ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

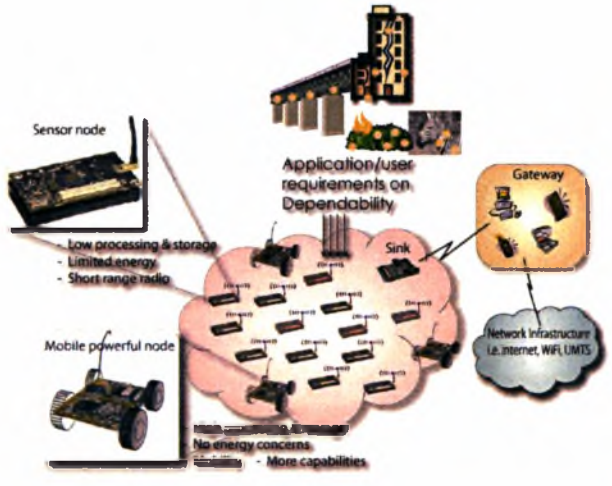
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ & ΔΙΚΤΥΩΝ

ΚΑΤΑΓΡΑΦΗ ΚΑΙ ΑΠΕΙΚΟΝΙΣΗ ΔΕΔΟΜΕΝΩΝ ΔΙΚΤΥΟΥ ΑΣΥΡΜΑΤΩΝ ΑΙΣΘΗΤΗΡΩΝ ΣΕ ΠΡΑΓΜΑΤΙΚΟ ΧΡΟΝΟ

ΠΕΡΛΕΠΕΣ ΛΕΩΝΙΔΑΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Αναπ. Καθηγητής Γεώργιος Σταμούλης
2ος Βαθμολογητής : Δρ. Παναγιώτης Κίκιρας



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Περίληψη

Με τον όρο «ασύρματο δίκτυο αισθητήρων» (Wireless Sensor Network) εννοούμε ένα ασύρματο δίκτυο που αποτελείται από χωρικά κατανεμημένες αυτόνομες συσκευές. Αυτές οι συσκευές χρησιμοποιούν αισθητήρες για να ελέγξουν φυσικές ή περιβαλλοντικές συνθήκες, όπως την θερμοκρασία, τον ήχο , τις ταλαντώσεις, την πίεση, την κίνηση ή τους ρύπους σε διαφορετικές θέσεις.

Στην συγκεκριμένη εργασία περιγράφουμε ένα σύστημα καταγραφής και απεικόνισης των παραπάνω μετρήσεων. Οι μετρήσεις αυτές αρχικά παρουσιάζονται μέσα από ένα γραφικό περιβάλλον στον χρήστη, και έπειτα αποθηκεύονται σε μια κατάλληλη βάση δεδομένων, ώστε να είναι διαθέσιμες για μελλοντική χρήση.

Abstract

With the term "wireless network of sensors" (Wireless Sensor Network) we suggest a wireless network that is constituted by territorial distributed autonomous appliances. These appliances use sensors in order to check natural or environmental conditions, as the temperature, the sound, the oscillations, the pressure, the movement or the pollutants in different places.

In this particular thesis we describe a system of recording and depiction of the above measurements. These measurements initially are presented through a graphic environment from the side of the user, and then are stored in a suitable base of data, so as to be available for future use.

Πίνακας περιεχομένων

ΠΕΡΙΛΗΨΗ 2

ABSTRACT 3

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ 4

1 ΕΙΣΑΓΩΓΗ 6

1.1 Αντικειμενο της πτυχιακης 6
1.2 Οργανωση του τομου 6

2 ΔΙΚΤΥΑ ΑΙΣΘΗΤΗΡΩΝ 8

2.1 Τι ειναι ενα ασυρματο δικτυο αισθητηρων 8
2.2 Εφαρμογες στα ασυρματα δικτυα αισθητηρων 8
2.3 Στοχοι και απαιτησεις ενος δικτυου αισθητηρων 9

3 ΑΙΣΘΗΤΗΡΕΣ 11

3.1 Δομη των αισθητηρων 11
3.2 Το λειτουργικο συστημα TinyOS 12
3.3 Η γλωσσα προγραμματισμου nesC 13

4 ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ ΤΗΣ ΕΦΑΡΜΟΓΗΣ 15

4.1 Η χρηση της UML. 15
4.2 Διαδικασιας αναπτυξης της εφαρμογης. 15
4.2.1 Διάγραμμα περιπτώσεων χρήσης - Use Case Diagram *16*
4.2.2 Διαγράμματα Ακολουθίας - Sequence Diagrams *17*
4.2.3 Διαγράμματα Καταστάσεων - State Diagrams *20*
4.2.4 Διάγραμμα Δραστηριότητας – Activity Diagram *21*

5 ΤΑ ΣΥΣΤΑΤΙΚΑ ΤΗΣ ΕΦΑΡΜΟΓΗΣ. 22

5.1 Οι αισθητηρες « Tmote Sky ». 22
5.1.1 Τι είναι τα « Tmote Sky ». *22*
5.1.2 Το λειτουργικό σύστημα των Tmote Sky (Boomerang) *24*
5.1.3 Εγκατάσταση και προγραμματισμός των Tmote Sky. *24*
5.2 ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ 26
5.2.1 Η χρήση της MySQL *26*
5.2.2 Η χρήση του JDBC *27*

6 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ 28

6.1 Παρουσιαση της αρχιτεκτονικης 28
6.2 Δομη της βασης δεδομενων 30

7 ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ 32

7.1 Προετοιμασια των αισθητηρων. 32
7.2 Παρουσιαση του GUI 33
7.3 Μετασχηματισμοι των δεδομενων 34
7.4 Εκτελεση της εφαρμογης 35

8 ΕΠΙΛΟΓΟΣ 39

8.1 Μελλοντικες δυνατοτητες επεκτασης. 39
8.2 Συμπερασματα 39

9 ΒΙΒΛΙΟΓΡΑΦΙΑ 40

1 ΕΙΣΑΓΩΓΗ

1.1 Αντικείμενο της πτυχιακής

Σκοπός της πτυχιακής εργασίας είναι να διαβάσουμε τις μετρήσεις του περιβάλλοντος από τους κόμβους αισθητήρων που διαθέτουμε και οι οποίοι απαρτίζουν ένα ασύρματο δίκτυο αισθητήρων. Οι μετρήσεις αυτές θα λαμβάνονται ασύρματα από ένα κόμβο «δέκτη», ο οποίος θα προωθεί τι μετρήσεις στην εφαρμογή. Έπειτα θα υπάρχει η δυνατότητα εμφάνισης αυτών σε γραφικό περιβάλλον, αλλά και την αποθήκευση τους σε βάση δεδομένων.

Οι έννοιες που περιλαμβάνουν το μελετητικό τμήμα της εργασίας και οι οποίες θα αναλυθούν στα διάφορα κεφάλαια αυτής, περιλαμβάνουν τους όρους: TinyOS, tmote SKY sensor, Cygwin, UML, MySQL.

1.2 Οργάνωση του τόμου

Στα κεφάλαια που θα ακολουθήσουν θα αναλυθούν οι ανωτέρω αναφερθείσες έννοιες, οι οποίες είναι αναγκαίες για την κατανόηση της ανάλυσης της υλοποίηση της εφαρμογής.

Συγκεκριμένα, στο πρώτο κεφάλαιο παρουσιάζεται μια σύντομη περιγραφή της δομής, και του αντικειμένου που πραγματεύεται η συγκεκριμένη εργασία.

Στο δεύτερο κεφάλαιο γίνεται μια περιγραφή των ασύρματων δικτύων αισθητήρων. Παρουσιάζονται οι δυνατότητες και ιδιαιτερότητες τους μέσα από ένα πλήθος εφαρμογών.

Στο τρίτο κεφάλαιο γίνεται μια περιγραφή των ασύρματων αισθητήρων. Περιγράφονται οι δυνατότητες τους, η αρχιτεκτονική τους, το λειτουργικό σύστημα το οποίο χρησιμοποιούν, καθώς και διάφορα εργαλεία τα οποία είναι απαραίτητα για τον προγραμματισμό τους.

Στο τέταρτο κεφάλαιο γίνεται μια περιγραφή της διαδικασίας που ακολουθήθηκε για την σχεδίαση και ανάπτυξη της εφαρμογής μας. Παρουσιάζεται δηλαδή η χρήση της γλώσσα UML, καθώς και τα αποτελέσματα που πήραμε από την χρήση της.

Στο πέμπτο κεφάλαιο παρουσιάζουμε τα συστατικά της εφαρμογής. Περιγράφουμε δηλαδή, τους αισθητήρες Tmote Sky και τα χαρακτηριστικά που έχουν αυτοί. Ακόμα παρουσιάζεται το λειτουργικό των Tmote Sky, το Boomerang. Στο δεύτερο μέρος του κεφαλαίου υπάρχει η περιγραφή της βάσης δεδομένων και του σχήματος που αυτή έχει. Τέλος περιγράφεται η χρησιμότητα και η λειτουργικότητα του JDBC API.

Στο επόμενο κεφάλαιο γίνεται μια πλήρη περιγραφή της αρχιτεκτονικής τους συστήματος. Περιγράφεται δηλαδή αναλυτικά τα διάφορα μέρη της εφαρμογής, και την λειτουργικότητα που εκτελούν αυτά.

Η περιγραφή της υλοποίησης της εφαρμογής περιγράφεται στο έβδομο κεφάλαιο. Περιέχεται δηλαδή η διαδικασία προετοιμασίας των αισθητήρων, η υλοποίηση του προγράμματος GUI, η σύνθεση ειδικών εντολών που χρησιμοποιούνται για τον έλεγχο και την εκτέλεση της εφαρμογής. Τέλος παρουσιάζονται διάφορα στιγμιότυπα, της εκτέλεσης της εφαρμογής.

Στο όγδοο κεφάλαιο συνοψίζουμε τα αποτελέσματα τις εργασίας, παρουσιάζουμε τα εξαγόμενα συμπεράσματα και δίνουμε μια περιγραφή των δυνατοτήτων επέκτασης της εφαρμογής.

Τέλος παρουσιάζονται οι πηγές από οπού αντλήσαμε τις πληροφορίες για την υλοποίηση και ανάπτυξη του αντικειμένου της εργασίας αυτής.

2 Δίκτυα αισθητήρων

2.1 Τι είναι ένα ασύρματο δίκτυο αισθητήρων

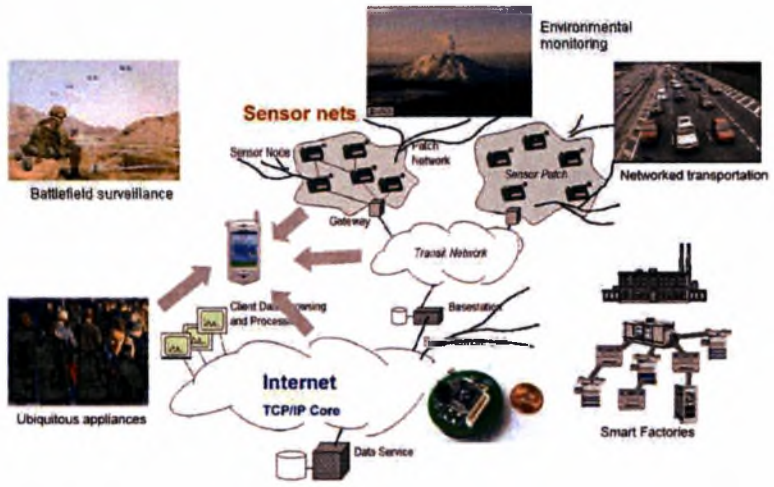
Ένα ασύρματο δίκτυο αισθητήρων (Wireless Sensor Network) είναι ένα ασύρματο δίκτυο που αποτελείται από χωρικά κατανεμημένες αυτόνομες συσκευές. που χρησιμοποιούν αισθητήρες για να ελέγξουν τις φυσικές ή περιβαλλοντικές συνθήκες, όπως η θερμοκρασία, ο ήχος , η δόνηση, η πίεση, η κίνηση ή οι ρύποι στις διαφορετικές θέσεις. Κάθε αισθητήρας έχει ικανότητα ασύρματης επικοινωνίας, δυνατότητες επεξεργασίας σήματος και δικτύωσης των στοιχείων. Η ανάπτυξη των ασύρματων δικτύων αισθητήρων παρακινήθηκε αρχικά από τι στρατιωτικές εφαρμογές, όπως η επιτήρηση πεδίων μαχών. Εντούτοις, τα ασύρματα δίκτυα αισθητήρων χρησιμοποιούνται τώρα σε πολλούς πολιτικούς τομείς, συμπεριλαμβανομένου του περιβάλλοντος και της παρακολούθησης βιότοπων, των εφαρμογών υγειονομικής περίθαλψης, της οικιακής αυτοματοποίησης και του ελέγχου της κυκλοφορίας.

