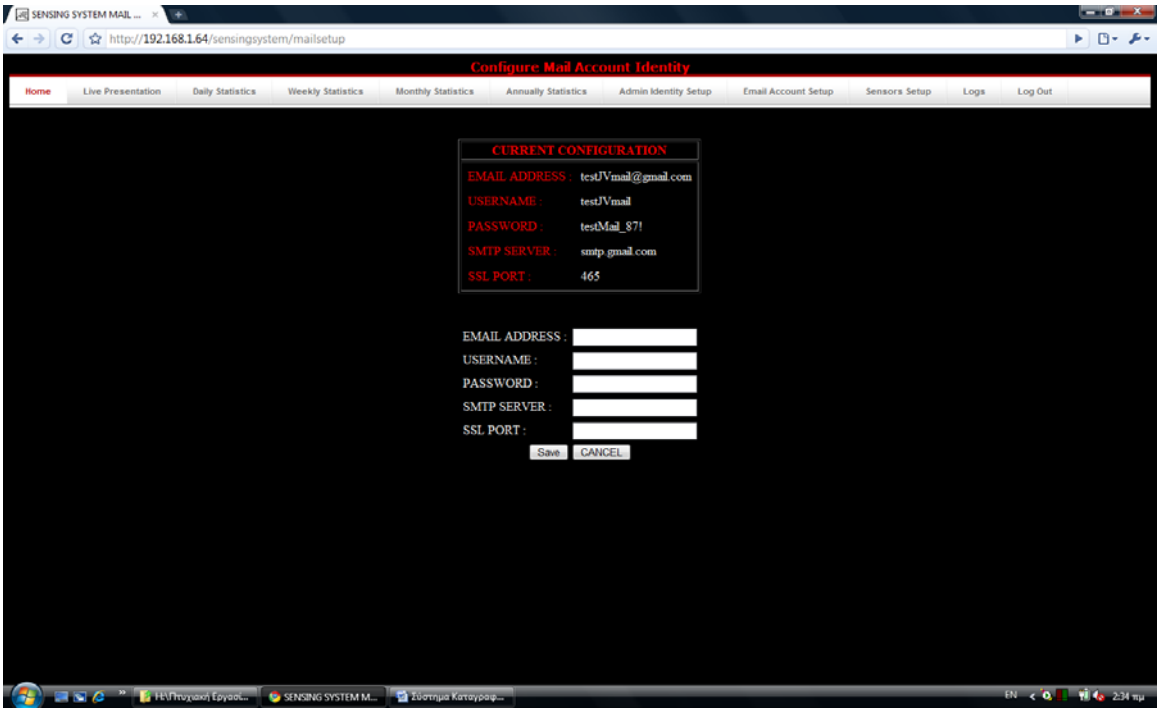
Ο χρήστης της εφαρμογής μας, μέσω της επιλογής “Sensors Setup”, μπορεί να αλλάξει την ταυτότητα που χρησιμοποιει για την σύνδεση του στην εφαρμογή, μπορεί να αλλάξει τον αριθμό του κινητού τηλεφώνου στον οποίο θα ειδοποιηθεί εάν ξεπεραστούν τα όρια που έχει θέσει για κάθε ένα από τους αναλογικούς αισθητήρες, είτε εάν στείλουν την τιμή της αλήθειας οι ψηφιακοί αισθητήρες, αλλά και επίσης μπορεί να αλλάξει και τον λογαριασμό ηλεκτρονικού ταχυδρομείου στον οποίο θα λάβει το μήνυμα ειδοποίησης που ο ίδιος έχει θέσει. Για να μπορέσει βέβαια να πραγματοποιήσει οποιαδήποτε από

τις παραπάνω επιλογές, θα πρέπει να δώσει εκ νέου το όνομα χρήστη και τον κωδικό που χρησιμοποιήσε για να συνδεθεί στην εφαρμογή. Στην ίδια σελίδα μπορεί να δεί τις τρέχουσες αποθηκευμένες παραμέτρους, εκτός από τον κωδικό πρόσβασης, ώστε να διασφαλίσουμε ότι δεν θα τον δεί καποιο τρίτο μάτι και θα τον χρησιμοποιήσει για να συνδεθεί κρυφά στην εφαρμογή.



