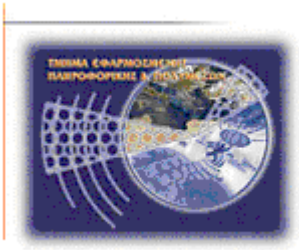
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής



Πτυχιακή Εργασία

Τίτλος: Σχεδίαση και ανάπτυξη πλατφόρμας Ασύρματου Δικτύου Αισθητήρων (WSN), με χρήση expander(MDA300) για συλλογή και επεξεργασία Αναλογικών και Ψηφιακών σημάτων

Παπαδογιάννης Νικόλαος (ΑΜ:2732)
Κοντάκη Ελευθερία (ΑΜ:1959)

Επιβλέπων καθηγητής: PH.D Βλησίδης Ανδρέας

Επιτροπή Αξιολόγησης**:** PH.D Βλησίδης Ανδρέας
PH.D Παναγιωτάκης Σπυρίδων
PH.D Στρατάκης Δημήτριος

Ημερομηνία παρουσίασης: 22 – 09 - 2014

Ευχαριστίες

Θα θέλαμε να ευχαριστήσουμε τον υπεύθυνο καθηγητή κ. Ανδρέα Βλησίδη και τον Μιχάλη Φραγκιαδάκη για την πολύτιμη βοήθειά τους κατά την διάρκεια διεκπεραίωσης της πτυχιακής μας εργασίας, καθώς και όλο το υπόλοιπο προσωπικό του εργαστηρίου ΕΝΠΕΤ.

Σύνοψη

Η παρούσα πτυχιακή εργασία έχει σκοπό την ανάπτυξη ενός Ασύρματου Δικτύου Αισθητήρων και μιας εφαρμογής λογισμικού Android μέσω της οποίας ένας χρήστης θα έχει τη δυνατότητα να παρατηρήσει τα δεδομένα των αισθητήρων του δικτύου. Για την υλοποίηση του δικτύου γίνεται χρήση των πλακετών MDA300CA, MICAZ, και MIB520CB της Crossbow και των εξωτερικών αισθητήρων θερμοκρασίας (LM35 – αναλογικός) και κίνησης (DSN-FIR800 - ψηφιακός). Το δίκτυο αποτελείται από έναν σταθμό βάσης και δύο ασύρματα nodes. Τα δεδομένα των nodes αποθηκεύονται σε μία βάση δεδομένων PostgreSQL σε έναν απομακρυσμένο υπολογιστή τα οποία ο χρήστης θα μπορεί ανά πάσα στιγμή να δει σε μορφή πίνακα και διαγραμμάτων μέσω της εφαρμογής Android. Η υλοποίηση της εφαρμογής γίνεται με τη γλώσσα προγραμματισμού JAVA μέσω του Eclipse IDE.

Abstract

The purpose of this thesis is to develop a Wireless Sensor Network and an Android application through which a user will be able to observe the sensor data of the network. For the implementation of the network we use Crossbow's boards MDA300CA, MICAZ, and MIB520CB and external temperature sensor (LM35 - analog) and motion sensor (DSN-FIR800 - digital). The network consists of a base station and two wireless nodes. The node data are stored in a PostgreSQL database on a remote computer that the user can at any time see them in tabular and graphic form through the Android application. For the implementation of the application we used the programming language JAVA with Eclipse IDE.

Πίνακας Περιεχομένων

Κατάλογος εικόνων

1 Εισαγωγή

1.1 Περίληψη

Στην σημερινή εποχή η ανάγκη για αυτοματισμούς είναι έντονη γιατί προσφέρει μεγάλη ευκολία στην σύγχρονη ζωή του ανθρώπου. Στη συγκεκριμένη πτυχιακή αναπτύσσουμε ένα σύστημα μετρήσεων με αισθητήρες που μπορούν να έχουν ποικίλες εφαρμογές όπως τηλεϊατρική, γεωργία και άλλες. Σε συνδυασμό με την εφαρμογή Android για την παρουσίαση των αποτελεσμάτων μπορεί οποιοσδήποτε και οπουδήποτε να δει τα αποτελέσματα των μετρήσεων. Τα smartphones είναι πλέον αναπόσπαστο κομμάτι της ζωής μας που ένα πολύ μεγάλο ποσοστό των ανθρώπων διαθέτει, κάτι που βοηθάει πολύ στην εξέλιξη τέτοιων συστημάτων.

1.2 Κίνητρο για την Διεξαγωγή της Εργασίας

Η εξέλιξη των Internet Of Things με τις εφαρμογές που προσφέρουν και την ευκολία υλοποίησης τους είναι ένας τομέας της πληροφορικής που τα τελευταία χρονιά έχει αναπτυχθεί σε τεράστιο βαθμό. Επίσης παρέχουν μια τεράστια γκάμα αντικειμένων και διαφορετικών project που μπορούν να γίνουν με ελάχιστο κόστος. Έτσι λαμβάνοντας υπ' όψιν την μεγάλη ζήτηση και το ενδιαφέρον γύρω από τον συγκεκριμένο τομέα, αποτέλεσε κίνητρο για την συγκεκριμένη πτυχιακή εργασία.

1.3 Σκοπός και Στόχοι Εργασίας

Σκοπός μας είναι η συγκεκριμένη πτυχιακή να χρησιμοποιηθεί σε περιβαλλοντικές, αγροτικές, ιατρικές εφαρμογές καθώς και εφαρμογές παρακολούθησης χώρου. Με τους κατάλληλους εξωτερικούς αισθητήρες μπορούμε να αλλάξουμε άρδην της εφαρμογές της πτυχιακής. Με τους συγκεκριμένους αισθητήρες που χρησιμοποιούμε έχει εφαρμογές παρακολούθησης χώρου και περιβαλλοντικών μετρήσεων που μπορούν να καλύψουν τον αγροτικό και μετεωρολογικό τομέα. Με την εφαρμογή Android που αναπτύσσουμε ένας χρήστης θα έχει τη δυνατότητα να παρακολουθεί τα δεδομένα οποιαδήποτε στιγμή από όποιο μέρος κι αν βρίσκεται.

1.4 Δομή Εργασίας

Η εργασία μπορεί να χωριστεί στα εξής μέρη:

- Το μέρος της εγκατάστασης του λογισμικού στον υπολογιστή και του ασύρματου δικτύου αισθητήρων.
- Το μέρος ανάπτυξης της εφαρμογής Android.
- Το μέρος της παρακολούθησης των δεδομένων.

2 Μεθοδολογία Υλοποίησης

Για την υλοποίηση της πτυχιακής θα χρησιμοποιηθούν δύο ασύρματα nodes τα οποία αποτελούνται το καθένα από την πλακέτα MDA300CA και την MICAZ, τα οποία θα επικοινωνούν ασύρματα μεταξύ τους και με τον σταθμό βάσης ο οποίος θα αποτελείται από την πλακέτα MIB520CB μία επίσης MICAZ. Ως εξωτερικούς αισθητήρες θα χρησιμοποιήσουμε τον αναλογικό αισθητήρα θερμοκρασίας LM35 και έναν ψηφιακό PIR αισθητήρα κίνησης DSN-FIR800. Οι εξωτερικοί αισθητήρες θα συνδέονται με το MDA300CA και θα τροφοδοτούνται από εξωτερική πηγή. Για τον προγραμματισμό του ασύρματων αισθητηρίων και του σταθμού βάσης καθώς και για την επικοινωνία μεταξύ τους θα χρησιμοποιηθεί το πρόγραμμα MoteView. Η σύνδεση στη βάση δεδομένων και η λήψη των δεδομένων θα γίνει μέσω ενός αρχείου PHP και η εφαρμογή Android θα υλοποιηθεί με το IDE Eclipse Kepler με τη χρήση XML και JAVA.

3 Σχέδιο Δράσης για την εκπόνηση της εργασίας

Σε αυτό το κεφάλαιο γίνεται μία βιβλιογραφική αναζήτηση των τεχνολογιών που σχετίζονται άμεσα ή έμμεσα με την παρούσα πτυχιακή εργασία. Στο τέλος του κεφαλαίου αναφέρονται οι στόχοι για την ολοκλήρωση της πτυχιακής.

3.1 Internet Of Things

Το Internet Of Things ή «Διαδίκτυο των Πραγμάτων» είναι η επερχόμενη εξέλιξη του Διαδικτύου των υπηρεσιών, που υπάρχει σήμερα. Πρόκειται για ένα δίκτυο όχι μόνο υπολογιστών αλλά και διασυνδεδεμένων αντικειμένων. Τα αντικείμενα αυτά θα περιέχουν ενσωματωμένα ηλεκτρονικά συστήματα και μπορούν να είναι διάφορες οικιακές συσκευές, μέσα μεταφοράς, μέσα τηλεπικοινωνίας, βιβλία, αυτοκίνητα, ακόμα και τρόφιμα. Πέρα από την εξασφάλιση της καλής λειτουργίας των διασυνδεδεμένων αυτών αντικειμένων, θα γίνει προσπάθεια να επιτευχθεί και συνεργασία μεταξύ των συστημάτων αυτών. Κάθε αντικείμενο θα χρησιμοποιεί συστήματα αναγνώρισης ραδιοσυχνοτήτων (τα γνωστά ως RFID), ένα είδος αισθητήρων, δηλαδή. Απαραίτητη προϋπόθεση για την επιτυχία του καινούριο αυτού Διαδικτύου είναι να καταστεί το σημερινό Διαδίκτυο πιο ασφαλές.

Το Διαδίκτυο των Πραγμάτων θα είναι η κορύφωση της προσπάθειας για την ολοκλήρωση και αυτοματοποίηση των υπηρεσιών που παρέχουν τα ενσωματωμένα συστήματα παντός είδους. Το διαδίκτυο θα γίνει διαδραστικό, ένα τεράστιο ιεραρχικά οργανωμένο «νευρικό σύστημα» που θα απολήγει σε συσκευές με αισθητήρες και ενεργοποιητές (actuators) που θα συνεργάζονται για έξυπνες υπηρεσίες για την υγεία, τις μεταφορές, τη διανομή και κατανάλωση ενέργειας κλπ. Στις μεταφορές σύντομα θα έχουμε συστήματα αυτόματης οδήγησης και οργάνωσης των μεταφορικών μέσων για περισσότερη ασφάλεια και οικονομία. Στον τομέα της υγείας προβλέπονται μία σειρά από καινοτομίες, από τη διαδραστική τηλεπαρακολούθηση των ασθενών, μέχρι την τηλεχειρουργική και τα έξυπνα φάρμακα. Η μεγάλη πρόκληση είναι η αυτοματοποίηση της διαχείρισης πόρων όπως για παράδειγμα σε αυτό που ονομάζεται smart grids, συνδυασμένη και αποτελεσματική χρήση εναλλακτικών μορφών ενέργειας.

Όλες αυτές θα είναι μερικές εφαρμογές που θα αλλάξουν ριζικά το σημερινό τρόπο ζωής τις ερχόμενες δεκαετίες. Ο καθηγητής τόνισε ωστόσο ότι ακόμα η κατάσταση του διαδικτύου παραμένει ιδιαίτερα επισφαλής και χρειάζεται μεγάλη δράση και κινητοποίηση για να μπορέσουν όλα αυτά να

λειτουργήσουν με ασφάλεια και αποτελεσματικότητα.

Ο συνδυασμός του internet, των αντικειμένων και κινητών υπηρεσιών, ανοίγει το δρόμο σε αυτό που λέμε «διάχυτη νοημοσύνη». Πανταχού παρούσα και απρόσκοπτη πρόσβαση σε παντοειδείς υπηρεσίες, αποτελεσματικός έλεγχος πόρων, διαδραστικότητα και συνέργεια για την επίτευξη ολοκληρωμένων στόχων. Μεγάλες εταιρίες δεν αποσκοπούν πλέον στη μεμονωμένη πώληση λογισμικού ή υπολογιστών αλλά μελετούν ολοκληρωμένες λύσεις όπου τα πληροφορικά συστήματα χρησιμοποιούνται για τη βέλτιστη διαχείριση φυσικών πόρων και την ανάπτυξη έξυπνων υπηρεσιών για τη δημιουργία ενός «έξυπνου πλανήτη». [1]

3.2 Ασύρματα Δίκτυα Αισθητήρων

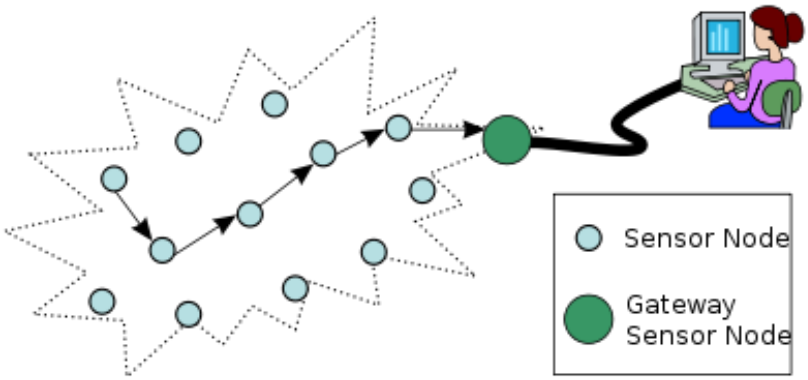
3.2.1 Γενικά

Τα Ασύρματα Δίκτυα Αισθητήρων (Wireless Sensor Networks) είναι ολοκληρωμένα συστήματα συλλογής, μετάδοσης και επεξεργασίας πληροφοριών. Βασικό χαρακτηριστικό τους γνώρισμα είναι ότι είναι λειτουργικά ανεξάρτητοι και ενεργειακά αυτόνομοι κόμβοι αισθητήρων, οι οποίοι τοποθετούνται (διασπείρονται) σε μια γεωγραφική περιοχή (περιορισμένης ή ευρείας έκτασης) με στόχο να μεταδώσουν πληροφορίες προς μια Κεντρική Μονάδα Επεξεργασίας.

Ο σχεδιασμός ενός Ασύρματου Δικτύου Αισθητήρων στηρίζεται στις δύο βασικές αρχές λειτουργίας του: τη δυνατότητα για περιοδική λήψη δεδομένων και τη δυνατότητα για εντοπισμό συμβάντων που χρίζουν άμεσης αντίδρασης/αντιμετώπισης.

Η σημαντικότητα και η προτεραιοποίηση των αρχών αυτών καθορίζεται από το είδος και τις ιδιαίτερες ανάγκες της εκάστοτε εφαρμογής. Όταν η εφαρμογή απαιτεί την περιοδική λήψη δεδομένων, πιθανές έκτακτες καθυστερήσεις στην αποστολή των στοιχείων δεν παίζουν σημαντικό ρόλο, άρα το δίκτυο μπορεί να σχεδιαστεί με προτεραιότητα την αύξηση του χρόνου ζωής. Όταν η εφαρμογή είναι προσανατολισμένη στον εντοπισμό επειγόντων συμβάντων, το δίκτυο οφείλει να στοχεύει στην αποστολή δεδομένων αμέσως χωρίς να λαμβάνει σημαντικά υπόψη το κόστος λειτουργίας. [2]

Ένα Ασύρματο Δίκτυο Αισθητήρων αποτελείται από πολλούς κόμβους, όπου ο καθένας από αυτούς συνδέεται σε έναν ή περισσότερους αισθητήρες. Κάθε τέτοιος κόμβος του δικτύου έχει χαρακτηριστικά κάποια συγκεκριμένα κομμάτια: ένα ραδιοπομποδέκτη με μια εσωτερική κεραία ή μια σύνδεση με μια εξωτερική κεραία, ένα μικροελεγκτή, ένα ηλεκτρονικό κύκλωμα για τη διασύνδεση με τους αισθητήρες και μια πηγή ενέργειας, συνήθως μια μπαταρία. Οι περιορισμοί σε μέγεθος και κόστος έχουν ως αποτέλεσμα αντίστοιχους περιορισμούς σε πόρους όπως ενέργεια, μνήμη, υπολογιστική ταχύτητα και στο εύρος ζώνης των επικοινωνιών. [3]



Εικόνα 1: Παράδειγμα δικτύου ασύρματων αισθητήρων

3.2.2 Χαρακτηριστικά Ασύρματων Δικτύων Αισθητήρων

Προσανατολισμός στην Εφαρμογή (Application Specific): Τα Ασύρματα Δίκτυα Αισθητήρων σχεδιάζονται με βάση τις εξειδικευμένες ανάγκες και απαιτήσεις της εκάστοτε εφαρμογής. Σπανίως ένα Δίκτυο το οποίο έχει υλοποιηθεί για μία συγκεκριμένη εφαρμογή μπορεί να χρησιμοποιηθεί (χωρίς προσαρμογές) σε κάποια άλλη εφαρμογή. Για παράδειγμα, ένα δίκτυο από αισθητήρες που έχουν ενταχθεί στην κεντρική θέρμανση ενός κτιρίου ρυθμίζοντας τη μέγιστη επιμέρους θερμοκρασία των ορόφων, στηρίζεται σε εντελώς διαφορετικές αρχές σχεδιασμού από ένα στατικό δίκτυο αισθητήρων που μετρούν το ύψος της βροχόπτωσης μιας γεωγραφικής περιοχής.

Κλίμακα Μεγέθους: Τα Ασύρματα Δίκτυα Αισθητήρων μπορούν να είναι όσο εκτενή ή όσο περιορισμένα απαιτεί η εφαρμογή, αποτελούμενα από μερικές δεκάδες έως και αρκετές εκατοντάδες αισθητήρες. Τόσο η αρχιτεκτονική όσο και τα δικτυακά πρωτόκολλα θα πρέπει να είναι σε θέση να διαχειριστούν τέτοια μεγέθη.

Αυτορρύθμιση (Self-Configurability) και Ανοχή Σφαλμάτων (Fault Tolerance): Τα δίκτυα λειτουργούν με αλγορίθμους ικανούς να ρυθμίσουν την τοπολογία τους τη στιγμή της εγκατάστασης, αλλά και να διαχειριστούν τυχόν μελλοντικές ανάγκες για μεταβολές σε αυτή: χρησιμοποιούν αλγορίθμους ικανούς να αντιμετωπίσουν το φαινόμενο της απώλειας αισθητήρων (π.χ. λόγω φθοράς ή και κλοπής) με τρόπο ώστε να μη διαταράσσεται η συνολική σταθερότητα λειτουργίας τους. Η συνολική κατάσταση του Δικτύου ελέγχεται ανά τακτά χρονικά διαστήματα.

Χρόνος Ζωής: Όταν πρόκειται για δίκτυα αποτελούμενα από αισθητήρες οι οποίοι τροφοδοτούνται από ηλεκτρικά στοιχεία (π.χ. μπαταρίες), ο χρόνος ζωής τους είναι άμεσα συνυφασμένος με το χρόνο ζωής των πηγών ενέργειας των αισθητήρων. Σημαντικό ρόλο στη διατήρηση της βιωσιμότητας ενός δικτύου για μεγάλο χρονικό διάστημα παίζει ο προσεκτικός σχεδιασμός γύρω από τη διαχείριση της ενέργειας η οποία απαιτείται για τη λειτουργία του κάθε αισθητήρα. Η σχέση ανάμεσα στην ποιότητα λειτουργίας και την απαιτούμενη τροφοδοσία των αισθητήρων είναι αντιστρόφως ανάλογη: ξοδεύοντας περισσότερη ενέργεια, επιτυγχάνουμε μεν καλύτερη απόδοση αλλά το δίκτυο έχει μικρότερο συνολικό χρόνο ζωής. Ακριβώς επειδή τα δίκτυα αυτά σχεδιάζονται με τρόπο ώστε να εξυπηρετήσουν συγκεκριμένες ανάγκες, ο τρόπος εξισορρόπησης της σχέσης ποιότητας-ενέργειας εξαρτάται από το είδος της εφαρμογής.

Αυτόνομη Λειτουργία και Προγραμματισμός: Κάθε ένας από τους αισθητήρες πρέπει να είναι σε θέση να λάβει αποφάσεις για τη διατήρηση της εύρυθμης λειτουργίας του δικτύου και χωρίς την παρέμβαση του χρήστη, (π.χ. αυξομειώνοντας τη συχνότητα δειγματοληψίας) αλλά και να δεχθεί επαναπρογραμματισμό (π.χ. σε περίπτωση που αλλάξει η στρατηγική χρήσης του δικτύου).

Απλότητα Σχεδιασμού: Με δεδομένο ότι οι αισθητήρες διαθέτουν περιορισμένους πόρους λειτουργίας, δεν υπάρχει η δυνατότητα να υποστηριχθεί ιδιαιτέρως υψηλή πολυπλοκότητα, ούτε σε επίπεδο λειτουργικού συστήματος ούτε σε επίπεδο αλγορίθμου λειτουργίας του δικτύου.

Ποιότητα Υπηρεσιών (Quality of Service): Σε τέτοια δίκτυα η έννοια της ποιότητας υπηρεσιών μπορεί να διαφέρει σημαντικά από τα συνήθη πρότυπα των Ad Hoc δικτύων. Για παράδειγμα, σε ένα Ασύρματο Δίκτυο Αισθητήρων, θα μπορούσε να μην παίζει τόσο σημαντικό ρόλο η ταχύτητα μετάδοσης των δεδομένων όσο η αξιόπιστη μετάδοση του συνόλου χωρίς απώλειες, αποφεύγοντας τις επανεκπομπές.

Για να καλυφθούν οι παραπάνω απαιτήσεις έχουν αναπτυχθεί νέοι τρόποι ασύρματης επικοινωνίας μεταξύ των δομικών στοιχείων (κόμβων) του δικτύου. Οι κυριότεροι από αυτούς είναι:

Επικοινωνία Πολλαπλών Βημάτων (Multi-Hop): Οι ενδιάμεσοι κόμβοι του δικτύου λειτουργούν ως «προωθητές» μηνυμάτων. Ένα τέτοιο είδος επικοινωνίας είναι χρήσιμο όταν η απευθείας σύνδεση δύο κόμβων είναι ανέφικτη (ή κρίνεται ασύμφορη, λόγω του υψηλού κόστους της ενέργειας μετάδοσης).

Δεδομενο-κεντρική Επικοινωνία (Data-Centric): Αισθητήρες οι οποίοι πραγματοποιούν μετρήσεις του ίδιου φυσικού μεγέθους, μπορούν να ομαδοποιηθούν από την Κεντρική Μονάδα Παρακολούθησης βάσει του φαινομένου παρακολούθησης (και όχι π.χ. βάσει της διεύθυνσης IP).

Συνάθροιση Δεδομένων (Data Aggregation): Αισθητήρες οι οποίοι πραγματοποιούν μετρήσεις του ίδιου φυσικού μεγέθους, υπάρχει η δυνατότητα να μεταδώσουν προς την Κεντρική Μονάδα Παρακολούθησης πλεονάζοντα (ή επικαλυπτόμενα) στοιχεία. Η «περιττή» πληροφορία εξαλείφεται ώστε να εξοικονομηθεί ενέργεια και να βελτιωθεί ο συνολικός ρυθμός απόκρισης του δικτύου. [2]

3.2.3 Πρότυπα Επικοινωνίας

Παρακάτω γίνεται μία αναφορά στα γνωστότερα πρότυπα επικοινωνίας που σχετίζονται με τα ασύρματα δίκτυα. Ιδιαίτερη έμφαση δίνεται στο πρότυπο ZigBee.