2.2 Εφαρμογές στα ασύρματα δίκτυα αισθητήρων

Οι εφαρμογές για τα ασύρματα δίκτυα αισθητήρων είναι πολλές και ποικίλες. Χρησιμοποιούνται σε εμπορικές και βιομηχανικές εφαρμογές για να ελέγχουν τα στοιχεία που θα ήταν δύσκολο ή ακριβό να ελεγχθούν από αισθητήρες συνδεδεμένους με καλώδιο. Θα μπορούσαν να επεκταθούν σε περιοχές όπου επικρατούν αντίξοες συνθήκες, όπου θα παρέμεναν για πολλά έτη (ελέγχοντας μερικές περιβαλλοντικές μεταβλητές) χωρίς την ανάγκη επαναφόρτισης /αντικατάστασης τις παροχές ηλεκτρικού ρεύματός τους. Μερικά παραδείγματα των ασύρματων δικτύων αισθητήρων είναι τα ακόλουθα:

- Στρατιωτικά δίκτυα αισθητήρων για την ανίχνευση και την λήψη όσο το δυνατόν περισσότερων πληροφοριών για τις εχθρικές κινήσεις, τις εκρήξεις, και άλλα φαινόμενα ενδιαφέροντος.
- Δίκτυα αισθητήρων για την ανίχνευση και τον χαρακτηρισμό των χημικών, βιολογικών, ραδιολογικών, πυρηνικών, και εκρηκτικών επιθέσεων καθώς και υλικών.
- Δίκτυα αισθητήρων για την ανίχνευση και τον έλεγχο των περιβαλλοντικών αλλαγών στις πεδιάδες, τα δάση, τους ωκεανούς, κλπ.

- Ασύρματα δίκτυα αισθητήρων ρύθμισης κυκλοφορίας για τον έλεγχο της κυκλοφορίας οχημάτων στις εθνικές οδούς ή στα κορεσμένα μέρη μιας πόλης.
- Ασύρματα δίκτυα αισθητήρων επιτήρησης για παροχή ασφάλειας σε εμπορικά πολυκαταστήματα, χώρους στάθμευσης, και άλλων εγκαταστάσεων.
- Ασύρματα δίκτυα αισθητήρων χώρων στάθμευσης για καθορισμό των σημείων που είναι κατειλημμένα και αυτών που είναι ελεύθερα.



Σχήμα 1: Εφαρμογές των Ασύρματων Δικτύων Αισθητήρων

(εικόνα από http://www.vancouver.wsu.edu/fac/song/projects.html)

2.3 *Στόχοι και απαιτήσεις ενός δικτύου αισθητήρων*

Οι βασικοί στόχοι ενός ασύρματου δικτύου αισθητήρων εξαρτώνται γενικά από την εφαρμογή, αλλά οι ακόλουθοι στόχοι είναι κοινοί για πολλές κατηγορίες δικτύων:

- Καθορισμός της αξίας κάποιας παραμέτρου σε μια δεδομένη θέση. Ένα δεδομένος κόμβος αισθητήρων δηλαδή να μπορεί να συνδεθεί με τους διαφορετικούς τύπους αισθητήρων, κάθε ένας με ένα διαφορετικό ρυθμό δειγματοληψίας των τιμών.
- Ανίχνευση της πραγματοποίησης ενός καθορισμένου γεγονότος, καθώς και ικανότητα πραγματοποίησης υπολογισμών πάνω στις παραμέτρους του πραγματοποιημένου γεγονότος.
- Ταξινόμηση ενός αντικειμένου. Π.χ. Σε ένα στρατιωτικό δίκτυο αισθητήρων, ακολουθείστε ένα εχθρικό όχημα καθώς κινείται μέσω της γεωγραφικής περιοχής που καλύπτεται από το δίκτυο.

Οι απαιτήσεις των ασύρματων δικτύων αισθητήρων περιλαμβάνουν τα εξής:

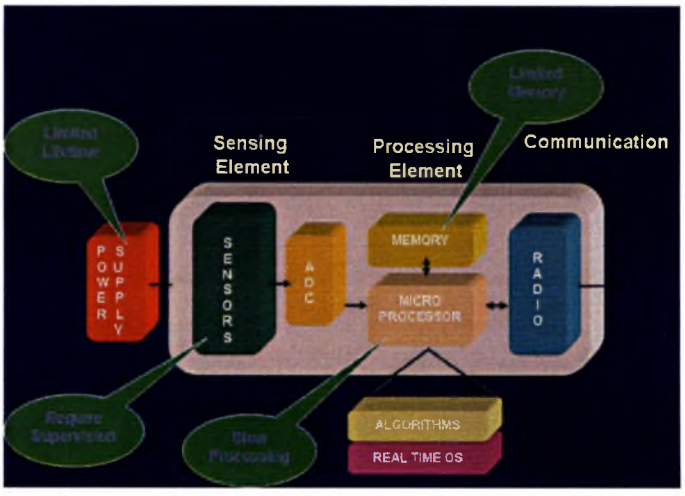
- Ικανότητα υλοποίησης και διαχείρισης δικτύων ασύρματων αισθητήρων, το μέγεθος των οποίων να φτάνει ακόμα και τους 100.000 κόμβους.
- Χαμηλές ενεργειακές απαιτήσεις. Δεδομένου ότι σε πολλές εφαρμογές οι κόμβοι αισθητήρων θα τοποθετηθούν σε μια απομακρυσμένη περιοχή, η τροφοδότηση ενός κόμβου μπορεί να μην είναι δυνατή. Σε αυτήν την περίπτωση, η διάρκεια ζωής ενός κόμβου μπορεί να καθοριστεί από τη ζωή μπαταριών, για αυτό τον λόγο απαιτείται η ελαχιστοποίηση τω ενεργειακών δαπανών.
- Δυνατότητα αυτό-οργάνωσης των δικτύων. Το δίκτυο πρέπει να είναι σε θέση να μετατρέπεται περιοδικά έτσι ώστε να μπορεί να συνεχίσει να λειτουργεί, ακόμα και στις περιπτώσεις όπου κάποιοι κόμβοι αποτύχουν (π.χ. λόγω έλλειψης ενέργειας), είτε ενσωματωθούν καινούριοι κόμβοι στο δίκτυο.
- Δυνατότητα άντλησης πληροφοριών. Κάποιος χρήστης μπορεί να θέλει τις πληροφορίες ενός μόνο κόμβου ή μιας ομάδας από κόμβους για συλλογή πληροφοριών για συγκεκριμένη περιοχή. Εξαιτίας της συγχώνευσης δεδομένων που πραγματοποιείται μπορεί να μην είναι δυνατόν να μεταδοθεί ένας μεγάλος όγκος πληροφοριών από το δίκτυο. Αντί αυτού, διάφοροι τοπικοί κόμβοι θα συλλέξουν δεδομένα από δεδομένες περιοχές και θα δημιουργήσουν περιληπτικά μηνύματα. Έτσι μία αίτηση για άντληση πληροφοριών από έναν κόμβο μπορεί να κατευθυνθεί στον πλησιέστερο τοπικό κόμβο που συλλέγει δεδομένα.

3 Αισθητήρες

3.1 Δομή των αισθητήρων

Οι κόμβοι αισθητήρων που χρησιμοποιούνται σε ένα δίκτυο αισθητήρων μπορούν να χαρακτηριστούν ως μικροί υπολογιστές. Είναι εξοπλισμένοι συνήθως από μια μονάδα επεξεργασίας με περιορισμένη υπολογιστική δύναμη και περιορισμένη μνήμη, με ένα πομποδέκτη ή άλλη συσκευή ασύρματης επικοινωνίας, και μια πηγή ενέργειας, συνήθως μια μπαταρία. Το μέγεθος ενός κόμβου αισθητήρων μπορεί να ποικίλει από δεκάδες εκατοστά, ως συσκευές με μέγεθος κόκκου σκόνης[2]. Το κόστος των κόμβων αισθητήρων είναι ομοίως μεταβλητό, κυμαινόμενο από εκατοντάδες δολάρια ως μερικά σεντς, ανάλογα με το μέγεθος του δικτύου αισθητήρων και της πολυπλοκότητας που απαιτείται από τους μεμονωμένους κόμβους αισθητήρων. Περιορισμοί μεγέθους και κόστους στους κόμβους αισθητήρων οδηγούν σε αντίστοιχους περιορισμούς σε πόρους όπως η ενέργεια, η μνήμη, η υπολογιστική ταχύτητα και το εύρος ζώνης[3].

Παρακάτω φαίνεται η δομή ενός ασύρματου αισθητήρα:



Σχήμα 2: Δομή ασύρματων αισθητήρων

(εικόνα από http://www.ee.unimelb.edu.au/sen_net/multimedia/research_prog/intsens.gif)

Οι αισθητήρες που απαρτίζουν τα δίκτυα αυτά μπορούν να μετρήσουν:

- Απόσταση, κατεύθυνση, ταχύτητα
- Υγρασία, σύσταση εδάφους
- Θερμοκρασία, χημικά
- Ηλιακή ακτινοβολία, κίνηση, δονήσεις
- Σεισμικά και ακουστικά δεδομένα.

3.2 Το λειτουργικό σύστημα TinyOS

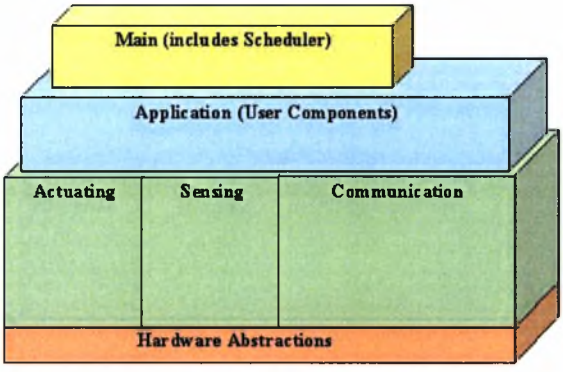
Τα λειτουργικά συστήματα για τους ασύρματους κόμβους δικτύων αισθητήρων είναι λιγότερο σύνθετα από τα γενικής χρήσης λειτουργικά συστήματα και λόγω των πρόσθετων απαιτήσεων των εφαρμογών δικτύων αισθητήρων και λόγω των περιορισμών των πόρων στο hardware των πλατφόρμων των δικτύων αισθητήρων. Παραδείγματος χάριν, οι εφαρμογές δικτύων αισθητήρων δεν είναι αλληλοδραστικές με τον ίδιο τρόπο όπως οι εφαρμογές για τα PC. Για αυτόν τον λόγο, το λειτουργικό σύστημα δεν χρειάζεται να υποστηρίζει interfaces για την επικοινωνία με τον χρήστη. Επιπλέον, οι περιορισμοί του hardware όσον αφορά την μνήμη και την χαρτογράφηση της μνήμης καθιστά τους μηχανισμούς όπως η εικονική μνήμη είτε άχρηστους είτε αδύνατους να εφαρμοστούν.

Το TinyOS[6] είναι ίσως το πρώτο λειτουργικό σύστημα που σχεδιάστηκε συγκεκριμένα για τα ασύρματα δίκτυα αισθητήρων, από το οποίο προέρχεται σε μεγάλο βαθμό η επιτυχία και η δημοτικότητα που γνωρίζουν τα ασύρματα δίκτυα αισθητήρων τα τελευταία χρόνια.

Οι προκλήσεις που είχε να αντιμετωπίσει και να λύσει το TinyOS ήταν:

- Η χαμηλή κατανάλωση ενέργειας.
- Ιδιαίτερα υψηλές απαιτήσεις για συγχρονισμό:
  1. Ροή πληροφορίας από πολλές πηγές (αισθητήρες, πομποδέκτης)
  2. Μικρή μνήμη, που σημαίνει ότι δεν μπορεί να γίνει buffering, άρα πρέπει να επεξεργαστούμε γρήγορα τα μηνύματα που δεχόμαστε, αλλιώς μπορεί να τα χάσουμε
- Μικρό μέγεθος συνολικά του συστήματος
- Η σχεδίαση να είναι modular για να μπορούμε να φτιάξουμε γρήγορα και εύκολα εφαρμογές

Μια απλοποιημένη εκδοχή της αρχιτεκτονικής του TinyOS φαίνεται στο παρακάτω σχήμα:



Σχήμα 3: Η αρχιτεκτονική του TinyOS

(εικόνα από http://www.ceid.upatras.gr/courses/katanemhmena/wiki)

Όταν λέμε ότι ένας κόμβος τρέχει TinyOS, εννοούμε ότι έχει εγκατεστημένο στη flash μνήμη του ένα binary εκτελέσιμο image με τις βιβλιοθήκες του TinyOS που χρειαζόμαστε, συνδεδεμένες με την εφαρμογή που θέλουμε να εκτελέσουμε. Το image αυτό από εδώ και πέρα θα το αναφέρουμε ως εφαρμογή TOS (TinyOS application). Το TinyOS από μόνο του δεν εκτελεί κάποια ιδιαίτερη λειτουργία και ούτε έχει κάποιο user interface (όπως π.χ. το shell στο Unix), οπότε δεν έχει κανένα νόημα να το εγκαταστήσουμε μόνο του σε έναν κόμβο.

Αντίθετα από τα περισσότερα άλλα λειτουργικά συστήματα, το TinyOS είναι βασισμένο σε ένα event-driven πρότυπο προγραμματισμού αντί για multithreading. Τα προγράμματα TinyOS αποτελούνται από event-handler και tasks. Όταν συμβαίνει ένα εξωτερικό γεγονός, όπως ένα εισερχόμενο πακέτο στοιχείων ή μια ανάγνωση αισθητήρων, το TinyOS καλεί τον αρμόδιο event-handler για αν χειριστεί το γεγονός. Το σύστημα TinyOS καθώς και τα προγράμματα που γράφονται για TinyOS γράφονται σε μια ειδική γλώσσα προγραμματισμού αποκαλούμενη nesC που είναι μια επέκταση της γλώσσας προγραμματισμού C. Η nesC[8] έχει ως σκοπό να ανιχνεύει τις συνθήκες μετάβασης μεταξύ tasks και event-handlers.