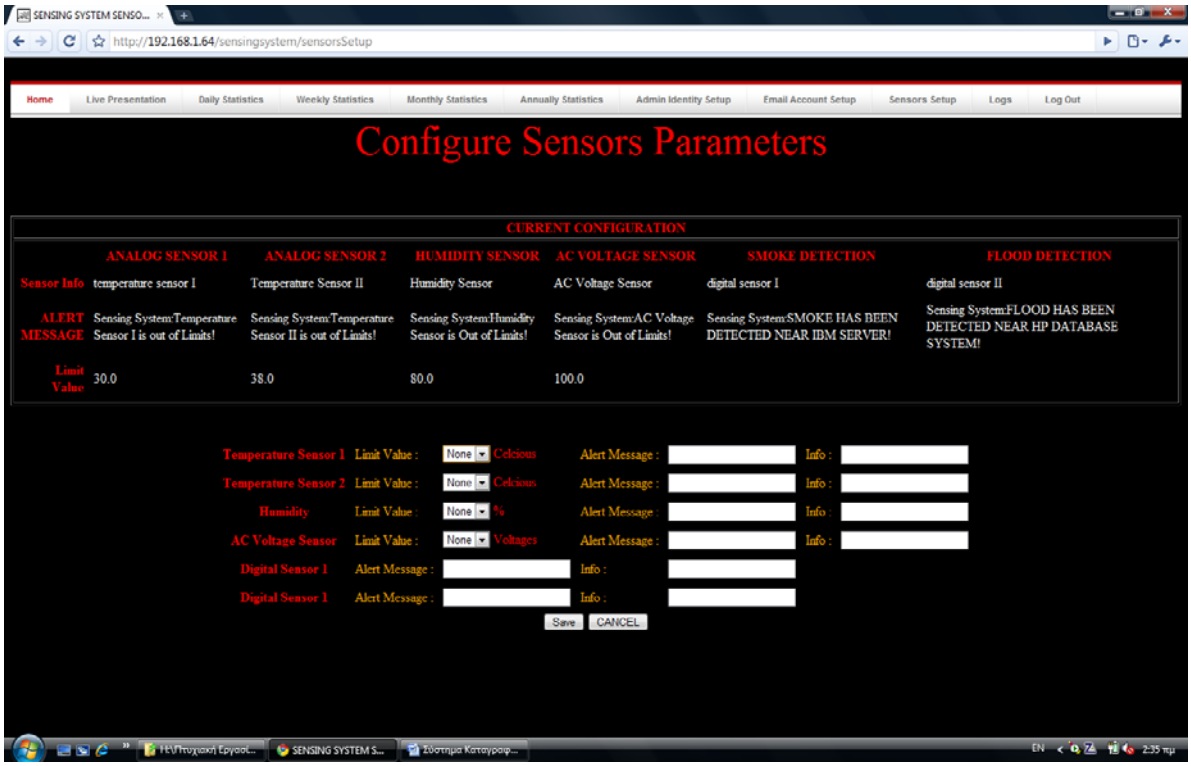
Σχήμα 12.5 – Σελίδα τροποποίησης της ταυτότητας του διαχειριστή

Επιλέγοντας από το menu “Email Account Setup” ο διαχειριστής μπορεί να ορίσει τα στοιχεία του λογαριασμού που θα χρησιμοποιεί η εφαρμογή επιτήρησης περιβαλλοντολογικών συνθηκών, για να αποστέλλει ειδοποίηση με μήνυμα ηλεκτρονικού ταχυδρομείου στο λογαρισμό του, εάν ξεπεραστούν τα όρια που έχει θέσει για τους αισθητήρες. Ταυτόχρονα προβάλλονται οι τρέχουσες αποθηκευμένες παράμετροι.