3.2.3.1 IEEE 802.15.4

Το πρωτόκολλο IEEE 802.15.4 προσδιορίζει τα χαρακτηριστικά του φυσικού επιπέδου (PHY) και του υπό-επιπέδου Ελέγχου Προσπέλασης στο Μέσο Μετάδοσης MAC (Media Access Control) για ασύρματη, χαμηλής ροής και περιορισμένης εμβέλειας επικοινωνία μεταξύ σχετικά απλών συσκευών, που καταναλώνουν ελάχιστη ενέργεια και τυπικά λειτουργούν σε ένα προσωπικό χώρο λειτουργίας (Personal Operating Space, POS). Είναι σχεδιασμένο συνεπώς για Low Rate Wireless Personal Area Networks (LR-WPAN) πλεονέκτημα των οποίων συνιστά η ευκολία στην εγκατάσταση, η αξιόπιστη μετάδοση δεδομένων, η λειτουργία περιορισμένης έκτασης, το χαμηλό κόστος και η λογική διάρκεια ζωής της μπαταρίας, ενώ διατηρούν παράλληλα μια απλή και ευέλικτη στοίβα πρωτοκόλλων. Οι ασύρματες ζεύξεις υπό το πρότυπο 802.15.4 μπορούν να λειτουργήσουν σε τρεις ISM (Industrial Scientific Medical) ζώνες συχνοτήτων, με ρυθμούς δεδομένων 250kbps στη ζώνη των 2.4 GHz, 40kbps στη ζώνη των 915 MHz και 20 kbps στη ζώνη των 868 MHz. Στο πρωτόκολλο 802.15.4 εκχωρούνται συνολικά 27 κανάλια, με 16 κανάλια στη ζώνη των 2.4 GHz, 10 κανάλια στη ζώνη των 915 MHz και 1 κανάλι στη ζώνη των 868 MHz. [2]

Το IEEE802.15.4 είναι η βάση για τα πρότυπα ZigBee, ISA100.11a και WirelessHART που περιγράφονται παρακάτω. Κάθε ένα από αυτά τα πρότυπα επεκτείνει το IEEE 802.15.4 υλοποιώντας τα πάνω layers τα οποία δεν καθορίζονται σε αυτό. [10]

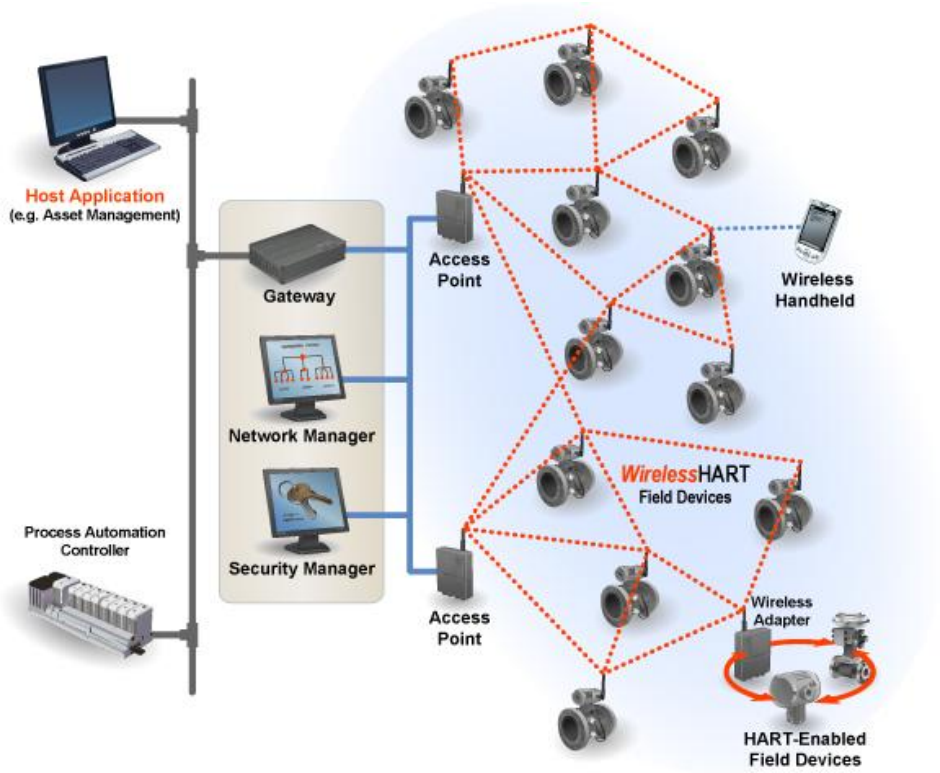
3.2.3.2 WirelessHART

Το WirelessHART Standard είναι η ασύρματη εκδοχή του πρωτοκόλλου HART. Είναι πρωτόκολλο ασύρματης επικοινωνίας για μετρήσεις διεργασιών και εφαρμογές έλεγχου, στηρίζεται στο πρότυπο IEEE 802.15.4 για λειτουργία χαμηλής ισχύος 2.4GHz και είναι συμβατό με όλες τις υπάρχουσες συσκευές, εργαλεία και συστήματα. Διαθέτει αξιοπιστία, ασφάλεια και ενεργειακή αποδοτικότητα. [4]

Το πρωτόκολλο HART για έξυπνες δικτυακές συσκευές πεδίου υπάρχει από τα τέλη της δεκαετίας του ’80. Το WirelessHART κυκλοφόρησε μαζί με το HART 7 το 2007. Το WirelessHART κάνει πιο εύκολη και φθηνή τη χρήση των τεχνολογιών HART οι οποίες είχαν ήδη αναπτυχθεί.

Το HART περιλαμβάνει πέντε επίπεδα του μοντέλου OSI: το φυσικό επίπεδο, το επίπεδο ζεύξης δεδομένων, το επίπεδο δικτύου, το επίπεδο μεταφοράς και το επίπεδο εφαρμογής. Η κύριες διαφορές ανάμεσα στην ενσύρματη και την ασύρματη έκδοση του HART είναι στο φυσικό, το δικτύου και το επίπεδο ζεύξης δεδομένων, συγκεκριμένα το ενσύρματο δεν έχει καθόλου επίπεδο δικτύου. Παρακάτω αναλύονται περισσότερο τα επίπεδα του WirelessHART.

- **Φυσικό επίπεδο:** Το φυσικό επίπεδο προέρχεται από από το πρωτόκολλο IEEE 802.15.4. Ουσιαστικά αποτελεί ένα υποσύνολο αυτού με κάποιες μετατροπές. Λειτουργεί μόνο στις συχνότητες που καθορίζονται από το IEEE 802.15.4 – στα 2450MHz της ζώνης ISM (Industrial, Scientific, Medical). Το φυσικό επίπεδο χρησιμοποιεί 15 κανάλια της ζώνης τα οποία το επίπεδο ζεύξης δεδομένων εκμεταλλεύεται για να αυξήσει την αξιοπιστία.
- **Επίπεδο ζεύξης δεδομένων:** Το επίπεδο ζεύξης δεδομένων κάνει χρήση superframes και της τεχνολογίας TDMA (Time Dimension Multiple Access) για να προσφέρει επικοινωνία χωρίς collisions. Τα superframes χρησιμοποιούνται για να ελέγξουν το συγχρονισμό των μεταδόσεων εξασφαλίζοντας αξιόπιστη επικοινωνία και μείωση των collisions.
- **Επίπεδα Δικτύου και Μεταφοράς:** Τα επίπεδα δικτύου και μεταφοράς συνεργάζονται για να χειριστούν διάφορους τύπους κίνησης, δρομολόγησης, δημιουργίας session και ασφάλειας. Το WirelessHART εγκαθιδρύει ένα δίκτυο mesh το οποίο απαιτεί κάθε συσκευή να είναι ικανή να προωθήσει πακέτα σε άλλες συσκευές. Στην πραγματικότητα οι λειτουργίες του επιπέδου δικτύου είναι ένας συνδυασμός των επιπέδων δικτύου-μεταφοράς-συνόδου που χειρίζονται όλες τις λειτουργίες που απαιτούνται από το πρωτόκολλο σε αυτά τα τρία επίπεδα του μοντέλου OSI.
- **Επίπεδο Εφαρμογής:** Το επίπεδο εφαρμογής χειρίζεται την επικοινωνία μεταξύ του gateway και των συσκευών μέσω μιας σειράς εντολών και αποκρίσεων. Αυτό το επίπεδο παίρνει την εντολή από ένα μήνυμα, την εκτελεί και παράγει μία απόκριση. Σε αυτό το επίπεδο ουσιαστικά δεν υπάρχει καμία διαφορά ανάμεσα στο ενσύρματο πρωτόκολλο HART και στο WirelessHART. [5]



Εικόνα 2: Παράδειγμα WirelessHart Δικτύου

3.2.3.3 ISA100.11a

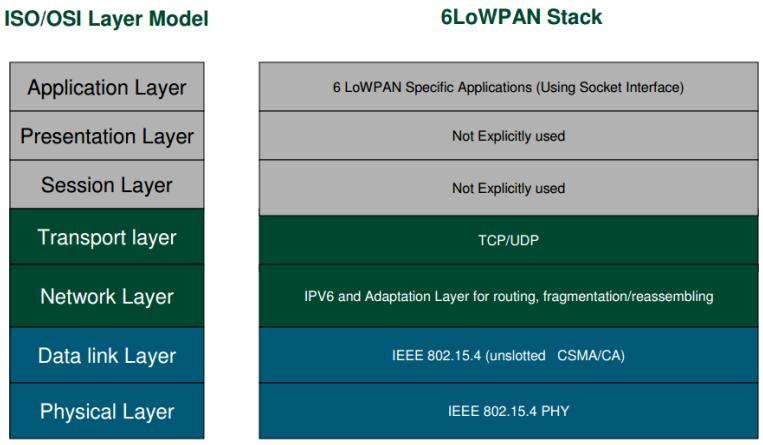
Το πρότυπο ISA100.11a βασίζεται στο IEEE 802.15.4 και είναι σχεδιασμένο για βιομηχανικούς αυτοματισμούς. Το πρότυπο αυτό προσφέρει εξαιρετική διάρκεια μπαταρίας. Μικρές καθυστερήσεις, interoperability (ικανότητα συστημάτων να δουλεύουν μαζί), επεκτασιμότητα και γενικά είναι σχεδιασμένο για εύκολη χρήση και ανάπτυξη. [6] Το ISA100.11a βασίζεται στο μοντέλο OSI και έχει πέντε επίπεδα: φυσικό επίπεδο, επίπεδο ζεύξης δεδομένων, επίπεδο δικτύου, επίπεδο μεταφοράς και επίπεδο εφαρμογής.

- **Φυσικό επίπεδο:** Το φυσικό επίπεδο βασίζεται στο IEEE802.15.4 στα 2,4 GHz.
- Επίπεδο Ζεύξης Δεδομένων: Το επίπεδο αυτό παρέχει στήριξη για τη δημιουργία, τη συντήρηση και την προώθηση πακέτων, λειτουργίες που χρειάζονται στους ασύρματους αισθητήρες. Στο μοντέλο OSI, το επίπεδο ζεύξης δεδομένων είναι ανάμεσα στο φυσικό και στο δικτύου. Καθορίζει την δομή των πακέτων, την πλαισίωση, τον εντοπισμό σφαλμάτων και την διαιτησία του διαύλου. Στο ISA100 το επίπεδο ζεύξης δεδομένων επεκτάθηκε για να περιλάβει παραπάνω λειτουργίες όπως προώθηση μηνυμάτων, εντοπισμό και επανάκτηση χαμένων μηνυμάτων και συγχρονισμό ρολογιού.
- **Επίπεδο Δικτύου:** Το επίπεδο δικτύου χρησιμοποιεί το 6LoWPAN. Κάνει χρήση του IPv6 για δρομολόγηση end-to-end.
- **Επίπεδο Μεταφοράς:** Το επίπεδο μεταφοράς προσφέρει μία ασυνδεσμική υπηρεσία βασισμένη στο UDP με έναν βελτιωμένο έλεγχο ακεραιότητας μηνυμάτων και ασφάλεια end-to-end.
- **Επίπεδο Εφαρμογής:** Το ISA100.11a καθορίζει μόνο ένα σύνολο υπηρεσιών για εφαρμογές χρηστών και όχι μία εφαρμογή για αυτοματισμό διεργασιών. Μόνο η εφαρμογή διαχείρισης συστήματος είναι καθορισμένη. [7]

3.2.3.4 6LoWPAN

Το 6LoWPAN προέκυψε από την ιδέα ότι «το πρωτόκολλο του Internet μπορεί και πρέπει να εφαρμόζεται ακόμα και στις μικρότερες συσκευές» και ότι οι συσκευές χαμηλής ενέργειας με περιορισμένες επεξεργαστικές δυνατότητες θα πρέπει να είναι σε θέση να συμμετέχουν στο Internet Of Things.

Η ομάδα του 6loWPAN έχει καθορίσει μηχανισμούς ενθυλάκωσης και συμπίεσης των κεφαλίδων που επιτρέπουν στα IPv6 πακέτα να στέλνονται και να λαμβάνονται από δίκτυα βασισμένα στο IEEE 802 15.4.[8] Αυτό γίνεται με την ενσωμάτωση ενός επιπέδου προσαρμογής πάνω από το επίπεδο ζεύξης δεδομένων του IEEE 802.15.4 το οποίο παρέχει την δυνατότητα TCP/IP επικοινωνίας πάνω από αυτό το επίπεδο προσαρμογής. [9] Τα IPv4 και IPv6 αναλαμβάνουν την παράδοση των δεδομένων για τα τοπικά, τα μητροπολιτικά και τα δίκτυα ευρείας περιοχής, όπως το Internet. [8]



Εικόνα 3: Τα επίπεδα του 6LoWPAN

3.2.3.5 ZigBee

Το ZigBee είναι μία από τις πιο νέες τεχνολογίες στο χώρο των ασύρματων δικτύων προσωπικού χώρου (WPANs). Προήλθε από τη συνεργασία της εταιρίας ZigBee Alliance με την επιτροπή IEEE 802.15.4 και παρέχει τη δυνατότητα για συνδέσεις συσκευών με χαμηλό ρυθμό μετάδοσης, χαμηλό κόστος και χαμηλή κατανάλωσης ισχύος.

Το ZigBee είναι μια ασύρματη τεχνολογία που αναπτύσσεται ως ανοικτά σφαιρικά πρότυπα για να καλύψει τις μοναδικές ανάγκες των χαμηλού κόστους, χαμηλής ισχύος, ασύρματων δικτύων αισθητήρων. Συγκεκριμένα το ZigBee είναι το όνομα μιας προδιαγραφής για μια ακολουθία υψηλού επιπέδου πρωτοκόλλων επικοινωνίας που χρησιμοποιούν οι μικροί, χαμηλής ισχύος ψηφιακοί δεκτές βασισμένοι στο 802.15.4 πρότυπο της IEEE για τα ασύρματα προσωπικά τοπικά δίκτυα(WPAN), όπως για παράδειγμα τα ασύρματα ακουστικά που συνδέονται με τα κινητά τηλέφωνα. Η τεχνολογία προορίζεται να είναι απλούστερη και φτηνότερη από άλλα ασύρματα προσωπικά, τοπικά δίκτυα (WPAN), όπως το Bluetooth. Το ZigBee στοχεύει στις εφαρμογές ραδιοσυχνότητας (RF) που απαιτούν ένα χαμηλό ρυθμό μεταφοράς δεδομένων, μεγάλη ζωή μπαταριών, και εξασφαλισμένη δικτύωση. Τα πρότυπα εκμεταλλεύονται πλήρως το 802.15.4 πρότυπο της IEEE και λειτουργούν στις χωρίς άδεια ζώνες παγκοσμίως στις ακόλουθες συχνότητες: 2.400-2.484 GHZ, 902-928 MHZ και 868.0-868.6 MHZ. [11]

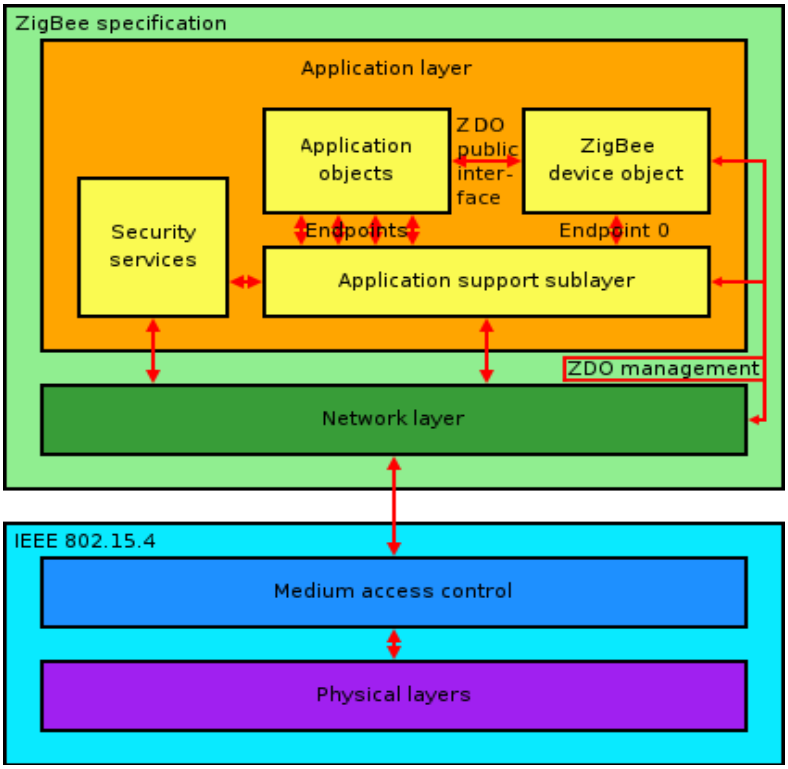
Χαρακτηριστικά γνωρίσματα του ZigBee είναι:

- Χαμηλός κύκλος καθήκοντων - παρέχει μακρά ζωή μπαταριών
- Χαμηλή λανθάνουσα κατάσταση
- Υποστηρίζει πολλές τοπολογίες δικτύων
- Άμεσο απλωμένο φάσμα ακολουθίας (DSSS - Direct Sequence Spread Spectrum)
- Μέχρι 65.000 κόμβοι σε ένα δίκτυο
- Κρυπτογράφηση - παρέχει ασφαλείς συνδέσεις μεταξύ των συσκευών
- Αποφυγή collisions
- Ποιοτική ένδειξη συνδέσεων
- Σαφής αξιολόγηση των καναλιών

Η στοίβα πρωτοκόλλων του ZigBee αποτελείται από 4 επίπεδα. Κάθε επίπεδο εκτελεί ένα συγκεκριμένο σύνολο λειτουργιών και παρέχει τις υπηρεσίες του στο ανώτερο επίπεδο μέσω μιας διεπαφής που ονομάζεται σημείο πρόσβασης υπηρεσιών (service access point, SAP). Τα 4 επίπεδα της στοίβας πρωτοκόλλων του ZigBee είναι τα παρακάτω:

- Το **φυσικό επίπεδο** (Physical layer, PHY). Είναι υπεύθυνο για την ενεργοποίηση και απενεργοποίηση του πομποδέκτη, μετάδοση και λήψη δεδομένων, ανίχνευση ενέργειας στο κανάλι, εκτίμηση της κατάστασης των καναλιών για την πολλαπλή πρόσβαση με ανίχνευση φέροντος και με αποφυγή συγκρούσεων (CSMA-CA) και τη μέτρηση της ποιότητας των λαμβανομένων πακέτων.
- Το **επίπεδο ελέγχου πρόσβασης στο μέσο** (Medium access control layer, MAC). Παρέχει υπηρεσίες μεταφοράς δεδομένων και διαχείρισης. Είναι υπεύθυνο για την πρόσβαση στο κανάλι, για τη διαχείριση των χρονοσχισμών και για την παροχή μιας αξιόπιστης σύνδεσης μεταξύ δύο επιπέδων MAC. Επιπρόσθετα παρέχει τα μέσα για την εφαρμογή διαφόρων μηχανισμών ασφάλειας.
- Το **επίπεδο δικτύου** (Network layer, NWK). Είναι υπεύθυνο για τη δημιουργία του δικτύου, για την είσοδο και την έξοδο μία συσκευής από ένα δίκτυο, για την ασφάλεια και για τη δρομολόγηση των μεταδιδόμενων πακέτων.
- Το **επίπεδο εφαρμογών** (Application layer, APL). Περιλαμβάνει το υποεπίπεδο υποστήριξης εφαρμογών (Application support sublayer, APS), το πλαίσιο εφαρμογών (Application framework,

AF), τα αντικείμενα συσκευής ZigBee (ZigBee Device Objects, ZDO) και τις καθορισμένες από τον κατασκευαστή εφαρμογές. Το υποεπίπεδο APS είναι υπεύθυνο για τη σύνδεση δύο συσκευών βάση των αναγκών και των υπηρεσιών τους και για την αποστολή δεδομένων μεταξύ τους. Τα ZDO είναι αυτά που καθορίζουν το ρόλο της κάθε συσκευής στο δίκτυο και το επίπεδο ασφάλειας. Επίσης συμβάλλουν στην ανίχνευση των συσκευών σε ένα δίκτυο και στον προσδιορισμό των υπηρεσιών που αυτές παρέχουν. Το πλαίσιο εφαρμογών είναι το περιβάλλον στο οποίο φιλοξενούνται οι εφαρμογές μέσα σε μία συσκευή ZigBee. [3]



Εικόνα 4: ZigBee Protocol Stack

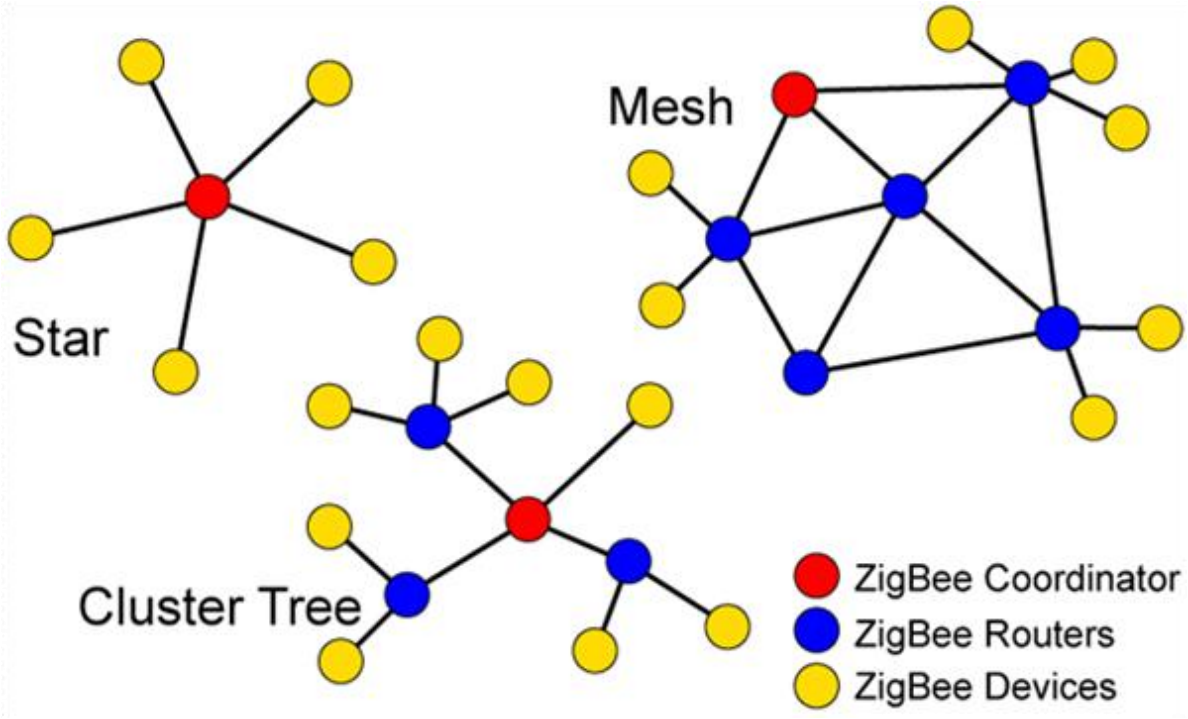
Τοπολογίες Δικτύων ZigBee

Η απλούστερη τοπολογία είναι η **τοπολογία αστεριού**, αποτελείται από ένα ενιαίο συντονιστή συνδεδεμένο με μια σειρά από συσκευές. Ο συντονιστής έχει την αρμοδιότητα να ελέγχει και να συντονίζει το δίκτυο, οι άλλες συσκευές είναι απευθείας συνδεδεμένες με το συντονιστή και επικοινωνούν μόνο μέσω αυτού με άλλες συσκευές. Ένα παράδειγμα ενός αστέρα δικτύου είναι ένα οικιακό σύστημα ασφαλείας. Το πάνελ ασφαλείας θα ενεργεί ως συντονιστής του δικτύου παρακολούθησης, π.χ με αισθητήρες κίνησης και διακόπτες θα ελέγχει πόρτες και παράθυρα. Οι τοπολογίες αστεριού είναι κοινές και παρέχουν μεγαλύτερη διάρκεια ζωής των μπαταριών.