3.3 Η γλώσσα προγραμματισμού nesC

Η γλώσσα προγραμματισμού nesC αναπτύχθηκε στο πανεπιστήμιο της Καλιφόρνια Berkeley σε συνεργασία με την ομάδα που δημιούργησε το TinyOS.

Μερικές από τις αρχές που διέπουν τη σχεδίαση της nesC είναι:

- **Η nesC βασίζεται στη C**: Αυτή η επιλογή έγινε διότι οι μεταγλωττιστές της C παράγουν αποδοτικό κώδικα για όλους του microcontrollers που μπορούν να

χρησιμοποιηθούν ως επεξεργαστές σε συστήματα έξυπνης σκόνης. Επίσης, ένα μεγάλο μέρος των προγραμματιστών σε embedded συστήματα χρησιμοποιεί τη C ως γλώσσα προγραμματισμού.

- **Ολική ανάλυση προγράμματος κατά τη διάρκεια της μεταγλώττισης (compiling)**: Η ξεχωριστή μεταγλώττιση μερών ενός προγράμματος δεν είναι διαθέσιμη ως επιλογή στη nesC . Αυτό γίνεται για να είναι πιο ακριβής ο έλεγχος για λάθη και συγχρονισμό (race conditions) στο πρόγραμμα και για πιο αποδοτικό κώδικα (μικρότερο μέγεθος), το οποίο είναι αρκετά βολικό, δεδομένου του μικρού μεγέθους της διαθέσιμης μνήμης.
- **Στατικός κώδικας**: Η μνήμη κατανέμεται στατικά σε μια εφαρμογή TOS κατά τη διάρκεια της μεταγλώττισης και επίσης το γράφημα διασυνδέσεων μεταξύ των διάφορων component είναι σταθερό και γνωστό. Το μοντέλο των component εξαλείφει την ανάγκη για δυναμική δέσμευση μνήμης και ενθαρρύνει ένα ευέλικτο σχεδιασμό.
- **Η nesC υποστηρίζει και αντικατοπτρίζει τη φιλοσοφία του** TinyOS **.**

Μια εφαρμογή **nesC** αποτελείται από ένα ή περισσότερα **components** που συνδέονται για να διαμορφώσουν ένα εκτελέσιμο. Ένα **component** παρέχει και χρησιμοποιεί τα **interfaces**. Αυτά τα **interfaces** είναι το μόνο σημείο της πρόσβασης στο **component** και είναι αμφίδρομες. Ένα **interface** δηλώνει ένα σύνολο λειτουργιών, τις οποίες ονομάζουμε **commands**, και τις οποίες πρέπει να υλοποιήσει ο **interface provider**. Καθώς και ένα σύνολο λειτουργιών, τις οποίες ονομάζουμε **events**, και τις οποίες πρέπει να υλοποιήσει ο χρήστης. Για να μπορέσει ένα **component** να καλέσει τα **commands** ενός **interface**, θα πρέπει πρώτα να υλοποιήσει τα **enents** αυτού του **interface**. Ένα **interface** μπορεί να χρησιμοποιεί ή να παρέχει πολλαπλά αντίγραφα του ίδιου **interface**.

Υπάρχουν δύο είδη component στη **nesC**: τα **modules** και τα **configurations.** Τα **modules** παρέχουν τον κώδικα εφαρμογής, υλοποιώντας ένα ή περισσότερα **interfaces**. Τα **configurations** χρησιμοποιούνται για να συγκεντρώσουν άλλα **components** μαζί, ενώνοντας τα **interface** διαφορετικών **component** μεταξύ τους. Αυτό ονομάζεται **wiring**. Κάθε **nesC** εφαρμογή περιγράφεται από ένα **top-level configuration** το οποίο ενώνει μεταξύ τους τα εσωτερικά **components**.

4 ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

4.1 Η χρήση της UML.

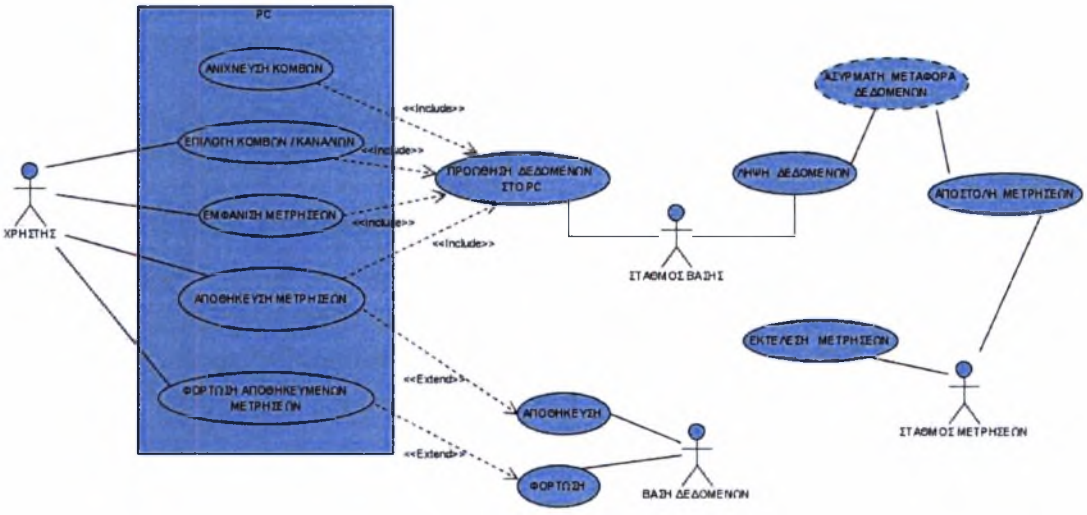
Η UML (Unified Modeling Language)[9] είναι μια τυποποιημένη γλώσσα για τη διευκρίνιση, την απεικόνιση, την κατασκευή και την τεκμηρίωση των δεδομένων των συστημάτων λογισμικού, καθώς επίσης και για τη μοντελοποίηση επιχειρήσεων, αλλά και άλλων συστημάτων. Η UML αντιπροσωπεύει μια συλλογή των καλύτερων πρακτικών εφαρμοσμένης μηχανικής που έχουν αποδειχθεί επιτυχείς στη διαμόρφωση μεγάλων και σύνθετων συστημάτων. Η χρησιμοποίηση της UML βοηθά τις ομάδες ανάπτυξης στην επικοινωνία, συμβάλει στην μελέτη όλων των πιθανών σχεδίων, και επικυρώνει την επιλεγμένη αρχιτεκτονική σύμφωνα με την οποία θα αναπτυχθεί το λογισμικό.

4.2 Διαδικασίας ανάπτυξης της εφαρμογής.

Το ζητούμενο της εφαρμογής μας είναι η συλλογή των μετρήσεων από ένα ασύρματο δίκτυο αισθητήρων, η επιλεκτική απεικόνιση και η αποθήκευση αυτών σε πραγματικό χρόνο, σύμφωνα με τις επιλογές του χρήστη. Όπως είναι φανερό από τα παραπάνω, για την υλοποίηση του συστήματός μας χρειάζεται να γίνει χρήση διάφορων επιμέρους εργαλείων, όπως είναι το TinyOS, η Βάση Δεδομένων, η Java, αλλά και διάφορων συστατικών, όπως είναι ο SerialForwarder, οι εικόνες(προγράμματα) των αισθητήρων, βιβλιοθήκες γραφικών, κλπ. Για την αποδοτική χρήση, και την επίτευξη αποτελεσματικής συνεργασίας αυτών ήταν αναγκαίο να γίνει ένα είδος τυποποίησης αυτών. Η τυποποίηση αυτή έγινε με χρήση της γλώσσας UML. Στην συνέχεια της ενότητας παρουσιάζονται μερικά από τα διαγράμματα που χρησιμοποιήθηκαν, και τα οποία είναι χρήσιμα στην καλύτερη κατανόηση της δομής, αλλά και της λειτουργίας του συστήματός μας.

4.2.1 Διάγραμμα περιπτώσεων χρήσης - Use Case Diagram

Αρχικά θα παρουσιάσουμε το διάγραμμα περιπτώσεων χρήσης του συστήματος (use case diagram). Το συγκεκριμένο διάγραμμα μας δίνει την δυνατότητα να περιγράψουμε την αλληλεπίδραση του χρήστη, αλλά και άλλων εξωτερικών συστημάτων με την εφαρμογή μας.



Σχήμα 6: Διάγραμμα Χρήσης

Όπως βλέπουμε και στο διάγραμμα ο χρήστης του συστήματος έχει την δυνατότητα να εκτελέσει 4 λειτουργίες:

1. Την επιλογή των κόμβων και των καναλιών των οποίων τις μετρήσεις θα λάβει,
2. την εμφάνιση των συγκεκριμένων μετρήσεων στον χρήστη,
3. την αποθήκευση των μετρήσεων αυτών σε μια βάση δεδομένων, και
4. την μελλοντική χρήση αυτών μέσω της λειτουργίας της φόρτωσης των μετρήσεων από την βάση δεδομένων.

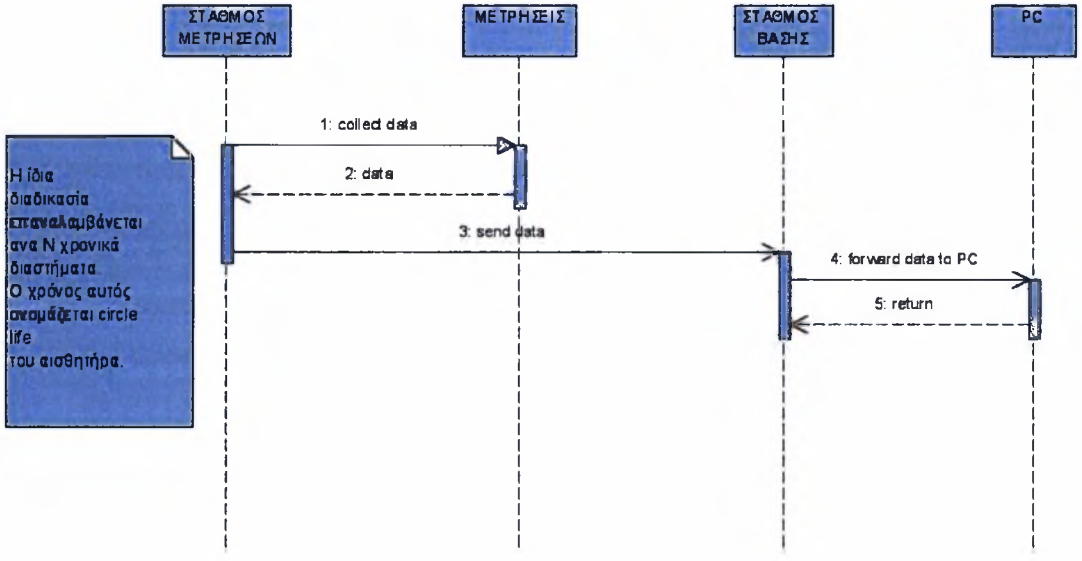
Οι μετρήσεις αυτές για να φτάσουν στον χρήστη θα πρέπει να ακολουθήσουν μια σειρά από διαδικασίες. Οι σημαντικότερες από αυτές είναι:

1. η εκτέλεση και αποστολή αυτών από τους σταθμούς μετρήσεων στον σταθμό βάσης,
2. η λήψη των μετρήσεων από τον σταθμό βάσης και
3. η προώθηση αυτών στο PC, από όπου θα έχουν την δυνατότητα στη συνέχεια να τις επεξεργαστούν οι λειτουργίες που βρίσκονται εκεί.