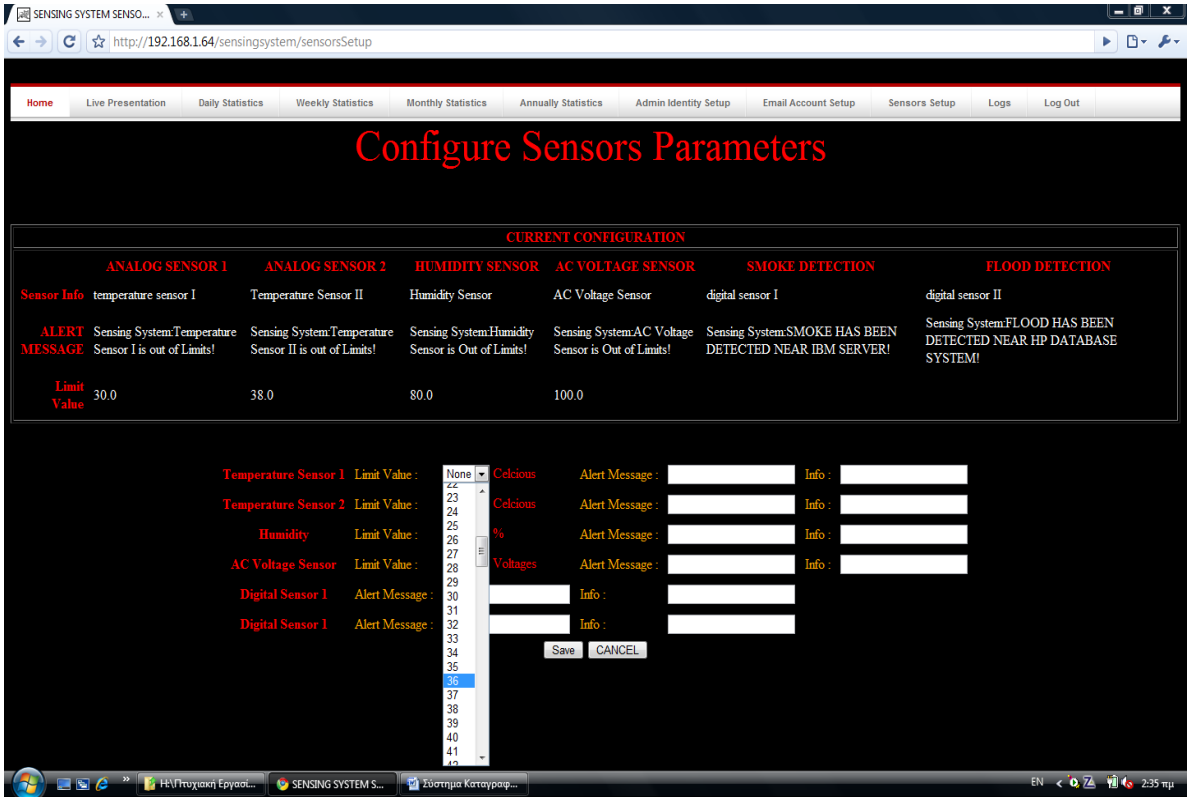


Σχήμα 12.6 – Σελίδα τροποποίησης του λογαριασμού email του συστήματος

Τέλος ο χρήστης από το menu “Sensors Setup” μπορεί να ορίσει οποιαδήποτε στιγμή το θελήσει τα όρια για κάθε έναν από τους αναλογικούς αισθητήρες, τα μηνύματα που θα λάβει στο κινητό του τηλέφωνο και στον λογαριασμό ηλεκτρονικής αλληλογραφίας, εάν ξεπεραστούν τα όρια αυτά, ή εάν καταφθάσει η τιμή της αλήθειας (true) για κάποιον από τους ψηφιακούς αισθητήρες, αλλά και μήνυματα περιγραφής της τοποθεσίας για κάθε έναν από αυτούς.