Μια άλλη τοπολογία είναι η **τοπολογία σημείου προς σημείο**, στην οποία κάθε συσκευή εγκαθιστά συνδέσεις σημείου προς σημείο με άλλες συσκευές που βρίσκονται μέσα στην εμβέλεια της. Με αυτό τον τρόπο δημιουργούνται δίκτυα που έχουν τη μορφή δένδρου ή πλέγματος. Με τη βοήθεια αλγορίθμων δρομολόγησης, όλες οι συσκευές μπορούν να επικοινωνήσουν μεταξύ τους. Πολλά τέτοια δίκτυα μπορούν να ενωθούν μεταξύ τους και να σχηματίσουν ένα μεγαλύτερο. Στο μεγαλύτερο δίκτυο υπάρχει μόνο ένας συντονιστής δικτύου, ενώ κάθε μικρότερο δίκτυο έχει από ένα δρομολογητή. Οι τοπολογίες αυτές παρέχουν μεγαλύτερα επίπεδα αξιοπιστίας και επεκτασιμότητας.

Συνδυασμός των δύο τοπολογιών είναι εφικτός σχηματίζοντας ένα λεγόμενο **δίκτυο δέντρου** και συνδάυζει τα οφέλη από τις δυο τοπολογίες, δηλαδή τη μεγαλύτερη διάρκεια ζωής των μπαταριών και τα

μεγαλύτερα επίπεδα αξιοπιστίας και επεκτασιμότητας. Στα μεγαλύτερα αυτά δίκτυα υπάρχει μόνο ένας συντονιστής δικτύου ενώ στα μικρότερα δίκτυα υπάρχει μόνο ένας δρομολογητής. [3]



Εικόνα 5: Τοπολογίες δικτύου ZigBee

3.2.4 Εφαρμογές Ασύρματων Δικτύων Αισθητήρων

Παρακολούθηση περιοχής

Η παρακολούθηση περιοχής είναι μια κοινή εφαρμογή των αισθητηριακών δικτύων. Στην παρακολούθηση περιοχής, το ασύρματο δίκτυο αισθητήρων έχει αναπτυχθεί σε μια περιοχή όπου κάποιο φαινόμενο πρέπει να παρακολουθηθεί. Ένα παράδειγμα από τον στρατό είναι η χρήση των αισθητήρων ώστε να ανιχνευθεί η εχθρική εισβολή. Ένα πολιτικό παράδειγμα είναι η γεωπερίφραξη του φυσικού αερίου ή στους αγωγούς πετρελαίου .

Περιβαλλοντική / γεωσκόπηση

Ο όρος Περιβαλλοντικά Δίκτυα Αισθητήρων έχει εξελιχθεί για να καλύψει πολλές εφαρμογές των ασύρματων δικτύων αισθητήρων για την έρευνα της γεωλογίας. Αυτό περιλαμβάνει την παρακολούθηση με αισθητήρες ηφαιστείων, ωκεανών, παγετώνων, δασών κτλ. Ορισμένοι από τους κύριους τομείς αναφέρονται παρακάτω.

Παρακολούθηση της ποιότητας του αέρα

Ο βαθμός ρύπανσης του αέρα πρέπει να μετράται συχνά προκειμένου να προστατευθεί ο άνθρωπος και το περιβάλλον από κάθε είδους ζημιά που οφείλεται στην ατμοσφαιρική ρύπανση. Σε επικίνδυνο περιβάλλον,η παρακολούθηση των επιβλαβών αερίων σε πραγματικό χρόνο είναι μια ανυσηχητική διαδικασία γιατί ο καιρός μπορεί να αλλάξει με σοβαρές επιπτώσεις με άμεσο τρόπο. Ευτυχώς, τα ασύρματα δίκτυα αισθητήρων έχουν ξεκινήσει να παράγουν συγκεκριμένες λύσεις για τους ανθρώπους.

Εσωτερικός έλεγχος

Για την παρακολούθηση των επίπεδων του φυσικού αερίου σε ευάλωτες περιοχές απαιτείτε η χρήση εξειδικευμένου, σύγχρονου εξοπλισμόυ, ικανού να ικανοποιήσει τους βιομηχανικούς κανονισμούς. Οι ασύρματες εσωτερικές λύσεις παρακολούθησης διευκολύνουν την συνεχή ενημέρωση μεγάλων περιοχών καθώς και την εξασφάλιση της ακριβούς συγκέντρωσης αερίου.

Εξωτερικός έλεγχος

Ο εξωτερικός έλεγχος της ποιότητας του αέρα χρειάζεται την χρήση ακριβών ασύρματων αισθητήρων,ανθεκτικά στην βροχή και στον άνεμο, καθώς και μεθόδους εξοικονόμισης ενέργειας για να βεβαιωθεί η επάρκεια ενέργειας στο μηχάνημα που θα έχει πιθανόν δύσκολη πρόσβαση.

Παρακολούθηση της ρύπανσης του αέρα

Ασύρματα δίκτυα αισθητήρων έχουν αναπτυχθεί σε διάφορες πόλεις (Στοκχόλμη, Λονδίνο και Μπρισμπέιν) για την παρακολούθηση της συγκέντρωσης των επικίνδυνων αερίων για τους πολίτες. Αυτά μπορούν να επωφεληθούν από τις ασύματες ζεύξεις ad-hoc και όχι από τις ενσύρματες εγκαταστάσεις που επίσης τα κάνουν πιο ευκίνητα για δοκιμαστικές μετρήσεις σε διάφορες περιοχές. Υπάρχουν διάφορες αρχιτεκτονικές που μπορούν να χρησιμοποιηθούν για τέτοιες εφαρμογές, καθώς και διάφορα είδη ανάλυσης δεδομένων και εξόρυξης δεδομένων που μπορούν να διεξαχθούν.

Ανίχνευση δασικών πυρκαγιών

Ένα δίκτυο αισθητήριων κόμβων μπορεί να εγκατασταθεί σε ένα δάσος για να ανιχνεύει πότε έχει εκδηλωθεί πυρκαγιά. Οι κόμβοι μπορούν να είναι εξοπλισμένοι με αισθητήρες για τη μέτρηση της θερμοκρασίας, την υγρασία και τα αέρια που παράγονται από φωτιά στα δέντρα ή τη βλάστηση. Η έγκαιρη ανίχνευση είναι ζωτικής σημασίας για την επιτυχή δράση των πυροσβεστών, χάρη στα ασύρματα δίκτυα αισθητήρων, η πυροσβεστική θα είναι σε θέση να γνωρίζει πότε μια πυρκαγιά ξεκίνησε και πώς εξαπλώνεται.

Ανίχνευση κατολισθήσεων

Ένα σύστημα ανίχνευσης κατολίσθησης κάνει χρήση ενός ασύρματου δικτύου αισθητήρων για να ανιχνεύσει τις μικρές κινήσεις του εδάφους και αλλαγές στις διάφορες παραμέτρους που μπορεί να συμβούν πριν ή κατά τη διάρκεια μιας κατολίσθησης. Μέσα από τα δεδομένα που συλλέγονται μπορεί να είναι δυνατόν να γνωρίζουμε την εμφάνιση των κατολισθήσεων πολύ πριν αυτό συμβεί στην πραγματικότητα.

Παρακολούθηση της ποιότητας των υδάτων

Η παρακολούθηση της ποιότητας του νερού περιλαμβάνει την ανάλυση των ιδιοτήτων του νερού σεφράγματα, ποτάμια, λίμνες και ωκεανούς, καθώς και τα υπόγεια αποθέματα νερού.Η χρήση πολλών ασύρματων αισθητήρων που διανέμοντε επιτρέπει τη δημιουργία μιας πιο ακριβούς εικόνας της κατάστασης των υδάτων, και επιτρέπει τη μόνιμη εγκατάσταση σταθμών παρακολούθησης σε περιοχές με δύσκολη πρόσβαση, χωρίς την ανάγκη του εγχειριδίου ανάκτησης δεδομένων.

Πρόληψη φυσικών καταστροφών

Τα ασύρματα δίκτυα αισθητήρων μπορούν να ενεργήσουν αποτελεσματικά για να αποτραπούν οι συνέπειες των φυσικών καταστροφών, όπως οι πλημμύρες. Ασύρματοι κόμβοι έχουν αναπτυχθεί με επιτυχία σε ποτάμια όπου οι μεταβολές της στάθμης του νερού θα πρέπει να παρακολουθείτε σε πραγματικό χρόνο.

Βιομηχανική παρακολούθηση

Ασύρματα δίκτυα αισθητήρων έχουν αναπτυχθεί για την βασική συντήρηση των μηχανημάτων (Condition-Based Maintenance - CBM), δεδομένου ότι προσφέρουν σημαντική εξοικονόμηση κόστους και επιτρέπουν νέες λειτουργίες. Σε ενσύρματα συστήματα, η εγκατάσταση των αισθητήρων συχνά περιορίζεται από το κόστος της καλωδίωσης. Προηγουμένως απρόσιτες περιοχές, περιστρεφόμενα μηχανήματα, επικίνδυνες ή ζώνες περιορισμένης πρόσβασης και τα κινητά περιουσιακά στοιχεία μπορούν πλέον να επιτευχθούν με ασύρματους αισθητήρες.

Καταγραφή δεδομένων

Τα ασύρματα δίκτυα αισθητήρων χρησιμοποιούνται επίσης για τη συλλογή δεδομένων για την παρακολούθηση των περιβαλλοντικών πληροφοριών, αυτό μπορεί να είναι τόσο απλό όσο η παρακολούθηση της θερμοκρασίας σε ένα ψυγείο η περίπλοκο όσο η παρακολούθηση του επιπέδου του νερού σε δεξαμενές υπερχείλισης σε πυρηνικούς σταθμούς ηλεκτροπαραγωγής. Οι στατιστικές πληροφορίες μπορούν στη συνέχεια να χρησιμοποιηθούν για να δείξουν πώς τα συστήματα λειτουργούσαν. Το πλεονέκτημα των Ασύρματων Δικτύων Αισθητήρων έναντι των συμβατικών καταγραφέων είναι η «ζωντανή» τροφή δεδομένων που έχουν σαν δυνατότητα.

Βιομηχανική λογική και έλεγχος των αιτήσεων

Σε πρόσφατη έρευνα ένας τεράστιος αριθμός πρωτοκόλλων επικοινωνίας ασύρματων δίκτυων αισθητήρων έχουν αναπτυχθεί. Ενώ η προηγούμενη έρευνα εστιάζεται κυρίως στην ενημέρωση για την ενέργεια, πιο πρόσφατες έρευνες έχουν αρχίσει να εξετάζουν ένα ευρύτερο φάσμα θεμάτων, όπως η αξιοπιστία των ασύρματων συνδέσεων,τις δυνατότητες σε πραγματικό χρόνο, ή την ποιότητα της παρεχόμενης υπηρεσίας. Τα νέα αυτά στοιχεία θεωρούνται καταλυτικά για μελλοντικές εφαρμογές σε βιομηχανικές και εφαρμογές ελέγχου σχετικών ασύρματων εννοιών και μερική αντικατάσταση ή την ενίσχυση συμβατικών ενσύρματων δικτύων με τεχνικές WSN.

Παρακολούθηση νερού/αποβλήτων υδάτων

Η παρακολούθηση της ποιότητας και του επιπέδου του νερού περιλαμβάνει πολλές δραστηριότητες, όπως τον έλεγχο της ποιότητας των υπόγειων ή επιφανειακών υδάτων και την εξασφάλιση υποδομών ύδρευσης της χώρας, προς όφελος ανθρώπων και ζώων. Η περιοχή της παρακολούθησης της ποιότητας του νερού χρησιμοποιεί ασύρματα δίκτυα αισθητήρων και πολλοί κατασκευαστές έχουν ξεκινήσει νέες και προηγμένες εφαρμογές για το σκοπό αυτό.

Παρατήρηση της ποιότητας των υδάτων: Η όλη διαδικασία περιλαμβάνει την εξέταση των ιδιοτήτων του νερού σε φράγματα, ποτάμια, ωκεανούς, λίμνες και στους υπόγειους υδάτινους πόρους. Ασύρματοι αισθητήρες που διαχέονται στο νερό επιτρέπουν στους χρήστες να κάνουν έναν ακριβή χάρτη της κατάστασης των υδάτων καθώς και τη μόνιμη κατανομή των σταθμών παρατήρησης σε περιοχές με δύσκολη πρόσβαση χωρίς χειρωνακτική ανάκτηση των δεδομένων.

Διαχείριση του δικτύου διανομής των υδάτων: Οι κατασκευαστές των αισθητήρων του δικτύου διανομής νερού επικεντρώνονται στην παρατήρηση των δομών διαχείρισης των υδάτων, όπως βαλβίδες και σωληνώσεις, αλλά και στην απομακρυσμένη πρόσβαση σε μετρητές νερού.

Πρόληψη των φυσικών καταστροφών: Οι συνέπειες των φυσικών κινδύνων, όπως οι πλημμύρες μπορεί να προληφθούν αποτελεσματικά με τα Ασύρματα Δίκτυα Αισθητήρων. Οι ασύρματοι κόμβοι κατανέμονται σε ποτάμια, έτσι ώστε οι αλλαγές της στάθμης του νερού να μπορεί να ελέγχεται αποτελεσματικά.

Γεωργία

Η χρήση Ασύρματων Δικτύων Αισθητήρων στο πλαίσιο του γεωργικού κλάδου είναι όλο και περισσότερο κοινή. Η χρήση ένος ασύρματου δίκτυου απαλλάσσει τους αγρότες από τη διατήρηση της καλωδίωσης σε ένα δύσκολο περιβάλλον. Συστήματα νερού, τροφοδοσίας, βαρύτητας, μπορούν να παρακολουθούνται χρησιμοποιώντας πομπούς πίεσης για να παρακολουθούν τα επίπεδα μιας δεξαμενής νερού, αντλίες μπορούν να ελέγχονται με τη χρήση ασύρματων I/O συσκευών και η χρήση του νερού μπορεί να μετρηθεί και να μεταδίδεται ασύρματα σε ένα κεντρικό σημείο ελέγχου για τιμολόγηση. Ο αυτοματισμός άρδευσης επιτρέπει την πιο αποτελεσματική χρήση του νερού και μειώνει τα απόβλητα.

Ακριβής γεωργία: Τα Ασύρματα Δίκτυα Αισθητήρων επιτρέπουν στους χρήστες να κάνουν ακριβή παρακολούθηση της καλλιέργειας κατά το χρόνο της ανάπτυξής της. Ως εκ τούτου, οι αγρότες μπορούν να γνωρίζουν άμεσα την κατάσταση του αντικειμένου σε όλα τα στάδια του, κάτι το οποίο θα διευκολύνει τη διαδικασία λήψης απόφασης σχετικά με το χρόνο της συγκομιδής.

Διαχείριση της άρδευσης: Όταν παραδίδονται τα δεδομένα σε πραγματικό χρόνο, οι αγρότες είναι σε θέση να επιτύχουν έξυπνη άρδευση. Τα στοιχεία που αφορούν τα πεδία, όπως η θερμοκρασία και το επίπεδο υγρασίας του εδάφους παραδίδονται στους αγρότες μέσω των Ασύρματων Δικτύων Αισθητήρων. Όταν κάθε φυτό ενώνεται με ένα προσωπικό σύστημα άρδευσης, οι αγρότες μπορούν να παρέχουν το ακριβές ποσό του νερού που χρειάζεται κάθε φυτό και ως εκ τούτου, να επιτύχουν τη μείωση του κόστους και την βελτίωση της ποιότητας του τελικού προϊόντος. Τα δίκτυα μπορούν να χρησιμοποιηθούν για τη διαχείριση των διαφόρων ενεργοποιητών στα συστήματα χρησιμοποιώντας μη ενσύρματη υποδομή.

Θερμοκήπια: Τα Ασύρματα Δίκτυα Αισθητήρων μπορούν επίσης να χρησιμοποιηθούν για να ελέγχουν τα επίπεδα θερμοκρασίας και υγρασίας στο εσωτερικό εμπορικών θερμοκηπίων. Όταν η θερμοκρασία και η υγρασία πέφτει κάτω από συγκεκριμένα επίπεδα, ο διαχειριστής του θερμοκηπίου πρέπει να ειδοποιείται μέσω e-mail ή στο κινητό τηλέφωνο με μήνυμα κειμένου, ή τα συστήματα υποδοχής μπορεί να πυροδοτήσουν τα συστήματα υδρονέφωσης, να ανοίξουν τους αεραγωγούς, να ενεργοποιήσουν τις περσίδες, ή να έλεγξουν μια ευρεία ποικιλία αντιδράσεων του συστήματος. Πρόσφατες έρευνες σε Ασύρματα Δίκτυα Αισθητήρων στη βιομηχανία γεωργίας δίνουν έμφαση στη χρήση της σε θερμοκήπια, ιδιαίτερα για τις μεγάλες εκμεταλλεύσεις με συγκεκριμένες καλλιέργειες. Τέτοια μικροκλίματα έχουν ανάγκη την διατήρηση ακριβούς κατάστασης καιρικών συνθήκων ανά πάσα στιγμή. Επιπλέον, με τη χρήση πολλαπλών κατανεμημένων αισθητήρων θα ελέγχεται καλύτερα η παραπάνω διαδικασία, σε ανοικτή επιφάνεια, καθώς και στο έδαφος.

Παθητικός εντοπισμός και παρακολούθηση

Η εφαρμογή των WSN στον παθητικό εντοπισμό και την παρακολούθηση των μη συνεργάσιμων στόχων (δηλαδή, άτομα που δεν φορούν οποιοδήποτε ταμπέλα) έχει προταθεί από τη διάχυτη αξιοποίηση χαμηλού κόστους φύσης της εν λόγω τεχνολογίας και τις ιδιότητες των ασύρματων ζεύξεων που είναι εγκατεστημένα σε ένα δίκτυο υποδομής WSN.

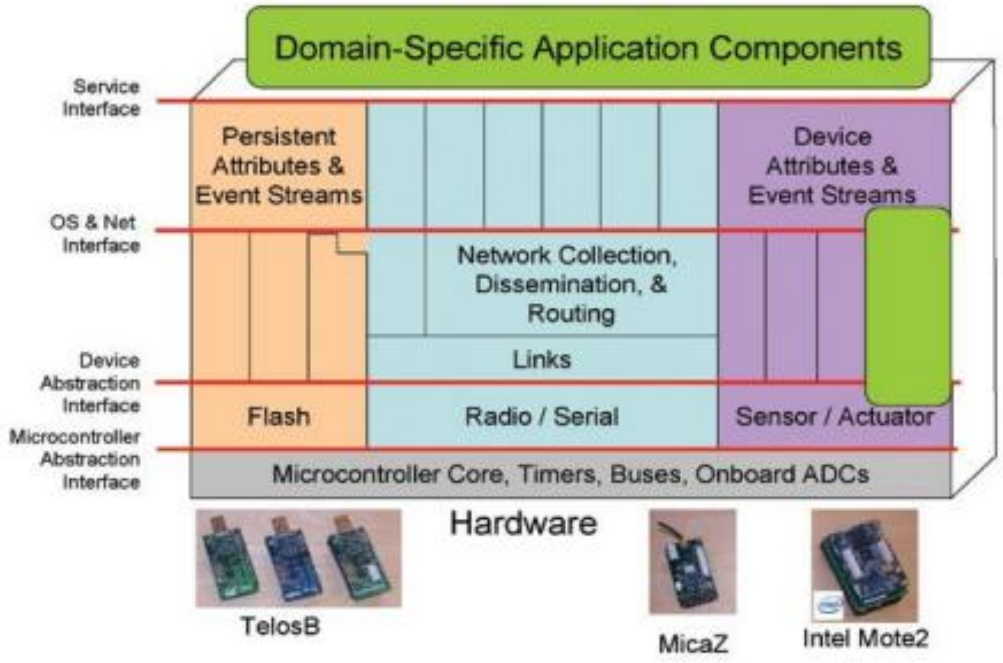
Παρακολούθηση έξυπνου σπιτιού

Παρακολούθηση των δραστηριοτήτων που εκτελούνται σε ένα έξυπνο σπίτι επιτυγχάνονται με τη χρήση ασύρματων αισθητήρων, ενσωματωμένων σε αντικείμενα καθημερινής χρήσης, σχηματίζοντας ένα

WSN. Όταν η κατάσταση αλλάζει σε αντικείμενα που βασίζονται στην ανθρώπινη χειραγώγηση συλλαμβάνεται από το ασύρματο δίκτυο αισθητήρων που επιτρέπουν τη δραστηριότητα σε υπηρεσίες υποστήριξης. [3]

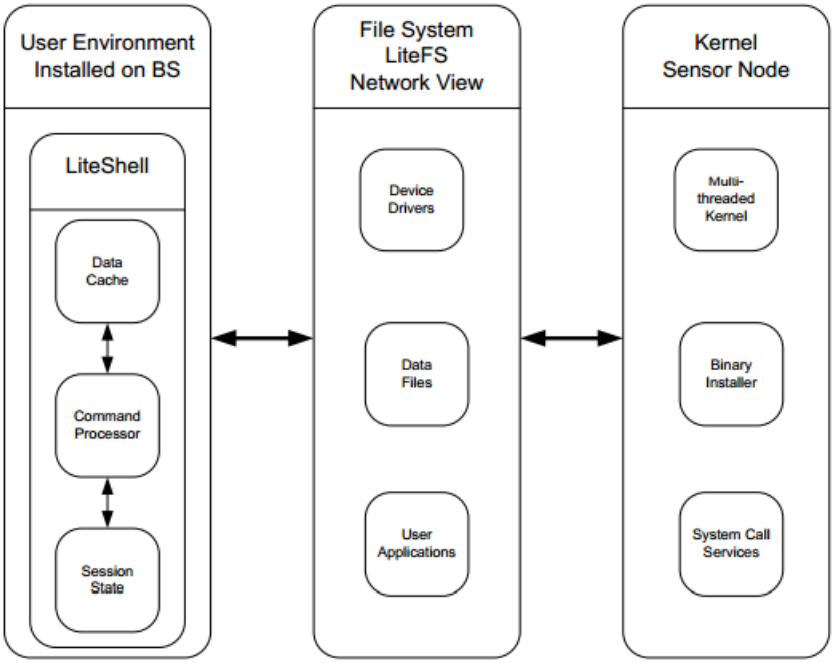
3.2.5 Λειτουργικά Συστήματα

Το **TinyOS** είναι ένα ευέλικτο, ελεύθερο λογισμικό ανοικτού κώδικα το οποίο είναι βασισμένο σε components. Το TinyOS μπορεί να υποστηρίξει ταυτόχρονα προγράμματα με πολύ χαμηλές απαιτήσεις μνήμης. Η βιβλιοθήκη των components του TinyOS περιλαμβάνει πρωτόκολλα δικτύου, κατανεμημένες υπηρεσίες, drivers αισθητήρων και εργαλεία απόκτησης δεδομένων. [12] Οι εφαρμογές του TinyOS είναι γραμμένες στη γλώσσα προγραμματισμού NesC, μία παραλλαγή της C προσαρμοσμένη στους περιορισμούς μνήμης των δικτύων αισθητήρων. [13]