4.2.2 Διαγράμματα Ακολουθίας - Sequence Diagrams

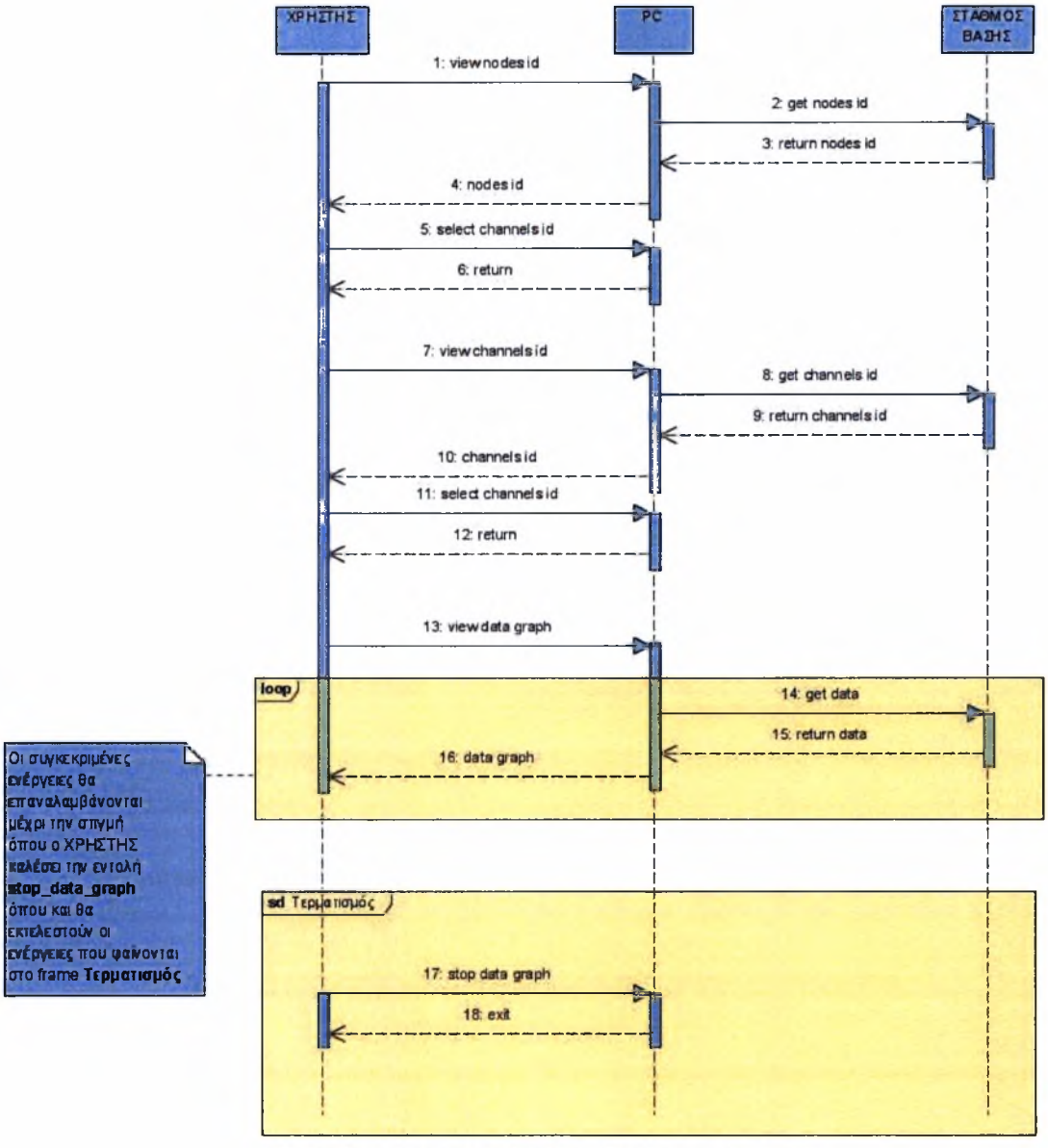
Για την καλύτερη κατανόηση των παραπάνω λειτουργιών παρουσιάζουμε τα παρακάτω διαγράμματα ακολουθίας (sequence diagram) μέσω των οποίων θα μπορέσουμε να περιγράψουμε τις διάφορες λειτουργίες, αλλά και τα μηνύματα που ανταλλάσσουν αυτές.

Το σχήμα 7 περιέχει το διάγραμμα ακολουθίας το οποίο περιγράφει την διαδικασία , αλλά και τα μηνύματα που χρειάζεται να σταλούν ώστε να γίνουν οι μετρήσεις από τους σταθμούς μετρήσεων, να αποσταλούν στον σταθμού βάσης, και από εκεί να προωθηθούν στο PC.



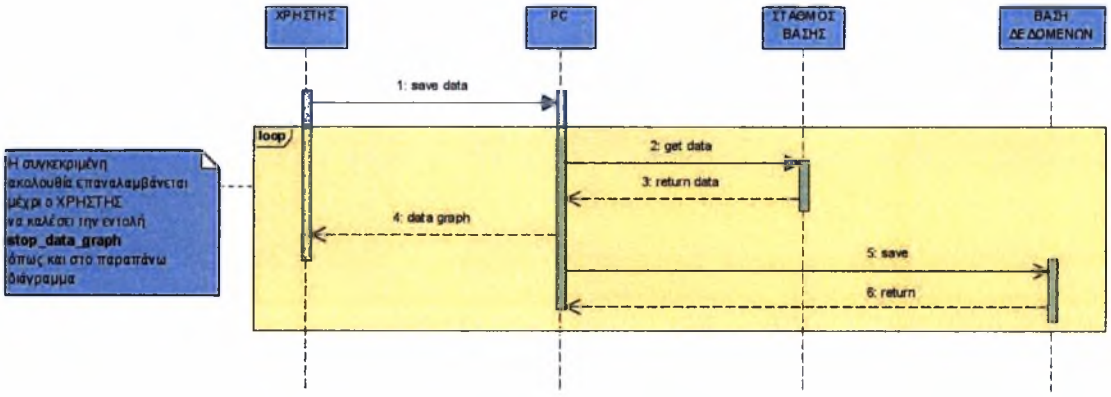
Σχήμα 7: Διάγραμμα ακολουθίας – Λήψη Δεδομένων

Στο διάγραμμα που ακολουθεί περιγράφονται οι λειτουργίες και τα μηνύματα που χρειάζονται ώστε να γίνει: η επιλογή από τον χρήστη των μετρήσεων που επιθυμεί, η παραλαβή αυτών από τον σταθμό βάσης, και τέλος η εμφάνιση αυτών στον χρήστη.



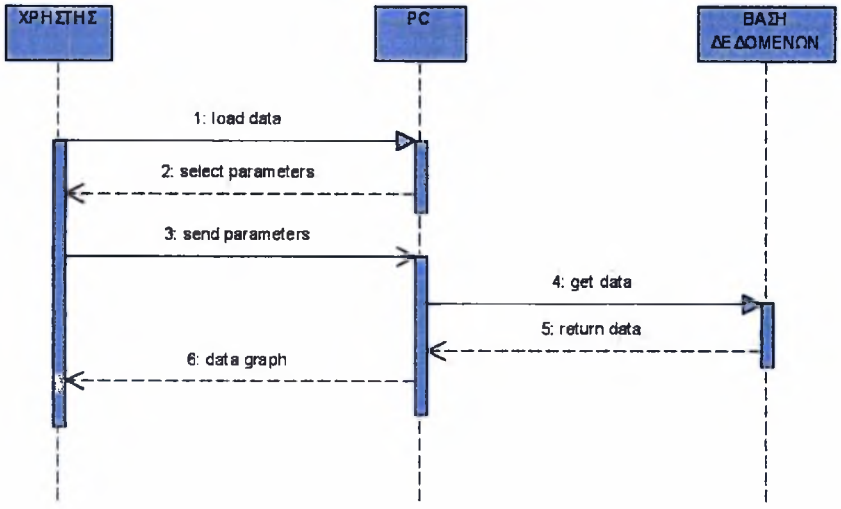
Σχήμα 8: Διάγραμμα ακολουθίας – Εμφάνιση Δεδομένων

Κατά την διάρκεια εμφάνισης των μετρήσεων σύμφωνα με την παραπάνω διαδικασία, ο χρήστης έχει την δυνατότητα να επιλέξει και την ταυτόχρονη αποθήκευση των μετρήσεων. Στο παρακάτω διάγραμμα ακολουθίας παρουσιάζονται τα μηνύματα που χρειάζεται να σταλούν ανάμεσα στον Χρήστη, στο PC, στον σταθμό βάσης και στη Βάση Δεδομένων, ώστε να επιτευχθεί η αποθήκευση.



Σχήμα 9: Διάγραμμα Ακολουθίας – Αποθήκευση Δεδομένων

Για την μελλοντική χρήση των αποθηκευμένων μετρήσεων, υπάρχει η λειτουργία της φόρτωσης και εμφάνισης των δεδομένων στο χρήστη, από το σημείο όπου βρίσκονται αποθηκευμένες στην Βάση Δεδομένων. Η λειτουργία αυτή περιγράφεται στο παρακάτω διάγραμμα ακολουθίας, όπου μπορούμε να δούμε με ακρίβεια τα μηνύματα που ανταλλάσσονται για το σκοπό αυτό ανάμεσα στον Χρήστη, στο PC, και στη Βάση Δεδομένων.

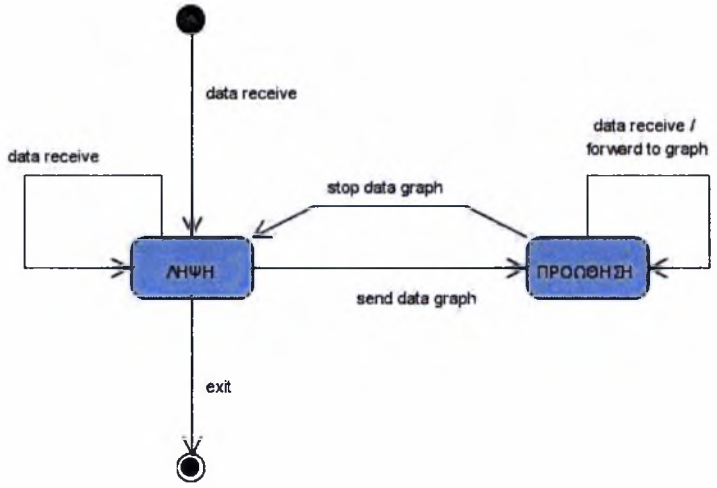


Σχήμα 10: Διάγραμμα Ακολουθίας - Φόρτωση Δεδομένων

4.2.3 Διαγράμματα Καταστάσεων - State Diagrams

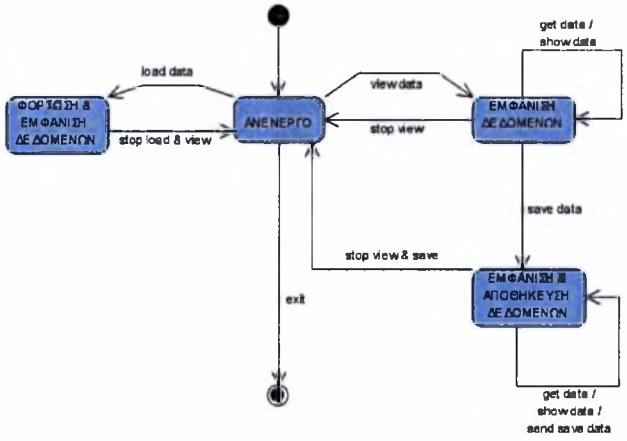
Για την καλύτερη κατανόηση των λειτουργιών που εκτελεί ο σταθμός βάσης και το PC θα χρησιμοποιήσουμε τα παρακάτω διαγράμματα περίπτωσης, από όπου μπορούμε να δείξουμε τις καταστάσεις τις οποίες παίρνει κάθε ένα από τα παραπάνω δύο συστατικά έπειτα από την λήψη ή παραλαβή ενός μηνύματος.

Για τον σταθμό βάσης βλέπουμε ότι υπάρχουν δύο καταστάσεις: η κατάσταση Λήψης και η κατάσταση Προώθησης. Στην κατάσταση Λήψης λαμβάνει τις μετρήσεις από τους σταθμούς μετρήσεων. Επιπλέον στην κατάσταση Προώθησης ο σταθμός βάσης, εκτός της λήψης των μετρήσεων, αναλαμβάνει και την προώθηση αυτών στο PC.



Σχήμα 11: Διάγραμμα Κατάστασης - Σταθμός Βάσης

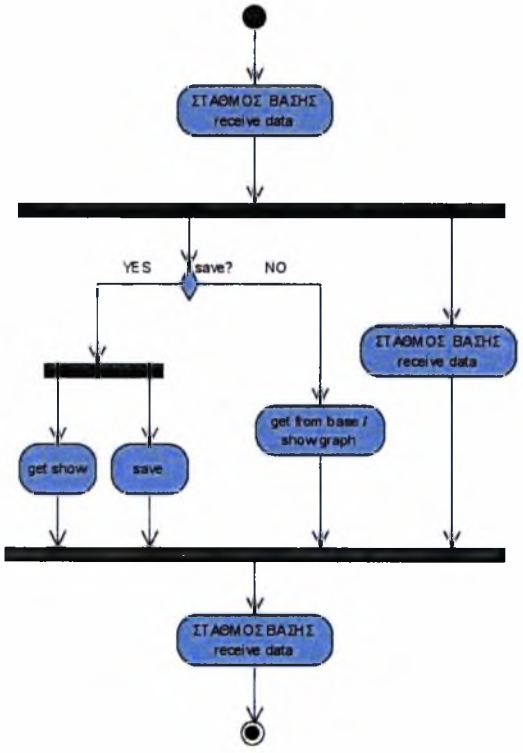
Στο διάγραμμα κατάστασης που περιγράφει τις λειτουργίες που αναλαμβάνει το PC, υπάρχουν τέσσερις καταστάσεις, η κατάσταση «Ανενεργό», η κατάσταση «Εμφάνιση Δεδομένων», η κατάσταση «Εμφάνιση & Αποθήκευση Δεδομένων» και η κατάσταση «Φόρτωση & Εμφάνιση Δεδομένων». Στην κατάσταση «Εμφάνιση Δεδομένων» οι μετρήσεις φτάνουν στο PC και εμφανίζονται στον χρήστη. Επιπλέον στην κατάσταση «Εμφάνιση & Αποθήκευση Δεδομένων» γίνεται και αποθήκευση των μετρήσεων αυτών. Τέλος στην κατάσταση «Φόρτωση & Εμφάνιση Δεδομένων» τα αποθηκευμένα δεδομένα φορτώνονται και εμφανίζονται στον χρήστη.



Σχήμα 12: Διάγραμμα Κατάστασης – PC

4.2.4 Διάγραμμα Δραστηριότητας – Activity Diagram

Όπως είναι φανερό από την μέχρι τώρα περιγραφή του συστήματος, πολλές από τις λειτουργίες εκτελούνται ταυτόχρονα και ανεξάρτητα από άλλες που ανήκουν στο ίδιο σύστημα. Με την χρήση του παρακάτω Διαγράμματος Δραστηριότητας, έχουμε την δυνατότητα να δούμε τις λειτουργίες αυτές.



Σχήμα 13: Διάγραμμα Δραστηριότητας

5 Τα συστατικά της εφαρμογής.