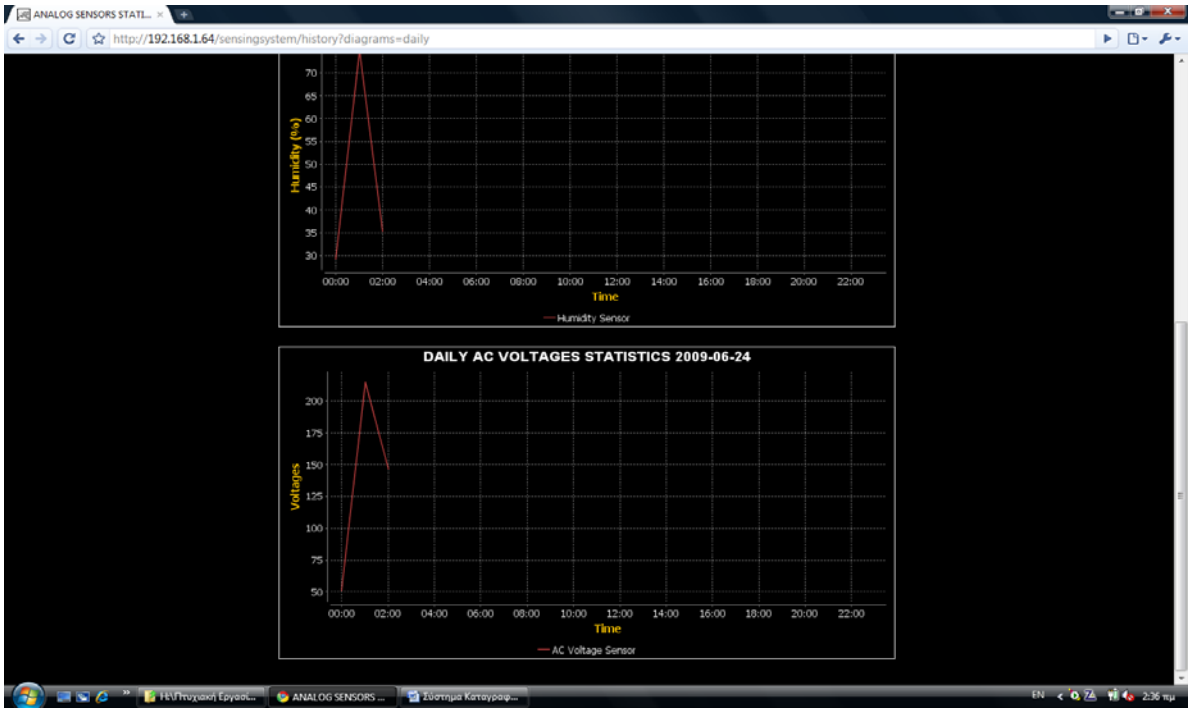
Σχήμα 12.7.α – Σελίδα τροποποίησης των ορίων τιμών των αισθητήρων, των μηνυμάτων περιγραφής της τοποθεσίας τους και των μηνυμάτων ειδοποίησης του διαχειριστή



Σχήμα 12.7.b – Σελίδα τροποποίησης των ορίων τιμών των αισθητήρων, των μηνυμάτων περιγραφής της τοποθεσίας τους και των μηνυμάτων ειδοποίησης του διαχειριστή

Από την επιλογή “Daily Statistics” του μενού, ο χρήστης μπορεί να δει συγκεντρωτικά όλες τις καταγεγραμμένες μετρήσεις θερμοκρασίας, υγρασίας και τάσης δικτύου για την τρέχουσα ημέρα.

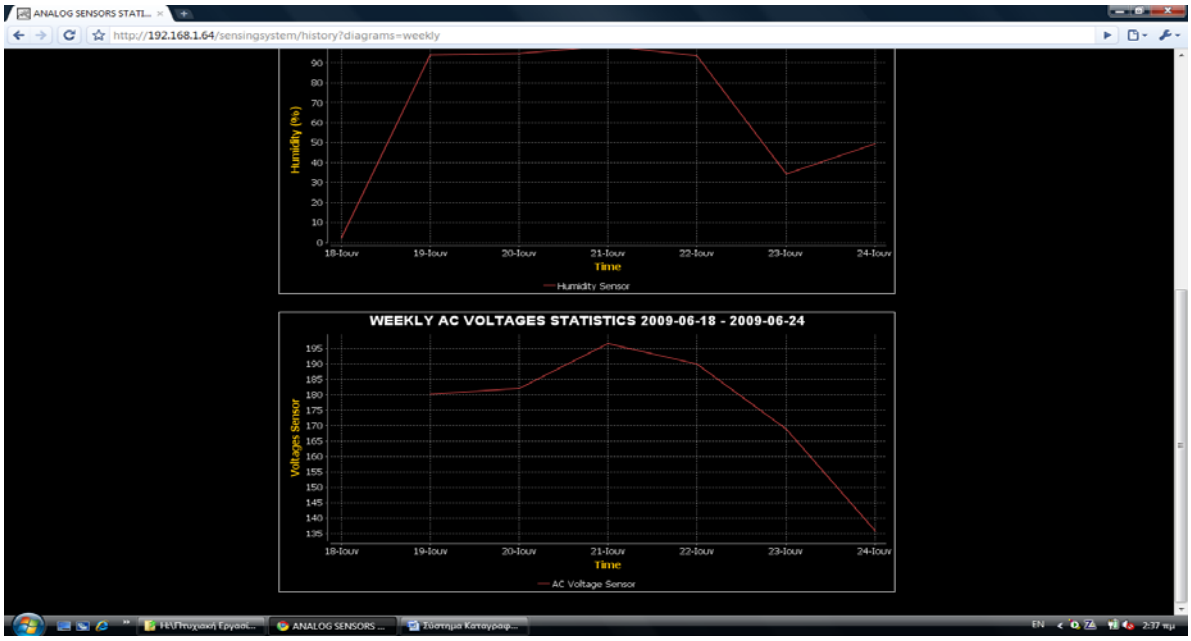
Σχήμα 12.8.α – Προβολή ημερήσιων διαγραμμάτων ολών των καταγεγραμμένων μετρήσεων των αισθητήρων