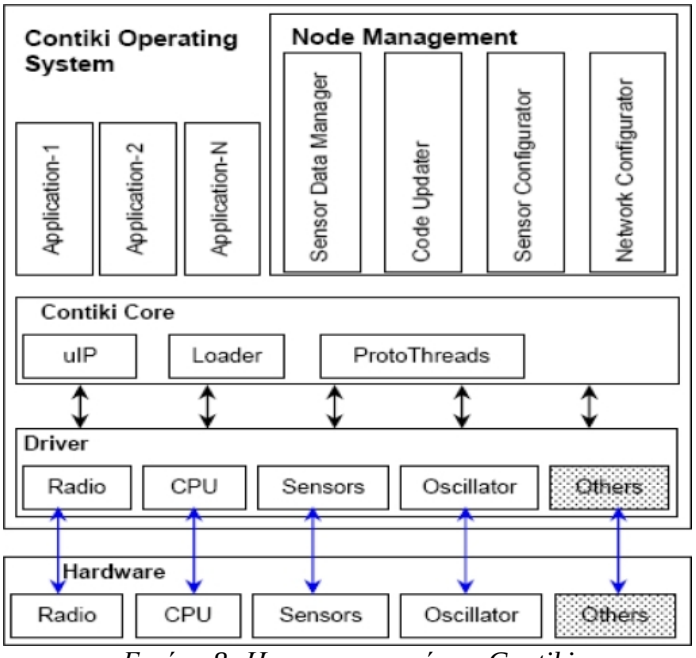
Εικόνα 6: Η αρχιτεκτονική του TinyOS

Το **LiteOS** είναι ένα real-time, βασισμένο στο UNIX, ανοιχτού κώδικα λειτουργικό σύστημα από το Πανεπιστήμιο του Illinois. Επιτρέπει στους χρήστες να χειρίζονται τα ασύρματα δίκτυα αισθητήρων όπως χειρίζονται το UNIX. Το προγραμματιστικό περιβάλλον του LiteOS είναι βασισμένο σε UNIX, threads και C. Επιτρέπει event-driven και thread-driven προγραμματισμό. [14]



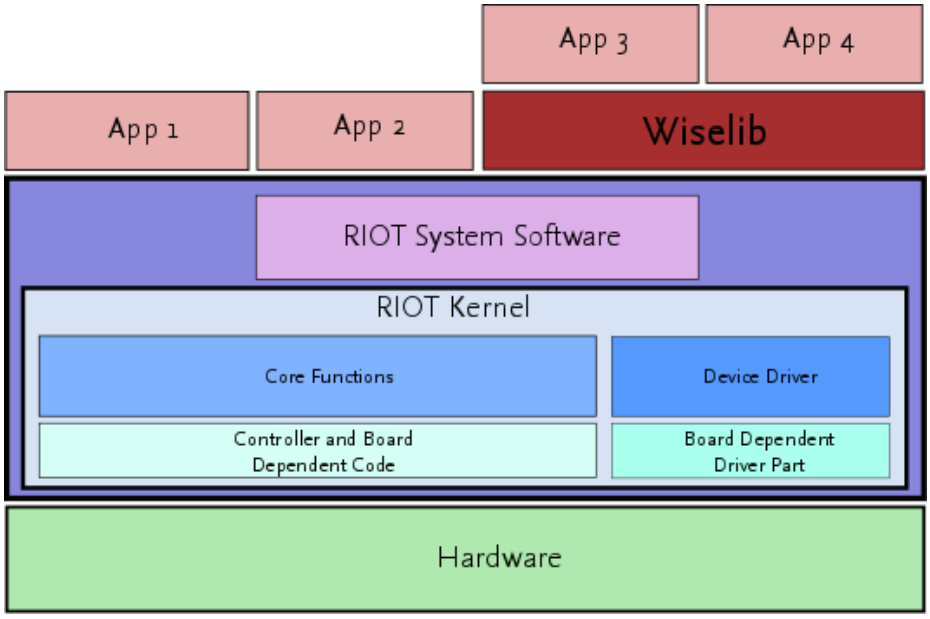
Εικόνα 7: Η αρχιτεκτονική του LiteOS

Το **Contiki** είναι ένα ελαφρύ λειτουργικό σύστημα ανοιχτού κώδικα. Είναι ένα πολύ φορητό λειτουργικό και είναι σχεδιασμένο γύρω από ένα event-driven kernel. Το Contiki παρέχει multitasking το οποίο μπορεί να χρησιμοποιηθεί σε επίπεδο διεργασιών. Μία πλήρης εγκατάσταση του Contiki περιλαμβάνει χαρακτηριστικά όπως multitasking kernel, multithreading, TCP/IP δικτύωση, IPv6, γραφικό περιβάλλον χρήστη, έναν web browser, έναν προσωπικό web server, έναν απλό telnet client, κ.α. [12]



Εικόνα 8: Η αρχιτεκτονική του Contiki

Το **RIOT** είναι ένα λειτουργικό σύστημα βασισμένο σε μια microkernel αρχιτεκτονική. Το kernel του RIOT προέρχεται κυρίως από το FireKernel το οποίο είχε αναπτυχθεί αρχικά για δίκτυα αισθητήρων. Το RIOT επιτρέπει προγραμματισμό εφαρμογών σε γλώσσες C και C++, και σε αντίθεση με άλλα λειτουργικά συστήματα με παρόμοιες απαιτήσεις μνήμης, το RIOT παρέχει πλήρες multithreading και real-time δυνατότητες μαζί. [15]



Εικόνα 9: Η αρχιτεκτονική του RIOT

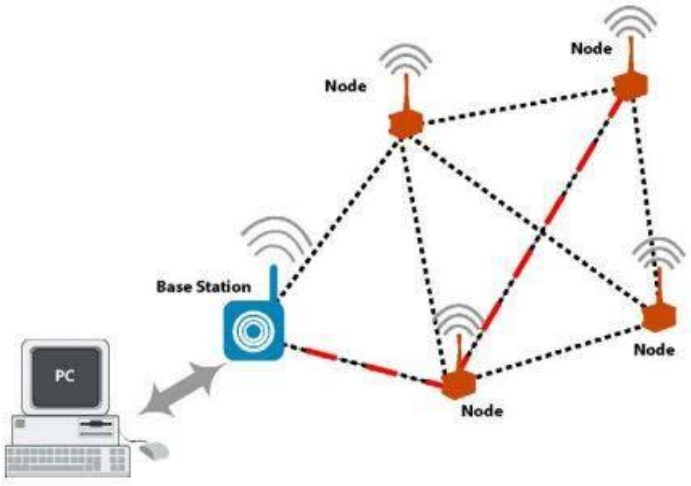
3.2.6 XMesh

Στην παρούσα πτυχιακή χρησιμοποιούμε το XMesh το οποίο είναι ένα πλήρες multi-hop, ad-hoc, πρωτόκολλο πλέγματος που αναπτύχθηκε από την Crossbow για ασύρματα δίκτυα αισθητήρων. Ένα δίκτυο XMesh αποτελείται από κόμβους (Motes) που επικοινωνούν ασύρματα μεταξύ τους και είναι σε θέση να μεταφέρουν μηνύματα σε έναν σταθμό βάσης που είναι συνδεδεμένος σε έναν υπολογιστή .Η μεταφορά από τον έναν κόμβο στον άλλον επεκτείνει αποτελεσματικά το φάσμα της επικοινωνίας και μειώνει την ισχύ που απαιτείται για τη μετάδοση μηνυμάτων.

Με μεταφορά δεδομένων με τον τρόπο αυτό, το XMesh μπορεί να παρέχει δύο κρίσιμα πλεονεκτήματα: βελτιωμένη κάλυψη περιοχής και βελτιωμένη αξιοπιστία. Δύο κόμβοι δεν πρέπει να είναι εντός της περιοχής εμβέλειας του άλλου για να επικοινωνήσουν. Ένα μήνυμα μπορεί να παραδοθεί σε έναν ή περισσότερους κόμβους μεταξύ των οποίων θα δρομολογήσει τα δεδομένα. Ομοίως, αν υπάρχει μια κακή σύνδεση μεταξύ δύο κόμβων το εμπόδιο αυτό μπορεί να ξεπεραστεί με την αλλαγή δρομολογίου γύρω από την περιοχή της κακής εξυπηρέτησης Συνήθως οι κόμβοι λειτουργούν σε κατάσταση χαμηλής κατανάλωσης ενέργειας, περνούν τον περισσότερο χρόνο τους σε μια κατάσταση ύπνου, προκειμένου να επιτευχθεί διάρκεια ζωής μπαταρίας ενός έτους.

Το XMesh παρέχει την υπηρεσία δικτύωσης TrueMesh που τα χαρακτηριστικά της είναι η αυτό-οργάνωση και αυτό-ίαση του δικτύου. Το XMesh μπορεί να μεταφέρει τα δεδομένα της διαδρομής από τους κόμβους σε ένα σταθμό βάσης (upstream) ή σε ανεξάρτητους κόμβους(downstream). Μπορεί επίσης να μεταδοθεί μέσα σε έναν ενιαίο χώρο κάλυψης ή αυθαίρετα ανάμεσα σε δύο κόμβους ενός συμπλέγματος. QOS (Quality of Service) παρέχεται είτε από best-efford delivery είτε από guaranteed delivery (end-to-end

αναγνώριση). Επίσης, το XMesh μπορεί να ρυθμιστεί σε διάφορες λειτουργίες κατανάλωσης ενέργειας συμπεριλαμβανομένης της HP (High Power), LP (Low Power), και ELP (Extended Low Power). [16]

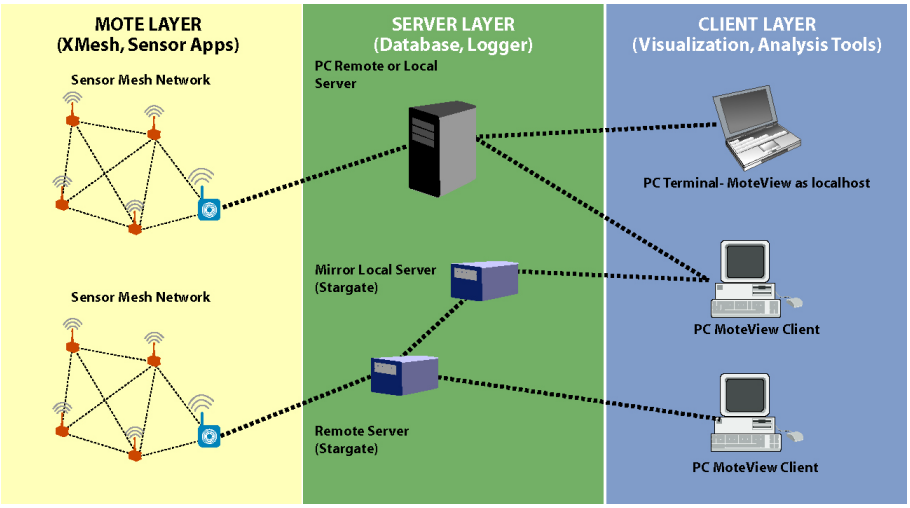


Εικόνα 10: Δίκτυο XMesh

3.2.7 XServe

Το XServe είναι η κύρια πύλη μεταξύ των mesh ασύρματων δικτύων και των εφαρμογών που αλληλεπιδρούν με αυτό. Στον πυρήνα του, το XServe παρέχει υπηρεσίες για δρομολόγηση δεδομένων από και προς το δίκτυο μαζί με υψηλότερου επιπέδου υπηρεσίες για προσπέλαση και επεξεργασία δεδομένων ενώ περνούν από το δίκτυο σε εξωτερικές εφαρμογές.

Το XServe προσφέρει διάφορες υπηρεσίες ανάλογα με το πώς είναι εγκατεστημένο και ρυθμισμένο: Σειριακή προώθηση: Επιτρέπει στις εφαρμογές να επικοινωνούν απευθείας με την εφαρμογή. Οι εφαρμογές στέλνουν και λαμβάνουν raw δεδομένα απευθείας στο δίκτυο χωρίς προσπέλαση, μετατροπή ή επεξεργασία υψηλού επιπέδου. Εφαρμογές XServe εγκατεστημένες σε πολλαπλές συσκευές μπορούν να δημιουργήσουν μία αλυσίδα δρομολόγησης προωθώντας δεδομένα από το Mote tier σε ένα δίκτυο.
Server εφαρμογών: Το XServe μπορεί να λειτουργήσει ως ένας server εφαρμογών προσφέροντας υψηλού επιπέδου υπηρεσίες όπως προσπέλαση, μετατροπή και επεξεργασία δεδομένων σε τρέχοντα χρόνο. [17]



Εικόνα 11: Το Software Framework ενός Ασύρματου Δικτύου Αισθητήρων

3.3 Android

3.3.1 Γενικά

Το Android είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα που χρησιμοποιείται σε κινητές συσκευές το οποίο έχει αναπτυχθεί από την Google. Ο πυρήνας του είναι βασισμένος πάνω στα Linux. Είναι σχεδιασμένο για οθόνη αφής και διαθέτει εικονικό πληκτρολόγιο. Πλέον εκτός από τα κινητά τηλέφωνα και τις ταμπλέτες εξελίσσεται και χρησιμοποιείται και σε άλλες συσκευές όπως τηλεοράσεις (Android TV), αυτοκίνητα (Auto Android), και τα ρολόγια χειρός (Android Wear). Το Android καταλαβαίνει ως εισόδους επιλογής το απλό πάτημα ,το παρατεταμένο πάτημα και το σύρσιμο του δάκτυλου πάνω στην οθόνη. Ακόμα και αν το Android είναι σχεδιασμένο για συσκευές αφής σήμερα βρίσκει εφαρμογές και σε παιχνιδοκονσόλες, ψηφιακές φωτογραφικές μηχανές και άλλες ηλεκτρονικές συσκευές.

Εφαρμογές (apps), που επεκτείνουν τη λειτουργικότητα των συσκευών αναπτύσσονται κυρίως στη γλώσσα προγραμματισμού Java, χρησιμοποιώντας το Android Software Development Kit (SDK). Το SDK περιλαμβάνει ένα ολοκληρωμένο σύνολο εργαλείων ανάπτυξης συμπεριλαμβανομένου ενός προγράμματος εντοπισμού σφαλμάτων, βιβλιοθήκες λογισμικού, ένα εξομοιωτή συσκευής που βασίζεται στο QEMU, δείγματα κώδικα, και tutorials. Το περιβάλλον ανάπτυξης (IDE) είναι το Eclipse χρησιμοποιώντας το plugin Android Development Tools (ADT). Άλλα εργαλεία ανάπτυξης είναι διαθέσιμα, συμπεριλαμβανομένου του Native Development Kit για εφαρμογές ή επεκτάσεις σε C ή C ++, το Google App Inventor και διάφορα άλλα. [18]

3.3.2 Αρχιτεκτονική Android

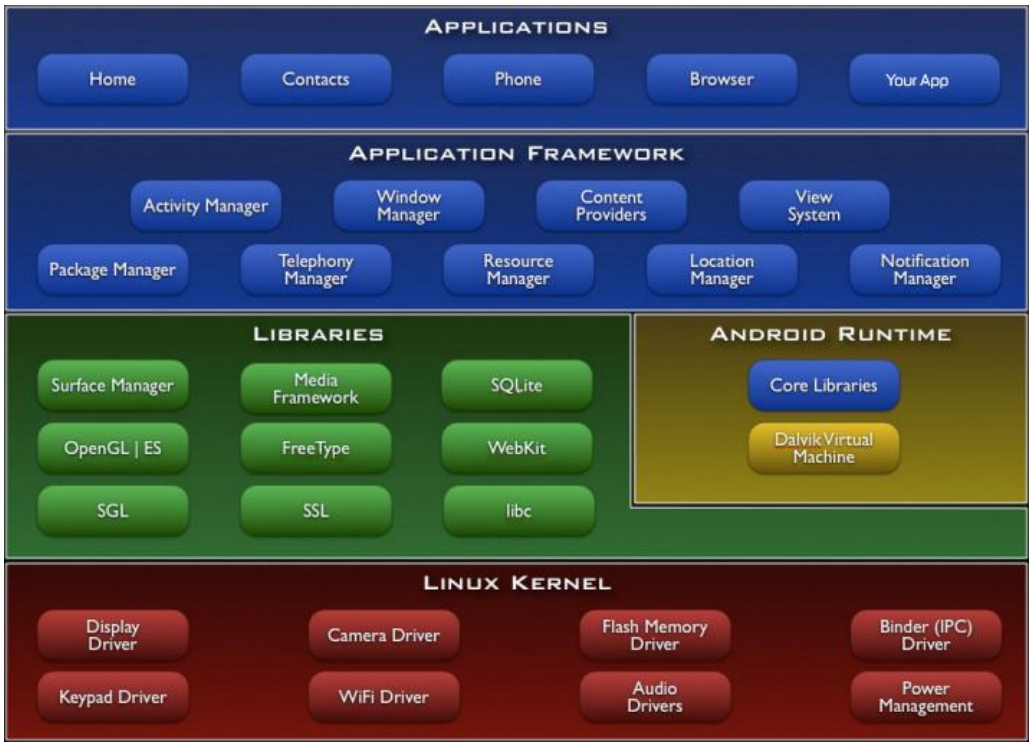
Ο πυρήνας του λειτουργικού συστήματος είναι βασισμένος σε Linux. Στην κορυφή του πυρήνα του Linux, υπάρχουν τα middleware, οι βιβλιοθήκες και τα APIs γραμμένα σε C. Το λογισμικό εφαρμογής που εκτελείται σε ένα framework εφαρμογής περιλαμβάνει βιβλιοθήκες συμβατες με Java και βασίζεται στον Apache Harmony.

Η πρότυπη βιβλιοθήκη C του Android, η Bionic, αναπτύχθηκε από την Google ειδικά για το Android. Στο Linux πυρήνα εκτελούνται οι βασικές διεργασίες της συσκευής όπως οι drivers, η διαχείριση διεργασιών και η δικτύωση.

Στο επόμενο επίπεδο βρίσκονται οι βιβλιοθήκες του λειτουργικού όπως η SQLite , η Media Framework και άλλες βασικές βιβλιοθήκες. Ακόμα βρίσκεται και το Android Runtime που διαθέτει τις βιβλιοθήκες για την εκτέλεση εφαρμογών Java και το Virtual Machine που χρησιμοποιείται για την δημιουργία των εκτελέσιμων αρχείων που θα τρέξει το λειτουργικό.

Στο επόμενο επίπεδο έχουμε το Application Framework που δίνει τη δυνατότητα στις εφαρμογές να έχουν άμεση πρόσβαση στις βασικές βιβλιοθήκες του λειτουργικού. Και μέσω αυτού μπορούν να παρέχουν περισσότερες λειτουργίες σε άλλες εφαρμογές όπως πρόσβαση σε δεδομένα άλλων εφαρμογών, πρόσβαση σε πόρους όπως γραφικά, διαχείριση κύκλου ζωής εφαρμογών και άλλα .

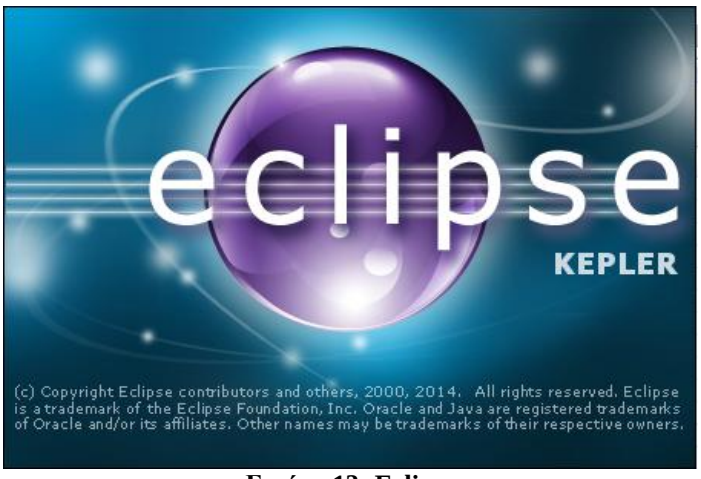
Τέλος, το τελευταίο επίπεδο είναι οι εφαρμογές που χρησιμοποιούν οι χρήστες και είναι το μόνο κομμάτι με το οποίο έχει αλληλεπίδραση ο χρήστης. Τέτοιες εφαρμογές είναι το τηλέφωνο, ο browser, το GPS, τα γραπτά μηνύματα και άλλα. [18]



Εικόνα 12: Αρχιτεκτονική Android

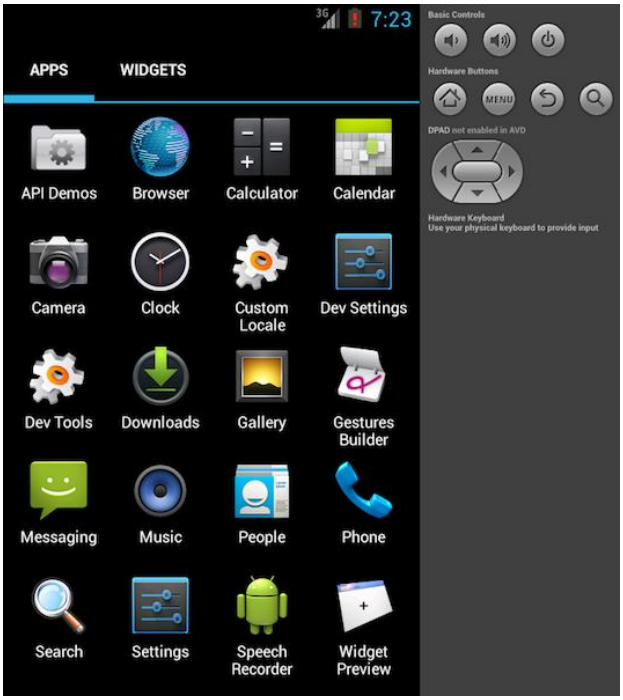
3.3.3 Eclipse IDE

Το Eclipse είναι μια πλατφόρμα ελεύθερου λογισμικού που χρησιμοποιείται ως Compiler για διάφορες γλώσσες προγραμματισμού. Είναι συμβατό με τις περισσότερες και πιο γνωστές από αυτές όπως C, C++, Java, Python, nesC, Cobol, Ruby, Perl, PHP και άλλες. Είναι γραμμένο σε Java και είναι συμβατό με όλα τα λειτουργικά συστήματα. Μέσω του κατάλληλου plugin με το Android SDK μπορεί ο οποιοσδήποτε να δημιουργήσει εφαρμογές. Για το γραφικό κομμάτι της εφαρμογής χρησιμοποιείται η γλωσσά XML και για το δυναμικό η Java. [19]



Εικόνα 13: Eclipse

Το Eclipse με το Android SDK μας παρέχουν κάποια εργαλεία όπως τη δημιουργία μιας Virtual συσκευής για να τρέχουμε τις εφαρμογές και του Android SDK Manager για να εγκαθιστούμε τις ενημερώσεις του SDK.



Εικόνα 14: Android Emulator

3.3.4 JAVA

Η java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε από την Sun Microsystems. Ένα από τα βασικά πλεονεκτήματα της Java έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία του λειτουργικού συστήματος και πλατφόρμας. Τα προγράμματα που είναι γραμμένα σε Java τρέχουν ακριβώς το ίδιο σε όλα τα γνωστά λειτουργικά συστήματα χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα. Για να επιτευχθεί όμως αυτό χρειαζόταν κάποιος τρόπος έτσι ώστε τα προγράμματα γραμμένα σε Java να μπορούν να είναι «κατανοητά» από κάθε υπολογιστή ανεξάρτητα του είδους επεξεργαστή αλλά και λειτουργικού συστήματος (Windows, Unix, Linux, MacOS). Ο λόγος είναι ότι κάθε κεντρική μονάδα επεξεργασίας κατανοεί διαφορετικό κώδικα μηχανής. Ο κώδικας χαμηλού επιπέδου (assembly) που μεταφράζεται και εκτελείται σε Windows είναι διαφορετικός από αυτόν που μεταφράζεται και εκτελείται σε έναν υπολογιστή Macintosh. Η λύση δόθηκε με την ανάπτυξη της Εικονικής Μηχανής(Virtual Machine).