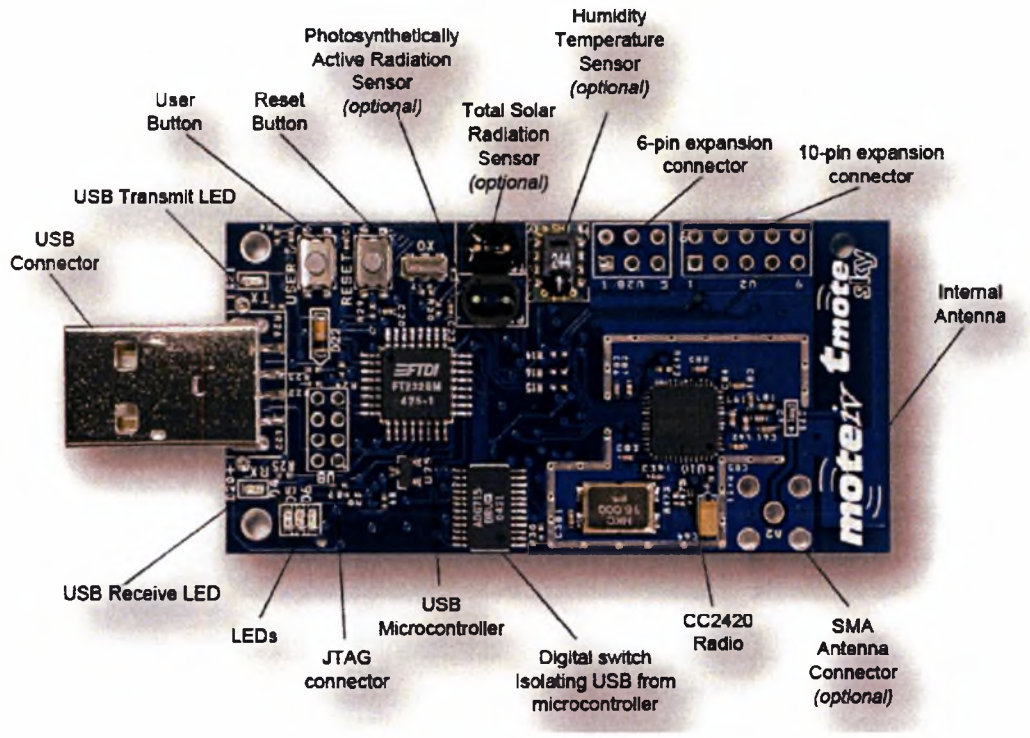
5.1 Οι αισθητήρες « Tmote Sky ».

5.1.1 Τι είναι τα « Tmote Sky ».

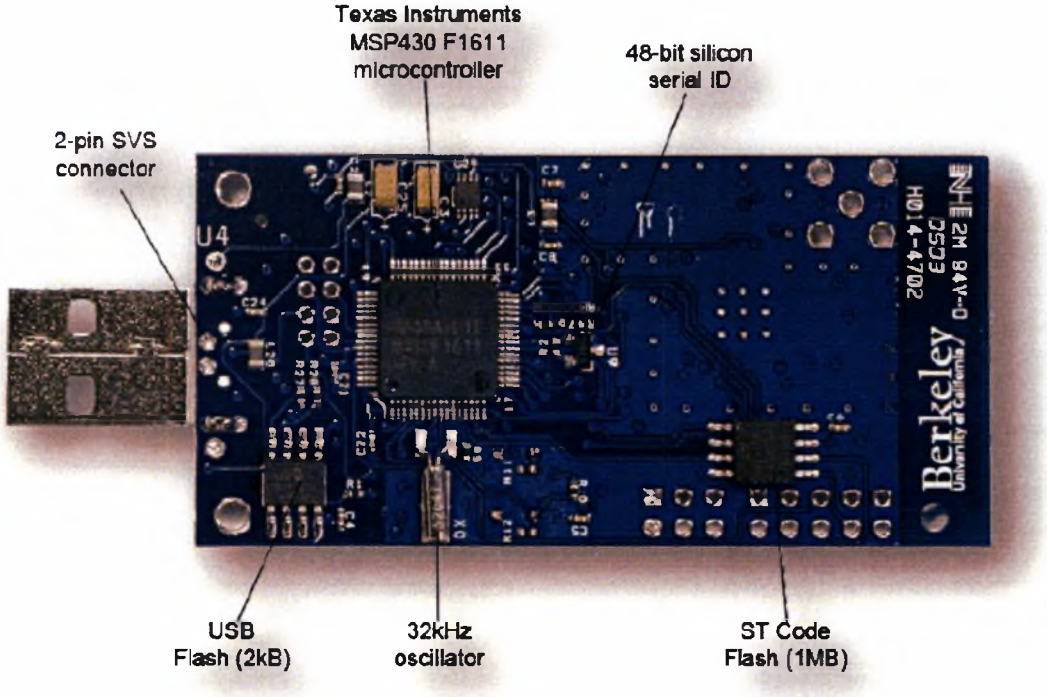
Οι αισθητήρες που θα χρησιμοποιήσουμε στην εφαρμογή είναι οι Tmote Sky[7]. Οι Tmote Sky είναι από τους πιο διαδεδομένους ασύρματους αισθητήρες σήμερα. Το Tmote Sky είναι ένας ασύρματος αισθητήρας χαμηλής ενέργειας για χρήση σε δίκτυα αισθητήρων και σε εφαρμογές ελέγχου. Είναι εφοδιασμένο με αισθητήρες μέτρησης θερμοκρασίας, υγρασίας, καθώς και ηλιακής ακτινοβολίας *(Photosynthetically Active Radiation*, και *Total Solar Radiation*). Ακόμα φέρει διάφορα βιομηχανικά πρότυπα (industry standards), όπως είναι η USB και το IEEE 802.15.4[15], έτσι ώστε να έχει την δυνατότητα να επικοινωνεί άμεσα με άλλες συσκευές. Εκμεταλλευόμενο τα βιομηχανικά πρότυπα και τους αισθητήρες που αναφέραμε παραπάνω, καθώς και την εύκολη επικοινωνία του με τις διάφορες περιφερειακές συσκευές, αποκτάει την δυνατότητα αποδοτικής συμμετοχής του σε ένα ευρύ φάσμα εφαρμογών ασύρματων αισθητήρων. Το Tmote Sky αποτελεί την συνέχεια του αισθητήρα Telos, ενός πετυχημένου σχεδίου της εταιρείας Moteiv. Διαθέτει 1MB μνήμη, 10KB RAM, 48 KB Memory Flash και 250 kbps Radio bandwidth. Υποστηρίζει το TinyOS ως λειτουργικό σύστημα.

Τα βασικά χαρακτηριστικά τους είναι:

- 250kbps 2.4GHz IEEE 802.15.4 Chipcon Wireless Transceiver.
- Σύνδεση με άλλες IEEE 802.15.4 συσκευές.
- 8MHz Texas Instruments MSP430 microcontroller (10k RAM, 48k Flash)
- Ενσωματωμένο ADC, DAC, Supply Voltage Supervisor, and DMA Controller
- Ενσωματωμένη onboard κεραία με 50m εμβέλεια σε εσωτερικούς σώρους / 125m εμβέλεια σε εξωτερικούς.
- Ενσωματωμένα sensors για μέτρηση υγρασίας, θερμοκρασίας και φωτός.
- Εξαιρετικά χαμηλή κατανάλωση ρεύματος.
- Γρήγορο ξύπνημα από ύπνο (<6κs).
- Hardware link-layer κρυπτογράφηση και πιστοποίηση.
- Προγραμματισμός και συλλογή δεδομένων μέσω USB καλωδίου.
- 16-pin υποδοχή επέκτασης και δυνατότητα επιλογής SMA σύνδεσης κεραίας.
- TinyOS υποστήριξη: mesh networking and communication implementation.
- Συμμορφώνεται με το FCC Part 15 και τους κανονισμούς Βιομηχανίας του Καναδά.



Σχήμα 4: Μπροστινό μέρος των Tmote Sky



Σχήμα 5: Πίσω μέρος των Tmote Sky

5.1.2 Το λειτουργικό σύστημα των Tmote Sky (Boomerang)

Το Boomerang[7] είναι το λειτουργικό σύστημα της εταιρίας Moteiv. Αποτελεί έκδοση του λειτουργικού συστήματος TinyOS, κατάλληλα τροποποιημένο για τις απαιτήσεις και τις ανάγκες των συσκευών της Moteiv, όπως είναι τα: Tmote Sky και Tmote Invent. Το Boomerang περιέχει αρκετές βιβλιοθήκες ικανές να οικοδομήσουν σταθερές, χαμηλής κατανάλωσης εφαρμογές ασύρματων αισθητήρων. Συνδυάζει τις καινοτόμες τεχνολογίες του open-source λογισμικού με ένα σταθερό και αξιόπιστο σύστημα, καθιστώντας το ικανό να δημιουργήσει αποτελεσματικές ασύρματες λύσεις.

Τα βασικά χαρακτηριστικά τους είναι:

- Αξιόπιστη, χαμηλής κατανάλωσης, δικτύωση πλέγματος.
- Εκτενείς βιβλιοθήκες αισθητήρων.
- Πλήρεις παραδείγματα εφαρμογών και server-side εργαλεία.
- Πλήρη συμβατότητα με υπάρχουσες εφαρμογές του TinyOS 1.x.
- Ενσωματώνει τα κύρια χαρακτηριστικά του TinyOS 2.x
- Ικανότητα συνύπαρξης με οποιαδήποτε υπάρχουσα εγκατάσταση TinyOS.
- Οι open-source βιβλιοθήκες επιτρέπουν την τροποποίηση κατ' επιλογή του χρήστη.
- Ιδανικό για χρήση στους Moteiv Tmote Sky και Tmote Invent αισθητήρες.

5.1.3 Εγκατάσταση και προγραμματισμός των Tmote Sky.

Για να προγραμματίσουμε τα Tmote Sky αρχικά θα πρέπει να γίνει εγκατάσταση του Boomerang στο σύστημα μας. Το αρχείο εγκατάστασης διατίθεται δωρεάν μέσω του Internet από την διεύθυνση URL: http://www.moteiv.com. Το συγκεκριμένο εκτελέσιμο αρχείο αναλαμβάνει να κάνει την εγκατάσταση όλων των αναγκαίων εργαλείων στο σύστημά μας ώστε να λειτουργήσει αποτελεσματικά το Boomerang.

Τα σημαντικότερα από αυτά τα εργαλεία είναι:

- **Cygwin**: είναι ένα περιβάλλον παρεμφερές με το Linux το οποίο όμως λειτουργεί στα Windows. Παρέχει το βασικό περιβάλλον ανάπτυξης για το TinyOS και τα Tmote.
- **Java**: χρησιμοποιείται για την υποστήριξη των εργαλείων τα οποία αναλαμβάνουν την επικοινωνία ανάμεσα σε ένα mote και στο PC.
- **JavaComm:** είναι ένα επιπλέον πακέτο βιβλιοθηκών της Java το οποίο παρέχει πρόσβαση στις serial ports. Είναι απαραίτητο για εργαλεία όπως είναι το SerialForwarder.
- **FTDI VirtualComm USB Drivers**: παρέχει τους drivers που είναι απαραίτητοι για την επικοινωνία των USB-to-RS232 chips των Windows, σαν μια κλασική COM θύρα.
- **TinyOS**: παρέχει όλες τις βιβλιοθήκες και τα εργαλεία του TinyOS, τα οποία αποτελούν την βάση πάνω στην οποία θα υλοποιηθούν οι εφαρμογές.

- **nesC**: παρέχει τον compiler της γλώσσας προγραμματισμού η οποία είναι απαραίτητη για την λειτουργία του TinyOS, αλλά και για την υλοποίηση των εφαρμογών.
- **Moteiv scripts**: παρέχει scripts και διάφορες εφαρμογές οι οποίες βοηθούν τα παραπάνω συστήματα να συνεργαστούν αποδοτικότερα με τις Tmote συσκευές. Για παράδειγμα το motelist είναι μια εντολή κονσόλας η οποία εμφανίζει όλες τις σειριακές θύρες οι οποίες έχουν επικοινωνία με μια Tmote συσκευή.

Αφού ολοκληρωθεί η διαδικασία της εγκατάστασης των απαραίτητων εργαλείων και βιβλιοθηκών, είμαστε έτοιμοι να προχωρήσουμε στον προγραμματισμό των Tmote Sky.

Το Cygwin[11] είναι το εργαλείο που χρησιμοποιείτε για να γίνει το compile και ο επαναπρογραμματισμός των Tmote. Αρχικά ανοίγουμε το κεντρικό shell του Cygwin. Αυτό ανοίγει στον φάκελο */opt/moteiv*. Πηγαίνουμε στον φάκελο στον οποίο υπάρχει η εφαρμογή η οποία θέλουμε να κάνουμε compile. Στον φάκελο αυτό θα υπάρχουν τα απαραίτητα nesC αρχεία της εφαρμογής. Τα αρχεία αυτά τα κάνουμε compile εκτελώντας την εντολή ***make tmote.*** Μέσω αυτής της εντολής δίνουμε εντολή στο μεταφραστή, η εκτελέσιμη εφαρμογή που θα δημιουργηθεί να είναι συμβατή με τις συσκευές tmote.