Σχήμα 12.8.b – Προβολή ημερήσιων διαγραμμάτων ολών των καταγεγραμμένων μετρήσεων των αισθητήρων

Από την επιλογή “Weekly Statistics” του μενού, ο χρήστης μπορεί να δει συγκεντρωτικά όλες τις καταγεγραμμένες μετρήσεις θερμοκρασίας, υγρασίας και τάσης δικτύου για την τρέχουσα εβδομάδα.

Σχήμα 12.9.a – Προβολή εβδομαδιαίων διαγραμμάτων ολών των καταγεγραμμένων μετρήσεων των αισθητήρων



Σχήμα 12.9.b – Προβολή εβδομαδιαίων διαγραμμάτων ολών των καταγεγραμμένων μετρήσεων των αισθητήρων

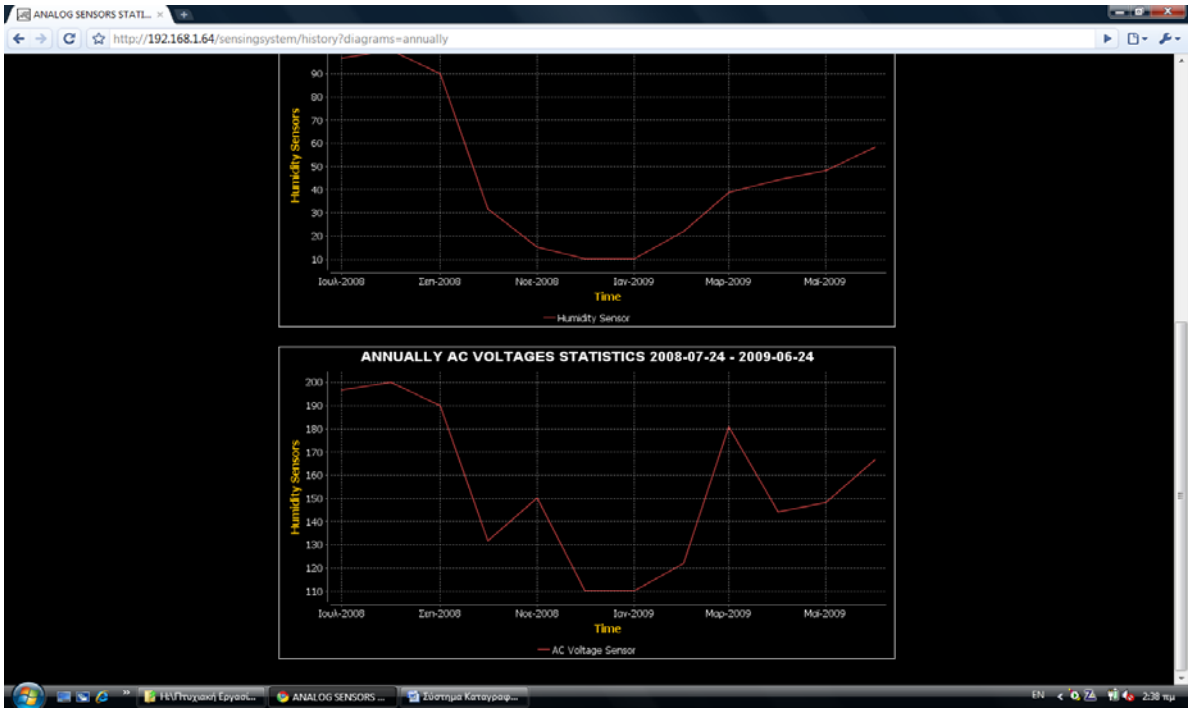
Από το επιλογή “Monthly Statistics” του μενού, ο χρήστης μπορεί να δει συγκεντρωτικά όλες τις καταγεγραμμένες μετρήσεις θερμοκρασίας, υγρασίας και τάσης δικτύου για τον τρέχον μήνα.