Αφού γραφτεί κάποιο πρόγραμμα σε Java, στη συνέχεια μεταγλωτίζεται μέσω του μεταγλωττιστή javac, ο οποίος παράγει έναν αριθμό από αρχεία .class (κώδικας byte ή bytecode). Ο κώδικας byte είναι η μορφή που παίρνει ο πηγαίος κώδικας της Java όταν μεταγλωττιστεί. Όταν πρόκειται να εκτελεστεί η εφαρμογή σε ένα μηχάνημα, το Java Virtual Machine που πρέπει να είναι εγκατεστημένο σε αυτό θα αναλάβει να διαβάσει τα αρχεία .class. Στη συνέχεια τα μεταφράζει σε γλώσσα μηχανής που να υποστηρίζεται από το λειτουργικό σύστημα και τον επεξεργαστή, έτσι ώστε να εκτελεστεί (αυτό συμβαίνει με την παραδοσιακή Εικονική Μηχανή (Virtual Machine). Πιο σύγχρονες εφαρμογές της εικονικής Μηχανής μπορούν και μεταγλωττίζουν εκ των προτέρων τμήματα bytecode απευθείας σε κώδικα μηχανής (εγγενή κώδικα ή native code) με αποτέλεσμα να βελτιώνεται η ταχύτητα). Χωρίς αυτό δε θα ήταν δυνατή η εκτέλεση λογισμικού γραμμένου σε Java.

Η JVM είναι λογισμικό που εξαρτάται από την πλατφόρμα, δηλαδή για κάθε είδος λειτουργικού

συστήματος και αρχιτεκτονικής επεξεργαστή υπάρχει διαφορετική έκδοση του. Έτσι υπάρχουν διαφορετικές JVM για Windows, Linux, Unix, Macintosh, κινητά τηλέφωνα, παιχνιδομηχανές κλπ. Οτιδήποτε θέλει να κάνει ο προγραμματιστής (ή ο χρήστης) γίνεται μέσω της εικονικής μηχανής. Αυτό βοηθάει στο να υπάρχει μεγαλύτερη ασφάλεια στο σύστημα γιατί η εικονική μηχανή είναι υπεύθυνη για την επικοινωνία χρήστη - υπολογιστή. Ο προγραμματιστής δεν μπορεί να γράψει κώδικα ο οποίος θα έχει καταστροφικά αποτελέσματα για τον υπολογιστή γιατί η εικονική μηχανή θα τον ανιχνεύσει και δε θα επιτρέψει να εκτελεστεί. Από την άλλη μεριά ούτε ο χρήστης μπορεί να κατεβάσει «κακό» κώδικα από το δίκτυο και να τον εκτελέσει. Αυτό είναι ιδιαίτερα χρήσιμο για μεγάλα κατανεμημένα συστήματα όπου πολλοί χρήστες χρησιμοποιούν το ίδιο πρόγραμμα συγχρόνως. [20]

Εικόνα 15: java logo

3.3.5 XML

Η XML (Extensible Markup Language) είναι μία γλώσσα σήμανσης, που περιέχει ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων. Ορίζεται, κυρίως, στην προδιαγραφή XML 1.0 (XML 1.0 Specification), που δημιούργησε ο διεθνής οργανισμός προτύπων W3C (World Wide Web Consortium), αλλά και σε διάφορες άλλες σχετικές προδιαγραφές ανοιχτών προτύπων.

Η XML σχεδιάστηκε δίνοντας έμφαση στην απλότητα, τη γενικότητα και τη χρησιμότητα στο Διαδίκτυο. Είναι μία μορφοποίηση δεδομένων κειμένου, με ισχυρή υποστήριξη Unicode για όλες τις γλώσσες του κόσμου. Αν και η σχεδίαση της XML εστιάζει στα κείμενα, χρησιμοποιείται ευρέως για την αναπαράσταση αυθαίρετων δομών δεδομένων, που προκύπτουν για παράδειγμα στις υπηρεσίες ιστού. Υπάρχει μία ποικιλία διεπαφών προγραμματισμού εφαρμογών, που μπορούν να χρησιμοποιούν οι προγραμματιστές, για να προσπελαύνουν δεδομένα XML, αλλά και διάφορα συστήματα σχημάτων XML, τα οποία είναι σχεδιασμένα για να βοηθούν στον ορισμό γλωσσών, που προκύπτουν από την XML.

Έως το 2009, έχουν αναπτυχθεί εκατοντάδες γλώσσες που βασίζονται στην XML, συμπεριλαμβανομένων του RSS, του SOAP και της XHTML. Προεπιλεγμένες κωδικοποιήσεις βασισμένες στην XML, υπάρχουν για τις περισσότερες σουίτες εφαρμογών γραφείου, συμπεριλαμβανομένων του Microsoft Office (Office Open XML), του OpenOffice.org (OpenDocument) και του iWork της Apple. [14]

Βασικά χαρακτηριστικά της XML είναι τα εξής:

Χαρακτήρας Unicode: Εξ ορισμού, ένα κείμενο XML είναι μία ακολουθία χαρακτήρων. Σχεδόν κάθε χαρακτήρας Unicode μπορεί να εμφανίζεται σε ένα κείμενο XML.

Επεξεργαστής και Εφαρμογή: Είναι το λογισμικό που επεξεργάζεται ένα κείμενο XML. Είναι αναμενόμενο ότι ένας επεξεργαστής δουλεύει για μία εφαρμογή. Υπάρχουν μερικές πολύ συγκεκριμένες απαιτήσεις σχετικά με το τι μπορεί και τι δεν μπορεί να κάνει ένας επεξεργαστής XML, αλλά καμία, όσον αφορά στη συμπεριφορά της εφαρμογής. Ο επεξεργαστής (όπως ονοματίζεται από την προδιαγραφή), αναφέρεται συχνά με τον αγγλικό όρο XML parser.

Σήμανση και Περιεχόμενο: Οι χαρακτήρες που απαρτίζουν ένα κείμενο XML, αποτελούν είτε τη σήμανση είτε το περιεχόμενό του. Η σήμανση και το περιεχόμενο, μπορούν να επισημανθούν και να διακριθούν, ύστερα από την εφαρμογή κάποιων απλών συντακτικών κανόνων. Όλα τα αλφαριθμητικά που συνιστούν τη σήμανση, είτε ξεκινούν με το χαρακτήρα "<" και καταλήγουν στο χαρακτήρα ">", είτε ξεκινούν με το χαρακτήρα "&" και καταλήγουν στο χαρακτήρα ";". Ακολουθίες χαρακτήρων που δε συνιστούν τη σήμανση, αποτελούν το περιεχόμενο ενός κειμένου XML.

Ετικέτα: Είναι ένα στοιχείο σήμανσης που ξεκινά με το χαρακτήρα "<" και καταλήγει στο χαρακτήρα ">". Υπάρχουν τρία είδη ετικέτας: ετικέτες αρχής, ετικέτες τέλους και ετικέτες χωρίς περιεχόμενο.

Στοιχείο: Ένα λογικό απόσπασμα ενός κειμένου, που είτε ξεκινά με μία ετικέτα αρχής και καταλήγει σε μία ετικέτα τέλους, είτε αποτελείται μόνο από μία ετικέτα χωρίς περιεχόμενο. Οι χαρακτήρες που υπάρχουν, αν υπάρχουν, μεταξύ μιας ετικέτας αρχής και μιας ετικέτας τέλους, συνιστούν το περιεχόμενο του στοιχείου, το οποίο μπορεί να περιέχει σήμανση, συμπεριλαμβανομένων και άλλων στοιχείων, που ονομάζονται στοιχεία παιδιά.

Χαρακτηριστικό: Ένα στοιχείο σήμανσης που αποτελείται από ένα ζευγάρι όνομα/τιμή, το οποίο υπάρχει μέσα σε μία ετικέτα αρχής ή σε μία ετικέτα χωρίς περιεχόμενο.
Δήλωση XML: Τα κείμενα XML μπορούν να αρχίζουν, με τη δήλωση κάποιων πληροφοριών σχετικών με αυτά, όπως <?xml version="1.0" encoding="UTF-8"?>
Ένα παράδειγμα XML είναι το παρακάτω που κάνει χρήση όλων των παραπάνω εννοιών και στοιχείων:

```
<?xml version="1.0" encoding='UTF-8'?>
<painting>
  <img src="madonna.jpg" alt='Foligno Madonna, by Raphael'/>
  <caption>This is Raphael's "Foligno" Madonna, painted in
  <date>1511</date>-<date>1512</date>.</caption>
</painting>
```

[21]

3.4 Η Crossbow

Η Crossbow Technology Inc (αναφέρεται επίσης και ως XBOW) ήταν μία εταιρία με έδρα την Καλιφόρνια των Ηνωμένων Πολιτειών. Ήταν από τους πρώτους προμηθευτές των motes MICA που αρχικά αναπτύχθηκαν στο Πανεπιστήμιο Berkeley. Στη συνέχεια ανέπτυξε τις επόμενες γενιές motes που περιλάμβαναν τα MICA2, MICAZ και το IMOTE2. Η Crossbow κέρδισε βραβεία για αυτά τα προϊόντα, συμπεριλαμβανομένων των βραβείων «Best of Sensors Expo Gold 2006» και «BP Helios Award». Το 2008 κυκλοφόρησε το σύστημα eKo Pro, ένα σύστημα ασύρματων αισθητήρων για την παρακολούθηση των καλλιεργειών και του περιβάλλοντος. Οι αισθητήρες του eKo Pro μπορούν να παρακολουθούν την υγρασία του εδάφους, τη θερμοκρασία του αέρα και την ποσότητα του νερού στα φύλλα των φυτών.

Τον Ιούνιο του 2011 η Crossbow εξαγοράστηκε από την Moog Inc. Ο τομέας των ασύρματων αισθητήρων αγοράστηκε από την Memsic Inc. [22][23]

Εικόνα 16: Crossbow Logo

3.5 Σημαντικοί στόχοι για την ολοκλήρωση της πτυχιακής

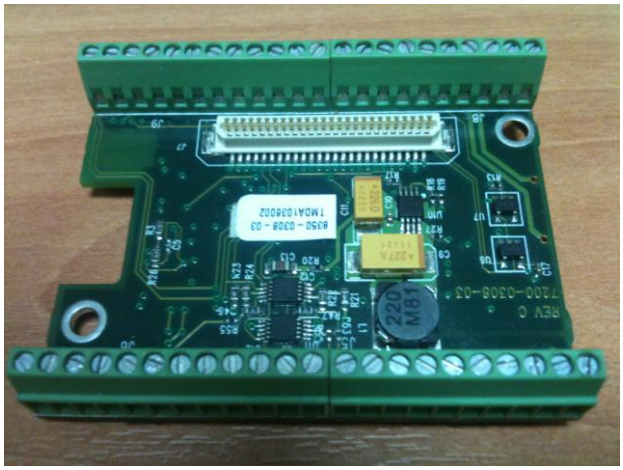
- Συνδεσμολογία δικτύου
- Μελέτη και παρακολούθηση δικτύου
- Εισαγωγή εξωτερικών αισθητήρων
- Παρακολούθηση αποτελεσμάτων μέσω MoteView
- Δημιουργία εφαρμογής Android
- Μορφοποίηση του GUI
- Σύνδεση εφαρμογής με βάση δεδομένων PostgreSQL
- Παρουσίαση αποτελεσμάτων με μορφή πίνακα
- Δημιουργία διαγραμμάτων με στοιχεία του πίνακα
- Μορφοποίηση διαγραμμάτων

4 Κύριο Μέρος Πτυχιακής

4.1 Εξοπλισμός

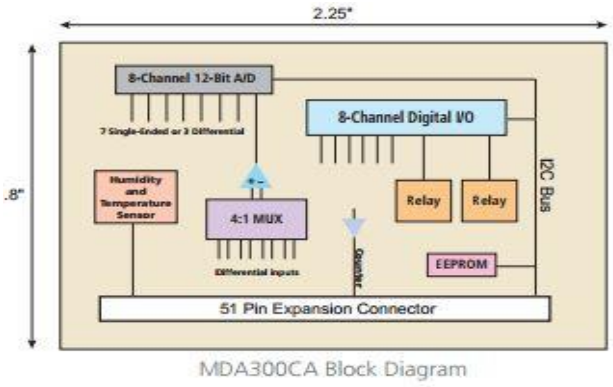
4.1.1 Crossbow MDA300CA

Το MDA300 είναι μια πλακέτα της Crossbow που έχει δυο on board αισθητήρες υγρασίας και θερμοκρασίας καθώς και την δυνατότητα προσαρμογής εξωτερικών αισθητήρων, αναλογικών και ψηφιακών. Επίσης μπορεί να χρησιμοποιηθεί για απομακρυσμένο έλεγχο συσκευών καθώς διαθέτει δυο κανάλια ρελέ. Οι εφαρμογές που υποστηρίζει είναι συλλογή περιβαλλοντικών δεδομένων, παρακολούθηση αγροτικών εφαρμογών, τηλεϊατρική, απομακρυσμένη διαχείριση συσκευών και δυνατότητα προσαρμογής σε πολλά ακόμα project. Διαθέτει low-power mode και προσφέρει αυτονομία με άπλες μπαταρίες για μακρά χρονική περίοδο.



Εικόνα 17: MDA300CA (1)

Εικόνα 18: MDA300CA (2)

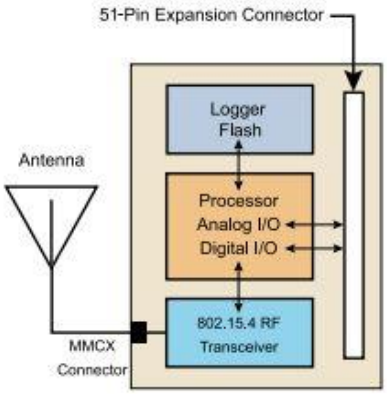


Εικόνα 19: MDA300CA Block Diagram

4.1.2 Crossbow MICAZ

Το MICAZ είναι μια πλακέτα της Crossbow που είναι υπεύθυνη για την επικοινωνία των κόμβων του ασυρμάτου δικτύου αισθητήρων. Μεταδίδει τα δεδομένα στη συχνότητα των 2.4 GH και είναι σύμφωνη με το πρότυπο επικοινωνίας IEEE 802.15.4. Επίσης προσφέρει 250 kbps ως ανώτερη ταχύτητα μεταφορά δεδομένων. [24]

Εικόνα 20: MICAZ

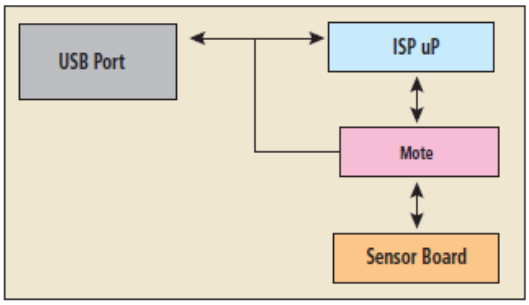


Εικόνα 21: MICAZ Block Diagram

4.1.3 Crossbow MIB520CB

Η πλακέτα της crossbow MIB520CB λειτουργεί ως η βάση ενός ασυρμάτου δικτύου αισθητήρων αφού προσφέρει την διασύνδεση με τον υπολογιστή μέσω USB. Μέσω του MIB520CB προγραμματίζονται οι πλακέτες και μαζί με το MICAZ δημιουργούν τον server του δικτύου μας που καταλήγουν οι μετρήσεις όλου του δικτύου και μεταφέρονται στο πρόγραμμα του υπολογιστή.

Εικόνα 22: MIB520CB



Εικόνα 23: MIB520CB Block Diagram

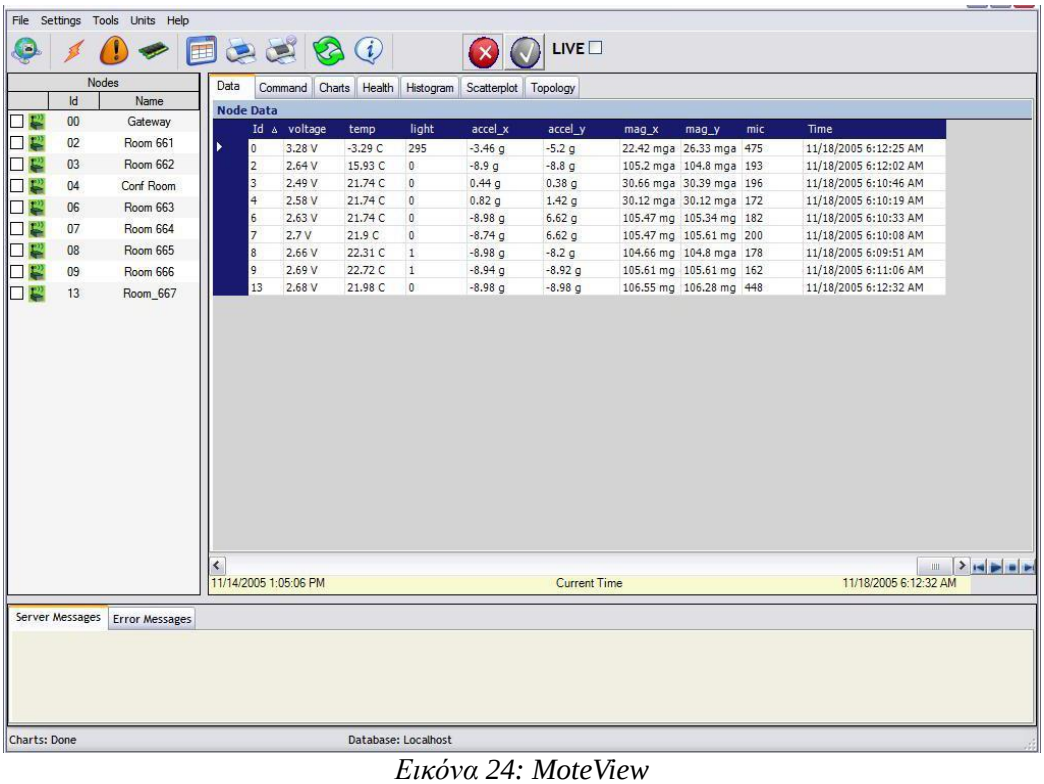
4.1.4 Ηλεκτρονικός υπολογιστής

Μπορούμε να χρησιμοποιήσουμε οποιοδήποτε ηλεκτρονικό υπολογιστή αφού το software δεν είναι βαρύ και δεν έχει μεγάλες υπολογιστικές απαιτήσεις. Εμείς χρησιμοποιήσαμε ένα laptop μάρκας Toshiba, μοντέλο Satellite A300 με μνήμη RAM 4 GB, επεξεργαστή Intel Core 2 Duo στα 2,53 GHz και λειτουργικό σύστημα Windows 7 Ultimate 64 bit.

4.2 Λογισμικό

4.2.1 Πρόγραμμα Διαχείρισης Δικτύου MoteView

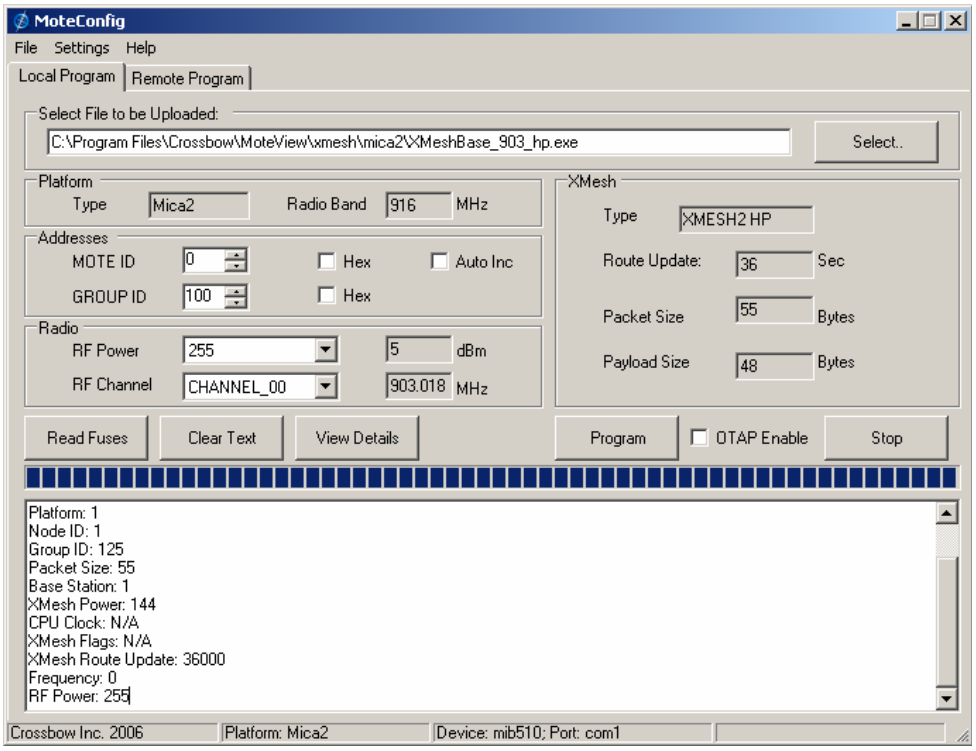
Το MoteView είναι ένα πρόγραμμα που δημιουργήθηκε από την Crossbow με σκοπό την διαχείριση του δικτύου και την αποθήκευση δεδομένων σε βάση Postgresql. Πλέον κυκλοφορεί από την Memsic. Δίνει την δυνατότητα παρουσίασης των αποτελεσμάτων των μετρήσεων σε μορφή πινάκα και διαγραμματική καθώς και προσαρμογής παραμέτρων όπως εξωτερικών αισθητήρων και παρατήρησης της τοπολογίας του δικτύου . Είναι συμβατό με όλες της πλακέτες της Crossbow.



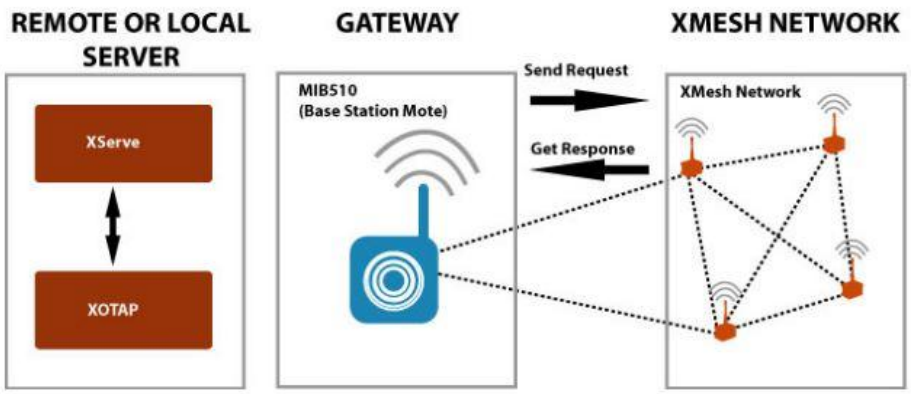
Εικόνα 24: MoteView

4.2.2 MoteConfig

Το MoteConfig είναι μέρος του MoteView και παρέχει τη δυνατότητα προγραμματισμού των motes. Ο προγραμματισμός γίνεται με έτοιμες εφαρμογές λογισμικού TinyOS οι οποίες είναι διαθέσιμες με την εγκατάσταση του MoteView. Ο προγραμματισμός μπορεί να γίνει ενσύρματα μέσω της πλακέτας MIB520 επίσης υπάρχει και ο ασύρματος προγραμματισμός μέσω μια άλλης πλακέτας – βάσης, της MIB600.



Εικόνα 25: MoteConfig



Εικόνα 26: Over The Air Programming

4.3 Υλοποίηση

4.3.1 Ασύρματο δίκτυο αισθητήρων

Στην εκτέλεση της πτυχιακής εργασίας αρχικά έπρεπε να φτιάξουμε κατάλληλα το δίκτυο έτσι ώστε να έχουμε τη σωστή συνδεσμολογία για να καταφέρουν τα motes να επικοινωνήσουν μεταξύ τους .