Τα motes θα τα συνδέσουμε μέσω της USB θύρας που διαθέτουν στο PC. Μέσω αυτής της σύνδεσης θα είναι δυνατή η μεταφορά των εκτελέσιμων αρχείων στα motes. Για να δούμε ποια tmote sky είναι συνδεδεμένα και σε ποιο port μπορούμε να εκτελέσουμε την εξής εντολή:

```
$ motelist
Reference    CommPort    Description
M4A3S3UF     COM8        tmote sky
```

Για να εγκαταστήσουμε μια εφαρμογή όταν μόνο ένα tmote είναι συνδεδεμένο εκτελούμε την επόμενη εντολή:

make tmote reinstall, 1

«reinstall, 1» σημαίνει προγραμμάτισε το mote με την ήδη compiled binary image και θέσε την διεύθυνση του δικτύου του mote σε 1.

Για να εγκαταστήσουμε μια εφαρμογή σε ένα tmote αν περισσότερα από ένα είναι συνδεδεμένα πρέπει να καθορίσουμε τη σειριακή port όπου πρέπει να εγκατασταθεί, χρησιμοποιώντας την εντολή:

make tmote reinstall, 1 bsl, 3

"bsl, 3" σημαίνει ότι πρέπει να στείλει το πρόγραμμα χρησιμοποιώντας το Boot Strap Loader στην COM4, αφού το πρόγραμμα bsl χρησιμοποιεί την δεικτοδότηση των Linux για τις COM θύρες, οπότε το COM4 χαρακτηρίζεται ως "3" για το bsl.

Για κάθε πλήρη εφαρμογή χρειάζονται 2 διαφορετικά εκτελέσιμα αρχεία. Το ένα παίρνει όλες τις μετρήσεις από τον αισθητήρα στον οποίο είναι εγκαταστημένη, και το δεύτερο φορτώνεται στον αισθητήρα εκείνον ο οποίος θα παίξει τον ρόλο της βάσης, δηλαδή θα είναι συνδεδεμένος με τον υπολογιστή και θα συλλέγει τις μετρήσεις από τους υπόλοιπους αισθητήρες στους οποίους έχουμε εγκαταστήσει το πρώτο.

Τέλος για να γίνει εκτέλεση της εφαρμογής που μόλις εγκαταστήσαμε εκτελούμε την εντολή:

MOTECOM=serial@COM4:tmote java com.moteiv.trawler.Trawler.

Η παραπάνω εντολή αναλαμβάνει να εκτελέσει την εφαρμογή ***com.moteiv.trawler.Trawler***. Η μεταβλητή "***MOTECOM=serial@COM4***", λέει στα Java tools να επικοινωνήσουν χρησιμοποιώντας το σειριακό πρωτόκολλο πάνω από την θύρα COM4 όπου είναι προσαρτημένο ένα mote.

5.2 ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

5.2.1 Η χρήση της MySQL

Βάση δεδομένων είναι ένα σύνολο εγγραφών ή αρχείων τα οποία έχουν ενταχθεί σε μια δομή ώστε να μπορούν να χρησιμοποιηθούν από πολλές εφαρμογές και πολλούς χρήστες. Τα δεδομένα τα οποία ανακτώνται έπειτα από την εκτέλεση ενός ερωτήματος στην βάση δεδομένων είναι πληροφορίες οι οποίες μπορούν να χρησιμοποιηθούν για την λήψη αποφάσεων. Η εφαρμογή η οποία χρησιμοποιείτε για την διαχείριση και την εκτέλεση ερωτημάτων στην βάση δεδομένων ονομάζεται, σύστημα διαχείρισης βάσης δεδομένων (database management system – DBMS).

Το σύστημα βάσης δεδομένων, το οποίο χρησιμοποιήσαμε στην υλοποίηση της εφαρμογής μας είναι η MySQL[12]. Είναι ένα πολυνηματικό, σύστημα βάσης δεδομένων (DBMS) με την δυνατότητα εξυπηρέτησης πολλών χρηστών ταυτόχρονα. Εξαιτίας της σταθερά υψηλής απόδοσης, υψηλής αξιοπιστίας και εύκολης χρήσης του, αποτελεί την δημοφιλέστερη «ανοικτού κώδικα» βάση δεδομένών στον κόσμο. Έχει την δυνατότητα να χρησιμοποιηθεί σε περισσότερες από 20 πλατφόρμες λογισμικού, όπως: Linux, Windows, OS/X, HP-UX, AIX, Netware, δίνοντας έτσι ένα είδος ευχρηστίας στον χρήστη ο οποίος χρησιμοποιεί τις συγκεκριμένες πλατφόρμες λογισμικού.

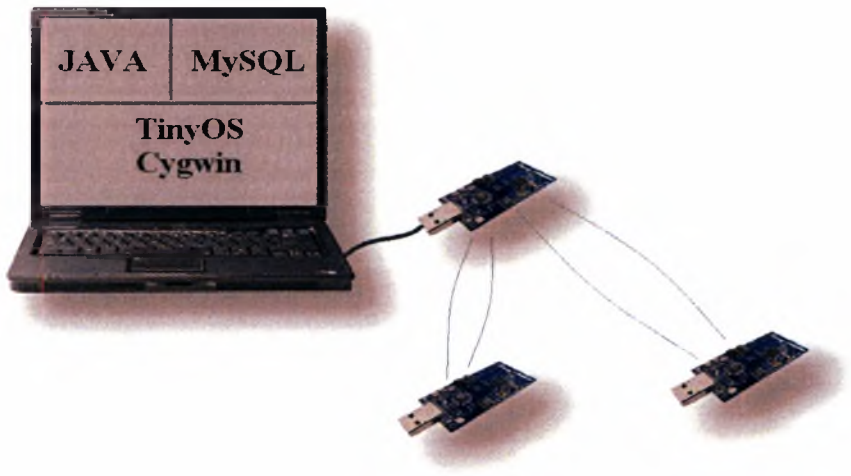
5.2.2 Η χρήση του JDBC

Το Java Database Connectivity (JDBC) API[13] αποτελεί ένα πρότυπο για την επίτευξη σύνδεσης ανάμεσα στην γλώσσα προγραμματισμού JAVA και σε ένα μεγάλο πλήθος βάσεων δεδομένων. Είναι δηλαδή το API το οποίο ορίζει τον τρόπο με τον οποίο ο χρήστης θα έχει πρόσβαση σε μια βάση δεδομένων μέσω της JAVA εφαρμογής του. Παρέχει κατάλληλες JAVA μεθόδους για την εκτέλεση ερωτημάτων και την ανανέωση των εγγραφών που υπάρχουν αποθηκευμένες σε μια βάση δεδομένων.

6 Αρχιτεκτονική της εφαρμογής

6.1 Παρουσίαση της αρχιτεκτονικής

Η αρχιτεκτονική της εφαρμογής που υλοποιήσαμε φαίνεται παρακάτω:



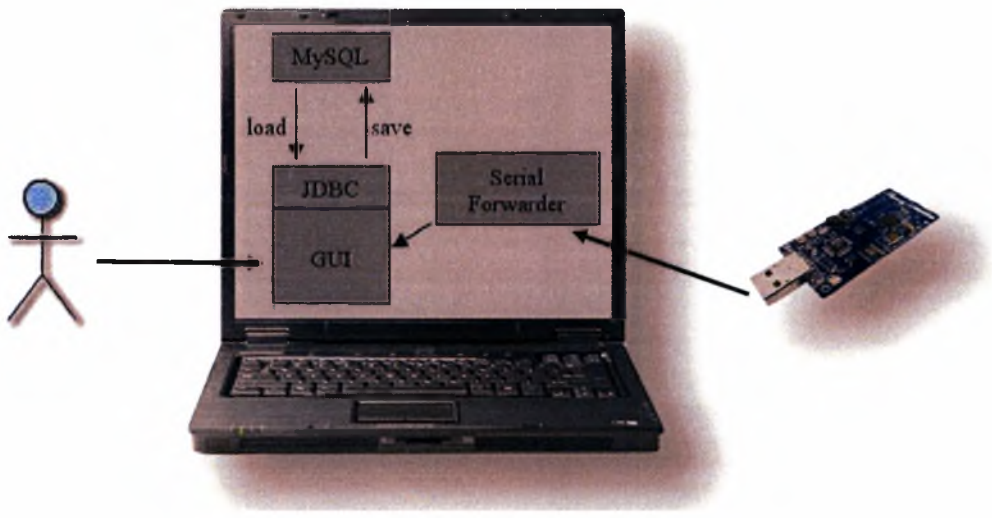
Σχήμα 14: Αρχιτεκτονική της εφαρμογής

Έχουμε έναν υπολογιστή ο οποίος αναλαμβάνει να συλλέξει τις μετρήσεις από όλους τους αισθητήρες. Για να το επιτύχουμε αυτό συνδέουμε έναν κόμβο αισθητήρα μέσω USB στον υπολογιστή και του φορτώνουμε την εφαρμογή TOSBase. Ο κόμβος αυτός θα παίξει τον ρόλο της βάσης. Δηλαδή θα είναι συνδεδεμένος με τον υπολογιστή και θα συλλέγει τις μετρήσεις από τους υπόλοιπους αισθητήρες, αφού η εφαρμογή TOSBase συμπεριφέρεται σαν μια απλή "γέφυρα" ανάμεσα στην σειριακή θύρα και στις ασύρματες συνδέσεις των αισθητήρων. Στους υπόλοιπους αισθητήρες έχουμε εγκαταστήσει την εφαρμογή Oscilloscope, μέσω της οποίας αναλαμβάνουν να κάνουν τις μετρήσεις και να τις στείλουν ασύρματα στον σταθμό βάσης ο οποίος όπως είπαμε τρέχει την εφαρμογή TOSBase.

Τα δεδομένα που μετράνε οι αισθητήρες που χρησιμοποιήσαμε είναι θερμοκρασία, υγρασία, ενεργή φωτοσυνθετική ακτινοβολία (PAR), συνολική ηλιακή ακτινοβολία (TSR) καθώς και εσωτερική τάση και θερμοκρασία. Κάθε αισθητήρας που μετράει κάποιο από τα παραπάνω μεγέθη αντιστοιχεί σε ένα κανάλι. Τα κανάλια αυτά αντιστοιχούν με την σειρά τους στους αριθμούς 0 έως 5. Η αντιστοιχία μεταξύ αριθμών και μετρούμενων μεγεθών είναι η ακόλουθη:

0: Υγρασία
1: Θερμοκρασία
2: Συνολική Ηλιακή Ακτινοβολία (TSR)
3: Ενεργή Φωτοσυνθετική Ακτινοβολία (PAR)
4: Εσωτερική Θερμοκρασία
5: Εσωτερική Τάση

Η μεταφορά των μετρήσεων από τον σταθμό βάσης στο PC, όπως φαίνεται και στο παρακάτω σχήμα:



Σχήμα 15: Αρχιτεκτονική της εφαρμογής (2)

γίνεται μέσω μιας JAVA εφαρμογής, του SerialForwarder, η οποία αναλαμβάνει να διαβάσει τα δεδομένα από την σειριακή θύρα και να τα προωθήσει στο GUI. Το GUI είναι ένα JAVA πρόγραμμα το οποίο υλοποιήσαμε με σκοπό να αναλάβει την επικοινωνία με τον χρήστη. Μέσω του GUI ο χρήστης έχει την δυνατότητα να επιλέξει τις ιδιότητες των μετρήσεων που επιθυμεί να εξετάσει, να παρακολουθήσει μέσω γραφικής διεπαφής τις μετρήσεις αυτές, και να αποφασίσει αν θα γίνει αποθήκευση αυτών στην βάση δεδομένων. Στην περίπτωση που ο χρήστης αποφασίσει την αποθήκευσή τους στη βάση δεδομένων ή την φόρτωση αυτών από την βάση δεδομένων στην γραφική διεπαφή, το GUI αναθέτει την επικοινωνία αυτού με την βάση δεδομένων στο JDBC. Το JDBC είναι η εφαρμογή η οποία αναλαμβάνει να εκτελέσει τις ενέργειες που είναι απαραίτητες για την αποθήκευση, ανάκτηση, και μεταφορά των μετρήσεων από και προς τη βάση δεδομένων.

6.2 Δομή της βάσης δεδομένων

Όπως αναφέραμε και παραπάνω για την αποθήκευση των μετρήσεων χρησιμοποιήσαμε μια σχεσιακή βάση δεδομένων, την MySQL. Το "σχήμα" στο οποίο γίνεται η αποθήκευση στη βάση έχει την παρακάτω μορφή:

Σχήμα 16: Δομή της Βάσης Δεδομένων

Όπου στο πεδίο:

- **result_time** αποθηκεύεται η χρονική στιγμή, την οποία γίνεται η αποθήκευση των μετρήσεων της συγκεκριμένης εγγραφής.
- **node_id** αποθηκεύεται το id του κόμβου από τον οποίο προέρχονται οι μετρήσεις της συγκεκριμένης εγγραφής.
- **packet_id** αποθηκεύεται το id του πακέτου με το οποίο στάλθηκαν οι μετρήσεις της εγγραφής.
- **temperature** αποθηκεύεται η μέτρηση της θερμοκρασίας
- **TSR** αποθηκεύεται η μέτρηση της συνολικής ηλιακής ακτινοβολίας.
- **PAR** αποθηκεύεται η μέτρηση της ενεργής φωτοσυνθετικής ακτινοβολίας.
- **humidity** αποθηκεύεται η μέτρηση της υγρασίας.
- **int_voltage** αποθηκεύεται η μέτρηση της εσωτερικής τάσης.
- **int_temper** αποθηκεύεται η μέτρηση της εσωτερικής θερμοκρασίας.

Σαν πρωτεύων κλειδί του "σχήματος" χρησιμοποιούμε τον συνδυασμό των πεδίων **nodeid & packetid**. Οι ιδιότητες των υπόλοιπων πεδίων φαίνονται στο παρακάτω σχήμα.