Σχήμα 12.10.a – Προβολή μηνιαίων διαγραμμάτων ολών των καταγεγραμμένων μετρήσεων των αισθητήρων

Σχήμα 12.10.b – Προβολή μηνιαίων διαγραμμάτων ολών των καταγεγραμμένων μετρήσεων των αισθητήρων

Με την επιλογή “Annually Statistics” από το μενού, ο χρήστης μπορεί να δεί συγκεντρωτικά όλες τις καταγεγραμμένες μετρήσεις θερμοκρασίας, υγρασίας και τάσης δικτύου για το τρέχον έτος.

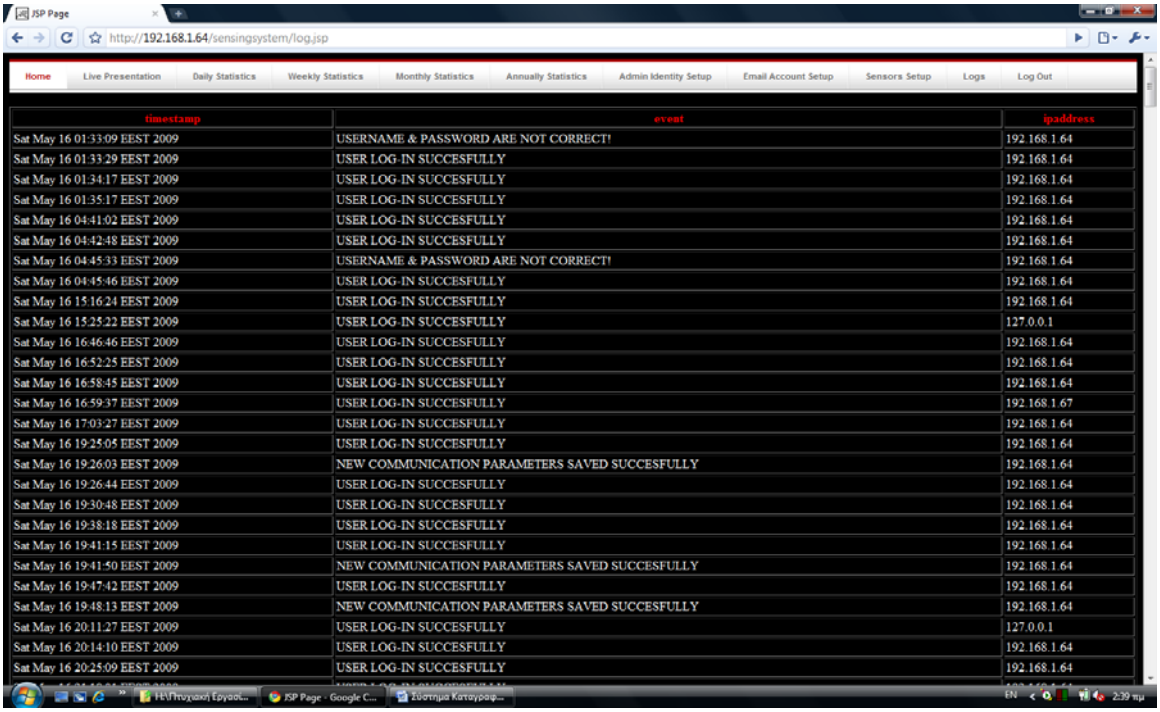
Σχήμα 12.11.a – Προβολή ετήσιων διαγραμμάτων όλων των καταγεγραμμένων μετρήσεων των αισθητήρων



Σχήμα 12.11.b – Προβολή ετήσιων διαγραμμάτων όλων των καταγεγραμμένων μετρήσεων των αισθητήρων