Εικόνα 27: MICAZ συνδεδεμένο με το MDA300CA

Εικόνα 28: MICAZ συνδεδεμένο με το MIB520CB

4.3.1.1 Προγραμματισμός των κόμβων

Για να προγραμματίσουμε τους κόμβους του δικτύου χρησιμοποιούμε το MoteConfig έτσι ώστε να περάσουμε το κατάλληλο firmware στην πλακέτα. Υπάρχουν δυο διαφορετικές κατηγορίες firmware το High Power Mode και το Low Power Mode.

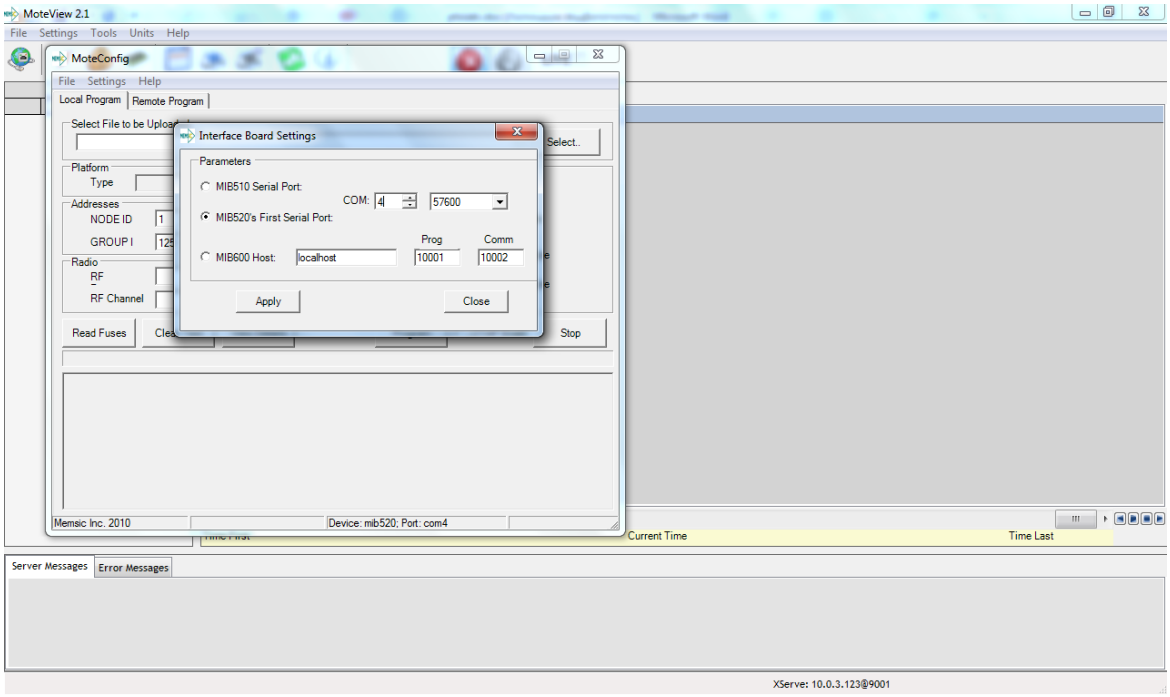
Το High Power Mode έχει τα εξής χαρακτηριστικά:

- Δυνατότητα TrueMesh
- Κάθε κόμβος του δικτύου μπορεί να δρομολογήσει δεδομένα
- Υψηλό bandwidth, χαμηλό latency (λανθάνουσα περίοδος)
- Οι κόμβοι είναι πάντα ενεργοποιημένοι

Το Low Power Mode έχει τα παρακάτω χαρακτηριστικά:

- Δυνατότητα TrueMesh
- Κάθε κόμβος του δικτύου μπορεί να δρομολογήσει δεδομένα
- Χαμηλό bandwidth, υψηλό latency (λανθάνουσα περίοδος)
- Οι κόμβοι βρίσκονται σε sleep mode και «ξυπνούν» περιοδικά για να ελέγξουν την κίνηση. [16]

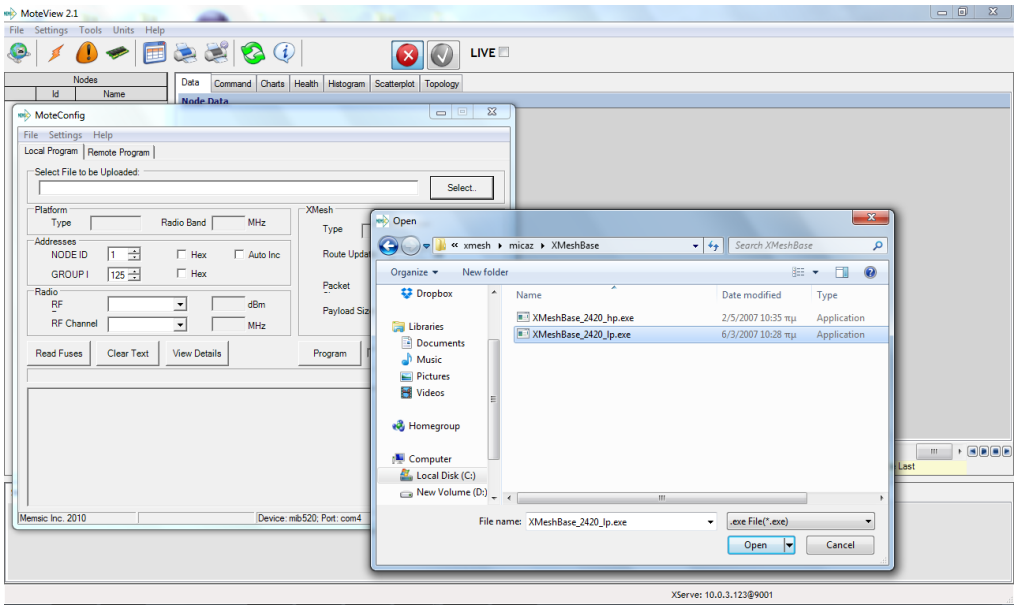
Αρχικά ανοίγουμε το MoteView και έπειτα το MoteConfig. Επιλέγουμε το Setting από το μενού για κάνουμε κάποιες ρυθμίσεις. Εφόσον χρησιμοποιούμε το MIB520 επιλέγουμε την αντίστοιχη επιλογή και επίσης στην επιλογή COM βάζουμε την μικρότερη σε αριθμό εικονική θύρα που έχουμε στον υπολογιστή μας μέσω του FTDI Driver που εγκαταστήσαμε (για παράδειγμα αν έχουμε τις θύρες COM4 COM5, COM6 και COM8 βάζουμε την COM4) και πατάμε Apply για να αποθηκευτούν οι ρυθμίσεις.



Εικόνα 29: Ρυθμίσεις

Στο κυρίως παράθυρο του MoteConfig επιλέγουμε την καρτέλα Local Program, πατάμε το Select και επιλέγουμε το πρόγραμμα που θέλουμε να περάσουμε στο εκάστοτε mote. Αρχικά προγραμματίζουμε τη βάση μας σε Low Power Mode επιλέγοντας το κατάλληλο πρόγραμμα. Το NODE ID της βάσης πρέπει να

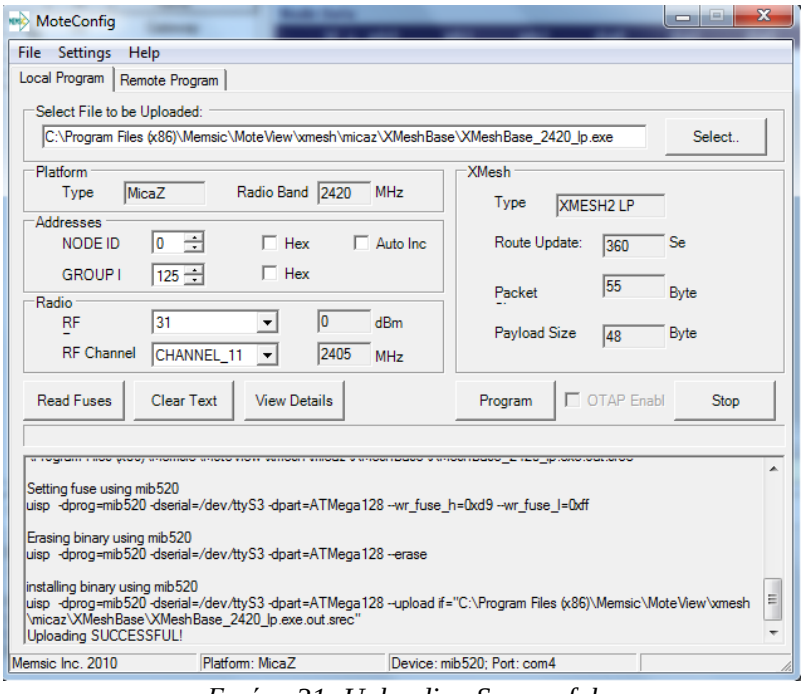
είναι πάντα 0. Αν δεν το βάλουμε εμείς μας το ρυθμίζει αυτόματα το MoteConfig κατά τον προγραμματισμό.



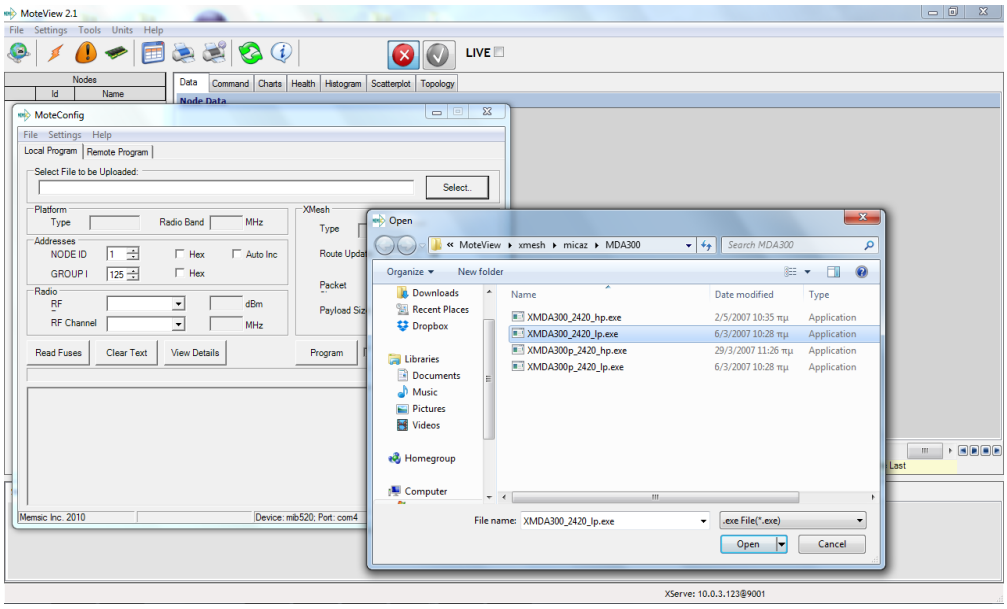
Εικόνα 30: Προγραμματισμός της βάσης

Πατάμε το Program και περιμένουμε μέχρι να εμφανιστεί το μήνυμα «UPLOADING SUCCESSFUL». Σε περίπτωση που εμφανιστεί το μήνυμα «Can not connect to Mote correctly. Please check your Mote» σημαίνει ότι υπάρχει κάποιο πρόβλημα με τη συνδεσμολογία και ελέγχουμε αν είναι όλα συνδεμένα σωστά.

Έπειτα προγραμματίζουμε τα ασύρματα node επιλέγοντας πάλι το κατάλληλο πρόγραμμα και ακολουθώντας την ίδια διαδικασία για το καθένα. Το NODE ID οποιουδήποτε άλλου κόμβου εκτός της βάσης πρέπει να είναι διαφορετικό του 0. Εμείς βάλαμε ως NODE ID 3 και 5 αντίστοιχα για κάθε node.



Εικόνα 31: Uploading Successful

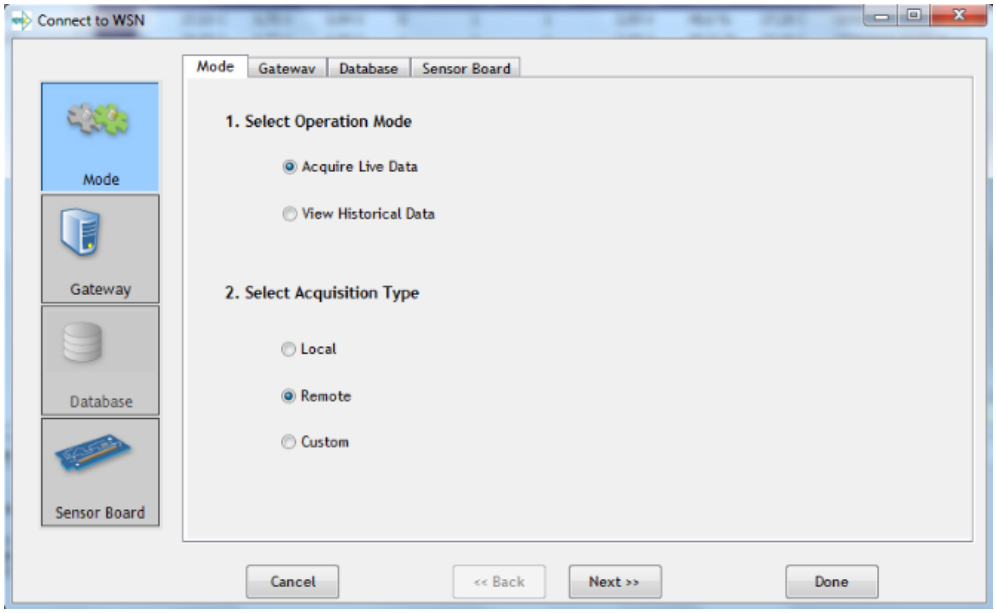


Εικόνα 32: Προγραμματισμός των ασύρματων nodes

4.3.1.2 Σύνδεση των κόμβων

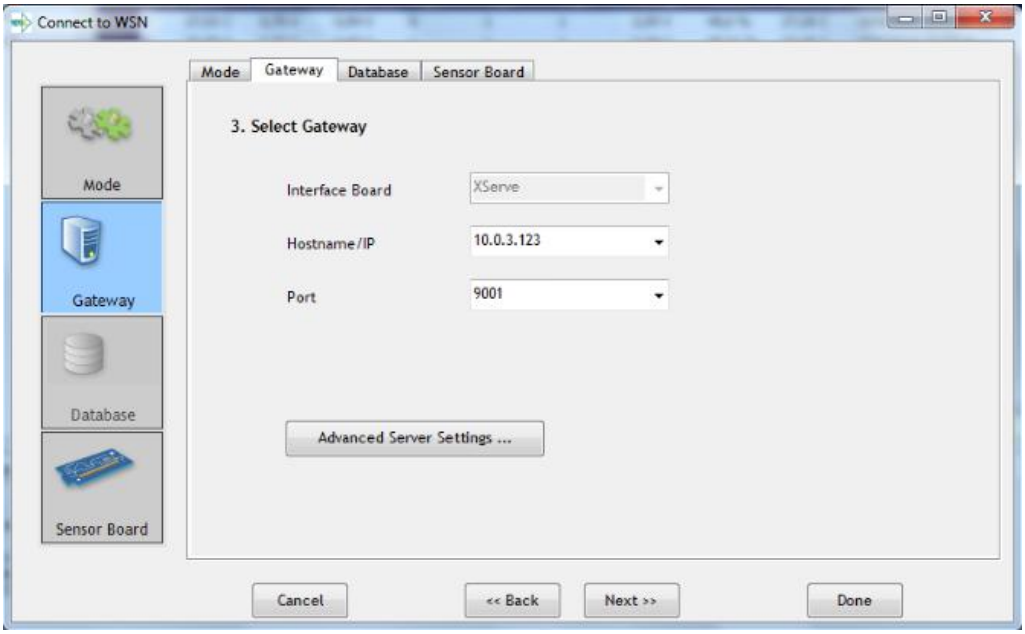
Αφού προγραμματίσουμε τους κόμβους του δικτύου χρησιμοποιούμε το MoteView για να ξεκινήσουμε την διαδικασία της λειτουργίας του δικτύου. Μέσω του MoteView συνδεόμαστε στη βάση δεδομένων που φορτώνονται τα δεδομένα από τους αισθητήρες και ρυθμίζουμε στους εξωτερικούς αισθητήρες. Οι εξωτερικοί αισθητήρες δίνουν αποτελέσματα σε Volt και μέσω διαφόρων τύπων μπορούμε να μετατρέψουμε το αποτέλεσμα στις μονάδες που θέλουμε (πχ. σε βαθμούς Κελσίου αν έχουμε αισθητήρα θερμοκρασίας).Ακόμα μας δίνει την δυνατότητα να φτιάξουμε την τοπολογία του δικτύου και να μας παρουσιάσει διαγραμματικά τα αποτελέσματα.

Αρχικά πατάμε το εικονίδιο Connect to WSN. Στην καρτέλα Mode επιλέγουμε Acquire Live Data στο Operation Mode και Remote στο Acquisition Type.



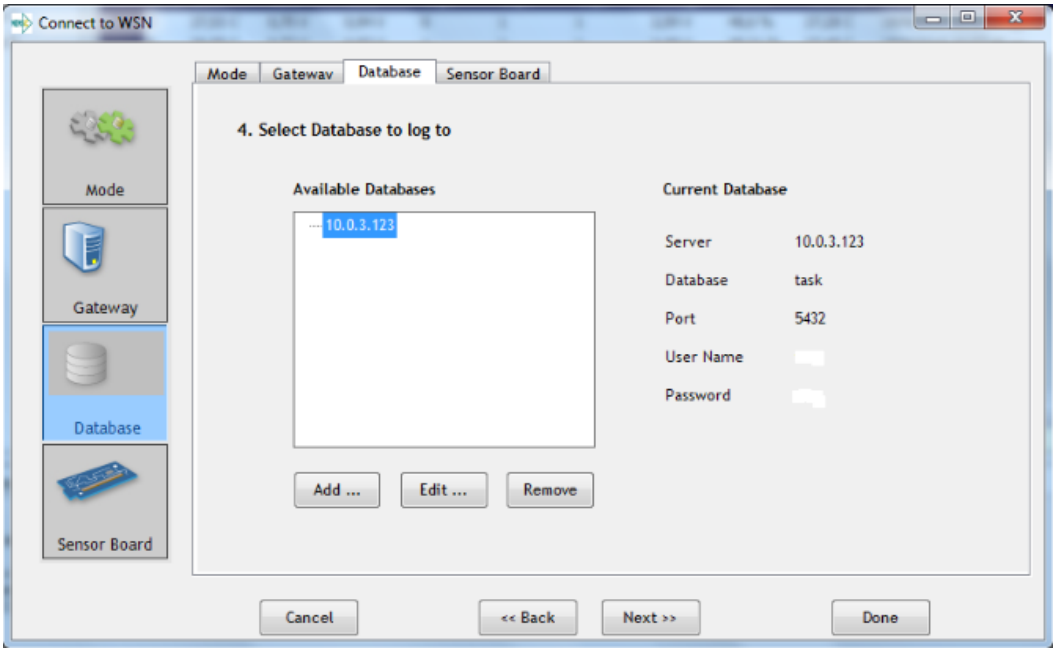
Εικόνα 33: Connect to WSN: Mode

Η επόμενη καρτέλα είναι η Gateway στην οποία εισάγουμε την IP του υπολογιστή στον οποίο βρίσκεται η βάση που θα αποθηκεύονται τα δεδομένα.



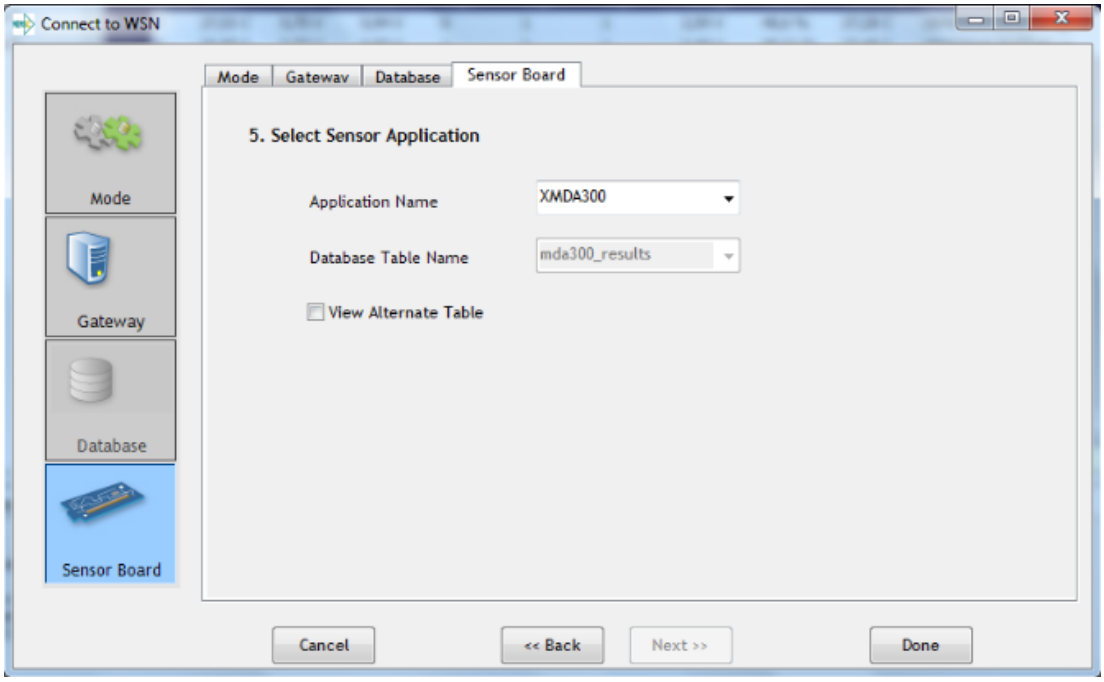
Εικόνα 34: Connect to WSN: Gateway

Στην καρτέλα Database βάζουμε τα στοιχεία της βάσης δεδομένων όπως username, password, database name κ.α.



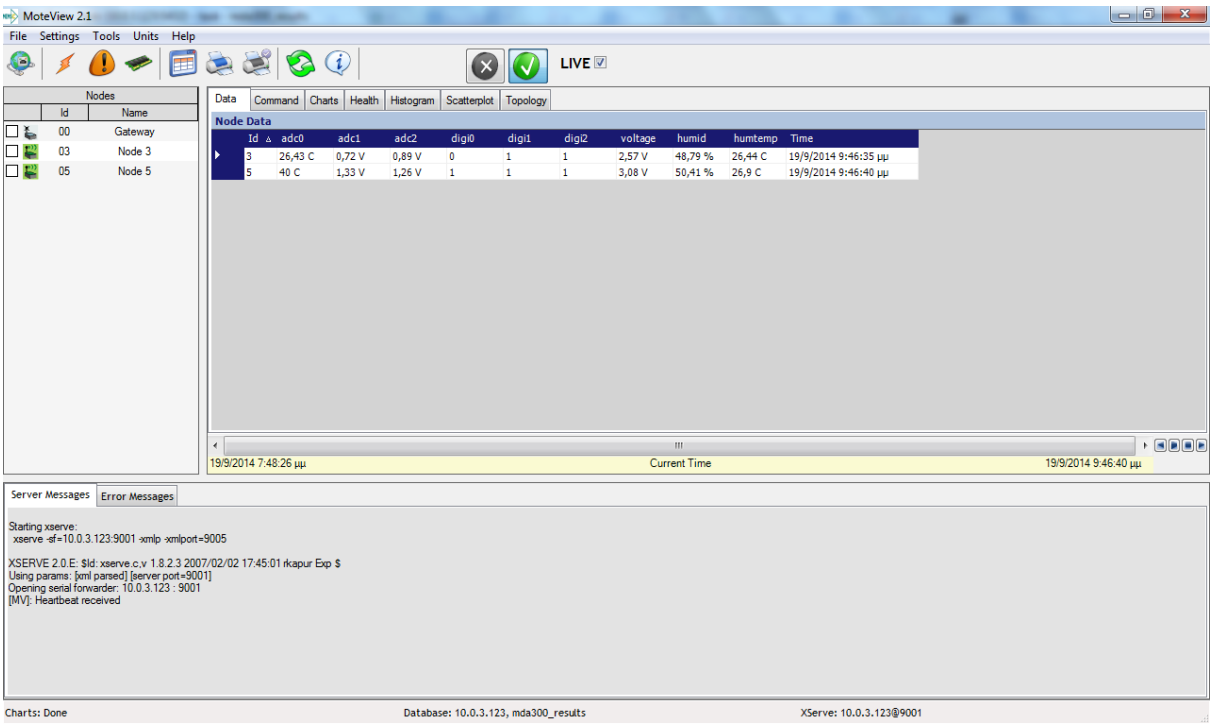
Εικόνα 35: Connect to WSN: Database

Και τέλος, στην καρτέλα Sensor Board επιλέγουμε το όνομα της εφαρμογής, στην περίπτωσής το XMDA300 και πατάμε Done.