Field	Type	Null	Key	Default
result_time	timestamp	NO		CURRENT_TIMESTAMP
nodeid	int(11)	NO	PRI	0
packetid	double	NO	PRI	
temp	double	YES		NULL
TSR	double	YES		NULL
PAR	double	YES		NULL
humidity	double	YES		NULL
int_voltage	double	YES		NULL
int_temp	double	YES		NULL

Σχήμα 17: Ιδιότητες των πεδίων της Βάσης Δεδομένων

Παρακάτω φαίνεται ο κώδικας σε SQL ο οποίος δημιουργεί στην βάση το "σχήμα" με τα παραπάνω πεδία.

```
CREATE TABLE `sensor` (
        `result_time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ,
        `nodeid` INT( 11 ) NOT NULL ,
        `packetid` DOUBLE NOT NULL ,
        `temp` DOUBLE NULL ,
        `TSR` DOUBLE NULL ,
        `PAR` DOUBLE NULL ,
        `humidity` DOUBLE NULL ,
        `int_voltage` DOUBLE NULL ,
        `int_temp` DOUBLE NULL ,
        PRIMARY KEY ( `nodeid` , `packetid` )
) ENGINE = MYISAM ;
```

Σχήμα 18: Κώδικας δημιουργίας της Βάσης Δεδομένων

7 Υλοποίηση της Εφαρμογής

7.1 Προετοιμασία των αισθητήρων.

Για την φόρτωση της εφαρμογής TOSBase στον κόμβο βάσης, συνδέουμε μέσω USB τον κόμβο στον υπολογιστή και εκτελώντας την εντολή *motelist* βλέπουμε σε ποιο COM είναι συνδεδεμένο. Στην συγκεκριμένη περίπτωση είναι στη θύρα COM5.

```
$ motelist
SerialNum   PortName          Description
M4AEMAOD    COM5              tmote sky
```

Στη συνέχεια θα οδηγηθούμε μέσω του Cygwin στον φάκελο με την εφαρμογή TOSBase εκτελώντας την εντολή:

cd /opt/moteiv/apps/TOSBase

Στην συνέχεια εκτελώντας την εντολή:

make tmote install

κάνουμε compile την εφαρμογή και την εγκαθιστούμε στον κόμβο.

Σχήμα 19: Η εντολή **make tmote install**

Επαναλαμβάνουμε την ίδια διαδικασία για τους κόμβους μετρήσεων, με την διαφορά ότι μέσω του Cygwin στον φάκελο

/opt/apps/Oscilloscope

κάνουμε compile και φορτώνουμε την εφαρμογή Oscilloscope. Η εντολή που θα εκτελέσουμε σε αυτή την περίπτωση είναι η:

make tmote install,#.

Χρησιμοποιούμε την συγκεκριμένη εντολή, καθώς εκτός από το να κάνει compile την εφαρμογή και να την φορτώνει στον κόμβο, θέτει και ένα διαφορετικό id σε κάθε κόμβο. Το συγκεκριμένο id, ονομάζεται node_id και είναι αυτό που θα μας βοηθήσει να ξεχωρίζουμε τους κόμβους μεταξύ τους, αλλά και να καθορίζουμε από ποιόν κόμβο προέρχονται κάθε φορά οι μετρήσεις που λαμβάνουμε. Η τιμή του id ορίζεται από εμάς κάθε φορά, και είναι η τιμή η οποία θα τοποθετηθεί στην παράμετρο (#) της παραπάνω εντολής.

7.2 Παρουσίαση του GUI

Το GUI είναι η JAVA εφαρμογή η οποία αναλαμβάνει την επικοινωνία με τον χρήστη. Την επικοινωνία αυτή την επιτυγχάνει μέσω της γραφικής διεπαφής που διαθέτει. Η κύρια λειτουργικότητα της εφαρμογής υλοποιείται στα παρακάτω **.java** αρχεία.

- **ScopeDriver.java**: βρίσκεται υλοποιημένη η κλάση ScopeDriver, η λειτουργεί ως διεπαφή, για την μεταφορά των μετρήσεων, ανάμεσα στον SerialForwarder και στις υπόλοιπες κλάσης της εφαρμογής.
- **ChoicePanel.java**: περιέχει την υλοποίηση της κλάσης ChoicePanel, η οποία δίνει την δυνατότητα στον χρήστη να επιλέξει τα χαρακτηριστικά που επιθυμεί να έχει το γράφημα που θα δημιουργήσει. (node_ids & channel_ids).
- **Channel.java**: περιέχει την υλοποίηση της κλάσης Channel η οποία περιέχει τις τιμές και τις ιδιότητες των μετρήσεων που λαμβάνει κάθε φορά από την κλάση ScopeDriver.
- **GraphPanel.java**: περιέχει την υλοποίηση της κλάσης GraphPanel, η οποία αναλαμβάνει να εμφανίσει τις μετρήσεις μέσω γραφικού περιβάλλοντος στον χρήστη. Οι μετρήσεις που παρουσιάζονται πρέπει να ικανοποιούν τα χαρακτηριστικά που έχει ορίσει ο χρήστης με την βοήθεια της κλάσης ChoicePanel.
- **Oscilloscope.java**: περιέχει την συνάρτηση main της εφαρμογής.

Τέλος αξίζει να σημειωθεί ότι σε πολλά από τα παραπάνω αρχεία (Channel.java, GraphPanel.java) γίνεται "εισαγωγή" (include) της βιβλιοθήκης *java.sql.** . Η βιβλιοθήκη αυτή αποτελεί μέρους του JDBC API και παρέχει τις απαραίτητες μεθόδους για την επικοινωνία της εφαρμογής μας με την βάση δεδομένων.

7.3 Μετασχηματισμοί των δεδομένων

Αρχικά οι μετρήσεις που λαμβάνουμε από τους είναι αισθητήρες είναι σε μη-σταθμισμένη μορφή. Για να μπορέσουμε να παρουσιάσουμε τις κανονικοποιημένες τιμές των μετρήσεων στον χρήστη, θα πρέπει να γίνουν διάφορες μαθηματικές μετατροπές σε αυτές. Οι συναρτήσεις που εκτελούν αυτές τις μαθηματικές μετατροπές βρίσκονται υλοποιημένες στο αρχείο **convertData.java** του προγράμματος GUI.

Για να μετατρέψουμε την μέτρηση του αισθητήρα της εσωτερικής τάσης στο αντίστοιχο αριθμό που δείχνει τάση εφαρμόζουμε την παρακάτω εξίσωση:

(1) value/4096 * Vref , όπου Vref = 1.5V.

Ομοίως με την εσωτερική τάση, η εσωτερική θερμοκρασία είναι ένα μη καλιμπραρισμένο μέγεθος. Η παρακάτω εξίσωση μετατρέπει το μέγεθος αυτό σε βαθμούς Κελσίου:

T = (Vtemp – 0.986) / 0.00355.

Όσον αφορά την ενεργή φωτοσυνθετική (PAR) και την ολική ηλιακή ακτινοβολία (TSR), παίρνουμε τις μη καλιμπραρισμένες τιμές από τον 12-bit converter με Vref = 1.5V. Οι φωτοδίοδοι δημιουργούν ρεύμα μέσω μιας αντίστασης 100kOhm. Αφού υπολογίσουμε την τάση από την εξίσωση (1), την μετατρέπουμε σε ρεύμα χρησιμοποιώντας την εξίσωση V = IR:

(2) I = Vsensor / 100.000 , όπου Vsensor το αποτέλεσμα της εξίσωσης (1).

Βασιζόμενοι σε γραφικές παραστάσεις που είναι διαθέσιμες στο Hamamatsu S1087 datasheet, το ρεύμα του αισθητήρα μπορεί να μετατραπεί σε Lux χρησιμοποιώντας τις παρακάτω εξισώσεις:

S1087	lx = 0.625 * le6 * I * 1000
S1087-01	lx = 0.769 * le5 * I * 1000

Η υγρασία και η θερμοκρασία μετρούνται από τον εξωτερικό Sensirion αισθητήρα. Η μετατροπή σε SI μονάδες γίνεται ως εξής:

Για την θερμοκρασία, η εφαρμογή Oscilloscope επιστρέφει μια 14-bit τιμή, η οποία μπορεί να μετατραπεί σε βαθμούς Κελσίου με την εξίσωση:

(3) temperature = -39.60 + 0.01 * Sot , όπου Sot είναι η τιμή του αισθητήρα.

Για την υγρασία η 12-bit τιμή που δίνει ο αισθητήρας δεν είναι αντισταθμισμένη σε σχέση με την θερμοκρασία. Η τιμή της δίνεται από την παρακάτω εξίσωση:

(4) humidity = -4 + 0.0405 * Sorh + (-2.8 * 10^-6) * (Sorh^2) , όπου Sorh η τιμή που δίνει ο αισθητήρας.

Λαμβάνοντας υπόψιν μας και την θερμοκρασία μπορούμε να διορθώσουμε την τιμή της υγρασίας ως εξής:

(5) humidity_true = (Tc - 25) * (0.01 + 0.00008*SOrh) + humidity , όπου Tc η θερμοκρασία σε βαθμούς Κελσίου όπως μετρήθηκε στην εξίσωση (3) και humidity η που προκύπτει από την εξίσωση (4).

7.4 Εκτέλεση της εφαρμογής

Για την μετάφραση των αρχείων της εφαρμογής, μέσω του Cygwin, πηγαίνουμε στον φάκελο:

cd /opt/moteiv/tools/java/com/moteiv/GUI,

όπου υπάρχουν τα αρχεία του GUI. Εκτελούμε την εντολή *make* έτσι ώστε τα αρχεία αυτά να γίνουν compiled.
Έπειτα θέλοντας να εκτελέσουμε την εφαρμογή καλούμε την παρακάτω εντολή:

MOTECOM=serial@COM5:tmote java com.moteiv.GUI.oscilloscope.

Η παραπάνω εντολή μπορεί ουσιαστικά να διαιρεθεί στα δύο παρακάτω επιμέρους τμήματα:

(1) MOTECOM=serial@COM5:tmote

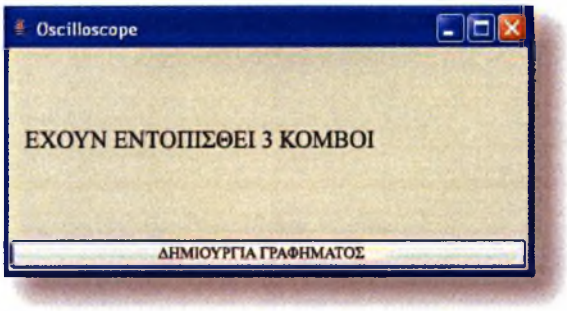
(2) java com.moteiv.GUI.oscilloscope

Στο (1) τμήμα της εντολής δίνονται οδηγίες, μέσω της παραμέτρου MOTECOM, στην εφαρμογή SerialForwarder. Οι οδηγίες αυτές αφορούν τον καθορισμό της σειριακής θύρας μέσω της οποίας θα γίνει η επικοινωνία με τον σταθμό βάσης, καθώς και τον καθορισμό του ρυθμού ανταλλαγής των δεδομένων ανάμεσα στον σταθμό βάσης και στην εφαρμογή. Στην συγκεκριμένη περίπτωση η επικοινωνία θα γίνει μέσω της θύρας 5 (*serial@COM5*) και ο ρυθμός επικοινωνίας θα είναι κατάλληλος για τους αισθητήρες Tmote Sky, δηλαδή 57600 (*:tmote*).

Στο (2) τμήμα της εντολής καλούμε την JAVA να τρέξει το εκτελέσιμο αρχείο, το οποίο δημιουργήθηκε από την μετάφραση, των αρχείων της εφαρμογής GUI, που δημιουργήσαμε παραπάνω.

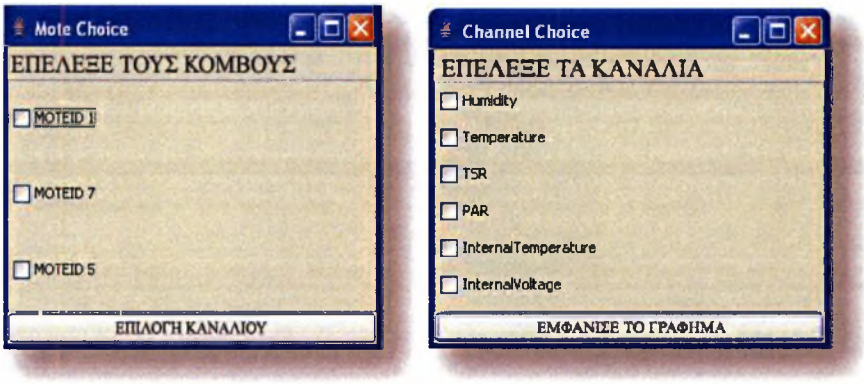
Η εφαρμογή, τρέχοντας την παραπάνω εντολή, θα αρχίσει την εκτέλεσή της. Αρχικά θα εμφανίσει το παρακάτω μήνυμα. Το συγκεκριμένο μήνυμα ενημερώνει τον χρήστη για τον αριθμό των "κόμβων μετρήσεων" οι οποίοι έχουν εντοπισθεί από τον κόμβο βάσης. Ακόμα διαθέτει το κουμπί « Δημιουργία Γραφήματος », το οποίο δίνει την

δυνατότητα στον χρήστη να δημιουργήσει ένα γράφημα με τις μετρήσεις τις οποίες λαμβάνει η εφαρμογή.