Μεσω της επιλογής “Logs” από το μενού, ο χρήστης μπορεί να δει τα περιεχόμενα του πίνακα καταγραφής συμβάντων και ενεργειών της εφαρμογής. Εδώ παρουσιάζονται με τη μορφή πίνακα σε κάθε εγγραφή: η

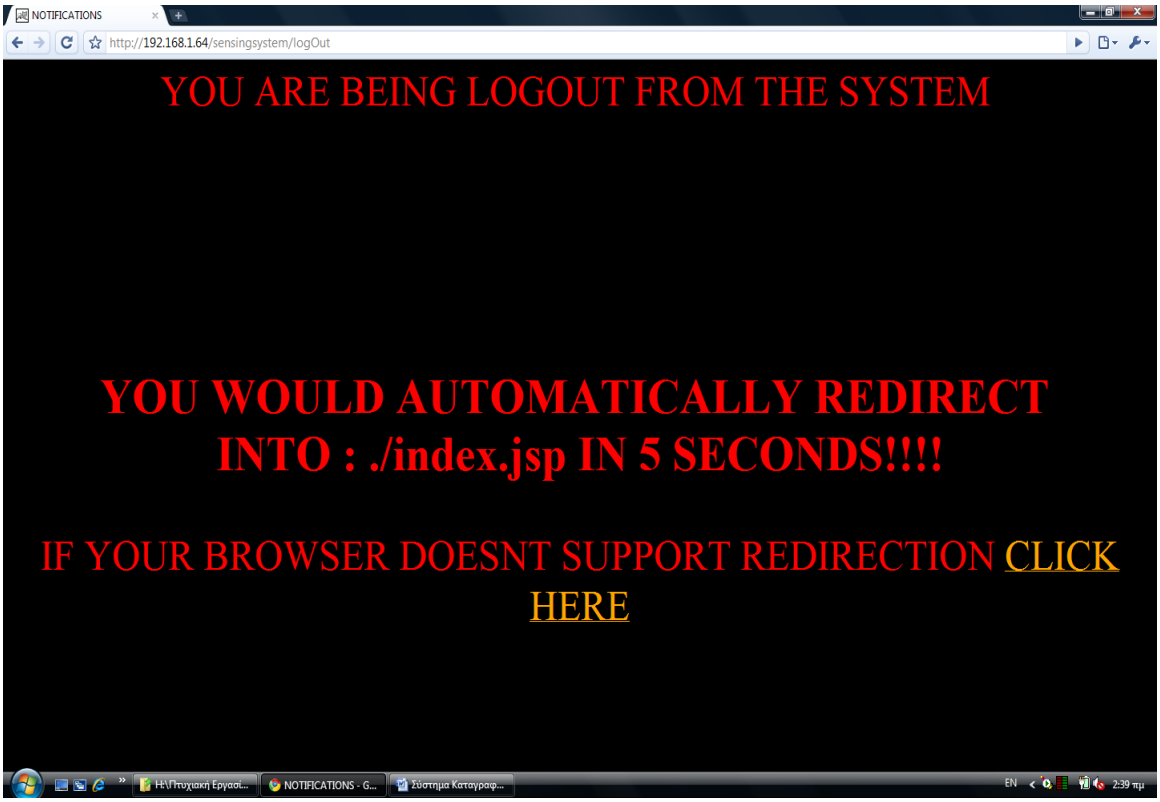
χρονική στιγμή που συνέβη κάποια επιτυχημένη, ή αποτυχημένη ενέργεια, μια περιγραφή του συμβάντος και η διεύθυνση διαδικτύου του υπολογιστή από τον οποίο προκλήθηκε η ενέργεια αυτή.