Εικόνα 36: Connect to WSN: Sensor Board

Έπειτα από λίγο θα αρχίσουμε να βλέπουμε τα αποτελέσματα στο MoteView.

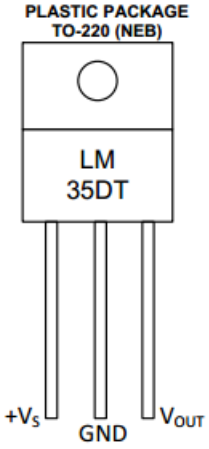


Εικόνα 37: Live Data

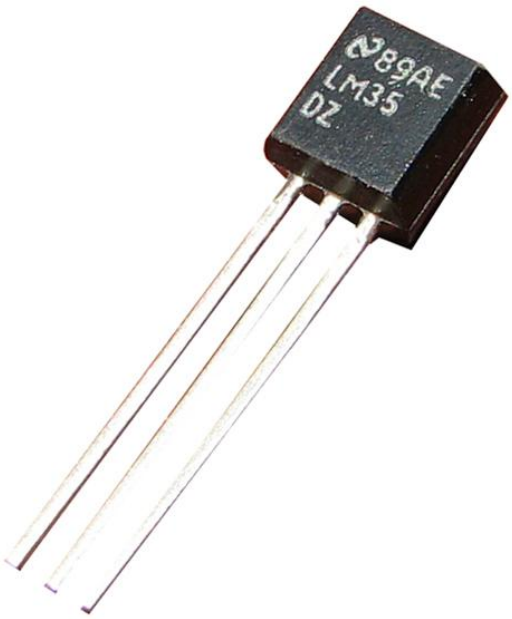
4.3.2 Εξωτερικοί Αισθητήρες

4.3.2.1 LM35

Ο LM35 είναι αισθητήρας θερμοκρασίας που δίνει αποτέλεσμα σε Volt. Χρειάζεται τροφοδοσία από 4 μέχρι 30 Volt για να λειτουργήσει και μπορεί να δώσει ένα εύρος τιμών από -55 έως 150 βαθμοί Κελσίου. Για να μετατρέψουμε τα Volts σε βαθμούς Κελσίου ισχύει ότι 10mV αντιστοιχούν σε 1 C. Κύριες εφαρμογές του βρίσκονται στη δημιουργία μετεωρολογικών σταθμών καθώς και σε αγροτικές καλλιέργειες. [25]



Εικόνα 38: Συνδεσμολογία του LM35



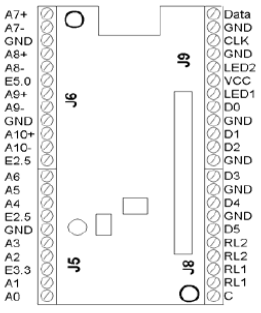
Εικόνα 39: LM35

4.3.2.2 DSN-FIR800

Ο DSN-FIR800 είναι ψηφιακός αισθητήρας κίνησης τύπου PIR (Passive Infrared). Τα αποτελέσματα του δίνονται με μορφή 0 ή 1. Χρειάζεται τροφοδοσία 3,3 ή 5 Volt και μπορεί να ανιχνεύσει κίνηση σε απόσταση 6 μέτρων. Λειτουργεί ανιχνεύοντας το υπέρυθρο φως των αντικειμένων που βρίσκονται στο οπτικό του πεδίο. Κύριες εφαρμογές του βρίσκονται σε συναγερμούς και σε συστήματα αυτόματου φωτισμού δωματίων ή εξωτερικών χώρων μόλις ανιχνεύσει κίνηση. [26][27]

Εικόνα 40: DSN-FIR800

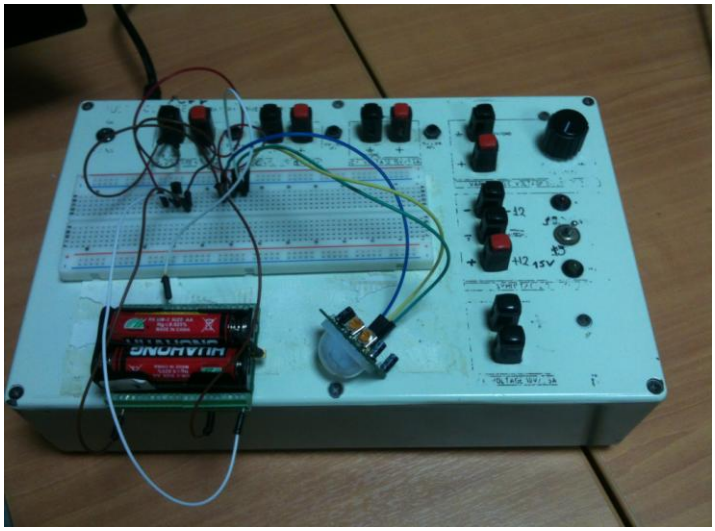
Για να εισάγουμε εξωτερικούς αισθητήρες στο MDA300 πρέπει να δούμε τα διαθέσιμα κανάλια της πλακέτας ανάλογα με τον τύπο του αισθητήρα που έχουμε, αν είναι δηλαδή αναλογικός ή ψηφιακός. Εμείς χρησιμοποιήσαμε δυο αισθητήρες, έναν αναλογικό θερμοκρασίας (LM35) και έναν ψηφιακό κίνησης (DSN-FIR800). Για το LM35 που είναι αναλογικός αισθητήρας χρησιμοποιήσαμε το κανάλι A0 για την πληροφορία.



A0 or A11+	Single-ended analog channel 0 **or** differential analog channel 11 positive side
A1 or A11-	Single-ended analog channel 1 **or** differential analog channel 11 negative side
A2 or A12+	Single-ended analog channel 2 **or** differential analog channel 12 positive side
A3 or A12-	Single-ended analog channel 3 **or** differential analog channel 12 negative side
A4 or A13+	Single-ended analog channel 4 **or** differential analog channel 13 positive side
A5 or A13-	Single-ended analog channel 5 **or** differential analog channel 13 negative side
A6	Single-ended analog channel 6
A7+ A7-	Differential analog channels 7
A8+ A8-	Differential analog channels 8
A9+ A9-	Differential analog channels 9
A10+ A10-	Differential analog channels 10
DATA	I2C Data
CLK	I2C Clock
D0 - D6	Digital Lines D0 to D6
C	Counter Channel
LED1	RED LED
LED2	GREEN LED
E5.0	5.0 V excitation
E3.3	3.3 V excitation
E2.5	2.5 V excitation
Vcc	Vcc of the Mote
RL1	Relay one sides (Normally-Open)
RL2	Relay two sides (Normally-Closed)

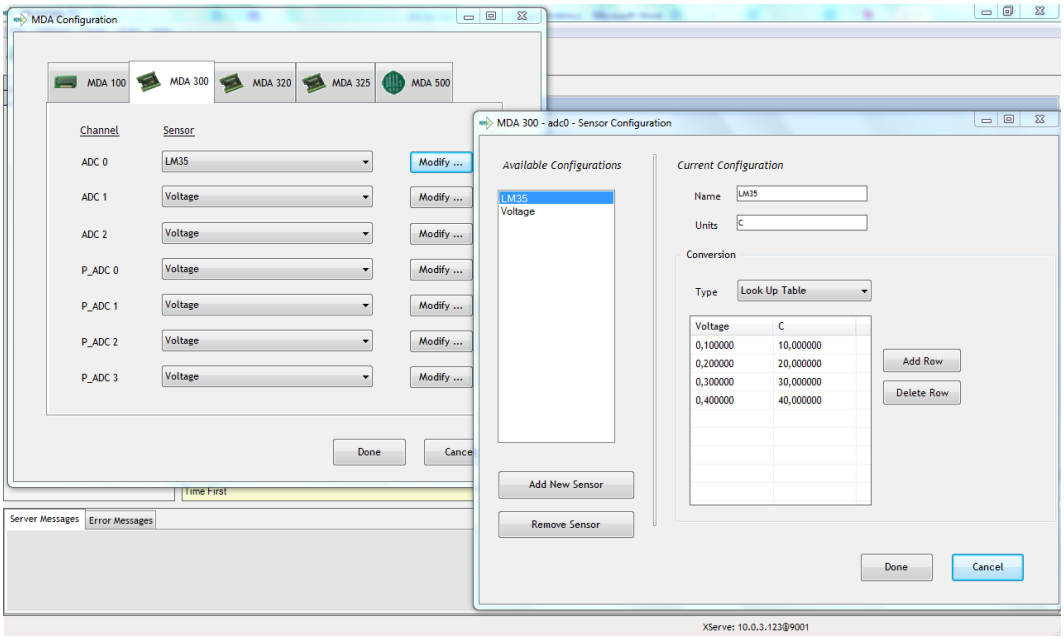
Εικόνα 41: Διάγραμμα του MDA300 με όλα τα διαθέσιμα κανάλια

Επειδή ο κόμβος του δικτύου τροφοδοτείται με 2 μπαταρίες 1,5 V δεν μπορεί να δώσει τροφοδοσία στον αισθητήρα μας ο οποίος χρειάζεται τουλάχιστον 4 V για να λειτουργήσει. Έτσι δώσαμε εξωτερική τροφοδοσία στον αισθητήρα 5V μέσω ενός breadboard. Ακόμα χρειάστηκε να δώσουμε 2 γειώσεις , μια εξωτερική και μια στο MDA300 στο κανάλι GND.



Εικόνα 42: Συνδεσμολογία

Μέσω του MoteView ρυθμίζουμε τις παραμέτρους για την μετατροπή του αποτελέσματος του LM35 από Volt σε βαθμούς Κελσίου. Πατάμε το εικονίδιο MDA Configuration από το μενού του MoteView και στο παράθυρο που εμφανίζεται πάμε στην καρτέλα MDA 300. Έχουμε συνδέσει τον LM35 στο κανάλι A0 οπότε επιλέγουμε Modify δίπλα από τον αντίστοιχο κανάλι για να κάνουμε τις ρυθμίσεις. Στο νέο παράθυρο που εμφανίζεται επιλέγουμε Add New Sensor, τον ονομάζουμε κατάλληλα και στο Units βάζουμε C (βαθμοί Κελσίου). Έπειτα στο Type επιλέγουμε το Look Up Table και εφόσον κάθε 0,1 Volt αντιστοιχεί σε 10 βαθμούς Κελσίου, βάζουμε στον πίνακα μερικές τιμές όπως φαίνεται στην εικόνα.

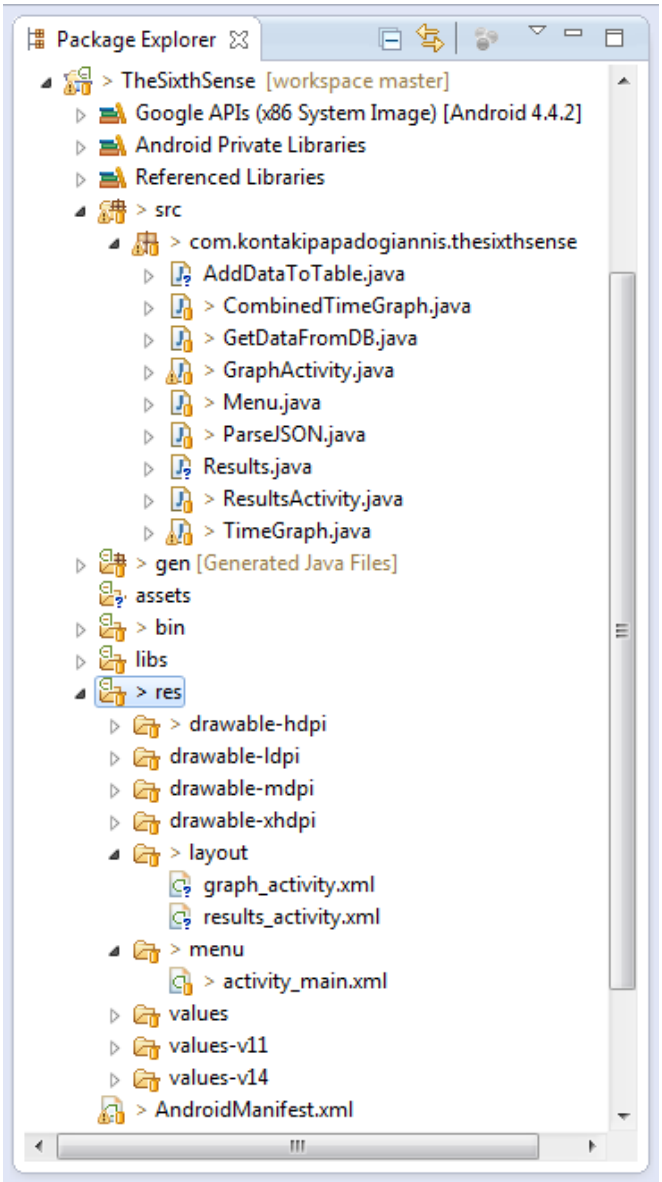


Εικόνα 43: Ρύθμιση του LM35

Για τον ψηφιακό αισθητήρα κίνησης ακολουθήθηκε η ίδια διαδικασία με τον αναλογικό όσον αφορά την γείωση και την τροφοδοσία. Αυτό που αλλάζει είναι το κανάλι πληροφορίας που δώσαμε στο mda300 που τώρα είναι το κανάλι D0. Ο αισθητήρας αυτός δεν χρειάζεται κάποια ρύθμιση. Τα αποτελέσματά του δίνονται σε 1 όταν έχουμε κίνηση και σε 0 όταν δεν έχουμε κίνηση.

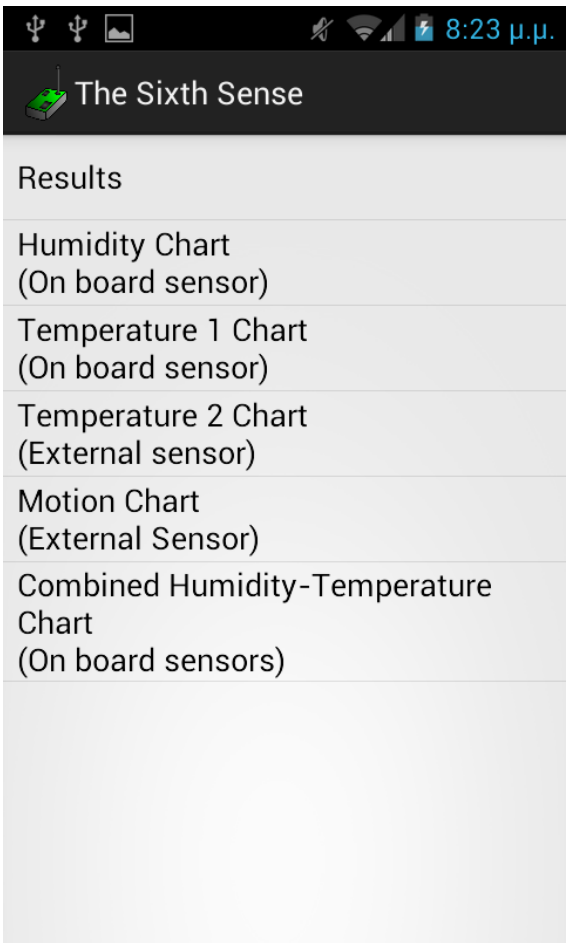
4.3.3 Εφαρμογή Android

Για την υλοποίηση της εφαρμογής Android χρησιμοποιήθηκε το Eclipse IDE. Ένα Android Application Project αποτελείται από διάφορα αρχεία και φακέλους. Περισσότερο χρήσιμα σε εμάς είναι ο φάκελος src στον οποίο βρίσκονται όλα τα αρχεία .java, δηλαδή ο κυρίως κώδικάς. Στον φάκελο res βρίσκονται όλα τα αρχεία .xml που έχουν να κάνουν κυρίως με το γραφικό κομμάτι της εφαρμογής.



Εικόνα 44: Οι φάκελοι και τα αρχεία ενός Android Project

Αρχικά η εφαρμογή έχει ένα μενού από το οποίο ο χρήστης μπορεί να διαλέξει σε τι μορφή καθώς και ποια δεδομένα θέλει να δει.



Εικόνα 45: Το μενού της εφαρμογής

Μόλις ο χρήστης επιλέξει κάτι από το μενού ξεκινά η διαδικασία της σύνδεσης με τη βάση δεδομένων, της διαχείρισης των JSON δεδομένων και της εμφάνισης σε πίνακα ή γράφημα ανάλογα με την επιλογή που έκανε ο χρήστης. Αυτές οι διαδικασίες περιγράφονται παρακάτω.

4.3.3.1 Σύνδεση με τη βάση δεδομένων

Η εφαρμογή χρειάζεται να συνδεθεί με τη βάση δεδομένων στη οποία στέλνονται τα δεδομένα των αισθητήρων έτσι ώστε να έχουμε τη δυνατότητα να τα δούμε. Η σύνδεση γίνεται μέσω ενός αρχείου php το οποίο είναι αποθηκευμένο στον server του εργαστηρίου. Τα δεδομένα κωδικοποιούνται σε μορφή JSON. Το Json (JavaScript Object Notation) είναι ένα πρότυπο κειμένου το οποίο μορφοποιεί τα δεδομένα σαν μία συλλογή από ζευγάρια ονομάτων-τιμών (Objects) ή σαν μία ταξινομημένη λίστα τιμών (Array). Εδώ κάθε εγγραφή του πίνακα γίνεται ένα JSON Object και όλα τα Objects δημιουργούν ένα Array [28]. Τα Objects είναι της μορφής:

{"result_time":"2014-07-09 13:48:55.092","humid":"759","humtemp":"7692","adc0":"2685","digi0":"1"}

Το αρχείο php έχει την παρακάτω μορφή:

```
<?php
$date = $_POST['result_time'];
$db connection = pg connect("host = localhost dbname = task user = tele password = tiny");
$q = pg query($db connection, "SELECT result time, humid, humtemp, adc0, digi0
                               FROM mda300 results
                               WHERE result_time >= '".$date"'
                               ORDER BY oid");
while($get_total_rows = pg_fetch_array($q)){
    $list[] = $get total rows;
}
json encode($list);
?>
```

Στην εφαρμογή τώρα, με την μέθοδο HttpPost στέλνουμε στον server που είναι αποθηκευμένο το αρχείο και μετά με τη χρήση του DefaultHttpClient στο οποίο περνάμε ως παραμέτρους ένα αντικείμενο τύπου HttpPost και ένα τύπου ResponseHandler, παίρνουμε το τελικό αποτέλεσμα σε μορφή String. Επίσης πρέπει να περάσουμε ως παράμετρο στο αρχείο php την ημερομηνία σύμφωνα με την οποία θα πάρει όλα τα αποτελέσματα που είναι πιο πρόσφατα από τη συγκεκριμένη ημερομηνία. Για να το κάνουμε αυτό αρχικά δημιουργούμε ένα αντικείμενο τύπου Calendar, το οποίο με τη μέθοδο getTime(), αφού έχουμε καθορίσει το κατάλληλο TimeZone για την περιοχή που βρισκόμαστε, μας δίνει την τωρινή ώρα. Όμως θέλουμε συγκεκριμένη ημερομηνία στο Calendar και όχι την τωρινή, οπότε με το `calendar.add(Calendar.DAY_OF_MONTH, -days);` και το `calendar.add(Calendar.HOUR_OF_DAY, -hours);` καθορίζουμε πόσες μέρες και ώρες θα αφαιρεθούν από την τωρινή ημερομηνία. Επίσης με την παράμετρο `nodeid` καθορίζουμε ποιού node αποτελέσματα θέλουμε να πάρουμε από τη βάση. Έπειτα περνάμε την ημερομηνία ως παράμετρο στο php αρχείο.

```
public class GetDataFromDB {

    public String getDataFromDB(int days, int hours, String nodeid) {
        try {
            HttpPost httppost = new HttpPost(
                    "http://bioklima5.prv3.teiher.gr/eleftheria/test2.php/");
            HttpClient httpclient = new DefaultHttpClient();

            SimpleDateFormat dateFormat = new SimpleDateFormat(
                    "yyyy-MM-dd HH:mm:ss");
            Calendar calendar = Calendar.getInstance();
            calendar.setTimeZone(TimeZone.getTimeZone("Europe/Athens"));
            dateFormat.setTimeZone(calendar.getTimeZone());
            calendar.add(Calendar.DAY_OF_MONTH, -days);
            calendar.add(Calendar.HOUR_OF_DAY, -hours);

            String result_time = dateFormat.format(calendar.getTime());

            List<BasicNameValuePair> nameValuePairs = new ArrayList<BasicNameValuePair>();
            nameValuePairs.add(new BasicNameValuePair("result_time",
                    result_time));
            nameValuePairs.add(new BasicNameValuePair("nodeid",
                    nodeid));
            httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));

            ResponseHandler<String> responseHandler = new BasicResponseHandler();
            final String response = httpclient.execute(httppost,
                    responseHandler);

            return response.trim();

        } catch (Exception e) {
            System.out.println("ERROR : " + e.getMessage());
            return "error";
        }
    }
}
```

4.3.3.2 *Διαχείριση των δεδομένων*

Εφόσον τα δεδομένα που παίρνουμε από τη βάση είναι σε μορφή JSON πρέπει να τα διαχειριστούμε κατάλληλα. Η κλάση ParseJSON παίρνει τα δεδομένα ως ένα String και δημιουργεί με αυτά ένα JSONArray. Έπειτα διατρέχουμε το JSONArray και παίρνουμε τα values από κάθε JSONObject τα οποία βάζουμε σε ArrayLists. Έτσι έχουμε μία ArrayList για κάθε τύπο δεδομένων. Επίσης επειδή τα δεδομένα θερμοκρασίας και υγρασίας τα παίρνουμε από τη βάση ως raw units (bytes) πρέπει να τα μετατρέψουμε με τους κατάλληλους τύπους σε engineering units. Η μέθοδος round() στρογγυλοποιεί τα δεδομένα κρατώντας μόνο δύο δεκαδικά ψηφία

```
public class ParseJSON {
    ArrayList<Date> dates = new ArrayList<Date>();
    ArrayList<Double> humidity = new ArrayList<Double>();
    ArrayList<Double> temperature = new ArrayList<Double>();
    ArrayList<Double> adc = new ArrayList<Double>();
    ArrayList<Double> digi = new ArrayList<Double>();

    public double round(double value, int places) {
        if (places < 0)
            throw new IllegalArgumentException();
        BigDecimal bd = new BigDecimal(value);
        bd = bd.setScale(places, RoundingMode.HALF_UP);
        return bd.doubleValue();
    }

    public void parse(String result) throws ParseException {
        try {
            JSONArray jArray = new JSONArray(result);
            for (int i = 0; i < jArray.length(); i++) {
                JSONObject json_data = jArray.getJSONObject(i);
                String result_time = json_data.getString("result_time");
                java.util.Date dateParser = new SimpleDateFormat(
                        "yy-MM-dd HH:mm:ss").parse(result_time);
                dates.add((Date) dateParser);
                if (json_data.getString("humid") != "null") {
                    Double humid = (Double.parseDouble(json_data.getString("humid")));
                    humidity.add(round(0.0405 * (humid) - 4 - humid * humid * 0.0000028, 2));
                } else {
                    humidity.add(0.0);
                }
                if (json_data.getString("humtemp") != "null") {
                    Double humtemp = Double.parseDouble(json_data.getString("humtemp"));
                    temperature.add(round((humtemp * 0.98 - 3840) / 100, 2));
                } else {
                    temperature.add(0.0);
                }
                if (json_data.getString("adc0") != "null") {
                    Double adc0 = Double.parseDouble(json_data.getString("adc0"));
                    adc.add(round(((adc0 * (5.0 / 1024)) * 100) / 8, 2));
                } else {
                    adc.add(0.0);
                }
                if (json_data.getString("digi0") != "null") {
                    Double digi0 = Double.parseDouble(json_data.getString("digi0"));
                    digi.add(digi0);
                } else {
                    digi.add(0.0);
                }
            }
        } catch (JSONException e) {
            Log.e("log_tag", "Error parsing data " + e.toString());
        }
    }
}
```