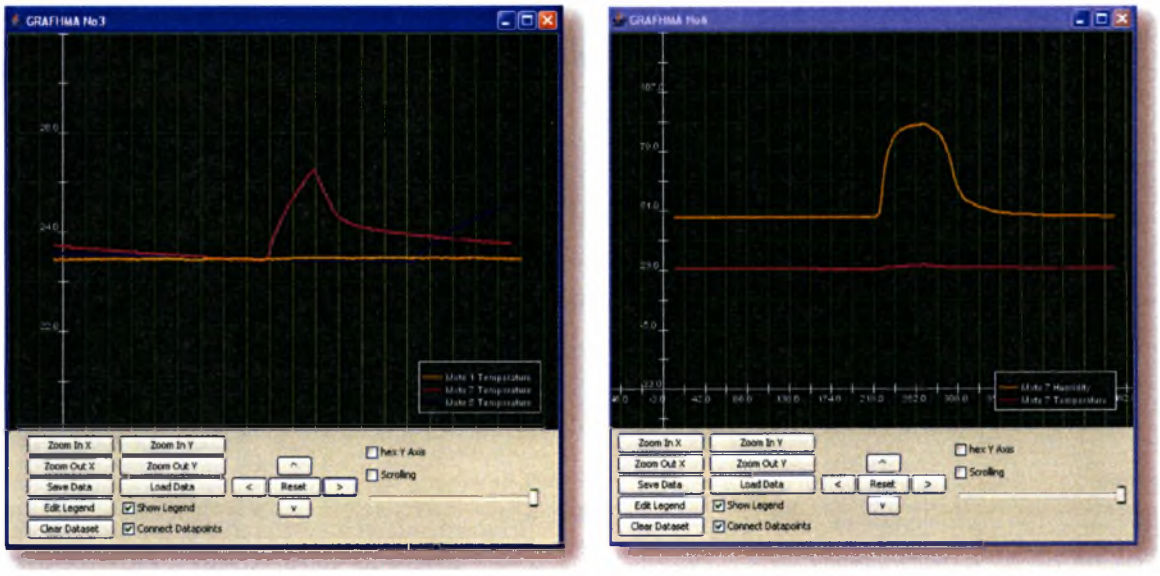
Σχήμα 20: Αρχικό παράθυρο του GUI

Αφού αποφασίσει την δημιουργία γραφήματος, ο χρήστης καλείται πρώτα να επιλέξει τους κόμβους των οποίων τις μετρήσεις επιθυμεί να δει και έπειτα να επιλέξει το είδος των μετρήσεων, τα κανάλια των οποίων οι μετρήσεις θα εμφανίζονται. Οι συγκεκριμένες επιλογές τους χρήστη γίνονται μέσα από τα παρακάτω "παράθυρα".



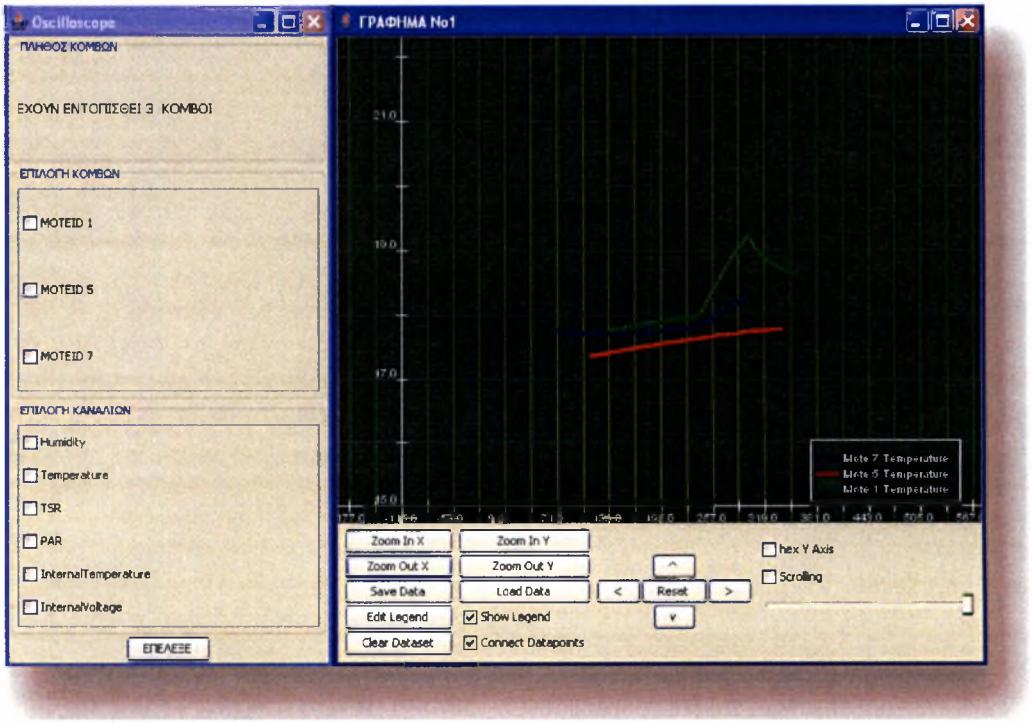
Σχήμα 21: Παράθυρα εκχώρησης επιλογών του χρήστη

Αφού ο χρήστης επιλέξει τα χαρακτηριστικά των μετρήσεων που επιθυμεί, θα εμφανιστεί ένα γράφημα με τις επιλεγμένες μετρήσεις. Παρακάτω φαίνονται 2 παρόμοια γραφήματα. Στο ένα εμφανίζονται οι τιμές τις θερμοκρασίας που λαμβάνονται και από τους τρεις κόμβους, και στο άλλο οι τιμές της θερμοκρασίας και της υγρασίας που λαμβάνονται από τον κόμβο με id = 7.



Σχήμα 22: Παράθυρα εμφάνισης των μετρήσεων στον χρήστη

Τελικά η μορφή που θα έχει τα γραφικό περιβάλλον της εφαρμογής μας θα είναι:



Στο παράθυρο του γραφήματος ο χρήστης έχει την δυνατότητα να επιλέξει το κουμπί "Save Data". Με την χρήση αυτού του κουμπιού ο χρήστης έχει την δυνατότητα να επιλέξει την αποθήκευση των εμφανιζόμενων μετρήσεων. Η αποθήκευση αυτή γίνεται είτε σε ένα .txt αρχείο, είτε στη βάση δεδομένων.

Παρακάτω βλέπουμε ένα παράδειγμα αποθήκευσης μετρήσεων σε ένα .txt αρχείο.

```
# Test Data File
# Tue Sep 25 07:43:15 EEST 2007
# BEGIN CHANNEL DATA: 6 SAMPLES
# Mote 5 Chan 1
8960.0 23.08
8961.0 23.1
8962.0 23.1
8963.0 23.090000000000003
8964.0 23.090000000000003
8965.0 23.1
# BEGIN CHANNEL DATA: 6 SAMPLES
# Mote 7 Chan 1
9260.0 22.979999999999997
9261.0 23.0
9262.0 22.990000000000002
9263.0 23.0
9264.0 22.990000000000002
9265.0 23.0
# BEGIN CHANNEL DATA: 6 SAMPLES
# Mote 1 Chan 1
9480.0 23.270000000000003
9481.0 23.28
9482.0 23.270000000000003
9483.0 23.259999999999998
9484.0 23.270000000000003
9559.0 23.270000000000003
```

Σχήμα 23: Αποθήκευση μετρήσεων σε **.txt** αρχείο.

Παράδειγμα αποθήκευσης στη βάση δεδομένων.

result_time	nodeid	packetid	temp	TSR	PAR	humidity	int_voltage	int_temp
2007-09-25 06:58:51	1	2398	23.37	0.1951304296875	0.1951318725585594	47.403092	0.9759521484375	2914
2007-09-25 06:58:51	1	2399	23.38	0.189090728759766	0.1951318725585594	47.403092	0.9759521484375	2915
2007-09-25 06:58:51	7	2340	23.16	0.18142089843750	0.1813664721679688	48.5404248	0.9071044921875	2926
2007-09-25 06:58:51	7	2341	23.15	0.18142089843750	0.1813664721679688	48.5404248	0.9071044921875	2926
2007-09-25 06:58:51	7	2342	23.16	0.18142089843750	0.1883421057128906	48.5081788	0.9071044921875	2923
2007-09-25 06:58:51	7	2343	23.15	0.18142089843750	0.1813664721679688	48.5081788	0.9071044921875	2926
2007-09-25 06:58:51	7	2344	23.16	0.18709030151367200	0.1883421057128906	48.5081788	0.9071044921875	2926
2007-09-25 06:58:51	7	2345	23.15	0.18709030151367200	0.1883421057128906	48.5081788	0.9071044921875	2924
2007-09-25 06:58:51	7	2346	23.16	0.19275970458984400	0.1883421057128906	48.5081788	0.9071044921875	2926
2007-09-25 06:58:51	7	2347	23.16	0.18709030151367200	0.1883421057128906	48.5081788	0.9071044921875	2923
2007-09-25 06:58:51	7	2348	23.17	0.19275970458984400	0.1883421057128906	48.4759272	0.9071044921875	2926
2007-09-25 06:58:51	7	2349	23.16	0.19275970458984400	0.1883421057128906	48.4759272	0.9071044921875	2923
2007-09-25 06:58:51	5	2260	23.21	0.09144287109375	0.09141543823242190	47.2535752	0.91552734375	2927
2007-09-25 06:58:51	5	2261	23.22	0.09715805053710940	0.1054793518066410	47.2535752	0.91589355468750	2927
2007-09-25 06:58:51	5	2262	23.23	0.09715805053710940	0.09844739501953130	47.2535752	0.91552734375	2926
2007-09-25 06:58:51	5	2263	23.23	0.09144287109375	0.09844739501953130	47.2535752	0.91552734375	2927
2007-09-25 06:58:51	5	2264	23.21	0.09715805053710940	0.1054793518066410	47.2535752	0.91552734375	2926
2007-09-25 06:58:51	5	2265	23.22	0.09715805053710940	0.09844739501953130	47.2535752	0.91552734375	2927
2007-09-25 06:58:51	5	2266	23.2	0.09144287109375	0.1054793518066410	47.2211004	0.91552734375	2927
2007-09-25 06:58:51	5	2267	23.22	0.09715805053710940	0.09844739501953130	47.2211004	0.91552734375	2927

Σχήμα 24: Αποθήκευση μετρήσεων σε βάση δεδομένων

8 ΕΠΙΛΟΓΟΣ

8.1 Μελλοντικές δυνατότητες επέκτασης.

Όπως είδαμε παραπάνω, το σύστημα μας μπορεί να χρησιμοποιηθεί αποδοτικά για την παρακολούθηση και τον έλεγχο των συνθηκών που επικρατούν σε κάποιο χώρο. Το σύστημα έχει πολλές δυνατότητες επέκτασης. Η σημαντικότερη από αυτές είναι η δυνατότητα προώθησης και παρακολούθησης των μετρήσεων μέσω του Διαδικτύου. Με αυτόν τον τρόπο ο κάθε χρήστης θα μπορεί να ελέγχει και να παρακολουθεί τις συνθήκες που επικρατούν στην περιοχή λειτουργίας του συστήματος, από οποιοδήποτε μέρος του κόσμου. Χρειάζεται δηλαδή η υλοποίηση μιας ρουτίνας η οποία θα αναλαμβάνει να προωθεί τις μετρήσεις στο Διαδίκτυο, και ενός ιστοχώρου, μέσω του οποίο θα παρουσιάζονται οι μετρήσεις που λαμβάνονται.

8.2 Συμπεράσματα

Τμήματα της εργασίας μας βασίστηκαν σε νέες, αναπτυσσόμενες τεχνολογίες, όπως είναι τα δίκτυα ασύρματων αισθητήρων. Οι τεχνολογίες αυτές βρίσκονται ακόμα στα πρωταρχικά στάδια ανάπτυξής τους, αλλά σε σύντομο χρονικό διάστημα θα έχουν την δυνατότητα να παρέχουν αποδοτικές λύσεις σε ένα ευρύ φάσμα εφαρμογών. Ένα μικρό παράδειγμα αυτών των λύσεων αποτελεί και η εφαρμογή που περιγράψαμε στην συγκεκριμένη εργασία. Αποτελεί μια αποδοτική λύση, στο ζήτημα του απομακρυσμένου ελέγχου και παρακολούθησης των συνθηκών που επικρατούν σε ένα χώρο.

9 ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] http://en.wikipedia.org/wiki/Wireless_sensor_network

[2] http://en.wikipedia.org/wiki/Smartdust

[3] Römer, Kay; Friedemann Mattern (December 2004). "The Design Space of Wireless Sensor Networks". *IEEE Wireless Communications*

[4] http://w3.antd.nist.gov/wahn_ssn.shtml

[5] http://en.wikipedia.org/wiki/Sensor_node

[6] http://www.tinyos.net/

[7] http://www.moteiv.com

[8] http://www.ceid.upatras.gr/courses/katanemhmena/wiki

[9] http://en.wikipedia.org/wiki/Unified_Modeling_Language

[10] http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/index.htm

[11] http://www.cygwin.com/

[12] http://www.mysql.com/

[13] http://java.sun.com/javase/technologies/database/index.jsp

[14] http://www.moteiv.com/community/Getting_Data_from_Tmote_Sky's_Sensors

[15] http://en.wikipedia.org/wiki/IEEE_802.15.4