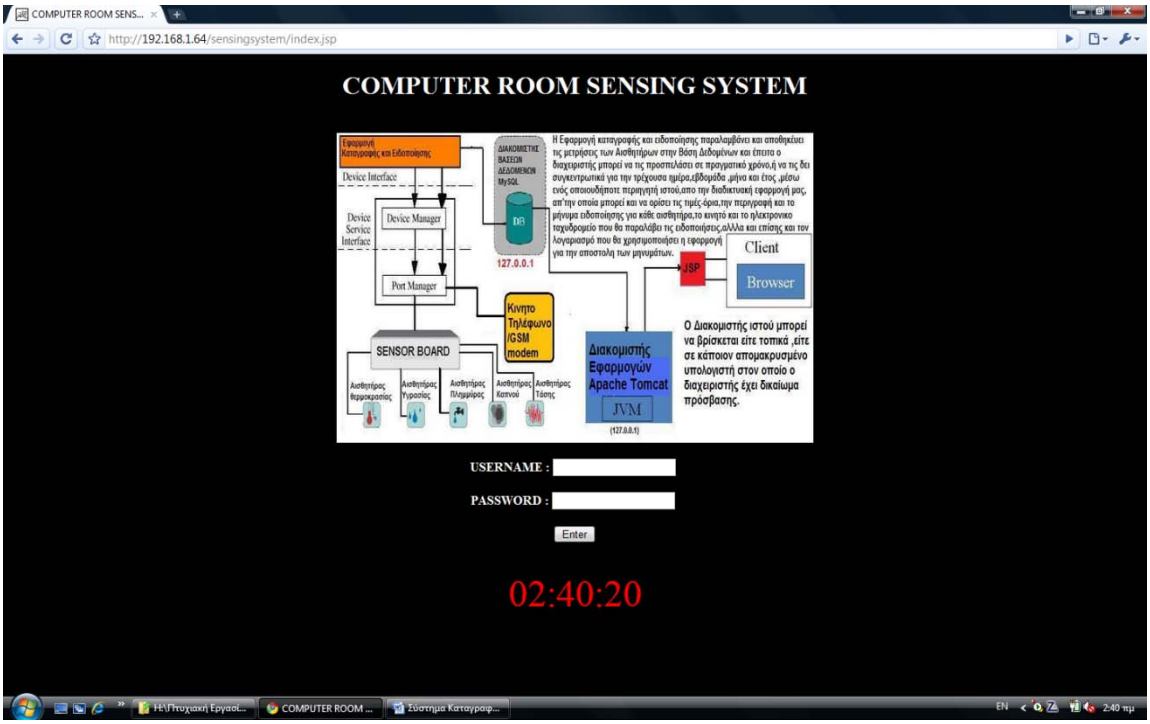
timestamp	event	ipaddress
Sat May 16 01:33:09 EEST 2009	USERNAME & PASSWORD ARE NOT CORRECT!	192.168.1.64
Sat May 16 01:33:29 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 01:34:17 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 01:35:17 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 04:41:02 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 04:42:48 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 04:45:33 EEST 2009	USERNAME & PASSWORD ARE NOT CORRECT!	192.168.1.64
Sat May 16 04:45:46 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 15:16:24 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 15:25:22 EEST 2009	USER LOG-IN SUCCESFULLY	127.0.0.1
Sat May 16 16:46:46 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 16:52:25 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 16:58:45 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 16:59:37 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.67
Sat May 16 17:03:27 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 19:25:05 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 19:26:03 EEST 2009	NEW COMMUNICATION PARAMETERS SAVED SUCCESFULLY	192.168.1.64
Sat May 16 19:26:44 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 19:30:48 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 19:38:18 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 19:41:15 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 19:41:50 EEST 2009	NEW COMMUNICATION PARAMETERS SAVED SUCCESFULLY	192.168.1.64
Sat May 16 19:47:42 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 19:48:13 EEST 2009	NEW COMMUNICATION PARAMETERS SAVED SUCCESFULLY	192.168.1.64
Sat May 16 20:11:27 EEST 2009	USER LOG-IN SUCCESFULLY	127.0.0.1
Sat May 16 20:14:10 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64
Sat May 16 20:25:09 EEST 2009	USER LOG-IN SUCCESFULLY	192.168.1.64

Σχήμα 12.12 – Προβολή επιτυχημένων/αποτυχημένων ενεργείων προς την εφαρμογή

Τέλος με την επιλογή “Log Out” από το μενού, ο χρήστης αποσυνδέεται από την εφαρμογή, τερματίζεται η τρέχουσα συνεδρία και ανακατευθύνεται διαμέσου της σελίδας των μηνυμάτων επιτυχίας/αποτυχίας στην αρχική σελίδα σύνδεσης της εφαρμογής.



Σχήμα 12.13 – Αποσύνδεση από την εφαρμογή



Σχήμα 12.14 – Επιστροφή στην σελίδα εισόδου της εφαρμογής