4.3.3.3 Εμφάνιση των δεδομένων σε μορφή πίνακα

Τα δεδομένα εμφανίζονται σε πίνακα ο οποίος δημιουργείται δυναμικά ανάλογα με τον αριθμό των δεδομένων που έχουμε. Τις επικεφαλίδες των στηλών του πίνακα, δηλαδή την πρώτη γραμμή αυτού, τις δημιουργούμε στατικά με ένα αρχείο xml:

```
<TableLayout
                android:id="@+id/maintable"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:stretchColumns="*" >
                <TableRow android:id="@+id/tableRow2" >
                    <TextView
                        android:id="@+id/result_time"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_margin="2dp"
                        android:background="@android:color/holo_red_light"
                        android:text="Result Time\n"
                        android:textColor="@android:color/black" />
                    <TextView
                        android:id="@+id/humidity"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_margin="2dp"
                        android:background="@android:color/holo_red_light"
                        android:text="Humid%\n(Internal)"
                        android:textColor="@android:color/black" />
                    <TextView
                        android:id="@+id/temperature"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_margin="2dp"
                        android:background="@android:color/holo_red_light"
                        android:text="Temp1\n(Internal)"
                        android:textColor="@android:color/black" />
                    <TextView
                        android:id="@+id/adc0"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_margin="2dp"
                        android:background="@android:color/holo_red_light"
                        android:text="Temp2\n(External)"
                        android:textColor="@android:color/black" />
                    <TextView
                        android:id="@+id/digi0"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_margin="2dp"
                        android:background="@android:color/holo_red_light"
                        android:text="Motion\n(External)"
                        android:textColor="@android:color/black" />
                </TableRow>
            </TableLayout>
```

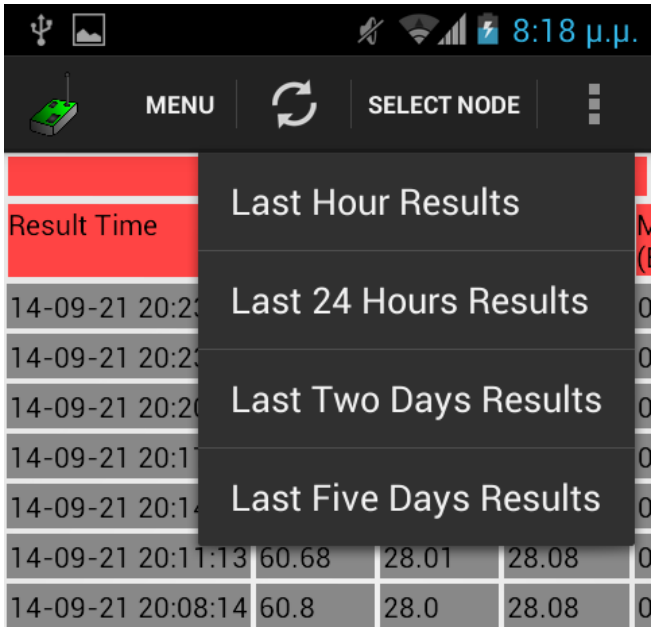
Η μέθοδος addData της κλάσης AddDataToTable παίρνει ως όρισμα ένα αντικείμενο της κλάσης Results, ένα αντικείμενο τύπου Context και το TableLayout που έχουμε ήδη φτιάξει στο xml αρχείο. Η κλάση Results περιέχει όλες τις ArrayLists με τα δεδομένα που δημιουργήθηκαν στην κλάση ParseJSON. Με μία for διατρέχουμε τις ArrayLists, παίρνουμε τα δεδομένα από κάθε ArrayList και τα βάζουμε σε TextViews. Αυτά τα TextViews δημιουργούν κάθε φορά μία γραμμή του πίνακα τα οποία προσθέτουμε στο TableLayout.

```
public class AddDataToTable {

    @SuppressLint("InlinedApi")
    public void addData(Results results, Context context, TableLayout tl) {
        TableRow tr;
        SimpleDateFormat df = new SimpleDateFormat("yy-MM-dd HH:mm:ss");
        TextView[] textView = new TextView[5];

        for (int i = 0; i < results.date.size(); i++) {
            tr = new TableRow(context);
            LinearLayout.LayoutParams params;

            textView[0] = new TextView(context);
            textView[0].setText(df.format(results.date.get(i)));
            textView[0].setLayoutParams(new LayoutParams(
                    LayoutParams.WRAP_CONTENT, LayoutParams.MATCH_PARENT));
            textView[0].setPadding(2, 2, 2, 2);
            textView[0].setBackgroundColor(Color.GRAY);
            textView[0].setTextColor(Color.BLACK);
            params = new TableRow.LayoutParams(LayoutParams.MATCH_PARENT,
                    LayoutParams.WRAP_CONTENT);
            params.setMargins(2, 2, 2, 2);
            tr.addView(textView[0], params);

            textView[1] = new TextView(context);
            textView[1].setText(Double.toString(results.humidity.get(i)));
            textView[1].setLayoutParams(new LayoutParams(
                    LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT));
            textView[1].setPadding(2, 2, 2, 2);
            textView[1].setBackgroundColor(Color.GRAY);
            textView[1].setTextColor(Color.BLACK);
            params = new TableRow.LayoutParams(LayoutParams.MATCH_PARENT,
                    LayoutParams.WRAP_CONTENT);
            params.setMargins(2, 2, 2, 2);
            tr.addView(textView[1], params);

            textView[2] = new TextView(context);
            textView[2].setText(Double.toString(results.temperature.get(i)));
            textView[2].setLayoutParams(new LayoutParams(
                    LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT));
            textView[2].setPadding(2, 2, 2, 2);
            textView[2].setBackgroundColor(Color.GRAY);
            textView[2].setTextColor(Color.BLACK);
            params = new TableRow.LayoutParams(LayoutParams.MATCH_PARENT,
                    LayoutParams.WRAP_CONTENT);
            params.setMargins(2, 2, 2, 2);
            tr.addView(textView[2], params);

            textView[3] = new TextView(context);
            textView[3].setText(Double.toString(results.adc0.get(i)));
            textView[3].setLayoutParams(new LayoutParams(
                    LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT));
            textView[3].setPadding(2, 2, 2, 2);
            textView[3].setBackgroundColor(Color.GRAY);
            textView[3].setTextColor(Color.BLACK);
            params = new TableRow.LayoutParams(LayoutParams.MATCH_PARENT,
                    LayoutParams.WRAP_CONTENT);
            params.setMargins(2, 2, 2, 2);
            tr.addView(textView[3], params);

            textView[4] = new TextView(context);
            textView[4].setText(Double.toString(results.digi0.get(i)));
            textView[4].setLayoutParams(new LayoutParams(
                    LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT));
            textView[4].setPadding(2, 2, 2, 2);
            textView[4].setBackgroundColor(Color.GRAY);
            textView[4].setTextColor(Color.BLACK);
            params = new TableRow.LayoutParams(LayoutParams.MATCH_PARENT,
                    LayoutParams.WRAP_CONTENT);
            params.setMargins(2, 2, 2, 2);
            tr.addView(textView[4], params);
```

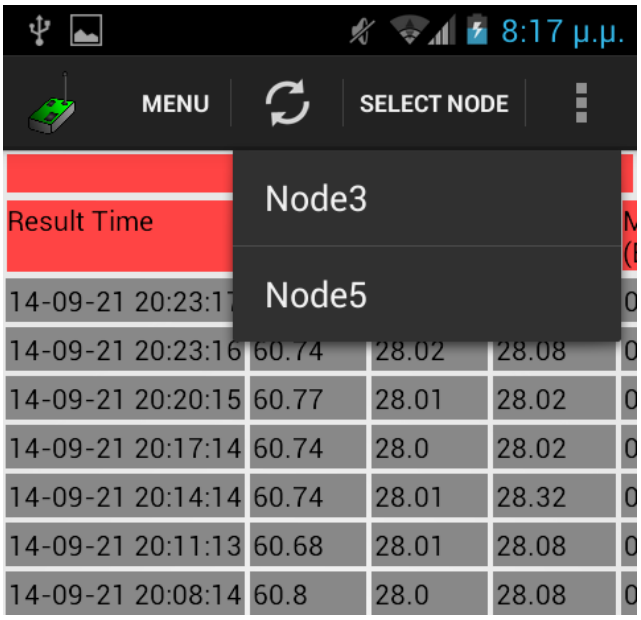
```
            tl.addView(tr, new TableLayout.LayoutParams(
                    LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT));
        }
    }
}
```

Τα δεδομένα που βλέπουμε αρχικά, μόλις δηλαδή επιλέξουμε το Results από το μενού της εφαρμογής, είναι της τελευταίας ώρας, έχουμε όμως τη δυνατότητα να επιλέξουμε δεδομένα των τελευταίων 24 ωρών, 2 και 5 ημερών. Επίσης μπορούμε να επιλέξουμε τα δεδομένα ποιού node θέλουμε να δούμε.

Εικόνα 46: Δεδομένα σε πίνακα



Εικόνα 47: Επιλογή ημερομηνίας για τα δεδομένα στον πίνακα



Εικόνα 48: Επιλογή node για τα δεδομένα στον πίνακα

4.3.3.4 Εμφάνιση δεδομένων σε γραφήματα

Για τη δημιουργία των γραφημάτων χρησιμοποιήσαμε την έτοιμη βιβλιοθήκη AChartEngine. Η κλάση που δημιουργεί τα γραφήματα είναι η TimeGraph. Στη μέθοδο getView περνάμε κάποια ορίσματα μεταξύ αυτών και τα δεδομένα που θέλουμε να έχουμε στον X και στον Y άξονα. Όλα τα γραφήματα στον X άξονα έχουν ημερομηνίες, δηλαδή την ArrayList dates η οποία και τις περιέχει. Χρησιμοποιούμε το TimeChart που υποστηρίζει η AChartEngine για να έχουμε μορφοποιημένες ημερομηνίες στον X άξονα.

```
public class TimeGraph {

    public GraphicalView getView(Context context, ArrayList<Date> dates,
            ArrayList<Double> data, String seriesTitle, String chartTitle,
            int yAxisMax, int density) {

        TimeSeries series = new TimeSeries(seriesTitle);
        int i;

        for (i = 0; i < dates.size(); i++) {

            series.add(dates.get(i), data.get(i));
        }
        XYMultipleSeriesDataset mDataset = new XYMultipleSeriesDataset();

        XYSeriesRenderer renderer = new XYSeriesRenderer();
        XYMultipleSeriesRenderer mRenderer = new XYMultipleSeriesRenderer();

        mDataset.addSeries(series);
        renderer.setColor(Color.RED);
        renderer.setPointStyle(PointStyle.CIRCLE);
        renderer.setFillPoints(true);
        renderer.setShowLegendItem(true);
        mRenderer.setMargins(new int[] { density / 3, density / 5, density / 3, density / 4 });
        mRenderer.setLegendTextSize(density / 10);
        mRenderer.setShowLegend(true);
        mRenderer.setLabelsTextSize(density / 12);
```

```
        mRenderer.setFitLegend(true);
        mRenderer.setZoomEnabled(true);
        mRenderer.setPanEnabled(true);
        mRenderer.setChartTitle(chartTitle);
        mRenderer.setChartTitleTextSize(density / 8);
        mRenderer.setZoomButtonsVisible(true);
        mRenderer.setApplyBackgroundColor(true);
        mRenderer.setBackgroundColor(Color.BLACK);
        mRenderer.setMarginsColor(Color.BLACK);
        mRenderer.setYLabelsAlign(Align.RIGHT);
        mRenderer.setYLabelsPadding(density / 12);
        mRenderer.setYAxisMin(0);
        mRenderer.setYAxisMax(yAxisMax);
        mRenderer.setShowGrid(true);
        mRenderer.setGridColor(Color.WHITE);
        mRenderer.addSeriesRenderer(renderer);

        GraphicalView gView = ChartFactory.getTimeChartView(context, mDataset,
                mRenderer, "yy-MM-dd\nHH:mm:ss");

        return gView;
    }
}
```

Το συνδυασμένο γράφημα υγρασίας θερμοκρασίας δημιουργείται από την κλάση CombinedGraphActivity που διαφέρει μόνο στο ότι αντί να έχουμε μία σειρά (TimeSeries) με δεδομένα, έχουμε δύο, μία για την υγρασία και μία για τη θερμοκρασία.

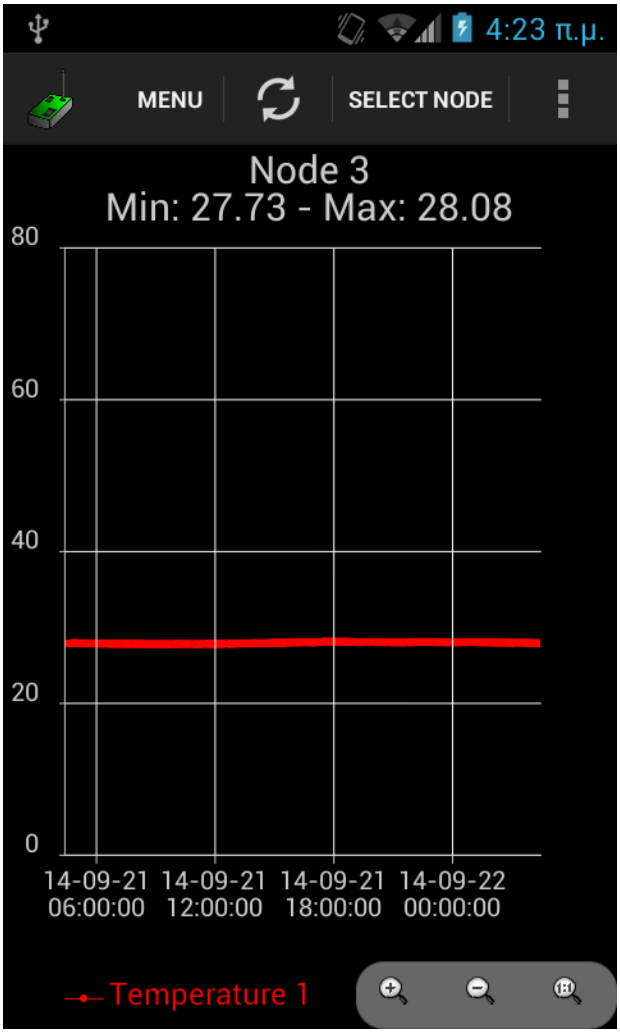
Η GraphicalView καλείται από την GraphActivity κάθε φορά που θέλουμε να δούμε κάποιο γράφημα. Αναλόγως ποιο γράφημα επιλέγουμε κάθε φορά από το μενού της εφαρμογής, καλείται η GraphicalView με τα κατάλληλα ορίσματα.

```
TimeGraph graph = new TimeGraph();

        if (whichGraph.equals("Humidity Chart\n(On board sensor)")) {
                gView = graph.getView(getApplicationContext(),date, humidity, "Humidity", "Node "
                + nodeid + "\n"
                + sortArrayList(humidity), 100,density);

        } else if (whichGraph.equals("Temperature 1 Chart\n(On board sensor)")) {
                gView = graph.getView(getApplicationContext(),date, temperature, "Temperature 1",
                "Node "
                + nodeid + "\n"
                + sortArrayList(temperature), 80,density);

        } else if (whichGraph.equals("Temperature 2 Chart\n(External sensor)")) {
                gView = graph.getView(getApplicationContext(), date,adc0, "Temperature 2", "Node "
                + nodeid + "\n"
                + sortArrayList(adc0), 100,density);

        } else if (whichGraph.equals("Motion Chart\n(External Sensor)")) {
                gView = graph.getView(getApplicationContext(),date, digi0, "Motion", "Node "
                + nodeid
                + "\n" + sortArrayList(digi0), 1,density);

        } else if (whichGraph.equals("Combined Humidity-Temperature Chart\n(On board sensors)")) {
                CombinedTimeGraph combinedGraph = new CombinedTimeGraph();
                gView = combinedGraph.getView(getApplicationContext(), date, humidity,temperature,
                "Humidity", "Temperature","Node "
                + nodeid + "\n"
                + "Humidity " + sortArrayList(humidity) + "\n"
                + "Temperature " + sortArrayList(temperature),density);
        }
```

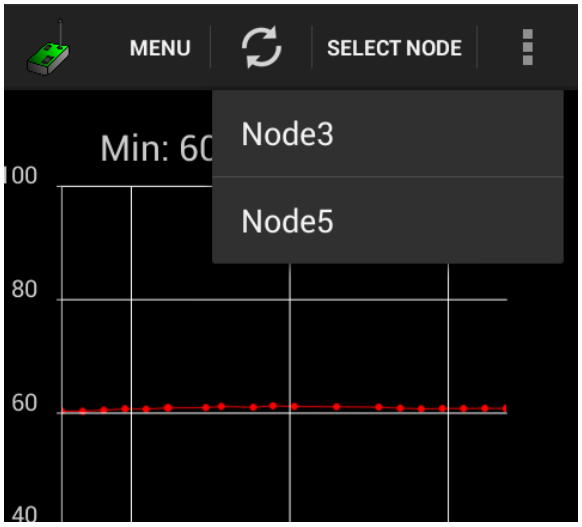
Η μέθοδος sortArrayList ταξινομεί τα δεδομένα με αύξουσα σειρά και βρίσκει τη μέγιστη και την ελάχιστη τιμή κάθε φορά η οποία εμφανίζεται πάνω από τα γραφήματα.

```
public String sortArrayList(ArrayList<Double> list) {

        ArrayList<Double> sortedArrayList = new ArrayList<Double>(list);
        Collections.sort(sortedArrayList);
        String humidityMinMax = "Min: " + (sortedArrayList.get(0)).toString()
                + " - Max: "
                + (sortedArrayList.get(sortedArrayList.size() - 1)).toString();

        return humidityMinMax;
}
```

Όπως και στον πίνακα, έτσι και στα γραφήματα μας εμφανίζονται αρχικά τα δεδομένα της τελευταίας ώρας αλλά έχουμε πάλι τη δυνατότητα να επιλέξουμε να δούμε περισσότερα και επίσης να διαλέξουμε node:



Εικόνα 49: Γράφημα θερμοκρασίας



Εικόνα 50: Επιλογή node για το γράφημα

Εικόνα 51: Επιλογή ημερομηνίας για το γράφημα

5 Αποτελέσματα - Συμπεράσματα

Με την ολοκλήρωση της πτυχιακής βλέπουμε ότι μπορεί ένας χρήστης να παρακολουθεί από παντού στο κινητό του τηλέφωνο τα δεδομένα που στέλνουν οι ασύρματοι αισθητήρες, κάτι που προσφέρει μεγάλη ευκολία.

Κατά την εκπόνηση της πτυχιακής προέκυψαν κάποια προβλήματα τα οποία οφείλονταν κυρίως στις ελλιπείς μας γνώσεις γύρω από τα θέματα που πραγματεύεται η πτυχιακή. Τα προβλήματα αυτά όμως ξεπεράστηκαν με πολύ έρευνα στο διαδίκτυο γύρω από αυτά τα θέματα και αυτό είχε ως αποτέλεσμα να ολοκληρώσουμε την εργασία και να αποκτήσουμε περισσότερες γνώσεις, κάτι που είναι και ο σκοπός κάθε πτυχιακής.

5.1 Μελλοντική εργασία και επεκτάσεις

Το δίκτυο των αισθητήρων θα μπορούσε να επεκταθεί με περισσότερα ασύρματα nodes έτσι ώστε να μπορεί να παρακολουθείτε μεγαλύτερη έκταση. Επίσης, θα μπορούσαν να προστεθούν περισσότεροι εξωτερικοί αισθητήρες έτσι ώστε η εργασία να έχει εφαρμογή π.χ. στην Τηλεϊατρική ή σε συστήματα αυτοματισμού.

Βιβλιογραφία

[1] http://www.webvistas.org/topic/816-το-διαδίκτυο-των-πραγμάτων/

[2] http://www.webvistas.org/topic/1940-wireless-sensor-networks

[3] http://el.wikipedia.org/wiki/Ασύρματο_δίκτυο_αισθητήρων

[4]http://nemertes.lis.upatras.gr/jspui/bitstream/10889/2069/1/Eleni%20Antigoni_Papageorgakopoulou%20thesis.pdf

[5] http://www.awiatech.com/an-overview-of-wirelesshart%E2%80%99s-osi-layers/

[6] http://www.nts-at.com/certification/isa

[7] http://www.controlglobal.com/assets/12WPpdf/120904-emerson-wirelesshart-isa.pdf

[8] http://en.wikipedia.org/wiki/6LoWPAN

[9] http://projets-gmi.univ-avignon.fr/projets//proj1112/M1/p09/doc/6LoWPAN_overview.pdf

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

[11] http://brain.ee.auth.gr/dokuwiki/doku.php?id=zigbee:zigbee

[12] Operating Systems for Wireless Sensor Networks: A Survey
Muhammad Omer Farooq and Thomas Kunz
Department of Systems and Computer Engineering, Carleton University Ottawa, Canada

[13] http://en.wikipedia.org/wiki/TinyOS

[14] http://en.wikipedia.org/wiki/LiteOS

[15] http://en.wikipedia.org/wiki/RIOT_(operating_system)

[16] Xmesh Users Manual

[17] XServe Users Manual

[18] http://en.wikipedia.org/wiki/Android_(operating_system)

[19] http://en.wikipedia.org/wiki/Eclipse_(software)

[20] http://el.wikipedia.org/wiki/Java

[21] http://el.wikipedia.org/wiki/XML

[22] http://en.wikipedia.org/wiki/Crossbow_Technology

[23] http://www.memsic.com/about-memsic/news-room.cfm?nid=MEMSIC%20Completes%20Crossbow%20Technology%20Acquisition&newsyear=2010

[24] MICAZ datasheet

[25] http://www.ti.com/lit/ds/symlink/lm35.pdf

[26] http://en.wikipedia.org/wiki/Passive_infrared_sensor

[27] http://eecs.oregonstate.edu/education/docs/pir/PIRSensor-V1.2.pdf

[28] http://json.org/

[29] MoteView Users Manual

[30] MoteConfig Users Manual

[31] MTS-MDA Series Users Manual

[32] Andreas Vlissidis,Kostas Michail,Giorgos Gounaris. " Treatment of Meteorological data and real time presentation at TEI Crete Heraklion Internet", TEMU 2005, TEI Cret/Heraklion, 30 June 2005 Andreas Vlissidis. "« A simplified method for the prediction of the produced energy from Wind and Irradiation, and application of results in a Hybrid Energy System". international Conference for Sustainable Energy in Transilvania University of Brasov, 7 July 2005

[33] Giorgos Gounaris, Andreas Vlissidis, Kostas Michail. " Weather stations distributed model for data processing and on-line internet presentation" TEMU 2006, TEI Crete/Heraklion, 3July 2006

[34] Stavros Charakopoulos, Andreas Vlissidis, Denia Kolokotsa, Giorgos Vassilakis , Manos Makrygiannakis: "The development of a Wireless Intergrated Sensor Network for enviromentals oversight in small buildings", 1th International Conference on Image Processing and Communications,September 16-18,2009 Bydgoszcz,Poland, proceedings